IBM WebSphere Commerce

프로그래밍 안내서 및 학습서

버전 5.5

IBM WebSphere Commerce

프로그래밍 안내서 및 학습서

버전 5.5

주!

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 주의사항의 일반 정보를 읽으십시오.

초판(2003년 6월)

이 개정판은 새 개정판에서 별도로 명시하지 않는 한, IBM WebSphere Commerce Business Edition 버전 5.5, IBM WebSphere Commerce Professional Edition 버전 5.5 및 모든 후속 릴리스와 수정에 적용됩니다. 제품 레벨에 맞는 올바른 버전을 사용하고 있는지 확인하십시오.

현지의 IBM 담당자나 IBM 지사를 통해 서적을 주문하십시오. 다음 주소에서는 책을 구비하고 있지 않습니다.

IBM에서는 귀하의 의견을 환영합니다. 다음 URL에서 사용 가능한 온라인 IBM WebSphere Commerce 문서 피드백 양식을 사용하여 의견을 보낼 수 있습니다.

http://www..ibm.com/software/webservers/commerce/rcf.html

IBM에 정보를 보내는 경우, IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다

시작하기 전에

이 책은 2003년 5월 30일에 갱신되었습니다.

*WebSphere Commerce 프로그래밍 안내서 및 학습서*에서는 WebSphere® Commerce 아키텍처 및 프로그래밍 모델에 대해 설명합니다. 특히 다음에 대해 자세히 설명합니다.

- 구성요소 상호 작용
- 설계 패턴
- Persistent 오브젝트 모델
- 액세스 제어
- 오류 처리 및 메시지
- 명령 구현
- 개발 도구
- 사용자 정의 코드 전개

또한 이 책에는 다음 학습 항목이 있습니다.

- 새 비즈니스 로직 작성
- 기존 컨트롤러 명령 수정
- 오브젝트 모델 확장 및 기존 태스크 명령 수정
- 기존 WebSphere Commerce 엔티티 bean 확장

이 책의 갱신 사항

이 책의 사본 및 모든 갱신된 버전은 다음 WebSphere Commerce 웹 사이트의 기술 라이브러리 절에서 PDF 파일로 사용 가능합니다.

`http://www.ibm.com/software/commerce/library/`

추가 지원 정보는 WebSphere Commerce 지원 사이트를 참조하십시오.

`http://www.ibm.com/software/commerce/support/`

이 책의 갱신된 버전은 다음 웹 사이트에 있는 WebSphere Developer Domain의 WebSphere Commerce Zone에서 사용 가능합니다.

`http://www.ibm.com/websphere/developer/zones/commerce`

이 책에 사용된 규칙

이 책은 다음과 같은 규칙을 사용합니다.

굵은체는 필드, 아이콘 또는 메뉴 선택사항의 이름과 같은 GUI(Graphical User Interface) 제어 또는 명령을 표시합니다.

모노체는 디렉토리 경로 및 정확하게 입력해야 하는 텍스트의 예를 표시합니다.

*기울임꼴*은 사용자가 값을 대체해야 하는 변수 및 강조에 사용됩니다.

이 아이콘은 태스크 완료에 도움을 주는 추가 정보를 표시합니다.

Windows Windows® 2000용 WebSphere Commerce에 고유하게 적용되는 정보를 나타냅니다.

AIX AIX®용 WebSphere Commerce에 고유하게 적용되는 정보를 나타냅니다.

Solaris Solaris Operating Environment용 WebSphere Commerce Operating Environment software에 고유하게 적용되는 정보를 나타냅니다.

400 IBM® @server iSeries™ 400®용(이전에는 AS/400)®WebSphere Commerce에 고유하게 적용되는 정보를 나타냅니다.

DB2 DB2 Universal Database™에 고유하게 적용되는 정보를 나타냅니다.

Oracle Oracle에 고유하게 적용되는 정보를 나타냅니다.

지식 요구사항

이 책은 WebSphere Commerce 응용프로그램을 사용자 정의하는 방법을 이해해야 하는 상점 개발자를 위한 책입니다. 프로그램 확장을 수행 중인 상점 개발자는 다음과 같은 영역에 대해 알아야 합니다.

- Java™
- EnterpriseJavaBeans 구성요소 아키텍처
- JSP 기술
- HTML
- 데이터베이스 기술
- WebSphere Studio Application Developer

경로 변수

이 안내서에서는 다음 변수를 사용하여 디렉토리 경로를 표시합니다.

WC_installdir

WebSphere Commerce의 설치 디렉토리입니다. 다음은 여러 운영체제에서의 WebSphere Commerce의 기본 설치 디렉토리입니다.

- Windows C:\Program Files\WebSphere\CommerceServer55
- AIX /usr/WebSphere/CommerceServer55
- Solaris /opt/WebSphere/CommerceServer55
- Linux /opt/WebSphere/CommerceServer55
- 400 /QIBM/ProdData/CommerceServer55

WC_userdir

WebSphere Commerce에서 사용하고 사용자가 수정할 수 있거나 구성해야 하는 모든 데이터의 디렉토리입니다.

- 400 /QIBM/UserData/CommerceServer55

WAS_installdir

WebSphere Application Server의 설치 디렉토리입니다. 다음은 여러 운영체제에서 WebSphere Application Server의 기본 설치 디렉토리입니다.

- Windows C:\Program Files\WebSphere\AppServer
- AIX /usr/WebSphere/AppServer
- Solaris /opt/WebSphere/AppServer
- Linux /opt/WebSphere/AppServer
- 400 /QIBM/ProdData/WebAs5/Base

WAS_userdir

WebSphere Application Server에서 사용하고 사용자가 수정할 수 있거나 구성해야 하는 모든 데이터의 디렉토리입니다.

- 400 QIBM/UserData/WebAS5/Base/*WAS_instancename*

  및 *WAS_instance_name*은 WebSphere Commerce 인스턴스와 연관되는 WebSphere Application Server의 이름을 나타냅니다.

WCStudio_installdir

WebSphere Commerce Studio의 설치 디렉토리입니다. 다음은 기본 설치 디렉토리입니다.

C:\WebSphere\CommerceStudio55

추가 정보

WebSphere Commerce에 관련된 추가 정보는 다음 웹 사이트를 참조하십시오.

http://www.ibm.com/software/commerce/library/

목차

제 1 부 개념 및 아키텍처

제 1 장 개요

WebSphere Commerce 소프트웨어 구성요소

WebSphere Commerce Server가 작동하는 방법을 살펴보기 전에 WebSphere Commerce와 관련된 소프트웨어 구성요소의 개요를 살펴보는 것이 좋습니다. 다음 도표는 이러한 소프트웨어 제품의 간단한 개요입니다.

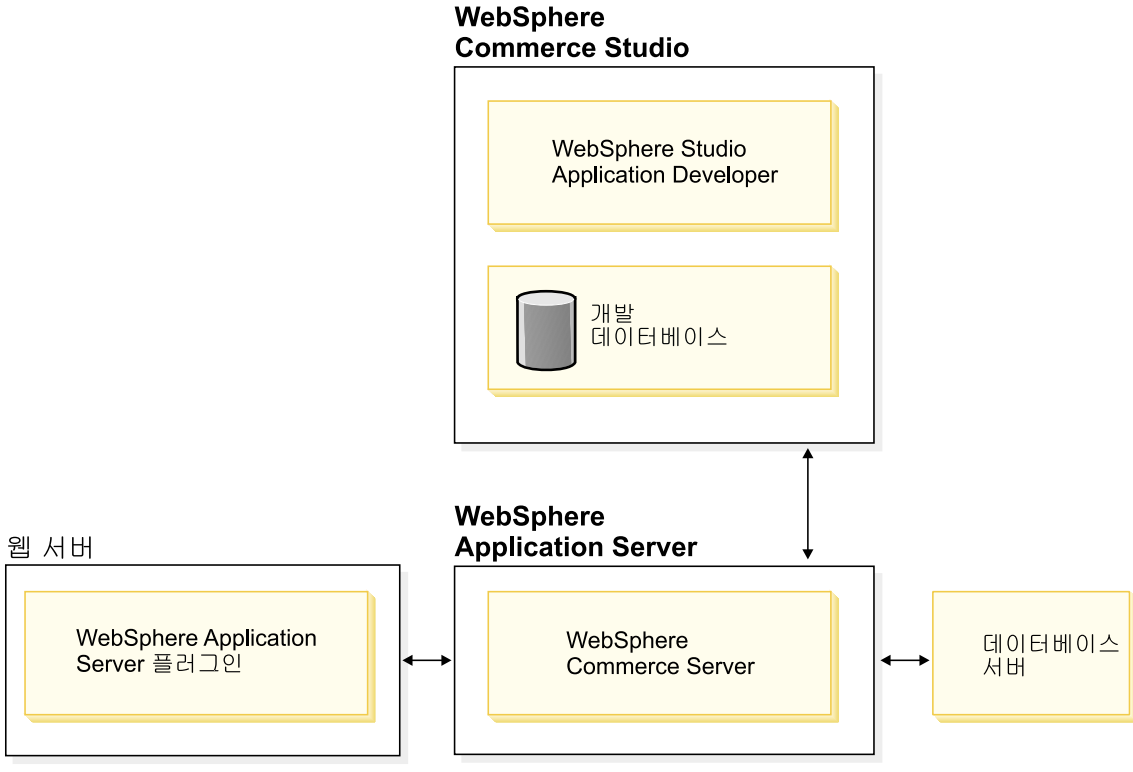


그림 1.

웹 서버는 전자 상거래 응용프로그램의 HTTP 수신 요청이 들어오는 첫 번째 지점입니다. WebSphere Application Server와 효과적으로 인터페이스하기 위해 WebSphere Application Server 플러그인을 사용합니다.

WebSphere Commerce Server는 WebSphere Application Server 내에서 실행되며 응용프로그램 서버의 여러 기능을 이용할 수 있습니다. 데이터베이스 서버는 상품 및 구매자 데이터를 포함한 대부분의 응용프로그램 데이터를 보관합니다. 일반적으로 WebSphere Commerce Server의 코드를 수정하거나 확장함으로써 응용프로그램을 확장할 수 있습니다. 또한 WebSphere Commerce 데이터베이스 스키마 영역 외부의 데이터를 데이터베이스에 저장할 수 있습니다.

개발자는 WebSphere Studio Application Developer를 사용하여 다음 태스크를 수행합니다.

- JSP 템플리트 및 HTML 페이지와 같은 상점 첫화면 자원 작성 및 사용자 정의
- Java로 새 비즈니스 로직 작성
- Java로 기존 비즈니스 로직 수정
- 코드 및 상점 첫화면 자원 테스트

WebSphere Commerce Studio는 개발 데이터베이스를 사용합니다. 개발자는 선호하는 데이터베이스 도구(WebSphere Studio Application Developer 포함)를 사용하여 데이터베이스를 수정합니다.

WebSphere Commerce 응용프로그램 아키텍처

WebSphere Commerce와 관련된 여러 가지 소프트웨어 구성요소들의 작동 방법을 살펴보았습니다. 이제 응용프로그램 아키텍처에 대해 살펴보고자 합니다. 이는 기초 계층이 되는 부분과 수정 가능한 부분을 이해하는 데 도움이 됩니다. 다음 도표는 응용프로그램 아키텍처를 형성하는 여러 계층을 보여줍니다.

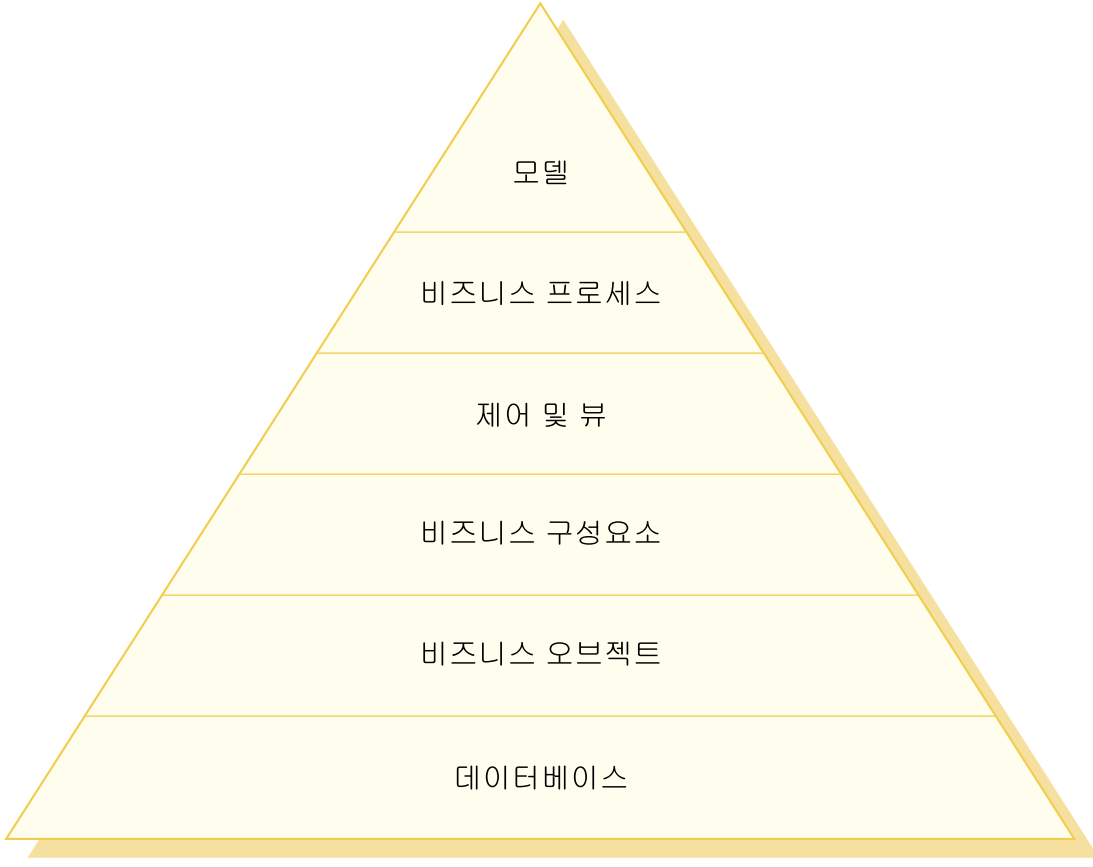


그림 2.

응용프로그램 아키텍처의 각 계층에 대한 설명은 다음과 같습니다.

데이터베이스

WebSphere Commerce는 전자 상거래 응용프로그램과 데이터 요구사항에 맞게 설계된 데이터베이스 스키마를 사용합니다. 다음은 이 스키마로 된 테이블 예입니다.

- USERS
- ORDERS
- INVENTORY

비즈니스 오브젝트

비즈니스 오브젝트는 상거래 도메인 내의 엔티티를 나타내고 데이터베이스 내에 들어 있는 정보를 가져오거나 해석하는 데 필요한 데이터 중심 로직을 캡슐화합니다. 이 엔티티는 Enterprise JavaBeans 스펙을 따릅니다.

이 엔티티 bean은 비즈니스 구성요소와 데이터베이스 사이의 인터페이스로 작동합니다. 또한 엔티티 bean은 각 데이터베이스의 테이블 열 간의 복잡한 관계를 이해하기 쉽게 해줍니다.

비즈니스 구성요소

비즈니스 구성요소는 비즈니스 로직의 단위입니다. 비즈니스 구성요소는 각 단위로 나뉘어진 프로시저 비즈니스 로직을 수행합니다. 로직은 컨트롤러 명령 및 태스크 명령의 WebSphere Commerce 모델을 사용하여 구현됩니다. 유형 구성요소의 예로는 OrderProcess 컨트롤러 명령이 있습니다. 이 명령은 일반 주문을 처리하는 데 필요한 모든 비즈니스 로직을 캡슐화합니다. 전자 상거래 응용프로그램은 OrderProcess 명령을 호출하고 이 명령은 차례대로 여러 태스크 명령을 호출하여 각 작업 단위를 수행합니다. 예를 들어, 각 태스크 명령은 주문 요구사항을 충족시킬 만큼 충분한 재고가 있는지 확인, 지불을 처리 및 주문 상태를 갱신하고 처리가 완료되면 적절한 분량만큼 재고를 감소시킵니다.

제어 및 뷰

웹 컨트롤러는 사용할 해당 컨트롤러 명령 구현 및 뷰를 판별합니다. 구현은 상점에 따라 다릅니다.

뷰는 명령 및 사용자 조치의 결과를 표시합니다. 이 뷰는 JSP 템플리트를 사용하여 구현됩니다. 뷰 예에는 ProductDisplayView(구매자가 선택한 상품 관련 정보를 표시하는 상품 페이지 리턴) 및 OrderCancelView가 포함됩니다.

비즈니스 프로세스

비즈니스 구성요소와 뷰가 한데 모여 비즈니스 프로세스라고 하는 워크플로우 및 사이트플로우를 작성합니다. 비즈니스 프로세스의 예는 다음과 같습니다.

전자 우편 캠페인 작성

전자 우편 캠페인 작성 프로세스에 포함되는 모든 단계와 관련된 비즈니스 구성요소 및 뷰가 포함됩니다.

온라인 카탈로그 준비

온라인 카탈로그 작성과 관련된 비즈니스 구성요소 및 서브프로세스가 포함됩니다. 이 프로세스에는 카탈로그 설계, 카탈로그 데이터 로드, 판매 계획 연관 작성 및 가격 책정 설정 정보가 포함됩니다.

모델 전체적으로 볼 때 도표의 하위 계층은 전자 상거래 비즈니스 모델을 구성합니다. 전자 상거래 비즈니스 모델의 한 예는 FashionFlow 견본 상점에 표시되는 직접형 B2C 모델입니다. 다른 예는 ToolTech 견본 상점에 표시되는 직접형 B2B 모델입니다.

WebSphere Commerce 런타임 아키텍처

이전 절에서는 응용프로그램 아키텍처를 소개하고 비즈니스 응용프로그램 관점에서 WebSphere Commerce 응용프로그램의 여러 계층에 대해 설명했습니다. 이 절에서는 런타임 아키텍처를 구현하는 방법에 대해 설명합니다.

WebSphere Commerce 런타임 아키텍처의 주요 구성요소는 다음과 같습니다.

- Servlet 엔진
- 프로토콜 리스너
- 어댑터 관리자
- 어댑터
- 웹 컨트롤러
- 명령
- 엔티티 bean
- 데이터 bean
- 데이터 bean 관리자
- 표시 페이지
- XML 파일

WebSphere Commerce 구성요소 간의 상호작용은 다음 도표에 표시됩니다. 다음 절에서는 각각의 구성요소에 대해 보다 자세히 설명합니다.

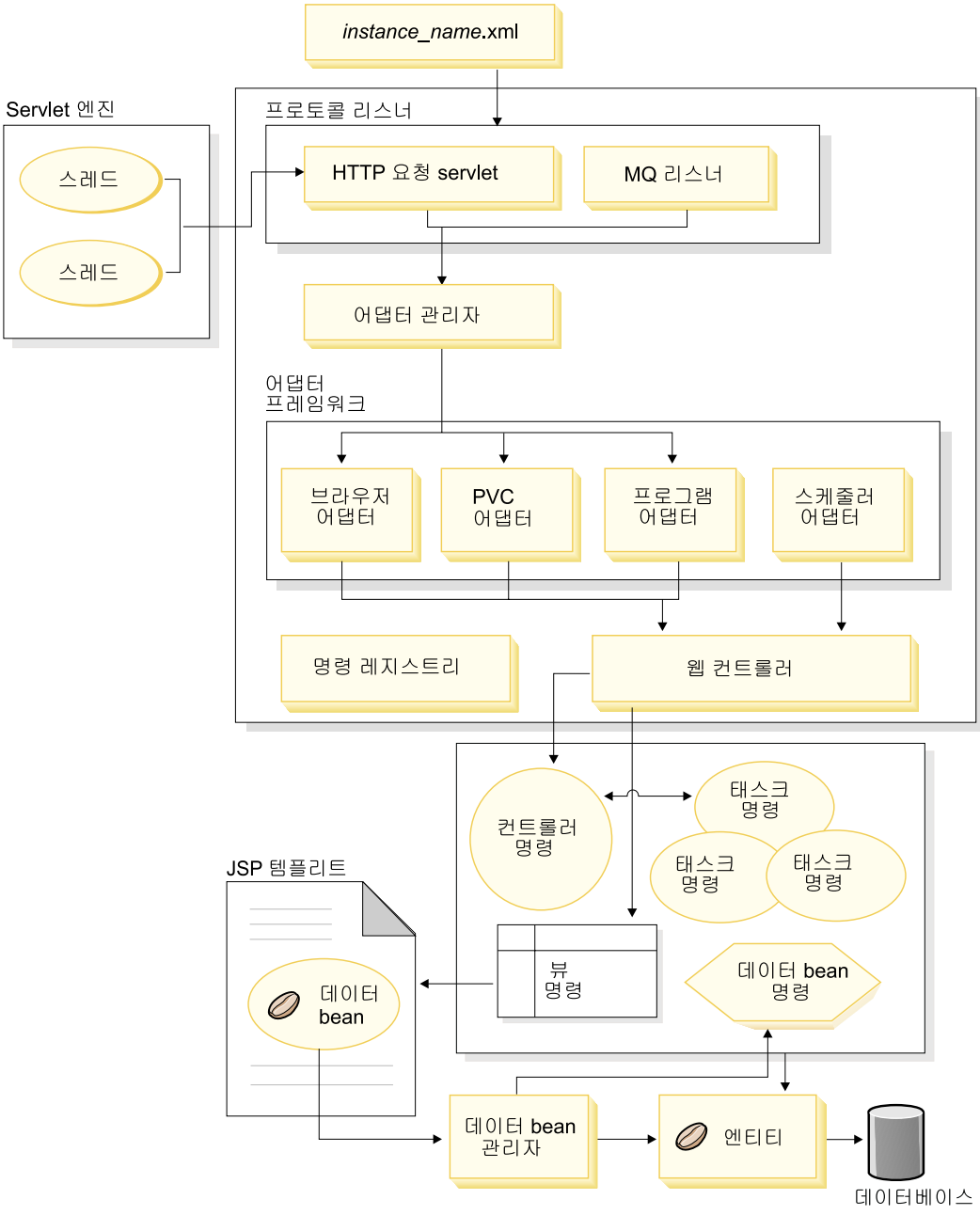

instance_name.xml
Servlet 엔진
스레드
스레드
프로토콜 리스너
HTTP 요청 servlet
MQ 리스너
어댑터 관리자
어댑터
프레임워크
브라우저
어댑터
PVC
어댑터
프로그램
어댑터
스케줄러
어댑터
명령 레지스트리
웹 컨트롤러
컨트롤러
명령
태스크
명령
태스크
명령
태스크
명령
JSP 템플리트
데이터
bean
뷰
명령
데이터 bean
명령
데이터 bean
관리자
엔티티
데이터베이스

그림 3.

Servlet 엔진

Servlet 엔진은 인바운드 URL 요청의 요청 디스패처 역할을 하는 WebSphere Application Server 런타임 환경의 일부입니다. Servlet 엔진은 요청을 처리하기 위해 스레드 풀을 관리합니다. 각 인바운드 요청은 개별 스레드로 실행됩니다.

프로토콜 리스너

WebSphere Commerce 명령은 다양한 장치에서 호출할 수 있습니다. 명령을 호출할 수 있는 장치의 예는 다음과 같습니다.

- 일반 인터넷 브라우저
- 인터넷 브라우저를 지원하는 이동 전화
- MQSeries®를 사용하여 XML 메시지를 보내는 B2B 응용프로그램
- HTTP에서 XML을 사용하여 요청을 전송하는 조달 시스템
- 백그라운드 작업을 실행하는 WebSphere Commerce 스케줄러

장치는 다양한 통신 프로토콜을 사용합니다. 프로토콜 리스너는 전송으로부터 인바운드 요청을 수신한 후 사용된 프로토콜에 따라 해당 어댑터로 요청을 지정하는 런타임 구성요소입니다. 프로토콜 리스너에는 다음이 포함됩니다.

- 요청 Servlet
- MQSeries 리스너

요청 Servlet이 Servlet 엔진에서 URL 요청을 수신하면 어댑터 관리자에게 요청을 전달합니다. 그러면 어댑터 관리자는 요청을 처리할 수 있는 어댑터를 결정하기 위해 어댑터 유형을 조회합니다. 일단 특정 어댑터가 결정되면 요청은 해당 어댑터로 전달됩니다.

요청 Servlet이 초기화되면 *instance_name*.xml 구성 파일을 읽습니다. 여기서, *instance_name*은 WebSphere Commerce 인스턴스 이름입니다. XML 파일의 구성 블록 중 하나에 모든 어댑터가 정의됩니다. 요청 Servlet의 init() 메소드는 정의된 모든 어댑터를 초기화합니다.

MQSeries 리스너는 원격 프로그램에서 XML 기반 MQSeries 메시지를 받아 비 HTTP 어댑터 관리자로 메시지를 지정합니다.

작업 스케줄러에는 프로토콜 리스너가 필요하지 않습니다.

어댑터 관리자

어댑터 관리자는 어떤 어댑터가 요청을 처리할 것인지를 결정한 후 요청을 해당 어댑터로 전달할 수 있는지 판별합니다.

어댑터

WebSphere Commerce 어댑터는 웹 컨트롤러에 요청을 전달하기 전에 처리를 수행하는 장치 고유의 구성요소입니다. 다음은 어댑터에서 수행되는 처리 태스크의 예입니다.

- 웹 컨트롤러에 장치 유형에 맞는 방식으로 요청을 처리하도록 지시. 예를 들어, PvC(Pervasive Computing) 장치 어댑터는 웹 컨트롤러에서 요청한 HTTPS 검사를 무시하도록 지시할 수 있습니다.
- WebSphere Commerce 명령이 구문 분석하는 특성 세트로 인바운드 요청의 메시지 포맷을 전달
- 장치 고유의 세션 지속성 제공

다음 도표는 WebSphere Commerce 어댑터 프레임워크의 구현 클래스 계층을 보여줍니다

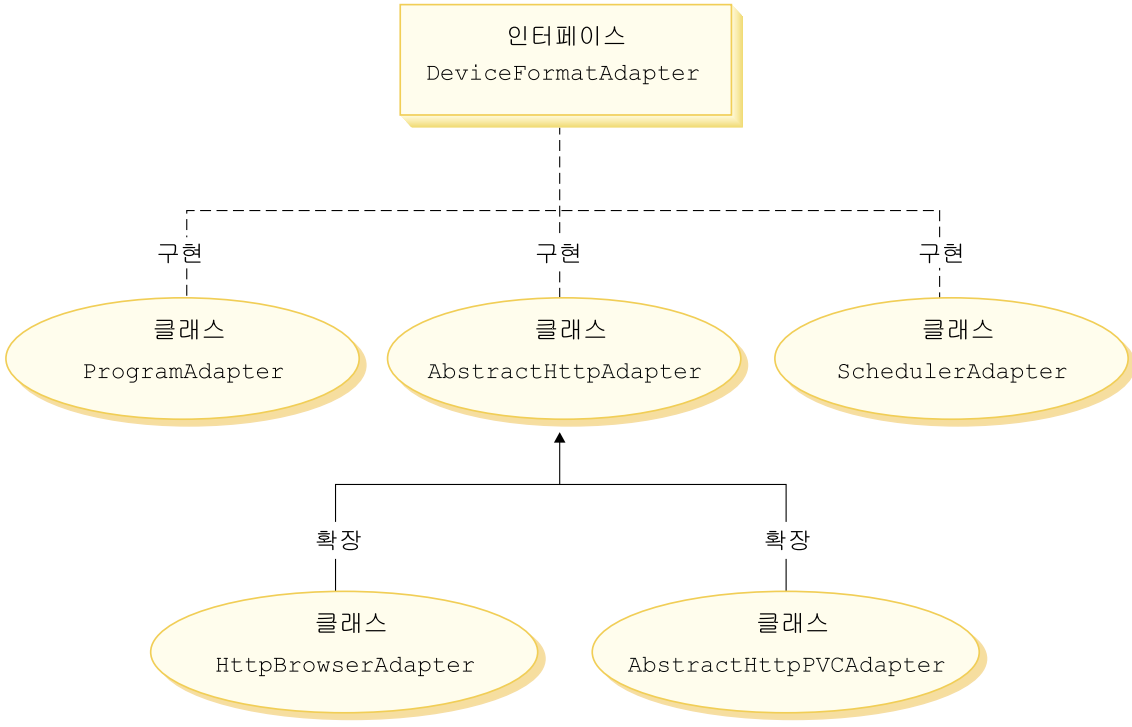


그림 4.

위의 도표에 표시된 대로 모든 어댑터는 DeviceFormatAdapter 인터페이스를 구현합니다. 다음은 WebSphere Commerce 런타임 환경에서 사용되는 어댑터입니다.

프로그램 어댑터

프로그램 어댑터는 WebSphere Commerce 명령을 호출하여 원격 프로그램을 지원합니다. 프로그램 어댑터는 요청을 받고 메시지 맵퍼를 사용하여 요청을 CommandProperty 오브젝트로 변환합니다. 변환 후에, 프로그램 어댑터는 CommandProperty 오브젝트를 사용하고 요청을 실행합니다.

스케줄러 어댑터

스케줄러 어댑터는 백그라운드 작업으로 실행되는 WebSphere Commerce 명령을 지원합니다

HTTP 브라우저 어댑터

HTTP 브라우저 어댑터는 HTTP 브라우저에서 수신되는 WebSphere Commerce 명령을 호출하는 요청을 지원합니다.

HTTP PvC 어댑터

특정 PvC 장치 어댑터를 개발하는 데 사용할 수 있는 추상 어댑터 클래스입니다. 예를 들어, 특정 휴대폰 응용프로그램용 어댑터를 개발해야 하는 경우, 이 어댑터에서 확장합니다.

필요한 경우, 어댑터 프레임워크는 다음의 두 가지 방법으로 확장될 수 있습니다.

- 특정 PvC 장치용 어댑터를 작성하십시오(예를 들어, i 모드 장치를 지원하는 HttpIModePVCAdapterImpl 클래스를 작성하십시오). 이러한 유형의 어댑터는 AbstractHttpAdapterImpl 클래스를 확장해야 합니다.
- 새 프로토콜 리스너에 연결되는 새 어댑터를 작성하십시오. 새 어댑터는 DeviceFormatAdapter 인터페이스를 구현해야 합니다.

웹 컨트롤러

WebSphere Commerce 웹 컨트롤러는 EJB 컨테이너의 설계 패턴과 유사한 설계 패턴을 따르는 응용프로그램 컨테이너입니다.이 컨테이너는 세션 관리(어댑터에서 설정한 세션 지속성에 따라), 트랜잭션 제어, 액세스 제어 및 인증과 같은 서비스를 제공하여 명령의 역할을 단순화합니다.

또한 웹 컨트롤러는 상거래 응용프로그램의 프로그래밍 모델을 실행하는 역할을 수행합니다. 예를 들어, 프로그래밍 모델은 응용프로그램이 작성할 수 있는 명령의 유형을 정의합니다. 각 유형의 명령은 특정 목적을 처리합니다. 비즈니스 로직은 컨트롤러 명령에서 구현되어야 하며 뷰 로직은 뷰 명령에서 구현되어야 합니다. 웹 컨트롤러에서는 컨트롤러 명령이 뷰 이름을 리턴할 것으로 예상합니다. 뷰 이름이 리턴되지 않으면 예외가 발생합니다.

HTTP 요청의 경우, 웹 컨트롤러는 다음과 같은 태스크를 수행합니다.

- javax.transaction 패키지에서 UserTransaction 인터페이스를 사용하여 트랜잭션을 시작합니다.
- 어댑터에서 세션 데이터를 가져옵니다.

- 명령을 호출하기 전에 사용자가 로그온되어야 할지 여부를 판별합니다. 필요한 경우, 사용자의 브라우저를 로그온 URL로 경로 재지정합니다.
- 보안 HTTPS가 URL에 필요한지 확인합니다. 필요하지만 현재 요청이 HTTPS를 사용하지 않는 경우 웹 브라우저를 HTTPS URL로 경로 재지정하십시오.
- 컨트롤러 명령을 호출하고 명령 컨텍스트 및 입력 특성 오브젝트를 전달합니다.
- 트랜잭션 롤백 예외가 발생하고 컨트롤러 명령이 다시 시도될 수 있는 경우, 컨트롤러 명령을 재시도합니다.
- 클라이언트로 돌려보낼 뷰 명령이 있으면 일반적으로 컨트롤러 명령은 뷰 이름을 리턴합니다. 웹 컨트롤러는 해당 뷰의 뷰 명령을 호출합니다. 응답 뷰를 작성하는 방법은 다양합니다. 여기에는 다른 URL로 재지정, JSP 템플리트로 전달 또는 응답 오브젝트에 HTML 문서 작성과 같은 방법이 있습니다.
- 세션 데이터를 저장합니다.
- 세션 데이터를 확약합니다.
- 성공한 경우 현재 트랜잭션을 확약합니다.
- 실패한 경우 현재 트랜잭션을 롤백합니다(환경에 따라).

명령

WebSphere Commerce 명령은 특정 요청의 처리와 관련된 프로그래밍 로직이 들어 있는 bean입니다.

WebSphere Commerce 명령에는 다음 4가지 주요 유형이 있습니다.

컨트롤러 명령

컨트롤러 명령은 특정 비즈니스 프로세스에 관련된 로직을 캡슐화합니다. 컨트롤러 명령의 예에는 주문 처리를 위한 *OrderProcessCmd* 명령 및 사용자의 로그온을 허용하는 *LogonCmd*가 포함됩니다. 일반적으로 컨트롤러 명령은 제어 명령문(예: if, then, else)을 포함하며 비즈니스 프로세스에서 각각의 태스크를 수행할 태스크 명령을 호출합니다. 완료시, 컨트롤러 명령은 뷰 이름을 리턴합니다. 그런 후 웹 컨트롤러는 뷰 명령을 위한 해당 구현 클래스를 판별한 후 뷰 명령을 실행합니다.

태스크 명령

태스크 명령은 특정 응용프로그램 로직 단위를 구현합니다. 일반적으로 컨

트롤러 명령과 한 세트의 태스크 명령이 함께 URL 요청에 대한 응용프로그램 로직을 구현합니다. 태스크 명령은 컨트롤러 명령과 같은 컨테이너에서 실행됩니다.

데이터 bean 명령

데이터 bean 명령은 데이터 bean의 인스턴스가 작성될 때 데이터 bean 관리자가 호출합니다. 데이터 bean 명령의 기본 기능은 데이터를 가진 데이터 bean에 대량 자료 반입하는 것입니다.

뷰 명령

뷰 명령은 클라이언트 요청에 대한 응답으로 뷰를 구성합니다. 뷰 명령에는 다음 세 가지 유형이 있습니다.

경로 재지정 뷰 명령

이 뷰 명령은 URL 경로 재지정과 같은 경로 재지정 프로토콜을 사용하여 뷰를 전송합니다. 컨트롤러 명령은 경로 재지정 프로토콜을 사용하여 뷰를 리턴하기 위해 이러한 뷰 유형의 뷰 명령을 리턴해야 합니다. 경로 재지정 프로토콜이 사용될 때는 브라우저에서 URL 스택을 변경합니다. 다시 로드 키가 입력되면 경로 재지정된 URL은 원래 URL 대신 실행됩니다.

직접 뷰 명령

이 뷰 명령은 응답 뷰를 직접 클라이언트에 전송합니다.

전달 뷰 명령

이 뷰 명령은 뷰 요청을 JSP 템플리트와 같은 다른 웹 구성요소로 전달합니다.

뷰 명령이 호출될 수 있는 세 가지 방식은 다음과 같습니다.

- 요청이 완료되면 컨트롤러 명령은 뷰 명령 이름을 지정합니다.
- 클라이언트에서는 뷰를 직접 요청합니다.
- 명령에서는 오류를 검출하며 오류를 처리하기 위해 오류 태스크가 실행되어야 합니다. 명령은 뷰 명령 이름 관련 예외를 발생시킵니다. 예외가 웹 컨트롤러로 전달되면 해당 명령은 오류 뷰 명령을 실행하고 클라이언트로 응답을 돌려보냅니다.

WebSphere Commerce 엔티티 bean

엔티티 bean은 WebSphere Commerce가 제공하는 지속적인 트랜잭션 상거래 오브젝트입니다. 상거래 도메인에 익숙한 경우, 엔티티 bean은 즉각적으로 WebSphere Commerce 데이터를 나타냅니다. 즉, 데이터베이스 스키마를 이해하지 않고도 상거래 도메인에서 개념과 오브젝트를 보다 근접하게 모델링하는 엔티티 bean의 데이터에 액세스할 수 있습니다. 기존의 엔티티 bean을 확장할 수 있습니다. 또한 고유 응용프로그램별 비즈니스 요구사항에 맞추어 완전한 새 엔티티 bean을 전개할 수 있습니다.

엔티티 bean은 EJB 구성요소 모델에 따라 구현됩니다.

엔티티 bean에 대한 자세한 내용은 53 페이지의 『WebSphere Commerce 엔티티 bean의 구현』을 참조하십시오.

데이터 bean

데이터 bean은 주로 웹 디자이너가 사용하는 Java bean입니다. 주로 이들은 WebSphere Commerce 엔티티에 대한 액세스를 제공합니다. 웹 디자이너는 이들 bean을 JSP 템플리트에 전개하여 표시할 때 페이지에 동적 정보로 대량 자료 반입 되도록 할 수 있습니다. 웹 디자이너는 bean이 제공할 수 있는 데이터와 bean이 입력으로 요구하는 데이터를 이해하기만 하면 됩니다. 비즈니스 로직에서 표시가 분리되므로 웹 디자이너가 bean 작동 방법을 알 필요는 없습니다.

데이터 bean 관리자

JSP 템플리트에 삽입된 WebSphere Commerce 데이터 bean은 페이지에 동적 컨텐츠를 포함할 수 있습니다. 데이터 bean 관리자는 페이지에 다음 코드 행이 삽입될 때 해당 값이 대량 반입될 수 있도록 데이터 bean을 활성화합니다.

```
com.ibm.commerce.beans.DataBeanManager.activate(data_bean, request)
```

여기서, *data_bean*은 활성화될 데이터 bean이고 *request*는 HTTPServletRequest 오브젝트입니다.

JavaServer Pages 템플리트

JSP 템플리트는 특별히 표시용으로 사용되는 특수한 Servlet입니다. URL 요청이 완료되면 웹 컨트롤러는 JSP 템플리트를 호출하는 뷰 명령을 호출합니다. 또한 클라이언트는 관련 명령없이 브라우저에서 직접 JSP 템플리트를 호출할 수 있습니다. 이 경우 JSP 템플리트에서 필요로 하는 모든 데이터 bean이 하나의 트랜잭션 내에서 활성화될 수 있도록 JSP 템플리트의 URL은 그 경로에 요청 servlet을 포함시켜야 합니다. 요청 servlet은 URL 요청을 JSP 템플리트로 전달하여 하나의 트랜잭션 내에서 JSP 템플리트를 실행할 수 있습니다.

데이터 bean 관리자는 그 경로에 요청 servlet을 포함하지 않는 JSP 템플리트의 URL을 거부합니다. JSP 템플리트 및 기타 자원의 보호에 대한 추가 정보는 105 페이지의 제 4 장 『액세스 제어』를 참조하십시오.

instance_name.xml 구성 파일

instance_name.xml 구성 파일(여기서, *instance_name*은 WebSphere Commerce 인스턴스 이름)은 WebSphere Commerce 인스턴스에 대한 구성 정보를 설정합니다. 요청 servlet이 초기화될 때 파일을 읽습니다.

요청 정보 요약

이 절은 요청에 대한 응답으로 발생하는 구성요소 간의 상호 작용 플로우의 정보를 요약합니다.

다음 도표는 각 단계에 대한 설명입니다.

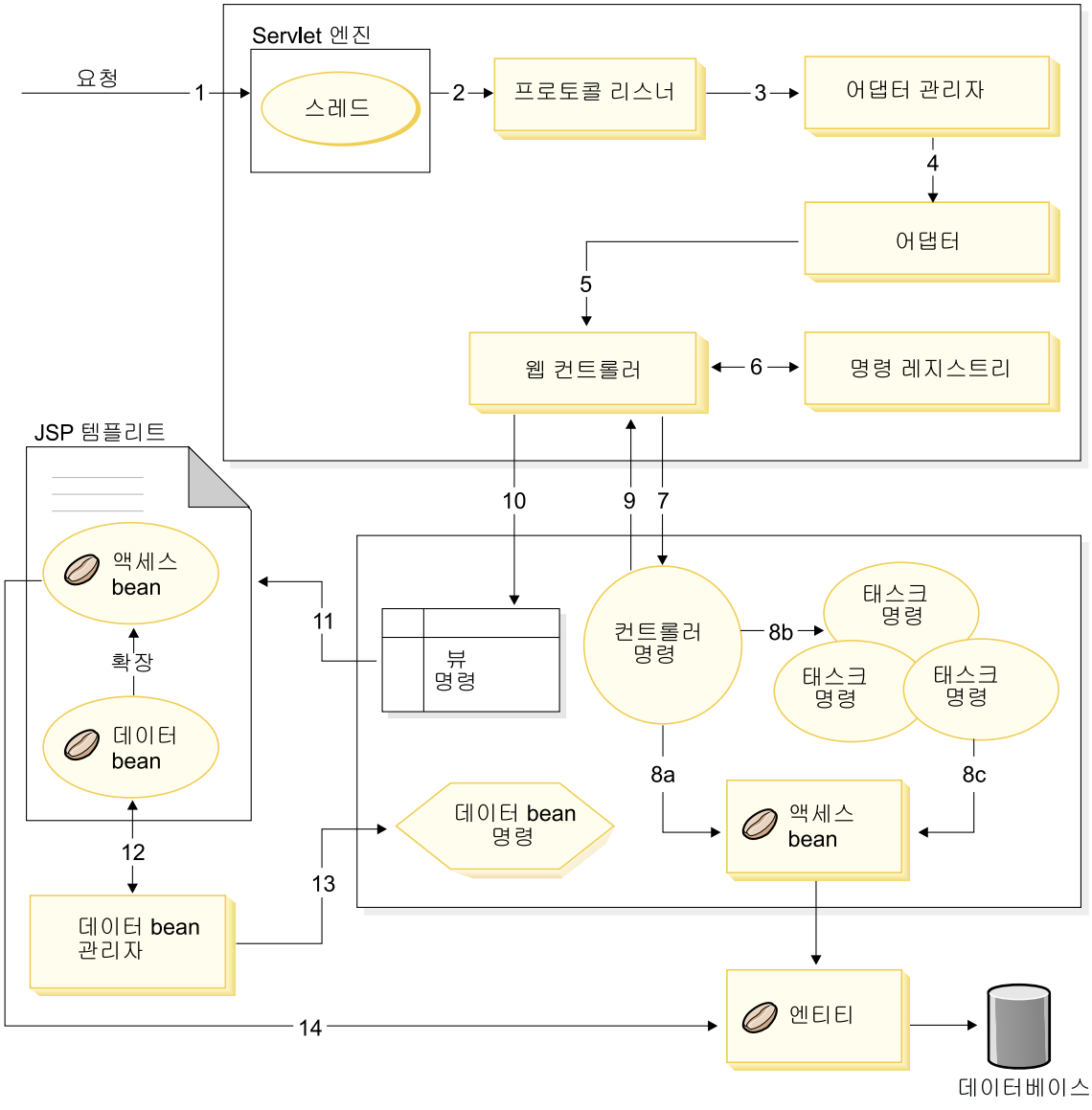


그림 5.

다음은 위의 도표를 설명합니다.

1. 요청은 WebSphere Application Server 플러그인에 의해 Servlet 엔진으로 지정됩니다.

2. 요청이 자체 스레드에서 실행됩니다. servlet 엔진은 프로토콜 리스너로 요청을 지정합니다. 프로토콜 리스너는 HTTP 요청 Servlet이거나 MQ 리스너입니다.
3. 프로토콜 리스너가 요청을 어댑터 관리자에게 전달합니다.
4. 어댑터 관리자는 어떤 어댑터가 요청을 핸들한 다음 해당 어댑터로 요청을 전달할 수 있는지 판별합니다. 예를 들어, 인터넷 브라우저에 요청이 제공되면 어댑터 관리자는 요청을 HTTP 브라우저 어댑터로 전달합니다.
5. 어댑터는 요청을 웹 컨트롤러로 전달합니다.
6. 웹 컨트롤러는 명령 레지스트리를 조회하여 호출할 명령을 판별합니다.
7. 요청의 컨트롤러 명령 사용을 전제로 하면 웹 컨트롤러는 해당 컨트롤러 명령을 호출합니다.
8. 컨트롤러 명령이 실행되면 다음과 같은 여러 경로를 사용할 수 있습니다.
   a. 컨트롤러 명령은 액세스 bean과 해당 엔티티 bean을 사용하여 데이터베이스에 액세스할 수 있습니다.
   b. 컨트롤러 명령은 하나 이상의 태스크 명령을 호출할 수 있습니다. 그런 다음 태스크 명령은 액세스 bean 및 해당 엔티티 bean(8(c) 참조)을 사용하여 데이터베이스에 액세스할 수 있습니다.
9. 완료시, 컨트롤러 명령은 웹 컨트롤러로 뷰 이름을 리턴합니다.
10. 웹 컨트롤러는 VIEWREG 테이블에서 뷰 이름을 찾고 요청자의 장치 유형에 대해 등록된 뷰 명령 구현을 호출합니다.
11. 뷰 명령은 요청을 JSP 템플리트로 전달합니다.
12. JSP 템플리트 내에서 데이터 bean은 데이터베이스로부터 동적 정보를 검색해야 합니다. 데이터 bean 관리자는 데이터 bean을 활성화합니다.
13. 데이터 bean 관리자는 필요한 경우 데이터 bean 명령을 호출합니다.
14. 데이터 bean이 확장된 액세스 bean은 해당 엔티티 bean을 사용하여 데이터베이스에 액세스합니다.

제 2 부 프로그래밍 모델

제 2 장 설계 패턴

WebSphere Commerce 프레임워크를 개발하는 데에는 다양한 설계 패턴 및 메커니즘을 사용합니다. WebSphere Commerce에는 각 WebSphere Commerce 응용프로그램에서 따라 고급 설계 패턴을 제공합니다. 이 장에서는 다음 설계 패턴에 대해 설명합니다.

- MVC(model-view-controller) 설계 패턴
- 명령 설계 패턴
- 표시 설계 패턴

MVC(model-view-controller) 설계 패턴

MVC 설계 패턴은 응용프로그램이 데이터 모델, 표시 정보 및 제어 정보로 구성되도록 지정합니다. 이 패턴에서는 이들 각각이 다른 오브젝트로 분리되어야 합니다.

모델(예: 데이터 정보)에는 응용프로그램 데이터만 들어 있으며 사용자에게 데이터 표시 방법을 설명하는 로직은 없습니다.

뷰(예: 표시 정보)는 사용자에게 모델 데이터를 표시합니다. 뷰를 통해서 모델 데이터에 액세스할 방법을 알 수 있지만 이 데이터의 의미나 사용자가 이를 조작하기 위해 수행할 수 있는 작업은 알 수 없습니다.

마지막으로 *컨트롤러*(예: 제어 정보)는 뷰와 모델 사이에 존재합니다. 컨트롤러는 뷰(또는 또다른 외부 소스)에 의해 트리거된 이벤트를 청취하고 이러한 이벤트에 대한 해당 반응을 실행합니다. 대부분의 경우, 반응은 모델에서 메소드를 호출하는 것입니다. 뷰와 모델이 알림 메커니즘을 통해 연결되므로 이 조치의 결과는 뷰에 자동으로 반영됩니다.

오늘날 대부분의 응용프로그램은 이러한 패턴을 따르며 조금씩 변형됩니다. 예를 들어, 일부 응용프로그램에서는 뷰와 컨트롤러가 이미 결합되었으므로 하나의 클

래스로 간주합니다. 모든 변형은 데이터와 해당 표시를 분리합니다. 이는 응용프로그램의 구조를 더 간단히 만들 뿐만 아니라 코드를 다시 사용할 수 있게 합니다.

견본과 패턴에 대해 참조할 만한 서적이 많으므로 이 책에서는 패턴을 자세하게 설명하고 있지는 않습니다.

다음 도표는 MVC 설계 패턴이 WebSphere Commerce에 적용되는 방법을 보여줍니다.

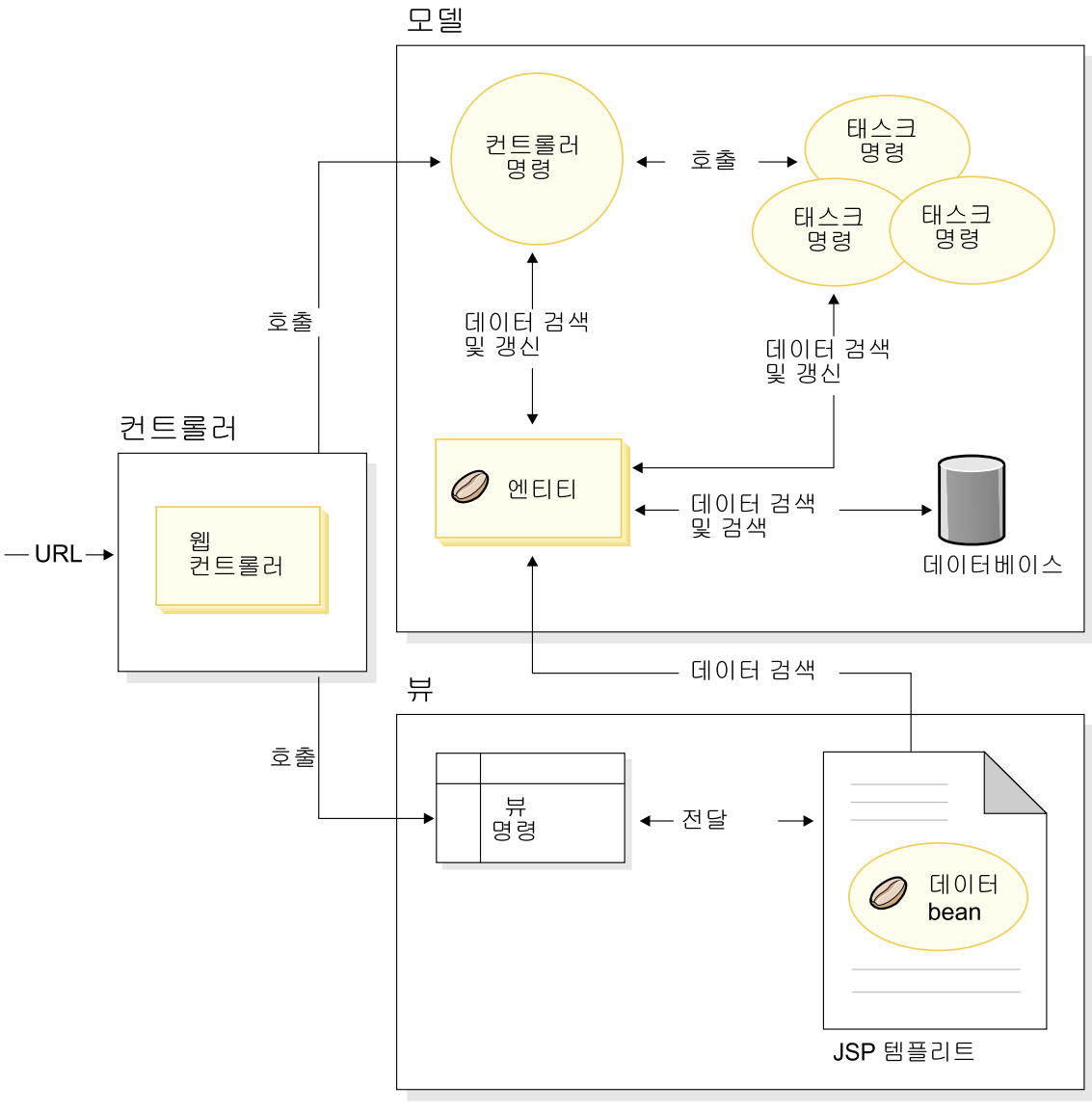

모델
컨트롤러 명령
호출
태스크 명령
태스크 명령
태스크 명령
호출
데이터 검색 및 갱신
데이터 검색 및 갱신
컨트롤러
엔티티
데이터 검색 및 검색
데이터베이스
URL
웹 컨트롤러
데이터 검색
뷰
호출
뷰 명령
전달
데이터 bean
JSP 템플리트

그림 6.

명령 설계 패턴

WebSphere Commerce Server는 다른 원격 응용프로그램에서는 물론 브라우저 기반 씬 클라이언트(browser-based thin-client) 응용프로그램의 요청도 승인합니다. 예를 들어, 원격 조달 시스템이나 다른 commerce 서버로부터 요청할 수 있습니다.

다양한 포맷으로 된 모든 요청은 어댑터 프레임워크를 구성하는 어댑터에 의해 일반 포맷으로 변환됩니다. 일반 포맷으로 요청되면 이 요청들은 WebSphere Commerce 명령으로 이해될 수 있습니다.

명령은 비즈니스 로직을 수행하는 bean입니다. 이들 명령은 고급 처리 로직 또는 분리 비즈니스 로직 태스크 형태로 프로시저 로직을 나타냅니다. 처리 기반 명령은 여러 엔티티 및 기타 명령을 연결하는 컨트롤러의 역할을 하는 반면 태스크 명령은 특정 태스크를 수행하고 하나의 오브젝트에만 액세스할 수 있습니다.

명령 프레임워크

명령 bean은 특정 설계 패턴을 따릅니다. 각 명령은 인터페이스 클래스 (예: `CategoryDisplayCmd`)와 구현 클래스(예: `CategoryDisplayCmdImpl`)를 모두 포함하고 있습니다. 호출자의 관점에서 보면 호출 로직에는 입력 특성 설정, `execute()` 메소드 호출 및 출력 특성 검색이 포함됩니다.

명령 구현의 관점에서 보면 명령은 호출자와 구현 사이의 간접 레벨을 허용하는 표준 명령 설계 패턴을 구현하는 WebSphere 명령 프레임워크를 따릅니다. 이 간접 레벨 내에서 작동 가능한 주요 메커니즘은 다음과 같습니다.

1. 사용자가 명령을 호출하도록 허용되었는지 여부를 판별하는 액세스 제어 정책 관리자를 호출하는 기능
2. 상점 식별자에 따라 상점마다 서로 다른 명령 구현을 실행할 수 있는 기능
3. 요청자의 신용 카드 유형에 근거하여 여러 가지 뷰 구현을 실행하는 기능

다음 도표는 네 가지 기본 명령 유형에 대한 인터페이스의 개념적인 개요를 보여줍니다.

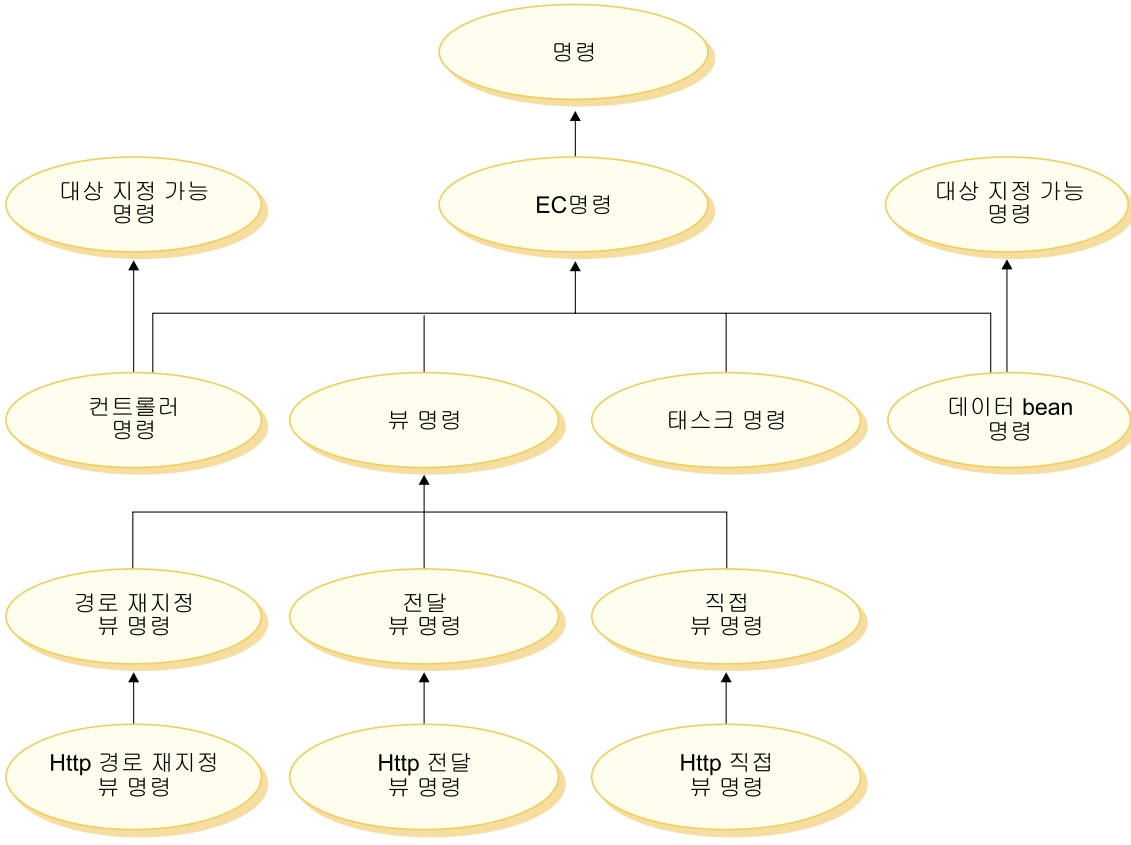


그림 7.

컨트롤러 명령

컨트롤러 명령은 특정 비즈니스 프로세스에 관련된 로직을 캡슐화합니다. 컨트롤러 명령의 예에는 주문 처리를 위한 *OrderProcessCmd* 명령 및 사용자의 로그온을 허용하는 *LogonCmd*가 포함됩니다. 일반적으로 컨트롤러 명령은 제어 명령문(예: if, then, else)을 포함하며 비즈니스 프로세스에서 각각의 태스크를 수행할 태스크 명령을 호출합니다. 완료시, 컨트롤러 명령은 뷰 이름을 리턴합니다. 그런 다음 웹 컨트롤러는 뷰 이름, 저장 식별자 및 장치 유형을 기반으로 뷰 명령을 위한 해당 구현 클래스를 판별한 후 뷰 명령을 실행합니다.

컨트롤러 명령은 대상을 지정할 수 있는 명령이긴 하지만 로컬의 대상 지정만 지원됩니다.

태스크 명령

태스크 명령은 특정 응용프로그램 로직 단위를 구현합니다. 일반적으로 컨트롤러 명령과 한 세트의 태스크 명령이 함께 URL 요청에 대한 응용프로그램 로직을 구현합니다. 태스크 명령은 컨트롤러 명령과 같은 컨테이너에서 실행됩니다.

데이터 bean 명령

데이터 bean 명령은 데이터 bean의 인스턴스가 작성될 때 JSP 페이지에서 호출됩니다. 데이터 bean 명령의 기본 기능은 데이터 bean의 필드를 대량 자료 반입하는 것입니다.

데이터 bean 명령은 대상을 지정할 수 있는 명령이긴 하지만 로컬의 대상 지정만 지원됩니다.

뷰 명령

뷰 명령은 클라이언트 요청에 대한 응답으로 뷰를 구성합니다. 뷰 명령이 호출될 수 있는 세 가지 방식은 다음과 같습니다.

- 요청이 완료되면 컨트롤러 명령은 뷰 명령 이름을 지정합니다.
- 클라이언트에서는 뷰를 직접 요청할 수 있습니다.
- 컨트롤러 또는 태스크 명령은 오류를 검출 및 처리하기 위해 오류 태스크를 실행해야 하는지 결정하고 뷰 명령 이름을 가진 예외를 발생시킵니다. 예외가 웹 컨트롤러로 전달되면 뷰 명령을 실행하고 클라이언트로 응답을 돌려보냅니다.

뷰 명령에는 다음 세 가지 유형이 있습니다.

경로 재지정 뷰 명령

URL 경로 재지정과 같은 경로 재지정 프로토콜을 사용하여 뷰를 전송합니다. 경로 재지정 프로토콜이 필요할 때 컨트롤러 명령은 이 뷰 유형의 뷰 명령을 리턴해야 합니다. 경로 재지정 프로토콜이 사용될 때는 브라우저에서 URL 스택을 변경합니다. 다시 로드 키가 입력되면 경로 재지정된 URL은 원래 URL 대신에 실행됩니다.

직접 뷰 명령

응답 뷰를 직접 클라이언트에 전송합니다.

전달 뷰 명령

뷰 요청을 JSP 템플리트와 같은 다른 웹 구성요소로 전달합니다.

명령 팩토리

새 명령 오브젝트를 작성하기 위해 명령 호출자는 *명령 팩토리*를 사용합니다. 명령 팩토리는 명령 인스턴스를 작성하는 데 사용되는 bean입니다. 명령 팩토리는 팩토리 설계 패턴에 따라 달라집니다. 이 설계 패턴은 호출 클래스가 아닌 구현 클래스(인스턴스 작성)의 정보가 있는 팩토리 클래스에서 오브젝트 인스턴스를 작성하도록 합니다.

팩토리는 새 오브젝트의 인스턴스를 작성하기 위한 수준 높은 방법을 제공합니다. 이 경우, 명령 팩토리는 각각의 상점에 따라 새 명령 오브젝트를 작성할 때 올바른 구현 클래스를 판별하는 방법을 제공합니다. 명령 인터페이스 이름 및 특정 상점 식별자는 인스턴스가 작성될 때 새 명령 오브젝트로 전달됩니다.

명령의 구현 클래스를 지정하는 데는 두 가지 방법이 있습니다.
`defaultCommandClassName` 변수를 사용하여 명령 인터페이스용 코드에 기본 구현 클래스를 직접 지정할 수 있습니다. 예를 들어, 다음 코드는 CategoryDisplayCmd 인터페이스에 존재합니다.

```
String defaultCommandClassName =
     "com.ibm.commerce.catalog.commands.CategoryDisplayCmdImpl"
```

구현 클래스를 지정하는 두 번째 방법은 WebSphere Commerce 명령 레지스트리를 사용하는 것입니다. 명령 레지스트리는 구현 클래스가 한 상점에서 다른 상점으로 변환할 때 항상 사용해야 합니다. 명령 레지스트리에 대한 자세한 내용은 31 페이지에 있습니다.

기본 구현 클래스가 인터페이스의 코드에 지정되고 다른 구현 클래스가 명령 레지스트리에 지정된 경우에 명령 레지스트리의 우선순위가 높습니다.

명령 팩토리 사용 구문은 다음과 같습니다.

```
cmd = CommandFactory.createCommand(interfaceName, storeId)
```

여기서, *interfaceName*은 새 명령 bean의 인터페이스 이름이고 *storeId*는 명령을 구현해야 하는 상점의 식별자입니다. 일반적으로 상점 ID는 commandContext.getStoreId() 메소드를 사용하여 검색할 수 있습니다.

주: 명령 팩토리를 사용하여 비즈니스 정책 명령을 작성하는 구문은 앞의 코드 부분과 다릅니다. 명령 팩토리를 사용한 비즈니스 정책 명령 작성에 대한 자세한 내용은 217 페이지의 『새 비즈니스 정책 호출』을 참조하십시오.

중첩된 컨트롤러 명령

대부분 태스크 명령의 인스턴스를 작성하기 위해 명령 팩토리를 사용하게 되지만, 한 컨트롤러 명령에서 다른 컨트롤러 명령의 인스턴스를 작성하기 위해서 사용할 수도 있습니다. 즉, 다른 컨트롤러 명령에서 하나의 컨트롤러 명령을 호출하는 데 사용합니다.

태스크 명령 및 컨트롤러 명령의 인스턴스 작성을 위한 구문은 동일합니다. 즉, 두 시나리오 모두에서 명령 인스턴스의 이름 및 상점 ID를 지정합니다.

하나의 컨트롤러 명령이 다른 컨트롤러 명령에 중첩되는 경우, 다음 사항에 유의하십시오.

- 일단 중첩된 명령의 인스턴스를 작성하면 해당 setCommandContext 메소드를 호출하고 현재 명령 컨텍스트에 전달하십시오. 다른 세트의 요청 특성을 중첩된 명령으로 전달하고 이러한 매개변수가 명령 컨텍스트에 영향을 미치는 경우, 중첩된 명령의 인스턴스를 작성하기 전에 명령 컨텍스트를 복제해야 합니다. 이를 통해 외부 명령에 대한 명령 컨텍스트 정보를 보존할 수 있습니다.
- 중첩된 명령의 setRequestProperties 메소드를 호출하고 입력 특성을 포함하는 TypedProperties 오브젝트를 전달하는 것이 바람직합니다. 또는 명령의 인터페이스에 정의되어 있는 개별 setter 메소드를 사용하여 필수 특성을 설정할 수 있습니다.
- 입력 특성이 설정되면 중첩된 명령의 execute 메소드를 호출하십시오.
- 처리 완료 시 모든 컨트롤러 명령은 뷰를 리턴해야 하므로 외부 명령은 중첩된 명령이 리턴한 뷰와는 관계가 없습니다.
- 중첩된 명령은 외부 명령의 트랜잭션 범위 안에서 실행됩니다.

중첩된 컨트롤러 명령의 한 예로 다음 코드 단편을 참조하십시오. 이 예는 외부 명령의 메소드 및 명령 팩토리를 사용한 제 2 컨트롤러 명령의 인스턴스 작성 방법을 보여줍니다.

```
yourControllerCmd ctrlCmd = null;

public void processAndCallOtherCommand()
   throws ECException
   {
      ctrlCmd = (yourControllerCmd)CommandFactory.createCommand(
         com.yourcompany.commands.yourControllerCmd, this.getStoreId());
      ctrlCmd.setCommandContext(this.getCommandContext());
      ctrlCmd.setRequestProperties(this.getRequestProperties());
      ctrlCmd.execute();
    }
```

다른 한 예로, 제 1 상점과 다른 상점에 대해 중첩된 명령을 실행하는 경우를 고려하십시오. 이런 경우, 외부 명령의 명령 컨텍스트를 내부 명령이 겹쳐쓰지 않도록 보존해야 합니다.

```
// Make a clone to preserve the command context of the outer commmand
CommandContext cloneCmdCtx = (CommandContext)this.getCommandContext().clone();

//Now pass in a new set of request properties to the cloned command context
cloneCmdCtx.setRequestProperties(reqProp);

yourControllerCmd ctrlCmd = null;

public void processAndCallOtherCommandForOtherStore(int aStoreId)
   throws ECException
   {
      ctrlCmd = (yourControllerCmd)CommandFactory.createCommand(
         com.yourcompany.commands.yourControllerCmd, aStoreId;
      ctrlCmd.setCommandContext(cloneCmdCtx);
      ctrlCmd.setRequestProperties(reqProp);
      ctrlCmd.execute();
    }
```

명령 플로우

이 절에서는 명령과 WebSphere Commerce 데이터베이스 사이의 논리 플로우 개요에 대해 설명합니다. 다음 도표 및 설명은 이러한 플로우를 보여줍니다.

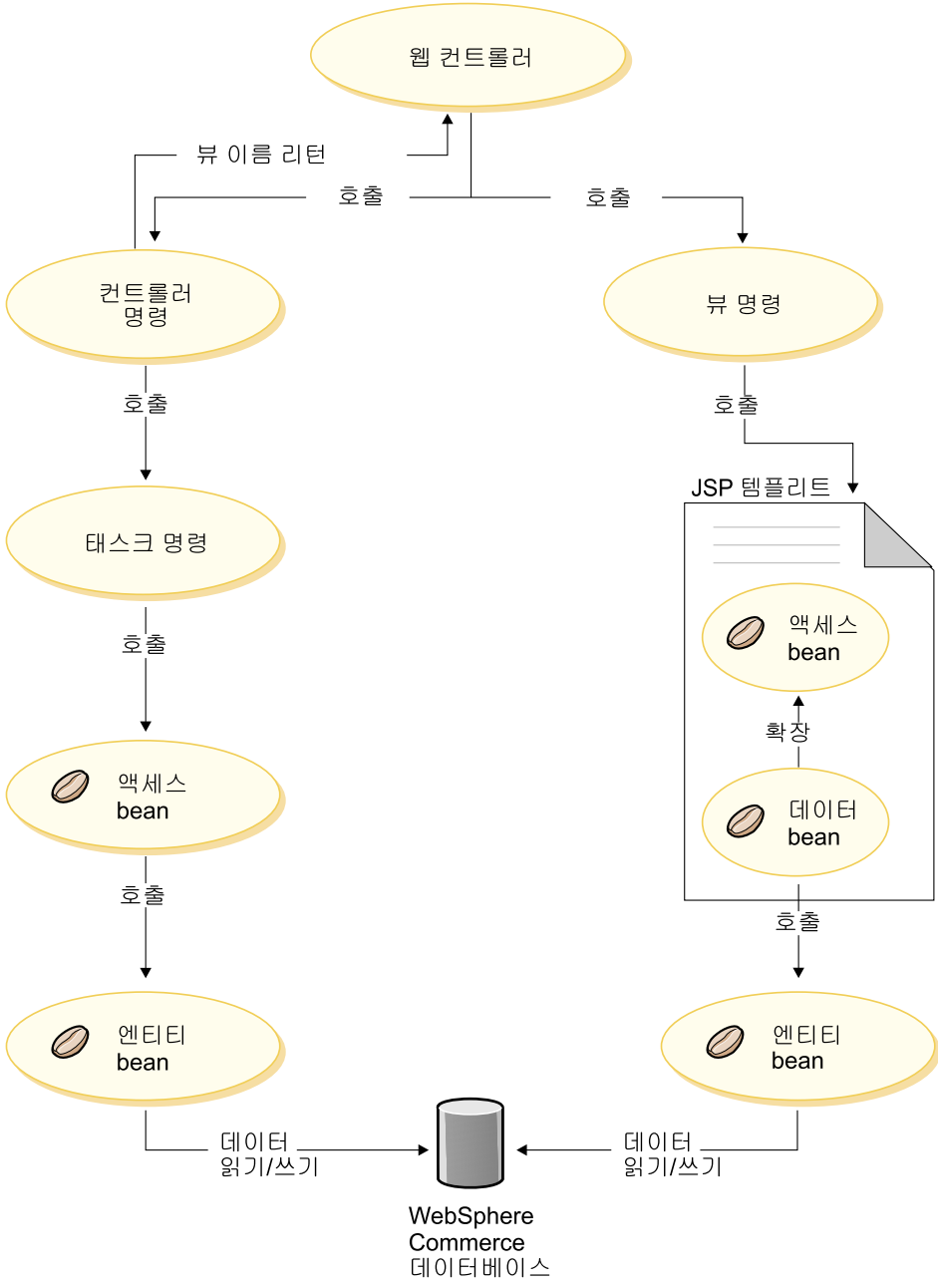

웹 컨트롤러
뷰 이름 리턴
호출
호출
컨트롤러 명령
뷰 명령
호출
호출
JSP 템플리트
태스크 명령
액세스 bean
호출
확장
액세스 bean
데이터 bean
호출
호출
엔티티 bean
엔티티 bean
데이터 읽기/쓰기
데이터 읽기/쓰기
WebSphere Commerce 데이터베이스

그림 8.

웹 컨트롤러가 요청을 수신하면 요청 시 컨트롤러 명령의 호출 여부를 결정합니다. 어떤 경우에도 웹 컨트롤러는 명령의 구현 클래스를 결정한 후 해당 클래스를 호출합니다.

먼저 도표의 왼쪽 부분을 살펴보십시오. 컨트롤러 명령이 비즈니스 처리의 로직을 캡슐화하므로 비즈니스 처리에서 특정 작업 단위를 수행하기 위해 각각의 태스크 명령을 자주 호출합니다. 액세스 bean은 데이터베이스의 정보를 검색하거나 갱신할 때 호출합니다. 태스크 명령 또는 컨트롤러 명령은 액세스 bean을 호출할 수 있습니다. 그런 다음 요청은 액세스 bean에서 WebSphere Commerce 데이터베이스를 읽고 쓸 수 있는 엔티티 bean으로 이동합니다.

이제 도표의 오른쪽 부분을 살펴보십시오. 뷰 명령은 컨트롤러 명령이 처리를 완료하고 호출할 뷰 명령 이름을 리턴한 경우나 오류가 발생하여 오류 보기가 표시되어야 하는 경우에 웹 컨트롤러에서 호출됩니다.

뷰 명령은 일반적으로 클라이언트에 대한 응답을 표시하기 위해 JSP 템플리트를 호출합니다. JSP 템플리트 내에서 데이터 bean은 페이지에 동적 정보를 대량 자료 반입하는 데 사용됩니다. 데이터 bean 관리자가 데이터 bean을 활성화합니다. 데이터 bean(액세스 bean에서 확장)은 해당 엔티티 bean을 호출합니다. JSP 템플리트에서 간접적으로 액세스할 때 엔티티 bean은 일반적으로 데이터베이스에서 정보를 검색합니다(데이터베이스에 정보를 기록하지 않음).

명령 등록 프레임워크

WebSphere Commerce 컨트롤러다음 세 테이블은 명령 레지스트리를 구성합니다.

- URLREG
- CMDREG
- VIEWREG

주: Business 이 절은 비즈니스 정책 명령 등록에는 적용되지 않습니다. 새 비즈니스 정책 명령 등록에 대한 자세한 내용은 196 페이지의 『새 비즈니스 정책 및 비즈니스 정책 명령 등록』을 참조하십시오.

URLREG 테이블

URLREG 테이블은 URI(Universal Resource Indicator)를 컨트롤러 명령 인터페이스에 맵핑합니다. URI는 자원 ID에 대해 간단하고 확장 가능한 메커니즘을 제공합니다. URI는 추상적 또는 물리적 자원을 식별하는 데 사용되는 상대적으로 간단한 문자열입니다. WebSphere Commerce에서 URI에는 명령 정보만 포함됩니다.다음 URL에서 URI 부분은 굵은체로 표시됩니다.

```
http://hostname/webapp/wcs/stores/servlet/StoreCatalogDisplay?
     storeId=store_Id&langId=-1
```

URI와 인터페이스 이름이 일대일로 대응하는 경우, 각 상점에서는 HTTPS 또는 AUTHENTICATION이 명령에 필요한지 지정할 수 있습니다. 각 인바운드 URL 요청에서 웹 컨트롤러는 컨트롤러 명령의 인터페이스 이름을 찾아본 후 해당 이름을 사용하여 CMDREG 테이블에 등록된 대로 올바른 구현 클래스를 판별합니다.

다음 표는 URLREG 데이터베이스 테이블에 포함된 정보에 대해 설명합니다.

열 이름	설명	주석
URL	URI 이름	예를 들어, MyNewCommand 또는 com.ibm.commerce.catalog.commands.ProductDisplayCmd 입니다.
STOREENT_ID	상점 엔티티 식별자	0으로 설정하여 모든 상점의 명령을 사용하거나 고유 상점 식별자로 설정하여 명령이 특정 상점에만 사용되도록 표시할 수 있습니다.
INTERFACENAME	컨트롤러 명령 인터페이스 이름	예: com.ibm.commerce.catalog.commands.ProductDisplayCmdImpl
HTTPS	URL 요청에 보안 HTTP가 필요함	HTTPS가 필수인 경우에는 1을 사용하고 필수가 아닌 경우에는 0을 사용하십시오.
DESCRIPTION	URI 설명	예: This command is used for testing purposes.
AUTHENTICATED	이 URL 요청에 사용자 로그온이 필요함	인증이 필수인 경우에는 1을 사용하고 필수가 아닌 경우에는 0을 사용하십시오.
INTERNAL	명령이 WebSphere Commerce의 내부 명령인지 여부 표시	내부 명령이면 1을 사용하고 외부 명령이면 0을 사용하십시오.

웹 컨트롤러가 URL 요청을 수신하면 요청된 컨트롤러 명령의 인터페이스 이름을 검색하고 이 이름을 사용하여 CMDREG 테이블에서 구현 클래스 이름을 찾아봅니다. 또한 URLREG 테이블에서 HTTPS 열을 확인하여 HTTPS가 URL 요청에 필요한지 판별하기도 합니다.

URL 요청에 의해 호출되는 명령만이 URLREG 테이블에 등록되어야 합니다. 따라서 태스크 또는 뷰 명령이 아닌 컨트롤러 명령만 여기에 등록되어야 합니다.

다음 SQL문은 특정 상점(상점 식별자가 5임)에서 사용되는 MyNewControllerCommand의 항목을 작성합니다.

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCommand',
5,'com.ibm.commerce.commands.MyNewControllerCommand',0,
'This is a test command.',null)
```

insert문의 일반 구문은 다음과 같습니다.

```
 insert into table_name (column_name1,column_name2, ... ,column_namen)
values (column1_value,column2_value,...,columnn_value)
```

문자열 값은 작은따옴표로 묶어야 합니다.

CMDREG 테이블

CMDREG는 명령 등록 테이블입니다. 이 테이블은 명령 인터페이스를 해당 구현 클래스에 맵핑하기 위한 메커니즘을 제공합니다. 상점별로 명령을 사용자 정의할 경우, 인터페이스의 복수 구현이 허용됩니다.

컨트롤러 명령 및 태스크 명령만 CMDREG 테이블에 등록됩니다. 뷰 명령은 VIEWREG 테이블에 등록됩니다.

다음은 CMDREG 데이터베이스 테이블에 포함된 정보에 대해 설명합니다.

열 이름	설명	주석
STOREENT_ID	상점 엔티티 식별자	0으로 설정하여 모든 상점의 명령을 사용하거나 고유 상점 식별자로 설정하여 명령이 특정 상점에만 사용되도록 표시할 수 있습니다.

열 이름	설명	주석
INTERFACENAME	명령 인터페이스 이름	인터페이스를 정의하며 URLREG 테이블에서 사용한 것과 동일한 이름을 사용합니다.
DESCRIPTION	해당 명령에 대한 설명	예: `This command is used for testing purposes.`
CLASSNAME	명령 구현 클래스 이름	일반적으로 끝에 ″Impl″이 추가된 인터페이스 이름
PROPERTIES	명령에 입력 특성으로 설정된 기본 이름-값 쌍	포맷은 URL 조회 문자열과 같습니다(예: ″parm1=val1&parm2=val2″).
LASTUPDATE	이 명령 항목이 마지막으로 갱신된 날짜	
TARGET	명령 대상 이름. 명령이 실제로 실행되는 위치입니다.	로컬 대상이 지원됩니다.

일반적으로 새 컨트롤러 또는 태스크 명령을 작성하면 CMDREG 테이블에 해당 항목을 작성해야 합니다. 예를 들어, 다음 SQL문은 특정 상점(상점 식별자는 5)에서 사용하는 MyNewCommand에 대한 항목을 작성합니다.

```
insert into CMDREG(STOREENT_ID, INTERFACENAME, DESCRIPTION,
CLASSNAME, PROPERTIES, LASTUPDATE,TARGET) values
(5,'com.mycompany.commands.MyNewCommand', 'This is a test command',
'com.mycompany.commands.MyNewCommandImpl',
'myDefaulParm1=myDefaultVal1', '0000-12-01', 'Local')
```

문자열 값은 작은 따옴표로 묶어야 합니다.

작성 중인 명령이 항상 동일한 구현 클래스를 사용할 경우, CMDREG 테이블에 명령을 등록해야 할 필요는 없습니다. 이 경우, 인터페이스에 defaultCommandClassName 속성을 사용하여 구현 클래스를 지정할 수 있습니다. 예를 들면 인터페이스 코드에서 다음을 사용합니다.

```
String defaultCommandClassName =
     "com.ibm.commerce.command.MyNewCommandImpl"
```

이 방법으로 구현 클래스를 지정하면 구현 클래스에 기본 특성을 전달할 수 없으며 모든 상점에 동일한 구현 클래스가 사용되어야 합니다.

등록된 컨트롤러 명령의 예

사이트에 StoreA 및 StoreB의 두 상점이 있는 시나리오를 고려해 보십시오. 상점마다 명령 구현 방식이 다르며 MyUrl 컨트롤러 명령에 대한 보안 요구사항도 다릅니다. 이 절에서는 다음과 같이 사용자 정의할 수 있도록 명령 레지스트리 사용 방법을 보여줍니다.

다음 표는 URLREG 테이블에 있는 StoreA 및 StoreB의 항목을 보여줍니다.

열 이름	**StoreA**의 항목	**StoreB**의 항목
URL	MyUrl	MyUrl
STOREENT_ID	11	22
INTERFACENAME	com.ibm.commerce. mycommands.myUrl	com.ibm.commerce. mycommands. myUrl
HTTPS	1	1
DESCRIPTION	URLREG 테이블의 항목 예	URLREG 테이블의 항목 예
AUTHENTICATED	1	0
INTERNAL	널(NULL)값	널(NULL)값

주: INTERFACENAME의 값에 있는 공백은 표시용입니다. 각각의 값은 실제로 하나의 연속 문자열입니다.

URLREG 테이블의 항목에 따라 웹 컨트롤러는 MyURL URI의 인터페이스 이름이 com.ibm.commerce.mycommands.MyUrl임을 판별합니다. 또한 StoreA에서는 명령이 HTTPS 및 인증을 모두 사용하여 실행되어야 하나 StoreB에서는 HTTPS만 필요합니다. HTTPS 및 인증에 대한 값은 인터페이스가 아니라 웹 컨트롤러에서 사용됩니다.

다음 도표는 이러한 플로우를 보여줍니다.

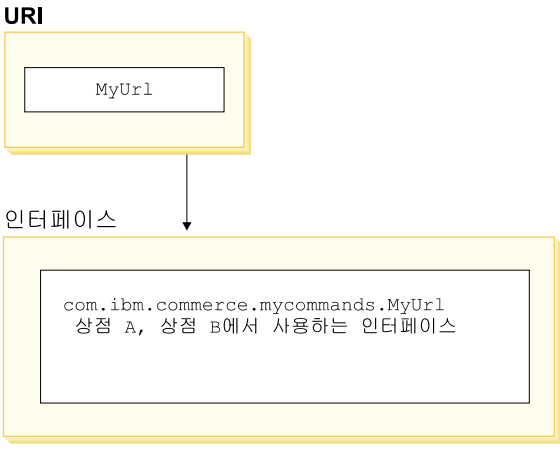


그림 9.

다음 표는 CMDREG 테이블의 항목을 보여줍니다. 이 예에 사용되는 열만 표시됩니다.

열 이름	**StoreA의 항목**	**StoreB의 항목**
STOREENT_ID	11	22
INTERFACENAME	com.ibm.commerce. mycommands.myUrl	com.ibm.commerce. mycommands. myUrl
CLASSNAME	com.ibm.commerce. mycommands. myUrlStoreAImpl	com.ibm.commerce. mycommands. myUrlStoreBImpl

주: INTERFACENAME 및 CLASSNAME의 값에 있는 공백은 표시용입니다. 각각의 값은 실제로 하나의 연속 문자열입니다.

CMDREG 테이블의 항목에 따라 웹 컨트롤러는 StoreA의 경우 com.ibm.commerce.mycommands.MyUrl 인터페이스 구현 클래스가 com.ibm.commerce.mycommands.MyUrlStoreAImpl임을 판별합니다. 또한 StoreB의 경우 동일한 인터페이스의 구현 클래스가 com.ibm.commerce.mycommands.MyUrlStoreBImpl임을 판별합니다. 다음 도표는 이러한 플로우를 보여줍니다.

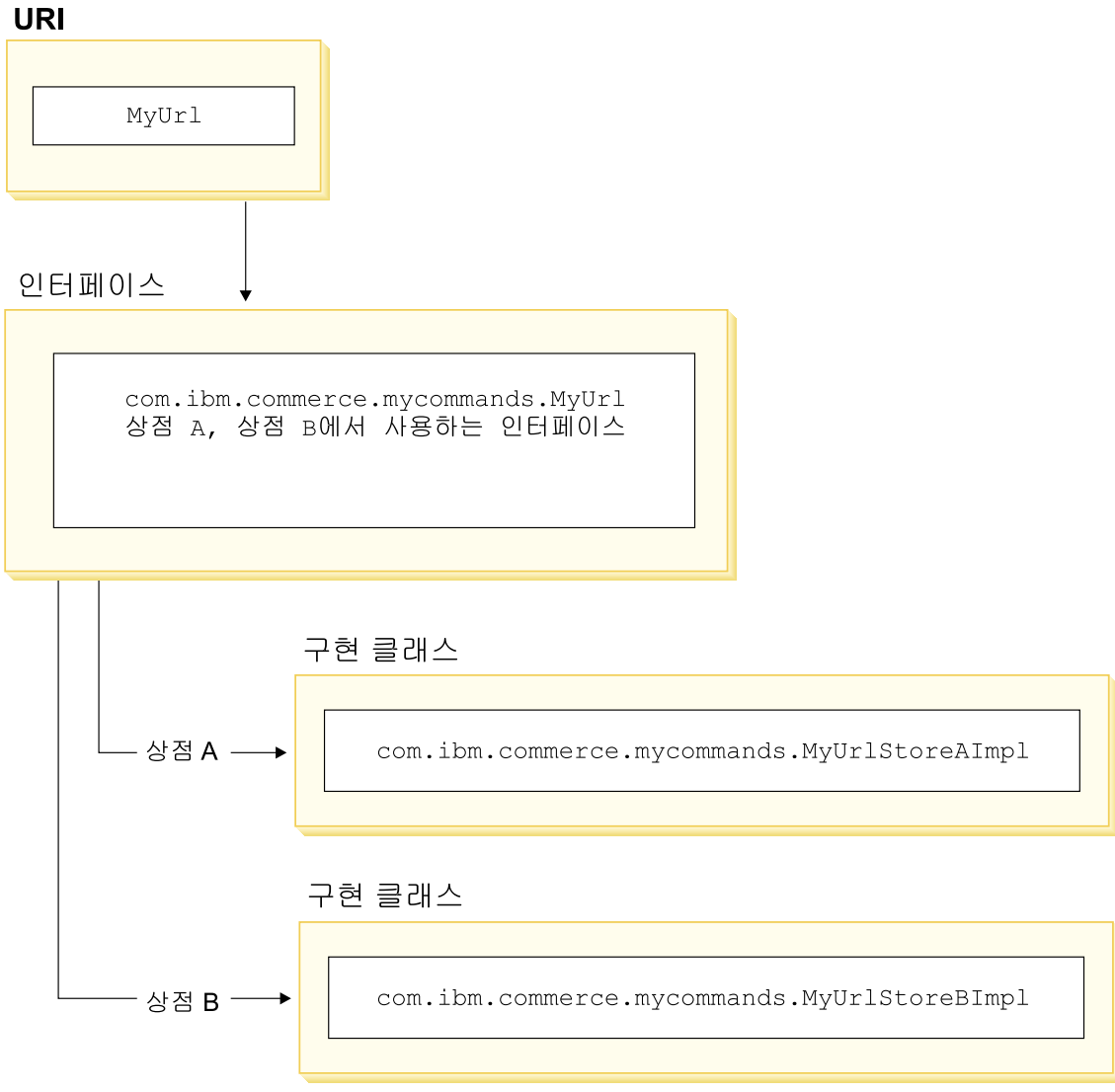


그림 10.

VIEWREG 테이블

VIEWREG 테이블은 장치 고유 및 상점 고유 뷰 구현을 등록할 수 있도록 합니다. 이 테이블을 사용하여 여러 가지 뷰 구현을 등록합니다. 그러면 명령 프레임워크가 다양한 장치에 다른 뷰를 리턴할 수 있습니다.

뷰 이름이 컨트롤러 명령에서 리턴되거나 예외에 지정되는 경우, 웹 컨트롤러가 VIEWREG 테이블의 뷰 구현을 판별합니다. 동일한 구현 클래스에 복수 뷰 이름을 맵핑할 수 있습니다.

열 이름	설명	주석
VIEWNAME	뷰 이름	예: `AddressForm`
DEVICEFMT_ID	장치 유형 식별자	사용 가능한 옵션은 다음과 같습니다. • BROWSER(기본값) • I_MODE • E-mail • MQXML • MQNC
STOREENT_ID	상점 엔티티 식별자	`0`으로 설정하여 모든 상점의 명령을 사용하거나 고유 상점 식별자로 설정하여 명령이 특정 상점에만 사용되도록 표시할 수 있습니다.
INTERFACENAME	뷰 명령 인터페이스 이름	기본 옵션은 `ForwardView`, DirectView 및 RedirectView입니다.
CLASSNAME	뷰 명령 구현 클래스 이름	기본 구현을 사용할 수 있습니다.
PROPERTIES	명령에 입력 특성으로 설정된 기본 이름-값 쌍	동일한 페이지가 항상 표시되면 이 특성에 JSP 파일 이름을 설정하십시오(`docname=`*`jsp_name`*`.jsp`). 동일한 JSP 템플리트가 모든 상점에 사용되면 상점 고유의 디렉토리가 사용되지 않도록 `storeDir=no`로 설정하십시오. 일반 사용자가 명령을 호출할 수 있는 경우, `isGeneric=true`를 설정하십시오.
DESCRIPTION	해당 명령에 대한 설명	
HTTPS	URL 요청에 보안 HTTP가 필요함	HTTPS가 필수인 경우에는 `1`을 사용하고 필수가 아닌 경우에는 `0`을 사용하십시오.
LASTUPDATE	항목이 마지막으로 갱신된 날짜	

열 이름	설명	주석
INTERNAL	명령이 WebSphere Commerce의 내부 명령인지 여부 표시	내부 명령이면 1을 사용하고 외부 명령이면 0을 사용하십시오.

새 뷰를 작성하는 경우, VIEWREG 테이블에 해당 항목을 작성해야 합니다. 다음 조건 중 하나가 충족되면 VIEWREG 테이블에 뷰를 등록해야 합니다.

- 액세스 제어에서 뷰를 실행합니다.
- 여러 개의 뷰 명령 구현이 있습니다.
- 특성이 PROPERTIES 열에 설정되어 있습니다.

등록된 뷰는 뷰 이름을 사용하여 뷰 레지스트리를 통하거나 실제 표시 파일 이름을 사용하여 직접 액세스할 수 있습니다. VIEWREG 테이블에 등록되지 않은 뷰는 클라이언트에서 실제 표시 파일 이름을 사용할 경우에만 액세스될 수 있습니다.

다음과 같이 VIEWREG 항목이 있는 *MyView* 뷰를 참조하십시오.

열 이름	항목
VIEWNAME	MyView
DEVICEFMT_ID	BROWSER
STOREENT_ID	0
INTERFACENAME	com.ibm.commerce.commands.ForwardViewCommand
CLASSNAME	com.ibm.commerce.commands.HTTPForwardViewCommandImp
PROPERTIES	docname=MyView.jsp
DESCRIPTION	An example for calling a JSP template using either the view name or directly from a URL.
HTTPS	0
LASTUPDATE	2000-11-30
INTERNAL	0

MyView는 등록된 뷰이므로 클라이언트는 뷰 이름을 사용하거나 실제 표시 파일 이름을 뷰 이름으로 대체하여 뷰에 액세스할 수 있습니다. 뷰 이름을 사용하는 견본 URL은 다음과 같습니다.

`http://`*`hostname`*`.com/webapp/wcs/stores/servlet/MyView`

파일 이름을 사용하는 견본 URL은 다음과 같습니다.

```
http://hostname.com/webapp/wcs/stores/servlet/MyView.jsp
```

클라이언트가 등록된 뷰를 직접 호출할 가능성이 있는 경우(표시 파일 이름을 사용하여), 이 예에 표시된 대로 실제 표시 파일 이름(MyView 및 MyView.jsp)과 동일한 뷰 이름을 사용하여 뷰를 등록해야 합니다.

테이블에 등록되지 않은 뷰는 표시 파일 이름을 사용해야 호출될 수 있습니다. 따라서 MyUnregisteredView.jsp 파일을 사용하는 미등록 뷰가 있으면 뷰에 액세스할 URL은 다음과 같습니다.

```
http://hostname.com/webapp/wcs/stores/servlet/MyUnregisteredView.jsp
```

다음 SQL문은 하나의 특정 상점에서 사용하는 *MyNewView*에 대한 항목을 작성합니다.

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID,
INTERFACENAME, CLASSNAME, PROPERTIES, DESCRIPTION,HTTPS, LASTUPDATE,
INTERNAL) values ('MyNewView', -1, 5, 'com.ibm.commerce.command.
ForwardViewCommand', 'com.ibm.commmerce.command.
HttpForwardViewCommandImpl', 'docname=MyNewView.jsp', 'A test view.',
0, '0000-12-01', 0)
```

다음 표는 주요 정보가 있는 또다른 견본 VIEWREG 테이블을 제공합니다.

VIEW NAME	INTERFACE - NAME	CLASSNAME	PROPERTIES
ProductDisplay 뷰	전달 뷰 명령	HttpForwardView CommandImpl	docname= UserArea/ ServiceSection/ InterestItemList Subsection/ WishListDisplay.jsp
일반 응용프로그램 오류	전달 뷰 명령	HttpForwardView CommandImpl	docname =Generic Application Error. jsp&storeDir=no
GenericSystem 오류	전달 뷰 명령	HttpForwardView CommandImpl	docname = Generic System Error.jsp &storeDir=no

VIEW NAME	INTERFACE - NAME	CLASSNAME	PROPERTIES
LogonForm	전달 뷰 명령	HttpForwardViewCommandImpl	docname = LoginForm.jsp &generic=true &storeDir=no

주: VIEWNAME, INTERFACENAME, CLASSNAME 및 PROPERTIES의 값에 있는 모든 공백은 표시용입니다. 각각의 값은 실제로 하나의 연속 문자열입니다. 열 이름의 하이픈 또한 표시용입니다.

앞에 나온 테이블은 다음과 같은 시나리오를 보여줍니다.

- *ProductDisplayView* 뷰 이름은 컨트롤러 명령(ProductDisplay)에서 웹 컨트롤러로 리턴됩니다. 웹 컨트롤러는 ProductDisplayView 뷰 명령 이름과 해당 장치 식별자를 사용하여 뷰 명령 인터페이스 및 클래스 이름을 판별합니다. 뷰 명령은 상점 및 장치 식별자마다 서로 다른 구현 클래스를 가질 수 있습니다. 그러나 인터페이스 이름은 뷰 명령 유형을 정의하므로 동일해야 합니다.
- 컨트롤러 명령 또는 태스크 명령에서 잘못된 사용자 매개변수에 대한 ECApplication 예외를 발생시킬 경우, 다음이 발생할 수 있습니다.
  - 컨트롤러 명령(응용프로그램 예외의 경우 호출)내에 지정된 뷰가 있는 경우, 해당 뷰 항목이 VIEWREG 테이블에서 검색되어 이에 따라 처리됩니다.
  - 뷰가 지정되어 있지 않은 경우, GenericApplicationError 명령이 호출되며 데이터베이스에 등록된 JSP 템플리트가 표시됩니다. 예를 들어, 앞의 테이블을 사용하는 경우, `GenericApplicationError.jsp` 템플리트가 표시됩니다.
- 컨트롤러 명령 또는 태스크 명령이 시스템 예외에 대한 ECSystem 예외를 발생시키는 경우, 다음과 같이 발생할 수 있습니다.
  - 컨트롤러 명령(시스템 예외의 경우 호출) 내에 지정된 뷰가 있는 경우, 해당 뷰의 항목이 VIEWREG 테이블에서 검색되어 이에 따라 처리됩니다.
  - 뷰가 지정되어 있지 않은 경우, GenericSystemError 명령이 호출되며 데이터베이스에 등록된 JSP 템플리트가 표시됩니다. 예를 들어, 앞의 테이블을 사용하는 경우, `GenericSystemError.jsp` 템플리트가 표시됩니다.

- 브라우저 클라이언트는 로그온 URL을 입력하여 로그온 페이지를 호출할 수 있습니다. storeDir 특성이 “no”로 설정되어 있으므로 상점 고유 정보가 JSP 템플리트의 경로에 포함되어 있지 않습니다. 그러므로 모든 상점의 고객에게 동일한 로그온 페이지가 표시됩니다.

표시 설계 패턴

표시 페이지는 클라이언트에 응답을 리턴합니다. 일반적으로 표시 페이지는 JSP 템플리트로 구현할 수 있지만(권장되는 메소드) Servlet으로 직접 작성될 수 있습니다.

여러 장치 유형을 지원하기 위해 뷰 명령에 액세스하는 URL은 실제 JSP 파일의 이름이 아니라 뷰 이름을 사용해야 합니다.

이러한 간접 레벨의 기본 원리는 JSP 템플리트가 뷰를 표시하는 것입니다. 적절한 뷰를 선택할 수 있는 기능은(예: 요청 컨텍스트의 로케일, 장치 유형 또는 기타 데이터 기준) 매우 바람직합니다(특히 하나의 요청에는 종종 여러 개의 사용 가능한 뷰가 있을 경우). 예를 들어, 두 구매자가 상점 홈페이지 요청합니다(한 구매자는 일반 웹 브라우저 사용, 다른 구매자는 휴대폰 사용). 분명 동일한 홈페이지가 각 구매자에게 표시되어서는 안됩니다. 요청을 승인한 다음, 명령 등록 프레임워크의 정보에 따라 각 구매자가 수신하는 뷰를 판별하는 것은 웹 컨트롤러의 작업입니다.

JSP 템플리트 및 데이터 bean

데이터 bean은 동적 컨텐츠를 제공하기 위해 JSP 템플리트 내에서 사용되는 Java bean입니다. 일반적으로 데이터 bean은 WebSphere Commerce 엔티티 bean을 간단히 나타냅니다.데이터 bean은 엔티티 bean 내에서 검색되거나 설정될 수 있는 특성을 캡슐화합니다. 이처럼 데이터 bean은 동적 데이터를 JSP 템플리트로 통합하는 태스크를 단순화합니다.

데이터 bean에는 표시 페이지에 사용될 수 있는 특성을 정의하는 *BeanInfo* 클래스가 있습니다. 또한 BeanInfo 클래스는 지원되는 모든 WebSphere Commerce 언어로 특성 이름을 제공함으로써 다국어 지원 사이트에서 데이터 bean을 사용할 데이터 bean은 다음과 같은 호출로 활성화됩니다.

```
com.ibm.commerce.beans.DataBeanManager.activate(data_bean, request)
```

여기서, *data_bean*은 활성화될 데이터 bean이고 *request*는 HTTPServletRequest 오브젝트입니다.

상점 개발자는 JSP 템플리트 개발 시 상점 특성 및 다국어 지원 문제를 고려해야 합니다. 다국어 지원에 대한 자세한 정보는 *WebSphere Commerce 상점 개발 안내서*를 참조하십시오.

데이터 bean 보안 고려사항

데이터 bean을 사용하는 특정 코딩 방식은 악의적인 사용자가 허가받지 않은 방법으로 데이터베이스에 액세스할 수 있는 가능성을 최소화합니다. SQL문의 insert, select, update 및 delete 부분은 개발시 작성되어야 합니다. 런타임 입력 정보를 수집하려면 매개변수 삽입을 사용하십시오.

매개변수 삽입을 사용한 런타임 입력 정보 수집의 예는 다음과 같습니다.

```
select * from Order where owner =?
```

반면 입력 문자열을 사용하여 SQL문을 작성할 수는 없습니다. 입력 문자열을 사용하는 예는 다음과 같습니다.

```
select * from Order where owner = "input_string"
```

데이터 bean 유형

데이터 bean은 JSP 템플리트에서 동적 데이터를 제공하는 데 주로 사용되는 Java bean입니다. 데이터 bean에는 두 가지 유형인 스마트 데이터 bean 및 명령 데이터 bean이 있습니다.

스마트 데이터 bean에서는 *lazy fetch* 메소드를 사용하여 고유의 데이터를 검색합니다. 이 데이터 bean 유형은 필요한 만큼만 데이터를 검색하므로 액세스 bean의 모든 데이터가 필요하지 않은 상황에서 더 나은 성능을 제공할 수 있습니다. 데이터베이스에 액세스해야 하는 스마트 데이터 bean은 해당 엔티티 bean의 액세스 bean에서 확장되어 com.ibm.commerce.SmartDataBean 인터페이스를 구현해야 합니다. 예를 들어, ProductData 데이터 bean은 상품 엔티티 bean에 해당하는 ProductAccessBean 액세스 bean을 확장합니다.

일부 스마트 데이터 bean의 경우 데이터베이스에는 액세스할 필요가 없습니다. 예를 들어, PropertyResource 스마트 데이터 bean은 데이터베이스가 아니라 자원 번들에서 데이터를 검색합니다. 데이터베이스에 액세스할 필요가 없는 경우 스마트 데이터 bean은 SmartDataBeanImpl 클래스를 확장해야 합니다.

명령 데이터 bean은 작은 데이터 bean으로 해당 데이터를 검색합니다. 명령은 JSP 템플리트에 필요한지 여부에 상관없이 한번에 모든 데이터 bean 속성을 검색합니다. 결과적으로 데이터 bean에서 선택된 속성만 사용하는 JSP 템플리트의 경우 명령 데이터 bean의 속도가 느려질 수 있습니다. 대부분의 속성 또는 모든 속성이 필요한 JSP 템플리트의 경우 명령 데이터 bean은 매우 편리합니다.

명령 데이터 bean은 해당 액세스 bean에서 확장되어 com.ibm.commerce.CommandDataBean 인터페이스를 구현할 수도 있습니다.

데이터 bean 인터페이스

데이터 bean은 다음 Java 인터페이스 중 하나 또는 모두를 구현합니다.

- com.ibm.commerce.SmartDataBean.
- com.ibm.commerce.CommandDataBean
- com.ibm.commerce.InputDataBean(선택사항)

각 Java 인터페이스는 데이터 bean에 대량 자료 반입되는 데이터 소스에 대해 설명합니다. 여러 인터페이스를 구현하면 데이터 bean은 다양한 소스의 데이터에 액세스할 수 있습니다. 각 인터페이스에 대한 추가 정보는 아래에 있습니다.

SmartDataBean 인터페이스: SmartDataBean 인터페이스를 구현하는 데이터 bean은 연관된 데이터 bean 명령없이 고유의 데이터를 검색할 수 있습니다. 스마트 데이터 bean은 보통 해당 엔티티 bean의 액세스 bean에서 확장합니다. 스마트 데이터 bean이 활성화되면 데이터 bean 관리자는 데이터 bean의 populate 메소드를 호출합니다. 데이터 bean은 populate 메소드를 사용하여 연관된 오브젝트의 속성을 제외하고 모든 속성을 검색할 수 있습니다. 예를 들어, 데이터 bean이 엔티티 bean의 액세스 bean 클래스에서 확장할 경우, 데이터 bean은 refreshCopyHelper 메소드를 호출합니다. 해당 엔티티 bean의 모든 속성은 스마트 데이터 bean에 자동으로 대량 자료 반입됩니다. 그러나 엔티티 bean에 연관된

오브젝트가 있는 경우, 해당 오브젝트의 속성은 검색되지 않습니다. 스마트 데이터 bean을 사용할 때의 기본적인 이점은 다음과 같습니다.

- 구현은 간단하며 데이터 bean 명령을 작성할 필요가 없습니다.
- 새 필드가 엔티티 bean에 추가되면 데이터 bean을 변경할 필요가 없습니다. 엔티티 bean이 수정되면 액세스 bean은 다시 작성되어야 합니다(WebSphere Studio Application Developer의 도구를 사용). 액세스 bean이 다시 작성되자마자 스마트 데이터 bean에 모든 새 속성을 자동으로 사용할 수 있습니다.
- 엔티티 bean에는 종종 연관된 오브젝트를 나타내는 속성이 포함됩니다. 성능상의 이유로 인해 스마트 데이터 bean은 자동으로 이러한 속성을 검색하지 않습니다. 대신 다음 도표에 표시된 대로 필요할 때까지 이러한 속성 검색을 지연하는 것이 좋습니다.

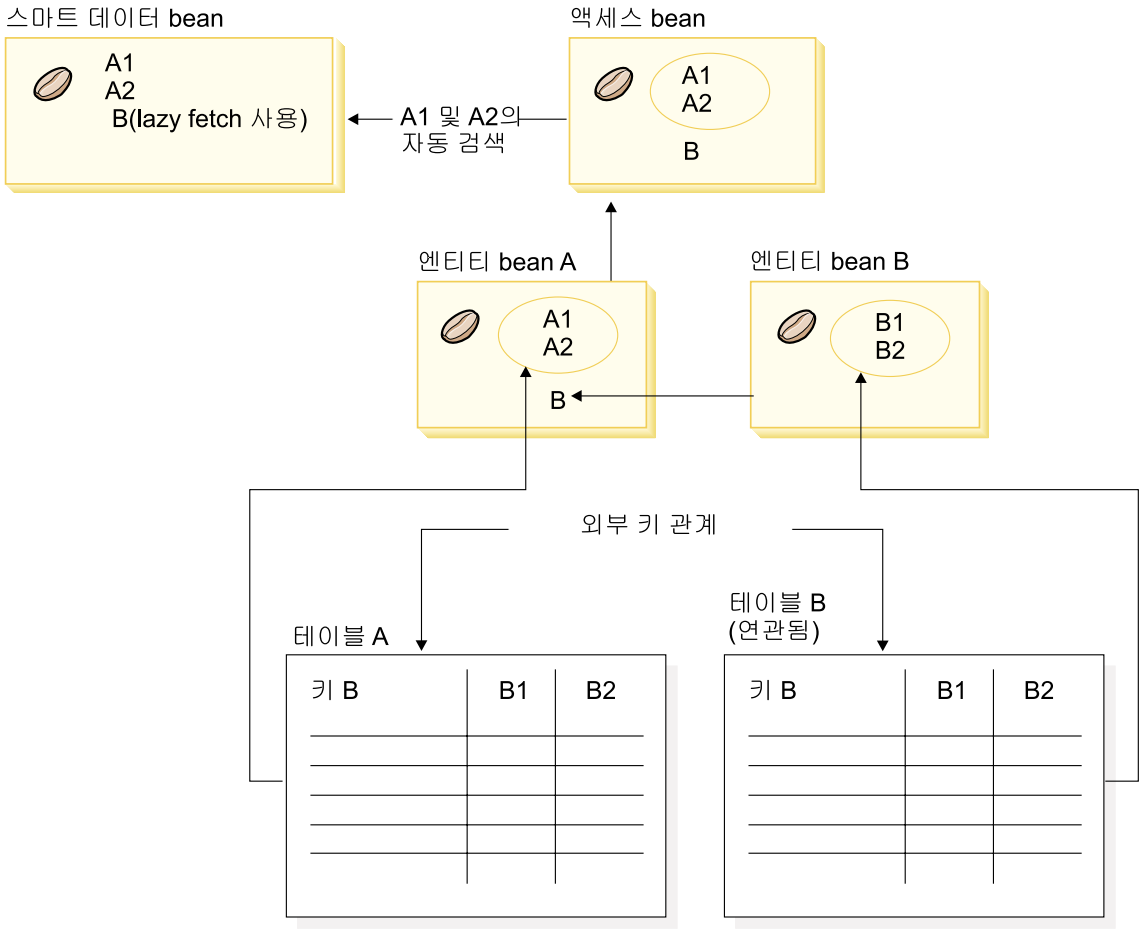


그림 11.

lazy fetch 검색 구현에 대한 자세한 내용은 48 페이지의 『Lazy fetch 데이터 검색』을 참조하십시오.

CommandDataBean 인터페이스: CommandDataBean 인터페이스를 구현하는 데이터 bean은 데이터 bean 명령에서 데이터를 검색합니다. 이 유형의 데이터 bean은 작은 오브젝트이며 데이터 bean 명령을 기초로 하여 해당 데이터에 대량 자료 반입합니다. 데이터 bean은 데이터 bean 명령의 인터페이스 이름을 리턴하는 getCommandInterfaceName() 메소드(com.ibm.commerce.CommandDataBean 인터페이스에서 정의한 대로)를 구현해야 합니다.

InputDataBean 인터페이스: InputDataBean 인터페이스를 구현하는 데이터 bean은 뷰 명령에 의해 설정된 URL 매개변수 또는 속성으로부터 데이터를 검색합니다.

이 인터페이스에 정의된 속성은 추가 데이터를 가져오기 위해 1차 키 필드로 사용될 수 있습니다. JSP 템플리트가 호출되면 작성된 JSP Servlet 코드는 URL 매개변수와 일치하는 모든 속성에 대량 자료 반입한 후 데이터 bean 관리자에게 데이터 bean을 전달하여 데이터 bean을 활성화합니다. 그런 후 데이터 bean 관리자는 데이터 bean의 setRequestProperties() 메소드(com.ibm.commerce.InputDataBean 인터페이스에서 정의한 대로)를 호출하여 뷰 명령에 의해 설정된 모든 속성을 전달합니다. 데이터 bean을 활성화하려면 다음 코드가 필요함에 유의해야 합니다.

```
com.ibm.commerce.beans.DataBeanManager.activate(data_bean, request)
```

여기서, *data_bean*은 활성화될 데이터 bean이고 *request*는 HTTPServletRequest 오브젝트입니다.

BeanInfo 클래스

데이터 bean은 java.lang.Object.BeanInfo 인터페이스를 구현하는 BeanInfo 클래스 없이 완성되지 않습니다. BeanInfo 클래스는 데이터 bean의 메소드 및 특성에 대한 명시적인 정보를 제공하는 데 사용됩니다. 웹 디자이너의 데이터 bean 구현 클래스에서 공용 런타임 메소드를 숨기거나 데이터 bean의 속성 각각에 대해 해당 표시 문자열을 설정하기 위해 해당 클래스가 사용될 수 있습니다.

BeanInfo 클래스에 대한 자세한 내용은 Sun Microsystems의 JavaBeans 스펙을 참조하십시오.

데이터 bean 활성화

데이터 bean은 com.ibm.commerce.beans.DataBeanManager 클래스에 있는 activate 또는 silentActivate 메소드를 사용하여 활성화될 수 있습니다. activate 메소드는 모든 속성을 사용할 수 있는 경우에만 활성화 이벤트가 완료되는 전체 활성화 메소드입니다. 하나의 속성만 사용할 수 없는 경우에도 전체 활성화 처리에 대해 예외가 발생합니다.

silentActivate 메소드에서는 개별 속성을 사용할 수 없을 때 예외가 발생하지 않습니다.

JSP 템플리트 내에서 컨트롤러 명령 호출

JSP 템플리트에서 컨트롤러 명령을 호출하는 것과 표시와 로직을 구분하는 것이 일관되지 않아도, 이를 수행해야 하는 상황이 발생할 수 있습니다. 그러한 경우 ControllerCommandInvokerDataBean을 사용할 수 있습니다.

이 데이터 bean을 사용하면 호출할 명령의 인터페이스 이름을 지정하거나, 호출할 명령의 이름을 직접 설정할 수 있습니다. 그 명령에 대해 요청 특성을 설정할 수도 있습니다.

이 데이터 bean이 데이터 bean 관리자에 의해 활성화될 경우, 컨트롤러 명령이 실행되어 JSP 템플리트에 대해 응답 특성을 사용할 수 있습니다.

이 데이터 bean을 활성화하기 전에 setRequestProperties 메소드를 사용하지 않는 경우, 요청 오브젝트의 매개변수가 bean으로 전달되고 이후 컨트롤러 명령으로도 전달됩니다. 그러나 이 데이터 bean을 활성화하기 전에 setRequestProperties 메소드를 호출하는 경우, 지정된 특성(setRequestProperties 메소드로 전달된 특성) 및 CMDREG 테이블의 PROPERTIES 열에 지정된 기본 특성만 명령에 사용할 수 있게 됩니다.

컨트롤러 명령이 실행되면 뷰를 실행할 수 있습니다.

기타 컨트롤러 명령을 호출하기 위해 동일한 데이터 bean 인스턴스를 다시 사용해서는 안 됩니다. 이 인스턴스에는 원래 용도의 데이터 및 상태 정보가 들어 있기 때문입니다.

Lazy fetch 데이터 검색

데이터 bean이 활성화되면 데이터 bean 명령 또는 데이터 bean의 populate() 메소드에 의해 대량 자료 반입될 수 있습니다. 검색된 속성은 데이터 bean의 해당 엔티티 bean에 있습니다. 또한 엔티티 bean에는 그 자체가 다수의 속성을 갖는 연관된 오브젝트가 있을 수도 있습니다.

활성화시, 연관된 모든 오브젝트의 속성이 자동으로 검색되면 성능 문제점이 발생할 수 있습니다. 성능은 연관된 오브젝트의 수가 증가함에 따라 저하될 수 있습니다.

다수의 비슷한 모델 소개, 나은 모델 소개 또는 부속 상품(연관된 오브젝트)이 들어 있는 상품 데이터 bean을 고려해 보십시오. 상품 데이터 bean이 활성화되자마자 연관된 모든 오브젝트에 대량 자료 반입할 수 있습니다. 그러나 이러한 방식으로 대량 자료 반입할 경우, 여러 번의 데이터베이스 조회가 필요할 수 있습니다. 페이지에서 모든 속성이 다 필요하지는 않은 경우, 여러 번의 데이터베이스 조회는 비능률적일 수 있습니다.

일반적으로 한 페이지에 모든 속성이 필요한 것은 아니므로 더 나은 설계 패턴은 아래에 표시된 대로 lazy fetch를 수행하는 것입니다.

```
getCrossSellProducts () {
   if (crossSellDataBeans == null)
     crossSellDataBeans= getCrossSellDataBeans();
   return crossSellDataBean;
    }
```

JSP 속성 설정 - 개요

WebSphere Commerce 프로그래밍 모델로 MVC 설계 패턴의 효과가 증대됩니다. 이와 같이 URL 요청 결과 표시는 컨트롤러 명령 및 태스크 명령과는 분리됩니다. 이러한 명령은 장치와는 관계 없습니다. 이러한 명령은 클라이언트에 대한 정보없이 비즈니스 로직을 구현하고 클라이언트로 리턴될 데이터를 작성합니다. 반면 뷰 명령은 장치에 따라 고유합니다.

컨트롤러 및 태스크 명령이 직접 뷰를 구성하지는 않지만, 뷰로 정보를 전달합니다. 정보가 뷰로 전달되는 방법을 이해하는 것이 중요합니다. 다음 도표는 웹 컨트롤러, 명령 레지스트리, 컨트롤러 명령 및 뷰 명령 간에 특성이 전달되는 방법을 보여줍니다.

CMREG

인터페이스 이름		특성
com.ibm.xxx. NewCommand		parm1=1&parm2=2

CCPd: parm1=1&parm2=2

VIEWREG

인터페이스 이름		특성
com.ibm.command.ForwardViewCommand		docname=NewView.jsp

VPd: docName=NewView.jsp

URL: http://*hostname*/webapp/wcs/stores/servlet/NewCommand?storeId=1&...

CCPu: storeID=1&...

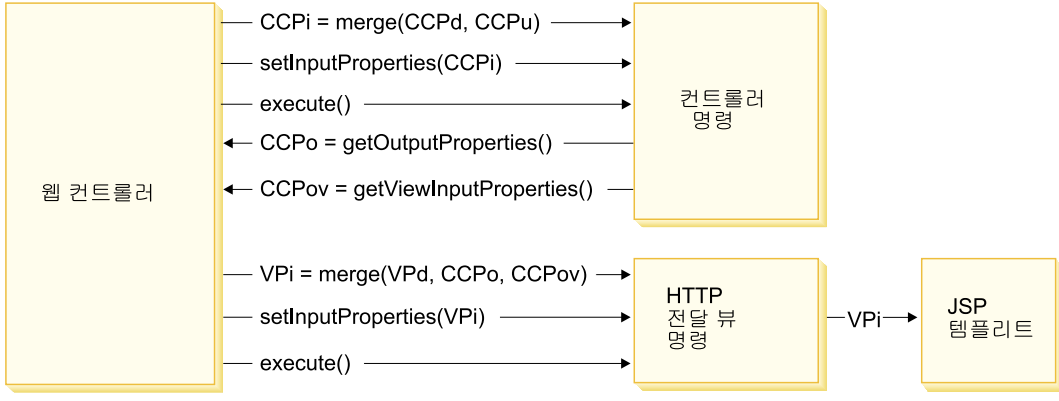


그림 12.

앞의 도표는 다음과 같은 상호 작용을 보여줍니다.

- 웹 컨트롤러는 URL 매개변수의 입력 특성(CCPu)과 컨트롤러 명령의 CMDREG 테이블에 있는 항목(CCPd)을 병합하여 CCPi를 작성합니다.
- 웹 컨트롤러는 병합된 특성(CCPi)을 컨트롤러 명령으로 전달하고 컨트롤러 명령을 실행합니다.

- 컨트롤러 명령은 CCPo로서 출력 특성을 설정합니다. 이러한 특성은 명령 자체에서 발생한 출력 특성입니다. 출력 특성 중 하나인 viewCommandName은 원하는 뷰 명령 이름으로 설정됩니다. 이러한 특성은 get 메소드를 사용하여 웹 컨트롤러에서 검색됩니다.
- 컨트롤러 명령은 CCPov로서 다른 출력 특성 세트를 설정합니다. 기본적으로 이러한 특성은 원래 병합된 입력 특성(CCPi)으로 설정되어 있습니다. 이러한 특성을 사용자 정의할 수 있습니다. 예를 들어, 뷰 명령으로 모든 입력 매개변수를 전달해야할 필요는 없을 수도 있습니다.
- 웹 컨트롤러는 세 가지 특성 세트인 CCPo, CCPov 및 VPd(VIEWREG 테이블에 등록된 특성)를 뷰 명령의 입력 특성(VPi)으로 병합합니다.
- 웹 컨트롤러는 병합된 특성 VPi를 설정하고 뷰 명령을 실행합니다.
- 뷰 명령은 입력 특성의 JSP로 속성을 설정합니다.

새 명령 작성시, 특성 병합을 명시적으로 수행할 필요는 없습니다. 추상 명령 클래스에는 mergeProperties 메소드가 들어 있습니다. 이 메소드에 대한 자세한 정보는 WebSphere Commerce Production 및 Development 온라인 도움말의 “참조” 주제를 참조하십시오.

필수 특성 설정

컨트롤러 명령은 뷰 명령 유형마다 다음과 같은 특성을 설정해야 합니다. 특성이 명령에서 설정되지 않으면 VIEWREG 테이블에 정의되어야 합니다.

- ForwardView 명령을 사용 중이면 `docname = view_file_name`을 설정하십시오. 여기서, *view_file_name*은 표시 템플리트의 이름입니다 (예: `docname=SearchResult.jsp`).
- DirectView 명령을 사용할 경우, 다음 중 하나를 수행하십시오.
  - `textDocument = xxx`를 설정하십시오. 여기서, *xxx*는 텍스트 양식의 문서를 포함하는 java.io.InputStream 오브젝트입니다.
  - `rawDocument = yyy`를 설정하십시오. 여기서, *yyy*는 2진 양식의 문서를 포함하는 java.io.InputStream 오브젝트입니다.

  DirectView 명령을 사용하는 경우 `contentType = ttt` 설정 여부를 선택할 수 있습니다. 여기서, *ttt*는 문서 컨텐츠 유형입니다.

- RedirectView 명령을 사용할 경우, `url = uuu`를 설정하십시오. 여기서, *uuu*는 경로 재지정 URL입니다.

제 3 장 Persistent 오브젝트 모델

WebSphere Commerce는 대용량 지속적 데이터를 처리합니다. 현재 데이터베이스 스키마에는 수많은 테이블이 정의되어 있습니다. 이러한 대규모 스키마를 사용해도 특정 비즈니스 요구에 맞게 데이터베이스 스키마를 확장 또는 사용자 정의해야 합니다.

WebSphere Commerce는 EJB(Enterprise JavaBean) 버전 1.1 구성요소 아키텍처를 기반으로 하는 엔티티 bean을 persistent 오브젝트 계층으로 사용합니다. 이 엔티티 bean은 commerce 도메인에 개념 및 오브젝트를 모델링하는 방법으로 WebSphere Commerce 데이터를 표시합니다. 이 지속성 계층은 확장 가능한 프레임워크를 제공합니다.

WebSphere Studio Application Developer는 이 프레임워크에 대한 개발을 지원하는 EJB 도구 및 단위 테스트 환경을 제공합니다.

다음 절은 EJB 1.1 스펙에서 WebSphere Commerce persistent 오브젝트 모델 구현의 구현 컨텍스트에 나와 있습니다.

WebSphere Commerce 엔티티 bean의 구현

WebSphere Commerce 엔티티 bean - 개요

앞에서 설명한 대로 WebSphere Commerce 아키텍처의 지속성 계층은 EJB 구성요소 아키텍처에 따라 구현됩니다. EJB 아키텍처는 두 가지 유형의 엔터프라이즈 bean(엔티티 bean 및 세션 bean)을 정의합니다. 엔티티 bean은 다시 CMP(Container-Managed Persistence) bean 및 BMP(Bean-Managed Persistence) bean으로 나뉘어집니다.

대부분의 WebSphere Commerce 엔티티 bean은 CMP 엔티티 bean입니다. 적은 수의 stateless 세션 bean을 사용하여 데이터베이스 집약 조작(예: 특정 열에서 모든 행의 합계 수행)을 처리합니다. CMP 엔티티 bean을 사용하는 데 따른 한 가

지 이점은 개발자가 WebSphere Studio Application Developer에서 제공하는 EJB 도구를 활용할 수 있다는 것입니다. 개발자는 이 도구를 사용하여 Java 오브젝트 도구는 엔티티 bean의 필수 지속자를 자동 작성합니다. 지속자는 Java 필드를 데이터베이스로 지속시키고 Java 필드에 데이터베이스 데이터의 대량 자료를 반입하는 Java 오브젝트입니다.

WebSphere Studio Application Developer는 EJB 1.1 스펙에 대한 두 개의 확장자, 즉 EJB 계승 및 연관을 제공합니다. EJB 계승을 사용하면 엔터프라이즈 bean이 동일한 그룹에 상주하는 다른 엔터프라이즈 bean에서 특성, 메소드 및 메소드 레벨 제어 설명자 속성을 상속할 수 있습니다. 연관은 두 CMP 엔티티 bean 간에 존재하는 관계입니다.

몇몇 WebSphere Commerce 엔티티 bean은 EJB 계승 특징을 이용합니다. WebSphere Commerce 엔티티 bean은 WebSphere Studio Application Developer에서 제공하는 연관 특징을 사용하지 않습니다. 사용자 고유 엔티티 bean 개발 시, WebSphere Studio Application Developer의 연관 특징을 사용하지 않을 것을 권장합니다. 이 권장사항은 오브젝트 모델의 복잡도를 최소화하기 위한 것입니다. WebSphere Studio Application Developer에서 제공하는 연관 특징을 사용하지 않고, 대신 엔터프라이즈 bean에 명시적 getter 메소드를 추가하여 엔터프라이즈 bean 사이의 오브젝트 관계를 설정할 수 있습니다.

WebSphere Commerce는 엔터프라이즈 bean의 두 세트, 즉 개인용 및 공개를 제공합니다. 개인용 엔터프라이즈 bean은 WebSphere Commerce 런타임 환경 및 도구에서 사용합니다. 이 bean을 사용 또는 수정해서는 *안 됩니다.*

반면에 공개 엔터프라이즈 bean은 상업용 응용프로그램이 사용하며 사용 및 확장도 가능합니다. 이 공개 엔터프라이즈 bean은 다음 EJB 모듈로 구성됩니다.

- Catalog-ProductManagementData
- Enablement-RelationshipManagementData
- Marketing-CampaignsAndScenarioMarketingData
- Marketing-CustomerProfilingAndSegmentationData
- Member-MemberManagementData
- Merchandising-PromotionsAndDiscountsData

- Order-OrderCaptureData
- Order-OrderManagementData
- Trading-AuctionsAndRFQsData

앞 목록의 일부 EJB 모듈에는 세션 bean이 들어 있습니다. 향후 이주를 쉽게 하려면 세션 bean 클래스를 수정하지 마십시오. 필요한 경우 WebSphereCommerceServerExtensionsData EJB 모듈에 새 세션 bean을 작성할 수 있습니다. 새 세션 bean 작성에 대한 추가 정보는 88 페이지의 『새 세션 bean 작성』을 참조하십시오.

WebSphere Commerce 엔터프라이즈 bean의 전개 설명자

EJB 전개 설명자에는 엔터프라이즈 bean의 전개 설정이 들어 있습니다. WebSphere Studio Application Developer는 이 전개 정보를 수정하는 데 사용할 수 있는 EJB 전개 설명자 편집기를 제공합니다.

새 엔터프라이즈 bean(엔티티 또는 세션 bean) 작성 시, WebSphere Studio Application Developer에서 J2EE perspective의 2EE 계층 보기에 전개 설명자 정보가 설정됩니다. 다음을 수행하여 WebSphere Commerce bean의 EJB 전개 설명자를 볼 수 있습니다.

1. WebSphere Studio Application Developer를 열고 J2EE perspective로 전환하십시오.
2. J2EE 계층 보기를 사용하여 전개 설명자 정보를 볼 EJB 모듈을 찾으십시오.
3. *EJB_moduleName* EJB 모듈을 마우스 오른쪽 버튼으로 누르고 **열기 프로그램 > 전개 설명자 편집기**를 선택하십시오.
   전개 설명자 편집기가 열립니다.
4. bean 탭을 선택하고 다음을 확인하십시오.
   a. bean 목록에서 bean을 선택하십시오. 해당 bean 정보에 대한 대량 자료가 다른 필드에 반입됩니다.
   b. bean은 컨테이너 관리 엔티티 1.x bean으로 표시되어야 합니다.
   c. 재진입 선택란은 선택해서는 안 됩니다.
   d. WebSphere 바인딩 절에서는 JNDI 이름의 기본값을 사용합니다.

e. WebSphere 확장자 절에서는 동시성 제어를 위해 최적 잠금을 사용하지 않습니다.

f. 또한 파인더 섹션에서 bean의 파인더를 볼 수 있음에 유의하십시오.

5. 어셈블리 설명자 탭을 선택하고 다음을 확인하십시오.

   a. 메소드 권한 섹션에서 WCSecurityRole이 엔터프라이즈 bean의 모든 메소드로 지정되어 있습니다.

   b. 컨테이너 트랜잭션 섹션에서는 엔터프라이즈 bean 내 모든 메소드에 “Required”가 지정됩니다.

6. 액세스 탭을 선택하고 다음을 확인하십시오.

   a. 엔티티 1.x에 대한 액세스 목적 절에 모든 read-only 메소드가 정의되어 있습니다. 예를 들어, _copyFromEJB() 및 handcoded getter 메소드는 read-only 메소드로 지정됩니다. 사용자 고유 엔티티 bean을 작성하는 경우, 올바른 메소드를 read-only로 표시하는지 확인하십시오. 이런 방법으로 읽기 전용 메소드가 표시되지 않는 경우, EJB 컨테이너가 트랜잭션 끝에서 불필요하게 데이터베이스를 갱신하려고 시도하여 읽기 전용 트랜잭션에 트랜잭션 롤백 오류가 발생합니다. 이로 인해 성능 문제가 야기됩니다.

   b. 분리 레벨은 다음과 같이 설정됩니다.

      - DB2 반복 가능 읽기
      - Oracle 읽기 확약

WebSphere Commerce 오브젝트 모델 확장

WebSphere Commerce 오브젝트 모델은 다음과 같은 방법으로 확장할 수 있습니다.

- WebSphere Commerce의 공개 엔터프라이즈 bean 확장
- 새 엔티티 bean 쓰기
- 새 stateless 세션 bean 쓰기

이 확장 수행 방법에 대한 정보는 다음 절에서 설명합니다.

오브젝트 모델 확장 방법

응용프로그램 요구사항에 의해 기존 WebSphere Commerce 오브젝트 모델을 확장할 수 있습니다. 이런 요구사항의 한 예는 응용프로그램에 추가 속성을 추가하는 것입니다. 이는 다음 중 한 가지 방법으로 수행될 수 있습니다.

기존 WebSphere Commerce 공개 엔티티 bean을 수정하지 않고

새 데이터베이스 테이블을 작성한 후 해당 테이블에 대한 새 엔티티 bean을 작성하십시오. 엔티티 bean에 필드 및 메소드를 추가하여 필요한 대로 새 속성을 조작하십시오. 새 엔티티 bean에 대한 액세스 bean 및 전개 코드를 작성하십시오. 응용프로그램에 새 속성이 필요한 경우, 액세스 bean 오브젝트의 인스턴스를 작성하고 해당 메소드를 사용하여 속성을 검색, 설정 또는 조작합니다.

기존 WebSphere Commerce 공개 엔티티 bean을 수정하여

새 데이터베이스 테이블을 작성하고, 수정 중인 기존 엔터프라이즈 bean에 대응하는 기존 테이블 및 새 테이블 사이의 테이블 결합을 작성하십시오. 기존 WebSphere Commerce 공개 엔티티 bean에 새 필드를 작성하고, 2차 테이블 맵을 사용하여 새 테이블의 해당 열에 필드를 맵핑하십시오. 필요한 메소드를 추가하십시오. 기존 엔티티 bean에 대한 액세스 bean 및 전개 코드를 다시 작성하십시오. 응용프로그램이 액세스 bean 오브젝트의 인스턴스를 작성하면 새 속성을 사용할 수 있습니다.

이 두 가지 접근 방법에는 장단점이 있습니다. 일반적으로 장단점은 성능 및 코드 유지보수 노력과 연관이 있습니다.

확장 예: 고객의 홈 유형을 사용자가 캡처해야 하는 응용프로그램의 예를 들어 봅니다. 고객 ID 및 거주 유형이 있는 USERRES 테이블을 작성하십시오. 여기서, 거주 유형(resType)은 종신부동산, 콘도미니엄 또는 아파트일 수 있습니다. 이러한 유형의 정보는 인구 통계적 정보로서 기존 Commerce Suite USERDEMO 테이블과 관련이 있습니다. WebSphere Commerce 코드 저장소를 점검하는 경우, Member-MemberManagementData EJB 모듈에 ″Demographics″ 엔터프라이즈 bean이 있는지 찾으십시오. 이 bean에는 USERDEMO 테이블에 저장되어 있는 인구 통계적 정보에 대한 getter 및 setter가 있습니다.

사용자 정의를 수행하는 데는 두 가지 옵션이 있습니다. USERRES 테이블과 상호작용하는 새 엔티티 bean을 작성하거나, Demographics bean에 새 필드(해당 getter 및 setter 메소드 포함)를 추가할 수 있습니다.

첫 번째 접근 방법을 사용하여(전혀 새로운 코드 작성), 새 Userres 엔티티 bean을 작성하고 해당 필드를 USERRES 테이블의 열로 맵핑하십시오. 응용프로그램에서 고객의 거주 유형이 필요한 경우, Userres 액세스 bean 오브젝트의 인스턴스를 작성하고 데이터를 검색해야 합니다. 응용프로그램이 동시에 다른 인구 통계적 정보를 필요로 하는 경우, Demographics 액세스 bean 오브젝트의 인스턴스도 작성하고 다른 필수 속성을 검색해야 합니다. 고객의 전체 인구 통계적 정보 세트를 검색하려고 시도하는 응용프로그램 로직의 임의 부분을 수정하여 원래 액세스 bean 및 새 액세스 bean의 인스턴스를 작성해야 합니다. 다음 도표는 오브젝트 모델 확장을 위한 이 접근 방법을 표시합니다.

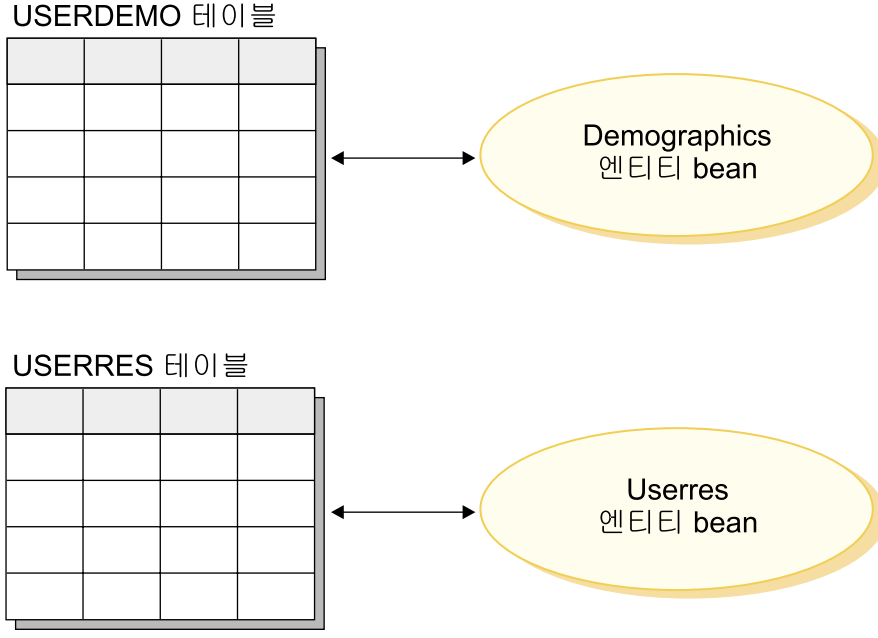


그림 13.

표시 템플리트 측면에서 볼 때, 데이터 bean은 새 속성에 액세스할 수 있어야 합니다. 그래야 JSP 템플리트에 정보를 사용할 수 있습니다. JSP 템플리트를 작성하는 웹 개발자에게 단일 뷰를 제공하려면 원래 기존 엔티티 bean에 대한 액세스

bean을 확장하는 새 데이터 bean을 작성해야 합니다. 데이터 bean은 또한 위임을 사용하여 새 액세스 bean에서 속성을 대량 자료 반입해야 합니다. 다음 도표는 이러한 데이터 bean 구현 시나리오를 보여줍니다.

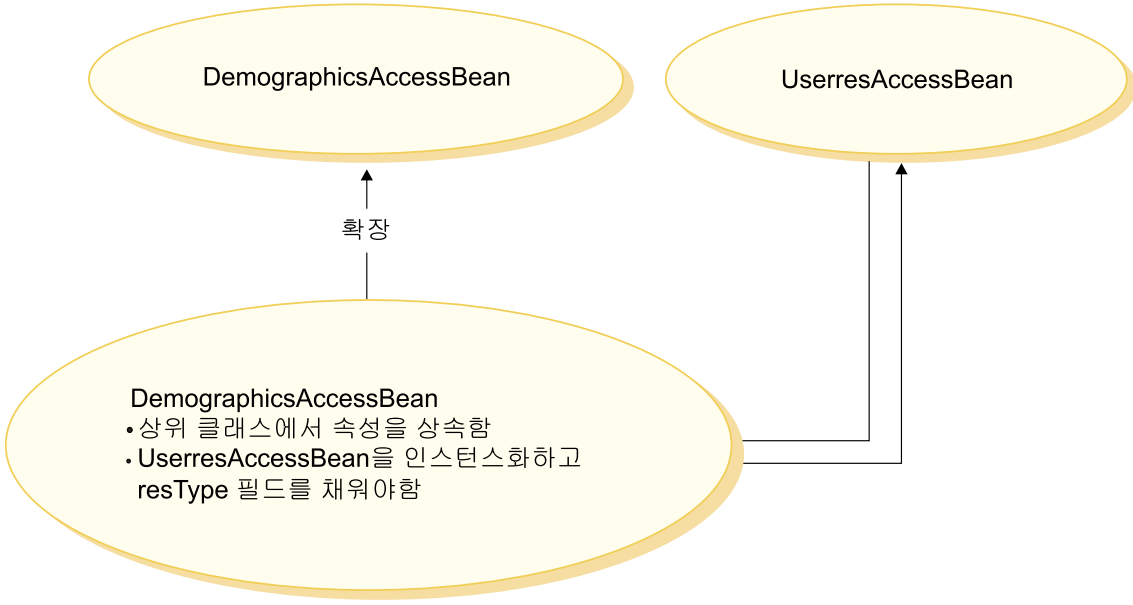


그림 14.

두 번째 접근 방법을 사용하여(기존 코드 수정), Demographics 엔티티 bean에 새 필드를 추가하고 USERRES 테이블의 적절한 열과 새 필드 사이에 2차 테이블 맵을 작성하십시오. 응용프로그램에서 고객의 거주 유형이 필요한 경우, Demographics 액세스 bean 오브젝트의 인스턴스를 작성하고 거주 유형을 검색합니다. 응용프로그램에서 고객에 대한 다른 인구 통계적 정보가 필요한 경우, 동일한 bean을 호출하면 됩니다. 다음 도표는 엔터프라이즈 bean 수정을 위한 이 접근 방법을 표시합니다.

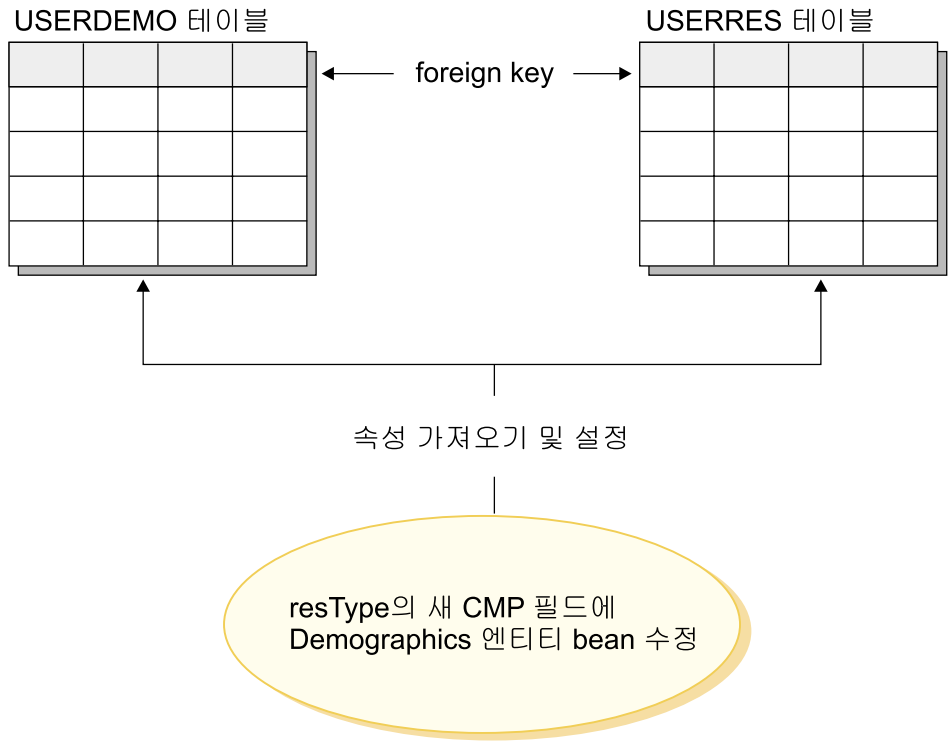


그림 15.

표시 템플리트 측면에서 볼 때, 새 속성(resType)은 DemographicsAccessBean이 다시 작성되자 마자 자동으로 데이터 bean에서 사용할 수 있습니다.

오브젝트 모델을 확장하는 경우, 기존 WebSphere Commerce 데이터베이스 테이블에 새 열을 추가해서는 *안 됨*에 유의하십시오. 새 속성에는 새 테이블을 작성해야 합니다. 기존 테이블에 새 열을 추가하려고 시도하는 경우, 향후 WebSphere Commerce 릴리스로 이주하면 새 속성을 잃게 됩니다.

성능 및 코드 유지보수 관련 의의: 두 번째 접근 방법에서는 런타임 성능이 보다 뛰어납니다. 이는 새 속성을 가져오거나 설정하려면 단일 엔티티 bean의 인스턴스만 작성하면 되고, 단일 가져오기를 사용하여 모든 필수 속성을 검색하기 때문입니다.

두 번째 접근 방법은 기존 WebSphere Commerce 코드를 수정하므로, 새 WebSphere Commerce 버전이 출시되면 이주 문제가 발생합니다. 사용자 정의 코

드와 새 코드를 병합해야 합니다. 그러나 새 WebSphere Commerce 작업 영역을 반입하는 경우 엔터프라이즈 bean에 추가한 필드와 새 테이블 사이의 맵핑 정보는 보존되지 않습니다. 결과적으로 WebSphere Commerce 코드의 새 릴리스로 이주하는 경우 다음 단계를 수행해야 합니다.

1. 사용자 정의 EJB 코드의 버전을 작성하십시오.
2. 새 WebSphere Commerce 코드 버전을 반입하십시오.
3. WebSphere Studio Application Developer의 도구를 사용하여 사용자 정의 코드 버전과 새 WebSphere Commerce 코드 릴리스를 비교하십시오. 사용자 정의 코드를 다시 작업 영역으로 병합하십시오.
4. WebSphere Commerce 공개 엔터프라이즈 bean에 추가한 속성을 데이터베이스 내 해당 열로 다시 수동 맵핑하십시오.
5. 4단계에서 수정한 엔터프라이즈 bean의 액세스 bean 및 전개 코드를 다시 작성하십시오.

이 이주를 보다 간단히 수행하려면 개발 시 오브젝트 모델 확장자를 정확히 기록하는 것이 중요합니다.

오브젝트 모델에 대한 확장자를 많이 작성하는 경우 두 가지 접근 방법을 함께 사용할 수 있습니다. 성능 저하의 가능성이 적은 시스템 영역에는 첫 번째 접근 방법을 사용하고, 성능이 문제가 되는 경우에는 두 번째 접근 방법을 사용할 수 있습니다. 이러한 방법으로 우수한 시스템 성능 레벨을 계속 유지하면서 향후 이주를 위한 노력을 최소화할 수 있습니다.

세션 bean의 권장 용도

WebSphere Commerce의 강점 중 하나는 CMP(Container-Managed Persistence) 엔티티 bean을 이용할 수 있는 기능입니다. CMP 엔티티 bean은 WebSphere Studio Application Developer에서 제공하는 도구로 작성할 수 있는, 지속성 있는 분배 트랜잭션 서버측 Java 구성요소입니다. 많은 경우에 있어 CMP 엔티티 bean은 오브젝트 지속성을 위한 탁월한 선택이며 기타 object-to-relational 맵핑 옵션 또는 이보다 더 효과적으로 작동하도록 작성할 수 있습니다. 이러한 이유로 WebSphere Commerce는 CMP 엔티티 bean을 사용하여 핵심 commerce 오브젝트를 구현했습니다.

그러나 어떤 경우에는 세션 bean JDBC 헬퍼를 사용하는 것이 바람직합니다. 이러한 상황은 다음과 같습니다.

- 조회 시 대량 결과 세트를 리턴하는 경우. 이러한 경우를 *대량 결과 세트* 경우라고 합니다.
- 조회 시 여러 테이블에서 데이터를 검색하는 경우. 이러한 경우를 *총계 엔티티* 경우라고 합니다.
- SQL문이 데이터베이스 집약 조작을 수행하는 경우. 이러한 경우를 *임의 SQL* 경우라고 합니다.

자세한 정보는 다음 절에서 설명합니다.

세션 bean을 JDBC 랩퍼로 사용하여 데이터베이스에서 정보를 검색하는 경우, 자원 레벨 액세스 제어를 구현하기가 보다 어려워짐에 유의하십시오. 이 방법으로 세션 bean을 사용하는 경우, 권한이 없는 사용자가 자원에 액세스하지 못하도록 하려면 세션 bean 개발자가 적절한 "`where`" 절을 "`select`" 문에 추가해야 합니다.

대량 결과 세트 경우: 조회 시 대량 결과 세트를 리턴하고 검색된 데이터가 주로 읽기 또는 표시 용도인 경우가 있습니다. 이런 경우 stateless 세션 bean을 사용하고, 이 세션 bean에서 엔티티 bean의 finder 메소드와 동일한 기능을 수행하는 finder 메소드를 작성하는 것이 보다 바람직합니다. 즉, stateless 세션 bean의 finder 메소드는 다음을 수행해야 합니다.

- SQL select 문 수행™
- 가져오는 각 행에 대해 액세스 bean의 인스턴스 작성
- 검색된 각 열에 대해 액세스 bean에 해당 속성 설정

액세스 bean이 리턴되면 액세스 bean이 세션 bean의 finder 메소드에서 의해 리턴되었는지 또는 엔티티 bean의 finder 메소드에서 리턴되었는지 명령이 알 수 없습니다. 결과적으로 세션 bean의 finder 메소드를 사용해도 프로그래밍 모델을 변경되지 않습니다. calling 명령만이 세션 bean 또는 엔티티 bean의 finder 메소드 호출 여부를 알 수 있습니다. 이 명령은 프로그래밍 모델의 기타 모든 파트에 투명합니다.

총계 엔티티 경우: 이런 경우 여러 오브젝트의 파트를 하나의 뷰로 결합하고, 여러 데이터베이스 테이블의 여러 정보가 단일 표시 페이지에 반입됩니다. 예를 들

어, “내 계정”의 개념을 참조하십시오. 이는 고객 정보 테이블의 정보(예: 고객 이름, 연령 및 고객 ID)와 주소 테이블의 정보(예: 구/군/시 및 동으로 구성된 주소)로 구성될 수 있습니다.

SQL 결합을 수행하여 여러 테이블에서 모든 정보를 검색하는 간단한 SQL문을 작성할 수 있습니다. 이를 “deep fetch” 수행이라고 합니다. 다음은 “내 계정” 예에 대한 SQL select 문의 한 예입니다. 여기서, CUSTOMER 테이블은 T1이고 ADDRESS 테이블은 T2입니다.

```
select T1.NAME, T1.AGE, T2.STREET, T2.CITY
  from CUSTOMER T1, ADDRESS T2
  where (T1.ID=? and T1.ID=T2.ID)
```

EJB 1.1 스펙에서 엔터프라이즈 bean용 WebSphere Studio Application Developer에 있는 도구는 이 deep fetch의 개념을 지원하지 않습니다. 대신 각 연관 오브젝트에 대해 SQL select를 나타내는 lazy fetch를 수행합니다. 이는 이러한 유형의 정보를 검색하는 데 있어 바람직한 방법이 아닙니다.

deep fetch를 수행하려면 세션 bean을 사용하는 것이 바람직합니다. 해당 세션 bean에서 필수 정보를 검색하기 위한 finder 메소드를 작성하십시오. finder 메소드는 다음을 수행해야 합니다.

- deep fetch에 대한 SQL select문 수행
- 각 연관 오브젝트 및 기본 테이블의 각 행에 대해 액세스 bean의 인스턴스 작성
- 가져온 각 열 및 가져온 각 연관 오브젝트에 대해 액세스 bean에 해당 속성 설정

액세스 bean은 예외를 발생시키는 getter 메소드를 캐시하지 않음에 유의하십시오. 이런 경우, 다음 패턴을 사용하여 액세스 bean에 대한 간단한 랩퍼 클래스를 작성해야 합니다.

```
public class CustomerAccessBeanCopy extends CustomerAccessBean {
   private AddressAccessBean address=null;

   /* The following method overrides the getAddress method in
      the CustomerAccessBean.
   */
   public AddressAccessBean getAddress() {
```

```
        if (address == null)
           address = super.getAddress();
        return address;
        }

    /* The following method sets the address to the copy. */

    public void _setAddress(AddressAccessBean aBean) {
       address = aBean;
        }
}
```

CUSTOMER 및 ADDRESS 예를 계속 진행하면서 session bean finder 메소드는 CUSTOMER 테이블의 각 행에 대한 CustomerAccessBean 및 ADDRESS 테이블의 각 해당 행에 대한 AddressAccessBean의 인스턴스를 작성합니다. 그런 다음 ADDRESS 테이블의 각 열에 대해 AddressAccessBean에 속성(구/군/시 및 동)을 설정합니다. ADDRESS 테이블의 각 열에 대해 CustomerAccessBean에 속성(이름, 연령 및 주소)을 설정합니다. 이는 다음 도표에 표시됩니다.

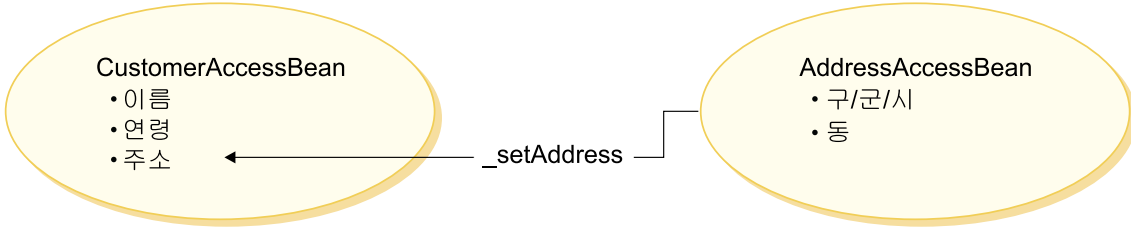


그림 16.

임의 SQL 경우: 이런 경우, 데이터베이스 집약 조작을 수행하는 한 세트의 임의 SQL문이 있습니다. 예를 들어, 한 테이블의 모든 행을 합하는 조작을 데이터베이스 집약 조작이라고 할 수 있습니다. 지속 모델의 경우 선택된 모든 행이 하나의 엔티티 bean에 대응되지 않아도 됩니다.

임의 SQL문을 작성할 수 있는 한 예는 고객이 대용량 데이터 세트를 찾아보려고 시도하는 경우입니다. 예를 들어, 고객이 온라인 공구 상점에서 모든 잠금 도구를 검색하려고 하거나 온라인 의류 상점에서 모든 드레스를 검색하려고 하는 경우입니다. 이는 대용량 결과 세트를 작성하지만 이 결과 세트의 경우 각 행의 몇몇 필드만 필요한 경우가 많습니다. 즉 처음에는 고객에게 항목 이름, 그림 및 가격을 표시하는 요약만이 표시될 수 있습니다.

이런 경우, session bean helper 메소드를 작성하십시오. 이 session bean helper 메소드는 읽기 또는 쓰기 조작을 수행합니다. 읽기 조작을 수행하면 표시 용도로 사용되는 읽기 전용 값 오브젝트를 리턴합니다.

일반적으로 올바른 데이터 모델링을 사용하면 임의 SQL문의 빈도 수를 최소화할 수 있습니다.

공개 엔티티 bean 확장

이 절에서는 WebSphere Commerce 공개 엔티티 bean의 설계 패턴에 대해 설명합니다. 이 설계 패턴을 사용하면 새 지속 필드, 새 비즈니스 메소드 또는 새 파인더 메소드 추가와 같은 확장을 수행할 수 있습니다.

다음 도표는 Catalog 엔티티 bean의 구현 클래스를 표시합니다.

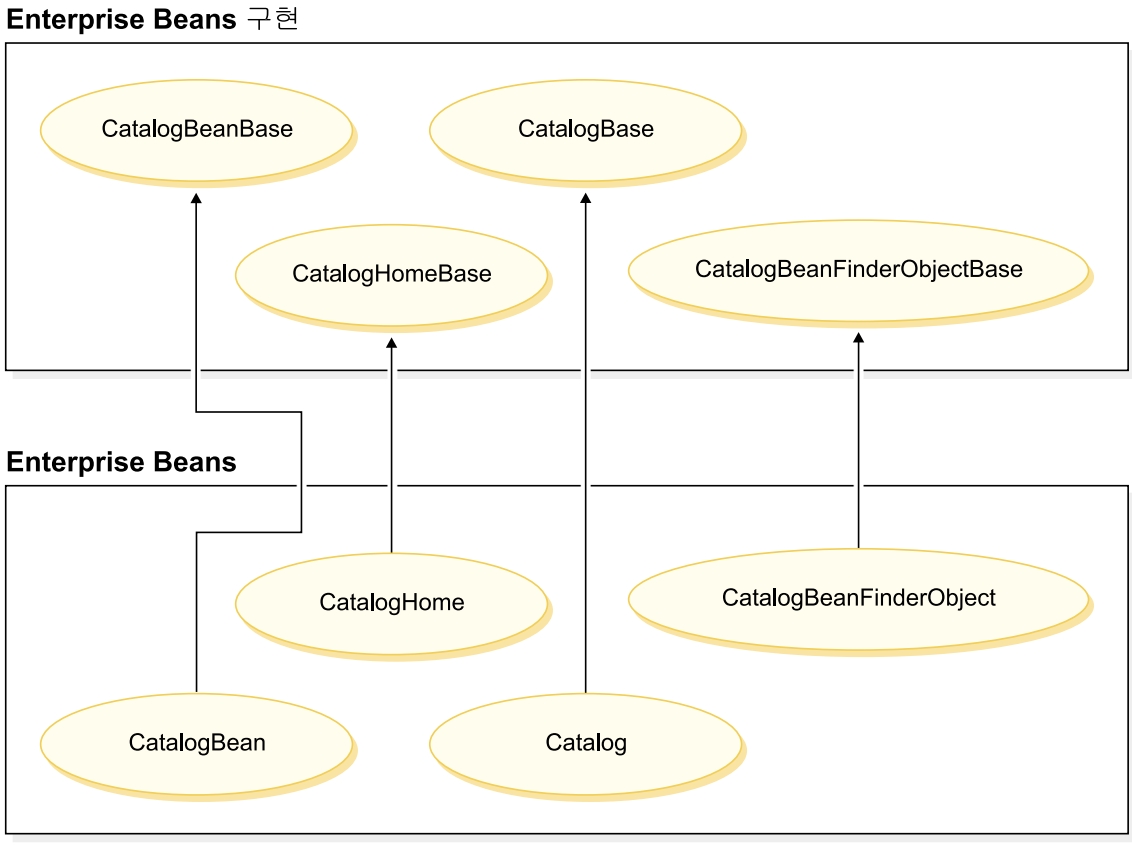


그림 17.

앞의 도표는 기타 엔티티 bean에도 적용됩니다. 이 bean 또한 유사한 방식으로 구성되고 동일한 이름 지정 규칙을 따르기 때문입니다. 이 도표를 다른 엔티티 bean에 적용하려면 엔티티 bean의 이름을 “Catalog”로 대체하십시오. 예를 들어, InterestItemBean 클래스는 InterestItemBeanBase 클래스를 확장하고, InterestItem 인터페이스는 InterestItemBase 인터페이스를 확장합니다.

도표에서는 Java 계승을 사용하여 공개 엔터프라이즈 bean의 구현 클래스 또는 인터페이스가 두 파트로 분리된 것으로 표시합니다. 상위 클래스 또는 인터페이스에는 WebSphere Commerce 구현 코드가 있습니다. 이 모든 상위 클래스와 인터페이스는 하위 클래스 및 인터페이스와는 별도로 Java 패키지에 정의됩니다.

WebSphere Commerce 작업 영역에는 이 모든 상위 클래스 및 인터페이스의 2진 코드가 들어 있습니다. 하위 클래스 및 인터페이스는 수정할 수 있습니다. 일반적으로 com.ibm.commerce.*xxx*.objects 및 com.ibm.commerce.*xxx*.objsrc 패키지는 수정할 수 있습니다. 여기서, *xxx*는 구성요소 이름입니다.

공개 엔터프라이즈 bean에 새 파인더 메소드를 추가하는 경우, 메소드에 대한 특정 이름 지정 규칙을 준수해야 합니다. 새 메소드의 이름을 `findX`*`a_description`* 으로 지정하십시오. 여기서, *`a_description`*은 원하는 설명입니다. 몇몇 이름 예는 `findXByOwnerId` 및 `findXByOrderStatus`입니다. 이 이름 지정 규칙을 사용하면 WebSphere Commerce 파인더 메소드와의 이름 충돌(이름 중복) 위험을 막을 수 있습니다. 새 파인더 추가 시 전개 설명자 편집기를 사용합니다.

기존 WebSphere Commerce 공개 엔티티 bean을 수정하기 위한 한 가지 방법은 추가 필드를 추가하는 것입니다. 이 경우, 새 필드를 추가한 후 bean의 각 파인더 메소드를 점검해야 합니다. 파인더 메소드의 `where` 절 부분에 데이터 별명(예: T1. 또는 T2.)이 있는 경우, 별명을 제거해야 합니다.

"findForUpdate" 유형의 파인더가 있는 공개 엔티티 bean: WebSphere Commerce 공개 엔티티 bean에 "findForUpdate" 유형의 파인더가 있는 경우, 작성한 새 테이블에 2차 맵을 작성하여 bean에 새 필드를 추가할 수 없습니다. 이것은 2차 맵을 작성하는 경우 작성된 SQL문이 올바르지 않아서 bean이 더 이상 원하는 기능을 수행하지 않기 때문입니다. 그러한 bean으로 표시되는 오브젝트 모델 파트를 확장하려는 경우, 새 엔티티 bean을 작성하여 원래 bean 및 새 bean을 사용자 정의 코드에 함께 사용해야 합니다.

새 CMP 엔터프라이즈 bean 작성

WebSphere Commerce 오브젝트 모델에 추가해야 하는 새 속성이 있는 경우, 필요한 속성의 열이 있는 새 데이터베이스 테이블을 작성할 수 있습니다. 그런 다음 엔터프라이즈 bean에 이 속성을 포함시켜 WebSphere Commerce 명령이 정보에 액세스할 수 있도록 해야 합니다.

WebSphere Commerce 오브젝트 모델로 새 속성을 통합하기 위한 한 가지 방법은 새 CMP 엔터프라이즈 bean을 작성하는 것입니다. 새 데이터베이스 테이블의 속성에 해당하는 필드를 이 bean에서 작성하십시오.

WebSphere Commerce 작업 영역에서 새 엔터프라이즈 bean에 대해 사전 정의된 EJB 프로젝트를 제공하므로, 추가 EJB 프로젝트를 작성하지 않아도 됩니다. 새 엔터프라이즈 bean은 WebSphereCommerceServerExtensionsData EJB 프로젝트에 배치해야 합니다. 나중에 사용자 정의 bean을 전개할 때 WebSphereCommerceServerExtensionsData.jar JAR 파일을 작성하고, WebSphere Application Server에서 실행되는 WebSphere Commerce 엔터프라이즈 응용프로그램에서 기존 JAR 파일을 바꾸십시오. 이 패키지 규칙을 사용하면 전개가 매우 간편해집니다.

새 CMP 엔터프라이즈 bean을 작성하려면 WebSphere Studio Application Developer에서 다음 단계를 수행해야 합니다.

1. 엔터프라이즈 bean 작성 마법사를 사용하여 새 CMP 엔터프라이즈 bean을 작성하십시오. 해당 데이터베이스 테이블의 각 열에 대해, 새 CMP 필드를 bean에 추가하십시오.
2. 새 bean의 트랜잭션 분리 레벨을 설정하십시오.
3. 새 bean의 보안 동일성을 설정하십시오.
4. 엔티티 컨텍스트 메소드를 수정하십시오.
5. 필요한 경우, EJB 전개 설명자 편집기를 사용하여 새 파인더를 정의하십시오.
6. 필요한 경우 새 ejbCreate 메소드를 작성하고, ejbCreate 메소드를 엔터프라이즈 bean의 홈 인터페이스로 승격시키십시오. 이 단계는 새 엔터프라이즈 bean이 해당 데이터베이스 테이블에 새 항목을 작성해야 하는 경우 필요합니다.
7. WebSphere Commerce 액세스 제어 시스템으로 bean을 보호하는 경우, bean에서 필수 액세스 제어 메소드를 구현하십시오. 엔터프라이즈 bean에서의 액세스 제어 구현에 대한 자세한 정보는 105 페이지의 제 4 장『액세스 제어』를 참조하십시오. 또는 액세스 bean을 작성한 후 액세스 제어를 구현할 수 있습니다.
8. 엔터프라이즈 bean의 필드를 데이터베이스 테이블 열에 맵핑하십시오.
9. 엔터프라이즈 bean의 해당 액세스 bean을 작성하십시오.
10. 엔터프라이즈 bean의 전개 코드를 작성하십시오.

11. WebSphere Studio Application Developer의 Universal Test Client를 사용하여 bean을 테스트하십시오.

이들 각 단계에 대한 자세한 정보는 다음 절에서 제공합니다. 이 절의 내용을 읽을 때는 사용자의 주거 유형에 대한 몇 가지 정보를 지정하는 XUSERRES라는 새 테이블이 있는 것으로 가정하십시오. 이 테이블에는 세 가지 열, 즉 USERID 열, 주택 유형을 지정하는 HOME 열 및 거주지의 침실 수를 지정하는 ROOMS 열이 있습니다.

새 CMP 엔터프라이즈 bean 작성: 새 CMP 엔터프라이즈 bean을 작성하려면 다음과 같이 엔터프라이즈 Bean 작성 마법사를 사용할 수 있습니다.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼치십시오.
2. **WebSphereCommerceServerExtensionsData** 모듈을 마우스 오른쪽 버튼으로 누르고 **새로 만들기 > (기타 >) 엔터프라이즈 bean**을 선택하십시오. 엔터프라이즈 bean 작성 마법사가 열립니다.
3. **EJB 프로젝트** 드롭 다운 목록에서 **WebSphereCommerceServerExtensionsData**를 선택하고 **다음**을 누르십시오.
4. 엔터프라이즈 bean 작성 창에서 다음을 수행하십시오.
   a. **CMP(container-managed persistence) 필드가 있는 엔티티 bean**을 선택하십시오.
   b. **bean 이름** 필드에 적절한 bean 이름을 입력하십시오. 일반적으로 bean 이름은 해당 데이터베이스 테이블의 이름과 일치합니다. 예를 들어, XUserRes bean의 이름은 XUSERRES 테이블과 일치합니다.
   c. **소스 폴더** 필드에서는 지정된 기본값(ejbModule)을 그대로 두십시오.
   d. **기본 패키지** 필드에 `com.mycompany.mycomponent.objects`를 입력하십시오.
   e. **다음**을 누르십시오.
5. 엔터프라이즈 bean 정보 창에서 다음을 수행하십시오.
   a. **추가**를 눌러 열의 새 CMP 속성을 데이터베이스 테이블에 추가하십시오. CMP 속성 작성 창이 열립니다. 이 창에서 다음을 수행하십시오.

1) **이름** 필드에 적절한 새 CMP 필드 이름을 입력하십시오. 나중에 이 필드를 데이터베이스 테이블 내 해당 열로 맵핑시킬 때 일치를 이름 기능으로 사용하려는 경우, 필드 이름을 열 이름과 똑같이 지정하십시오 (대소문자 미구분).
2) **유형** 필드에, 필드에 적절한 데이터 유형을 입력하십시오. 원시 데이터 유형의 랩퍼 클래스를 사용해야 함에 유의하십시오. 예를 들어, *long* 데이터 유형이 아닌 *java.lang.Long* 데이터 유형을 사용하십시오.
3) 필드가 1차 키인 경우, **키 필드** 선택란을 선택하고 **적용**을 누르십시오.
4) 필드가 1차 키가 아닌 경우, **getter 및 setter 메소드로 액세스** 선택란을 선택하십시오.
5) 필드가 1차 키가 아닌 경우, **getter 및 setter 메소드를 원격 인터페이스로 승격** 선택란을 지우십시오. **getter를 읽기 전용으로 만들기** 선택란을 사용할 수 없게 됩니다. 그런 다음 **적용**을 누르십시오.
6) 다시 **추가**를 누르고 단계를 반복하여 CMP 필드가 필요한 데이터베이스 테이블의 각 열에 새 필드를 추가하십시오.
7) **닫기**를 눌러 창을 닫으십시오.

b. **키 클래스에 단일 키 속성 유형 사용** 선택란을 지우고 **다음**을 누르십시오.

6. EJB Java 클래스 정보 창에서 다음을 수행하십시오.
   a. Bean의 상위 클래스를 선택하려면 **찾아보기**를 누르십시오.
   유형 선택사항 창이 열립니다.
   b. **사용하는 클래스 선택: (임의)** 필드에 `ECEntityBean`을 입력하고 **확인**을 누르십시오. 그러면 상위 클래스로 com.ibm.commerce.base.objects.ECEntityBean이 선택됩니다.
   c. WebSphere Commerce 액세스 제어 프레임워크로 새 엔터프라이즈 bean을 보호하려는 경우, **추가**를 눌러 원격 인터페이스가 확장해야 하는 인터페이스를 지정하십시오. 유형 선택 창이 열립니다.
   d. **사용하는 클래스 선택: (임의)** 필드에 `Protectable`을 입력하고 **확인**을 누르십시오. 그러면 com.ibm.commerce.security.Protectable이 선택됩니다. 이 인터페이스는 액세스 제어 하에 새 자원을 보호하기 위해 필요합니다.
   e. **완료**를 누릅니다.

WebSphere Studio Application Developer는 bean 클래스에 개인용 필드 EntityContext를 작성합니다. WebSphere Commerce는 ECEntityBean에 고유한 엔티티 컨텍스트 필드를 제공하며 새 엔티티 bean은 작성된 필드가 아닌 이 필드를 사용해야 합니다. 따라서 새 엔티티 bean에서 작성된 EntityContext를 제거해야 합니다. 이는 후속 절에서 설명됩니다.

상위 클래스로 com.ibm.commerce.base.objects.ECEntityBean을 지정하면 사용자 bean은 특정 함수를 상속합니다. 다음 코드 예는 이러한 함수를 표시합니다.

```
public class myEJB extends com.ibm.commerce.base.objects.ECEntityBean {
   public void ejbLoad() {
      super.ejbLoad();--the super method will add EJB trace
       --your logic --
   }
   public void ejbStore() {
      super.ejbStore();--the super method will add EJB trace
      --your logic --
   }
}
```

트랜잭션 분리 레벨 설정: bean의 트랜잭션 분리 레벨을 개발 데이터베이스 유형의 올바른 값으로 설정해야 합니다. 트랜잭션 분리 레벨을 설정하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼치십시오.
2. **WebSphereCommerceServerExtensionsData** 프로젝트를 두 번 눌러 전개 설명자 편집기에서 여십시오.
3. **액세스** 탭을 누르십시오.
4. 분리 레벨 텍스트 상자 옆에 있는 **추가**를 누르십시오.
   분리 레벨 추가 창이 열립니다.
5. DB2 **반복 가능 읽기**를 선택한 후 **다음**을 누르십시오.
   Oracle **읽기 확약**을 선택한 후 **다음**을 누르십시오.
6. **발견된 bean** 목록에서 *yourNewBean* bean을 선택한 후 **다음**을 누르십시오.
7. **발견된 메소드** 목록에서 *yourNewBean*을 선택하여 해당되는 모든 메소드를 선택하고 **완료**를 누르십시오.
8. 작업을 저장하고(Ctrl+S) 편집기는 열린 상태로 두십시오.

bean의 보안 동일성 설정: 다음을 수행하여 bean의 보안 동일성을 설정하십시오.

1. 전개 설명자 편집기에서 액세스 탭이 선택되어 있는지 확인하십시오.
2. 보안 동일성 텍스트 상자 옆에 있는 **추가**를 누르십시오.
   보안 동일성 추가 창이 열립니다.
3. **EJB 서버의 동일성 사용**을 선택한 후 **다음**을 누르십시오.
4. **발견된 bean** 목록에서 *yourNewBean* bean을 선택한 후 **다음**을 누르십시오.
5. **발견된 메소드** 목록에서 *yourNewBean*을 선택하여 해당되는 모든 메소드를 선택하고 **완료**를 누르십시오.
6. 작업을 저장하십시오(Ctrl+S). 편집기를 열린 상태로 두십시오.

bean의 보안 역할 설정: 이번에는 다음을 수행하여 bean의 메소드에 대해 보안 역할을 설정하십시오.

1. 전개 설명자 편집기에서 어셈블리 설명자 탭을 선택하십시오.
2. 메소드 권한 절에서 **추가**를 누르십시오.
3. 보안 역할로 **WCSecurityRole**을 선택하고 **다음**을 누르십시오.
4. 발견된 bean 목록에서 *yourNewBean*을 선택하고 **다음**을 누르십시오.
5. 메소드 요소 페이지에서 **모두에 적용**을 누른 후 **완료**를 누르십시오.
6. 작업을 저장한 다음 전개 설명자 편집기를 닫으십시오.

엔티티 컨텍스트 필드 및 메소드 삭제: 다음 단계는 WebSphere Studio Application Developer가 작성하는 엔티티 컨텍스트에 관련된 필드 및 메소드 중 일부를 제거하는 것입니다. 필드를 삭제해야 하는 이유는 ECEntityBean 기본 클래스가 해당 메소드의 구현을 제공하기 때문입니다. 작성된 엔티티 컨텍스트 필드와 메소드를 삭제하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **WebSphereCommerceServerExtensionsData** 프로젝트를 펼치십시오.
2. *yourNewBean* bean을 펼친 후 *yourNewBean***bean** 클래스를 두 번 누르십시오.
3. 아웃라인 보기에서 다음을 수행하십시오.

a. **myEntityCtx** 필드를 마우스 오른쪽 버튼으로 누르고 **삭제**를 선택하십시오.

b. **getEntityContext()** 메소드를 마우스 오른쪽 버튼으로 누르고 **삭제**를 선택하십시오.

c. **setEntityContext(EntityContext)** 메소드를 마우스 오른쪽 버튼으로 누르고 **삭제**를 선택하십시오.

d. **unsetEntityContext()** 메소드를 마우스 오른쪽 버튼으로 누르고 **삭제**를 선택하십시오.

4. 작업을 저장하십시오(Ctrl+S).

새 파인더 추가:

파인더 작업 소개: 파인더 헬퍼 인터페이스는 사용하지 않습니다. 새 개발 작업에서는 파인더 헬퍼 인터페이스 대신 EJB 전개 설명자 편집기를 사용하여 조회 및 메소드 선언을 정의해야 합니다.

파인더 작성을 위한 EJB 조회 언어 사용 및 파인더에 대한 다른 접근 방법에 비해 이 접근 방법의 이점에 대한 정보와 관련 도구에 대한 자세한 정보는 WebSphere Studio Application Developer 온라인 도움말을 참조하십시오.

새 bean에 새 파인더 추가: 엔터프라이즈 bean에 새 파인더를 추가해야 하는 경우 다음을 수행하십시오.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼치십시오.
2. **WebSphereCommerceServerExtensionsData** 프로젝트를 두 번 누르고 EJB 전개 설명자 편집기를 여십시오.
3. **Bean** 탭을 누르십시오.
4. bean 분할창에서 *yourNewBean* bean을 선택한 후 오른쪽 분할창에서 아래로 화면이동하여 **WebSphere Extension**을 펼치십시오.
5. **파인더** 텍스트 상자 옆에 있는 **추가**를 누르십시오.
   파인더 설명자 추가 창이 열립니다.

6. **새로 만들기**를 선택한 후 **이름** 필드에 `findByX`*yourArg*를 입력하십시오(여기서, *yourArg*는 검색 기준 인수의 이름). 필드 이름에 "findXBy" 이름 지정 규칙을 사용하여 필드 이름이 언제나 WebSphere Commerce 필드 이름에서 고유한지 확인하십시오.
7. **매개변수** 텍스트 상자 옆에 있는 **추가**를 누르고 다음을 수행하십시오.
   a. **이름** 필드에 *yourArg*를 입력하십시오.
   b. **유형** 필드에 적절한 데이터 유형을 입력하십시오.
   c. **확인**을 누르십시오.
8. **리턴 유형** 필드에 다음 중 하나를 입력하고 **다음**을 누르십시오.
   - FinderHelper 메소드가 1차 키를 사용하여 데이터베이스를 조회하고 메소드가 고유 레코드를 리턴해야 하는 경우, EJB 오브젝트를 리턴 유형으로 지정하십시오. 예를 들어, `UserRes`를 입력하십시오.
   - FinderHelper 메소드가 고유 레코드 대신 결과 세트를 리턴하는 경우, 리턴 유형을 `java.util.Enumeration`으로 지정하십시오.
9. **파인더 유형** 드롭 다운 목록에서 **WhereClauseFinderDescriptor**를 선택하십시오.
10. **파인더 명령문** 필드에서 적합한 파인더, 예를 들어, `T1.MEMBERID = ?`를 입력한 다음 **완료**를 누르십시오.
11. 작업을 저장하고 EJB 전개 설명자 편집기를 닫으십시오.

보안 이유로, 새 엔티티 bean에 대한 FinderHelper 메소드 작성 시 앞 단계에 표시된 대로 매개변수 삽입을 사용해야 합니다. 이를 권장하는 이유는 사용자가 조회를 변경하지 못하도록 보호하기 위해서입니다. 다른 접근 방법은 다음과 유사한 생성자를 사용하는 것입니다.

```
T1.MEMBERID = "input_string ";
```

여기서, *input_string*은 URL에서 전달된 문자열 값입니다. 이는 악의적 사용자가 SQL문을 변경하는 "'123' OR 1=1"과 같은 값을 입력할 수 있으므로 바람직하지 않습니다. 사용자가 SQL문을 변경할 수 있는 경우, 권한 없이도 데이터에 액세스할 수 있습니다. 따라서 권장되는 접근 방법은 매개변수 삽입을 사용하는 것입니다.

매개변수 삽입을 사용하지 못해 입력 문자열을 사용하여 SQL문을 작성해야 하는 경우, 입력 문자열에 대한 매개변수 확인을 강제 실행하여, 입력 매개변수가 데이터 액세스를 위한 악의적 시도가 아닌지 확인해야 합니다.

새 ejbCreate 메소드 작성: 엔터프라이즈 bean을 작성하면 ejbCreate 메소드가 자동 작성됩니다. 그러면 원격 인터페이스에 이 메소드가 프롬프트되어 액세스 bean에서 사용할 수 있게 됩니다. 기본 ejbCreate 메소드에는 1차 키 또는 1차 키의 일부인 매개변수만 있습니다. 이는 인스턴스 작성 시 해당 값의 인스턴스만 작성됨을 의미합니다.

1차 키의 일부가 아니며 널(Null)을 입력할 수 없는 필드가 엔터프라이즈 bean에 있는 경우, 해당 필드의 인스턴스를 작성하는 새 ejbCreate 메소드를 작성해야 합니다. 이를 통해, 새 레코드를 작성할 때마다 널(Null)을 입력할 수 없는 필드에 해당 데이터를 대량으로 반입합니다.

새 ejbCreate 메소드를 작성하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 ***yourNewBean*Bean** 클래스를 두 번 눌러 열고 해당 소스 코드를 보십시오.
2. 널(Null)을 입력할 수 없는 각 CMP 필드가 메소드에 대한 입력 필드로 포함되도록 소스 코드를 수정해야 하며 해당 값으로 각 CMP 필드의 인스턴스가 작성됩니다. UserId가 1차 키인 UserRes 예의 경우, 소스 코드는 처음에 다음과 같이 나타납니다.

   ```
   public void ejbCreate(int argUserId)
      throws javax.ejb.CreateException {
      _initLinks();
      userId = argUserId;
   }
   ```

   그러나 방의 수 및 자택 유형을 모두 초기화했는지 확인하려 할 수 있습니다. 이 경우, 코드를 다음과 같이 변경합니다.

   ```
   public void ejbCreate(int argUserId, String argHome, byte Rooms)
      throws javax.ejb.CreateException {
      _initLinks();
      // All CMP fields should be initialized here
   ```

```
    userId = argUserId;
    home = argHome;
    rooms = argRooms;
}
```

주: 시스템 작성 1차 키를 사용하려는 경우, 93 페이지의 『1차 키』의 정보를 참조하십시오.

3. 새 ejbCreate 메소드를 홈 인터페이스에 추가해야 합니다. 그러면 메소드가 작성된 액세스 bean에서 사용할 수 있게 됩니다. 홈 인터페이스에 메소드를 추가하려면 다음을 수행하십시오.
   a. 아웃라인 보기의 **ejbCreate(***yourParameters***)** 메소드를 마우스 오른쪽 버튼으로 누르고 **엔터프라이즈 bean > 홈 인터페이스로 승격**을 선택하십시오.

새 ejbPostCreate 메소드 작성: 다음으로 새 ejbCreate 메소드와 입력 매개변수가 동일한 새 ejbPostCreate 메소드를 작성해야 합니다. 이 새 메소드를 작성하려면 다음을 수행하십시오.

1. ***yourNewBean*Bean** 클래스를 두 번 눌러 열고 해당 소스 코드를 보십시오.
2. 새 ejbCreate 메소드와 입력 매개변수가 동일한 새 ejbPostCreate 메소드를 작성하십시오. 사용자 거주 예를 계속 진행하려면 다음 코드를 클래스에 추가합니다.

```
public void ejbPostCreate(int argUserId,
   String argHome, byte Rooms)
   {
   }
```

코드 변경사항을 저장하십시오.

bean에 액세스 제어 메소드 추가: 액세스 제어로 새 bean을 보호하려는 경우, getOwner 메소드를 추가해야 합니다. 액세스 제어 용도로 선택적인 또 다른 메소드는 fulfills 메소드입니다. 필수 및 선택 메소드에 대한 자세한 정보는 125 페이지의 『엔터프라이즈 bean에서 액세스 제어 구현』을 참조하십시오. 새 bean에 액세스 제어 메소드를 추가하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 ***yourNewBean*Bean** 클래스를 두 번 눌러 열고 해당 소스 코드를 보십시오.

2. 이 자원의 소유자를 리턴하기 위한 로직을 포함하는 getOwner 메소드를 소스 코드에 추가하십시오. 예를 들어, UserRes bean의 소유자를 리턴하려면 다음과 같이 사용자의 구성원 ID를 리턴합니다.

```
public java.lang.Long getOwner()
   throws java.lang.Exception {
      return getMemberId();
   }
```

3. 이 예의 용도로, 이 자원에 대한 조치를 수행하려면 사용자가 충족시켜야 하는 관계를 지정하는 fulfills 메소드를 추가합니다. 이런 경우, 이 UserRes 오브젝트의 작성자만 조치를 수행할 수 있도록 지정해야 합니다. 즉, 각 사용자만 해당 UserRes 오브젝트에 대한 조치를 수행할 수 있습니다. 이 관계 요구사항은 다음 코드 단편에 표시됩니다.

```
public boolean fulfills(Long member, String relationship)
   throws java.lang.Exception {
      if (relationship.equalsIgnoreCase("creator")) {
         return member.equals(getMemberId());
      }
      return false;
   }
```

4. 작업을 저장하십시오.

새 엔터프라이즈 bean에 데이터베이스 테이블 맵핑: 새 엔터프라이즈 bean을 작성하면 bean의 CMP 필드와 데이터베이스 테이블의 열 사이에 맵핑을 작성해야 합니다. 엔터프라이즈 bean 및 해당 데이터베이스 테이블이 모두 있는 경우, “Meet-in-the-middle” 유형의 맵핑이 사용됩니다. WebSphere Studio Application Developer는 이 태스크를 단순화하기 위한 도구를 제공합니다.

맵핑을 작성하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **WebSphereCommerceServerExtensionsData**에서 마우스 오른쪽 버튼을 누르고 **작성 > EJB에서 RDB로 맵핑**을 선택하십시오. EJB에서 RDB로 맵핑 창이 열립니다.
2. **중간 맞춤**을 선택하고 **다음을** 누르십시오.
3. 데이터베이스 연결 창에서 다음을 수행하십시오.
   a. **연결 이름** 필드에 WebSphereCommerceServerExtensionsData를 입력하십시오.

b. **데이터베이스 필드**에 *developmentDB*를 입력하십시오.

c. **사용자 ID** 필드에 *dbuser*를 입력하십시오.

d. **암호** 필드에 *dbpassword*를 입력하십시오.

e. 데이터베이스 드롭 다운 목록에서 개발 데이터베이스의 데이터베이스 공급업체 유형을 선택하십시오.

- DB2 DB2 Universal Database 8.1
- Oracle Oracle 9i

f. Oracle **호스트** 필드에 데이터베이스 서버의 완전한 호스트 이름을 입력하십시오. 예를 들어, dbserver.yourcompany.com을 입력하십시오.

g. Oracle 클래스 위치 필드에서 classes12.zip 파일의 위치를 입력하십시오. 예를 들어, D:\oracle\ora92\jdbc\lib\classes12.zip을 입력하십시오.

h. **다음**을 누르십시오. 연결되면 데이터베이스의 테이블 목록이 표시됩니다. 데이터 perspective의 데이터베이스 서버 보기를 참조하여 나중에 연결 문서를 볼 수도 있습니다.

4. ***yourNewTable*** 테이블을 선택하고 **다음**을 누르십시오.
5. **이름 및 유형 기준 일치**를 선택한 후 **완료**를 누르십시오. 맵핑 편집기가 열립니다.
6. Oracle 열의 데이터 유형이 “NUMBER”인 경우, 데이터 유형을 수정해야 합니다. ***yourNewTable*** 테이블을 마우스 오른쪽 버튼으로 누르고 **테이블 편집기 열기**를 선택하십시오. 테이블 편집기에서 다음을 수행하십시오.
   a. **열** 탭을 선택하십시오.
   b. 데이터 유형을 변경해야 하는 열을 선택하고 열 유형을 NUMBER에서 보다 구체적인 유형으로 변경하십시오. 예를 들어, INTEGER로 변경하십시오.
   c. 변경사항을 저장하십시오.
7. 엔터프라이즈 bean 분할창에서 ***yourNewBean*** bean을 펼치십시오. 테이블 분할창에서 ***yourNewTable*** 테이블을 펼치십시오.
8. 다음을 수행하여 *yourNewBean* bean의 필드를 *yourNewTable* 테이블의 열로 맵핑하십시오.

a. ***yourNewBean*** bean을 마우스 오른쪽 버튼으로 누르고 **이름 기준 일치**를 선택하십시오.

9. Map.mapxmi 파일의 변경사항를 저장한 후 파일을 닫으십시오.
10. Oracle 다음과 같이 텍스트 편집기를 사용하여 테이블 정의를 편집해야 합니다.
    a. 텍스트 편집기에서 *yourNewBean*.xmi 파일을 여십시오.
    b. 모든 SQLNumeric6 어커런스를 SQLNumeric3으로 바꾸십시오.
    c. 변경사항을 저장한 후 파일을 닫으십시오.

스키마 이름 수정: 다음 단계는 bean을 다른 데이터베이스에 이식할 수 있도록 스키마 이름을 수정하는 것입니다. 이 방법으로 bean을 이식할 수 있도록 하는 특별 값은 NULLID입니다. 스키마 이름을 수정하려면 다음을 수행하십시오.

1. J2EE Perspective에서 J2EE 계층 보기로 전환하십시오.
2. **데이터베이스**를 펼친 후 **WebSphereCommerceServerExtensionsData**를 펼치십시오.
3. 스키마 노드(예: db2user)를 마우스 오른쪽 버튼으로 누르고 **이름 바꾸기**를 선택하십시오.
4. 값을 NULLID로 설정하십시오.

액세스 bean 작성: 액세스 bean은 기타 구성요소가 엔터프라이즈 bean과 상호작용하는 방법을 단순화하는 엔터프라이즈 bean의 랩퍼와 같은 기능을 수행합니다. 새 엔터프라이즈 bean에 대한 액세스 bean을 작성해야 합니다. WebSphere Studio Application Developer의 도구는 이미 작성한 엔티티에 기초하여 액세스 bean을 작성하는 데 사용됩니다(특히 액세스 bean은 원격 인터페이스로 승격된 메소드만 사용합니다).

액세스 bean을 작성하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼친 후 **WebSphereCommerceServerExtensionsData**에서 마우스 오른쪽 버튼을 누르고 **새로 만들기 > 액세스 bean**을 선택하십시오.
   액세스 bean 추가 창이 열립니다.
2. **복사 헬퍼**를 선택한 후 **다음**을 누르십시오.

3. ***yourNewBean*** bean을 선택하고 **다음**을 누르십시오.
4. 생성자 메소드 드롭 다운 목록에서 **findByPrimaryKey(***yourPackageName***.** *yourNewBean***Key)**을 생성자 메소드로 선택하십시오.
5. 속성 헬퍼 섹션에서 모든 속성을 선택하십시오.
6. **완료**를 누릅니다.
7. 작업을 저장하십시오.

전개 코드 작성: 코드 작성 유틸리티는 bean을 분석하여 Sun Microsystems의 EJB 스펙에 맞는지 확인하고 EJB 서버 고유 규칙을 따랐는지 확인합니다. 또한 선택된 엔터프라이즈 bean마다 코드 작성 도구는 CMP bean용 JDBC 지속 및 finder 클래스뿐 아니라 홈 및 EJBObject(원격) 구현, 홈 및 원격 인터페이스용 구현 클래스를 작성합니다. 또한 홈 및 원격 인터페이스용 스텁(stub)뿐 아니라, IIOP를 통한 RMI 액세스에 필요한 Java ORB, 스텁, 타이 클래스를 작성합니다.

전개 코드를 작성하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼친 후 **WebSphereCommerceServerExtensionsData**에서 마우스 오른쪽 버튼을 누르고 **새로 만들기 > 전개 및 RMIC 코드**를 선택하십시오.
   전개 및 RMIC 코드 창이 열립니다.
2. ***yourNewBean*** bean을 선택하고 **완료**를 누르십시오.

J2EE 네비게이터 보기로 전환하여 새로 작성된 코드를 볼 수 있습니다. 다음을 볼 수 있습니다.

표 1.

코드 유형	클래스 이름
컨테이너 구현 작성 코드	EJSCMP*yourNewBean*HomeBean.java
	EJSRemoteCMP*yourNewBean*.java
	EJSRemoteCMP*yourNewBean*Home.java
	EJSFinder*yourNewBean*Bean.java
JDBC 액세스 코드	EJSJDBCPersisterCMP*yourNewBean*Bean.java

표 1. (계속)

코드 유형	클래스 이름
RMI 타이 및 스텁 코드	_EJSRemoteCMP*yourNewBean*_Tie.java
	*_yourNewBean*_Stub.java
	_EJSRemoteCMP*yourNewBean*Home_Tie.java
	*_yourNewBean*Home_Stub.java

테스트 클라이언트를 사용하여 엔터프라이즈 bean 테스트: WebSphere Studio Application Developer는 엔터프라이즈 bean을 테스트하는 데 사용할 수 있는 테스트 클라이언트를 제공합니다. 테스트 클라이언트를 사용하여 새 bean을 테스트하려면 다음을 수행하십시오.

1. 서버 perspective로 전환하십시오.
2. **WebSphereCommerceServer** 서버를 두 번 누르고 **구성** 탭을 누르십시오.
3. **범용 테스트 클라이언트 사용**을 선택하십시오. 변경사항을 저장하십시오.
4. **WebSphereCommerceServer** 서버에서 마우스 오른쪽 버튼을 누르고 **시작**을 선택하십시오.
5. **WebSphereCommerceServerExtensionsData**에서 마우스 오른쪽 버튼을 누르고 **서버에서 실행**을 선택하십시오.
   범용 테스트 클라이언트와 함께 웹 브라우저가 열립니다. 엔터프라이즈 bean 테스트를 시작할 위치는 이 서버의 EJB 컨테이너에서 실행되는 bean의 이름을 찾을 수 있는 JNDI Explorer입니다.
6. **JNDI Explorer**에서 마우스 오른쪽 버튼을 누르십시오.
7. 다음은 **cell > nodes > localhost > servers > server1 > ejb >** *yourPackageStructure*와 같이 계층을 펼쳐서 *yourNewBean*Home 인터페이스를 탐색해야 합니다.
8. ***yourNewBean*Home** 인터페이스를 누르십시오. 참조 분할창에서 **EJB References > *yourNewBean* > *yourNewBean*Home**을 선택하십시오. create 메소드를 누르십시오.
9. 오른쪽 분할창에서 create 메소드의 필수 입력 매개변수에 해당하는 필드에 적절한 값을 입력하십시오.
10. **Invoke**를 누르십시오. 결과는 맨 아래 분할창에 표시됩니다.

11. **오브젝트에 대한 작업**을 눌러서 원격 인터페이스를 참조 분할창에 추가하십시오. 오브젝트 참조에서 입력한 값이 표시됩니다. 새 테이블에 새 레코드가 작성되었습니다.
12. 기타 메소드를 적절히 테스트하십시오.
13. 테스트 클라이언트를 닫고 서버를 중지하십시오.

코딩 원칙: 다음 엔터프라이즈 bean 코딩 원칙을 준수해야 합니다.

- BLOB 또는 CLOB 데이터 유형을 사용하지 마십시오.
- 엔터프라이즈 bean 코드는 엔터프라이즈 bean 모듈의 외부 요소를 참조해서는 안 됩니다. 예를 들어, 엔터프라이즈 bean 코드의 명령 또는 데이터 bean을 참조해서는 안 됩니다.
- 이전 절은 bean을 처음으로 작성할 때 액세스 제어를 새로운 bean에 포함하는 방법에 대하여 설명합니다. com.ibm.commerce.security.Protectable 인터페이스를 추가하여 bean을 작성한 후에도 추가할 있습니다. 필요한 경우, com.ibm.commerce.security.Groupable 인터페이스 또한 엔터프라이즈 bean의 원격 인터페이스에 추가하십시오. 특정 메소드도 bean에서 구현되어야 합니다. 이러한 인터페이스 및 필수 메소드를 추가한 후 bean의 전개 코드 및 액세스 bean을 다시 작성하십시오. 자세한 정보는 125 페이지의 『엔터프라이즈 bean에서 액세스 제어 구현』을 참조하십시오.

단순 데이터 bean 작성

데이터 bean은 엔터프라이즈 bean에서 정보를 검색하기 위해 JSP 템플리트에서 사용하는 bean입니다. 단순 데이터 bean은 해당 액세스 bean을 확장하고 SmartDataBean 인터페이스를 구현합니다. 데이터 bean에 대한 대부분의 코드는 WebSphere Studio Application Developer에 의해 자동 작성됩니다.

새 데이터 bean은 WebSphereCommerceServerExtensionsLogic 프로젝트에 저장됩니다.

단순 데이터 bean을 작성하려면 다음 단계를 수행해야 합니다.

1. 데이터 bean 코드를 저장할 패키지를 작성하십시오.
2. 해당 액세스 bean을 확장하고 해당 데이터 bean 인터페이스를 구현하는 데이터 bean을 작성하십시오.

3. 데이터 bean에 대한 set 메소드를 작성하십시오.
4. 데이터 bean에 대한 get 메소드를 작성하십시오.

각 단계에 대해서는 후속 절에서 보다 자세히 설명합니다.

데이터 bean 코드에 대한 패키지 작성: 패키지를 작성하면 데이터 bean 코드를 저장할 수 있는 공간이 작성됩니다.

새 패키지를 작성하려면 다음을 수행하십시오.

1. WebSphere Studio Application Developer를 열고 Java perspective로 전환하십시오.
2. **WebSphereCommerceServerExtensionsLogic** 프로젝트를 마우스 오른쪽 버튼으로 누르고 **새로 만들기 > 패키지**를 선택하십시오. 새 Java 패키지 마법사가 열립니다.
3. **소스 폴더**의 값은 `WebSphereCommerceServerExtensionsLogic/src`로 사전 설정됩니다. 이 값을 변경하지 마십시오.
4. **이름** 필드에 적절한 새 패키지의 이름을 입력하십시오. 예를 들어, `com.mycompany.mydatabeans`를 입력하십시오.
5. **완료**를 누르십시오.

데이터 bean 작성: 데이터 bean은 페이지에 동적 컨텐츠를 제공하기 위해 JSP 템플리트에서 사용되는 Java bean입니다. 이 bean은 일반적으로 액세스 bean을 확장하여 엔티티 bean의 단순 표시를 간접적으로 제공합니다. 데이터 bean은 엔티티 bean 내에서 검색되거나 설정될 수 있는 특성을 캡슐화합니다.

데이터 bean을 작성하려면 다음을 수행하십시오.

1. 데이터 bean을 저장할 패키지를 마우스 오른쪽 버튼으로 누르고 **새로 만들기 > 클래스**를 선택하십시오.
   새 Java 클래스 마법사가 열립니다.
2. 프로젝트 및 패키지 이름 필드에는 이미 대량 자료가 반입되어 있습니다.
3. **이름** 필드에 새 데이터 bean의 이름을 입력하십시오. 예를 들어, UserResAccessBean을 확장하는 데이터 bean을 작성하려면 `UserResDataBean`을 입력하십시오.

4. **수정자** 목록에서 **공개**를 선택하십시오.
5. 상위 클래스를 지정하려면 **찾아보기**를 누른 후 패턴 필드에 해당 액세스 bean의 이름을 입력하십시오. 예를 들어, UserResAccessBean을 입력하고 **확인**을 누르십시오.
6. 데이터 bean이 구현해야 하는 인터페이스를 지정하려면 **추가**를 누르십시오. 인터페이스 창에서 다음을 수행하십시오.
   a. **패턴** 필드에 com.ibm.commerce.beans.SmartDataBean을 입력한 후 **추가**를 누르십시오.
   b. **패턴** 필드에 com.ibm.commerce.beans.InputDataBean을 입력한 후 **추가**를 누르십시오.
   c. **확인**을 누르십시오.
7. **완료**를 누르십시오.

데이터 bean에 필수 필드 추가: 이 절에서는 새 데이터 bean의 필수 필드를 수정하는 방법에 대해 설명합니다. 두 개의 필수 필드는 다음 유형의 정보용입니다.

- 명령 컨텍스트
- 요청 특성

iCommandContext 필드를 수정하려면 다음을 수행하십시오.

1. 새 데이터 bean(예: UserResDataBean)을 두 번 눌러 해당 소스 코드를 보십시오.
2. getCommandContext 메소드를 찾으십시오. 이 메소드는 다음과 같이 나타납니다.

```
public CommandContext getCommandContext() {
  return null;
```

소스 코드에 다음을 추가하십시오.

```
private CommandContext iCommandContext = null;
 public com.ibm.commerce.command.CommandContext getCommandContext()
{
 return iCommandContext;
}
```

3. setCommandContext 메소드를 수정하십시오. 처음에는 다음과 같이 나타납니다.

```
public void setCommandContext(CommandContext arg0) {
 }
```

코드가 다음과 같이 나타나도록 수정하십시오.

```
 public void setCommandContext(com.ibm.commerce.command.CommandContext
   aCommandContext)
{
 iCommandContext = aCommandContext;
}
```

4. getCommandContext 메소드를 수정하십시오. 처음에는 다음과 같이 나타납니다.

```
public CommandContext getCommandContext() {
return null;
 }
```

소스 코드가 다음과 같이 나타나도록 수정하십시오.

```
public com.ibm.commerce.command.CommandContext getCommandContext ()
{
   return iCommandContext;
}
```

5. 작업을 저장하십시오.

iRequestProperties 필드를 수정하려면 다음을 수정하십시오.

1. 새 데이터 bean(예: UserResDataBean)을 두 번 눌러 해당 소스 코드를 보십시오.
2. getRequestProperties 메소드를 찾으십시오. 처음에는 다음과 같이 나타납니다.

```
public TypedProperty getRequestProperties() {
return null;
 }
```

소스 코드가 다음과 같이 나타나도록 수정하십시오.

```
private com.ibm.commerce.datatype.TypedProperty
   requestProperties;

public TypedProperty getRequestProperties() {
 return requestProperties;
}
```

3. setRequestProperties 메소드를 수정하십시오. 처음에는 다음과 같이 나타납니다.

```
public void setRequestProperties(TypedProperty arg0) throws Exception {
 }
```

소스 코드가 다음과 같이 나타나도록 수정하십시오.

```
public void setRequestProperties(com.ibm.commerce.datatype.TypedProperty
   aParam)
   throws Exception
{
// copy input TypedProperteis to local

  requestProperties = aParam;
}
```

4. getRequestProperties 메소드를 수정하십시오. 처음에는 다음과 같이 나타납니다.

```
public TypedProperty getRequestProperties() {
return null;
 }
```

소스 코드가 다음과 같이 나타나도록 수정하십시오.

```
public TypedProperty getRequestProperties() {
 return requestProperties;
}
```

5. 작업을 저장하십시오.

해당 액세스 bean의 1차 키 대량 자료 반입: 해당 액세스 bean의 1차 키를 대량 자료 반입하기 위해 소스 코드를 수정하려 할 수 있습니다. 이를 수행하려면 데이터 bean 관리자를 사용하여 이 값을 간접 설정하는 것이 좋습니다. 이 간접 방법은 URL 특성에서 가져온 1차 키 값이 1차 키를 대체하지 않는지 확인하기 위한 것입니다(아직 설정하지 않은 경우). setRequestProperties 메소드가 이 모델을 따르도록 하려면 다음 코드 단편과 유사한 방식으로 코딩하십시오. 다음 예에

서 1차 키는 사용자 ID임에 유의하십시오. 이는 상황에 따라 다를 수 있습니다. 즉, 다음 코드가 응용프로그램에서 즉시 컴파일되지 않을 수 있습니다.

```
public void setRequestProperties(
   com.ibm.commerce.datatype.TypedProperty arg1)
   throws Exception
   {
         iRequestProperties = arg1;
      try {
          if (// check for nulls
              getDataBeanKeyUserId() == null)
             {
                super.setInitKey_UserId(aUserId);
             }
       } catch (com.ibm.commerce.exception.ParameterNotFoundException e)
           {
           }
   }
```

액세스 bean의 1차 키를 설정할 수 있는 두 가지 다른 방법이 있습니다. 이는 데이터 bean의 외부, 예를 들어, JSP 템플리트에서 수행할 수 있습니다. 이 경우, JSP 템플리트에서 데이터 bean을 활성화하기 전에 1차 키에 대한 데이터 bean의 set 메소드를 명시적으로 호출하십시오. 예를 들어, JSP는 다음과 유사한 코드를 포함할 수 있습니다. 여기서, db는 데이터 bean 오브젝트입니다.

```
db.setInitKey_UserId(/*input parameter*/)
db.activate();
```

또는 1차 키를 직접적인 방법으로 설정할 수 있습니다. 즉, JSP 템플리트에는 db.activate 메소드만 있으므로 데이터 bean 관리자가 명시적으로 액세스 bean에 1차 키를 설정합니다. 예를 들어, 데이터 bean의 setRequestProperties 메소드의 코드는 다음과 유사하게 나타납니다.

```
public void setRequestProperties(
   com.ibm.commerce.datatype.TypedProperty arg1)
   throws Exception
      {
         iRequestProperties = arg1;
         try
             {
                super.setInitKey_UserId(aUserId);
             }
       } catch (com.ibm.commerce.exception.ParameterNotFoundException e)
           {
           }
      }
```

1차 키 설정을 위한 권장 프로시저는 간접 방법임에 유의하십시오.

populate() 메소드 수정: 다음을 수행하여 populate 메소드를 수정해야 합니다.

1. 새 데이터 bean을 펼쳐 해당 필드 및 메소드를 보십시오.
2. 아웃라인 보기에서 **populate()** 메소드를 선택하여 해당 소스 코드를 보십시오.

   처음에는 다음과 같이 나타납니다.

   ```
   public void populate () throws Exception {}
   ```

3. 메소드가 다음과 같이 표시되도록 소스 코드를 수정하십시오.

   ```
        try {
      super.refreshCopyHelper();
   } catch (javax.ejb.FinderException e) {
   throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
        "UserResDataBean", "populate");
   ```

   작업을 저장하십시오(Ctrl+S).

새 데이터 bean이 인스턴스 작성 시 추가 입력 매개변수를 필요로 하는 액세스 bean을 확장하는 경우, 데이터 bean의 populate 메소드에도 해당 값을 설정해야 함에 유의하십시오.

새 세션 bean 작성

새 세션 bean을 작성하는 경우, WebSphereCommerceServerExtensionsData 프로젝트에서 작성하십시오.

새 세션 bean은 com.ibm.commerce.base.helpers.BaseJDBCHelper 클래스를 확장해야 합니다. 상위 클래스는 세션 bean이 다른 엔티티 bean과 동일한 트랜잭션에 참여할 수 있도록 WebSphere Commerce Server가 사용하는 데이터 소스 오브젝트에서 JDBC 연결 오브젝트를 확보할 수 있는 방법을 제공합니다. 다음은 상위 클래스에서 제공하는 기능을 보여주는 코드 예입니다.

```
public class mySessionBean extends com.ibm.commerce.base.helpers.BaseJDBCHelper
   implements SessionBean {

   public Object myMethod () throws javax.naming.NamingException,
      SQLException {

       ////////////////////////////////////////////////
      //  -- your logic, such as initialization --    //
      ////////////////////////////////////////////////
```

```
        try {
               // get a connection from the WebSphere Commerce data source
               makeConnection();
               PreparedStatement stmt = getPreparedStatement(
                  "your sql string");
               ////////////////////////////////////////////////////////
               // -- your logic such as set parameter into the prepared  //
               // statement --                                          //
               ////////////////////////////////////////////////////////
               ResultSet rs = executeQuery(stmt, false);

               ////////////////////////////////////////////////
               // -- your logic to process the result set -- //
              ////////////////////////////////////////////////

                  }
              finally {
                 // return the connection to the WebSphere Commerce data source
                closeConnection();
              }

             ////////////////////////////////////////////////////
            // -- your logic to return the result ---           //
            ////////////////////////////////////////////////////

     }

}
```

앞의 코드 예에서 executeQuery 메소드는 두 개의 입력 매개변수를 취합니다. 첫 번째는 prepared 문이고, 두 번째는 캐시 비우기 조작과 관련되는 부울 플래그입니다. 조회를 실행하기 전에 캐시에서 현재 트랜잭션에 대한 모든 엔티티 오브젝트를 비우기 위한 컨테이너가 필요한 경우, 이 플래그를 TRUE로 설정하십시오. 이 플래그는 일부 엔티티 오브젝트를 갱신하고 이 갱신된 오브젝트를 통해 검색하기 위해 조회가 필요한 경우 필요합니다. 플래그를 FALSE로 설정하는 경우, 트랜잭션을 종료해야 해당 엔티티 오브젝트 갱신이 데이터베이스에 작성됩니다.

꼭 필요한 경우를 제외하고는, 이 비우기 조작의 사용을 제한하고 일반적으로 플러그를 FALSE로 설정해야 합니다. 비우기 조작은 자원 집약 조작입니다.

오브젝트 사용 주기

오브젝트 모델의 엔터프라이즈 bean에는 독립 및 종속 오브젝트가 모두 포함됩니다. 독립 오브젝트에는 오브젝트를 호출하는 비즈니스 로직의 create 또는 remove 요청에 의해 직접 제어되는 고유한 사용 주기가 있습니다. 종속 오브젝트에는 다

른 오브젝트(*소유자 오브젝트*라고도 함)에 첨부되는 사용 주기가 있습니다. 이 오브젝트는 다시 종속 오브젝트도 될 수 있지만 연관 계층 상위에는 독립 오브젝트가 존재합니다. 소유자 오브젝트가 삭제되면 모든 종속 오브젝트 또한 삭제됩니다. 실제 삭제는 데이터베이스에서 스펙을 연속 삭제하여 제어됩니다.

예를 들어, 주소록 오브젝트 및 주문 목록 오브젝트를 리턴하는 사용자 오브젝트의 경우, 사용자 오브젝트가 삭제되면 해당 주소록 오브젝트 또한 삭제되며(주소록은 사용자 소유이므로), 주소록 내 모든 주소 오브젝트도 삭제됩니다(주소는 주소록의 소유이므로). 그러나 주문의 소유자는 사용자 오브젝트가 아닌 상점 오브젝트이므로 주문 오브젝트는 삭제되지 않습니다.

종속 오브젝트 작성에는 특정 설계 패턴이 사용됩니다. 종속 오브젝트의 create 메소드는 해당 소유자 오브젝트에 대한 참조를 제공해야 합니다. 따라서 종속 오브젝트를 작성하려면 먼저 소유자 오브젝트가 존재해야 합니다.

트랜잭션

엔터프라이즈 JavaBeans 버전 1.1 아키텍처는 인스턴스 상태에 대한 세 가지 대체 확약 시간 옵션을 지정합니다. 이 옵션은 스펙 문서에서 옵션 A, B 및 C로 설명됩니다. 이 옵션에 대한 자세한 정보는 Sun Microystem의 Enterprise JavaBean 버전 1.1 스펙 문서를 참조하십시오.

WebSphere Application Server는 옵션 A 및 C를 구현하지만, 옵션 A는 데이터베이스를 공유하지 않는 것으로 가정합니다.

옵션 C의 경우, 엔터프라이즈 bean 컨테이너가 트랜잭션 간에 “ready” 인스턴스를 캐시하지 않습니다. 트랜잭션이 완료되면 사용 가능한 인스턴스 풀로 인스턴스가 리턴됩니다. WebSphere Commerce는 옵션 C를 사용합니다. 복수 WebSphere Commerce 응용프로그램에서 데이터베이스를 공유하기 때문입니다. 이 구현에서는 각 트랜잭션 시작 시 컨테이너가 엔티티 bean에 대한 지속 데이터를 로드하고, 트랜잭션 지속시간에만 엔티티 bean이 캐시됩니다. 컨테이너는 엔티티를 액세스하는 각 트랜잭션마다 하나씩, 엔티티 bean의 복수 인스턴스를 활성화합니다. 데이터베이스에 의해 트랜잭션 동기화가 수행됩니다.

각 엔터프라이즈 bean의 트랜잭션 속성은 TX_REQUIRED로 설정됩니다. 웹 컨트롤러는 해당 액세스 bean을 통해 엔터프라이즈 bean에 액세스하는 명령을 실행하기 전에 트랜잭션을 시작하므로, 이 트랜잭션의 컨텍스트에서 엔터프라이즈 bean의 비즈니스 메소드가 호출됩니다.

엔티티 bean에 대한 기타 고려사항

find for update

복수 응용프로그램이 행 갱신을 위해 데이터베이스 내 동일한 행에 액세스하는 상황을 *동시 갱신*이라고 합니다. 동시 갱신이 허용되는 상황도 있고, 동시 갱신이 허용되지 않는 상황도 있습니다.

데이터베이스 갱신이 겹쳐쓰기인 경우, 즉 새 값이 데이터베이스 내 현재 값과 관계가 없는 경우, 동시 갱신이 허용될 수 있습니다. 동시 갱신이 허용되고 복수 응용프로그램이 데이터베이스 내 동일한 행을 갱신하려고 시도하는 경우, 마지막 시도만이 데이터베이스의 행을 갱신합니다.

데이터베이스 갱신이 데이터베이스 내 현재 값에 의존하는 경우, 동시 갱신은 바람직하지 않습니다. 예를 들어, 응용프로그램이 상품 재고를 갱신하는 경우, 한 번에 하나의 응용프로그램만 재고를 갱신할 수 있어야 합니다.

동시 갱신의 허용 여부에 영향을 주는 요소에는 데이터베이스 잠금 및 엔터프라이즈 bean 분리 레벨이 포함됩니다.

두 번째 응용프로그램이 행을 동시 갱신하지 못하도록 막으려면 행에 액세스하는 첫 번째 응용프로그램이 “find for update” 옵션을 사용하여 행을 가져와야 합니다. “for update” 옵션을 사용하는 경우, *쓰기 잠금*(독점 잠금이라고도 함)이 행에 적용됩니다. 이 쓰기 잠금을 행에 적용하면 “find for update”를 사용하여 행에 액세스하려고 시도하는 응용프로그램이 모두 차단됩니다.

응용프로그램이 동시 갱신을 허용하는 경우, 행을 잠그지 않고 데이터만 가져올 수 있습니다.

UpdateInventory가 주문에 포함되어 있는 모든 상품을 찾아야 하는 OrderProcess 시나리오를 고려하여 그에 따라 재고를 갱신하십시오. 많은 다른 주문에 동일한 상품이 포함될 수 있으므로 *find for update*를 사용해야 하며 트랜잭션 범위 내에서

가능한 빨리 이를 사용하여 교착 상태의 가능성을 줄여야 합니다. 따라서 UpdateInventory 알고리즘은 다음 의사 코드에 의해 표시될 수 있습니다.

```
UpdateInventory
 find all the order items in the order
 for each order item
  fetch its inventory using "find for update"
  ...
```

주문 항목이 많은 장기 B2B 시나리오의 경우, 가능한 빨리 find for update를 사용해야 합니다. 로직은 다음과 같습니다.

```
find for update the inventory of all the products in an order
for each product
 if (total quantity ordered for that product < inventory)
        deduct quantity from inventory
 else
        error
```

flush remote 메소드

WebSphere Application Server는 트랜잭션 확약 시간이 되어야 엔티티 bean의 변경사항을 데이터베이스에 기록하므로, 데이터베이스가 엔티티 bean의 컨테이너에 캐시된 데이터와 임시로 동기화되지 않을 수 있습니다.

모든 트랜잭션의 확약 변경사항을 기록하여(즉, 엔터프라이즈 bean 캐시에서 정보를 가져와) 데이터베이스를 갱신하는 flush remote 메소드가 `com.ibm.commerce.base.helpers.BaseJDBCHelper` 클래스에 제공됩니다. 이 remote 메소드는 명령으로 호출할 수 있습니다. 꼭 필요한 경우에만 이 메소드를 사용하십시오. 과부화 자원 측면에서 부담이 커 성능에 부정적인 영향을 주기 때문입니다.

다음 코드 단편이 있는 로그온 명령을 고려하십시오.

```
UserAccessBean uab = ...;
 uab.setRegisteredTimestamp(currentTimestamp);
 uab.commitCopyHelper();
```

트랜잭션을 확약해야 USERS 테이블의 REGISTRATIONUPDATE가 현재 시간소인으로 갱신됩니다. 갱신은 트랜잭션 확약 시간에만 발생합니다. 동일한 트랜잭션 내 직접 JDBC 조회(예: *select from user where registeredstamp...*)가 지정된 등록 시간소인으로 사용자를 리턴하도록 flush 메소드를 사용해야 합니다.

엔터프라이즈 bean 보안

엔터프라이즈 bean 보안을 위해 WebSphere Application Server를 사용하는 경우, 새 엔터프라이즈 bean의 메소드에 `WCSecurityRole` 역할을 지정해야 합니다. WebSphere Application Server 응용프로그램 어셈블리 도구를 사용하여 이 태스크를 수행합니다. 새 엔터프라이즈 bean을 전개하는 경우 이 단계를 수행하십시오. 또한 기존 WebSphere Commerce 엔티티 bean을 수정하는 경우, `WCSecurityRole` 역할을 수정된 EJB 프로젝트 내 각 엔티티 bean의 각 메소드에 지정해야 합니다.

사용자 정의 코드의 전개 프로세스에 대한 설명은 233 페이지의 제 9 장 『전개 정보』를 참조하십시오.

1차 키

1차 키는 테이블 정의의 일부분인 고유 키입니다. 하나의 레코드를 다른 레코드와 구별하는 데 사용할 수 있습니다. 모든 레코드에는 1차 키가 있어야 합니다. 테이블에 새 레코드를 작성하는 경우, 레코드에 대한 고유 1차 키를 작성해야 할 수 있습니다.

WebSphere Commerce 프로그래밍 모델의 경우, 지속성 계층에는 데이터베이스와 상호작용하는 엔티티 bean이 포함됩니다. 따라서 엔티티 bean의 인스턴스 작성 시 데이터베이스 레코드를 작성할 수 있습니다. 그러므로 엔티티 bean의 인스턴스 작성을 위한 ejbCreate 메소드가 새 레코드에 대한 1차 키를 작성하기 위한 로직을 포함해야 할 수 있습니다.

응용프로그램에서 데이터베이스 정보가 필요한 경우, bean의 해당 액세스 bean의 인스턴스를 작성한 후 다양한 필드를 가져오거나 설정하여 간접적으로 엔티티 bean을 사용합니다. 데이터베이스의 특정 레코드(예: 특정 사용자 프로파일)에 대해 액세스 bean의 인스턴스가 작성되며 이 bean은 1차 키를 사용하여 데이터베이스에서 올바른 정보를 선택합니다.

다음 절에서는 고유 1차 키 작성 방법 및 1차 키로 선택하는 방법에 대해 설명합니다.

1차 키 작성: 엔티티 bean의 새 인스턴스를 작성하기 위해서는 ejbCreate 메소드가 사용됩니다. 이 메소드는 자동 작성되지만 작성된 메소드에는 1차 키를 정적 값으로 초기화하기 위한 로직만 포함됩니다.

1차 키가 고유한 새 값인지 확인해야 합니다. 이 경우, 다음 코드 단편과 유사한 ejbCreate 메소드가 표시됩니다.

```
public void ejbCreate(int argMyOtherValue)
   throws javax.ejb.CreateException {
      //Initialize CMP fields
      MyKeyValue = com.ibm.commerce.key.ECKeyManager.
         singleton().getNextKey("table_name");
      MyOtherValue = argMyOtherValue;
}
```

앞의 코드 단편에서 getNextKey 메소드는 1차 키에 고유한 정수를 작성합니다. 메소드의 *table_name* 입력 매개변수는 KEYS 테이블에 정의되어 있는 TABLENAME 값과 정확히 일치해야 합니다. 문자 및 대소문자가 정확히 일치해야 합니다.

ejbCreate 메소드에 앞의 코드를 포함시키는 것 외에, KEYS 테이블에 항목도 작성해야 합니다. 다음은 KEYS 테이블에 항목을 작성하기 위한 SQL문 예입니다.

```
insert into KEYS (TABLENAME, COUNTER, KEYS_ID)
   values ("table_name", 0, 1)
```

앞의 SQL문을 사용하는 경우, KEYS 테이블에 있는 다른 열의 기본값이 승인됨에 유의하십시오. COUNTER 값은 계수를 시작해야 하는 값을 표시합니다. KEYS_ID 값은 양수 값이어야 합니다.

1차 키가 long 데이터 유형으로 정의되어 있는 경우(DB2®의 경우 BIGINT 또는 Oracle의 경우 NUMBER(38, 0))), getNextKeyAsLong 메소드를 사용하십시오.

1차 키로 선택: 액세스 bean에서 1차 키를 사용하여 해당 데이터베이스 레코드를 선택해야 합니다. 다음 코드 단편은 이 선택 수행 방법을 보여줍니다. 또한 나중에 설명하는 추가 로직이 포함되어 있습니다.

```
UserProfileAccessBean abUserProfile = new UserProfileAccessBean();
abUserProfile.setInitKey_UserId(getUserId().toString());
abUserProfile.refreshCopyHelper();
```

앞의 코드 단편의 첫 번째 행은 새 UserProfileAccessBean, ″abUserProfile″의 인스턴스를 작성합니다. 두 번째 행은 액세스 bean에 1차 키를 설정합니다. `setInitKey_xxx`(여기서, *xxx*는 1차 키 필드 이름) 이름 지정 규칙은 WebSphere Studio Application Developer가 1차 키의 set 메소드의 이름을 지정하기 위해 사용합니다. 액세스 bean의 인스턴스 작성 시, refreshCopyHelper 메소드를 사용하기 전에 setInitKey_xxx 메소드로 설정한 모든 필드가 초기화되었는지 확인해야 합니다. setInitKey_xxx 메소드의 호출 순서는 중요하지 않습니다.

모든 setInitKey_xxx 메소드가 호출되면 모든 필수 필드가 초기화되고, refreshCopyHelper 메소드를 사용하여 데이터베이스에서 정보를 검색할 수 있습니다.

액세스 bean의 로컬 캐시에서 값을 갱신하는 경우, 갱신된 정보로 데이터베이스를 갱신하기 위한 commitCopyHelper 호출 또한 포함해야 합니다. 예를 들어, refreshCopyHelper 메소드를 사용하여 데이터 검색 후 고객 이름을 갱신하는 경우(이름 값을 설정하여), `abUserProfile.commitCopyHelper()`를 호출하여 새 정보로 데이터베이스를 갱신해야 합니다.

엔티티 bean 사용

엔터프라이즈 bean을 사용하는 프로그램은 엔터프라이즈 bean의 홈 및 원격 인터페이스 뿐 아니라 JNDI(Java Naming and Directory Interface)를 처리해야 합니다. 프로그래밍 모델을 단순화하기 위해 각 엔터프라이즈 bean에 대한 액세스 bean이 작성됩니다. 사용자 고유 엔터프라이즈 bean 작성 시, WebSphere Studio Application Developer의 도구를 사용하여 이 액세스 bean을 작성하십시오.

WebSphere Commerce 명령은 엔티티 bean과 직접 상호작용하지 않고 액세스 bean과 상호작용합니다. 도표에서 설명하는 것처럼 액세스 bean을 사용하면 다음과 같은 이점을 제공합니다.

- 프로그래밍 인터페이스를 단순화합니다. 액세스 bean은 Java bean과 같은 기능을 하며 모든 엔터프라이즈 bean 고유 프로그래밍 인터페이스(예: JNDI), 홈 및 원격 인터페이스를 클라이언트로부터 숨깁니다.
- 런타임 시 액세스 bean은 엔터프라이즈 bean 홈 오브젝트를 캐시합니다. 시간 및 자원 사용량 측면에서 볼 때 홈 오브젝트 검색은 부담이 크기 때문입니다.

- 액세스 bean은 명령이 엔터프라이즈 bean 속성을 가져오고 설정할 때 엔터프라이즈 bean 호출 횟수를 줄이는 copyHelper 오브젝트를 구현합니다. 따라서 복수 엔터프라이즈 bean 속성을 읽거나 쓸 때 엔터프라이즈 bean은 한 번만 호출하면 됩니다.

다음 도표는 명령, 액세스 bean, 엔티티 bean 및 데이터베이스 간의 상호작용을 표시합니다.

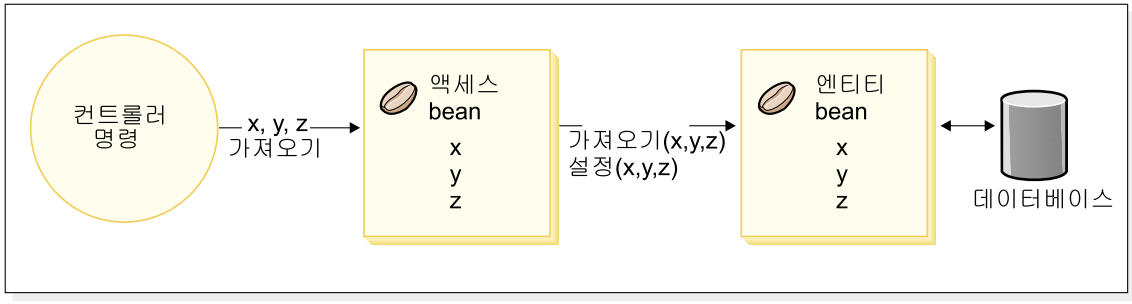


그림 18.

데이터베이스 고려사항

전자 상거래 응용프로그램을 사용자 정의하면 새 테이블을 작성할 수 있습니다. 이 테이블 작성 시, WebSphere Commerce 테이블과 일관된 방식으로 테이블을 작성하기 위해 일련의 규칙을 준수하도록 권장합니다.

데이터베이스 스키마 오브젝트 이름 지정 고려사항

다음 절에서는 데이터베이스 스키마 오브젝트의 이름 지정 규칙을 제공합니다.

테이블 및 뷰의 이름 지정 규칙

다음 목록은 새 테이블 및 뷰의 이름 지정 규칙을 제공합니다.

- 이후 릴리스의 WebSphere Commerce 테이블 및 뷰와의 이름 충돌(이름 중복)을 막기 위해 테이블 또는 뷰 이름의 처음 문자는 X여야 합니다(예: XMYTABLE).

- 테이블 또는 뷰 이름의 길이는 10자 이하여야 합니다. 원하는 이름이 이 한계를 초과하는 경우, 10자가 될 때까지 이름 끝에서 모음을 제거하여 길이를 줄이십시오.
- 테이블 또는 뷰 이름에는 특수 문자(예: “_”, “+”, “$”, “%” 또는 공백 문자)가 없어야 합니다.
- 테이블 또는 뷰 이름에는 데이터베이스 전용어를 사용하지 마십시오.
- 뷰 이름은 VW로 끝나야 합니다.
- 테이블 및 뷰 이름은 단수 명사여야 합니다.

열의 이름 지정 규칙

일반적으로 테이블 이름에 대한 이전 지정 규칙을 따르는 새 테이블을 작성하는 경우, 해당 테이블 내 열에 대하여 사용자 고유의 이름 지정 규칙을 구현할 수 있습니다. 사용자가 항상 SQL문에서 완전한 열 이름을 사용한다고 가정합니다. 그러나 기존 WebSphere Commerce 테이블과 결합은 수행하되 완전한 이름은 사용하지 않으려는 경우, 이 절에 설명된 열 이름 지정 규칙을 따라야 합니다.

다음 목록은 새 테이블에 있는 열의 이름 지정 규칙을 제공합니다.

- 이후 릴리스의 WebSphere Commerce 테이블 열과의 이름 충돌(이름 중복)을 막기 위해 열 이름의 처음 문자는 X여야 합니다(예: XMYCOLUMN).
- 열 이름의 길이는 18자 이하여야 합니다. 원하는 이름이 이 한계를 초과하는 경우, 18자가 될 때까지 이름 끝에서 모음을 제거하여 길이를 줄이십시오.
- 열 이름(foreign key 이외)에는 특수 문자(예: “_”, “+”, “$”, “%” 또는 공백 문자)가 없어야 합니다.
- 열 이름에 데이터베이스 전용어를 사용하지 마십시오.
- 능동태 조합을 사용하여 결합어를 열 이름으로 사용할 수 있습니다. 예를 들어, COMBINERESULT입니다.
- 작성된 1차 키 열의 이름은 *table*_id로 지정해야 합니다. 예를 들어, USERS 테이블의 1차 키는 USERS_ID입니다.
- 작성된 foreign key 열 이름을 변경해서는 안 됩니다.
- 향후 사용자 정의를 위해 열을 예약하는 경우, 이름을 field*x*로 지정해야 합니다. 여기서, *x*는 숫자입니다(최소값은 1).

색인의 이름 지정 규칙

다음 목록은 새 테이블에 있는 색인의 이름 지정 규칙을 제공합니다.

- 색인 이름의 길이는 18자 이하여야 합니다.
- 색인 이름은 공백을 포함해서는 안 됩니다.
- 색인 이름은 데이터베이스 전용어를 포함해서는 안 됩니다.
- 고유하지 않은 색인 이름은 I_*tablex*로 지정해야 합니다. 여기서, *table*은 테이블 이름이며 *x*는 숫자(최소 값은 1)입니다. 예를 들어, USERS 테이블의 고유하지 않은 색인은 I_USERS1입니다.
- 고유 색인 이름은 UI_*tablex*로 지정해야 합니다. 여기서, *table*은 테이블 이름이며 *x*는 숫자(최소 값은 1)입니다. 예를 들어, USERS 테이블의 고유 색인은 UI_USERS1입니다.
- 색인의 전체 *크기*는 254바이트 이하여야 합니다.
- 색인 이름은 전체 데이터베이스 스키마에서 고유해야 합니다.

1차 키의 이름 지정 규칙

다음 목록은 새 테이블에 있는 1차 키의 이름 지정 규칙을 제공합니다.

- 1차 키 이름의 길이는 18자 이하여야 합니다.
- 1차 키 이름은 공백을 포함해서는 안 됩니다.
- 1차 키 이름은 데이터베이스 전용어를 포함해서는 안 됩니다.
- 1차 키 이름은 P_*table*로 지정해야 합니다. 여기서, *table*은 테이블 이름입니다. 예를 들어, USERS 테이블의 1차 키는 P_USERS입니다.
- 1차 키 이름은 전체 데이터베이스 스키마에서 고유해야 합니다.

foreign key의 이름 지정 규칙

다음 목록은 새 테이블에 있는 foreign key의 이름 지정 규칙을 제공합니다.

- foreign key 이름의 길이는 18자 이하여야 합니다.
- foreign key 이름은 공백을 포함해서는 안 됩니다.
- foreign key 이름은 데이터베이스 전용어를 포함해서는 안 됩니다.
- foreign key 이름은 F_*table*로 지정해야 합니다. 여기서, *table*은 테이블 이름입니다. 예를 들어, USERS 테이블의 foreign key는 F_USERS1입니다.
- foreign key 이름은 전체 데이터베이스 스키마에서 고유해야 합니다.

데이터베이스 트리거의 이름 지정 규칙

다음 목록은 데이터베이스 트리거의 이름 지정 규칙을 제공합니다.

- 데이터베이스 트리거 이름의 길이는 18자 이하여야 합니다.
- 테이터베이스 트리거 이름은 공백을 포함해서는 안 됩니다.
- 데이터베이스 트리거 이름은 데이터베이스 전용어를 포함해서는 안 됩니다.
- 데이터베이스 트리거 이름은 T_*table*로 지정해야 합니다. 여기서, *table*은 테이블 이름입니다. 예를 들어, USERS 테이블의 데이터베이스 트리거 이름은 T_USERS1입니다.
- 데이터베이스 트리거 이름은 전체 데이터베이스 스키마에서 고유해야 합니다.

데이터베이스 열 데이터 유형 고려사항

이 절에서는 새 테이블 작성 시 사용할 수 있는 열 데이터 유형에 대해 소개합니다. 다양한 데이터 유형의 설명은 DB2 용어를 사용합니다. 101 페이지의 『데이터베이스 간의 데이터 유형 차이』는 다른 데이터베이스를 사용하는 경우 차이점에 대해 설명합니다.

BIGINT

서명된 64비트 정수입니다(-9223372036854775807 - 9223372036854775807). BIGINT 크기의 절반인 INTEGER와 이를 비교하십시오.

INTEGER

서명된 32비트 정수입니다(-2147483647 - 2147483647). 일반적으로 기본 유한 숫자 데이터 유형은 BIGINT가 아닌 INTEGER이어야 합니다. BIGINT를 사용해야 하는 강력한 비즈니스상의 이유가 있지 않는 한, 숫자 데이터 유형은 INTEGER를 사용하는 것이 더 좋습니다. BIGINT 데이터 유형의 일반 사용자는 시스템 작성 키입니다.

SMALLINT 또는 SHORT 데이터 유형은 절대로 사용하지 말아야 합니다. 이 데이터 유형은 비오브젝트 Java 데이터 유형으로 맵핑되고, 이러한 비오브젝트 데이터 유형이 일부 엔터프라이즈 bean 오브젝트 인스턴스 작성에 문제점을 야기하기 때문입니다.

TIMESTAMP

날짜 및 시간을 지정하는 일곱 파트(연도, 월, 일, 시간, 분, 초 및 마이크로세컨드)로 구성되는 값입니다. 시간에는 소량 스펙, 마이크로세컨드가 포함됩니다. 시간소인에 대한 내부 표시는 10바이트 문자열이며 각 바이트는 두 개의 팩형 10진수 숫자로 구성됩니다. 처음 4바이트는 날짜를, 다음 3바이트는 시간을, 마지막 3바이트는 마이크로세컨드를 타나냅니다.

CHAR

INTEGER 길이의 고정 길이 문자열입니다(1자 - 254자). 길이 스펙이 생략되면 한 자로 가정됩니다. CHAR은 고정 길이 데이터베이스 열이므로 사용되지 않은 문자 뒷 공간은 흰색 공백으로 변경됩니다. 성능 이유가 아닌 이상, CHAR 데이터 유형은 사용하지 않는 것이 바람직합니다. CHAR은 고정적이어서 나중에 길이를 변경할 수 없기 때문입니다. 일반적으로 문자열의 길이가 64자 미만이고 이를 정기적으로 검색 또는 갱신하는 경우, 보다 나은 성능을 위해 CHAR을 사용하십시오.

VARCHAR

최대 길이 정수의 길이 변동 문자열입니다(1자 - 32672자). 그러나 열 데이터가 테이블과 함께 저장되는 CHAR과 달리, VARCHAR은 데이터베이스 페이지 내 참조점으로서 내부적으로 표시됩니다. 따라서 VARCHAR 열의 길이는 작성 후 어느 때나 변경할 수 있습니다.

LONG VARCHAR

동일한 데이터베이스 페이지에서 VARCHAR을 작성할 수 없는 경우 사

용할 수 있는 길이 변동 문자열입니다. LONG VARCHAR은 복수 데이터베이스 페이지에 적용할 수 있다는 점을 제외하고는 VARCHAR과 매우 유사합니다. LONG VARCHAR 데이터 유형은 꼭 필요한 경우에만 사용하십시오. LONG VARCHAR 오브젝트는 일반적으로 성능면에서 부담이 *크기* 때문입니다.

CLOB

열 길이가 LONG VARCHAR의 32KB 한계를 초과해야 하는 경우 사용할 수 있는 또 다른 길이 변동 문자열입니다. CLOB 오브젝트의 길이는 데이터베이스 구성을 수정하지 않고 1GB까지 가능합니다. CLOB으로 저장되는 텍스트 데이터는 다른 시스템 사이에서 이동 시 적절히 변환됩니다.

BLOB

데이터베이스에 구조화되지 않은 데이터를 저장하는 길이 변동 2진 문자열입니다. BLOB 오브젝트는 최대 4GB의 2진 데이터를 저장할 수 있습니다. 일반적으로 꼭 필요한 경우를 제외하고는 BLOB을 열 데이터 유형으로 사용하지 않아야 합니다. 성능면에서 BLOB 오브젝트는 어느 데이터베이스에서나 가장 부담이 큰 오브젝트 중 하나로 간주됩니다.

DECIMAL(20,5)

이 데이터 유형은 특별히 고정 소수점 숫자(예: 통화 단위)에 사용하도록 정의됩니다. 기타 부동 소수점 숫자의 경우, 대신 FLOAT를 사용할 수 있습니다.

데이터베이스 간의 데이터 유형 차이

다음 테이블은 WebSphere Commerce 데이터베이스 스키마에서 사용하는 데이터 유형 및 다른 데이터베이스 구현에 대한 해당 데이터 유형을 표시합니다.

JDBC 오브젝트	Windows AIX Solaris Linux **DB2**	Windows AIX Solaris **Oracle®**	400 **DB2**
해시 테이블	BLOB()	BLOB	BLOB()
시간소인	TIMESTAMP	DATE	TIMESTAMP
정수	INTEGER	INTEGER	INTEGER
BigDecimal	DECIMAL(,)	DECIMAL(,)	DECIMAL(,)
Long	BIGINT	NUMBER(38,0)	BIGINT
Double	FLOAT	NUMBER(38,0)	FLOAT
문자열	CHAR()	VARCHAR2()	GRAPHIC() CCSID 13488
byte[]	CHAR()(비트 데이터용)	RAW()	CHAR()(비트 데이터용)
문자열	VARCHAR()	VARCHAR2()	VARGRAPHIC() CCSID 13488
문자열	LONG VARCHAR	VARCHAR2()(자세한 정보는 다음 테이블을 참조하십시오.)	VARGRAPHIC(4000) ALLOCATE() CCSID 13488
byte[]	LONG VARCHAR(비트 데이터용)	LONG RAW	VARCHAR(8000) ALLOCATE()(비트 데이터용)
문자열	CLOB()	CLOB()	DBCLOB() CCSID 13488

주:

Oracle JDBC 드라이버가 데이터 유형이 LONG인 정보를 성공적으로 처리할 확률이 일관적이지 않으므로 가능한 경우 LONG 데이터 유형은 사용하지 않는 것이 바람직합니다. 이 상황에서 가장 자주 보고되는 오류는 “스트림이 이미 종료되었습니다”입니다.

이 데이터 유형을 사용해야 하는 경우, 데이터베이스 테이블당 LONG 유형을 사용하는 열을 하나만 사용할 수 있습니다. 또한 select 문 구성 시, LONG 열을 select에 첫 번째 또는 마지막 요소로 두지 마십시오. 부하가 높은 작업에 대한 또 다른 문제 해결 방법은 이 특정 열을 엔티티 bean의 CMP 필드에 맵핑하지 않는 것입니다. 대신 세션 bean을 사용하여 이 열을 검색 및 갱신을 수행하십시오.

제 4 장 액세스 제어

액세스 제어 이해

WebSphere Commerce 응용프로그램의 액세스 제어 모델에는 세 가지 기본 개념, 즉 사용자, 조치 및 자원이 있습니다. 사용자는 시스템을 사용하는 사람입니다. 자원은 응용프로그램에서 유지보수되는 엔티티입니다. 예를 들어, 자원은 상품, 문서 또는 주문일 수 있습니다. 사람을 나타내는 사용자 프로파일 또한 자원입니다. 조치는 사용자가 자원에서 수행할 수 있는 활동입니다. 액세스 제어는 해당 사용자가 해당 자원에서 해당 조치를 수행할 수 있는지 여부를 판별하는 전자 상거래 응용프로그램의 구성요소입니다.

WebSphere Commerce 응용프로그램에는 두 가지 기본 액세스 제어 레벨이 있습니다. 첫 번째 액세스 제어 레벨은 WebSphere Application Server가 수행합니다. 이 점에 있어 WebSphere Commerce는 WebSphere Application Server를 사용하여 엔터프라이즈 bean 및 Servlet을 보호합니다. 두 번째 액세스 제어 레벨은 WebSphere Commerce의 정밀한 액세스 제어 시스템입니다.

WebSphere Commerce 액세스 제어 프레임워크는 액세스 제어 정책을 사용하여 해당 사용자가 해당 자원에서 해당 조치를 수행할 수 있는 권한이 있는지 여부를 판별합니다. 이 액세스 제어 프레임워크는 정밀한 액세스 제어를 제공합니다. 이 프레임워크는 WebSphere Application Server가 제공하는 액세스 제어와 함께 사용하되 이를 바꾸지는 않습니다.

WebSphere Application Server의 자원 보호에 대한 개요

다음은 WebSphere Application Server에 의해 액세스 제어에서 보호되는 WebSphere Commerce 자원입니다.

- 엔티티 bean
  이 bean은 전자 상거래 응용프로그램의 오브젝트를 모델링합니다. 이 bean은 원격 클라이언트가 액세스할 수 있는 분배 오브젝트입니다.

- JSP 템플리트

  WebSphere Commerce는 표시 페이지에 JSP 템플리트를 사용합니다. 각 JSP 템플리트에는 엔티티 bean에서 데이터를 검색하는 하나 이상의 데이터 bean이 있을 수 있습니다. 클라이언트는 URL 요청을 작성하여 JSP 페이지를 요청할 수 있습니다.
- 컨트롤러 및 뷰 명령

  클라이언트는 URL 요청을 작성하여 컨트롤러 및 뷰 명령을 요청할 수 있습니다. 또한 하나의 표시 페이지는 JSP 파일 이름 또는 뷰 이름(VIEWREG 테이블에 등록되어 있는 대로)을 사용하여 다른 표시 페이지에 대한 링크를 포함할 수 있습니다.

WebSphere Commerce Server는 일반적으로 다음 웹 경로를 사용하도록 구성됩니다.

- `/webapp/wcs/stores/servlet/*`

  이 경로는 요청 Servlet에 대한 요청에 사용됩니다.
- `/webapp/wcs/stores/*.jsp`

  이 경로는 JSP Servlet에 대한 요청에 사용됩니다.

다음 도표는 이전 웹 경로 구성에 있어 요청이 WebSphere Commerce 자원에 액세스하기 위해 잠재적으로 사용할 수 있는 루트를 표시합니다.

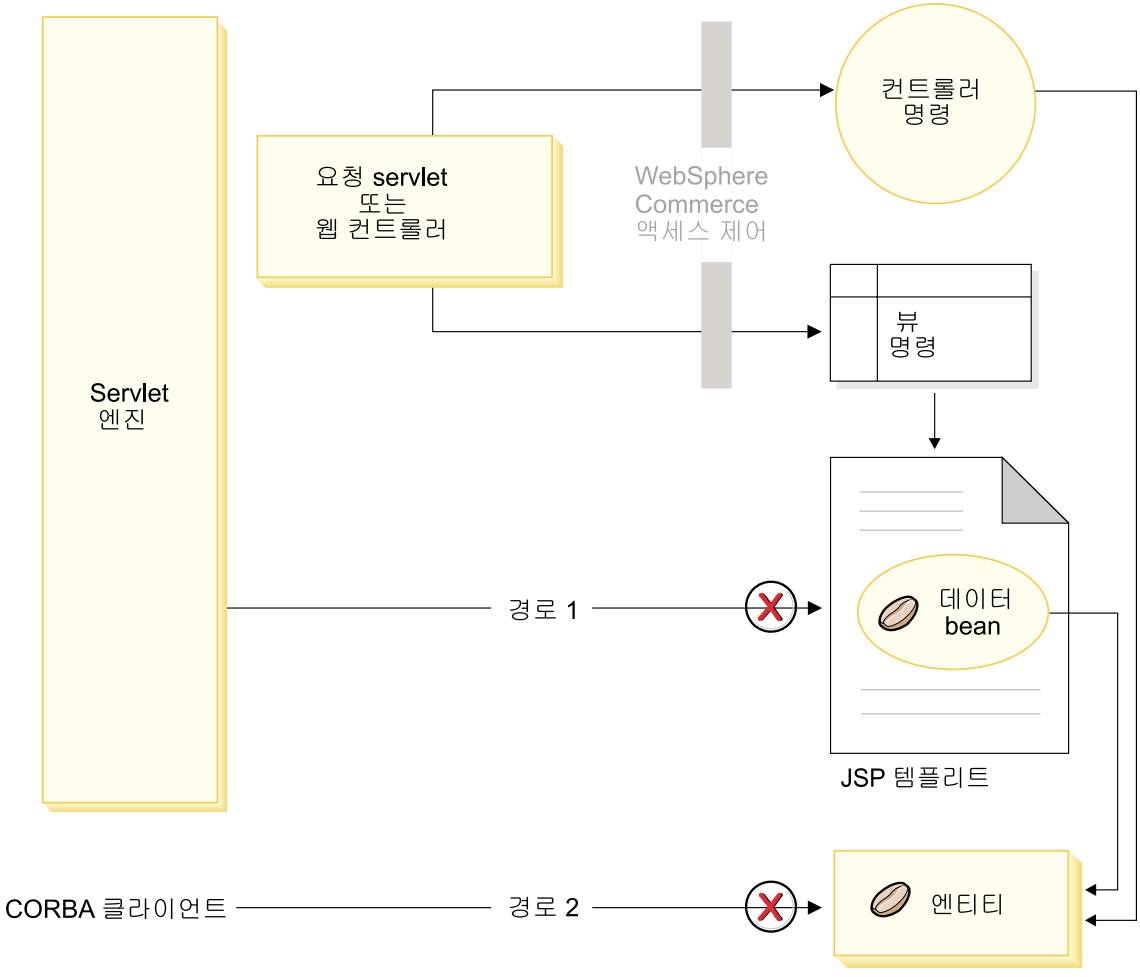


그림 19.

모든 올바른 요청은 요청 Servlet으로 지정되어야 하며 이후 이 Servlet은 해당 요청을 웹 컨트롤러로 지정합니다. 웹 컨트롤러는 컨트롤러 명령 및 뷰에 대한 액세스 제어를 구현합니다. 그러나 위에 표시된 웹 경로에서는 악의적인 사용자가 JSP 템플리트(경로 1) 및 엔티티 bean(경로 2)에 직접 액세스할 수 있습니다. 이러한 악의적 공격을 막으려면 이러한 액세스가 런타임에서 거부되어야 합니다.

JSP 템플리트 및 엔티티 bean에 대한 직접 액세스는 다음 접근 방법 중 하나를 사용하여 막을 수 있습니다.

WebSphere Application Server 보안

WebSphere Application Server는 여러 보안 기능을 제공합니다. WebSphere Commerce는 이러한 특성 중 하나를 사용하여 모든 엔터프라이즈 bean 메소드 및 JSP 템플리트가 선택한 동일성에 의해서만 호출되도록 구성되었음을 확인합니다. 이 WebSphere Commerce 자원에 액세스하려면 요청 Servlet으로 URL 요청을 라우트해야 합니다. 요청 Servlet은 선택한 동일성을 웹 컨트롤러로 전달하기 전에 현재 스레드에 설정합니다. 그러면 해당 컨트롤러 명령 또는 뷰로 요청을 전달하기 전에 호출자에게 필요한 권한이 있는지 웹 컨트롤러가 확인합니다. JSP 템플리트 및 엔티티 bean에 직접 액세스하려는 모든 시도(즉, 웹 컨트롤러를 사용하지 않고)는 WebSphere Application Server 보안 구성요소에 의해 거부됩니다.

WebSphere Commerce 자원에 보안을 설정하도록 WebSphere Application Server를 구성하는 정보는 *WebSphere Commerce 보안 안내서*를 참조하십시오. WebSphere Application Server의 보안에 대한 정보는 WebSphere Application Server 문서의 시스템 관리 주제를 참조하십시오.

방화벽 보호

방화벽 뒤에서 WebSphere Commerce Server를 실행하는 경우, 인터넷 클라이언트는 엔티티 bean에 직접 액세스할 수 없습니다. 이 접근 방법을 사용하면 페이지에 포함되어 있는 데이터 bean이 JSP 템플리트에 대한 보호를 제공합니다. 데이터 bean은 데이터 bean 관리자에 의해 활성화됩니다. 데이터 bean 관리자는 JSP 템프리트가 뷰 명령에 의해 전달되었는지 여부를 검출합니다. 뷰 명령에 의해 전달되지 않은 경우 예외가 발생하고 JSP 템플리트에 대한 요청이 거부됩니다.

URL 매개변수에 대한 보안 고려사항

URL 매개변수는 요청 고유 정보를 전달하는 유용한 방법을 제공합니다. 악의적인 사용자가 권한을 부여 받지 않은 상태에서 데이터베이스에 액세스할 수 있는 기회를 최소화하려면 특정 코딩 작업을 수행해야 합니다. SQL문의 Insert, select, update 및 delete 파트를 개발 시점에서 작성해야 합니다. 매개변수 삽입을 사용하여 런타임 입력 정보를 수집해야 합니다.

매개변수 삽입을 사용한 런타임 입력 정보 수집의 예는 다음과 같습니다.

```
select * from Order where owner =?
```

반면 입력 문자열을 사용하여 SQL문을 작성할 수는 없습니다. 입력 문자열을 사용하는 예는 다음과 같습니다.

```
select * from Order where owner = "input_string"
```

입력 문자열을 사용하여 select를 작성하지 못하도록 하는 이유는 입력 문자열이 조작하기 쉽기 때문입니다. 악의적인 사용자가 이러한 입력 문자열을 예상치 못한 값으로 설정하고, 권한이 없는 액세스를 얻거나 데이터베이스에 바람직하지 못한 조작을 수행할 수 있습니다.

WebSphere Commerce 액세스 제어 정책에 대한 소개

이 절에서는 WebSphere Commerce 액세스 제어 프레임워크의 기본 구성요소에 대해 설명합니다. 일부 액세스 제어 관련 프로그래밍 태스크가 올바른 컨텍스트로 표시하기 위해 제공됩니다. 프로덕션 시스템에 액세스 제어를 설정하는 데 대한 정보 또는 액세스 제어 프레임워크에 대한 자세한 정보는 *WebSphere Commerce 보안 안내서*를 참조하십시오.

WebSphere Commerce 액세스 제어 모델은 액세스 제어 정책의 강제 실행을 기반으로 합니다. 액세스 제어 정책을 사용하면 비즈니스 로직 코드에서 액세스 제어 규칙을 외부화할 수 있어 액세스 제어 명령문을 코드로 하드 코딩하지 않아도 됩니다. 예를 들어, 다음과 유사한 코드를 포함하지 않아도 됩니다.

```
if (user.isAdministrator())
   then {}
```

액세스 제어 정책은 액세스 제어 정책 관리자에 의해 강제 실행됩니다. 일반적으로 사용자가 보호된 자원에 액세스하려고 시도하면 먼저 액세스 제어 정책 관리자가 보호된 자원에 적용할 수 있는 액세스 제어 정책을 판별한 후 적용 가능한 액세스 제어 정책에 따라 사용자가 요청된 자원에 액세스할 수 있는지 여부를 판별합니다.

액세스 제어 정책은 ACPOLICY 테이블에 저장되어 있는 4 튜플 정책입니다. 각 액세스 제어 정책의 양식은 다음과 같습니다.

```
AccessControlPolicy [UserGroup, ActionGroup, ResourceGroup, Relationship]
```

4 튜플 액세스 제어 정책의 요소는 사용자가 관계 또는 관계 그룹에 지정되어 있는 해당 자원에 대한 조건을 충족시키는 한, 특정 사용자 그룹에 속하는 사용자가 지정된 자원 그룹에 속하는 자원에 대해 지정된 조치 그룹의 조치를 수행할 수 있는 것으로 지정합니다. 예를 들어, [AllUsers, UpdateDoc, doc, creator]는 모든 사용자가 문서 작성자인 경우, 문서를 업데이트할 수 있음을 지정합니다.

사용자 그룹은 MBRGRP 데이터베이스 테이블에 정의되어 있는 특정 유형의 구성원 그룹입니다. 사용자 그룹은 구성원 그룹 유형 -2와 연관되어야 합니다. -2 값은 액세스 그룹을 표시하며 MBRGRPTYPE 테이블에 정의되어 있습니다. 사용자 그룹과 구성원 그룹 유형 간의 연관은 MBRGRPUSG 테이블에 저장됩니다.

특정 사용자 그룹에 속하는 사용자의 멤버쉽은 명시적 또는 암시적으로 표현되어 있습니다. MBRGRPMBR 테이블에서 사용자가 특정 구성원 그룹에 속하는 것으로 설명하는 경우 명시적 스펙이 나타납니다. 사용자가 MBRGRPCOND 테이블에서 설명하는 조건(예를 들어, 상품 관리자의 역할을 수행하는 모든 사용자)을 충족시키는 경우 암시적 스펙이 나타납니다. 결합 조건(예: 상품 관리자의 역할을 수행하고 이 역할을 최소 6개월 동안 수행해 온 모든 사용자) 또는 명시적 제외 또한 가능합니다.

사용자 그룹에 사용자를 포함시키기 위한 대부분의 조건은 특정 역할을 수행하는 사용자를 기반으로 합니다. 예를 들어, 상품 관리자 역할을 수행하는 모든 사용자가 카탈로그 관리 조작을 수행할 수 있도록 하는 액세스 제어 정책이 있을 수 있습니다. 이런 경우, MBRROLE 테이블에서 상품 관리자 역할로 지정된 모든 사용자는 사용자 그룹에 암시적으로 포함됩니다.

구성원 그룹 서브시스템에 대한 자세한 정보는 WebSphere Commerce Production 및 Development 온라인 도움말을 참조하십시오.

ActionGroup 요소는 ACACTGRP 테이블에 있습니다. 조치 그룹은 명시적으로 지정된 조치 그룹입니다. 조치 목록은 ACACTION 테이블에 저장되고 각 조치와 해당 조치 그룹과의 관계는 ACACTACTGP 테이블에 저장됩니다. 조치 그룹의 한 예는 ″OrderWriteCommands″ 조치 그룹입니다. 이 조치 그룹에는 주문 갱신에 사용되는 다음 조치가 포함됩니다.

- com.ibm.commerce.order.commands.OrderDeleteCmd
- com.ibm.commerce.order.commands.OrderCancelCmd
- com.ibm.commerce.order.commands.OrderProfileUpateCmd
- com.ibm.commerce.order.commands.OrderUnlockCmd
- com.ibm.commerce.order.commands.OrderScheduleCmd
- com.ibm.commerce.order.commands.ScheduledOrderCancelCmd
- com.ibm.commerce.order.commands.ScheduledOrderProcessCmd
- com.ibm.commerce.order.commands.OrderItemAddCmd
- com.ibm.commerce.order.commands.OrderItemDeleteCmd
- com.ibm.commerce.order.commands.OrderItemUpdateCmd
- com.ibm.commerce.order.commands.PayResetPMCmd

자원 그룹은 특정 자원 유형을 함께 그룹화하기 위한 메커니즘입니다. 자원 그룹의 자원 멤버쉽은 다음 두 가지 방법 중 하나로 지정할 수 있습니다.

- ACRESGRP 테이블의 조건 열 사용
- ACRESGPRES 테이블 사용

대부분의 경우, 자원과 자원 그룹을 연관시키는 데는 ACRESGPRES 테이블을 사용하면 됩니다. 이 방법을 사용하는 경우, 자원은 해당 Java 클래스 이름을 사용하여 ACRESCGRY 테이블에 정의됩니다. 그런 다음, 이 자원은 ACRESGPRES 연관 테이블을 사용하여 적절한 자원 그룹(ACRESGRP 테이블)과 연관됩니다. Java 클래스 이름만으로 자원 그룹의 구성원을 정의할 수 없는 경우(예를 들어, 자원의 속성에 따라 이 클래스의 오브젝트를 더 제한해야 하는 경우), ACRESGRP 테이블의 조건 열을 사용하여 자원 그룹을 완전하게 정의할 수 있습니다. 속성을 기반으로 이 자원 그룹화를 수행하려면 자원이 Groupable 인터페이스 또한 구현해야 함에 유의하십시오.

다음 도표는 자원 그룹화 스펙 예를 표시합니다. 이 예에서 자원 그룹 10023은 ACRESGPRES 테이블의 해당 자원 그룹과 연관되는 모든 자원을 포함합니다. 자원 그룹 10070은 ACRESGRP 테이블의 조건 필드 열을 사용하여 정의됩니다. 이 자원 그룹은 Order 원격 인터페이스의 인스턴스를 포함하며 상태 또한 "Z"(공용 요청 목록 지정)입니다.

주: ACRESGRP 테이블 조건 열의 XML 정보에 대한 자세한 내용은 *WebSphere Commerce 보안 안내서*에 있습니다.

ACRESGRP

AcResGrp_Id	**Grp**이름	조건
10023	AccountRepresentatives CmdResourceGroup	null
10070	SharedRequisitionList ResourceGroup	<profile> <andListCondition> <simpleCondition> <variable name="Status"/> <operator name="="/> <value data="Z"/> </simpleCondition> <simpleCondition> <variable name="classname"/> <operator name="="/> <value data="com.ibm.commerce.order.objects.Order"/> </simpleCondition> </andListCondition> </profile>

ACRESGRPES

AcResGrp_Id	**AcResCgry_Id**
10023	10246
10023	10247
10023	10248
10023	10249
10023	10250

ACRESCGRY

AcResCgry_Id	해당 클래스 이름
10246	com.ibm.commerce.contract.commands.ContractCreateCmd
10247	com.ibm.commerce.contract.commands.ContractUpdateCmd
10248	com.ibm.commerce.contract.commands.ContractDeleteCmd
10249	com.ibm.commerce.contract.commands.ContractCancelCmd
10250	com.ibm.commerce.contract.commands.ContractCloseCmd

그림 20.

ACACTGRP, ACRESGRP 및 ACRELGRP 테이블의 MEMBER_ID 열에는 -2001 값(루트 조직)이 있어야 합니다.

액세스 제어 정책은 선택적으로 Relationship 또는 RelationshipGroup 요소를 네 번째 요소로 포함할 수 있습니다.

액세스 제어 정책이 Relationship 요소를 사용하는 경우, 이 요소는 ACRELATION 테이블에 있습니다. 반면에 RelationshipGroup 요소를 포함하는 경우, 이 요소는 ACRELGRP 테이블에 있습니다. 두 요소 모두 포함하지 않아도 되지만, 하나를 포함하는 경우 나머지를 포함할 수 없습니다. ACRELGRP 테이블의 RelationshipGroup 스펙은 ACRELATION 테이블의 Relationship 정보에 우선합니다.

ACRELATION 테이블은 사용자와 자원 간에 존재하는 관계 유형을 지정합니다. 몇몇 관계 유형 예에는 작성자, 제출자 및 소유자가 포함됩니다. Relationship 요소 사용의 한 예는 이 요소를 사용하여 주문 작성자가 언제나 주문을 갱신할 수 있도록 하는 것입니다.

ACRELGRP 테이블은 특정 자원과 연관될 수 있는 관계 그룹의 유형을 지정합니다. 관계 그룹은 하나 이상의 관계 체인의 그룹화입니다. 관계 체인은 하나 이상의 관계의 시리즈입니다. 관계 그룹의 한 예는 사용자가 자원 작성자이자 자원에서 참조되는 구매 조직 엔티티에 속해야 하는 것으로 지정하는 것입니다.

관계 그룹(또는 관계) 스펙은 액세스 제어 정책의 선택 파트입니다. 일반적으로 사용자 고유 명령을 작성하고 이 명령이 특정 역할로 제한되지 않는 경우 사용됩니다. 이런 경우, 사용자와 자원 간에 관계를 강제 실행할 수 있습니다. 일반적으로 명령이 특정 역할로 제한되는 경우, Relationship 요소를 사용하지 않고 액세스 제어 정책의 UserGroup 요소를 통해 설정됩니다.

다른 중요 액세스 제어 정책 관련 개념은 액세스 제어 정책 그룹의 개념입니다. WebSphere Commerce 버전 5.5는 다양한 비즈니스 모델을 지원하며 각 비즈니스 모델에는 고유한 액세스 제어 정책 세트가 있습니다. 모델의 정책 세트를 그룹화하기 위해 정책 그룹이 사용됩니다. 정책은 적절한 정책 그룹에 명시적으로 지

정되고, 이후 조직은 이 정책 그룹 중 하나 이상에 등록될 수 있습니다. 이전 WebSphere Commerce 버전의 경우, 모든 자원에 적용된 정책은 해당 정책의 소유자 조직의 하위가 소유했습니다.

WebSphere Commerce 버전 5.5에서 조직이 하나 이상의 정책 그룹에 등록되는 경우, 해당 정책 그룹의 정책만이 해당 조직의 자원에 적용됩니다. 어느 정책 그룹에도 등록하지 않은 조직이 자원을 소유하는 경우, 액세스 제어 정책 관리자가 최소 하나의 정책 그룹에 등록하는 가장 근접한 상위 조직을 발견할 때까지 조직 계층을 검색합니다. 조직을 찾으면 해당 정책 그룹에 속하는 정책을 적용합니다.

다음 도표를 참조하십시오.

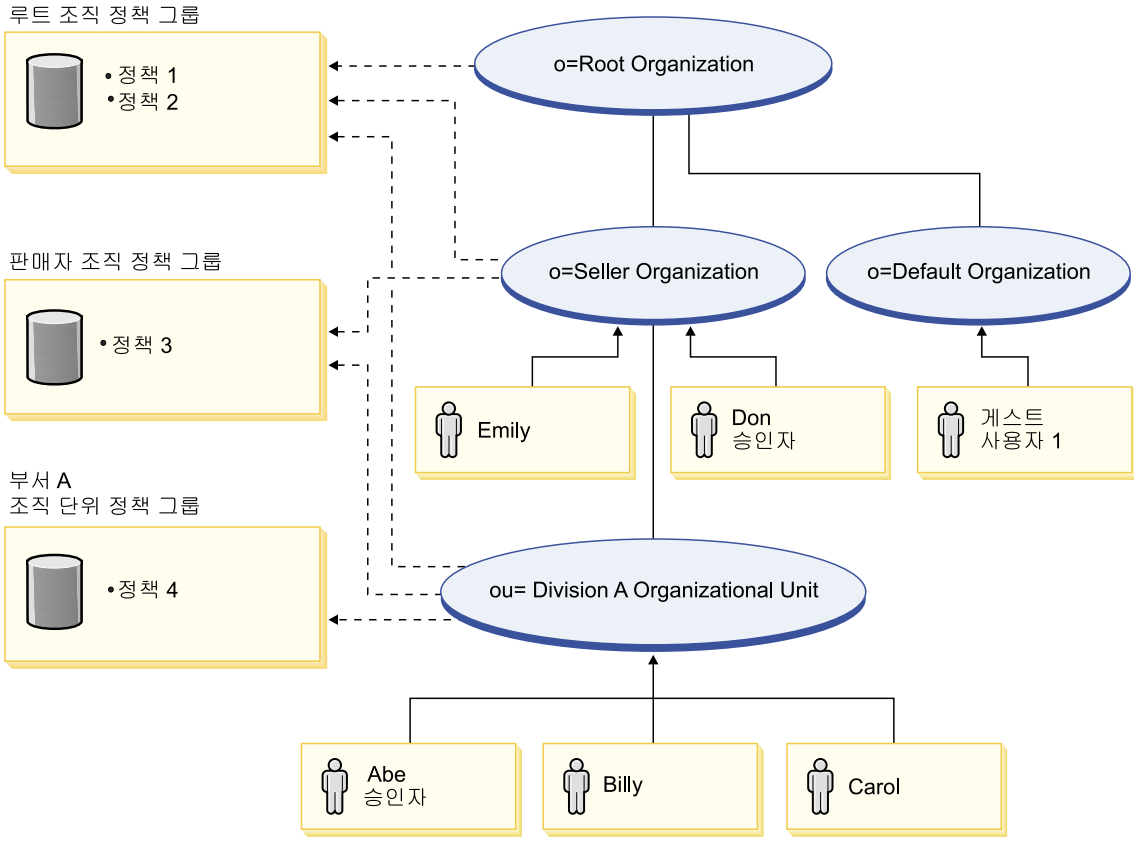


그림 21.

판매자 조직이 소유하는 자원에 대해 판매자 조직 정책 그룹 및 루트 조직 정책 그룹의 정책이 적용됩니다. 판매자 조직이 해당 두 정책 그룹에 명시적으로 등록하므로 이 정책을 적용할 수 있습니다(점선 화살표는 등록을 표시함). 총 세 개의 정책이 이 조직에 적용됩니다. 이 예는 자원을 소유한 조직이 정책 그룹에 명시적으로 등록된 경우입니다. 도표는 또한 조직이 어느 정책 그룹에도 명시적으로 등록되지 않아 액세스 제어 프레임워크가 계층을 더 검색해야 하는 경우의 예를 표시합니다. 기본 조직이 소유하는 자원의 경우, 루트 조직까지 계층을 검색해야 합니다. 이 루트 조직은 하나의 정책 그룹에만 등록되므로, 해당 정책(정책 1 및 2)만이 기본 조직의 자원에 적용됩니다.

관계 그룹

관계 그룹을 사용하여 복수 그룹을 지정할 수 있습니다. 관계는 사용자와 문제가 있는 자원간의 직접 관계이거나 사용자와 자원의 관계를 간접적으로 설정하는 관계 체인일 수 있습니다.

주: 관계 그룹과 관련된 다음 절의 경우, WebSphere Commerce Professional Edition에서 사용 가능한 유일한 조직이 루트 조직, 기본 조직 및 판매자 조직임을 인식하는 것이 중요합니다. 기타 조직을 참조하는 예는 WebSphere Commerce Business Edition에만 적용됩니다.

관계와 관계 그룹 비교: 액세스 제어 정책은 사용자가 액세스되는 자원과의 특정 관계를 수행해야 하는 것으로 지정하거나 사용자가 관계 그룹에 지정되어 있는 조건을 수행해야 하는 것으로 지정할 수 있습니다.

대부분의 경우, 관계 지정은 응용프로그램에 대한 액세스 제어 요구사항을 충족시켜야 합니다. 그러나 사용자가 사용자와 자원 간의 간접 관계를 지정해야 하지만 실제로 사용자와 자원 간의 일련의 관계가 있는 정책의 경우, 관계 그룹을 사용해야 합니다.

예를 들어, 사용자와 구매 조직 간에 사용자가 해당 조직에 대한 특정 역할을 수행하거나 사용자가 구매 조직의 구성원이어야 하는 연관을 지정해야 하는 경우, 관계 그룹 및 관계 체인을 사용해야 합니다.

사용자와 문제가 있는 자원 간의 직접 연관만 강제 실행해야 하는 경우, 단순 관계를 사용할 수 있습니다. 예를 들어, 사용자가 자원의 작성자가 되어야 하는 것으로 강제 실행해야 하는 경우입니다.

복수 단순 관계를 결합하고 예를 들어, 사용자가 작성자 또는 제출자여야 하는 경우, 이는 관계 체인이 되며 사용자는 관계 그룹을 사용해야 합니다. 단순 관계의 이러한 조합은 WebSphere Commerce Professional Edition 또는 WebSphere Commerce Business Edition 사용 시 발생할 수 있습니다.

관계 그룹에 대한 일반 정보: 관계 체인은 하나 이상의 관계의 시리즈입니다. 관계 체인의 길이는 포함되는 관계의 수에 따라 결정됩니다. 이 길이는 관계 체인의 XML 표시에 있는 `<parameter name="aName" value="aValue" />` 요소의 수를 점검하여 판별할 수 있습니다.

마지막 `<parameter name="Relationship" value="aValue" />` 요소만 자원의 fulfills() 메소드로 처리할 수 있습니다. 나머지는 액세스 제어 정책 관리자가 내부적으로 처리합니다.

관계 체인의 길이가 2인 경우, 첫 번째 `<parameter name="aName" value="aValue" />` 요소는 사용자와 조직 엔티티 사이에 있습니다. 마지막 `<parameter name="aName" value="aValue" />` 요소는 조직 엔티티와 자원 사이에 있습니다.

관계 그룹을 정의해야 하는 경우, 관계 그룹 정보를 XML 파일로 정의하여 이를 수행해야 합니다. `defaultAccessControlPolicies.xml` 파일을 기초로 사용자 고유의 XML 파일을 작성할 수 있습니다. 이 XML 기반 정보 작성에 대한 추가 정보는 *WebSphere Commerce 보안 안내서*를 참조하십시오.

액세스 제어 유형

액세스 제어 유형은 명령 레벨 액세스 제어 및 자원 레벨 액세스 제어의 두 가지로서 모두 정책을 기반으로 합니다.

명령 레벨(“역할 기반”이라고도 함) 액세스 제어는 다양한 정책 유형을 사용합니다. 특정 역할의 모든 사용자가 특정 명령 유형을 실행할 수 있도록 지정할 수 있습니다. 예를 들어, 계정 담당 역할의 사용자가

AccountRepresentativesCmdResourceGroup 자원 그룹의 모든 명령을 실행할 수 있도록 지정할 수 있습니다. 또는 다음 도표에서 설명하는 것처럼 또 다른 정책 예는 모든 상점 운영자가 StoreAdminCmdResourceGrp이 지정하는 자원에 ExecuteCommandActionGroup에 지정되어 있는 조치를 수행할 수 있도록 지정하는 것입니다.

주: MBRGRPCOND 테이블의 조건 열에 대한 XML 정보는 관리 콘솔을 사용하여 액세스 그룹을 설정할 때 작성됩니다. 관리 콘솔을 사용한 액세스 그룹 설정에 대한 정보는 WebSphere Commerce Production 온라인 도움말을 참조하십시오.

ACPOLICY

정책 이름	Member_Id	MbrGrp_Id	AcActGrp_id	AcResGrp_Id	AcRelGrp_Id
StoreAdministrators ExecuteStoreAdministrators CmdResourceGroup	-2001	-8	10052	10018	null

MBRGRP

MbrGrp_Id	MbrGrp이름
-8	StoreAdministrators

MBRGRPCOND

MbrGrp_Id	조건
-8	<profile> <simpleCondition> <variable name="role"/> <operator name="="/> <value data="Store Administrator"/> </simpleCondition> </profile>

ACACTGRP

AcActGrp_Id	그룹 이름
10052	ExecuteCommandActionGroup

ACRESGRP

AcResGrp_Id	그룹 이름
10018	StoreAdministratorsCmdResourceGroup

그림 22.

명령 레벨 액세스 제어 정책은 언제나 `ExecuteCommandActionGroup`을 컨트롤러 명령의 조치 그룹으로 갖습니다. 뷰의 경우, 자원 그룹은 언제나 `ViewCommandResourceGroup`입니다.

모든 컨트롤러 명령은 명령 레벨 액세스 제어로 보호해야 합니다. 또한 직접 호출할 수 있거나 다른 명령의 경로 재지정으로 실행할 수 있는(뷰로 전달하여 실행하는 것과 반대로) 뷰는 명령 레벨 액세스 제어로 보호해야 합니다.

명령 레벨 액세스 제어는 명령이 조치를 수행할 자원을 고려하지 않습니다. 사용자가 특정 명령을 실행할 수 있는지 여부만을 판별합니다. 사용자가 명령을 실행할 수 있는 경우, 후속 자원 레벨 액세스 제어 정책을 적용하여 사용자가 해당 자원에 액세스할 수 있는지 여부를 판별할 수 있습니다.

상점 운영자가 관리 태스크를 수행하려고 시도하는 시기를 고려하십시오. 액세스 제어 확인의 첫 번째 레벨은 이 사용자가 특정 상점 관리 명령을 실행할 수 있는지 여부를 판별하는 것입니다. 사용자가 실제적으로 이를 수행할 수 있는 것으로 판별되면(상점 운영자가 storeAdminCmds 그룹에서 명령을 실행할 수 있으므로), 자원 레벨 액세스 제어 정책을 호출할 수 있습니다. 이 정책은 상점 운영자만이 사용자가 상점 운영자인 조직이 소유하는 상점에 대한 관리 태스크를 수행할 수 있는 것으로 설명합니다.

즉, 명령 레벨 액세스 제어의 경우, “자원”이 명령이며 “조치”는 단지 명령을 실행하는 것입니다(즉, 명령 오브젝트의 인스턴스 작성). 액세스 제어 확인은 사용자가 명령을 실행할 수 있는지 여부를 판별합니다. 반대로 자원 레벨 액세스 제어의 경우, “자원”은 명령 또는 bean이 액세스하는 임의의 보호 가능 자원이며 “조치”가 명령입니다.

액세스 제어 상호작용

이 절에서는 WebSphere Commerce 액세스 제어 정책 프레임워크에서 액세스 제어의 작동 방법을 표시하는 상호작용 도표에 대해 설명합니다.

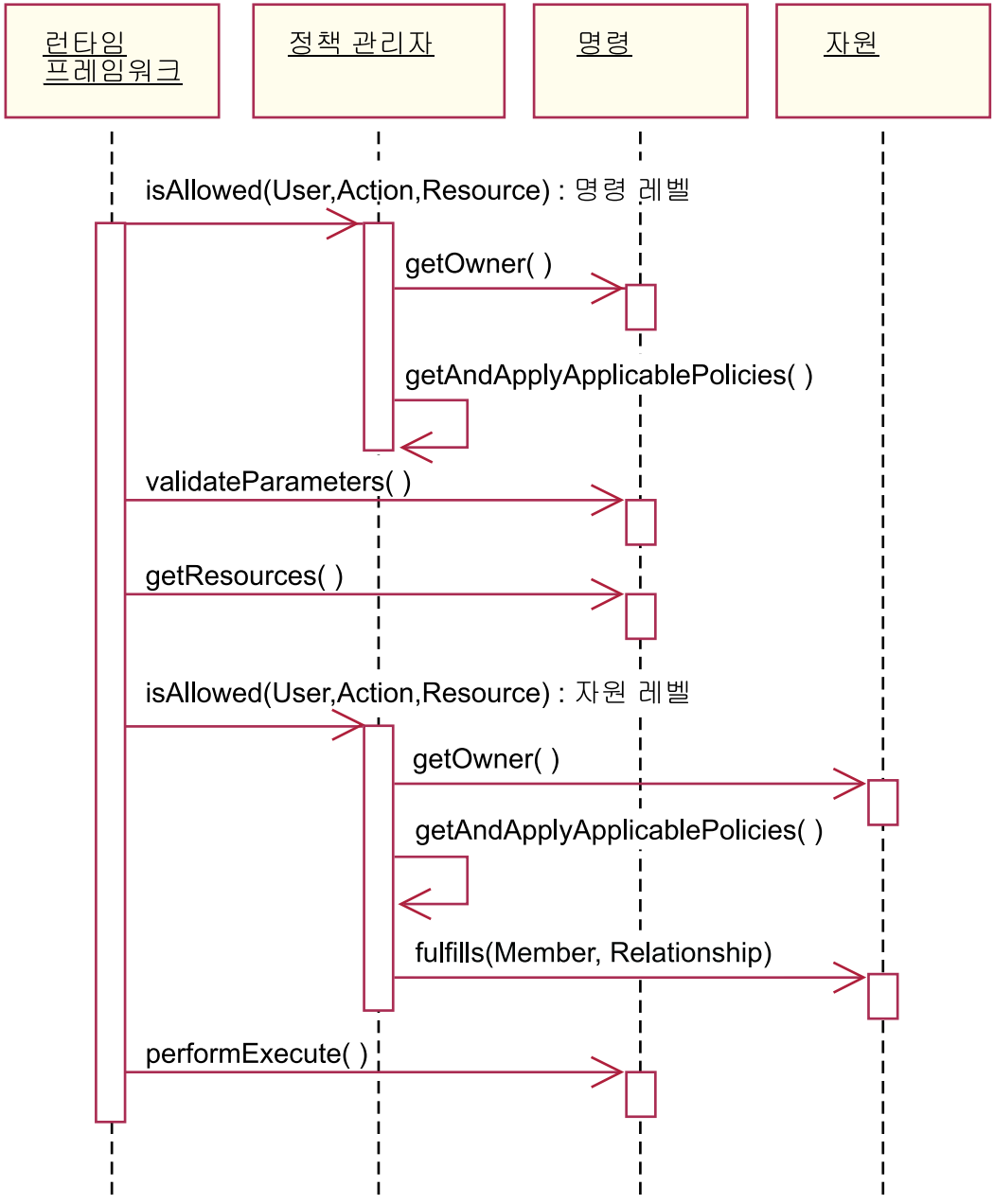


그림 23.

앞의 도표는 액세스 제어 *정책 관리자*가 수행하는 조치를 표시합니다. 액세스 제어 정책 관리자는 현재 사용자가 지정된 자원에 대해 지정된 조치를 실행할 수 있

는지 여부를 판별하는 액세스 제어 구성요소입니다. 자원의 소유자가 등록한 그룹의 정책을 검색하여 이를 판별합니다. 자원의 소유자가 어떤 정책 그룹에도 등록하지 않은 경우, 자원 소유자와 가장 가까운 이전 소유자가 등록한 그룹의 정책을 검색합니다. 하나 이상의 정책이 액세스를 부여하는 경우, 권한이 부여됩니다.

다음 목록은 앞의 상호작용 도표의 조치에 대해 설명합니다. 이 조치는 도표 맨 위에서 맨 아래까지 정렬되어 있습니다.

1. `isAllowed()`
   런타임 구성요소는 사용자에게 컨트롤러 명령 또는 뷰에 대한 명령 레벨 액세스 권한이 있는지 여부를 판별합니다.
2. `getOwner()`
   액세스 제어 정책 관리자는 명령 레벨 자원의 소유자를 판별합니다. 기본 구현은 명령 컨텍스트에 있는 상점 소유자(`storeId`)의 구성원 식별자(`memberId`)를 리턴합니다. 명령 컨텍스트에 상점 식별자가 없는 경우, 루트 조직(`-2001`)이 리턴됩니다.
3. `getAndApplyApplicablePolicies()`
   액세스 제어 정책 관리자가 지정된 사용자, 조치 및 자원을 기반으로 적용 가능한 정책을 찾아 처리합니다. 최소 하나의 적용 가능한 정책이 액세스를 부여하는 경우, 명령 레벨 액세스 제어 확인이 전달되고, 정책 관리자가 자원 레벨 권한에 대한 확인을 시작하는 다음 단계를 계속 진행합니다. 반대로 적용 가능한 정책이 명령 레벨 액세스를 부여하지 않는 경우 이 시점에서 정책 관리자가 리턴되고 액세스를 거부합니다.
4. `validateParameters()`
   초기 매개변수를 확인 및 분석합니다.
5. `getResources()`
   자원 조치 쌍의 벡터인 액세스 벡터를 리턴합니다.

   아무 것도 리턴되지 않는 경우, 자원 레벨 액세스 제어 확인이 수행되지 않습니다. 보호해야 하는 자원이 있는 경우, 액세스 벡터(자원 조치 쌍으로 구성)를 리턴해야 합니다.

   각 *자원*은 보호 가능 오브젝트(`com.ibm.commerce.security.Protectable` 인터페이스를 구현하는 오브젝트)의 인스턴스입니다. 많은 경우, 자원은 액세스 bean입니다.

액세스 bean은 com.ibm.commerce.security.Protectable 인터페이스를 구현하지 않을 수 없습니다. 그러나 125 페이지의 『엔터프라이즈 bean에서 액세스 제어 구현』에 포함되어 있는 정보에 따라, 해당 엔터프라이즈 bean이 보호되는 한 액세스 제어 확인은 계속 이루어질 수 있습니다.

*조치*는 자원에 수행될 조작을 나타내는 문자열입니다. 대부분의 경우, 조치는 명령의 인터페이스 이름입니다.

6. isAllowed()
   런타임 구성요소는 사용자에게 getResources()로 지정되는 모든 자원 조치 쌍에 대한 자원 레벨 액세스 권한이 있는지 여부를 판별합니다.
7. getOwner()
   자원이 해당 소유자의 memberId를 리턴합니다. 이는 적용할 정책을 판별합니다.
8. getAndApplyApplicablePolicies()
   액세스 제어 정책 관리자가 적용 가능한 정책을 검색한 후 적용합니다. 자원에 액세스할 수 있는 사용자 권한을 부여하는 정책이 자원 조치 쌍에 하나라도 있는 경우 액세스가 부여되고, 그렇지 않은 경우 액세스가 거부됩니다.
9. fulfills()
   적용 가능한 정책이 관계 또는 관계 그룹을 지정한 경우, 구성원이 자원과의 지정된 관계를 충족시키는지 여부에 대해 자원을 확인합니다.
10. performExecute()
    명령의 비즈니스 로직입니다.

Protectable 인터페이스

WebSphere Commerce 액세스 제어 정책으로 자원을 보호하기 위한 주요 요인은 자원이 com.ibm.commerce.security.Protectable 인터페이스를 구현해야 하는 것입니다. 이 인터페이스는 일반적으로 엔터프라이즈 bean 및 데이터 bean과 함께 사용되지만, 보호가 필요한 특정 bean만이 인터페이스를 구현해야 합니다.

Protectable 인터페이스의 경우, 자원은 두 가지 키 메소드, 즉 getOwner() 및 fulfills(Long member, String relationship)을 제공해야 합니다.

액세스 제어 정책은 조직 또는 조직 엔티티가 소유합니다. getOwner 메소드는 보호 가능 자원 소유자의 memberId를 리턴합니다. 액세스 제어 정책 관리자가 자원 소유자를 판별하고 나면 구성원 계층에 있는 소유자의 각 상위의 memberId 또한 가져옵니다. 그러면 원래 getOwner 요청의 소유자에 속하는 모든 액세스 제어 정책 및 소유자의 상위에 속하는 모든 액세스 제어 정책이 적용됩니다.

지정된 소유자에 적용되는 액세스 제어 정책 및 구성원 계층에서 소유자의 상위 레벨 상위에 적용되는 액세스 제어 정책이 적용됩니다.

해당 구성원이 자원에 대한 필수 관계를 충족시키는 경우 fulfills 메소드는 TRUE만을 리턴합니다. 일반적으로 구성원은 단일 사용자이나, 조직일 수도 있습니다. 액세스 제어 정책에서 관계 그룹을 사용하는 경우 조직이 됩니다.

Groupable 인터페이스

액세스 제어 정책의 응용프로그램은 자원 그룹에 고유합니다. 자원 그룹화는 클래스 이름, 주문 상태 또는 storeId 값과 같은 속성을 기반으로 작성할 수 있습니다.

액세스 제어 정책을 적용하기 위해 해당 클래스 이름이 아닌 속성으로 자원을 그룹화하는 경우, com.ibm.commerce.grouping.Groupable 인터페이스를 구현해야 합니다.

다음 코드 단편은 Groupable 인터페이스를 나타냅니다.

```
Groupable interface {
 Object getGroupingAttributeValue (String attributeName, GroupContext context)
}
```

예를 들어, 보류 상태(status = P (pending))인 주문에만 적용되는 정책을 구현하기 위해, Order 엔티티 bean의 원격 인터페이스가 Groupable 인터페이스를 구현하고 attributeName의 값은 "status"로 설정됩니다.

Groupable 인터페이스를 사용하는 경우는 드뭅니다.

액세스 제어에 대한 추가 정보 찾기

WebSphere Commerce 액세스 제어 모델에 대한 추가 정보는 *WebSphere Commerce 보안 안내서*를 참조하십시오. 이 안내서는 액세스 제어에 대한 자세한 개요를 제공하고 관리 콘솔을 사용한 정책, 조치 그룹 및 자원 그룹의 작성 또는 수정 방법에 대해 설명합니다.

액세스 제어 구현

이 절에서는 사용자 정의 코드로 액세스 제어를 구현하는 방법에 대해 설명합니다.

보호 가능 자원 식별

일반적으로 엔터프라이즈 bean 및 데이터 bean은 보호할 수 있는 자원입니다. 그러나 모든 엔터프라이즈 bean 및 데이터 bean을 보호해야 하는 것은 아닙니다. 기존 WebSphere Commerce 응용프로그램의 경우, 보호가 필요한 자원은 이미 Protectable 인터페이스를 구현합니다. 일반적으로 보호 대상의 문제는 새 엔터프라이즈 bean 및 데이터 bean 작성 시 결정합니다. 보호할 자원의 결정은 응용프로그램에 따라 다릅니다.

명령이 `getResources` 메소드에 엔터프라이즈 bean을 리턴하는 경우, 액세스 제어 정책 관리자가 엔터프라이즈 bean에 getOwner 메소드를 호출하므로 엔터프라이즈 bean을 보호해야 합니다. 해당 자원 레벨 액세스 제어 정책에 관계가 지정되는 경우에도 `fulfills` 메소드가 호출됩니다.

모든 사용자 고유 엔터프라이즈 bean 및 데이터 bean에 Protectable 인터페이스를 구현하려는 경우(따라서 자원을 보호하려는 경우), 응용프로그램이 많은 정책을 필요로 할 수 있습니다. 정책 수가 늘어나면 성능이 저하되고 정책 관리가 더 어려워집니다.

기본 자원과 종속 자원은 이론적인 차이가 있습니다. *기본 자원*은 스스로 존재할 수 있습니다. *종속 자원*은 관련 기본 자원이 존재하는 경우에만 존재합니다. 예를 들어, 초기 WebSphere Commerce 응용프로그램 코드에서 Order 엔티티 bean은 보호 가능 자원이지만 OrderItem 엔티티 bean은 그렇지 않습니다. 그 이유는

OrderItem의 존재가 Order에 종속되기 때문입니다. 즉, Order는 기본 자원이고 OrderItem은 종속 자원입니다. 사용자가 주문에 액세스해야 하는 경우, 주문 내 항목에도 액세스해야 합니다.

마찬가지로 User 엔티티 bean은 보호 가능 자원이지만 Address 엔티티 bean은 그렇지 않습니다. 이런 경우 주소의 존재는 사용자에 종속되므로, 사용자에 액세스할 수 있는 것은 주소에도 액세스할 수 있어야 합니다.

기본 자원은 보호해야 하지만 종속 자원은 보호가 필요하지 않은 경우도 있습니다. 사용자가 기본 자원에 액세스할 수 있는 경우, 기본적으로 사용자가 해당 종속 자원에도 액세스할 수 있어야 하는 것이 타당합니다.

엔터프라이즈 bean에서 액세스 제어 구현

액세스 제어 정책에 의한 보호가 필요한 새 엔터프라이즈 bean을 작성하는 경우, 다음을 수행해야 합니다.

1. 새 엔터프라이즈 bean을 작성하고 `com.ibm.commerce.base.objects.ECEntityBean`에서 확장되었는지 확인하십시오.
2. bean의 원격 인터페이스가 `com.ibm.commerce.security.Protectable` 인터페이스를 확장하는지 확인하십시오.
3. 액세스 제어 정책을 적용하기 위해 해당 Java 클래스 이름이 아닌 속성으로 자원을 그룹화하는 경우, bean의 원격 인터페이스는 `com.ibm.commerce.grouping.Groupable` 인터페이스 또한 확장해야 합니다.
4. 엔터프라이즈 bean 클래스는 com.ibm.commerce.base.objects.ECEntityBean에서 다음 메소드의 기본 구현을 상속합니다.
   - `getOwner`
   - `fulfills`
   - `getGroupingAttributeValue`

   필요한 메소드를 대체하십시오. 최소한 `getOwner` 메소드를 대체해야 합니다.

   자원 그룹에 이 자원을 포함하는 액세스 제어 정책이 있는 경우 fulfills 메소드를 구현해야 하며 관계 또는 관계 그룹 또한 지정해야 합니다. 액세스 제어 정책에 이 자원의 특정 인스턴스를 포함하는 암시적 자원 그룹이 있는 경우,

특정 속성 값에 따라 getGroupingAttributeValue 메소드를 구현해야 합니다 (예를 들어, status = 'P'(보류)인 주문에만 관련되는 액세스 정책이 있는 경우).

"소유자" 관계만 필요한 경우 fulfills 메소드를 대체하지 않아도 됨에 유의하십시오. 이런 경우, 정책 관리자는 getOwner() 메소드의 결과를 사용합니다.

이러한 메소드의 기본 구현은 다음 코드 단편에 표시됩니다. 이 구현은 ECEntityBean 클래스에서 이루어집니다.

```
*****************************************************************************
public Long getOwner() throws Exception
{
return null;
}
*****************************************************************************
*****************************************************************************
public boolean fulfills(Long member, String relationship)
   throws Exception
{
      return false;
}
*****************************************************************************
*****************************************************************************
public Object getGroupingAttributeValue(String attributeName,
   GroupingContext context) throws Exception
{
return null;
}
*****************************************************************************
```

다음은 OrderBean bean에서 사용하는 구현에 따른 이 메소드의 견본 구현입니다.

- getOwner 메소드의 경우, 해당 메소드의 로직은 다음과 같습니다.

```
   *************************************************************************
      com.ibm.commerce.common.objects.StoreEntityAccessBean storeEntAB =
      new com.ibm.commerce.common.objects.StoreEntityAccessBean();
      storeEntAB.setInitKey_storeEntityId(getStoreEntityId().toString());
      return storeEntAB.getMemberIdInEJBType();
   *************************************************************************
```

- fulfills 메소드의 경우, 해당 메소드의 로직은 다음과 같습니다.

```
   *************************************************************************
   if ("creator".equalsIgnoreCase(relationship))
   {
      return member.equals(bean.getMemberId());
   }
```

```
else if ("BuyingOrganizationalEntity".equalsIgnoreCase(relationship))
{
   return (member.equals(bean.getOrganizationId()));
}
else if ("sameOrganizationalEntityAsCreator".
   equalsIgnoreCase(relationship))
{
   com.ibm.commerce.user.objects.UserAccessBean creator =
      new com.ibm.commerce.user.objects.UserAccessBean();
   creator.setInitKey_MemberId(bean.getMemberId().toString());
   com.ibm.commerce.user.objects.UserAccessBean ab =
      new com.ibm.commerce.user.objects.UserAccessBean();
            ab.setInitKey_MemberId(member.toString());
            if (ab.getParentMemberId().equals(creator.getParentMemberId()))
         return true;
}
      return false;
*****************************************************************************
```

- getGroupingAttributeValue 메소드의 경우, 해당 메소드의 로직은 다음과 같습니다.

```
*****************************************************************************
   if (attributeName.equalsIgnoreCase("Status"))
       return getStatus();
return null;
 *****************************************************************************
```

5. 엔터프라이즈 bean의 액세스 bean 및 작성 코드를 작성(또는 다시 작성)하십시오.

getOwner, fulfills 및 getGroupingAttributeValue 메소드의 구현 방법을 이해하기 위해 WebSphere Commerce 공개 엔티티 bean을 점검하는 경우, 이들 메소드가 bean의 액세스 헬퍼 클래스에서 구현됨을 알게 됩니다. 메소드가 bean 클래스에서 직접 구현되지 않고 액세스 헬퍼 클래스에서 구현되기 때문에 메소드 서명은 약간 다릅니다. 특히 오브젝트가 액세스 헬퍼로 자동 전달하기 위한 별도의 입력 매개변수를 메소드가 취하기 때문입니다.

새 bean을 작성할 때는 *반드시* 이 메소드를 bean 클래스에서 직접 구현해야 합니다. 또한 WebSphere Commerce 공개 엔티티 bean의 액세스 헬퍼 클래스에서 이 메소드를 수정해서는 안 됩니다.

데이터 bean에서 액세스 제어 구현

데이터 bean을 보호하려는 경우, 액세스 제어 정책으로 직접 또는 간접 보호할 수 있습니다. 데이터 bean을 직접 보호하는 경우, 해당 특정 데이터 bean에 적용되는

액세스 제어 정책이 존재합니다. 데이터 bean을 간접 보호하는 경우, 액세스 제어 정책이 존재하는 다른 데이터 bean으로 보호를 위임합니다.

데이터 bean을 보호해야 하는지 여부를 판별하려면 다음을 고려하십시오.

1. 데이터 bean에 보호해야 할 정보가 있습니까? 예를 들어, 개인적인 내용입니까? 그렇지 않은 경우 액세스 제어에 의한 직접 보호는 필요하지 않습니다. 예인 경우 계속하십시오.
2. 뷰에서 데이터 bean의 인스턴스를 작성합니다. 뷰가 명령 컨텍스트의 정보(또는 사전 결정된 일부 기타 정보)를 사용하여 데이터 bean의 인스턴스를 작성합니까? 예이고 액세스 제어가 이미 액세스를 허용하기로 결정한 경우, 이 데이터 bean을 직접 보호하지 않아도 됩니다. 아니오인 경우 계속하십시오.
3. 뷰가 일부 입력 매개변수의 정보를 사용하여 데이터 bean의 인스턴스를 작성합니까? 이런 경우, 사용자가 이 정보에 액세스할 수 있는지 여부를 액세스 제어가 이미 결정했는지 알 수 없으므로 새 데이터 bean을 보호해야 합니다.

액세스 제어 정책이 직접 보호할 새 데이터 bean을 작성하는 경우, 데이터 bean은 다음을 수행해야 합니다.

1. `com.ibm.commerce.security.Protectable` 인터페이스 구현. 또한 bean은 `getOwner()` 및 `fulfills(Long member, String relationship)` 메소드를 구현해야 합니다.

   데이터 bean이 Protectable 인터페이스를 구현하면 데이터 bean 관리자가 `isAllowed` 메소드를 호출하여 기존 액세스 제어 정책에 따라 사용자에게 적절한 액세스 제어 권한이 있는지 판별합니다. `isAllowed` 메소드는 다음 코드 단편으로 설명합니다.

   ```
   isAllowed(Context, "Display", protectable_databean);
   ```

   여기서, *protectable_databean*은 보호될 데이터 bean입니다.
2. bean과 상호작용하는 자원이 자원의 Java 클래스 이름이 아닌 속성으로 그룹화되는 경우, bean은 `com.ibm.commerce.grouping.Groupable` 인터페이스를 구현해야 합니다.
3. `com.ibm.commerce.security.Delegator` 인터페이스를 구현하십시오. 이 인터페이스는 다음 코드 단편으로 설명합니다.

```
Interface Delegator {
 Protectable getDelegate();
}
```

주: 직접 보호를 받으려면 getDelegate 메소드가 데이터 bean을 리턴해야 합니다. 즉, 데이터 bean이 액세스 제어를 위해 위임됩니다.

직접 보호해야 하는 데이터 bean과 간접 보호해야 하는 데이터 bean의 차이는 기본 자원과 종속 자원의 차이와 유사합니다. 데이터 bean 오브젝트가 독립적으로 존재할 수 있는 경우 직접 보호해야 합니다. 데이터 bean의 존재가 다른 데이터 bean의 존재에 종속되는 경우, 보호를 위해 다른 데이터 bean으로 위임해야 합니다.

직접 보호하게 될 데이터 bean의 한 예는 Order 데이터 bean입니다. 간접 보호하게 될 데이터 bean의 한 예는 OrderItem 데이터 bean입니다.

액세스 제어 정책이 간접 보호할 새 데이터 bean을 작성하는 경우, 데이터 bean은 다음을 수행해야 합니다.

1. com.ibm.commerce.security.Delegator 인터페이스를 구현하십시오. 이 인터페이스는 다음 코드 단편으로 설명합니다.

   ```
   Interface Delegator {
    Protectable getDelegate();
   }
   ```

   주: getDelegate에 의해 리턴되는 데이터 bean은 Protectable 인터페이스를 구현해야 합니다.

데이터 bean이 위임자 인터페이스를 구현하지 않는 경우, 액세스 제어 정책의 보호 없이 대량 자료가 반입됩니다.

컨트롤러 명령에서 액세스 제어 구현

새 컨트롤러 명령 작성 시, 새 명령의 구현 클래스는 com.ibm.commerce.commands.ControllerCommandImpl 클래스를 확장하고 해당 인터페이스는 com.ibm.commerce.command.ControllerCommand 인터페이스를 확장해야 합니다.

컨트롤러 명령의 명령 레벨 정책의 경우, 명령의 인터페이스 이름은 자원으로 지정됩니다. 자원을 보호하려면 Protectable 인터페이스를 구현해야 합니다. 이는 WebSphere Commerce 프로그래밍 모델에 따라 명령의 인터페이스가 com.ibm.commerce.command.ControllerCommand 인터페이스에서 확장되고 명령의 구현이 com.ibm.commerce.command.ControllerCommandImpl에서 확장되도록 함으로써 완성됩니다. ControllerCommand 인터페이스는 com.ibm.commerce.command.AccCommand 인터페이스를 확장하고 그 다음 Protectable을 확장합니다. AccCommand 인터페이스는 명령 레벨 액세스 제어로 보호하기 위해 명령이 구현해야 하는 최소 인터페이스입니다.

보호해야 하는 자원에 명령이 액세스하는 경우, AccessVector 개인용 인스턴스 변수를 작성하여 자원을 보유하십시오. 그런 다음 getResources 메소드를 대체하십시오. 이 메소드의 기본 구현은 널 값을 리턴하여 자원을 확인하지 않는 것이기 때문입니다.

새 getResources 메소드에서 자원 또는 명령이 활동할 수 있는 자원 조치 쌍의 배열을 리턴해야 합니다. 조치를 명시적으로 지정하지 않는 경우, 조치의 기본값은 실행 중인 명령의 인터페이스 이름입니다.

명령이 동일한 자원 클래스의 다른 인스턴스에 대한 읽기 및 쓰기 조작을 모두 수행하는 경우에만 조치를 지정하면 됩니다. 예를 들어, OrderCopy 명령의 경우, 소스 주문에서 읽어 대상 주문에 쓸 수 있습니다. 이런 경우, 두 조치 사이에 차이가 있어야 합니다. 이는 소스 주문에 “-Read” 조치를 지정하고 대상 주문에 “-Write” 조치를 지정하여 수행할 수 있습니다. 액세스 제어 프레임워크가 이러한 조치를 검출하게 되면 적용 가능한 정책을 검색하기 전에 명령의 인터페이스 이름으로 조치 앞에 추가합니다. 이런 경우, 정책에서 궁극적으로 사용할 조치는 “com.ibm.commerce.order.commands.OrderCopyCmd-Read” 및 “com.ibm.commerce.order.commands.OrderCopyCmd-Write” 조치입니다.

또한 메소드가 자원의 인스턴스를 작성해야 하는지 또는 자원을 계속 참조하면서 기존 인스턴스 변수를 사용할 수 있는지 여부를 판별하는 것이 바람직합니다. 자원 오브젝트가 이미 존재하는지 여부를 확인함으로써 시스템 성능을 향상시킬 수 있습니다. 그런 다음 필요한 경우, 새 컨트롤러 명령의 performExecute 메소드에서 동일한 인스턴스 변수를 사용할 수 있습니다.

getResources 메소드를 대체해야 하는지 여부를 판별하려면 다음을 고려하십시오.

- 사전 정의된 정보 소스(예: 명령 컨텍스트)에 따라 자원을 가져오는 경우, getResources 메소드를 대체하지 않아도 됩니다. 예를 들어, WebSphere Commerce UserRegistrationUpdate 명령은 명령 컨텍스트에서 사용자의 ID를 가져옵니다. 이런 경우, 사용자는 고유한 등록 정보에 따라 조치를 수행할 수 있는 권한을 이미 갖고 있으므로 getResources 메소드를 대체하지 않아도 됩니다.
- 명령이 임의대로 새 자원을 지정하는 경우(또한 이 자원의 특성이 개인용인 경우), getResources 메소드를 대체해야 합니다. 예를 들어, WebSphere Commerce OrderItemUpdate 명령은 주문 ID를 입력 매개변수로 사용합니다. 이런 경우 주문 자원의 인스턴스가 작성되면 사용자에게 해당 특정 자원에 대한 조치를 수행할 수 있는 권한이 있는지 여부를 알 수 없습니다. 이런 경우 getResources 메소드가 대체됩니다.

다음은 getResources 메소드의 한 예입니다.

```
private AccessVector resources = null;

public AccessVector getResources() throws ECException {

   if (resources == null) {
      OrderAccessBean orderAB = new OrderAccessBean();
      orderAB.setInitKey_orderId(getOrderId().toString());
      resouces = new AccessVector(orderAB);
      }
   return resources;
}
```

예를 들어, OrderItemUpdate 명령을 고려하십시오. 이 명령의 getResources 메소드는 기존 주문 갱신 시 하나 이상의 Order 오브젝트(보호 가능한)를 리턴합니다. 조치를 지정하지 않았으므로, 조치의 기본값은 OrderItemUpdate 명령의 인터페이스입니다.

getResources 메소드에 복수 자원이 리턴될 수 있습니다. 이런 경우 조치가 수행되면 지정된 모든 자원에 대한 사용자 액세스를 부여하는 정책이 있어야 합니다. 사용자가 세 개의 자원 중 두 개에 액세스할 수 있는 경우 조치가 수행되지 않습니다(세 개의 자원에 모두 액세스해야 함).

컨트롤러 명령의 매개변수에 대한 추가 매개변수 확인 또는 분석을 수행해야 하는 경우, validateParameters() 메소드를 사용할 수 있습니다. 이 메소드의 사용은 선택적입니다.

추가 자원 레벨 확인

컨트롤러 명령의 getResources 메소드 호출 시, 보호해야 하는 모든 자원을 언제나 판별할 수 있는 것은 아닙니다.

필요한 경우 태스크 명령 또한 getResources 메소드를 구현하여 명령이 조치를 수행할 수 있는 자원의 목록을 리턴할 수 있습니다.

자원 레벨 확인을 호출하는 다른 방법은 checkIsAllowed(Object resource,String action) 메소드를 사용하여 액세스 제어 정책 관리자를 직접 호출하는 것입니다. 이 메소드는 com.ibm.commerce.command.AbstractECTargetableCommand 클래스에서 확장하는 모든 클래스에 사용 가능합니다. 예를 들어, AbstractECTargetableCommand 클래스에서 다음 클래스가 확장됩니다.

- com.ibm.commerce.command.ControllerCommandImpl
- com.ibm.commerce.command.DataBeanCommandImpl

checkIsAllowed 메소드 또한 com.ibm.commerce.command.AbstractECCommand 클래스에서 확장하는 클래스에 사용 가능합니다. 예를 들어, AbstractECCommand 클래스에서 다음 클래스가 확장됩니다.

- com.ibm.commerce.command.TaskCommandImpl

다음은 checkIsAllowed 메소드의 서명을 표시합니다.

```
void checkIsAllowed(Object resource, String action)
   throws ECException
```

이 메소드는 현재 사용자가 지정된 자원에 지정된 조치를 수행할 수 없는 경우 ECApplicationException을 발생시킵니다. 액세스가 부여되면 메소드가 리턴됩니다.

"create" 명령에 대한 액세스 제어

getResources 메소드는 명령에서 performExecute 메소드 앞에 호출되므로, 아직 작성되지 않은 자원에 대한 액세스 제어에는 다른 접근 방법을 취해야 합니다.

예를 들어, WidgetAddCmd의 경우, getResources 메소드는 곧 작성될 자원을 리턴할 수 없습니다. 이런 경우, getResources 메소드는 새 자원의 컨테이너를 리턴해야 합니다. 예를 들어, 주문을 작성하는 경우, 상점 자원에서 수행됩니다. 새 사용자는 조직 자원에서 작성됩니다.

명령 레벨 액세스 제어의 기본 구현

명령 레벨 액세스 제어의 경우, getOwner() 메소드의 기본 구현은 상점 소유자의 memberId를 리턴합니다(storeId가 지정되어 있는 경우). storeId가 지정되어 있지 않은 경우, 루트 조직의 memberId가 리턴됩니다(memberId = -2001).

getResources() 메소드의 기본 구현은 null을 리턴합니다.

validateParameters()의 기본 구현은 아무 것도 수행하지 않습니다.

뷰에서 액세스 제어 정책 구현

뷰에 대한 자원 레벨 액세스 제어는 데이터 bean 관리자에 의해 수행됩니다. 데이터 bean 관리자는 다음과 같은 경우 호출됩니다.

1. JSP 템플리트에 <useBean> 태그가 포함되어 있고 데이터 bean이 속성 목록에 없는 경우
2. JSP 템플리트에 다음 activate 메소드가 포함되어 있는 경우

```
DataBeanManager.activate(xyzDatabean, request);
```

주: 보호할(직접 또는 간접) 데이터 bean은 위임자 인터페이스를 구현해야 합니다. 직접 보호할 데이터 bean은 자신에게 위임되므로 Protectable 인터페이스 또한 구현해야 합니다. 간접 보호하는 데이터 bean은 Protectable 인터페이스를 구현하는 데이터 bean으로 위임되어야 합니다.

권장사항은 아니지만 다음과 같은 경우 액세스 제어 확인이 생략됩니다.

1. JSP 템플리트가 데이터 bean을 사용하지 않고 액세스 bean을 직접 호출하는 경우
2. JSP 템플리트가 데이터 bean의 populate() 메소드를 직접 호출하는 경우

컨트롤러 명령의 결과를 뷰로 전달해야 하는 경우(ForwardViewCommand 사용), 뷰에 대한 명령 레벨 액세스 제어가 수행되지 않습니다. 또한 컨트롤러 명령이 대

량 자료가 반입된 데이터 bean(뷰에서 사용됨)을 응답 특성의 속성 목록에 둔 후 뷰로 전달하는 경우, JSP 템플리트는 데이터 bean 관리자를 사용하지 않고 데이터에 액세스할 수 있습니다. 이를 위해서는 <useBean> 태그를 JSP 템플리트에서 사용해야 합니다. 이는 JSP 템플리트의 효율성을 높이는 방법이 될 수 있습니다. 컨트롤러 명령을 통해 사용자에게 이미 액세스가 허용된 자원에 대한 중복된 자원 레벨 액세스 제어 확인을 생략할 수 있기 때문입니다.

기존 WebSphere Commerce 자원에 대한 액세스 제어 수정

이 절에서는 기존 WebSphere Commerce 자원에 대한 액세스 제어 수정 방법에 대해 설명합니다. 특히 다음 시나리오를 검토합니다.

- 이미 액세스 제어로 보호하는 기존 WebSphere Commerce 엔티티 bean에 새 관계 추가
- 아직 액세스 제어로 보호하지 않는 기존 WebSphere Commerce 엔티티 bean에 액세스 제어 보호 추가
- 기존 컨트롤러 명령 확장 시 액세스 제어에 대한 의미 이해

기존 WebSphere Commerce 엔티티 bean에 새 관계 추가

Protectable 인터페이스를 구현하는 WebSphere Commerce 엔티티 bean은 이미 액세스 제어로 보호하고 있습니다. 액세스 제어의 요구조건은 WebSphere Commerce의 초기 기능 및 특징에서 bean이 사용되는 방법에 의해 결정됩니다. 해당 bean의 액세스 제어에 추가 관계를 추가해야 하는 경우가 발생할 수 있습니다. 예를 들어, 일부 사용자 정의 코드에서 기존 bean을 사용하거나 기존 WebSphere Commerce 공개 엔티티 bean을 수정하는 경우, bean에 추가 관계를 추가해야 합니다.

다음 목록은 이미 액세스 제어로 보호하고 있는 기존 WebSphere Commerce 엔티티 bean에 새 관계를 추가하는 상위-레벨 단계를 제공합니다.

1. 엔티티 bean의 기존 fulfills 메소드를 점검하십시오. 이 메소드는 bean의 액세스 헬퍼 클래스에 있습니다. 이 클래스는 수정하지 말고, 이 로직에 하나 이상의 새 관계를 추가해야 하는지 여부를 결정해야 하는 경우 또는 이 메소드

를 대체해야 하는 경우에만 사용하십시오. 예를 들어, 다음 fulfills 메소드는 com.ibm.commerce.fulfillment.objsrc.FulfillmentCenterBeanAccessHelper 클래스에 나타납니다.

```
public boolean fulfills(Object obj, Long member, String relationship)
  throws Exception {

 FulfillmentCenterBean bean = (FulfillmentCenterBean) obj;

 if ("ShippingArrangementOrganizationalEntity".
   equalsIgnoreCase(relationship))
 {
      FulfillmentJDBCHelperAccessBean ffmJDBCAB =
         new FulfillmentJDBCHelperAccessBean();
      int count = ffmJDBCAB.
         checkFulfillmentCenterByMemberIdAndFulfillmentCenterId(
         member,bean.getFulfillmentCenterId());
      if(count>0)
         return true;
 }
      return false;
}
```

2. 다음 단계는 bean 클래스에 새 fulfills 메소드를 작성하는 것입니다. 예를 들어, com.ibm.commerce.fulfillment.objects.FulfillmentBean.java 클래스에 새 fulfills 메소드를 작성할 수 있습니다. 메소드 선언은 다음과 같이 표시되어야 합니다.

```
public boolean fulfills(Long member, String relationship)
  throws Exception {
      // Place holder for relationship information
 }
```

3. 기존 관계에 추가 관계를 추가하는 경우, 메소드의 처음 행은 상위 클래스에서 fulfills 메소드를 호출하기 위한 것이어야 합니다. 그런 다음 해당 메소드가 false 를 리턴하는 경우, 다음과 같이 새 관계를 확인하십시오.

```
public boolean fulfills(Long member, String relationship)
  throws Exception {
      if (super.fulfills().equals(false))
      {
      // Check if new relationship is met
         return true;
      }

      return false;
 }
```

4. 원래 구현에서 관계를 완전히 바꾸는 경우, 다음과 같이 super.fulfills 메소드를 호출해서는 안 됩니다.

```
public boolean fulfills(Long member, String relationship)
  throws Exception {
      // Check if new relationship is met
      // If it is, then return true;

     // If the relationship is not met, return false;
 }
```

5. 변경사항을 저장하십시오. 해당 액세스 bean 및 bean의 전개 코드 및 RMIC 코드를 다시 작성하십시오.

아직 보호하지 않는 기존 WebSphere Commerce 엔티티 bean에 액세스 제어 추가

기존 WebSphere Commerce 엔티티 bean을 사용하고 응용프로그램에서 액세스 제어로 bean을 보호해야 하는 경우, 이 보호를 추가할 수 있습니다.

다음 목록은 WebSphere Commerce 액세스 제어 시스템으로 기존 WebSphere Commerce 엔티티 bean을 보호하는 상위-레벨 단계를 제공합니다.

1. *BeanName*.java 클래스를 여십시오. 원격 인터페이스입니다. `com.ibm.commerce.security.Protectable` 인터페이스를 확장하도록 수정하십시오.
2. 액세스 제어 정책을 적용하기 위해 해당 Java 클래스 이름이 아닌 속성으로 자원을 그룹화하는 경우, bean의 원격 인터페이스는 `com.ibm.commerce.grouping.Groupable` 인터페이스를 확장해야 합니다.
3. 변경사항을 원격 인터페이스에 저장하십시오.
4. *BeanName*Bean.java 클래스를 여십시오. 여기서, *BeanName*은 액세스 제어 보호를 추가하는 엔티티 bean의 이름입니니다.
5. 엔터프라이즈 bean 클래스는 com.ibm.commerce.base.objects.ECEntityBean에서 다음 메소드의 기본 구현을 상속합니다.
   - `getOwner`
   - `fulfills`
   - `getGroupingAttributeValue`

필요한 메소드를 대체하십시오. 최소한 getOwner 메소드를 대체해야 합니다. 이 메소드에 대한 추가 정보는 125 페이지의 『엔터프라이즈 bean에서 액세스 제어 구현』을 참조하십시오.

6. 변경사항을 저장하십시오. 해당 액세스 bean 및 bean의 전개 코드 및 RMIC 코드를 다시 작성하십시오.

컨트롤러 명령 확장 시 액세스 제어 구현에 대한 이해

WebSphere Commerce 프로그래밍 모델에 따라 기존 컨트롤러 명령의 사용자 고유 구현을 작성할 수 있습니다. 이런 경우, 새 구현 클래스를 작성한 후 명령 레지스트리를 갱신하여 기존 인터페이스에 새 구현 클래스를 연관시킬 수 있습니다.

이러한 확장 수행에 대한 세 가지 잠재적인 액세스 제어 관련 의미는 다음과 같습니다.

1. getResources 메소드에 대한 영향
2. 명령 레벨 액세스 제어 정책에 대한 영향
3. 자원 레벨 액세스 제어 정책에 대한 영향

이들 각 사항은 후속 절에서 자세히 설명합니다.

getResources 메소드에 대한 영향

기존 컨트롤러 명령을 확장하는 경우, 즉 명령의 새 사용자 정의 로직 뿐 아니라 기존 로직을 수행하는 경우, 새 명령이 기존 명령의 서브클래스가 됩니다. 새 구현의 performExecute 메소드는 상위 클래스의 performExecute 메소드를 호출합니다. 새 명령이 보호가 필요한 새 자원에 액세스하지 않는 경우, getResources 메소드를 대체하지 않아도 됩니다. 그러나 새 보호 가능 자원에 액세스하는 경우, 새 명령은 고유한 getResources 로직을 구현하기 전에 고유한 getResources 메소드를 구현해야 하며 이 메소드에서 상위 클래스로부터 getResources 메소드를 호출해야 합니다.

상위 클래스로부터의 getResources 메소드 결과는 AccessVector 유형의 개인용 인스턴스 변수에 저장해야 합니다. 그런 다음 로컬 getResources 메소드의 결과는 이 벡터 끝에 추가해야 합니다. 예를 들어, 새 구현 클래스는 다음 의사 코드와 유사한 코드를 포함하게 됩니다.

```
private AccessVector resources = null;

public AccessVector getResources() throws ECException {
   // First, get the resources from the original implementation
   resources = super.getResources();

   // Now, append the new resources

   ////////////////////////////////////
   //  Logic for getting new resources   //
   //  and appending to the vector.      //
  ////////////////////////////////////

   return resources;
}
```

기존 컨트롤러 명령의 로직을 확장하지 않고 로직을 완전히 바꾸는 경우, 새 구현에서 super.performExecute 메소드를 호출하지 않습니다. 그러면 사용자 고유 구현에서 super.getResources 메소드를 호출하지 않아도 됩니다. 대신, 사용자 고유 getResources 메소드만 적절하게 구현하십시오.

명령 레벨 액세스 제어 정책에 대한 영향

컨트롤러 명령에 대한 명령 레벨 정책은 명령 자원에서 수행되는 "실행" 조치로 구성됩니다. 명령 자원은 해당 인터페이스 이름으로 지정됩니다. 기존 명령을 확장하는 경우 명령은 원래 인터페이스를 계속 구현하고, 마찬가지로 기존 명령 레벨 정책은 이전 명령 레벨 액세스 제어를 유지보수할 수 있습니다. 따라서 변경하지 않아도 됩니다.

자원 레벨 액세스 제어 정책에 대한 영향

기존 명령의 새 구현을 작성하고 이 구현을 해당 기존 명령의 인터페이스에 연관시킨 경우, 자원 레벨 액세스 제어 정책은 변경하지 않아도 됩니다.

기존 구현을 확장하는 새 확장 클래스를 작성하고 이 클래스에서 super.performExecute 메소드를 호출하며 새 인터페이스 또한 구현하는 경우, 자원 레벨 액세스 제어 정책을 변경해야 합니다.

이 후자의 경우, 새 명령이 getResources 메소드를 구현하거나 기본 명령에서 getResources 명령의 중요 구현을 상속하는 경우, 자원 레벨 정책을 변경해야 합니다. 일반적으로 자원 레벨 정책은 비즈니스 오브젝트 자원에서 수행되는 명령 조

치로 구성됩니다. 조치는 단지 명령 인터페이스의 문자열 표시이므로 새 명령은 일반적으로 기본 명령의 조치 그룹에 추가해야 합니다. 그러나 단지 널(Null)값을 리턴하여 새 명령이 getResources 메소드의 기본 구현을 대체하는 경우, 이 새 명령에 대한 액세스 제어 확인이 수행되지 않습니다. 이를 주의 깊게 수행하지 않는 경우 악의적인 사용자에게 잠재적으로 새 명령을 노출시킬 수도 있음에 유의하십시오.

자원 레벨 액세스 제어 정책을 수정하는 경우 다음과 같은 상위-레벨 단계를 수행합니다.

1. 다음 디렉토리에서 defaultAccessControlPolicies.xml 파일을 찾으십시오.
   - Studio *WCStudio_installdir*\Commerce\xml\policies\xml
2. 이 파일의 사본을 작성하십시오. 예를 들어, *myDefaultAccessControlPolicies.xml* 파일의 이름을 지정하십시오.
3. 이 새 파일에 다음 인터페이스를 새 조치로 정의해야 합니다. 예를 들어, 다음을 추가합니다.

   ```
   <Action Name="yourNewInterface"
      CommandName="yourNewInterface">
      </Action>
   ```

   여기서, *yourNewInterface*는 새 인터페이스의 이름입니다.
4. 다음으로 파일을 검색하여 원래 조치가 속해 있는 모든 조치 그룹을 찾은 후 새 조치에 추가해야 합니다.
5. 이전에 base 명령으로 지정하지 않은 자원에서 새 명령을 수행하는 경우, 해당 액세스 제어 정책의 자원 그룹 또한 새 자원을 포함하도록 변경해야 합니다.
6. XML 파일을 변경하고 나면 수정하지 않은 섹션을 제거할 수 있습니다. `<Policies>` 태그 앞에 텍스트를 두었는지 확인하십시오.
7. *WebSphere Commerce 보안 안내서*의 지시사항에 따라 새 정책 정보를 데이터베이스에 로드하십시오.

개발 용도의 견본 액세스 제어 정책

이 절에서는 개발 환경에서 사용할 수 있는 몇 가지 매우 간단한 액세스 제어 정책을 제공합니다. 이를 통해 새 자원을 빨리 테스트할 수 있습니다. 이 정책은 모든 WebSphere Commerce 프로덕션 환경에서 사용할 수 있도록 설계되었습니다. 적절한 자원 보호를 제공하지 않기 때문입니다.

이 정책을 로드하는 방법에 대한 자세한 정보는 *WebSphere Commerce 보안 안내서*를 참조하십시오.

새 뷰에 대한 견본 액세스 제어 정책

새 뷰를 작성하는 경우, 다음 액세스 제어 정책을 사용하여 개발 환경에서 새 뷰를 테스트할 수 있습니다. 사용자 환경에 맞게 정책을 수정하고 acpload 명령을 사용하여 로드하십시오.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">

<Policies>
   <Action Name="YourNewView"
        CommandName="YourNewView">
   </Action>
   <ActionGroup Name="AllSiteUsersViews"
         OwnerID="RootOrganization">
      <ActionGroupAction Name="YourNewView"/>
   </ActionGroup>
</Policies>
```

여기서, *YourNewView*는 새로 작성한 뷰 이름입니다. 앞의 액세스 제어 정책은 기존 AllSiteUsersViews 조치 그룹에 새 뷰를 추가합니다. 모든 사용자가 이 정책을 사용하여 새 뷰에 액세스할 수 있습니다.

새 컨트롤러 명령에 대한 견본 명령 레벨 액세스 제어 정책

컨트롤러 명령이 액세스 제어 프레임워크의 요구사항을 충족시키려면 액세스 제어 정책이 필요합니다. 새 컨트롤러 명령을 작성하는 경우, 명령 인터페이스의 이름은 자원으로 지정됩니다. 다음 XML 단편은 새 명령에 맞게 수정할 수 있으며 acpload 명령을 사용하여 로드할 수 있습니다.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">

<Policies>

   <Action Name="ExecuteCommand"
         CommandName="Execute">
   </Action>

   <ResourceCategory Name="com.yourcompany.yourpackage.commands.
         YourControllerCmdResourceCategory"
         ResourceBeanClass="com.yourcompany.yourpackage.commands.
         YourControllerCmd">
        <ResourceAction Name="ExecuteCommand"/>
   </ResourceCategory>

   <ResourceGroup Name="AllSiteUserCmdResourceGroup"
         OwnerID="RootOrganization">
      <ResourceGroupResource Name="com.yourcompany.yourpackage.commands.
         YourControllerCmdResourceCategory" />
   </ResourceGroup>

</Policies>
```

여기서,

- *com.yourcompany.yourpackage.commands*는 패키징 구조를 나타냅니다.
- *YourControllerCmd*는 새 컨트롤러 명령의 이름을 나타냅니다.

한 예로 다음 XML 파일을 사용하여 이 책의 학습에서 작성하는 새 컨트롤러 명령에 대한 액세스 제어 정책을 로드합니다.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">

<Policies>
   <Action Name="ExecuteCommand"
   CommandName="Execute">
   </Action>

   <ResourceCategory Name="com.ibm.commerce.sample.commands.
          MyNewControllerCmdResourceCategory"
          ResourceBeanClass="com.ibm.commerce.sample.commands.
          MyNewControllerCmd">
         <ResourceAction Name="ExecuteCommand"/>
   </ResourceCategory>

   <ResourceGroup Name="AllSiteUserCmdResourceGroup"
         OwnerID="RootOrganization">
      <ResourceGroupResource Name="com.ibm.commerce.sample.commands.
```

```
        MyNewControllerCmdResourceCategory" />
    </ResourceGroup>

</Policies>
```

새 명령 및 엔터프라이즈 bean에 대한 견본 자원 레벨 액세스 제어 정책

다음 XML 파일은 이 안내서에 포함되어 있는 학습에서 가져온 것입니다. 이 파일은 새 엔티티 bean 작성 시 액세스 제어 요구사항에 대한 템플리트 기능을 수행할 수 있습니다. 다음 파일의 경우, 새 엔티티 bean은 XBONUS 데이터베이스 테이블에 해당하는 보너스 bean이며 MyNewControllerCmd 컨트롤러 명령에서 사용합니다. 이 액세스 제어 정책의 경우, 보너스 bean 오브젝트의 작성자만이 이 오브젝트에 대한 MyNewControllerCmd 조치를 수행할 수 있습니다.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">

<Policies>
   <Action Name="MyNewControllerCmd"
      CommandName="com.ibm.commerce.sample.commands.MyNewControllerCmd">
   </Action>

   <ResourceCategory Name="com.ibm.commerce.extension.objects.
       BonusResourceCategory"
       ResourceBeanClass="com.ibm.commerce.extension.objects.Bonus" >

         <ResourceAction Name="MyNewControllerCmd" />
   </ResourceCategory>

   <ActionGroup Name="MyNewControllerCmdActionGroup"
         OwnerID="RootOrganization">
   <ActionGroupAction Name="MyNewControllerCmd"/>
   </ActionGroup>

   <ResourceGroup Name="BonusResourceGroup"  OwnerID="RootOrganization" >
      <ResourceGroupResource Name="com.ibm.commerce.extension.objects.
            BonusResourceCategory" />
   </ResourceGroup>
   <Policy Name="AllUsersUpdateBonusResourceGroup"
      OwnerID="FashionFlowMemberId"
      UserGroup="AllUsers"
      UserGroupOwner="RootOrganization"
      ActionGroupName="MyNewControllerCmdActionGroup"
      ResourceGroupName="BonusResourceGroup"
      RelationName="creator"
      PolicyType="groupableStandard">
   </Policy>

   <PolicyGroup Name="ManagementAndAdministrationPolicyGroup"
         OwnerID="RootOrganization">
```

```
   <!-- Define policies in this policy group -->
   <PolicyGroupPolicy  Name="AllUsersUpdateBonusResourceGroup"
            PolicyOwnerID="FashionFlowMemberId" />

   </PolicyGroup>

</Policies>
```

여기서, *FashionFlowMemberId*는 새 자원을 사용하는 상점의 구성원 ID입니다.

앞의 액세스 제어 정책에서 컨트롤러 명령의 인터페이스 이름은 해당 패키지 이름을 포함하는 완전한 이름이 아닌 조치로 지정됩니다. 응용프로그램에 같은 이름을 갖는 여러 인터페이스가 있는 경우, 액세스 제어 정책에서 조치로 지정할 때 패키지 이름을 포함하는 완전한 이름으로 규정해야 합니다. 예를 들어, 인터페이스 이름이 모호한 경우 앞의 액세스 제어 정책은 다음과 같이 변경을 요구합니다(주: 변경된 행만이 표시되며 수정사항은 굵은체로 표시됨).

```
<Action Name="com.ibm.commerce.sample.commands.
MyNewControllerCmd"
      CommandName="com.ibm.commerce.sample.commands.
      MyNewControllerCmd">
.
.
.
<ResourceAction Name="com.ibm.commerce.sample.commands.
MyNewControllerCmd" />
.
.
.
<ActionGroupAction Name="com.ibm.commerce.sample.commands.
MyNewControllerCmd"/>
```

제 5 장 오류 처리 및 메시지

명령 오류 처리

WebSphere Commerce는 사용자 정의 코드로 간편하게 사용할 수 있는 정의된 명령 오류 처리 프레임워크를 사용합니다. 프레임워크는 다국어 지원 상점을 지원하는 설계로 오류를 처리합니다. 이 절에서는 명령이 발행할 수 있는 예외, 예외 처리 방법, 메시지 텍스트의 저장 방법과 사용 방법, 예외 로그 방법 및 사용자 고유 명령으로 제공된 프레임워크를 사용하는 방법에 대해 설명합니다.

예외 유형

명령은 다음과 같은 예외를 발행할 수 있습니다.

ECApplicationException

사용자와 관련이 있는 경우 발생합니다. 예를 들어, 사용자가 올바르지 않은 매개변수를 입력하는 경우, ECApplicationException이 발생합니다. 이 예외가 발생하면 재시도 가능 명령으로 지정되어도 웹 컨트롤러가 명령을 재시도하지 않습니다.

ECSystemException

런타임 예외 또는 WebSphere Commerce 구성 오류가 검출되는 경우 발생합니다. 이러한 유형의 예외 예에는 널 포인터 예외 및 트랜잭션 롤백 예외가 포함됩니다. 이 예외가 발생하면 명령을 재시도할 수 있고 데이터베이스 교착상태 또는 데이터베이스 롤백에 의해 예외가 발생한 경우 웹 컨트롤러가 명령을 재시도합니다.

위에 표시된 두 가지 예외는 모두 com.ibm.commerce.exception 패키지의 ECException 클래스에서 확장되는 클래스입니다.

이 예외 중 하나를 발행하려면 다음 정보를 지정해야 합니다.

- 오류 뷰 이름
  웹 컨트롤러가 VIEWREG 테이블에서 이 이름을 검색합니다.

- ECMessage 오브젝트
  이 값은 특성 파일에 있는 메시지 텍스트입니다.
- 오류 매개변수
  이 이름 값 쌍은 오류 메시지로 정보를 바꾸는 데 사용됩니다. 예를 들어, 예외를 발행한 메소드의 이름을 보관할 매개변수가 메시지에 있을 수 있습니다. 이 매개변수는 예외 발생 시 설정되며 이후 오류 메시지가 로그되면 로그 파일이 실제 메소드 이름을 갖게 됩니다.
- 오류 데이터
  오류 데이터 bean을 통해 JSP 템플리트에서 사용할 수 있는 선택 속성입니다.

예외 처리는 로그 작성 시스템과 밀접한 관련이 있습니다. 시스템 예외가 발생하면 자동으로 로그됩니다.

오류 메시지 특성 파일

오류 메시지의 유지보수를 용이하게 하고 다국어 지원 상점을 지원하기 위해, 오류 메시지의 텍스트는 특성 파일에 저장됩니다. WebSphere Commerce 메시지 텍스트는 ecServerMessages_*XX_XX*.properties 파일에 저장됩니다. 여기서, *_XX_XX*는 로케일 지시자입니다(예: _en_US).

명령 컨텍스트는 클라이언트가 사용하는 언어를 표시하기 위한 식별자를 리턴합니다. 메시지가 필요한 경우, 웹 컨트롤러가 언어 식별자에 따라 사용할 특성 파일을 결정합니다.

ecServerMessages*XX_XX*.properties 파일에는 두 가지 유형의 메시지, 즉 사용자 메시지 및 시스템 메시지가 정의되어 있습니다. 사용자 메시지는 해당 브라우저로 고객에게 표시됩니다. 시스템 메시지 및 사용자 메시지 모두 메시지 로그에 자동으로 캡처됩니다.

오류가 발생하는 경우, 필수 매개변수 중 하나가 메시지 오브젝트입니다. ECSystemExceptions의 경우, 메시지 오브젝트에는 두 개의 키가 있어야 합니다. 하나는 시스템 메시지용이고 다른 하나는 사용자 메시지용입니다.
ECApplicationExceptions의 경우, 메시지 오브젝트에는 사용자 메시지용 키가 있습니다. 시스템 메시지는 사용하지 않습니다.

모든 시스템 메시지는 사전 정의됩니다. 사용자 고유 시스템 메시지를 작성할 수 없습니다. 따라서 사용자 정의 코드가 ECSystemException를 발행하는 경우, 사전 정의된 시스템 메시지 중 하나에 대한 메시지 키를 지정해야 합니다. 사용자 정의 사용자 메시지를 작성할 수 있습니다. 새 사용자 메시지는 개별 특성 파일에 저장해야 합니다.

예외 처리 플로우

다음 도표는 예외 발생 시 정보 플로우를 표시합니다. 다음은 각 단계에 대한 설명입니다.

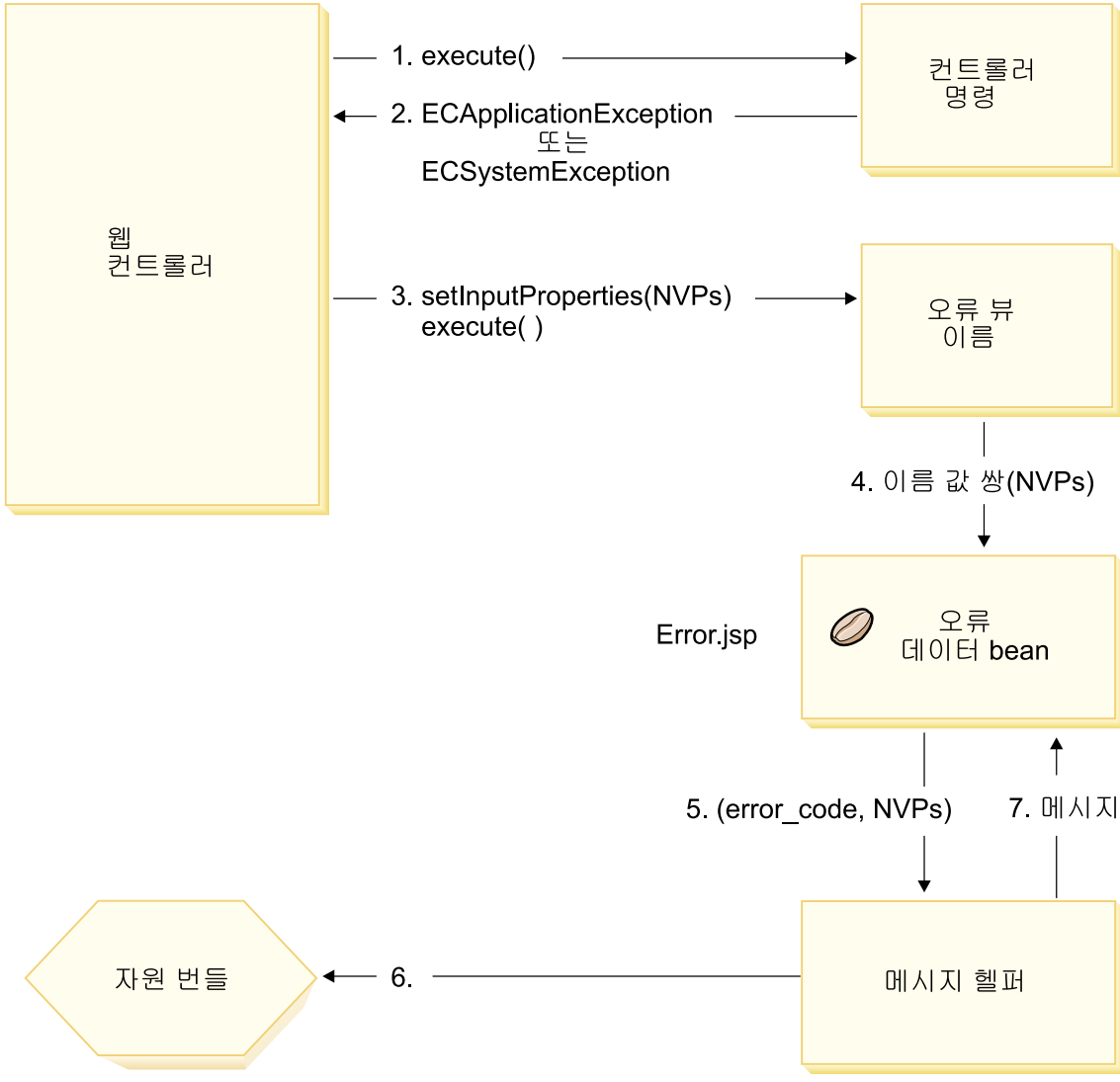


그림 24.

1. 웹 컨트롤러가 컨트롤러 명령을 호출합니다.
2. 명령이 웹 컨트롤러가 발견한 예외를 발행했습니다. 이 예외는 ECApplicationException 또는 ECSystemException일 수 있습니다. 예외 오브젝트에는 다음 정보가 들어 있습니다.
   - 오류 뷰 이름

- ECMessage 오브젝트
- 오류 매개변수
- 오류 데이터(선택)

3. 웹 컨트롤러가 VIEWREG 테이블의 오류 뷰 이름을 판별하고 지정된 오류 뷰 명령을 호출합니다. 명령을 호출하면 웹 컨트롤러가 ECException 오브젝트에서 특성 세트를 작성하고 뷰 명령의 setInputProperties 메소드를 사용하여 뷰 명령으로 설정합니다.
4. 뷰 명령이 오류 JSP 템플리트(이 경우 Error.jsp)를 호출하고 이름 값 쌍이 JSP 템플리트로 전달됩니다.
5. ErrorDataBean이 메시지 헬퍼 오브젝트로 오류 매개변수를 전달합니다.
6. 메시지 헬퍼 오브젝트가 해당 특성 파일에서 필요한 메시지를 가져옵니다(메시지 오브젝트 및 오류 매개변수를 사용하여).
7. 오류 데이터 bean이 JSP 템플리트로 메시지를 리턴합니다.

사용자 정의 코드로 예외 처리

새 명령 작성 시 적절한 예외 처리를 포함하는 것이 중요합니다. 예외 발견 시 필요한 정보를 지정하여, WebSphere Commerce에서 제공하는 오류 처리 및 메시지 전달 프레임워크를 이용할 수 있습니다.

사용자 고유 예외 처리 로직을 작성하는 단계는 다음과 같습니다.

1. 명령에서 특별 처리가 필요한 예외 발견
2. 발견한 예외의 유형에 따라 ECApplicationException 또는 ECSystemException 구성
3. ECApplicationException이 새 메시지를 사용하는 경우, 새 특성 파일에 메시지 정의

예외 발견 및 구성

다음은 처음 두 단계를 설명하기 위해 명령에서 시스템 예외를 발견하는 예를 표시합니다.

```
try {
// your business logic
}
```

```
catch(FinderException e) {
     throw new ECSystemException (ECMessage._ERR_FINDER_EXCEPTION,
          className, methodName, new Object [] {e.toString()}, e);
}
```

앞의 _ERR_FINDER_EXCEPTION ECMessage 오브젝트는 다음과 같이 정의됩니다.

```
public static final ECMessage _ERR_FINDER_EXCEPTION =
new ECMessage (ECMessageSeverity.ERROR, ECMessageType.SYSTEM,
     ECMessageKey._ERR_FINDER_EXCEPTION);
```

ERR_FINDER_EXCEPTION 메시지 텍스트는 다음과 같이 ecServerMessages_*xx_XX*.properties 파일에 정의됩니다. 여기서, *_xx_XX*는 로케일 지시자(예: *_en_US*)입니다.

```
_ERR_FINDER_EXCEPTION  =
     The following Finder Exception occurred during processing: "{0}".
```

시스템 예외 발견 시, 사용할 수 있는 메시지 세트가 사전 정의되어 있습니다. 이 메시지는 다음 테이블에서 설명합니다.

메시지 오브젝트	설명
_ERR_FINDER_EXCEPTION	EJB 파인더 메소드 호출에서 오류가 리턴될 때 발생합니다.
_ERR_REMOTE_EXCEPTION	EJB 원격 메소드 호출에서 오류가 리턴될 때 발생합니다.
_ERR_CREATE_EXCEPTION	EJB 인스턴스 작성 중 오류가 발생할 때 발생합니다.
_ERR_NAMING_EXCEPTION	이름 서버에서 오류가 리턴될 때 발생합니다.
_ERR_GENERIC	예상치 못한 시스템 오류 발생 시 발생합니다. 예를 들어, 널 포인터 예외입니다.

응용프로그램 예외 발견 시, 해당 ecServerMessages_xx_xx.properties 파일에 지정되어 있는 기존 메시지를 사용하거나 새 특성 파일에 저장되는 새 메시지를 작성할 수 있습니다. 앞에서 설명한 대로 ecServerMessages_xx_XX.properties 파일을 수정해서는 *안 됩니다*.

다음 코드 단편은 명령에서 응용프로그램 예외를 발견하는 예를 표시합니다.

```
try {
// your business logic

}
// catch some new type of application exception
catch(//your new exception)
 {
     throw new ECApplicationException (MyMessages._ERR_CUSTOMER_INVALID,
          className, methodName, errorTaskName, someNVPs);
}
```

앞의 _ERR_CUSTOMER_INVALID ECMessage 오브젝트는 다음과 같이 정의됩니다.

```
public static final ECMessage _ERR_CUSTOMER_INVALID =
     new ECMessage (ECMessageSeverity.ERROR, ECMessageType.USER,
     MyMessagesKey._ERR_CUSTOMER_INVALID, "ecCustomerMessages");
```

새 사용자 메시지 구성 시, 다음과 같이 USER 유형으로 메시지를 지정해야 합니다.

```
ECMessageType.USER
```

_ERR_CUSTOMER_INVALID 메시지의 텍스트는 ecCustomerMessages.properties 파일에 들어 있습니다. 이 파일은 클래스 경로에 있는 디렉토리에 상주해야 합니다. 텍스트는 다음과 같이 정의됩니다.

```
_ERR_CUSTOMER_INVALID = Invalid ID "{0}"
```

메시지 작성

명령이 새 메시지를 사용하는 ECApplicationException을 발행하는 경우, 이 새 메시지를 작성해야 합니다. 새 메시지 작성에는 다음 단계가 포함됩니다.

1. 메시지 키가 들어 있는 새 클래스 작성
2. ECMessage 오브젝트가 들어 있는 새 클래스 작성
3. 자원 번들 작성

각 단계에 대한 정보는 다음 절에서 제공합니다.

메시지 키 클래스 작성

새 사용자 메시지를 작성하는 첫 단계는 새 메시지 키가 있는 클래스를 작성하는 것입니다. 메시지 키는 자원 번들에서 해당 메시지 텍스트를 찾기 위해 로그 작성

서비스가 사용하는 고유 지시자입니다. 이 새 클래스는 사용자 고유 패키지에서 작성하고 WebSphereCommerceServerExtensionsLogic 프로젝트에 저장해야 합니다.

_ERR_CUSTOMER 및 _ERR_CUSTOMER_INVALID_ID 메시지 키를 포함하는 새 클래스, MyMessageKeys를 작성하는 MyNewMessages가 있고 이 클래스를 com.mycompany.messages 패키지에 배치하는 예를 들어 보겠습니다. 이 경우, 클래스 정의는 다음과 같이 나타납니다.

```
public class MyMessageKeys
{
    public static String _ERR_CUSTOMER="_ERR_CUSTOMER";
    public static String _ERR_CUSTOMER_INVALID_ID="_ERR_CUSTOMER_INVALID_ID";
}
```

메시지 키의 문자열 랩퍼를 제공하면 컴파일러가 그 유효성을 확인할 수 있습니다.

ECMessage 오브젝트 클래스 작성

메시지 키 클래스를 작성한 동일한 패키지에서 ECMessage 오브젝트가 있는 다른 클래스를 작성하십시오. ECMessage 클래스는 메시지 오브젝트의 구조를 정의합니다. 로케일 구분 텍스트 메시지를 검색 및 지속시키는 데 사용됩니다.

메시지 오브젝트의 속성은 심각도, 유형, 키, 자원 번들 및 연관 자원 번들입니다. 이 클래스에는 여러 생성자 메소드가 있습니다. 전체 정보는 WebSphere Commerce 온라인 도움말의 “참조” 절을 참조하십시오.

MyNewMessages 예에 따라 다음과 같이 com.mycompany.messages 패키지에 새 클래스, MyMessages를 작성하십시오.

```
import com.ibm.commerce.ras.*;

public class MyMessages
    {
        static String myResourceBundle = "ecCustomerMessages";

        public static ECMessage _ERR_CUSTOMER = new ECMessage
             (ECMessageSeverity.ERROR,ECMessageType.USER,
                  MyMessageKeys._ERR_CUSTOMER, myResourceBundle);
        public static ECMessage _ERR_CUSTOMER_INVALID_ID = new ECMessage
             (ECMessageSeverity.ERROR, ECMessageType.USER,
```

```
                MyMessageKeys._ERR_CUSTOMER_INVALID_ID,
                myResourceBundle);

    }
```

앞의 코드 단편의 경우, ECMessage 오브젝트를 작성하려면 import 문이 필요합니다. MyMessage._ERR_CUSTOMER 오브젝트는 심각도 ERROR의 사용자 메시지입니다. MyMessageKeys._ERR_CUSTOMER는 WebSphere Commerce 로그 작성 서비스가 *ecCustomerMessages* 특성 파일에 있는 메시지를 찾기 위해 사용합니다.

사용자 메시지 자원 번들 작성

해당 메시지 텍스트가 있는 메시지 키를 저장할 새 자원 번들을 작성해야 합니다. 이 자원 번들은 Java 오브젝트 또는 특성 파일로서 구현될 수 있습니다. 변환 및 유지보수가 간편하므로 특성 파일을 사용할 것을 권장합니다. 특성 파일은 WebSphere Commerce 메시지에 사용됩니다.

MyNewMessages 예를 계속 진행하려면 `ecCustomerMessages.properties` 이름으로 텍스트 파일을 작성하십시오. 단일 상점 Servlet에서 메시지를 사용하는 경우, 이 파일을 다음 디렉토리에 두십시오.

Studio *workspace_dir*\Stores\Web Content\WEB-INF\classes*storeDir*

여기서, *storeDir*은 상점의 이름입니다.

WebSphere Commerce 액셀러레이터가 메시지를 사용하는 경우, 이 파일을 다음 디렉토리에 두십시오.

Studio *workspace_dir*\CommerceAccelerator\Web Content\WEB-INF\classes

관리 콘솔이 메시지를 사용하는 경우, 이 파일을 다음 디렉토리에 두십시오.

Studio *workspace_dir*\SiteAdministration\Web Content\WEB-INF\classes

Business Organization 관리 콘솔이 메시지를 사용하는 경우, 이 파일을 다음 디렉토리에 두십시오.

Studio *workspace_dir*\OrganizationAdministration\Web Content\WEB-INF\classes

엔터프라이즈 응용프로그램의 모든 Servlet이 전체적으로 이 메시지를 사용하는 경우, 이 파일을 다음 디렉토리에 두십시오.

Studio *workspace_dir*\WebSphereCommerceServer\properties

위의 디렉토리는 개발 환경의 컨텍스트에 지정됩니다. 해당 환경에서 테스트를 완료했으면 대상 WebSphere Commerce Server로의 전개에 대한 정보는 233 페이지의 제 9 장『전개 정보』를 참조하십시오.

특성 파일에는 메시지 키 및 해당 메시지 텍스트 쌍이 있으므로, `ecCustomerMessages.properties` 파일에는 다음 행이 있습니다.

```
_ERR_CUSTOMER_MESSAGE = The customer message "{0}".
_ERR_CUSTOMER_INVALID_ID = Invalid ID "{0}".
```

실행 플로우 추적

Commerce 응용프로그램의 실행 플로우를 추적해야 하는 경우, WebSphere Application Server JRas 기능을 사용해야 합니다. 이는 응용프로그램이 사용할 수 있는 메시지 로그 작성 및 진단 추적 API입니다.

사용자 정의 코드에서 이 기능을 사용하는 방법에 대한 정보는 WebSphere Application Server 정보 센터를 참조하십시오.

개발 환경에서 구성요소 추적 구성에 대한 정보는 439 페이지의 부록 A『WebSphere Commerce Studio에 WebSphere Commerce 구성요소 추적 구성』을 참조하십시오.

JSP 템플리트 오류 처리

JSP 템플리트에 대한 오류 처리는 다음과 같은 다양한 방법으로 수행할 수 있습니다.

- 페이지에서 오류 처리
  보다 복잡한 오류 처리 및 복구가 필요한 JSP 파일의 경우, 데이터 bean에서 오류를 직접 처리하기 위해 파일을 작성할 수 있습니다. JSP 파일은 데이터 bean의 활성화 방법에 따라, 데이터 bean에서 발생된 예외를 발견하거나 각 데이터 bean에서 오류 코드 세트를 확인할 수 있습니다. 그러면 JSP 파일은 수신된 오

류에 따라 적절한 복구 조치를 수행할 수 있습니다. JSP 파일은 다음 오류 처리 범위의 조합을 사용할 수 있음에 유의하십시오.

- 페이지 레벨의 오류 JSP
  JSP 파일은 JSP 오류 태그를 통해 자신에게서 발생하는 예외에서 고유한 기본 오류 JSP 템플리트를 지정할 수도 있습니다. 이를 통해 JSP 프로그램이 자체 오류 처리를 지정할 수 있습니다. JSP 오류 태그를 지정하지 않는 JSP 파일은 응용프로그램 레벨 JSP 오류 템플리트에 오류를 발생시킵니다. 페이지 레벨 오류 JSP의 경우, JSP 헬퍼 클래스(com.ibm.server.JSPHelper)를 호출하여 현재 트랜잭션을 롤백해야 합니다.
- 응용프로그램 레벨의 오류 JSP
  WebSphere 내의 응용프로그램은 해당 Servlet 또는 JSP 파일에서 예외 발생 시 기본 오류 JSP 템플리트를 지정할 수 있습니다. 응용프로그램 레벨 오류 JSP 템플리트는 몰 레벨 또는 상점 레벨(단일 상점 모델용) 오류 핸들러로 사용할 수 있습니다. 응용프로그램 레벨 오류 JSP 템플리트의 경우, Servlet 헬퍼 클래스를 호출하여 현재 트랜잭션을 롤백해야 합니다. 이는 웹 컨트롤러가 트랜잭션을 롤백하기 위한 실행 경로에 있지 않기 때문입니다. 가능한 경우, 앞의 두 가지 JSP 오류 처리 유형에 의존해야 합니다. 응용프로그램 레벨 오류 처리 전략은 필요한 경우에만 사용하십시오.

제 6 장 명령 구현

이 장에서는 새 컨트롤러, 태스크 및 데이터 bean 명령의 작성 방법에 대해 설명합니다. 또한 기존 컨트롤러, 태스크 및 데이터 bean 명령의 확장 방법에 대해서 설명합니다.

주: Business 이 장에서는 비즈니스 정책 명령에 대해 설명하지 않습니다. 비즈니스 정책 명령에 대한 정보는 185 페이지의 제 7 장『거래 계약 및 비즈니스 정책(Business Edition)』을 참조하십시오.

새 명령 - 소개

WebSphere Commerce 프로그래밍 모델은 네 가지 유형의 명령, 즉 컨트롤러, 태스크, 뷰 및 데이터 bean 명령을 정의합니다. 전자 상거래 응용프로그램의 새 비즈니스 로직 작성 시, 새 컨트롤러, 태스크 및 데이터 bean 명령을 작성해야 할 수 있습니다. 새 뷰 명령은 작성하지 않아도 됩니다. 뷰 명령에 대한 추가 정보는 이 절의 뒷 부분에 있습니다.

새 명령은 해당 인터페이스를 구현해야 하며 이는 기존 인터페이스에서 확장되어야 합니다. 명령 작성을 간소화하기 위해 WebSphere Commerce에는 각 명령 유형에 대한 추상 구현 클래스가 포함됩니다. 새 명령은 이 클래스에서 확장되어야 합니다.

다음 테이블은 새 명령을 확장할 구현 클래스 및 새 명령이 구현해야 하는 인터페이스에 대한 정보의 개요를 제공합니다.

명령 유형	명령 이름 예	클래스	인터페이스 예 구현
컨트롤러 명령	MyControllerCmdImpl	com.ibm.commerce.command.ControllerCommandImpl	MyControllerCmd
태스크 명령	MyTaskCmdImpl	com.ibm.commerce.command.TaskCommandImpl	MyTaskCmd

명령 유형	명령 이름 예	클래스	인터페이스 예 구현
데이터 bean 명령	MyDataBeanCmdImpl	com.ibm.commerce.command.DataBeanCommandImpl	MyDataBean

주: 구현 클래스 이름의 공백은 표시 목적일 뿐입니다.

다음 도표는 새 컨트롤러 명령의 인터페이스 및 구현 클래스와 기존 추상 구현 클래스 및 인터페이스와의 관계에 대해 설명합니다. 추상 클래스 및 인터페이스 모두 com.ibm.commerce.command package 패키지에 있습니다.

새 컨트롤러 명령

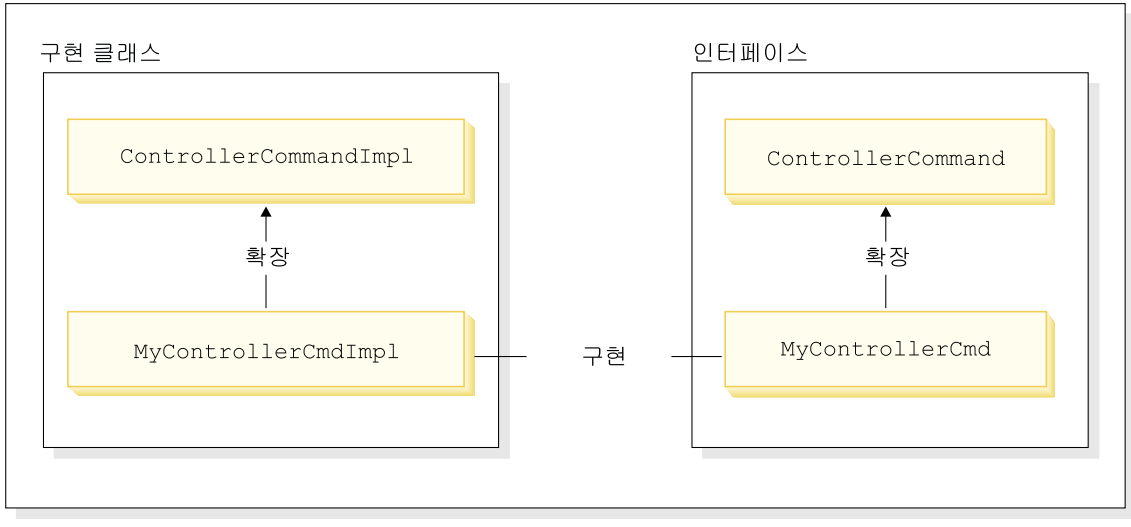


그림 25.

다음 도표는 새 태스크 명령의 인터페이스 및 구현 클래스와 기존 추상 구현 클래스 및 인터페이스와의 관계에 대해 설명합니다. 추상 클래스 및 인터페이스 모두 com.ibm.commerce.command 패키지에 있습니다.

새 태스크 명령

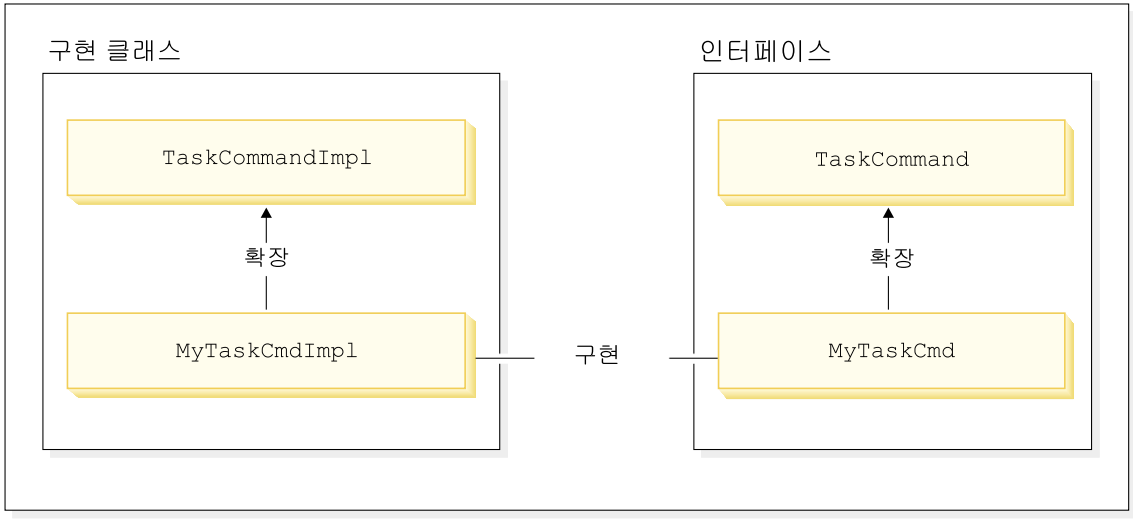


그림 26.

다음 도표는 새 데이터 bean 명령의 인터페이스 및 구현 클래스와 기존 추상 구현 클래스 및 인터페이스와의 관계에 대해 설명합니다. 추상 클래스 및 인터페이스 모두 com.ibm.commerce.command 패키지에 있습니다.

새 데이터 **bean** 명령

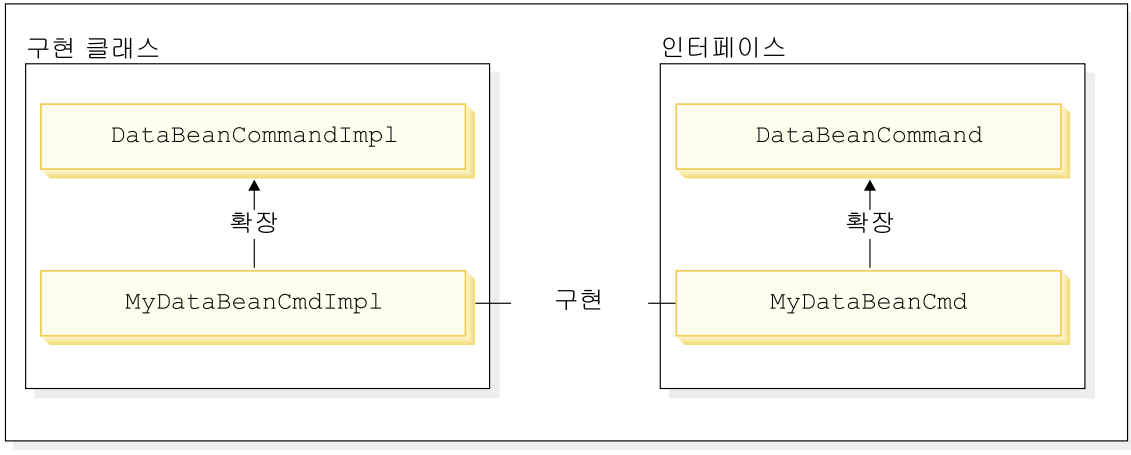


그림 27.

뷰 명령의 두 가지 기본 기능은 응답 포맷 및 클라이언트로 응답 보내기입니다. 다른 프로토콜을 사용하여 클라이언트에 응답을 보내는 여러 개의 일반 뷰 명령이 제공됩니다. 포맷팅 기능은 일반적으로 JSP 템플리트를 호출하는 뷰 명령이 처리합니다. 예를 들어, RedirectViewCommand 뷰 명령은 응답을 가져올 수 있는 URL 경로로 클라이언트를 지정합니다. 그런 다음 응답은 지정된 JSP 템플리트에 의해 포맷됩니다. ForwardViewCommand 뷰 명령은 포맷을 위해 요청을 JSP 템플리트로 전달하고, 클라이언트에게 페이지가 표시됩니다.

이 뷰 명령 모델을 사용하면 새 JSP 템플리트를 작성하여 새 *뷰*(클라이언트 응답)를 작성할 수 있습니다. 그러나 JSP 템플리트는 기존 뷰 명령 중 하나에 의해 호출되어야 합니다.

사용자 정의 코드 패키지화

사용자 정의 코드 작성 시 특정 코드 조직 구조를 따라야 합니다. 일반적으로 사용자 정의 코드는 사용자 정의 코드에 사전 정의되어 있는 WebSphere Commerce 작업 영역의 프로젝트에서 유지보수됩니다. 두 개의 사전 정의 프로젝트, 즉 WebSphereCommerceServerExtensionsLogic 프로젝트 및 WebSphereCommerceServerExtensionsData 프로젝트가 제공됩니다. 첫 번째 프로젝트는 명령 및 데이터 bean 로직용이고, 두 번째 프로젝트는 사용자가 작성하는 엔터프라이즈 bean용입니다.

새 명령 작성 시 비즈니스 요구사항에 맞게 이름 지정한 패키지에 이 명령을 배치해야 합니다. 즉, 명령이 특정 상점에 적용되는 경우, 상점에 고유한 패키지에 이 명령을 패키지화하십시오. 둘 이상의 상점에 명령을 적용하는 경우, 그에 따라 패키지화하십시오. 예를 들어, 다음 패키지가 있을 수 있습니다.

- `com.bigbusiness.storeA.commands`
- `com.bigbusiness.storeB.commands`
- `com.bigbusiness.commands`

위의 패키지화 구조는 상점 레벨에서의 비즈니스 로직 간의 차별화를 허용합니다.

새 데이터 bean 작성 시, 이 bean은 명령 로직과 다른 패키지에 보관해야 합니다. 그러나 이 패키지는 명령 패키지를 저장하는 프로젝트

(WebSphereCommerceServerExtensionsLogic)에 보관해야 합니다. 위 예에서 com.bigbusiness.databeans 패키지를 WebSphereCommerceServerExtensionsLogic 프로젝트에 배치합니다.

새 엔티티 bean을 작성하면 WebSphereCommerceServerExtensionsData 프로젝트에 저장해야 합니다. 이로써 com.bigbusiness.objects 패키지가 있는 WebSphereCommerceServerExtensionsData 프로젝트를 갖게 됩니다.

코드 전개를 위해 이 패키징 전략이 필요합니다.

명령 컨텍스트

명령은 명령 컨텍스트를 사용하여 웹 컨트롤러에서 정보를 얻을 수 있습니다. 사용 가능한 정보 예에는 사용자 ID, 사용자 오브젝트, 언어 식별자 및 상점 식별자가 포함됩니다.

명령을 작성하면 명령의 상위 클래스에서 getCommandContext() 메소드를 호출하여 명령 컨텍스트에 액세스할 수 있습니다. 웹 컨트롤러에서 명령이 호출되는 경우 명령 컨텍스트는 컨트롤러 명령으로 설정됩니다. 컨트롤러 명령은 처리 시 호출되는 태스크 또는 컨트롤러 명령으로 명령 컨텍스트를 전파해야 합니다. 명령은 명령 컨텍스트에서 다음 키 정보를 가져올 수 있습니다.

getUserId() and getUser()

현재 사용자 ID 또는 사용자 오브젝트를 가져옵니다. 현재 세션의 사용자 ID는 세션 컨텍스트에 저장됩니다. 세션 컨텍스트는 두 가지 방법 중 하나, 즉 WebSphere Commerce 쿠키를 사용하거나 WebSphere Application Server 지속 세션 오브젝트를 사용하여 지속될 수 있습니다. 명령 컨텍스트는 명령에서 세션 관리의 복잡도를 숨깁니다.

getStoreId(), getStore(), and getStore(storeId)

현재 요청과 연관된 상점을 가져옵니다. 웹 컨트롤러는 URL에 상점 ID를 리턴합니다. 상점 ID가 URL에 지정되어 있지 않은 경우, 이전 요청에서 저장한 세션 오브젝트에서 검색할 수 있습니다. WebSphere Commerce 런타임 환경은 자주 액세스하는 오브젝트 세트를 유지보수합니다. 예를 들어, 상점 오브젝트 세트를 유지보수합니다. 명령은 언제나 명령 컨텍스트에서 상점 오브젝트를 가져와 웹 컨트롤러에서 오브젝트 캐시를 이용해야

합니다. getStore() 메소드를 호출하여 현재 상점을 가져오거나, 명령 컨텍스트에서 getStore(storeId) 메소드를 호출하여 특정 상점 오브젝트를 가져올 수 있습니다.

getLanguageId()

현재 요청에 사용해야 하는 언어 ID를 리턴합니다. 웹 컨트롤러는 자국어 프레임워크를 구현합니다. 이 프레임워크의 배경 개념은 사용자가 선호하고 상점에서 지원하는 언어를 판별하는 것입니다. URL에 언어 ID가 있는 경우, 웹 컨트롤러가 상점에서 이 언어를 지원하는지 여부를 판별하며 지원하는 경우 이것이 getLanguageId() 메소드가 리턴하는 언어 ID입니다. URL에 언어 ID가 없는 경우, 웹 컨트롤러가 의사결정 트리를 탐색하여 현재 세션 오브젝트 또는 사용자의 등록 환경설정에 언어 ID(상점에서 지원하는)가 있는지 여부를 판별하거나, 마지막으로 상점의 기본 언어 ID를 리턴합니다.

getCurrency()

현재 요청에 사용할 통화를 리턴합니다. 통화는 자국어 프레임워크의 일부이므로 이 메소드의 배경 로직은 getLanguageId() 메소드의 배경 로직과 유사합니다.

getCurrentTradingAgreements() 및 getTradingAgreement (tradingAgreementId)

현재 세션에 사용되는 거래 계약 세트를 리턴합니다. 이 세트는 사용자가 권리를 부여 받은 모든 거래 계약이거나, ContractSetInSession 명령으로 정의된 서브세트일 수 있습니다. 명령은 언제나 명령 컨텍스트에서 거래 계약 오브젝트를 가져와 웹 컨트롤러에서 오브젝트 캐시를 이용해야 합니다. getCurrentTradingAgreements() 메소드를 호출하여 현재 거래 계약을 가져오거나, 명령 컨텍스트에서 getTradingAgreement(tradingAgreementId) 메소드를 호출하여 특정 거래 계약을 가져올 수 있습니다.

명령 컨텍스트는 읽기 전용 오브젝트로 사용되어야 합니다. 해당 setter 메소드를 호출해서는 안 됩니다. setter 메소드는 WebSphere Commerce 런타임 환경에서 사용하기 위해 예약되었으며 향후 릴리스에서는 무시될 수 있습니다.

명령 컨텍스트 API(Application Programming Interface)에 대한 모든 정보는 WebSphere Commerce Production 및 Development 온라인 도움말의 "참조"를 참조하십시오.

URL 명령에 대한 컨텍스트 정보의 임시 변경

일부 명령 컨텍스트 정보를 대체하고, 다른 상점의 컨텍스트에서 또는 다른 사용자를 대신해 URL 명령을 실행할 수 있습니다. URL 명령에는 명령 컨텍스트의 이러한 임시 전환을 허용하는 다음 URL 입력 매개변수가 있습니다.

- forStoreId
- forUser
- forUserId

forStoreId URL 입력 매개변수를 사용하면 이 특정 URL 요청에 사용할 상점 ID를 지정할 수 있습니다. 이 매개변수는 결과적으로 명령 컨텍스트의 상점 ID 값을 지정된 상점의 ID 값으로 임시 변경하지만, 이 변경은 URL 명령의 지속 기간 동안에만 유효합니다.

forUser 및 forUserId URL 입력 매개변수를 사용하면 현재 로그인되어 있는 사용자가 다르더라도 지정된 사용자에 대해 명령이 실행되도록 지정할 수 있습니다. 이는 특히 고객 서비스 담당자가 고객을 지원해야 하는 경우 유용합니다. 예를 들어, 고객 서비스 담당자는 URL 입력 매개변수를 사용하여 고객의 사용자 이름 또는 사용자 ID를 지정함으로써, 고객 대신 고객의 주소 정보를 갱신할 수 있습니다. 이러한 사용자 정보 변경은 변경이 지정된 URL 요청의 지속 기간 동안에만 유효합니다.

새 컨트롤러 명령

앞에서 설명한 대로 새 컨트롤러 명령은 추상 컨트롤러 명령 클래스(com.ibm.commerce.command.ControllerCommandImpl)에서 확장해야 합니다. 새 컨트롤러 명령 작성 시 추상 클래스에서 다음 메소드를 대체해야 합니다.

- isGeneric()
- isRetriable()

- setRequestProperties(com.ibm.commerce.datatype.TypedProperty reqParms)
- validateParameters()
- getResources()
- performExecute()

이 각각의 메소드에 대한 추가 정보는 다음 절에서 설명합니다.

isGeneric 메소드

표준 WebSphere Commerce 구현에는 여러 사용자 유형이 있습니다. 이 유형에는 일반, 게스트 및 등록된 사용자가 포함됩니다. 등록된 사용자의 그룹에는 고객 및 운영자가 있습니다.

일반 사용자에는 전체 시스템에서 사용되는 일반 사용자 ID가 있습니다. 이 일반 사용자 ID는 시스템 자원 사용량을 최소화하는 방법으로 사이트에서 일반 찾아보기를 지원합니다. 웹 컨트롤러는 사용자 오브젝트에서 일반 사용자가 호출할 수 있는 명령을 검색하지 않아도 되므로 일반 찾아보기에는 이 일반 사용자 ID를 사용하는 것이 보다 효율적입니다.

isGeneric 메소드는 일반 사용자가 명령을 호출할 수 있는지 여부를 지정하는 부울 값을 리턴합니다. 컨트롤러 명령의 상위 클래스의 isGeneric 메소드는 값을 FALSE로 설정합니다. 즉, 호출자는 등록된 사용자 또는 게스트 사용자여야 합니다. 일반 사용자가 새 컨트롤러 명령을 호출할 수 있는 경우, TRUE를 리턴하도록 이 메소드를 대체하십시오.

새 명령이 사용자와 연관된 자원을 가져오거나 작성하지 않는 경우, TRUE를 리턴하도록 이 메소드를 대체해야 합니다. 일반 사용자가 호출할 수 있는 명령 예는 ProductDisplay 명령입니다. 사용자가 상품을 볼 수 있도록 허용하는 것이 바람직합니다. 사용자가 게스트 또는 등록된 사용자여야 하는(즉, isGeneric이 FALSE 리턴) 명령 예는 OrderItemAdd 명령입니다.

isGeneric이 TRUE 값을 리턴하면 웹 컨트롤러가 현재 세션에 대한 새 사용자 오브젝트를 작성하지 않습니다. 즉, 웹 컨트롤러가 사용자 오브젝트를 검색하지 않아도 되므로 일반 사용자가 호출할 수 있는 명령이 더 빠르게 실행됩니다.

일반 사용자가 명령을 호출할 수 있도록 이 메소드를 사용하는 구문은 다음과 같습니다.

```
public boolean isGeneric()
{
     return true;
}
```

isRetriable 메소드

isRetriable 메소드는 트랜잭션 롤백 예외에서 명령을 재시도할 수 있는지 여부를 지정하는 부울 값을 리턴합니다. 새 컨트롤러 명령의 상위 클래스의 isRetriable 메소드는 FALSE 값을 리턴합니다. 트랜잭션 롤백 예외에서 명령을 재시도할 수 있는 경우, 이 메소드를 대체하고 TRUE 값을 리턴해야 합니다.

트랜잭션 예외 시 재시도해서는 안 되는 명령 예는 OrderProcess 명령입니다. 이 명령은 제삼자 지불 권한 부여 프로세스를 호출합니다. 이 권한 부여는 되돌릴 수 없으므로 이 명령을 재시도할 수 없습니다. 재시도할 수 없는 명령 예는 ProductDisplay 명령입니다.

트랜잭션 롤백 예외 시 명령을 재시도 하기 위한 구문은 다음과 같습니다.

```
public boolean isRetriable()
{
   return true;
}
```

setRequestProperties 메소드

setRequestProperties 메소드는 모든 입력 특성을 컨트롤러 명령으로 전달하기 위해 웹 컨트롤러가 호출합니다. 컨트롤러 명령은 입력 특성의 구문을 분석하고 각 개별 특성을 이 메소드에 명시적으로 설정해야 합니다. 컨트롤러 명령으로 특성을 명시적으로 설정하면 유형 안전 특성의 개념이 승격됩니다.

이 메소드 사용 구문은 다음과 같습니다.

```
public void setRequestProperties(
   com.ibm.commerce.datatype.TypedProperty reqParms)
{
```

```
        // parse the input properties and explicitly set each parameter
    }
```

validateParameters 메소드

validateParameters 메소드는 초기 매개변수 확인 및 매개변수의 필요 분석을 수행하는 데 사용됩니다. 예를 들어, `orderId=*`를 분석하는 데 사용될 수 있습니다. 이 메소드는 getResources 및 performExecute 메소드 이전에 호출됩니다. 이 순서에 대한 자세한 정보는 119 페이지의 『액세스 제어 상호작용』을 참조하십시오.

getResources 메소드

이 메소드는 자원 레벨 액세스 제어를 구현하는 데 사용됩니다. 명령이 조치를 수행하려는 자원 조치 쌍의 벡터를 리턴합니다. 아무 것도 리턴되지 않는 경우, 자원 레벨 액세스 제어가 수행되지 않습니다. 액세스 제어에 대한 추가 정보는 105 페이지의 제 4 장 『액세스 제어』를 참조하십시오.

performExecute 메소드

performExecute 메소드에는 명령에 대한 비즈니스 로직이 들어 있습니다. 이 메소드는 새 비즈니스 로직이 실행되기 이전에 명령의 상위 클래스의 performExecute 메소드를 호출해야 합니다. 결국 뷰 이름을 리턴해야 합니다.

다음은 새 컨트롤러 명령의 performExecute 메소드에 대한 구문 예를 표시합니다. 이 경우, 응답은 경로 재지정 뷰 명령을 사용하지만, 전달 뷰 명령 또는 경로 지정 뷰 명령을 사용할 수 있습니다.

```
public void performExecute() throws ECException {
 super.performExecute();

//////////////////////////////////////////////
//  your business logic                     //
//////////////////////////////////////////////

// Create a new TypedProperty for response properties.
 TypedProperty rspProp = new TypedProperty();

// set response properties
rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");
//////////////////////////////////////////////////
```

```
//  The following line is optional.  The VIEWREG   //
//  table can specify the redirect URL.            //
////////////////////////////////////////////////////

rspProp.put(ECConstants.EC_REDIRECTURL, MyURL);

//////////////////////////////////////////////////////////////////////////
//   If you are using a forward view, you can set the                   //
//   response properties as follows:                                    //
//   TypedProperty rspProp = new TypedProperty();                       //
//   rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");                //
//   rspProp.put(ECConstants.EC_DOCPATHNAME, "MyJSP.jsp");              //
//                                                                      //
//   Again, it is optional to explicitly set the name of the JSP template.//
//   The VIEWREG table can specify the JSP template.                    //
//////////////////////////////////////////////////////////////////////////

 setResponseProperties( rspProp );
}
```

performExecute 메소드에 URL을 다시 지정하고 VIEWREG 테이블에 항목이 존재하는 경우, 코드에 지정되어 있는 값이 VIEWREG 테이블의 값에 우선합니다. 코드 내 JSP 템플리트의 스펙에도 동일한 우선순위가 적용됩니다.

장기 컨트롤러 명령

컨트롤러 명령을 실행하는 데 긴 시간이 소요되는 경우, 명령을 두 개의 명령으로 분할할 수 있습니다. URL 요청의 결과로 실행되는 첫 번째 명령은 스케줄러에 두 번째 명령을 추가하여 배경 작업으로 실행되도록 합니다. 이 내용은 다음 도표에서 설명합니다.

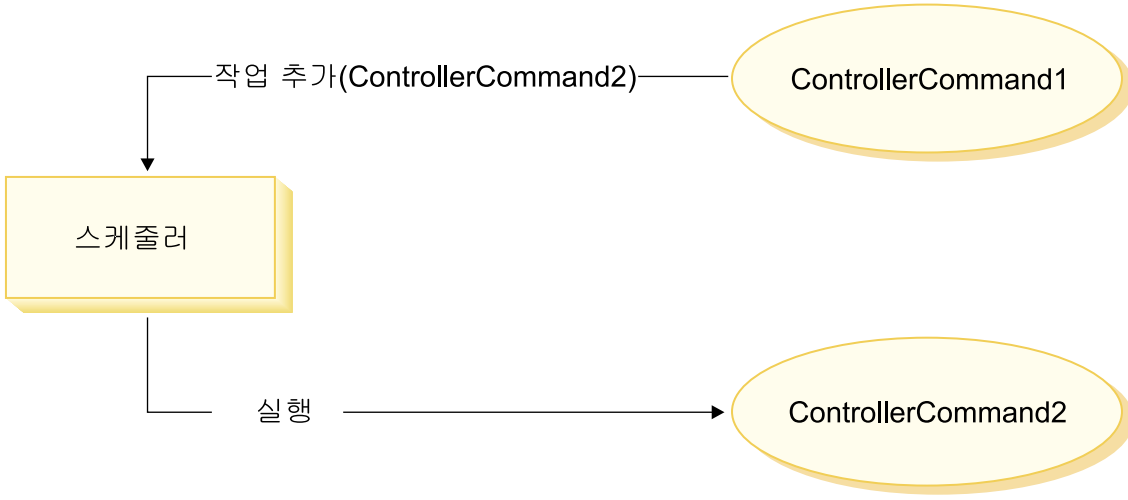


그림 28.

앞 도표에 표시되는 플로우는 다음과 같습니다.

1. ControllerCommand1이 URL 요청의 결과로 실행됩니다.
2. ControllerCommand1이 스케줄러에 작업을 추가합니다. 작업은 ControllerCommand2입니다. ControllerCommand1은 스케줄러에 작업을 추가한 직후 뷰를 리턴합니다.
3. 스케줄러가 ControllerCommand2를 배경 작업으로 실행합니다.

이 시나리오의 경우, 클라이언트는 일반적으로 ControllerCommand2의 결과를 폴합니다. ControllerCommand2는 작업 상태를 데이터베이스에 기록해야 합니다.

뷰 명령에 대한 입력 특성 포맷팅

컨트롤러 명령이 완료되면 실행해야 하는 뷰 이름을 리턴합니다. 이 뷰에는 몇 가지 입력 특성이 전달되어야 합니다. 이 입력 매개변수의 세 가지 소스는 다음 목록에서 설명합니다.

- CMDREG 테이블의 PROPERTIES 열에 저장되어 있는 기본 특성
- VIEWREG 테이블에 있는 PROPERTIES열의 기본 특성
- URL의 입력 특성

JSP 템플리트의 속성에 이러한 특성을 병합하고 설정하는 방법에 대한 추가 정보는 49 페이지의 『JSP 속성 설정 - 개요』를 참조하십시오. 이 절에서는 뷰 명령에 대한 입력 특성의 포맷 방법에 대해 설명합니다.

뷰 명령을 경로 재지정하려면 다음 두 주제를 점검하십시오.

- URL 경로 재지정을 지원하도록 조회 문자열 수정
- URL 경로 재지정 길이 제한 처리

전달 뷰 명령의 경우, 입력 매개변수의 열거 및 HttpServletRequestObject의 속성으로 설정 주제를 점검해야 합니다.

입력 매개변수를 HttpRedirectView에 대한 조회 문자열로 수정

경로 재지정 뷰 명령으로 전달되는 모든 입력 매개변수는 URL 경로 재지정을 위한 조회 문자열로 수정되어야 합니다. 예를 들어, 경로 재지정 뷰 명령의 입력이 다음 특성을 갖는 것으로 가정합니다.

```
URL = "MyView?p1=v1&p2=v2";
ip1 = "iv1";   // input to orginal controller command
ip2 = "iv2" ;  // input to original controller command
op1 = "ov1";
op2 = "ov2";
```

앞의 입력 매개변수에 따라 최종 URL은 다음과 같습니다.

```
MyView?p1=v1&p2=v2&ip1=iv1&ip2=iv2&op1=ov1&op2=ov2
```

명령이 SSL을 사용하기 위한 것인 경우, 매개변수가 암호화되고 최종 URL은 다음과 같이 나타남에 유의하십시오.

```
MyView?krypto=encrypted_value_of"p1=v1&p2=v2&ip1=iv1&ip2=iv2&op1=ov1&op2=ov2"
```

URL 경로 재지정의 길이 제한 처리

기본적으로 컨트롤러 명령에 대한 모든 입력 매개변수는 경로 재지정 뷰 명령으로 전달됩니다. URL 경로 재지정의 문자 수에 제한이 있는 경우, 문제점이 발생할 수 있습니다. 길이가 제한되는 경우의 예는 클라이언트가 Internet Explorer 브라우저를 사용하는 경우입니다. 이 브라우저의 경우, URL은 2083바이트를 초과할 수 없습니다. URL이 이 한계를 초과하는 경우, URL은 잘리게 됩니다. 또한 입

력 매개변수가 많거나 암호화를 사용하는 경우, 암호화된 문자열 길이는 일반적으로 암호화되지 않은 문자열 길이의 두세배이므로 문제점이 발생할 수 있습니다.

URL 경로 재지정의 길이 제한을 처리하는 데는 다음 두 가지 접근 방법이 있습니다.

1. 경로 재지정 뷰 명령으로 전달되어야 하는 매개변수 세트만을 리턴하도록 컨트롤러 명령에서 getViewInputProperties 메소드를 대체하십시오.
2. URL 매개변수에 지정된 특수 문자를 사용하여 입력 매개변수 문자열에서 제거될 수 있는 매개변수를 표시하십시오.

앞의 접근 방법 각각을 설명하려면 컨트롤러 명령에 대한 다음 입력 매개변수 세트를 고려하십시오.

```
URL="MyView";
// All of the following are inputs to the original controller command.
ip1="ipv1";
ip2="ipv2";
ip3="ipv3";
iq1="iqv1";
iq2="iqv2";
ir1="ipr1";
ir2="ipr2";
is="isv";
```

getViewInputProperties 메소드를 대체하는 경우, 다음 매개변수만이 뷰 명령으로 전달되도록 새 메소드를 작성할 수 있습니다.

```
ir2="ipr2";
is="isv";
```

두 번째 접근 방법을 사용하는 경우, 제거되어야 할 특정 입력 매개변수를 표시하는 특수 매개변수를 사용하여 뷰 명령을 호출할 수 있습니다. 예를 들어, URL 매개변수로 다음을 지정하여 같은 결과를 얻을 수 있습니다.

```
URL="MyView?ip*=&iq*=&ir1="
```

이 URL 매개변수는 WebSphere Commerce 런타임 프레임워크에 다음을 지시합니다.

- `ip*=` 스펙은 이름이 `ip`로 시작하는 모든 매개변수를 제거해야 함을 의미합니다.

- `iq*=` 스펙은 이름이 `iq`로 시작하는 모든 매개변수를 제거해야 함을 의미합니다.
- `ir1=` 스펙은 `ir1` 매개변수를 제거해야 함을 의미합니다.

HttpForwardView의 HttpServletRequest 오브젝트에 속성 설정

기본 HttpForwardViewCommandImpl은 명령에 전달된 모든 매개변수를 열거하고 이를 HttpServletRequest 오브젝트에 속성으로 설정합니다.

예를 들어, 전달 뷰 명령으로 전달된 requestProperties 오브젝트에 다음 매개변수가 있다고 가정하십시오.

```
p1="pv1";
p2="pv2";
p3=pv3;   // pv3 is an object
```

그러면 다음 속성이 request.setAttribute() 메소드를 사용하여 JSP 템플리트로 전달됩니다.

```
request.setAttribute("p1", "pv1");
request.setAttribute("p2", "pv2");
request.setAttribute("p1", pv1);
request.setAttribute("RequestProperties", requestProperties);
request.setAttribute("CommandContext", commandContext);
```

여기서, `requestProperties`는 명령으로 전달되는 TypedProperty 오브젝트이고, `commandContext`는 명령으로 전달되는 명령 컨텍스트 오브젝트이며 `p1`, `p2` 및 `p3`은 requestProperties 오브젝트에 정의되어 있는 매개변수입니다.

데이터베이스 확약 및 컨트롤러 명령에 대한 롤백

컨트롤러 명령 실행 동안 자주 데이터가 작성 또는 갱신됩니다. 대부분의 경우, 트랜잭션 종료 시 데이터베이스에 새 정보를 갱신해야 합니다. 트랜잭션은 웹 컨트롤러가 관리합니다.

웹 컨트롤러는 컨트롤러 명령을 호출하기 전에 트랜잭션 시작을 표시합니다. 컨트롤러 명령의 실행이 완료되면 컨트롤러 명령이 웹 컨트롤러에 뷰 이름을 리턴합니다. 웹 컨트롤러는 트랜잭션 종료를 표시해야 합니다. 실제 트랜잭션 종료 시점(뷰 호출 이전 또는 이후)은 사용된 뷰 유형에 따라 다릅니다.

뷰 명령에는 다음 세 가지 유형이 있습니다.

- 전달 뷰 명령
- 경로 재지정 뷰 명령
- 직접 뷰 명령

웹 컨트롤러는 VIEWREG 테이블에서 뷰 이름을 검색하여, 뷰에 사용될 뷰 명령을 결정합니다.

VIEWREG 테이블의 항목이 ForwardViewCommand 사용을 지정하는 경우, 웹 컨트롤러가 컨트롤러 명령의 결과를 해당 ForwardViewCommand 구현 클래스(이 또한 VIEWREG에 지정되어 있음)로 전달합니다. 뷰 명령은 현재 트랜잭션의 컨텍스트에서 실행됩니다. 이 경우, 뷰 명령이 완료되어야 데이터베이스 확약 또는 롤백이 발생합니다.

VIEWREG 테이블의 항목이 RedirectViewCommand 사용을 지정하는 경우, 웹 컨트롤러가 컨트롤러 명령의 결과를 해당 RedirectViewCommand implementation 클래스로 전달합니다. 그러면 뷰 명령은 현재 트랜잭션 범위의 외부에서 실행되며 경로 재지정 뷰 명령이 호출되기 전에 데이터베이스 확약 또는 롤백이 발생합니다.

VIEWREG 테이블의 항목이 DirectViewCommand 사용을 지정하는 경우, 웹 컨트롤러가 컨트롤러 명령의 결과를 해당 DirectViewCommand 구현 클래스로 전달합니다. 뷰 명령은 현재 트랜잭션의 컨텍스트에서 실행됩니다. 이 경우, 뷰 명령이 완료되어야 데이터베이스 확약 또는 롤백이 발생합니다. ForwardViewCommand와 DirectViewCommand는 유사함에 유의하십시오. ForwardViewCommand는 결과를 JSP 템플리트로 전달합니다. 반면에 DirectViewCommand는 결과를 입력 스트림으로 받아 출력 스트림으로 전달합니다. 데이터를 바이트로 취급하는 getRawDocument 메소드를 사용하거나 데이터를 텍스트로 취급하는 getTextDocument를 사용합니다.

컨트롤러 명령과 동일한 트랜잭션 범위에서 뷰 명령을 실행하는 경우, 뷰 명령의 오류로 전체 트랜잭션의 롤백이 발생합니다. 이는 사용자의 비즈니스 로직에 따라 바람직하거나 바람직하지 않은 결과일 수 있습니다.

컨트롤러 명령을 사용하는 트랜잭션 범위 예

컨트롤러 명령에 대한 트랜잭션 범위의 차이점을 설명하려면 사용하는 뷰 명령의 유형에 따라 다음 예를 참조하십시오.

사례 1: 컨트롤러 명령 트랜잭션의 범위에서 뷰 실행

새 컨트롤러 명령 YourControllerCmdA를 작성한 것으로 가정하십시오. 그러면 명령의 performExecute 메소드는 다음을 포함하게 됩니다.

```
.
.
// Create a new TypedProperty object for output.
 TypedProperty rspProp = new TypedProperty();

//////////////////////
// Business logic   //
//////////////////////

// Return the view
rspProp.put(ECConstants.EC_VIEWTASKNAME, "YourView");
SetResponseProperties(rspProp);
```

이 코드 단편의 경우, 컨트롤러 명령은 “YourView”를 뷰로 리턴합니다. YourView는 VIEWREG 테이블에 등록됩니다. 다음은 YourView를 등록하기 위한 insert 문 예입니다.

```
insert into VIEWREG (ViewName, DeviceFmt_id, storeEnt_id, interfacename,
classname, properties)

values ('YourView',  -1, XX,'com.ibm.commerce.command.ForwardViewCommand',
'com.ibm.commerce.command.HttpForwardViewCommandImpl','docname=YourView.jsp');
```

여기서, XX는 상점 식별자입니다. 뷰가 com.ibm.commerce.command.HttpForwardViewCommandImpl 구현 클래스를 사용하므로, 웹 컨트롤러는 일반 전달 뷰 명령을 사용합니다.

웹 컨트롤러는 앞의 명령 등록에 따라 컨트롤러 명령 트랜잭션의 범위에서 YourView.jsp 파일을 실행합니다. YourView.jsp에 오류가 발생하는 경우, 트랜잭션이 실패하고 데이터베이스 롤백이 발생합니다. 결과적으로 전체 컨트롤러 명령이 실패합니다.

사례 2: 컨트롤러 명령 트랜잭션의 범위 밖에서 뷰 실행

뷰에 오류가 발생하는 경우에도 데이터베이스에 정보를 확약하려고 하는 것으로 가정하십시오. 컨트롤러 명령의 트랜잭션 범위 밖에서 뷰를 실행하려면 뷰를 경로 재지정으로 실행해야 합니다.

뷰를 경로 재지정으로 실행하기 위해 컨트롤러 명령의 performExecute 메소드가 다음과 같은 방식으로 뷰를 리턴합니다.

```
.
.
// Create a new TypedProperty object for output.
 TypedProperty rspProp = new TypedProperty();

//////////////////////
// Business logic   //
//////////////////////

// Return the view
rspProp.put(ECConstants.EC_VIEWTASKNAME, EC_GENERIC_REDIRECTVIEW);
rspProp.put(EC_Constants.EC_REDIRECTURL, "YourView2");
```

다음 예 SQL문은 경로 재지정 전략을 지원합니다.

```
insert into VIEWREG (ViewName, DeviceFmt_id, storeEnt_id, interfacename,
classname, properties)

values ('YourView2',  -1, XX,'com.ibm.commerce.command.ForwardViewCommand',
'com.ibm.commerce.command.HttpForwardViewCommandImpl','docname=YourView2.jsp');
```

여기서, XX는 상점 식별자입니다.

명령이 EC_GENERIC_REDIRECTVIEW 값을 응답 특성 매개변수로 전달하므로, 웹 컨트롤러는 일반 경로 재지정 뷰 명령을 사용합니다. 일반 경로 재지정 뷰는 다음과 같은 정보로 VIEWREG 테이블에 등록됩니다.

- ViewName = RedirectView
- DeviceFmt_Id = -1
- InterfaceName = com.ibm.commerce.command.RedirectViewCommand
- ClassName = com.ibm.commerce.command.HttpRedirectViewCommandImpl

웹 컨트롤러는 URL 경로 재지정을 입력 특성으로 사용하는 일반 경로 재지정 뷰 명령을 호출합니다. 응답은 URL 경로 재지정입니다. 경로 재지정이 발생하면 YourView2가 호출됩니다. 그런 다음 이것은 일반 전달 뷰로 구현됩니다.

새 태스크 명령

새 태스크 명령은 추상 태스크 명령 클래스(com.ibm.commerce.command.TaskCommandImpl)에서 확장해야 하며 com.ibm.commerce.command.TaskCommand 인터페이스를 확장하는 인터페이스를 구현해야 합니다. 159 페이지의 도표에 표시되어 있는 것처럼 새 태스크 명령은 다음과 같이 정의되어야 합니다.

```
public class MyTaskCmdImpl extends com.ibm.commerce.command.TaskCommandImpl
     implements MyTaskCmd {

}
```

태스크 명령의 모든 입력 및 출력 특성은 명령 인터페이스(예: MyTaskCmd)에 정의되어야 합니다. 호출자는 태스크 명령 구현 클래스가 아닌 태스크 명령 인터페이스에 대한 프로그램을 수행합니다. 이를 통해 호출자가 호출할 구현 클래스에 신경쓰지 않고 태스크 명령을 복수 구현할 수 있습니다(각 상점에 하나씩).

인터페이스에 정의되어 있는 모든 메소드는 구현 클래스에서 구현해야 합니다. 명령 컨텍스트는 호출자(컨트롤러 명령)가 설정해야 하므로, 태스크 명령은 명령 컨텍스트를 설정하지 않아도 됩니다. 그러나 태스크 명령은 명령 컨텍스트를 사용하여 웹 컨트롤러에서 정보를 얻을 수 있습니다.

태스크 명령 인터페이스에 정의되어 있는 메소드의 구현 외에, com.ibm.commerce.command.TaskCommandImpl 클래스에서 performExecute 메소드를 대체해야 합니다.

performExecute 메소드에는 태스크 명령이 수행하는 특정 작업 단위에 대한 비즈니스 로직이 있습니다. 이 메소드는 새 비즈니스 로직을 실행하기 전에 태스크 명령의 상위 클래스의 performExecute 메소드를 호출해야 합니다. 다음 코드 단편은 태스크 명령에 대한 performExecute 메소드 예를 표시합니다.

```
public void performExecute() throws ECException {
 super.performExecute();

     // Include your business logic here.

     // Set output properties so the controller command
     // can retrieve the result from this task command.
}
```

런타임 프레임워크는 컨트롤러 명령의 getResources 메소드를 호출하여 명령이 액세스할 보호 가능 자원을 판별합니다. 이는 컨트롤러 명령의 범위에서 태스크 명령이 실행되고, 컨트롤러 명령의 getResources 메소드에 의해 리턴되지 않은 자원에 액세스하려고 시도하는 경우일 수 있습니다. 이런 경우, 태스크 명령이 getResources 메소드를 구현하여, 보호 가능 자원에 액세스 제어가 제공되는지 확인할 수 있습니다.

기본적으로 getResources는 태스크 명령에 대해 null을 리턴하고 자원 레벨 액세스 제어 확인이 수행되지 않음에 유의하십시오. 따라서 태스크 명령이 보호 가능 자원에 액세스하는 경우 이를 대체해야 합니다.

기존 명령의 사용자 정의

이 절에서는 기존 컨트롤러, 태스크 및 데이터 bean 명령을 사용자 정의할 수 있는 다양한 방법에 대해 설명합니다.

기존 컨트롤러 명령의 사용자 정의

컨트롤러 명령은 비즈니스 프로세스에 대한 비즈니스 로직을 캡슐화합니다. 비즈니스 프로세스 내 개별 작업 단위는 태스크 명령에 의해 수행됩니다. 따라서 컨트롤러 명령을 사용자 정의할 수 있는 여러 가지 방법이 있으며 몇몇 경우 태스크 명령의 사용자 정의가 포함됩니다.

컨트롤러 명령 사용자 정의 시 다음을 수행할 수 있습니다.

- 기존 컨트롤러 명령에 추가 처리 및 로직을 추가하십시오. 기존 비즈니스 로직 이전, 기존 로직 이후 또는 이전 및 이후 모두에 추가할 수 있습니다.
- 하나 이상의 태스크 명령을 바꾸십시오. 이를 통해 비즈니스 프로세스의 특정 단계를 수행하는 방법을 수정할 수 있습니다.

- 컨트롤러 명령으로 호출한 뷰를 바꾸십시오.

다음 절에서는 앞의 수정을 작성하는 방법에 대한 정보를 제공합니다.

컨트롤러 명령에 새 비즈니스 로직 추가

기존 WebSphere Commerce 컨트롤러 명령, ExistingControllerCmd가 있다고 가정하십시오. WebSphere Commerce 이름 지정 규칙에 따라, 이 컨트롤러 명령은 인터페이스 클래스 ExistingControllerCmd 및 구현 클래스 ExistingControllerCmdImpl을 갖게 됩니다. 이제 비즈니스 요구사항이 발생하고, 기존 명령에 새 비즈니스 로직을 추가해야 하는 것으로 가정하십시오. 로직의 한 부분은 기존 명령 로직 이전에 실행해야 하며 다른 부분은 기존 명령 로직 이후에 실행해야 합니다.

새 비즈니스 로직을 추가하는 첫 번째 단계는 원래 구현 클래스를 확장하는 새 구현 클래스를 작성하는 것입니다. 이 예의 경우, ExistingControllerCmdImpl 클래스를 확장하는 새 ModifiedControllerCmdImpl 클래스를 작성하게 됩니다. 새 구현 클래스는 원래 인터페이스(ExistingControllerCmd)를 구현해야 합니다.

새 구현 클래스에서 기존 명령의 performExecute를 대체할 새 performExecute 메소드를 작성해야 합니다. 새 performExecute 메소드의 경우, 새 비즈니스 로직을 삽입할 수 있는 두 가지 방법이 있습니다. 즉, 컨트롤러 명령에 코드를 직접 포함시키거나, 새 비즈니스 로직을 수행할 새 태스크 명령을 작성할 수 있습니다. 새 태스크 명령을 작성하는 경우, 컨트롤러 명령에서 새 태스크 명령 오브젝트의 인스턴스를 작성해야 합니다.

다음 코드 단편은 컨트롤러 명령에 직접 로직을 포함시켜 기존 컨트롤러 명령의 시작과 끝에 새 비즈니스 로직을 추가하는 방법을 보여줍니다.

```
public class ModifiedControllerCmdImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd
   {
      public void performExecute ()
         throws com.ibm.commerce.exception.ECException
         {

            /* Insert new business logic that must be
               executed before the original command.
            */

            // Execute the original command logic.
```

```
 super.performExecute();

            /* Insert new business logic that must be
               executed after the original command.
            */
          }
    }
```

다음 코드 단편은 컨트롤러 명령에서 새 태스크 명령의 인스턴스를 작성하여 기존 컨트롤러 명령의 시작에 새 비즈니스 로직을 추가하는 방법을 보여줍니다. 또한 새 태스크 명령 인터페이스 및 구현 클래스를 작성하고 명령 레지스트리에 태스크 명령을 등록하게 됩니다.

```
// Import the package with the CommandFactory
import com.ibm.commerce.command.*;

public class ModifiedControllerCmdImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd
   {

      public void performExecute ()
         throws com.ibm.commerce.exception.ECException
         {
     MyNewTaskCmd cmd = null;
     cmd = (MyNewTaskCmd) CommandFactory.createCommand(
               "com.mycompany.mycommands.MyNewTaskCommand",
               getStoreId());

            /*
            Set task command's input parameters, call its
            execute method and retrieve output
            parameters, as required.
            */

 super.performExecute();
         }
   }
```

컨트롤러 명령에 새 비즈니스 로직을 포함시키거나 로직을 수행할 태스크 명령을 작성하거나 그 여부에 관계 없이, WebSphere Commerce 명령 레지스트리의 CMDREG 테이블을 갱신하여 새 컨트롤러 명령 구현 클래스와 기존 컨트롤러 명령 인터페이스를 연관시켜야 합니다. 다음 SQL문은 갱신 예를 표시합니다.

```
update CMDREG
set CLASSNAME='ModifiedControllerCmdImpl'
where INTERFACENAME='ExistingControllerCmd'
```

컨트롤러 명령으로 호출한 태스크 명령 바꾸기

컨트롤러 명령은 개별 태스크를 수행하는 몇 가지 태스크 명령을 종종 호출합니다. 이 태스크들이 모여 컨트롤러 명령으로 표시되는 비즈니스 프로세스를 구성합니다. 컨트롤러 명령의 시작 또는 끝에 새 비즈니스 로직을 추가하지 않고, 프로세스 내 특정 단계를 수행하는 방법을 변경해야 할 수 있습니다. 이런 경우, 대체하려는 태스크 명령의 구현을 원하는 방법으로 태스크를 수행하는 새 태스크 명령의 구현으로 바꿔야 합니다.

WebSphere Commerce 프로그래밍 모델의 설계에 따라, 태스크 명령을 바꿀 새 컨트롤러 명령 구현 클래스를 작성하지 않아도 됩니다. 컨트롤러 명령은 명령 팩토리의 createCommand 메소드를 호출하여 태스크 명령의 인스턴스를 작성합니다. 명령 팩토리는 태스크 명령의 인터페이스 이름을 사용한 후 명령 레지스트리에 따라 올바른 구현 클래스를 판별합니다. 따라서 인스턴스를 작성할 태스크 명령을 바꾸려면 새 태스크 명령 구현 클래스를 작성한 후 원래 태스크 명령 인터페이스 이름이 새 태스크 명령 구현 클래스와 연관되도록 명령 레지스트리를 갱신해야 합니다. 추가 정보는 181 페이지의 『기존 태스크 명령 사용자 정의』를 참조하십시오.

컨트롤러 명령으로 호출한 뷰 바꾸기

컨트롤러 명령에 의해 호출되는 뷰를 바꾸려면 컨트롤러 명령에 대한 새 구현 클래스를 작성하십시오. 예를 들어, ExistingControllerCmdImpl을 확장하고 ExistingControllerCmdImpl 인터페이스를 구현하는 새 ModifiedControllerCmdImpl을 작성하십시오.

ModifiedControllerCmdImpl 클래스에서 performExecute 메소드를 대체하십시오. 새 performExecute 메소드의 경우, 모든 명령 처리가 발생하는지 확인하도록 super.performExecute를 호출하십시오. 명령 로직이 실행되면 응답 특성을 사용하여 호출된 뷰를 대체할 수 있습니다. 다음 코드 단편은 뷰를 경로 재지정으로 실행할 때 뷰를 대체하는 방법을 표시합니다.

```
// Import the packages containing TypedProperty, and ECConstants.
import com.ibm.commerce.datatype.*;
import com.ibm.commerce.server.*;

public class ModifiedControllerCmdImplImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd
   {
      public void performExecute ()
         throws com.ibm.commerce.exception.ECException
```

```
        {

           // Execute the original command logic.
 super.performExecute();

           // Create a new TypedProperty for response properties.
 TypedProperty rspProp = new TypedProperty();

          // set response properties
          rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");
          ////////////////////////////////////////////////////
         //  The following line is optional.  The VIEWREG   //
         //  table can specify the redirect URL.            //
         ////////////////////////////////////////////////////

         rspProp.put(ECConstants.EC_REDIRECTURL, MyURL);

 setResponseProperties( rspProp );

        }
   }
```

다음 코드 단편은 뷰를 전달 뷰로 실행할 때 뷰를 대체하는 방법을 표시합니다.

```
// Import the packages containing TypedProperty, and ECConstants.
import com.ibm.commerce.datatype.*;
import com.ibm.commerce.server.*;

public class ModifiedControllerCmdImplImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd
   {
      public void performExecute ()
         throws com.ibm.commerce.exception.ECException
         {

           // Execute the original command logic.
 super.performExecute();

           // Create a new TypedProperty for response properties.
 TypedProperty rspProp = new TypedProperty();

          // set response properties
          rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");

          ////////////////////////////////////////////////
         //  It is optional to explicitly set the name   //
         //  of the JSP template. The VIEWREG table can  //
         //  specify the JSP template.                   //
         ////////////////////////////////////////////////

         rspProp.put(ECConstants.EC_DOCPATHNAME, "MyJSP.jsp");
```

```
setResponseProperties( rspProp );

        }
    }
```

기존 컨트롤러 명령이 사용하는 뷰를 판별하려면 WebSphere Commerce Production 및 Development 온라인 도움말의 "참조"를 참조하십시오.

기존 태스크 명령 사용자 정의

기존 WebSphere Commerce 태스크 명령을 수정하는 데에는 두 가지 표준 방법이 있습니다. 이 수정 방법을 사용하면 다음을 수행할 수 있습니다.

- 기존 태스크 명령에 추가 처리 및 로직을 추가하십시오. 이는 기존 비즈니스 로직 이전, 기존 로직 이후 또는 이전 및 이후 모두에 추가할 수 있습니다.
- 기존 비즈니스 로직을 사용자 고유 비즈니스 로직으로 완전히 바꾸십시오.

위 수정을 수행하려면 실제로 새 태스크 명령 구현 클래스를 작성해야 합니다. 자세한 정보는 다음 절에서 제공합니다.

태스크 명령에 새 비즈니스 로직 추가

기존 WebSphere Commerce 태스크 명령, ExistingTaskCmd가 있다고 가정하십시오. WebSphere Commerce 이름 지정 규칙에 따라, 이 태스크 명령은 인터페이스 클래스 ExistingTaskCmd 및 구현 클래스 ExistingTaskCmdImpl을 갖게 됩니다. 이제 비즈니스 요구사항이 발생하고, 기존 명령에 새 비즈니스 로직을 추가해야 하는 것으로 가정하십시오. 로직의 한 부분은 기존 명령 로직 이전에 실행해야 하며 다른 부분은 기존 명령 로직 이후에 실행해야 합니다.

새 비즈니스 로직을 추가하는 첫 번째 단계는 원래 구현 클래스를 확장하는 새 구현 클래스를 작성하는 것입니다. 이 예의 경우, ExistingTaskCmdImpl 클래스를 확장하는 새 ModifiedTaskCmdImpl 클래스를 작성하게 됩니다. 새 구현 클래스는 원래 인터페이스(ExistingTaskCmd)를 구현해야 합니다.

새 명령에서 기존 performExecute 메소드를 대체하고, super.performExecute 메소드 호출 전후에 새 로직을 포함시키십시오.

다음의 코드 단편은 기존 태스크 명령에 새 비즈니스 로직을 추가하는 방법을 보여줍니다.

```
public class ModifiedTaskCmdImpl extends ExistingTaskCmdImpl
     implements ExistingTaskCmd {


         /* Insert new business logic that must be
            executed before the original command.
         */

         // Execute the original command logic.
 super.performExecute();

         /* Insert new business logic that must be
            executed after the original command.
         */
}
```

또한 새 구현 클래스와 기존 인터페이스가 연관되도록 CMDREG 테이블을 갱신해야 합니다. 다음 SQL문은 갱신 예를 표시합니다.

```
update CMDREG
set CLASSNAME='ModifiedTaskCmdImpl'
where INTERFACENAME='ExistingTaskCmd'
```

기존 태스크 명령의 비즈니스 로직 바꾸기

기존 태스크 명령의 비즈니스 로직을 바꾸려면 태스크 명령에 대한 새 구현 클래스를 작성해야 합니다. 이 새 구현 클래스는 기존 태스크 명령에서 확장되어야 하지만, 기존 인터페이스를 구현해서는 안 됩니다. 또한 새 구현 클래스에서 상위 클래스의 performExecute 메소드를 호출하지 마십시오.

바꿀 해당 명령에서 확장하는 것이 이해가 안 될 수도 있겠지만, 이러한 접근 방법을 사용하는 이유는 이후 WebSphere Commerce 버전을 지원하기 위한 것과 관련이 있습니다. 이 접근 방법은 이후 WebSphere Commerce 버전에서 일어날 수 있는 명령 인터페이스 변경으로부터 코드를 보호합니다.

예를 들어, OrderNotifyCmdImpl 태스크 명령의 비즈니스 로직을 바꾸려는 것으로 가정하십시오. 이런 경우 새 태스크 명령, CustomizedOrderNotifyCmdImpl을 작성하게 됩니다. 이 명령은 OrderNotifyCmdImpl을 확장합니다. 새 CustomizedOrderNotifyCmdImpl에서 새 비즈니스 로직을 작성하되 상위 클래스에서 performExecute 메소드를 호출하지 마십시오. 이후 WebSphere Commerce 버전이 인터페이스에 새 메소드, newMethod를 처음 사용하는 경우, 해당 버전의

OrderNotifyCmdImpl 명령은 newMethod 메소드의 기본 구현을 포함하게 됩니다. 그러면 새 명령이 OrderNotifyCmdImpl에서 확장되므로, 컴파일러가 OrderNotifyCmdImpl 명령에서 이 새 메소드의 기본 구현을 찾고 새 명령은 인터페이스 변경으로부터 보호받게 됩니다.

새 구현 클래스가 기존 클래스와 동일한 외부 행위를 제공하는지 확인하려면 WebSphere Commerce Production 및 Development 온라인 도움말의 "참조" 주제를 참조하십시오.

데이터 bean 사용자 정의

일반적으로 데이터 bean은 액세스 bean을 확장합니다. WebSphere Studio Application Developer로 작성할 수 있는 액세스 bean은 엔티티 bean의 정보에 액세스할 수 있는 간단한 방법을 제공합니다. 엔티티 bean을 수정하는 경우(예를 들어, 새 필드, 새 비즈니스 메소드 또는 새 파인더 추가), 액세스 bean을 다시 작성하는 즉시 액세스 bean에 갱신이 반영됩니다. 데이터 bean은 액세스 bean을 확장하므로, 새 속성을 자동으로 상속합니다. 이런 관계에 따라 데이터 bean이 엔티티 bean의 새 속성을 사용할 수 있도록 하기 위한 코딩이 필요하지 않습니다.

엔티티 bean에서 파생되지 않은 데이터 bean에 새 속성을 추가해야 하는 경우, Java 계승을 사용하여 기존 데이터 bean을 확장할예를 들어, OrderDataBean에 새 필드를 추가하려는 경우, 다음과 같이 MyOrderDataBean을 정의하십시오.

```
public class MyOrderDataBean extends OrderDataBean
{
     public String myNewField () {
     //  implement the new field here
     }

}
```

새 데이터 bean에는 반드시 BeanInfo 클래스가 있어야 합니다. 이 클래스 선언 견본은 다음과 같습니다.

```
public class MyOrderDataBeanInfo extends java.beans.SimpleBeanInfo
{

}
```

WebSphere Studio Application Developer는 이 BeanInfo 클래스를 작성할 수 있는 도구를 제공합니다.

제 7 장 거래 계약 및 비즈니스 정책(Business Edition)

이 장은 WebSphere Commerce Business Edition에만 적용됩니다.

소개

B2B(Business-to-Business) 상거래의 핵심 요소 중 하나는 관계 관리입니다. 구매자와 판매자 조직 사이의 비즈니스 관계를 관리하기 위해 거래 계약이 사용됩니다. WebSphere Commerce Business Edition에서 사용하는 거래 계약 모델은 장기 구매 계약 및 RFQ(Request For Quote) 등과 같은 다양한 거래 계약 유형을 지원합니다.

거래 계약의 기본 요소는 일련의 규정입니다. 각 규정은 거래 시 사용할 특정 비즈니스 규칙을 정의합니다. WebSphere Commerce Business Edition을 사용하는 경우, RFQ 온라인 프로세스를 사용하여 규정 세트를 조정하거나 오프라인으로 조정한 후 WebSphere Commerce 액셀러레이터의 비즈니스 관계 관리 인터페이스를 사용하여 캡처할 수 있습니다.

규정은 다음과 같은 몇 가지 방법으로 모델링할 수 있습니다.

- 사전 정의된 비즈니스 정책(예: 가격 목록 및 반품 정책) 중 하나를 선택하는 규정. 또는 사용자가 작성한 비즈니스 정책을 선택할 수 있습니다. 하나의 규정 오브젝트가 복수 비즈니스 정책 오브젝트를 참조할 수도 있습니다.
- 비즈니스 정책에 대한 특정 조정(예: 표준 가격 책정에 대한 조정)을 적용하는 규정
- 비즈니스 프로세스를 관리하는 매개변수 세트를 정의하는 규정. 예를 들어, 특정 장기 구매 계약에서 특정 서비스 센터를 사용하도록 지정할 수 있습니다.

장기 구매 계약은 규정 세트로 구성됩니다. 이는 다음 도표에 표시됩니다.

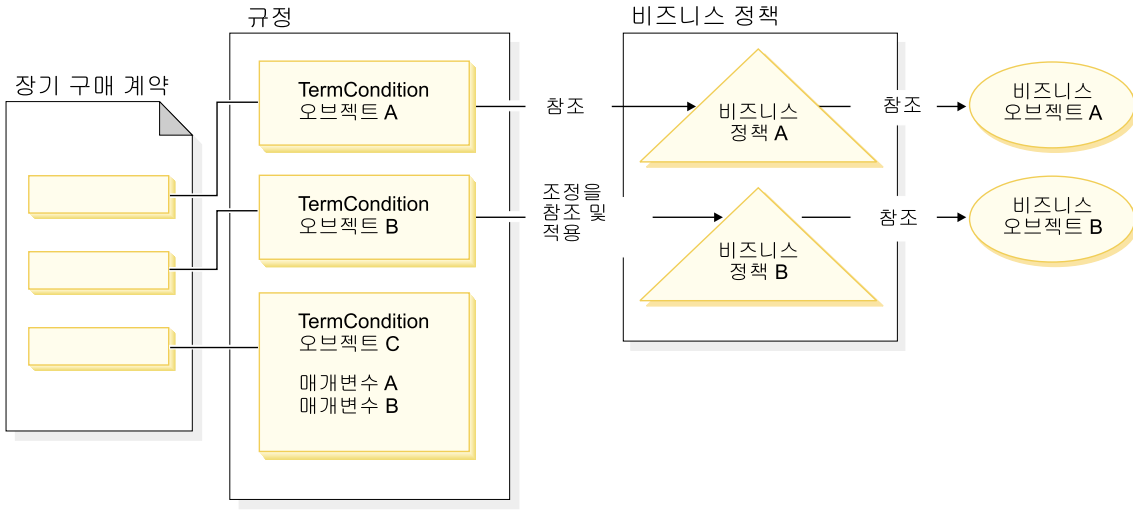


그림 29.

앞 도표에서 다음에 유의하십시오.

- "조정"은 비즈니스 정책의 수정을 의미합니다. 예를 들어, 표준 가격에 10%의 할인이 비즈니스 정책 결과에 적용되도록 사용될 수 있습니다. 매개변수 세트로 비즈니스 정책에 영향을 줄 수도 있습니다.
- 예를 들어, 도표에서 TermCondition 오브젝트 A는 운송 규정 오브젝트를 나타냅니다. 이 경우, 비즈니스 정책 A는 운송 방법 비즈니스 정책을 나타내고, 비즈니스 오브젝트 A는 운송 회사 XYZ의 "A3" 운송 방법을 나타냅니다.
- 다른 예로, 도표에서 TermCondition 오브젝트 B는 비즈니스 정책 B에 의해 정의되는 50% 가격 할인을 적용하는 가격 규정 오브젝트를 나타냅니다. 이 경우, 비즈니스 정책 B는 가격 정책이고 비즈니스 오브젝트 B는 마스터 카탈로그의 거래 포지션을 정의하는 거래 포지션 컨테이너입니다.

이 장에서는 프로그래머에게 새 비즈니스 정책 및 새 규정의 작성 방법에 대한 지침을 제공합니다.

ToolTech 견본 상점은 해당 비즈니스 플로우에서의 운송 규정 오브젝트 및 가격 규정 오브젝트에 대해 설명합니다. 이 예를 지원하는 장기 구매 계약 데이터에 대한 추가 정보는 188 페이지의 『ToolTech 견본 장기 구매 계약 데이터』를 참조하십시오.

비즈니스 정책 오브젝트 및 명령

비즈니스 정책 오브젝트에는 다음 정보가 들어 있습니다.

- 정책 ID
  비즈니스 정책 오브젝트의 1차 키입니다.
- 정책 유형
  비즈니스 정책 유형을 정의합니다. `Price` 및 `ProductSet`가 정책 유형의 예입니다.
- 정책 이름
  각 비즈니스 정책의 이름은 고유해야 합니다.
- 상점 엔티티
  비즈니스 정책이 전개되는 상점 또는 상점 그룹입니다.
- 특성
  비즈니스 정책 명령으로 전달될 수 있는 기본 특성 세트입니다. 비즈니스 정책 오브젝트와 연관된 명령은 BusinessPolicyCmd 테이블에 저장됩니다.
- 유효 기간
  비즈니스 정책 오브젝트가 유효한 기간입니다.
- 비즈니스 정책 명령
  비즈니스 정책을 구현하는 비즈니스 정책 명령(있을 수도 있고 없을 수도 있음)입니다. 비즈니스 정책 명령은 일반적으로 태스크 명령이나 컨트롤러 명령일 수 있는 비즈니스 프로세스에 의해 호출됩니다. 예를 들어, `getContractPrice()` 명령은 가격 규정을 가져옵니다. 이 가격 규정은 특정 가격 정책 명령을 의미하며 이 가격 정책 명령은 가격을 계산하는 데 사용됩니다.

복수 비즈니스 정책 명령이 단일 비즈니스 정책 오브젝트와 연관될 수 있습니다. 각 비즈니스 정책 명령은 비즈니스 정책 유형 오브젝트에서 정의되는 동일한 인터페이스를 구현해야 합니다. 새 비즈니스 정책 명령의 구조는 다음 도표에 나타나 있습니다.

새 비즈니스 정책 명령

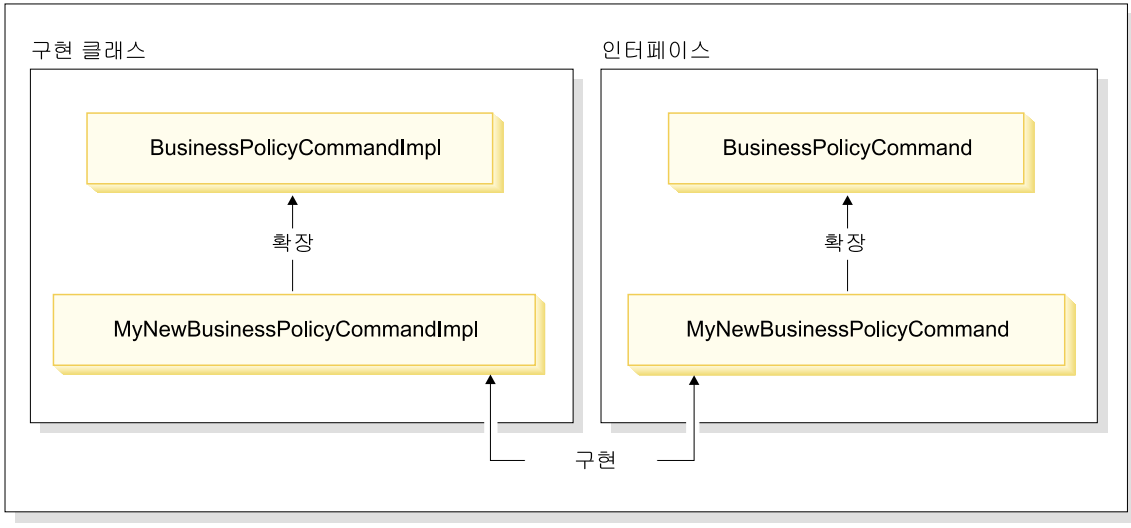


그림 30.

앞 도표에 표시되어 있는 것처럼 새 비즈니스 정책 명령을 작성하려면 WebSphere Commerce BusinessPolicyCmdImpl 구현 클래스를 확장하는 새 구현 클래스를 작성하십시오. 또한 BusinessPolicyCmd 인터페이스를 확장하는 새 인터페이스를 작성하십시오.

ToolTech 견본 장기 구매 계약 데이터

이 절에서는 ToolTech 견본 상점에서 사용하는 몇몇 장기 구매 계약 데이터에 대해 설명합니다.

다음 절의 견본 데이터는 데이터베이스 테이블로 구성됩니다. 관련 행 및 열만 표시됩니다. 또한 견본 설치 시, 고유 식별자(예: CONTRACT_ID)의 값이 여기 표시된 값과 다를 수 있으므로 유의하십시오.

CONTRACT 테이블 견본 데이터

다음 테이블은 CONTRACT 데이터베이스 테이블의 관련 견본 데이터를 표시합니다. 데이터베이스 열 이름은 첫 번째 열에 표시되고 테이블의 견본 데이터 행은 두 번째 열에 표시됩니다.

열 이름	견본 데이터
CONTRACT_ID	10007
MAJORVERSION	1
MINORVERSION	0
NAME	ToolTechContractNumber 4567
MEMBER_ID	-2001
ORIGIN	0
STATE	3
USAGE	1
MARKFORDELETE	0

TERMCOND 테이블 견본 데이터

다음 테이블은 TERMCOND 데이터베이스 테이블의 관련 견본 데이터를 표시합니다. 데이터베이스 열 이름은 첫 번째 열에 표시되고 테이블의 견본 데이터 행은 두 번째 및 세 번째 열에 표시됩니다.

열 이름	견본 데이터 행 1	견본 데이터 행 2
TERMCOND_ID	10025	10030
TCSUBTYPE_ID	PriceTCPriceListWith SelectiveAdjustment	ShippingTCShippingMode
TRADING_ID	10007	10007
STRINGFILED1	ProductSet2	
INTEGERFIELD2	10002	
INTEGERFIELD3	1	
BIGINTFIELD1	10051	
FLOATFIELD1	-50.0	
SEQUENCE	1	6

POLICYTC 테이블 견본 데이터

다음 테이블은 POLICYTC 데이터베이스 테이블의 관련 견본 데이터를 표시합니다. 이 테이블은 정책 오브젝트와 규정 오브젝트 간의 관계를 설정합니다.

	열 이름	
	POLICY_ID	**TERMCOND_ID**
견본 데이터 행 1	10053	10025
견본 데이터 행 2	10056	10030

POLICY 테이블 견본 데이터

다음 테이블은 POLICY 데이터베이스 테이블의 관련 견본 데이터를 표시합니다.

열 이름	견본 데이터 행 1	견본 데이터 행 2
POLICY_ID	10053	10056
POLICYNAME	MasterCatalogPriceList	A3
POLICYTYPE_ID	Price	ShippingMode
STOREENT_ID	10051	10051
PROPERTIES	name=ToolTech& member_id=-2001	shippingMode=A3
STARTTIME	널(NULL)값	널(NULL)값
ENDTIME	널(NULL)값	널(NULL)값

TRADEPOSCN 테이블 견본 데이터

다음 테이블은 TRADEPOSCN 데이터베이스 테이블의 관련 견본 데이터를 표시합니다.

	열 이름			
	TRADEEPOSCN_ID	**MEMBER_ID**	**NAME**	**TYPE**
견본 데이터 행	10051	-2001	ToolTech	S

SHIPMODE 테이블 견본 데이터

다음 테이블은 SHIPMODE 데이터베이스 테이블의 관련 견본 데이터를 표시합니다.

	열 이름			
	SHIPMODE_ID	**STOREENTITY_ID**	**CODE**	**CARRIER**
견본 데이터 행	10053	10051	A3	XYZ 운송 회사

기존 장기 구매 계약 모델 확장

장기 구매 계약은 하나 이상의 규정 오브젝트로 구성될 수 있으며 각 오브젝트는 정책을 참조합니다. 이 절에서는 새 비즈니스 정책을 작성하고 이 정책을 비즈니스 플로우에 통합하는 데 필요한 단계에 대해 설명합니다.

다음은 이 태스크를 수행하기 위한 상위 레벨 단계의 간단한 개요입니다.

1. 새 비즈니스 정책 작성
   다음 태스크는 새 비즈니스 정책 명령의 작성과 관련이 있습니다.
   a. 새 비즈니스 정책 유형 작성(필요한 경우)
      여러 비즈니스 정책 유형이 제공되지만, 표준 유형이 사용자의 비즈니스 요구사항에 적합하지 않은 경우 새 비즈니스 정책 유형을 작성하십시오.
   b. 새 비즈니스 정책 명령 작성
   c. 새 비즈니스 정책 및 비즈니스 정책 명령 등록
2. 규정 오브젝트와 새 비즈니스 정책의 관계 설정
   이는 기존 규정 오브젝트와 새 비즈니스 정책의 관계를 설정하거나 새 규정 오브젝트를 작성하여 수행할 수 있습니다. 새 규정 오브젝트를 작성하는 경우, 다음 단계를 수행해야 합니다.
   a. 데이터베이스에 새 규정 등록
   b. 장기 구매 계약 XSD(XML Schema Definition)에 새 규정 등록
   c. 규정에 대한 새 CMP 엔터프라이즈 bean 작성
   d. 새 규정을 반영하도록 WebSphere Commerce 액셀러레이터 갱신
3. 비즈니스 플로우에서 새 비즈니스 정책 호출

WebSphere Commerce 버전 5.5는 장기 구매 계약의 새로운 유형을 소개합니다. 사용 가능한 장기 구매 계약 유형에 대한 정보는 WebSphere Commerce Production 온라인 도움말에 있는 “개념”에서 “비즈니스 계정”을 참조하십시오.

다음 절에서는 확장자 예에서의 장기 구매 계약 유형으로 BuyerContract를 사용합니다. 비슷한 확장자 방법이 기타 장기 구매 계약 유형에 사용됩니다.

새 비즈니스 정책 작성

새 비즈니스 정책의 작성에는 일반적으로 새 비즈니스 정책 명령 작성 및 데이터베이스에 고유한 비즈니스 정책 등록이 포함됩니다.

새 비즈니스 정책 명령의 작성에는 다음 상위 레벨이 포함됩니다.

1. 새 비즈니스 정책 유형 작성(필요한 경우)
2. 새 비즈니스 정책 명령 작성
3. 데이터베이스에 새 비즈니스 정책 및 비즈니스 정책 명령 등록

앞 단계 각각에 대해서는 후속 절에서 자세히 설명합니다.

새 비즈니스 정책 유형 작성

이 절에서는 새 비즈니스 정책 유형의 작성 방법에 대해 설명합니다. 비즈니스 정책 유형은 정책을 적용하는 트랜잭션의 범주를 나타냅니다. 비즈니스 정책 유형의 예는 다음과 같습니다.

- Price
- ProductSet
- ShippingMode
- ShippingCharge
- Payment
- ReturnCharge
- ReturnApproval
- ReturnPayment
- InvoiceFormat

기존 비즈니스 정책 유형이 사용자의 비즈니스 요구사항을 충족시키지 못하는 경우, 새 비즈니스 정책 유형을 작성해야 합니다. 새 비즈니스 정책 유형의 작성은 비즈니스 정책 유형의 정의 및 등록으로 구성됩니다.

새 정책 유형의 정의 및 등록 시, 다음 데이터베이스 테이블을 갱신해야 합니다.

- POLICYTYPE
- PLCYTYCMIF
- PLCYTYPDSC

POLICYTYPE 테이블은 작성 중인 비즈니스 정책의 유형을 지정합니다. 이 테이블에는 1차 키인 단일 열 POLICYTYPE_ID가 있습니다. 값 예는 Price입니다. 새 비즈니스 정책 유형을 작성하는 경우, 고유 POLICYTYPE_ID를 지정하는지 확인하십시오.

PLCYTYCMIF 테이블은 비즈니스 정책 유형 대 명령 인터페이스 관계 스펙 테이블입니다. 즉, 각 비즈니스 정책 유형에 대해 비즈니스 정책 오브젝트의 Java 명령 인터페이스를 지정합니다. 비즈니스 정책을 구현하는 비즈니스 정책 명령은 있을 수도 없을 수도 있지만, 각 비즈니스 정책 명령은 여기서, 지정된 인터페이스를 구현해야 합니다.

PLCYTYPDSC 테이블은 비즈니스 정책 유형에 대한 설명을 지정합니다. 이 테이블에는 설명의 언어 식별자 및 비즈니스 정책 유형에 대한 설명이 포함됩니다.

새 비즈니스 정책 유형을 작성하려면 이들 테이블 각각에 새 비즈니스 정책 유형에 대한 항목을 작성하십시오. 다음 SQL문을 예로 들 수 있습니다.

```
insert into POLICYTYPE (POLICYTYPE_ID) values ('MyNewPolicyType');
insert into PLCYTYCMIF (POLICYTYPE_ID, BUSINESSCMDIF)
   values ('MyNewPolicyType',
   'com.mycompany.mybusinesspolicycommands.MyNewPolicy');
insert into  PLCYTYPDSC (POLICYTYPE_ID, LANGUAGE_ID, DESCRIPTION)
   values ('MyNewPolicyType', -1,
   'My new policy type for example purposes.');
```

새 비즈니스 정책 유형의 작성을 위한 마지막 단계로서 하나 이상의 새 비즈니스 정책 유형 인터페이스를 코딩할 수 있습니다. 그러면 이 인터페이스는 이 비즈니스 정책 유형의 범주에 해당하는 비즈니스 정책 명령에 의해 구현됩니다. 예를 들어, ToolTech 견본 상점의 경우, Price가 비즈니스 정책 유형으로 정의됩니다. 즉, 모든 가격 관련 비즈니스 정책 명령에 의해 구현되는 com.ibm.commerce.price.commands.ResolvePriceListsCmd 및 com.ibm.commerce.price.commands.RetrievePricesCmd 인터페이스가 있습니다.

새 비즈니스 정책 유형에 대한 조작을 수행하는 비즈니스 정책 명령이 없는 경우, 새 인터페이스를 작성하지 않아도 됩니다. 이는 드문 경우이며 대부분의 경우 새 비즈니스 정책 유형 작성 시 새 비즈니스 정책 유형 인터페이스 또한 작성해야 합니다.

비즈니스 정책 유형 인터페이스를 작성하는 경우, 새 인터페이스가 com.ibm.commerce.command.BusinessPolicyCommand 인터페이스를 확장해야 합니다.

새 비즈니스 정책 명령 작성

새 비즈니스 정책 명령을 작성하려면 명령과 관련이 있는 비즈니스 정책 유형의 인터페이스를 구현하는 새 명령을 작성해야 합니다. 새 명령은 또한 com.ibm.commerce.command.BusinessPolicyCommandImpl 구현 클래스를 확장해야 합니다. 이는 새 컨트롤러 또는 태스크 명령을 작성하는 것과 매우 유사합니다.

비즈니스 정책 명령으로 입력 특성을 전달할 수 있는 두 가지 다른 접근 방법이 있습니다. 첫 번째 방법은 POLICY 테이블의 PROPERTIES 열에 기본 입력 특성을 지정하는 것입니다. 이 테이블에 대한 추가 정보는 다음 절을 참조하십시오.

두 번째 접근 방법은 각 입력 특성에 대한 새 필드를 명령에 작성하는 것입니다. 각 필드에 대해, 새 getter 및 setter 메소드 쌍을 작성하십시오.

비즈니스 정책 명령에 requestProperties 설정

비즈니스 정책 명령 오브젝트에 requestProperties를 설정하기 위한 두 가지 방법이 있습니다. 첫 번째 방법은 POLICY 테이블의 PROPERTIES 열을 사용하여 기본 특성을 설정하는 것입니다. 이 방법은 setRequestProperties 메소드에 의해 수행됩니다. 특성을 설정하는 두 번째 방법은 비즈니스 정책 명령을 호출하는 명령(컨트롤러 또는 태스크)으로 기타 필수 특성을 명시적으로 설정하는 것입니다.

새 비즈니스 정책 명령 작성 시, 로직을 포함할 기본 setRequestProperties 메소드를 대체하여 requestProperties 오브젝트에 포함되어 있는 각 매개변수를 명시적으로 설정해야 합니다.

인터페이스 이름이 MyNewBusinessPolicyCmd이고 구현 클래스 이름이 MyNewBusinessPolicyCmdImpl인 새 비즈니스 정책 명령의 예를 고려하십시오.

이 비즈니스 정책 명령에 대한 POLICY 테이블의 항목이 PROPERTIES 열의 다음 값을 포함하는 것으로 가정하십시오.

- defaultProperty1=apple
- defaultProperty2=orange
- defaultProperty3=banana

이 새 비즈니스 정책 명령의 인터페이스는 다음과 같이 정의됩니다.

```
public interface MyNewBusinessPolicyCmd extends
   com.ibm.commerce.command.BusinessPolicyCmd {
      java.lang.String defaultCommandClassName =
         'com.mycompany.mycommands.MyNewBusinessPolicyCmdImpl';
      public void setProperty1();
      public void setProperty2();
}
```

이 새 비즈니스 정책 명령의 구현 클래스는 다음과 같이 정의됩니다.

```
public class MyNewBusinessPolicyCmdImpl extends
   com.ibm.commerce.command.BusinessPolicyCmdImpl
   implements com.mycompany.mycommands.MyNewBusinessPolicyCmd {
      // Establish default properties that are stored in the POLICY table

      private java.lang.String defaultProperty1;
      private java.lang.String defaultProperty2;
      private java.lang.String defaultProperty3;

      // Begin to establish properties that must be set
     // by the calling command.

      //  *** property1   ***
      private java.lang.String property1;
      public java.lang.String getProperty1() {
         return property1;
         }
      public void setProperty1(java.lang.String newProperty1) {
         property1 = newProperty1;
      }

      //  *** property2   ***
      private java.lang.String property2;
      public java.lang.String getProperty2() {
         return property2;
         }
      public void setProperty1(java.lang.String newProperty2) {
         property2 = newProperty2;
         }

      // End establishing properties that must be set
```

```
      // by the calling command.

      /* Upon instantiation the business policy command sets all
         default properties from the POLICY table into the
         requestProperties object.  The calling command
         is responsible for setting any other required properties.
      */

      public void setRequestProperties(com.ibm.commerce.datatype.TypedProperty
         requestProperties) {
         //  Get the default properties defined in the POLICY table
         setDefaultProperty1(requestProperties.get("defaultProperty1"));
         setDefaultProperty2(requestProperties.get("defaultProperty2"));
         setDefaultProperty3(requestProperties.get("defaultProperty3"));
         }

}
```

새 비즈니스 정책 명령을 호출하는 명령은 다음과 유사한 방법으로 정의될 수 있습니다.

```
public class MyCallerCommandImpl
   extends com.ibm.commerce.command.TaskCommandImpl
   implements com.mycompany.mycommands.MyCallerCommand {

   /* Include all elements and processing required for the
      task command.
   */

   // Determine the policy ID and setPolicyId

   // Call the business policy command.

   cmd = (MyNewBusinessPolicyCmd) CommandFactory
            createPolicyCommand(policyId);

   // Set required properties

   cmd.setProperty1("Fruit salad");
   cmd.setProperty2("Favorite food");

   cmd.execute();
}
```

새 비즈니스 정책 및 비즈니스 정책 명령 등록

새 비즈니스 정책 명령을 작성한 후 데이터베이스에 비즈니스 정책 및 비즈니스 정책 명령을 모두 등록해야 합니다.

비즈니스 정책은 POLICY 테이블에 등록됩니다. 이 테이블에는 다음과 같은 열이 있습니다.

- POLICY_ID
  1차 키. 정책 식별자입니다.
- POLICYNAME
  고유 정책 이름.
- POLICYTYPE_ID
  정책 유형 식별자. POLICYTYPE 테이블의 foreign key입니다.
- STOREENT_ID
  정책이 적용되는 상점 또는 상점 그룹.
- PROPERTIES
  비즈니스 정책 명령으로 설정될 수 있는 기본 특성. 이름 값 쌍으로 지정됩니다 (예: `parm1=val1&parm2=val2`).
- STARTDATE
  정책의 시작 날짜(시간소인으로 지정). 널(Null)값인 경우, 시작 날짜는 즉시입니다.
- ENDDATE
  정책의 종료 날짜(시간소인으로 지정). 널(Null)값인 경우, 종료 날짜가 없습니다.

POLICY 테이블에 새 정책이 등록되면 비즈니스 정책을 구현하는 비즈니스 정책 명령과 정책 사이의 관계를 등록해야 합니다. POLICYCMD 테이블이 이 용도로 사용됩니다. POLICYCMD 테이블에는 다음 열이 있습니다.

- POLICY_ID
  POLICY 테이블에 대한 foreign key 참조.
- BUSINESSCMDCLASS
  정책을 구현하는 비즈니스 정책 명령.
- PROPERTIES
  비즈니스 정책 명령으로 설정될 수 있는 기본 특성. 이름 값 쌍으로 지정됩니다 (예: `parm1=val1&parm2=val2`).

규정 오브젝트와 새 비즈니스 정책의 관계 설명

WebSphere Commerce 장기 구매 계약 및 정책 프레임워크에서 규정(*조건*이라고도 함)은 구매자와 판매자 간의 계약을 설명하는 한 방법을 제공합니다. 규정은 다양한 거래 계약 유형(예: 장기 구매 계약 및 RFQ(Request For Quotation))으로 사용될 수 있습니다. 규정 오브젝트는 일반적으로 선택 조정이 가능한 비즈니스 정책입니다. 예를 들어, 가격 규정 오브젝트는 가격 정책 오브젝트 중 하나를 선택하여 작성할 수 있습니다. 가격 조건의 경우, 계정 관리자는 다음과 같이 상점 표준 가격을 조정할 수 있습니다.

- 표준 가격 목록에 대한 할인률
- 지정 상품 세트에 대한 할인률

각 조정은 규정으로 지정됩니다.

새 비즈니스 정책을 작성하고 이 정책을 장기 구매 계약에서 사용하려는 경우, 이 비즈니스 정책을 참조하는 최소 하나의 규정 오브젝트가 있어야 합니다. 기존 규정 오브젝트를 새 비즈니스 정책과 관련짓거나(이는 기존 규정 오브젝트와 새 비즈니스 정책 간의 관계를 XSD(XML 스키마 정의) 파일에서 캡처하여 수행), 새 비즈니스 정책과 관련되는 새 규정 오브젝트를 작성할 수 있습니다.

새 규정 작성

WebSphere Commerce 아키텍처의 경우, 새 규정 오브젝트는 다음 단계를 수행하여 작성됩니다.

1. 새 규정을 포함하도록 데이터베이스 스키마 갱신
2. 새 규정을 반영하도록 XSD 파일 갱신
3. 규정에 대한 새 엔터프라이즈 bean 작성
4. 새 규정을 반영하도록 WebSphere Commerce 액셀러레이터 갱신 또는 contract load 명령을 사용하여 새 규정을 사용하는 새 장기 구매 계약 작성

다음 절에서 `MyTC` 예는 새 규정 오브젝트입니다.

데이터베이스에 새 규정 등록

새 규정 오브젝트를 작성하는 경우, 이 오브젝트를 포함하도록 데이터베이스 스키마를 갱신해야 합니다. 갱신해야 하는 데이터베이스 테이블은 TCTYPE 및 TCSUBTYPE입니다.

다음 SQL문은 데이터베이스에 새 규정을 등록하는 방법에 대한 한 예입니다.

```
insert into TCTYPE (TCTYPE_ID) values ('MyTC');
insert into TCSUBTYPE (TCSUBTYPE_ID, TCTYPE_ID, ACCESSBEANNAME,
   DEPLOYCOMMAND)
values ('MySubTC', 'MyTC',
   'com.ibm.commerce.contract.objects.MySubTCAccessBean',
   'packagename.MySubTCDeployCmd');
```

장기 구매 계약 XSD에 새 규정 등록

장기 구매 계약에 새 규정을 사용할 수 있도록 하려면 새 규정을 정의하는 새 XSD 파일을 작성해야 합니다. 또한 새 XSD 파일을 포함하도록 Package.xsd 파일을 갱신해야 합니다.

새 XSD 파일을 작성하려면 다음을 수행하십시오.

1. 다음 디렉토리로 이동하십시오.
   - Windows *WC_installdir*\xml\trading\xsd
   - AIX Solaris Linux *WC_installdir*/xml/trading/xsd
   - 400 *WC_userdir*/instances/*instanceName*/xml/trading/xsd

   여기서, *instanceName*은 WebSphere Commerce 인스턴스 이름입니다.
2. 이 디렉토리에 새 XSD 파일을 작성하십시오. 다음은 MyTC 예에 사용할 XSD를 표시합니다. 파일은 CustomizedBuyerContract.xsd입니다.

```
<?xml version="1.0"?>
<schema targetNamespace="http://www.ibm.com/WebSphereCommerce"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:wc="http://www.ibm.com/WebSphereCommerce"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">

  <!-- include basic trading agreement xsd -->

  <include schemaLocation="BuyerContract.xsd" />
  <complexType name="MyTCType">
            <complexContent>
            <extension base="wc:TermConditionType"/>
            </complexContent>
        </complexType>
    <element name="MySubTC" substitutionGroup="wc:AbstractCustomizedTC">
        <complexType>
            <complexContent>
```

```
                <extension base="wc:MyTCType">
                    <sequence>
                        <element ref="wc:ProductSetPolicyRef"/>
                    </sequence>
                    <attribute name="attr1" type="normalizedString"
                         use="required"/>
                    <attribute name="attr2" type="int" use="required"/>
                </extension>
            </complexContent>
        </complexType>
    </element>
</schema>
```

3. 새 파일을 저장하십시오.

다음으로 BuyerContract.xsd를 제거하고 대신 CustomizedBuyerContract.xsd를 포함하도록 다음과 같이 Package.xsd를 갱신해야 합니다.

1. 다음 디렉토리로 이동하십시오.
   - Windows *WC_installdir*\xml\trading\xsd
   - AIX Solaris Linux *WC_installdir*/xml/trading/xsd
   - 400 *WC_userdir*/instances/*instanceName*/xml/trading/xsd

   여기서, *instanceName*은 WebSphere Commerce 인스턴스 이름입니다.
2. 텍스트 편집기에서 Package.xsd 파일을 여십시오.
3. BuyerContract.xsd에 대한 섹션을 찾아 다음과 같이 수정하십시오.

```
<!--include schemaLocation="BuyerContract.xsd"/-->
<include schemaLocation="CustomizedBuyerContract.xsd"/>
```

규정에 대한 새 CMP 엔터프라이즈 bean 작성

규정 오브젝트에 대한 새 CMP 엔터프라이즈 bean을 작성해야 합니다. 규정 하위 유형에 대한 bean이 작성됩니다.

일반적으로 새 엔터프라이즈 bean을 작성할 때는 WebSphere Commerce 엔티티 bean이 들어 있는 EJB 그룹 중 하나가 아닌 WebSphereCommerceServerExtensionsData 프로젝트에 bean을 배치함에 유의하십시오. 그러나 이런 경우 규정에 대한 모든 새 엔티티 bean은 WebSphere Commerce TermCondition bean에서 상속되어야 하므로, 새 규정 bean은 Enablement-RelationshipManagementData 프로젝트에 배치해야 합니다. 다음 절에서는 WebSphere Studio Application Developer의 도구를 사용하여 새 엔터프라이즈 bean을 작성하는 방법에 대해 설명합니다.

새 엔터프라이즈 bean 작성: 새 규정에 대한 새 CMP 엔터프라이즈 bean을 작성하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼치십시오.
2. **Enablement-RelationshipManagementData** 모듈을 마우스 오른쪽 버튼으로 누르고 **새로 만들기 > (기타 >) 엔터프라이즈 Bean**을 선택하십시오.
   엔터프라이즈 bean 작성 마법사가 열립니다.
3. **EJB 프로젝트** 드롭 다운 목록에 **Enablement-RelationshipManagementData**가 이미 선택되어 있습니다. **다음**을 누르십시오.
4. 엔터프라이즈 bean 작성 창에서 다음을 수행하십시오.
   a. **CMP(container-managed persistence) 필드가 있는 엔티티 bean**을 선택하십시오.
   b. **bean 이름** 필드에 적절한 bean 이름을 입력하십시오. 이 예의 경우 `MySubTC`를 입력하십시오.
   c. **소스 폴더** 필드에서는 지정된 기본값(ejbModule)을 그대로 두십시오.
   d. **기본 패키지** 필드에 `com.ibm.commerce.contract.objects`를 입력하십시오.
   e. **다음**을 누르십시오.
5. 엔터프라이즈 bean 정보 창에서 다음을 수행하십시오.
   a. **bean supertype** 드롭 다운 목록에서 **TermCondition**을 선택하십시오.
   b. **추가**를 눌러 새 CMP 속성을 추가하십시오.
      CMP 속성 작성 창이 열립니다. 이 창에서 다음을 수행하십시오.
      1) **이름** 필드에 적절한 새 CMP 필드 이름을 입력하십시오. 이 예의 경우 `attr1`을 입력하십시오.
      2) **유형** 필드에, 필드에 적절한 데이터 유형을 입력하십시오. 이 예의 경우 `String`을 입력하십시오.
      3) **getter 및 setter 메소드로 액세스** 선택란을 선택하십시오.
      4) **원격 인터페이스로 getter 및 setter 메소드 승격** 선택란을 선택하십시오.
      5) **getter를 읽기 전용으로 만들기** 선택란을 지우십시오.

6) **적용**을 누르십시오.

7) 추가 속성을 작성하십시오. **이름** 필드에 attr2를 입력하십시오.

8) **유형** 필드에, 필드에 적절한 데이터 유형을 입력하십시오. 이 예의 경우 Integer를 입력하십시오.

9) **getter를 읽기 전용으로 만들기** 선택란을 지우십시오.

10) **적용**을 누르십시오.

11) **닫기**를 눌러 창을 닫으십시오.

6. **완료**를 누릅니다.

새 bean의 필드를 TERMCOND 테이블로 맵핑: 다음 단계는 새 bean의 필드를 TERMCOND 테이블의 열로 맵핑하는 것입니다. 이 맵핑을 작성하려면 다음을 수행하십시오.

1. J2EE 네비게이터 보기로 전환하십시오.
2. **Enablement-RelationshipManagementData > ejbModule > META-INF** 폴더를 펼치십시오.
3. **Map.mapxmi** 파일을 두 번 누르십시오.
4. 엔터프라이즈 bean 분할창에서 **TermCondition** bean을 펼친 후 **MySubTC** bean을 펼쳐 해당 속성을 볼 수 있도록 하십시오.
5. 테이블 분할창에서 해당 열을 볼 수 있도록 **TERMCOND** 테이블을 펼치십시오.
6. MySubTC에서 TERMCOND 테이블의 **STRINGFIELD3** 열로 **attr1** 필드를 끄십시오.
7. MySubTC 에서 TERMCOND 테이블의 **INTEGERFIELD3** 열로 **attr2** 필드를 끄십시오.
8. 변경사항을 저장하십시오.

새 ejbCreate 메소드 추가: 이 단계에서는 다음을 수행하여 새 ejbCreate 메소드를 MySubTC bean에 추가합니다.

1. J2EE 계층 보기에서 **MySubTCBean** 클래스를 두 번 눌러 열고 해당 소스 코드를 보십시오.

2. 다음 코드를 클래스에 추가하여 새 ejbCreate(Long, Element) 메소드를 작성하십시오.

```
public com.ibm.commerce.contract.objects.TermConditionKey
   ejbCreate(java.lang.Long argTradingId,
   org.w3c.dom.Element argElement)
      throws javax.ejb.CreateException,
     javax.ejb.FinderException,
     javax.naming.NamingException,
     javax.ejb.RemoveException {
   _initLinks();
          super.ejbCreate (argTradingId, argElement);
           this.attr1= null;
           this.attr2 = null;
return null;
}
```

코드 변경사항을 저장하십시오.

3. 새 ejbCreate(Long, Element) 메소드를 홈 인터페이스에 추가해야 합니다. 그러면 메소드가 작성된 액세스 bean에서 사용할 수 있게 됩니다. 홈 인터페이스에 메소드를 추가하려면 다음을 수행하십시오.
   a. 아웃라인 보기의 **ejbCreate(Long, Element)** 메소드를 마우스 오른쪽 버튼으로 누르고 **엔터프라이즈 bean > 홈 인터페이스로 승격**을 선택하십시오.

새 ejbPostCreate 메소드 추가: 이제 다음을 수행하여 ejbCreate(Long, Element) 메소드와 동일한 매개변수를 갖는 새 ejbPostCreate(Long, Element) 메소드를 작성하십시오.

1. **MySubTCBean** 클래스를 두 번 눌러 열고 해당 소스 코드를 보십시오.
2. 다음 코드를 클래스에 추가하여 새 ejbPostCreate(Long, Element) 메소드를 작성하십시오.

```
public void ejbPostCreate(java.lang.Long argTradingId,
   org.w3c.dom.Element argElement)
      throws javax.ejb.CreateException,
     javax.ejb.FinderException,
     javax.naming.NamingException,
     javax.ejb.RemoveException
        {
           parseXMLElement(argElement);
        }
```

코드 변경사항을 저장하십시오.

parseXMLElement 메소드 추가: 이 단계에서는 다음과 같이 MySubTCBean에 parseXMLElement 메소드를 작성해야 합니다.

1. **MySubTCBean** 클래스를 두 번 눌러 열고 해당 소스 코드를 보십시오.
2. 다음과 같이 메소드를 갱신하십시오.

```
public void parseXMLElement(org.w3c.dom.Element argElement) throws
    javax.ejb.CreateException,
    javax.ejb.FinderException,
    javax.naming.NamingException,
    javax.ejb.RemoveException
{
 super.parseXMLElement(argElement);
 if (argElement == null)
  return;
 String nodeName = argElement.getNodeName();
 if (nodeName.equals("TCCopy"))
  return;

 this.attr1 = argElement .getAttribute("attr1").trim();
 this.attr2 = new Integer (argElement.
     getAttribute("attr2").trim());
 // get element "ProductSetPolicyRef" from "MySubTC"
 Element ePolicyReference = null;
 ePolicyReference = ContractUtil.getElementByTag(
      argElement, "ProductSetPolicyRef");

 parseElementPolicyReference(ePolicyReference);

}
```

3. 작업을 저장하십시오.

createNewVersion 메소드 추가: 이 단계에서는 다음과 같이 MySubTCBean에 새 createNewVersion 메소드를 작성해야 합니다.

1. **MySubTCBean** 클래스를 두 번 눌러 열고 해당 소스 코드를 보십시오.
2. 다음과 같이 메소드를 갱신하십시오.

```
public Long createNewVersion(Long argNewTradingId) throws
  javax.ejb.CreateException,
  javax.ejb.FinderException,
  javax.naming.NamingException,
  javax.ejb.RemoveException,
```

```
  org.xml.sax.SAXException,
  java.io.IOException
{
   // Contract a seqElement since tcSequence can not be null
   Element seqElement = ContractUtil.
      getSeqElementFromTCSequence(this.tcSequence);
   MySubTCAccessBean newTC = new MySubTCAccessBean(
      argNewTradingId, seqElement);
   Long newTCId = newTC.getReferenceNumberInEJBType();
   newTC.setInitKey_referenceNumber(newTCId);
   newTC.setMandatoryFlag(this.mandatoryFlag);
   newTC.setChangeableFlag(this.changeableFlag);
   // set columns for this specific TC
   newTC.setAttr1(this.attr1);
   newTC.setAttr2(this.attr2);
   newTC.commitCopyHelper();
   return newTCId;
}
```

3. 작업을 저장하십시오.

getXMLString 메소드 추가: 이 단계에서는 다음과 같이 MySubTCBean에 새 getXMLString 메소드를 작성해야 합니다.

1. **MySubTCBean** 클래스를 두 번 눌러 열고 해당 소스 코드를 보십시오.
2. 다음과 같이 메소드를 대체하십시오.

```
public String getXMLString() throws javax.ejb.CreateException,
   javax.ejb.FinderException,
   javax.naming.NamingException
{
 return getXMLString(false);
}


public String getXMLString(boolean tcdata) throws
 javax.ejb.CreateException,
 javax.ejb.FinderException,
 javax.naming.NamingException
{
 String xmlTC = " <MySubTC %TC_DATA% " +
  "  attr1=\"" + this.attr1 +
  "\" attr2=\"" + this.attr2.toString() + "\">" +
  "%TC_DESC%" +
  "%PARTICIPANT%" +
  "%XML_POLICYREFERENCE%" +
  " </MySubTC>";
```

```
xmlTC = ContractUtil.replace( xmlTC, "%TC_DATA%",
     getXMLStringForTCData(tcdata));
String xmlPolicy = getXMLStringForElementPolicyReference(
    "ProductSet");
xmlTC = ContractUtil.replace( xmlTC, "%XML_POLICYREFERENCE%",
    xmlPolicy);
xmlTC = ContractUtil.replaceAll( xmlTC, "%POLICY_REF_TYPE%",
    "ProductSetPolicyRef");
return xmlTC;
}
```

3. 작업을 저장하십시오.

markForDelete 메소드 추가: 이 단계에서는 다음과 같이 MySubTCBean에 새 markForDelete 메소드를 작성해야 합니다.

1. **MySubTCBean** 클래스를 두 번 눌러 열고 해당 소스 코드를 보십시오.
2. 다음과 같이 메소드를 대체하십시오.

```
public void markForDelete() throws
 javax.ejb.CreateException,
 javax.ejb.FinderException,
 javax.naming.NamingException
{
 // code: remove entries from associated tables which
 // cannot be deleted though delete cascade
}
```

3. 작업을 저장하십시오.

원격 인터페이스 갱신: 다음과 같이 원격 인터페이스에 다음 메소드를 추가했는지 확인해야 합니다.

1. J2EE 네비게이터 보기로 전환하십시오.
2. **Enablement-RelationshipManagementData** 프로젝트를 펼치십시오.
3. **com.ibm.commerce.contract.objects** 패키지를 펼치십시오.
4. **MySubTCBean** bean을 두 번 누르십시오.
5. 아웃라인 보기에서 **getXMLString()** 메소드를 마우스 오른쪽 버튼으로 누르고 **엔터프라이즈 bean > 원격 인터페이스로 승격**을 선택하십시오.
6. 아웃라인 보기에서 **getXMLString(boolean tcdata)** 메소드를 마우스 오른쪽 버튼으로 누르고 **엔터프라이즈 bean > 원격 인터페이스로 승격**을 선택하십시오.

7. 아웃라인 보기에서 **parseXMLElement(org.w3c.dom.Element argElement)** 메소드를 마우스 오른쪽 버튼으로 누르고 **엔터프라이즈 bean > 원격 인터페이스로 승격**을 선택하십시오.
8. 아웃라인 보기에서 **createNewVersion(Long argNewTradingId)** 메소드를 마우스 오른쪽 버튼으로 누르고 **엔터프라이즈 bean > 원격 인터페이스로 승격**을 선택하십시오.
9. 아웃라인 보기에서 **markForDelete()** 메소드를 마우스 오른쪽 버튼으로 누르고 **엔터프라이즈 bean > 원격 인터페이스로 승격**을 선택하십시오.
10. 변경사항을 저장하십시오.

MySubTC의 액세스 bean 작성: 액세스 bean을 작성하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼친 후 **Enablement-RelationshipManagementData**를 마우스 오른쪽 버튼으로 누르고 **새로 만들기 > 액세스 bean**을 선택하십시오.
   액세스 bean 추가 창이 열립니다.
2. **복사 헬퍼**를 선택한 후 **다음**을 누르십시오.
3. ***MySubTC*** bean을 선택하고 **다음**을 누르십시오.
4. 생성자 메소드 드롭 다운 목록에서 **findByPrimaryKey(com.ibm.commerce.contract.objects.*MySubTC*Key)**를 생성자 메소드로 선택하십시오.
5. 속성 헬퍼 섹션에서 모든 속성을 선택하십시오.
6. **완료**를 누릅니다.
7. 작업을 저장하십시오.

새 문자열 변환기 추가: 다음과 같이 새 문자열 변환기를 추가해야 합니다.

1. J2EE 계층 보기에서 **EJB 모듈 > Enablement-RelationshipManagementData > ejbModule > META-INF**를 펼치십시오.
2. **ibm-ejb-access-bean.xmi** 파일을 두 번 누르십시오.
3. MySubTC에 대한 섹션을 찾으십시오.
4. 각 copyHelperProperties 요소에 대해 다음 속성을 추가하십시오.

```
converterClassName="com.ibm.commerce.base.objects.WCSStringConverter"
```

5. 작업을 저장하십시오.
6. 이제 다음과 같이 MySubTC의 액세스 bean을 다시 작성해야 합니다.
   a. **Enablement-RelationshipManagementData**를 마우스 오른쪽 버튼으로 누르고 **액세스 bean > 액세스 bean 다시 작성**을 선택하십시오.
   b. **MySubTC**를 선택하고 **완료**를 누르십시오.

전개 코드 작성: 다음과 같이 TermCondition bean 및 MySubTC bean 모두에 대한 전개 코드를 작성해야 합니다.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼치십시오.
2. **Enablement-RelationshipManagementData**를 마우스 오른쪽 버튼으로 누르고 **작성 > 전개 및 RMIC 코드**를 선택하십시오.
3. **MySubTC**를 선택하고 **완료**를 누르십시오.
4. **Enablement-RelationshipManagementData**를 마우스 오른쪽 버튼으로 누르고 **작성 > 전개 및 RMIC 코드**를 선택하십시오.
5. **모두 선택**을 누르고 **완료**를 누르십시오.

   주: 반드시 상위 bean(TermCondition bean) 및 모든 형제 bean(이름에 "TC"가 있는, Enablement-RelationshipManagementData 그룹의 모든 나머지 bean)에 대한 전개 코드를 다시 작성해야 합니다. 새 필드를 추가했거나 기존 TermCondition bean의 원격 인터페이스를 수정한 경우, 자신 및 모든 하위 bean의 액세스 bean을 다시 작성해야 함에 유의하십시오. 편의상, 앞의 지시사항은 이 프로젝트의 모든 bean을 선택합니다.

validateContract 태스크 명령의 메소드 대체: 다음 단계는 `ValidateContractCmd` 태스크 명령에 있는 메소드를 대체하는 것입니다. 이 명령에는 새 규정 오브젝트를 지원하기 위해 대체하려는 세 가지 메소드가 있습니다. 해당 메소드는 다음과 같습니다.

- `validateTCType()`
  이 메소드는 장기 구매 계약에서 사용할 수 있는 조건 유형을 확인합니다. 예를 들어, InvoiceTC는 계정에 속하므로 장기 구매 계약에 나타날 수 없습니다.

- validateTCOccurrence()
  이 메소드는 조건 어커런스를 확인합니다. 예를 들어, 이 메소드의 기본 구현의 경우 장기 구매 계약이 최소 하나의 PriceTC를 가져야 합니다.
- otherValidateCheck()
  이 메소드의 기본 구현은 비어 있습니다. 처음 두 개의 메소드에 해당하지 않는 추가 유효성 검증을 추가할 수 있습니다.

이 수정 방법에 대한 정보는 181 페이지의 『기존 태스크 명령 사용자 정의』를 참조하십시오.

새 전개 명령 작성: 규정을 전개해야 하는 경우, 새 전개 명령을 작성하고 이 명령을 데이터베이스에 등록해야 합니다. 필요한 경우 다음을 수행하십시오.

1. 이 예의 경우, 새 전개 명령 인터페이스는 MySubTCDeployedCmd이고 구현 클래스는 MySubTCDeployedCmdImpl입니다. 또한 명령은 packagename 패키지에 패키지화됩니다. 이 명령을 등록하려면 다음 SQL 명령을 발행하십시오.

   ```
   insert into CMDREG (STOREENT_ID, INTERFACENAME, CLASSNAME, TARGET)
   values (0, 'packagename.MySubTCDeployCmd',
   'packagename.MySubTCDeployCmdImpl', 'Local');
   ```

2. packagename 패키지에서 새 MySubTCDeployedCmd 인터페이스를 작성하십시오. 이 인터페이스는 com.ibm.commerce.contract.commands.DeployTCCmd 명령 인터페이스를 확장해야 합니다. 다음은 새 명령 인터페이스에 대해 설명합니다.

   ```
   public interface MySubTCDeployCmd extends
      com.ibm.commerce.contract.commands.DeployTCCmd
      {
          // customized code
      }
   ```

   DeployTCCmd에는 보호된 매개변수 abTC 및 getTargetStoreId() 메소드가 있습니다. abTC의 값은 MySubTCAccessBean이고 getTargetStoreId() 메소드는 장기 구매 계약이 전개되는 상점의 식별자를 리턴합니다.

3. 동일한 패키지에서 MySubTCDeployCmdImpl 구현 클래스를 작성하십시오. 이 구현 클래스는 com.ibm.commerce.contract.commands.DeployTCCmdImpl을 확장해야 합니다. 다음은 새 명령 구현 클래스에 대해 설명합니다.

```
public class MySubTCDeployCmdImpl
   extends com.ibm.commerce.contract.commands.DeployTCCmdImpl
   implements MySubTCDeployCmd
   {
     // customer code
   }
```

새 규정을 사용하도록 WebSphere Commerce 액셀러레이터 갱신

규정을 새로 만들면 이 새 규정을 포함하는 새 장기 구매 계약을 작성하는 데 사용할 수 있도록 WebSphere Commerce 액셀러레이터를 갱신할 수 있습니다. 이러한 목적으로 WebSphere Commerce 액셀러레이터를 갱신하는 데는 다음 단계가 포함됩니다.

1. 새 규정에 대한 새 JavaScript 파일 작성. 이 절에서 예를 들기 위해 이 파일을 Extensions.js라고 합니다.
2. 사용자가 새 규정의 필수 정보를 입력할 수 있는 HTML 섹션을 포함하는 새 JSP 템플리트 작성. 이 절에서 예를 들기 위해 이 파일을 ContractMyTC.jsp라고 합니다.
3. 새 규정에 대한 새 데이터 bean 작성. 이 절에서 예를 들기 위해 이 파일을 MyTCDataBean이라고 합니다.
4. VIEWREG 테이블에 새 뷰 등록.
5. 새 자원을 포함하도록 ContractRB_*locale*.properties 파일 갱신.
6. 새 페이지를 포함하도록 ContractNotebook.xml 파일 편집.

이들 각 단계는 다음 절에서 자세히 설명합니다.

새 JavaScript 파일 작성: 새 조건을 사용하도록 WebSphere Commerce 액셀러레이터를 갱신하기 위한 첫 번째 단계는 새 조건에 대한 새 JavaScript 파일을 작성하는 것입니다. 다음 견본 파일을 참조할 수 있습니다.

- Windows *WC_installdir*\samples\contract\Extensions.js
- Studio *WCStudio_installdir*\samples\contract\Extensions.js
- AIX Solaris Linux 400 *WC_installdir*/samples/contract/Extensions.js

이 견본 파일을 사용하려면 다음 디렉토리로 복사하십시오.

- Windows *WAS_installdir*\installedApps*cell_name*\WC_*instanceName*.ear\CommerceAccelerator.war\tools\contract
- Studio *workspace_dir*\CommerceAccelerator\Web Content\tools\contract
- AIX Solaris Linux *WAS_installdir*/installedApps/*cell_name*/WC_*instanceName*.ear/CommerceAccelerator.war/tools/contract
- 400 *WAS_userdir*/installedApps/*cell_name*/WC_*instanceName*.ear/ CommerceAccelerator.war/tools/contract

여기서, *instanceName*은 WebSphere Commerce 인스턴스의 이름이며 *cell_name*은 WebSphere Application Server 셀 이름입니다.

이 새 파일에서 새 규정에 대한 데이터를 저장할 JavaScript 오브젝트를 작성해야 합니다. 이 내용은 다음 코드 단편에 표시됩니다.

```
function ContractMyTCModel() {

   this.tcReferenceNumber = "";
   this.policyReferenceNumber = "";

   this.attr1 = "";
   this.attr2 = "";

   this.policyList = new Array();
   this.selectedPolicyIndex = "0";
}
```

새 규정을 제출할 새 JavaScript 오브젝트 또한 작성해야 합니다. 이는 XSD 파일의 확장과 일관된 방법으로 수행해야 합니다. 이 내용은 다음 코드 단편에 표시됩니다.

```
function submitMyTC(contract) {

  var tcModel = get("ContractMyTCModel");

  if (tcModel != null) {

 var myTC = new Object();
 myTC.attr1 = tcModel.attr1;
 myTC.attr2 = tcModel.attr2;
```

```
myTC.ProductSetPolicyRef = new Object();
myTC.ProductSetPolicyRef.policyName = tcModel.policyList[
     tcModel.selectedPolicyIndex].policyName;
myTC.ProductSetPolicyRef.StoreRef = new Object();
myTC.ProductSetPolicyRef.StoreRef.name = tcModel.policyList[
     tcModel.selectedPolicyIndex].storeIdentity;
myTC.ProductSetPolicyRef.StoreRef.Owner = new Object();
myTC.ProductSetPolicyRef.StoreRef.Owner = tcModel.policyList[
     tcModel.selectedPolicyIndex].member;

if (tcModel.tcReferenceNumber != "") {
 // Change the term and condition
 myTC.action = "update";
 myTC.referenceNumber = tcModel.tcReferenceNumber;
}
else {
 // Create a new term and condition
 myTC.action = "new";
}

contract.MySubTC = myTC;
 }

 return true;
}
```

새 JSP 템플리트 작성: 다음 단계는 사용자가 새 규정에 필요한 정보를 입력할 수 있는 HTML 섹션을 포함하는 새 JSP 템플리트 작성하는 것입니다. 다음 견본 파일을 참조할 수 있습니다.

- Windows *WC_installdir*\samples\contract\ContractMyTC.jsp
- Studio *WCStudio_installdir*\samples\contract\ContractMyTC.jsp
- AIX Solaris Linux 400 *WC_installdir*/samples/contract/ContractMyTC.jsp

이 견본 파일을 사용하려면 다음 디렉토리로 복사하십시오.

- Windows *WAS_installdir*\installedApps*cell_name*\WC_*instanceName*.ear\CommerceAccelerator.war\javascript\tools\contract

- Studio *workspace_dir*\CommerceAccelerator\Web Content\tools\contract
- AIX Solaris Linux *WAS_installdir*/installedApps/*cell_name*/WC_*instanceName*.ear/CommerceAccelerator.war/ javascript/tools/contract
- 400 *WAS_userdir*/installedApps/*cell_name*/WC_*instanceName*.ear/ CommerceAccelerator.war/javascript/tools/contract

여기서, *instanceName*은 WebSphere Commerce 인스턴스의 이름이며 *cell_name*은 WebSphere Application Server 셀의 이름입니다.

다음 코드 단편은 MyTC에 사용할 수 있는 JSP 템플리트의 HTML 섹션 예를 표시합니다.

```
<!--
////////////////////////////////////
// HTML SECTION
////////////////////////////////////
-->

<BODY onLoad="onLoad()" class="content">

   <H1>
   <%= contractsRB.get("MyTCHeading") %>
   </H1>

<FORM NAME="MyTCForm">

   <%= contractsRB.get("MyTCAttr1Label") %>
   <BR>
   <INPUT type=text name=Attr1 value="" size=10 maxlength=10>
   <BR>

   <%= contractsRB.get("MyTCAttr2Label") %>
   <BR>
   <INPUT type=text name=Attr2 value="" size=10 maxlength=10>
   <BR>

   <%= contractsRB.get("MyTCPolicyLabel") %>
   <BR>
```

```
   <SELECT NAME="PolicyList" SIZE="1">
   </SELECT>

</FORM>
```

새 데이터 bean 작성: 이 단계에서는 MySubTC 액세스 bean에서 필요한 데이터를 로드하는 새 데이터 bean을 작성합니다. 관련 코드 섹션은 다음 코드 단편에 표시됩니다.

```
public class MyTCDataBean extends MySubTCAccessBean
   implements SmartDataBean, Delegator {
      private java.lang.Long contractId;
      private boolean hasMyTC = false;
      private CommandContext iCommandContext;


      /**
       * MyTCDataBean default constructor.
       */
      public MyTCDataBean() {
      }
      /**
       * MyTCDataBean constructor.
       */
      public MyTCDataBean(Long newContractId) {
       contractId = newContractId;
      }


      /*
       * populate the attributes from TermConditionAccessBean
       */
      public void populate() throws Exception {

                Enumeration myTCEnum = new TermConditionAccessBean().
                     findByTradingAndTCSubType(contractId, "MySubTC");
                if (myTCEnum != null) {
                        // assume a contract only has one MyTC for this example

                        setEJBRef(((TermConditionAccessBean)
                              myTCEnum.nextElement()).getEJBRef());
                        refreshCopyHelper();
                      hasMyTC = true;
                  }

}
```

VIEWREG 테이블에 새 뷰 등록: VIEWREG 테이블에 새로 작성한 뷰를 등록해야 합니다. 다음은 새 뷰를 등록하기 위한 SQL문 예입니다.

```
insert into VIEWREG(VIEWNAME,DEVICEFMT_ID,STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, HTTPS, INTERNAL)
values ('ContractMyTCPanelView', -1, 0,
   'com.ibm.commerce.tools.command.ToolsForwardViewCommand',
   'com.ibm.commerce.tools.command.ToolsForwardViewCommandImpl',
   'docname=tools/contract/ContractMyTC.jsp', 1, 1)
```

ContractRB_locale.properties **파일 갱신:** 새 규정에 고유한 정보로 다음 특성 파일을 갱신해야 합니다.

- Windows *WAS_installdir*\installedApps*cell_name*\WC_*instanceName*.ear\properties\com\ibm\commerce\tools\contract\properties\ ContractRB_*locale*.properties
- Studio *workspace_dir*\WebSphereCommerceServer\properties\com\ibm\commerce\tools\contract\properties\ContractRB_*locale*.properties
- AIX Solaris Linux *WAS_installdir*/installedApps/*cell_name*/WC_*instanceName*.ear/properties/ com/ibm/commerce/tools/contract/properties/ ContractRB_*locale*.properties
- 400 *WAS_userdir*/installedApps/*cell_name*/WC_*instanceName*.ear/ properties/com/ibm/commerce/tools/contract/properties/ ContractRB_*locale*.properties

여기서, *instanceName*은 WebSphere Commerce 인스턴스의 이름이며 *cell_name*은 WebSphere Application Server 셀의 이름입니다.

다음은 파일에 추가할 정보의 한 예입니다.

```
MyTCHeading=My TC
attr1Empty=Attribute One must be entered.
attr2Empty=Attribute Two must be entered.
attr1TooLong=Attribute One is too long.
attr2TooLong=Attribute Two is too long.
MyTCAttr1Label=Attribute One (required)
MyTCAttr2Label=Attribute Two (required)
MyTCPolicyLabel=Policy
```

ContractNotebook.xml **파일 편집:** WebSphere Commerce 액셀러레이터에 새 규정을 포함시키기 위한 마지막 단계는 새 페이지를 포함하도록 다음 파일을 갱신하는 것입니다.

- Windows *WC_installdir*\xml\tools\contract\ContractNotebook.xml
- Studio *WCStudio_installdir*\Commerce\xml\tools\contract\ContractNotebook.xml
- AIX Solaris Linux *WC_installdir*/xml/tools/contract/ContractNotebook.xml
- 400 *WC_installdir*/xml/tools/contract/ContractNotebook.xml

다음은 이 예의 새 페이지를 포함시키는 데 사용되는 코드 단편 예입니다.

```
<panel name="MyTCHeading"
    url="ContractMyTCPanelView"
    parameters="contractId,accountId"
    helpKey="MC.contract.MyTCPanel.Help" />
```

새 규정을 사용하는 새 장기 구매 계약 반입

새 규정을 사용하도록 WebSphere Commerce 도구를 갱신하기 위한 대안으로 contract import 명령(이 명령에 대한 정보는 WebSphere Commerce 온라인 도움말 참조)을 사용하여 이 새 규정을 포함하는 새 장기 구매 계약을 반입할 수 있습니다. 반입 후 Contract.xml 파일의 관련 섹션은 다음과 같이 나타납니다.

```
<MySubTC attr1="abc" attr2="123">
       <ProductSetPolicyRef policyName = "Product Set 1">
           <StoreRef name = "StoreGroup1">
               <Owner>
                   <OrganizationRef distinguishName = "o=Root Organization"/>
               </Owner>
           </StoreRef>
       </ProductSetPolicyRef>
   </MySubTC>
```

새 비즈니스 정책 호출

새 비즈니스 정책을 작성하고 이 비즈니스 정책이 최소 하나의 규정 오브젝트와 연관되면 새 비즈니스 정책 명령을 호출하도록 응용프로그램 로직을 갱신해야 합니다.

비즈니스 정책 명령은 컨트롤러 및 태스크 명령에서 호출됩니다.

명령 팩토리를 사용하여 비즈니스 정책 명령을 호출합니다. 비즈니스 정책 명령을 호출하는 데 사용할 수 있는 두 개의 create 메소드가 있습니다. 첫 번째 메소드는 비즈니스 정책과 연관된 비즈니스 정책 명령이 하나 뿐인 경우 비즈니스 정책 명령을 호출하는 데 사용됩니다. 이 메소드는 다음 코드 단편에 표시됩니다.

```
CommandFactory createBusinessPolicyCommand(Long policyId);
```

두 번째 메소드는 비즈니스 정책과 연관된 비즈니스 정책 명령이 두 개 이상인 경우 비즈니스 정책 명령을 호출하는 데 사용됩니다. 이 메소드는 다음 코드 단편에 표시됩니다.

```
CommandFactory createBusinessPolicyCommand(Long policyId, String cmdIfName);
```

앞 예에서 cmdIfName은 작성할 비즈니스 정책 명령의 인터페이스 이름을 지정합니다.

명령 팩토리는 POLICYCMD 테이블에서 정책 오브젝트를 검색하여 이 정책을 구현하는 명령을 판별합니다. 또한 테이블에서 기본 특성을 가져와 비즈니스 정책 명령에 requestProperties로 설정합니다.

다음 코드 단편은 환불 정책을 호출하는 예를 표시합니다.

```
RefundPolicyCmd  cmd;

   ////////////////////////////////////////////////////////
   // Get the refund policy id from the refundTC object     //
   //  and use it to create the policy command.             //
   ////////////////////////////////////////////////////////
 cmd = (RefundPolicyCmd) CommandFactory
         createPolicyCommand (refundTC.getRefundPolicy);

cmd.execute()
```

장기 구매 계약 작성

장기 구매 계약 모델로의 확장을 비즈니스 프로세스와 완전 통합하기 위한 다음 단계는 새 비즈니스 정책을 참조하는 규정을 포함하는 장기 구매 계약을 작성하는 것입니다. 장기 구매 계약은 WebSphere Commerce 액셀러레이터를 사용하거나 장기 구매 계약 URL 명령(ContractImportApprovedVersion and ContractImportDraftVersion) 중 하나를 사용하여 작성할 수 있습니다. 장기 구매 계약 작성에 대한 추가 정보는 WebSphere Commerce Production 및 Development 온라인 도움말을 참조하십시오.

장기 구매 계약 사용자 정의 시나리오

이 절에서는 다음 장기 구매 계약 사용자 정의 시나리오와 관련된 단계의 개요를 제공합니다.

- 리베이트 사용

리베이트 시나리오

이 시나리오 예에서는 균일 요금 리베이트가 작성됩니다. ToolTech 견본 상점에는 리베이트 시나리오와 일치하는 정책 유형 또는 규정이 없으므로 이를 작성해야 합니다. 또한 리베이트 코드를 저장할 데이터베이스 테이블 및 새 비즈니스 정책을 작성해야 합니다.

이 리베이트 시나리오 구현에는 다음과 같은 상위 레벨 단계가 포함됩니다.

1. XREBATECODE 데이터베이스 테이블 및 이 테이블 정보에 액세스하는 데 사용되는 해당 XRebateCodeBean 엔티티 bean 작성
2. 다음 서브태스크를 수행하여 새 5DollarRebate 비즈니스 정책 작성
   a. 해당 새 비즈니스 정책 유형을 작성하십시오. 이는 새 비즈니스 정책 명령이 구현할 인터페이스(RebatePolicyCmd)를 정의합니다.
   b. 새 CalculateRebateCmdImpl 비즈니스 정책 명령을 작성하십시오.
   c. 새 비즈니스 정책 명령 및 비즈니스 정책 유형을 데이터베이스에 등록하십시오.
3. 다음 서브태스크를 수행하여 새 리베이트 규정(RebateTC)을 작성하십시오.

a. 데이터베이스에 RebateTC 규정을 등록하십시오.
b. 새 RebateTC를 반영하도록 XSD 파일을 갱신하십시오.
c. RebateTC에 대한 새 엔터프라이즈 bean을 작성하십시오.
d. 새 RebateTC를 반영하도록 WebSphere Commerce 액셀러레이터를 갱신하십시오.

4. RebateTC를 사용하는 새 장기 구매 계약을 작성하십시오.
5. 쇼핑 플로우로 새 비즈니스 정책 통합

이러한 각 단계는 후속 절에서 자세히 설명합니다.

1단계: 새 테이블 및 엔터프라이즈 bean 작성

기본 데이터베이스 스키마에는 리베이트 총계 및 코드의 스펙이 포함되어 있지 않으므로 새 테이블을 작성해야 합니다. 일반적으로 새 테이블이 작성되면 이 테이블에 들어 있는 정보에 액세스 시 사용되는 새 엔티티 bean 또한 작성됩니다.

예를 들기 위해 다음 XREBATECODE 데이터베이스 테이블이 작성되는 것으로 가정하십시오.

표 2. XREBATECODE 데이터베이스 테이블

	열 이름		
	REBATECODE_ID	AMOUNT	CURRENCY
견본 데이터	201	5	CAD
	202	10	CAD

또한 새 CMP 엔티티 bean(XRebateCodeBean)이 작성됩니다. 이 bean 작성에 대한 자세한 정보는 67 페이지의 『새 CMP 엔터프라이즈 bean 작성』을 참조하십시오.

2단계: "5DollarRebate" 비즈니스 정책 작성

이 새 비즈니스 정책을 작성하려면 다음 단계를 수행해야 합니다.

1. 새 비즈니스 정책 유형 인터페이스를 작성하십시오. 이는 `CalculateRebateCmdImpl`이 구현할 `RebatePolicyCmd` 인터페이스입니다.
2. 새 `CalculateRebateCmdImpl` 비즈니스 정책 명령을 작성하십시오.
3. 새 비즈니스 정책 및 비즈니스 정책 명령을 데이터베이스에 등록하십시오.

"리베이트" 비즈니스 정책 유형 작성: 리베이트에 해당하는 기존 비즈니스 정책 유형이 없으므로 새 유형을 작성해야 합니다. 새 비즈니스 정책 유형의 작성에는 데이터베이스에 정책 유형의 정의 및 등록이 포함됩니다. 다음 테이블을 갱신해야 합니다.

- POLICYTYPE
- PLCYTYCMIF
- PLCYTYPDSC

이 시나리오의 경우, 새 REBATE 정책 유형을 작성하려면 다음 SQL문을 사용하게 됩니다.

```
insert into POLICYTYPE (POLICYTYPE_ID) values ('Rebate');
insert into PLCYTYCMIF (POLICYTYPE_ID, BUSINESSCMDIF)
   values ('Rebate',
   'com.mycompany.mybusinesspolicycommands.RebatePolicyCmd');
insert into  PLCYTYPDSC (POLICYTYPE_ID, LANGUAGE_ID, DESCRIPTION)
   values ('Rebate', -1,
   'Rebate policy type.');
```

결과적으로 정책 유형과 관련 비즈니스 정책 명령 사이의 관계를 표시하는 PLCYTYCMIF 테이블의 관련 열이 다음 테이블에 표시됩니다.

표 3. PLCYTYCMIF 테이블 갱신사항

	열 이름	
	POLICYTYPE_ID	BUSINESSCMDIF
견본 데이터	리베이트	com.mycompany.mybusinesspolicycommands.RebatePolicyCmd

또한 새 RebatePolicyCmd 인터페이스를 코딩해야 합니다. 이 인터페이스는 com.ibm.commerce.command.BusinessPolicyCommand 인터페이스를 확장해야 합니다. 이전 테이블에서 제안한 대로 이 인터페이스를 사용자 고유 패키지에 함께 포함하십시오.

CalculateRebateCmdImpl 비즈니스 정책 명령 작성: 새 비즈니스 정책 명령을 작성하려면 com.ibm.commerce.command.BusinessPolicyCommandImpl 구현 클래스를 확장하는 새 명령 CalculateRebateCmdImpl을 작성해야 합니다. 이 명령은 이전 단계에서 작성된 RebatePolicyCmd 인터페이스를 구현해야 합니다.

이 예에서는 인터페이스 이름과 명령 이름이 다름에 유의하십시오. 이러한 이름은 비즈니스 정책의 리베이트 유형을 구현하는 비즈니스 정책 명령이 많다는 것을 의도적으로 보여주기 위해 선택되었습니다. 그러면 각 구현(즉, 각 비즈니스 정책 명령)은 고유한 방식으로 리베이트를 구현하게 됩니다.

명령의 로직은 고객이 상품을 오더피킹하는 방법의 구현에 따라 다릅니다. 또한 이 `CalculateRebateCmdImpl`은 응용프로그램의 개별 컨트롤러 또는 태스크 명령으로 호출해야 합니다.

새 비즈니스 정책 및 새 비즈니스 정책 명령 등록: 새 비즈니스 정책은 데이터베이스에 등록해야 합니다. 새 비즈니스 정책 및 새 비즈니스 정책 명령 간의 관계 또한 등록해야 합니다.

이 정보를 등록하려면 `com.ibm.commerce.contract.commands.PolicyAddCmd` 명령을 사용할 수 있습니다. 다음은 이 시나리오에서의 PolicyAdd 명령 사용법 예를 표시합니다.

```
http://localhost:8080/webapp/wcs/stores/servlet/PolicyAdd?
   type=Rebate&name=5DollarRebate&plcyStoreId=-1
   &cmd_1=com.mycompany.mybusinesspolicycommands.CalculateRebateCmdImpl
   &startDate=2002-05-08%2000:00:00&endDate=2003-05-09%2000:00:00
   &commonProps=rebatecode_id%3D501&URL=aRedirectURL
```

URL 전용어를 입력 특성에 대한 ASCII 코드로 바꿔야 함에 유의하십시오. 즉, typical = (같음) 기호는 “%3D”로 바꾸고, &(앰퍼샌드)는 “%26”으로 바꾸고, 공백 문자는 “%20”으로 바꿉니다. 앞 예에서는 날짜 포맷으로 yyyy-mm-dd hh:mm:ss를 사용했으며 URL 전용어를 ASCII 코드로 바꾸었습니다.

다음 테이블은 갱신 후 영향을 받은 데이터베이스 테이블의 관련 열을 표시합니다.

표 4. POLICY 테이블 갱신사항

	열 이름				
	POLICY_ID	POLICY NAME	POLICYTYPE_ID	STOREENT_ID	PROPERTIES
견본 데이터	301	5Dollar 리베이트	리베이트	-1	rebatecode_id= 201

또한 시작 날짜 및 종료 날짜 값을 null로 설정된 것으로 가정함에 유의하십시오.

표 5. POLICYCMD 테이블 갱신사항

	열 이름		
	POLICY _ID	BUSINESS CMDCLASS	PROPERTIES
견본 데이터	301	com.mycompany. mybusinesspolicycommands. CalculateRebateCmdImpl	널(NULL)값

결과적으로 CalculateRebateCmd 비즈니스 정책 명령과 관련되는 새 "5DollarRebate" 비즈니스 정책이 작성됩니다.

3단계: "RebateTC" 규정 작성

"RebateTC" 규정을 작성하려면 다음 단계를 수행해야 합니다.

1. 데이터베이스에 RebateTC 규정을 등록하십시오.
2. 새 RebateTC를 반영하도록 XSD 파일을 갱신하십시오.
3. RebateTC에 대한 새 엔터프라이즈 bean을 작성하십시오.
4. 새 RebateTC를 반영하도록 WebSphere Commerce 액셀러레이터를 갱신하십시오.

데이터베이스에 "RebateTC" 규정 등록: 새 규정 오브젝트를 작성하는 경우, 이 오브젝트를 포함하도록 데이터베이스 스키마를 갱신해야 합니다. 갱신해야 하는 데이터베이스 테이블은 TCTYPE 및 TCSUBTYPE입니다.

다음 SQL문은 데이터베이스에 RebateTC를 등록하는 방법 예를 표시합니다.

```
insert into TCTYPE (TCTYPE_ID) values ('RebateTC');
insert into TCSUBTYPE (TCSUBTYPE_ID, TCTYPE_ID, ACCESSBEANNAME,
   DEPLOYCOMMAND)
values ('RebateTC', 'RebateTC',
   'com.ibm.commerce.contract.objects.RebateTCAccessBean',
   null);
```

다음 테이블은 TCTYPE 및 TCSUBTYPE 테이블에 있는 관련 열을 표시합니다.

표 6. TCTYPE 테이블 갱신사항

	열 이름
	TCTYPE_ID
견본 데이터	RebateTC

표 7. TCSUBTYPE 테이블 갱신사항

	열 이름			
	TCSUBTYPE _ID	TCTYPE _ID	ACCESSBEAN NAME	D E P L O Y COMMAND
견본 데이터	RebateTC	RebateTC	com.ibm.commerce. contract. objects. RebateTCAccessBean	널(NULL)값

장기 구매 계약 문서 유형 정의에 리베이트 규정 등록: 장기 구매 계약에 리베이트 규정을 사용할 수 있도록 하려면 새 규정을 정의하는 새 XSD 파일을 작성해야 합니다. 또한 새 XSD 파일을 포함하도록 Package.xsd 파일을 갱신해야 합니다.

새 XSD 파일을 작성하려면 다음을 수행하십시오.

1. 다음 디렉토리로 이동하십시오.
   - Windows *WC_installdir*\xml\trading\xsd
   - AIX Solaris Linux *WC_installdir*/xml/trading/xsd
   - 400 *WC_userdir*/instances/*instanceName*/xml/trading/xsd

   여기서, *instanceName*은 WebSphere Commerce 인스턴스 이름입니다.
2. 이 디렉토리에 새 XSD 파일을 작성하십시오. 다음은 RebateTC 예에 사용할 XSD를 표시합니다. 파일은 RebateCustomizedBuyerContract.xsd입니다.

```
<?xml version="1.0"?>
<schema targetNamespace="http://www.ibm.com/WebSphereCommerce"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:wc="http://www.ibm.com/WebSphereCommerce"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">

  <!-- include basic trading agreement xsd -->
```

```
<include schemaLocation="BuyerContract.xsd" />
<complexType name="RebateTCType">
          <complexContent>
          <extension base="wc:TermConditionType"/>
          </complexContent>
      </complexType>
  <element name="RebateTC" substitutionGroup=
  "wc:AbstractCustomizedTC">
      <complexType>
          <complexContent>
              <extension base="wc:RebateTCType">
                  <sequence>
                      <element ref="wc:RebatePolicyRef"/>
                  </sequence>
              </extension>
          </complexContent>
      </complexType>
  </element>

  <element name="RebatePolicyRef"
  type="wc:BusinessPolicyRef" />

</schema>
```

3. 새 파일을 저장하십시오.

그 다음 BuyerContract.xsd를 제거하고 대신 RebateCustomizedBuyerContract.xsd를 포함하도록 다음과 같이 Package.xsd를 갱신해야 합니다.

1. 다음 디렉토리로 이동하십시오.
   - Windows *WC_installdir*\xml\trading\xsd
   - AIX Solaris Linux *WC_installdir*/xml/trading/xsd
   - 400 *WC_userdir*/instances/*instanceName*/xml/trading/xsd

   여기서, *instanceName*은 WebSphere Commerce 인스턴스 이름입니다.
2. 텍스트 편집기에서 Package.xsd 파일을 여십시오.
3. BuyerContract.xsd에 대한 섹션을 찾아 다음과 같이 수정하십시오.

```
<!--include schemaLocation="BuyerContract.xsd"/-->
<include schemaLocation="RebateCustomizedBuyerContract.xsd"/>
```

RebateTC에 대한 새 엔터프라이즈 bean 작성: 새 RebateTC에 새 엔터프라이즈 bean을 작성해야 합니다. 이 새 bean은 WebSphere Commerce TermCondition bean에서 상속되어야 합니다.

규정에 대한 새 엔터프라이즈 bean의 이름은 일반적으로 하위 유형의 이름을 따릅니다. 이 경우 규정 하위 유형은 규정 유형과 동일하므로, bean의 이름도 규정 유형과 동일함에 유의하십시오.

다음 테이블은 작성해야 하는 새 bean에 대한 몇 가지 일반 정보를 표시합니다. 대체할 메소드를 포함한 bean에 대한 자세한 정보는 200 페이지의 『규정에 대한 새 CMP 엔터프라이즈 bean 작성』을 참조하십시오.

표 8.

속성	값
EJB 프로젝트	Enablement-RelationshipManagementData
bean 유형	CMP(Container-Managed Persistence) 필드가 있는 엔티티 bean
bean 이름	RebateTC
패키지	com.ibm.commerce.contract.objects
bean supertype	TermCondition
bean 클래스	RebateTCBean

이 새 bean에서 다음 값에 사용되는 세 개의 CMP 필드를 작성하십시오.

- 리베이트 코드 ID
- 총계
- 통화

RebateTC를 포함하도록 WebSphere Commerce 액셀러레이터 갱신: 규정을 새로 만들면 이 새 규정을 포함하는 새 장기 구매 계약을 작성하는 데 사용할 수 있도록 WebSphere Commerce 액셀러레이터를 갱신할 수 있습니다. 이 도구 갱신 방법에 대한 정보는 210 페이지의 『새 규정을 사용하도록 WebSphere Commerce 액셀러레이터 갱신』을 참조하십시오.

4단계: 새 장기 구매 계약 작성

"RebateTC" 규정을 포함하고 "5DollarRebate" 비즈니스 정책을 참조하는 새 장기 구매 계약을 작성해야 합니다. WebSphere Commerce 액셀러레이터 또는 XML

을 사용하여 새 장기 구매 계약을 작성할 수 있습니다. 새 장기 구매 계약 작성을 위한 이러한 각각의 방법은 WebSphere Commerce Production 및 Development 온라인 도움말에 설명되어 있습니다.

다음 테이블은 장기 구매 계약 작성 후 TERMCOND 및 POLICYTC 데이터베이스 테이블의 관련 열에 대한 갱신사항을 표시합니다.

표 9. TERMCOND 테이블 갱신사항

	열 이름		
	TRADING _ID	TERMCOND _ID	TCSUBTYPE _ID
견본 데이터	25	901	RebateTC

표 10. POLICYTC 테이블 갱신사항

	열 이름	
	POLICY _ID	TERMCOND_ID
견본 데이터	301	901

5단계: 쇼핑 플로우로 새 비즈니스 정책 통합

이 시나리오에서는 고객이 로그온하여 리베이트를 청구할 수 있는 새 페이지가 상점에 추가되는 것으로 가정합니다. 고객이 리베이트 청구를 누르면 새 `RebatePolicyCmd` 인터페이스를 호출하는 명령이 호출되어야 합니다. 예를 들어, `RebatePolicyCmd`를 호출하는 새 ClaimRebateCmd 컨트롤러 명령이 있을 수 있습니다. 그러면 올바른 비즈니스 정책을 찾을 수 있고 이 경우 “5DollarRebate” 비즈니스 정책이 적용됩니다.

제 3 부 개발 환경

제 8 장 개발 환경

이 장에서는 WebSphere Commerce 응용프로그램의 사용자 정의에 사용되는 기본 개발 도구에 대해 설명합니다.

일반 개발 환경

WebSphere Commerce Business Edition에서 사용할 사용자 정의 코드 작성을 위한 권장 개발 패키지는 WebSphere Commerce Studio, Business Developer Edition 제품입니다. WebSphere Commerce Professional Edition에서 사용할 사용자 정의 코드 작성을 위한 권장 개발 패키지는 WebSphere Commerce Studio, Professional Developer Edition 제품입니다. 이 두 패키지 모두 사용자 정의 코드 작성 및 웹 개발 태스크 수행에 필요한 모든 도구를 포함합니다. 이 책에서는 이들 제품을 총칭하여 WebSphere Commerce Studio라고 합니다.

WebSphere Commerce Studio의 네 가지 기본 구성요소는 다음과 같습니다.

1. WebSphere Studio Application Developer에서 사용되는 WebSphere Commerce 작업 영역
2. 개발 데이터베이스
3. 파일 시스템 자원
4. WebSphere Studio Application Developer에 대한 WebSphere Commerce plug-in

이 개발 환경을 사용하면 사용자 정의 코드를 작성하고 WebSphere 개발 환경 컨텍스트에서 테스트할 수 있습니다.

개발 환경에서 상점을 작성하려면 WebSphere Studio Application Developer에 로컬로 정의되어 있는 WebSphereCommerceServer 테스트 서버에서 실행 중인 관리 콘솔을 실행하고, 이 도구를 사용하여 견본 상점 중 하나를 기반으로 상점을 공개하십시오. 대안으로 사용자 고유 상점을 작성할 수 있습니다.

WebSphere Commerce Studio는 제공되는 WebSphere Commerce 작업 영역 외에, 추가 도구 및 plug-in을 제공합니다. 다음 plug-in이 제공됩니다.

- WebSphere Commerce(및 WebSphere Commerce Payments) 인스턴스 관리를 지원하는 구성 관리자 plug-in
- WebSphere Studio Application Developer에서 WebSphere Commerce 온라인 도움말에 액세스할 수 있는 WebSphere Commerce 온라인 도움말 plug-in. 또한 이 plug-in을 사용하면 WebSphere Commerce 도구(예: 관리 콘솔) 실행 시 F1을 눌러 WebSphere Commerce 컨텍스트 구분 온라인 도움말을 실행할 수 있습니다.
- WebSphere Studio Application Developer에서 WebSphere Commerce API 참조 정보에 액세스할 수 있는 WebSphere Commerce API 참조 정보 plug-in

WebSphere Commerce 엔터프라이즈 bean 변환 도구 또한 제공됩니다. 이 도구를 사용하면 대상 프로덕션 데이터베이스와 다른 개발 데이터베이스를 사용하여 엔터프라이즈 bean을 개발할 수 있습니다. 예를 들어, 이 도구를 사용하면 개발 용도로 로컬 DB2 데이터베이스를 사용할 수 있지만, 프로덕션 데이터베이스는 iSeries 플랫폼에 있거나 또는 Oracle 데이터베이스일 수 있습니다.

WebSphere Studio Application Developer

WebSphere Commerce Studio 패키지에는 IBM의 핵심 개발 환경인 WebSphere Studio Application Developer가 포함됩니다. 이는 모범 사례, 템플리트, 코드 작성 및 가장 포괄적인 개발 환경을 해당 클래스에 제공함으로써 J2EE(Java2 Enterprise Edition) 및 웹 서비스 개발을 최적화하고 단순화합니다. IDE(Integrated Development Environment)는 Java 구성요소, 엔터프라이즈 bean, servlets, JSP 파일, HTML, XML 및 웹 서비스를 모두 하나의 개발 환경에서 통합 지원합니다.

여러 가지 특징이 있는데 그 중에서도 테스트 클라이언트를 빨리 작성할 수 있는 로컬 테스트 도구가 포함되어 있습니다. 또한 로컬 환경에서 단말간 코드를 테스트할 수 있는 전체 WebSphere Application Server 테스트 환경이 포함됩니다.

iSeries의 개발 환경

iSeries의 사용자 정의 코드를 작성하기 위해 특별한 개발 환경은 필요하지 않습니다. 개발 워크스테이션은 *WebSphere Commerce Studio 설치 안내서*에서 설명하는 것과 동일한 프로시저에 따라 설정됩니다. 사용된 로컬 개발 데이터베이스는 DB2여야 합니다. 이 구성을 사용하면 로컬 DB2 데이터베이스 및 로컬 WebSphereCommerceServer 테스트 서버를 사용하여 사용자 정의 코드를 작성 및 테스트할 수 있습니다.

테스트 결과 테스트 서버의 컨텍스트에서 사용자 정의 코드의 기능이 만족스러운 경우, iSeries 플랫폼에서 실행 중인 대상 WebSphere Commerce Server로 전개해야 합니다. Windows와 iSeries 플랫폼의 데이터베이스 차이점을 설명하기 위해 WebSphere Commerce 엔터프라이즈 bean 변환 도구가 제공됩니다. 이 도구에 대한 자세한 정보는 『WebSphere Commerce 엔터프라이즈 bean 변환 도구의 개요』에 제공됩니다.

프로덕션 환경에서 Oracle 데이터베이스 사용 시 개발 용도로 로컬 DB2 데이터베이스 사용

개발자는 프로덕션 환경의 데이터베이스가 Oracle 데이터베이스인 경우에도 개발 시스템에서 로컬 DB2 데이터베이스를 사용할 수 있습니다. 이런 경우, WebSphere Commerce 엔터프라이즈 bean 변환 도구를 사용하여 bean의 메타데이터를 DB2 포맷에서 Oracle 포맷으로 변환합니다. 이 도구에 대한 자세한 정보는 『WebSphere Commerce 엔터프라이즈 bean 변환 도구의 개요』에 제공됩니다.

WebSphere Commerce 엔터프라이즈 bean 변환 도구의 개요

엔터프라이즈 bean 변환 도구를 사용하는 경우는 다음 두 가지입니다.

- 대상 프로덕션 환경이 iSeries 플랫폼에서 실행 중인 경우
- 대상 프로덕션 데이터베이스가 Oracle 데이터베이스이지만 개발 시스템이 로컬 DB2 데이터베이스를 사용하는 경우

이 WebSphere Commerce 고유 도구를 사용하면 한 가지 데이터베이스 유형을 개발하고 나중에 다른 데이터베이스 유형으로 전개할 수 있습니다. 이 도구를 사

용하면 엔터프라이즈 bean의 메타데이터가 대상 데이터베이스에 적절한 포맷과 정보로 변환되며 전개 코드 또한 이 새 메타데이터를 사용하여 작성됩니다. 이 도구 사용 방법에 대한 자세한 정보는 236 페이지의『변환으로 EJB JAR 파일 작성』을 참조하십시오.

개발 환경에서 지불 옵션

이전 WebSphere Commerce Studio 버전의 경우, 개발자가 원격 지불 제공업체를 호출하지 않고 테스트 환경 상점에서 구매를 완료할 수 있는 테스트 지불 방법을 제공했습니다. 이제 WebSphere Commerce Studio 버전 5.5에서는 테스트 환경에서 WebSphere Commerce Payments 구성요소를 실행할 수 있습니다.

이는 이제 로컬 WebSphere Commerce Payments 인스턴스를 사용하는 옵션을 갖거나 원격 WebSphere Commerce Payments 인스턴스를 사용하도록 WebSphere Commerce 개발 인스턴스를 구성할 수 있음을 의미합니다.

기본적으로 로컬 WebSphere Commerce Payments 인스턴스는 WebSphere Commerce Studio 설치 시 작성됩니다. 또한 견본 상점 공개 시 이 인스턴스를 실행하면 상점에서 해당 로컬 WebSphere Commerce Payments 인스턴스를 사용하도록 자동 구성됩니다.

지불 옵션 구성에 대한 정보는 *WebSphere Commerce Studio 설치 안내서*에서 제공합니다.

제 9 장 전개 정보

WebSphere Commerce Studio에서 사용자 정의 코드를 작성하고 WebSphere 개발 환경에서 테스트한 후 WebSphere 개발 환경 외부에서 실행 중인 대상 WebSphere Commerce Server에 이를 전개해야 합니다. 이 대상 WebSphere Commerce Server는 개발 시스템에서 로컬로 실행되거나 다른 시스템(동일하거나 다른 운영체제를 사용하는)에서 실행될 수 있습니다.

이 장에서는 사용자 정의 코드를 WebSphere 개발 환경 외부에서 실행 중인 대상 WebSphere Commerce Server에 전개하는 데 필요한 단계를 설명합니다.

이 장은 사용자 정의 응용프로그램이 포함할 수 있는 다양한 코드 유형을 전개하는 방법에 대해 설명하는 절로 구성됩니다. 다음 태스크에 대해 설명합니다.

- 엔터프라이즈 bean 전개
- 명령 및 데이터 bean 전개
- 상점 자원 전개
- 대상 데이터베이스 갱신

위의 각 태스크에 대해 후속 절에서 더 자세히 설명합니다.

전개 단계를 위한 사용자 권한 요구사항

AIX Solaris Linux WebSphere Commerce 설치 프로세스 준비 시 작성한 루트가 아닌 사용자 ID를 사용하여 대상 WebSphere Commerce Server에서 모든 전개 단계(액세스 제어 갱신 제외)를 수행해야 합니다. 또한 이 사용자가 파일 자원(예: JAR 파일) 및 이 자원을 배치한 디렉토리에 대해 파일 읽기, 쓰기 및 실행 권한이 있는지 확인하십시오.

액세스 제어 갱신 수행에 필요한 사용자 권한에 대한 정보는 *WebSphere Commerce 보안 안내서*의 "데이터베이스에 XML 변경사항 로드"를 참조하십시오.

증분 전개

이 절에서 설명하는 전개 유형은 증분 전개입니다. 증분 전개에서는 대상 WebSphere Commerce 서버에 이미 WebSphere Commerce 엔터프라이즈 응용프로그램을 설치한 상태여야 합니다. 따라서 전개 프로세스에서는 이러한 자원(명령, 데이터 bean 및 엔터프라이즈 bean 등)만 기존 엔터프라이즈 응용프로그램에 전개합니다.

대상 WebSphere Commerce 서버에 초기 엔터프라이즈 응용프로그램을 작성하려면 WebSphere Commerce를 설치한 후 구성 관리자를 사용하여 WebSphere Commerce 인스턴스를 작성해야 합니다(이 다음에 WebSphere Commerce 엔터프라이즈 응용프로그램을 작성하십시오).

엔터프라이즈 bean 전개

이 절에서는 엔터프라이즈 bean을 전개하는 방법에 대해 설명합니다. 이 bean은 전자 상거래 응용프로그램에 대해 작성한 새 엔터프라이즈 bean이거나 수정한 WebSphere Commerce 엔티티 bean일 수 있습니다. 어느 경우에도, 전개 단계는 기본적으로 같습니다.

WebSphere Commerce 응용프로그램의 전개 프로세스를 이해하기 위한 주요 요소 중 하나는 사용자 정의 WebSphere Commerce 코드에 사용하는 패키징 설계를 이해하는 것입니다. 특히, WebSphere Commerce 작업 영역에서는 새 EJB 프로젝트를 작성하지 않아도 됩니다. 새 엔터프라이즈 bean은 WebSphereCommerceServerExtensionsData 프로젝트에 놓이고 수정한 WebSphere Commerce 엔티티 bean의 사용자 정의 코드는 원래의 WebSphere Commerce EJB 프로젝트에 남아 있습니다.

이 전개 프로세스에는 기본적인 두 단계가 있습니다.

- EJB JAR 파일 작성
- 대상 WebSphere Commerce Server에서 EJB JAR 파일 갱신

EJB JAR 파일 작성

다음과 같이 전개 시나리오에 따라, EJB JAR 파일을 작성하는 데는 두 가지 다른 접근 방법이 있습니다.

- 개발 환경과 동일한 유형의 데이터베이스를 사용하고 대상 WebSphere Commerce Server로 전개할 EJB JAR 파일을 작성하는 경우, 235 페이지의 『변환 없이 EJB JAR 파일 작성』의 지시사항을 따르십시오.
- 개발 환경과 다른 유형의 데이터베이스를 사용하고 대상 WebSphere Commerce Server로 전개할 EJB JAR 파일을 작성하는 경우, 236 페이지의 『변환으로 EJB JAR 파일 작성』의 지시사항을 따르십시오.

변환 없이 EJB JAR 파일 작성

EJB JAR 파일을 작성하려면 다음을 수행하십시오.

1. WebSphere Commerce Studio(**시작 > 프로그램 > IBM WebSphere Commerce Studio > WebSphere Commerce Development Environment**)를 열고 J2EE 네비게이터 보기로 전환하십시오.
2. 다음과 같이 전개 중인 bean이 들어 있는 EJB 프로젝트를 펼치십시오.
   - 새 엔터프라이즈 bean을 작성한 경우, **WebSphereCommerceServerExtensionsData** EJB 프로젝트를 펼치십시오.
   - WebSphere Commerce 엔티티 bean을 수정한 경우, 수정한 bean을 포함하는 프로젝트를 펼치십시오. 예를 들어, 사용자 bean을 수정한 경우, **Member-MemberManagementData** EJB 프로젝트를 펼치십시오.
3. **EJB 전개 설명자**를 두 번 누르십시오.
4. 개요 탭을 선택한 상태에서 분할창의 맨 아래로 화면이동하여 **WebSphere 바인딩** 절을 찾으십시오.
5. **DataSource JNDI** 이름 필드에서 대상 WebSphere Commerce Server의 데이터 소스 JNDI 이름을 입력하십시오. 다음은 예제 값입니다.

   DB2 `jdbc/WebSphere Commerce DB2 DataSource demo`

   여기서, 대상 WebSphere Commerce Server는 DB2 데이터베이스를 사용 중이고 WebSphere Commerce 인스턴스 이름은 “demo”입니다.

   Oracle `jdbc/WebSphere Commerce Oracle DataSource demo`

여기서, 대상 WebSphere Commerce Server는 Oracle 데이터베이스를 사용 중이고 WebSphere Commerce 인스턴스 이름은 “demo”입니다.

DataSource JNDI 이름에 대한 값은 “jdbc/”를 대상 WebSphere Commerce Server의 데이터 소스 이름에 추가하여 작성됩니다. 대상 WebSphere Commerce Server에서 *instanceName*.xml 파일을 열고 파일에 있는 DatasourceName=을 검색하여 데이터 소스 이름을 확인할 수 있습니다.

6. 전개 설명자 변경사항을 저장하십시오(ctrl+s).
7. J2EE 네비게이터 보기에서 EJB 프로젝트 (WebSphereCommerceServerExtensionsData 또는 수정한 WebSphere Commerce 엔티티 bean을 포함하는 프로젝트)에서 마우스 오른쪽 버튼으로 누르고 **반출**을 선택하십시오.
   반출 마법사가 열립니다.
8. 반출 마법사에서 다음을 수행하십시오.
   a. **EJB JAR 파일**을 선택하고 **다음**을 누르십시오.
   b. **반출할 자원은?**에 대한 값은 EJB 프로젝트의 이름으로 채워집니다. 이 값은 있는 그대로 두십시오.
   c. **자원 반출 위치** 필드에서 사용할 완전한 JAR 파일 이름을 입력하십시오. 예를 들어, C:\ExportTemp*JarFileName*.jar을 입력하십시오. 여기서, *JarFileName*은 JAR 파일의 이름입니다. 새 엔터프라이즈 bean을 작성한 경우, *yourDir*\WebSphereCommerceServerExtensionsData.jar을 입력해야 합니다. 기존의 WebSphere Commerce 공개 엔티티 bean을 수정한 경우, 이 EJB 그룹에 대해 사전 정의된 JAR 파일 이름을 사용해야 합니다. 예를 들어 Member-MemberManagementData EJB 모듈을 수정한 경우, *yourDir*\Member-MemberManagementData.jar을 입력하십시오.
   d. **소스 파일 반출**이 선택되어 있지 않도록 확인하십시오.
   e. **완료**를 누르십시오.
9. JAR 파일을 작성하였으면 EJB 전개 설명자를 열어 5단계에서 작성된 수정을 로컬 테스트 서버에 필요한 설정으로 복원하십시오. 변경사항을 저장하십시오.

변환으로 EJB JAR 파일 작성

메타데이터를 변환하고 EJB JAR 파일을 작성하려면 다음을 수행하십시오.

1. WebSphere Commerce Studio(**시작 > 프로그램 > IBM WebSphere Commerce Studio > WebSphere Commerce Development Environment**)를 열고 J2EE 네비게이터 보기로 전환하십시오.
2. 다음과 같이 전개 중인 bean이 들어 있는 EJB 프로젝트를 펼치십시오.
   - 새 엔터프라이즈 bean을 작성한 경우, **WebSphereCommerceServerExtensionsData** 프로젝트를 펼치십시오.
   - WebSphere Commerce 엔티티 bean을 수정한 경우, 수정한 bean을 포함하는 프로젝트를 펼치십시오. 예를 들어, 사용자 bean을 수정한 경우, **Member-MemberManagementData** EJB 프로젝트를 펼치십시오.
3. **EJB 전개 설명자**를 두 번 누르십시오.
4. 개요 탭을 선택한 상태에서 분할창의 맨 아래로 화면이동하여 **WebSphere 바인딩** 절을 찾으십시오.
5. **DataSource JNDI** 이름 필드에서 대상 WebSphere Commerce Server의 데이터 소스 JNDI 이름을 입력하십시오. 다음은 예제 값입니다.

   DB2 `jdbc/WebSphere Commerce DB2 DataSource demo`
   여기서, 대상 WebSphere Commerce Server는 DB2 데이터베이스를 사용 중이고 WebSphere Commerce 인스턴스 이름은 “demo”입니다.

   Oracle `jdbc/WebSphere Commerce Oracle DataSource demo`
   여기서, 대상 WebSphere Commerce Server는 Oracle 데이터베이스를 사용 중이고 WebSphere Commerce 인스턴스 이름은 “demo”입니다.

DataSource JNDI 이름에 대한 값은 “jdbc/”를 대상 WebSphere Commerce Server의 데이터 소스 이름에 추가하여 작성됩니다. 대상 WebSphere Commerce Server에서 *instanceName*.xml 파일을 열고 파일에 있는 DatasourceName=을 검색하여 데이터 소스 이름을 확인할 수 있습니다.

6. 전개 설명자 변경사항을 저장하십시오(ctrl+s).
7. WebSphere Studio Application Developer를 닫으십시오.
8. 명령 프롬프트에서 다음 디렉토리로 이동하십시오.

   WCStudio_installdir\Commerce\bin
9. 다음 명령을 입력하십시오.

```
ejbDeploy.bat projName outputJarName
mapFile workspace_dir eclipseDir
ejbDeployXmlFile WCStudio_commercedir
```

여기서,

- *projName*은 변환할 EJB 프로젝트의 이름입니다.
- *outputJarName*은 출력 JAR 파일의 완전한 이름입니다. 이미 WebSphere Commerce 엔터프라이즈 응용프로그램에 있는 JAR 파일 이름을 사용해야 함에 유의하십시오. 다음은 예입니다.

  ```
  C:\ExportTemp\WebSphereCommerceServerExtensionsData.jar
  ```

- *mapFile*은 맵핑 파일의 이름입니다. 이 파일의 예는 다음과 같습니다.

  ```
  WCStudio_installdir\Commerce\properties\com\ibm\commerce\
     metadata\conversion\oracle.mapping
  WCStudio_installdir\Commerce\properties\com\ibm\commerce\
     metadata\conversion\as400.mapping
  ```

- *workspace_dir*은 현재 개발 작업 영역의 디렉토리입니다.
- *eclipseDir*은 eclipse 디렉토리의 경로입니다. 기본값은 다음과 같습니다.

  ```
  WCStudio_installdir\Studio5\eclipse
  ```

- *ejbDeployXmlFile*은 완전한 ejbDeploy.xml 파일입니다. 기본값은 다음과 같습니다.

  ```
  WCStudio_installdir\Commerce\xml\ejbDeploy.xml
  ```

  주: 이 명령을 실행할 때, 일부 파일을 펼칠 수 없었음을 표시하는 오류가 발생할 수도 있습니다. 이것은 문제가 되지 않습니다.

- *WCStudio_commercedir*은 WebSphere Commerce Studio 설치 시 작성되는 Commerce 디렉토리의 경로입니다. 기본값은 다음과 같습니다.

  ```
  WCStudio_installdir\Commerce
  ```

  보다 상세한 표시는 다음과 같습니다.

  ```
  C:\WebSphere\CommerceStudio55\Commerce
  ```

다음은 이 명령을 지정된 모든 값과 함께 사용하는 방법입니다.

```
ejbDeploy.bat WebSphereCommerceServerExtensionsData
WebSphereCommerceServerExtensionsData.jar
C:\WebSphere\CommerceStudio55\Commerce\properties\com\ibm\
```

```
   commerce\metadata\conversion\oracle.mapping
C:\WebSphere\workspace_db2 C:\WebSphere\Studio5\eclipse
C:\WebSphere\CommerceStudio55\Commerce\xml\ejbDeploy.xml
C:\WebSphere\CommerceStudio55\Commerce
```

행 바꾸기는 표시 용도로만 사용됨을 주의하십시오.

10. WebSphere Commerce Studio를 연 다음 EJB 전개 설명자를 열어 5단계에서 작성된 수정을 로컬 데스트 서버에 필요한 설정으로 복원하십시오. 변경사항을 저장하십시오.
11. 전개될 모든 자원을 단일 디렉토리(예: C:\ExportTemp)에 수집하고 있으면 새로 작성된 EJB JAR 파일을 그 디렉토리로 복사하십시오.

대상 WebSphere Commerce Server에서 EJB JAR 파일 갱신

다음 단계는 새로 작성한 EJB JAR 파일을 대상 WebSphere Commerce Server에서 해당 위치에 복사하는 것입니다.

대상 WebSphere Commerce Server에서 EJB JAR 파일을 갱신하려면 다음을 수행하십시오.

1. WebSphere Application Server에서 실행 중인 WebSphere Commerce 인스턴스를 중지하십시오. 이 인스턴스 중지 방법에 대한 자세한 정보는 사용자 플랫폼 및 데이터베이스 *WebSphere Commerce 설치 안내서*를 참조하십시오.
2. WebSphere Commerce 인스턴스에서 EJB JAR 파일 원본을 찾으십시오. 예를 들어, 다음 파일을 찾으십시오.
   - Windows *WAS_installdir*\installedApps*cellName*\ WC_*instance_name*.ear*JarFileName*.jar
   - AIX Solaris Linux *WAS_installdir*/installedApps/*cellName*/WC_*instance_name*.ear/*JarFileName*.jar
   - 400 *WAS_userdir*/installedApps/*WAS_node_name*/WC_*instance_name*.ear/*JarFileName*.jar

   여기서,
   - *instance_name*은 WebSphere Commerce 인스턴스 이름입니다.

- 400 *WAS_node_name*은 WebSphere Application Server 제품이 설치된 iSeries 시스템을 표시합니다.
- *JarFileName*은 사용자 정의 코드가 들어 있는 JAR 파일 이름입니다.

3. 원본 EJB JAR 파일의 사본을 작성하십시오.
4. 개발 시스템의 새 EJB JAR 파일을 2단계의 위치로 복사하십시오.
5. EJB 전개 설명자를 수정한 경우, 다음을 수행하십시오.
   a. 이 WebSphere Application Server 셀의 전개 저장소(META-INF 디렉토리)를 찾으십시오. 일반적인 양식은 다음과 같습니다.
      - Windows *WAS_installdir*\config\cells*cellName*\applications\WC_*instance_name*.ear\deployments\WC_*instance_name**EJBModuleName*.jar\META-INF
      - AIX Solaris Linux *WAS_installdir*/config/ cells/*cellName*/applications/WC_*instance_name*.ear/ deployments/WC_*instance_name*/*EJBModuleName*.jar/META-INF
      - 400 *WAS_userdir*/config/cells/*WAS_node_name*/applications/WC_*instance_name*.ear/deployments/WC_*instance_name*/*EJBModuleName*.jar/META-INF

      여기서,
      - *instance_name*은 WebSphere Commerce 인스턴스 이름입니다.
      - 400 *WAS_node_name*은 WebSphere Application Server 제품이 설치된 iSeries 시스템을 표시합니다.
      - *EJBModuleName*은 수정된 EJB 모듈 이름입니다.

      다음은 플랫폼별 고유한 META-INF 디렉토리의 예입니다.
      - Windows *WAS_installdir*\config\cells\myCell\applications\WC_demo.ear\deployments\WC_demo\Member-MemberManagementData.jar\META-INF

- AIX Solaris Linux *WAS_installdir*/config/cells/myCell/applications/WC_demo.ear/deployments/WC_demo/Member-MemberManagementData.jar/META-INF
- 400 *WAS_userdir*/config/cells/myNode/applications/WC_demo.ear/deployments/WC_demo/Member-MemberManagementData.jar/META-INF

여기서,

- myCell은 WebSphere Application Server 셀 이름이고,
- demo는 WebSphere Commerce 인스턴스 이름이며
- Member-MemberManagementData는 수정된 EJB 모듈 이름입니다.
- 400 myNode는 WebSphere Application Server 노드 이름입니다.

b. 앞의 디렉토리에 있는 모든 파일을 백업하십시오.

c. 도구를 사용하여 *JarFileName*.jar 파일을 열고 내용을 보십시오.

d. *JarFileName*.jar 파일에서 META-INF 디렉토리의 컨텐츠를 5a단계의 디렉토리로 추출하십시오.

6. 사용자 플랫폼 및 데이터베이스 *WebSphere Commerce 설치 안내서*에 설명된 대로 WebSphere Commerce를 다시 시작하십시오.

명령 및 데이터 bean 전개

다음 태스크를 수행하는 경우, 사용자 정의 코드를 WebSphereCommerceServerExtensionsLogic 프로젝트로 패키지화해야 합니다.

- 새 명령 작성
- 새 데이터 bean 작성
- 기존 WebSphere Commerce 명령 수정
- 기존 WebSphere Commerce 데이터 bean 수정

명령 또는 데이터 bean에 대한 기존 클래스(또는 기존 클래스가 들어 있는 프로젝트)를 직접 수정할 수 없습니다.

사용자 정의 명령 및 데이터 bean 전개에는 다음 단계가 포함됩니다.

- JAR 파일 작성
- 대상 WebSphere Commerce Server에서 JAR 파일 갱신

위의 단계에 대해서는 후속 절에서 더 자세히 설명합니다.

사용자 정의 코드를 명령 레지스트리, 새 액세스 제어 정책 또는 기타 데이터베이스 갱신으로 갱신해야 하는 경우, 246 페이지의 『대상 데이터베이스 갱신』을 참조하십시오.

JAR 파일 작성

JAR 파일을 작성하려면 다음을 수행하십시오.

1. WebSphere Commerce Studio(**시작 > 프로그램 > IBM WebSphere Commerce Studio > WebSphere Commerce Development Environment**)를 열고 J2EE 네비게이터 보기로 전환하십시오.
2. **WebSphereCommerceServerExtensionsLogic** 프로젝트에서 마우스 오른쪽 버튼을 누른 후 **반출**을 선택하십시오.
   반출 마법사가 열립니다.
3. 반출 마법사에서 다음을 수행하십시오.
   a. **JAR 파일**을 선택하고 **다음**을 누르십시오.
   b. **반출할 자원 선택**에서 왼쪽 분할창에는 프로젝트 이름을 입력합니다. 이 값은 있는 그대로 두십시오.
   c. **반출할 자원 선택**의 오른쪽 분할창에서 다음 자원만이 선택되었음을 확인하십시오.
      - .classpath
      - .project
      - .serverPreference
   d. **작성된 클래스 파일 및 자원 반출**이 선택되어 있는지 확인하십시오.
   e. Java 소스 파일 및 자원 **반출**을 선택하지 *마십시오*.
   f. **반출 대상 선택** 필드에서 사용할 완전한 JAR 파일 이름을 입력하십시오. 예를 들어,
      `C:\ExportTemp\WebSphereCommerceServerExtensionsLogic.jar`를 입

력하십시오. JAR 파일 이름은 WebSphereCommerceServerExtensionsLogic.jar이어야 합니다.

g. **완료**를 누르십시오.

JAR 파일이 성공적으로 작성되었으면 다음 단계는 대상 WebSphere Commerce Server에서 해당 위치로 파일을 전송하는 것입니다.

대상 WebSphere Commerce Server에서 JAR 파일 갱신

다음 단계는 새로 작성한 JAR 파일을 대상 WebSphere Commerce Server에서 해당 위치에 복사하는 것입니다.

JAR 파일을 갱신하려면 다음을 수행하십시오.

1. WebSphere Application Server에서 실행 중인 WebSphere Commerce 인스턴스를 중지하십시오. 이 인스턴스 중지 방법에 대한 자세한 정보는 사용자 플랫폼 및 데이터베이스 *WebSphere Commerce 설치 안내서*를 참조하십시오.
2. WebSphere Commerce 인스턴스에서 EJB JAR 파일 원본을 찾으십시오. 예를 들어, 다음 파일을 찾으십시오.
   - Windows *WAS_installdir*\installedApps*cellName*\ WC_*instance_name*.ear\WebSphereCommerceServerExtensionsLogic.jar
   - AIX Solaris Linux *WAS_installdir*/installedApps/*cellName*/WC_*instance_name*.ear/WebSphereCommerceServerExtensionsLogic.jar
   - 400 *WAS_userdir*/installedApps/*WAS_node_name*/WC_*instance_name*.ear/WebSphereCommerceServerExtensionsLogic.jar

   여기서,
   - *instance_name*은 WebSphere Commerce 인스턴스 이름입니다.
   - 400 *WAS_node_name*은 WebSphere Application Server 제품이 설치된 iSeries 시스템을 표시합니다.
3. 백업 위치에 원본 JAR 파일의 사본을 작성하십시오.
4. 개발 시스템에서 2단계의 위치로 JAR 파일을 복사하십시오.
5. 사용자 플랫폼 및 데이터베이스 *WebSphere Commerce 설치 안내서*에 설명된 대로 WebSphere Commerce를 다시 시작하십시오.

상점 자원 전개

상점 자원에는 다음과 같은 자원이 포함됩니다.

- JSP 템플리트
- HTML 파일
- 이미지 파일
- XML 파일
- 특성 파일 및 자원 번들

이 자원은 개발 환경에서 대상 WebSphere Commerce Server로 전개해야 합니다. 여기에는 다음과 같은 단계가 포함됩니다.

- WebSphere Studio Application Developer에서 상점 자원 반출
- 대상 WebSphere Commerce Server로 자원 전송

위의 단계에 대해서는 후속 절에서 더 자세히 설명합니다.

상점 자원 반출

개발 환경에서 상점 자원을 반출하려면 다음을 수행하십시오.

1. WebSphere Commerce Studio(**시작 > 프로그램 > IBM WebSphere Commerce Studio > WebSphere Commerce 개발 환경**)을 열고 J2EE 네비게이터 보기로 전환하십시오.
2. **Stores** 폴더를 펼치십시오.
3. **Web Content** 폴더에서 마우스 오른쪽 버튼을 누르고 **반출**을 선택하십시오. 반출 마법사가 열립니다.
4. 반출 마법사에서 다음을 수행하십시오.
   a. **파일 시스템**을 선택하고 **다음**을 누르십시오.
   b. 전개할 모든 자원을 선택하십시오. 즉, 전개할 모든 JSP 템플리트, HTML 파일, 이미지, 특성 파일 및 기타 상점 자원을 선택하십시오.
   c. **선택한 파일의 디렉토리 구조 작성**을 선택하십시오.
   d. **디렉토리** 필드에 이 자원을 배치할 임시 디렉토리를 입력하십시오. 예를 들어, `C:\ExportTemp\StoreAssets`를 입력하십시오.

e. **완료**를 누르십시오.

다음 단계는 대상 WebSphere Commerce Server에서 해당 위치에 이 자원을 복사하는 것입니다.

상점 자원 전송

개발 시스템에서 대상 WebSphere Commerce Server로 상점 자원을 전송하려면 다음을 수행하십시오.

1. 전개 중인 자언의 유형과 특정 구성 세부정보에 따라, WebSphere Application Server에서 실행 중인 WebSphere Commerce 인스터를 중지해야 할 경우가 있습니다. 특정 전개 시나리오를 다시 시작해야할지 여부가 불확실하다면 인스턴스를 중지해야 합니다. 이 인스턴스 중지 방법에 대한 자세한 정보는 사용자 플랫폼 및 데이터베이스 *WebSphere Commerce 설치 안내서*를 참조하십시오.
2. 대상 시스템에서 Stores.war 디렉토리를 찾으십시오. 다음은 이 디렉토리의 한 예입니다.

   - Windows *WAS_installdir*\installedApps*cellName*\ WC_*instance_name*.ear\Stores.war
   - AIX Solaris Linux *WAS_installdir*/installedApps/*cellName*/WC_*instance_name*.ear/Stores.war
   - 400 *WAS_userdir*/installedApps/*WAS_node_name*/WC_*instance_name*.ear/Stores.war

   여기서,

   - *instance_name*은 WebSphere Commerce 인스턴스 이름입니다.
   - 400 *WAS_node_name*은 WebSphere Application Server 제품이 설치된 iSeries 시스템을 표시합니다.
3. "상점 자원 반출"에서 반출된 파일을 Stores.war 디렉토리로 복사하십시오.
4. 이전에 WebSphere Commerce 인스턴스를 중지했다면 인스턴스를 시작하십시오. 이 인스턴스 시작 방법에 대한 자세한 정보는 사용자 플랫폼 및 데이터베이스 *WebSphere Commerce 설치 안내서*를 참조하십시오.

대상 데이터베이스 갱신

사용자의 대상 WebSphere Commerce Server가 개발 시스템이 아닌 다른 데이터베이스를 사용하므로, 대상 WebSphere Commerce Server에서 사용되는 데이터베이스에서 개발 데이터베이스에 수행한 모든 갱신사항을 수행해야 합니다. 여기에는 새 명령이나 수정된 명령 또는 뷰의 등록, 작성된 추가 테이블 그리고 작성된 새 자원에 대한 액세스 제어 정책 작성에 대한 갱신사항이 포함됩니다.

400 실행 중인 SQL문에 대한 유틸리티를 가지고 있어야 합니다. 이를 수행하는 한 가지 방법은 Windows용 IBM iSeries Access를 사용하는 것입니다. 이 유틸리티를 열려면 다음을 수행하십시오.

1. **iSeries 네비게이터**를 여십시오.
2. 작업 탐색기가 열리면 특정 시스템에 사인온해야 합니다. 대상 iSeries 시스템을 선택하고 WebSphere Commerce 인스턴스 사용자 프로파일 작성된 새로운 모든 테이블을 WebSphere Commerce 인스턴스 사용자 프로파일이 소유하는지 확인하십시오.
3. 왼쪽의 패널에서 iSeries 시스템을 펼친 후 **DATABASES**를 펼치십시오. 관계형 데이터베이스를 마우스 오른쪽 버튼으로 누르고 드롭 다운 목록 목록에서 **SQL 스크립트 실행**을 선택하십시오.
   SQL 스크립트 실행 창이 열립니다. 이 창을 사용하면 SQL문을 잘라내서 붙여넣거나 SQL 스크립트를 열 수 있습니다. **연결** 선택의 **JDBC 설정** 옵션을 사용하여 기본 스키마를 설정할 수 있습니다.

액세스 제어 갱신사항

액세스 제어 정보가 데이터베이스에 포함되지만, 이는 개발 환경에서 대상 환경으로 직접 복제하지 않아도 되는 특수 유형의 정보입니다. 특히, 개발 환경에서 프로덕션(또는 테스트 다음 레벨) 환경에 적절하지 않은 매우 자유로운 액세스 제어 정책을 사용하도록 결정할 수 있습니다. 예로, 개발 환경 범위 내에서 새 명령에 관한 정책을 모든 사용자가 명령을 실행할 수는 있는 것으로 설정할 수 있지만, 다른 경우에는 이것이 적절하지는 않을 수 있습니다.

결과적으로 개발 환경에서 대상 환경으로 액세스 제어 정보를 복사하기 전에 새 환경에서 액세스 제어 요구사항을 고려하여 그에 맞게 정책을 조정해야 합니다.

액세스 제어 정책 로드에 대한 정보(여러 플랫폼 및 디렉토리 권한 요구사항에 대한 명령 구문 포함)는 *WebSphere Commerce 보안 안내서*를 참조하십시오.

제 4 부 학습

이 부분에서는 WebSphere Commerce 응용프로그램에 맞게 사용자 정의된 코드 작성과 연관된 여러 태스크에 대해 설명합니다. 개발 관련 단계는 WebSphere Commerce Business Edition 또는 WebSphere Commerce Studio, Professional Developer Edition 개발 환경에서 수행됩니다. 개발 단계는 Windows 2000에서 실행 중인 WebSphere Commerce(사용자의 개발 환경에 따라 Business 또는 Professional Edition)에서 수행됩니다.

제 10 장 학습: 새 비즈니스 로직 작성

이 장에서는 새 비즈니스 로직 작성에 관련된 단계에 대해 배웁니다. 작성한 자원의 유형에는 새 뷰, 새 컨트롤러 명령, 새 태스크 명령, 새 데이터 bean 및 새 엔티티 bean이 포함됩니다. 이 학습은 사용자가 보너스 점수 잔고를 수정하도록 소형 인터페이스를 개발하는 시나리오를 사용합니다. 이것은 단지 데모용이며 로열티 프로그램 응용프로그램을 빌드하는 데 필요한 로직을 반영하지 않습니다. 대신, 이 학습에서 이전에 표시된 각 코드 자산 유형을 작성하는 일반적인 개발 단계를 배우게 됩니다.

이 학습은 다음의 서브태스크로 구성됩니다.

1. 작업 영역 준비
2. 새 뷰 작성
3. 새 컨트롤러 명령 작성
4. 컨트롤러 명령에서 뷰로 정보 전달
5. 컨트롤러 명령에서 URL 매개변수 구문 분석 및 유효성 확인
6. 새 태스크 명령 작성
7. 새 태스크 명령 수정
8. 새 엔터프라이즈 bean 작성
   a. 새 테이블 작성
   b. 새 CMP 엔터프라이즈 bean 작성
   c. 테이블에 새 bean 맵핑 및 스키마 작성
   d. 연관된 액세스 bean 작성
   e. 전개된 코드 작성
   f. 범용 테스트 클라이언트로 bean 테스트
9. Bonus bean을 MyNewControllerCmd에 통합
   a. MyNewTaskCmdImpl 클래스의 performExecute 메소드를 수정하여 새 보너스 점수를 계산하고 XBONUS 테이블에 점수 저장

b. getResources 메소드를 MyNewControllerCmdImpl 클래스에 추가하여 명령이 사용하는 자원 목록 리턴. 이 메소드는 액세스 제어 목적으로 포함됩니다.

c. JSP 템플리트에 보너스 점수를 쉽게 표시할 수 있도록 BonusDataBean 작성

d. 새 자원에 대한 새 액세스 제어 정책 작성

e. 사용자가 보너스 점수를 입력하고 결과를 표시할 수 있도록 MyNewJSPTemplate.jsp 파일 수정

f. 통합 코드 테스트

10. WebSphere Application Server에서 실행 중인 대상 WebSphere Commerce Server에 모든 앞의 코드, 액세스 제어 정책, JSP 템플리트, 이미지 및 자원 번들 전개

각 단계는 후속 절에서 자세히 설명합니다.

견본 코드 찾기

이 학습을 시작하기 전에 이 프로그래밍 학습의 시작점을 포함하는 WC_SAMPLE_55.zip 패키지를 다운로드하십시오. 이 파일을 개발 시스템에 저장하십시오. 예를 들어, *WCStudio_installdir* 디렉토리에 파일을 저장할 수 있습니다.

이 패키지는 다음 웹 사이트의 *WebSphere Commerce 프로그래밍 안내서 및 학습서*에 있습니다.

http://www.ibm.com/software/commerce/library/

작업 영역 준비

이 단계에서는 견본 코드를 WebSphere Commerce 작업 영역에 반입합니다. 이 견본 코드는 학습의 시작 부분입니다.

작업 영역을 준비하려면 다음을 수행하십시오.

1. WebSphere Commerce Studio 버전 5.5를 설치했고 개발 환경 구성을 완료했는지 확인하십시오. 또한 개발 환경 내에서 FashionFlow 견본 상점(고객 직접형 B2C 모델의 예제)을 기반으로 상점을 공개했어야 합니다. 상점 공개 방법에 대한 지시사항은 WebSphere Commerce Production 및 Development 온라인 도움말에 있습니다.
2. 다음을 수행하여 견본 코드를 작업 영역에 반입하십시오.
   a. WebSphere Commerce Studio를 시작하십시오(**시작 > 프로그램 > IBM WebSphere Commerce Studio > WebSphere Commerce Development Environment**).
   b. J2EE Perspective로 전환한 후(**창 > Perspective 열기 > J2EE**) J2EE 네비게이터 보기를 선택하십시오.
   c. **WebSphereCommerceServerExtensionsLogic** 프로젝트를 펼치십시오.
   d. **src** 폴더에서 마우스 오른쪽 버튼을 누른 후 **반입**을 선택하십시오. 가져오기 마법사가 열립니다.
   e. **반입 소스 선택** 목록에서 **Zip 파일**을 선택하고 **다음**을 누르십시오.
   f. **찾아보기(Zip 파일** 필드 옆에 있는)를 누르고 견본 코드를 탐색하십시오. 이 파일은 다음 위치에 지정됩니다.
      yourDirectory\WC_SAMPLE_55.zip
      여기서, *yourDirectory*는 패키지를 다운로드한 디렉토리입니다.
   g. **모두 선택 취소**를 누른 후 디렉토리를 펼쳐 다음 파일을 반입하도록 선택하십시오.
      - com\ibm\commerce\sample\commands\MyNewControllerCmd.java
      - com\ibm\commerce\sample\commands\MyNewControllerCmdImpl.java
      - com\ibm\commerce\sample\commands\MyNewTaskCmd.java
      - com\ibm\commerce\sample\commands\MyNewTaskCmdImpl.java
      - com\ibm\commerce\sample\databeans\MyNewDataBean.java
   h. **폴더** 필드에서는 이미 `WebSphereCommerceServerExtensionsLogic/src` 폴더가 지정되어 있습니다. 이 값을 변경하지 마십시오.
   i. **완료**를 누르십시오.

3. **WebSphereCommerceServerExtensionsLogic** 프로젝트에서 마우스 오른쪽 버튼을 누른 후 **프로젝트 재빌드**를 선택하십시오.
4. 다음을 수행하여 학습용 JSP 템플리트를 해당 디렉토리로 반입하십시오.
   a. J2EE 계층 보기에서 **Stores** 프로젝트를 펼치십시오. 그런 다음 **Web Content > *FashionFlow_name***을 펼치십시오. 여기서, *FashionFlow_name*은 FashionFlow 견본 상점을 기반으로 하는 상점의 이름입니다.

상점을 지금 막 공개한 경우, *FashionFlow_name* 디렉토리가 표시되지 않을 수도 있습니다. 이런 경우, 상점 프로젝트에서 마우스 오른쪽 버튼을 누르고 **최신 정보로 고침**을 선택하십시오. *FashionFlow_name* 디렉토리가 이제 작업 영역에서 사용 가능합니다.

   b. ***FashionFlow_name*** 디렉토리에서 마우스 오른쪽 버튼을 누르고 **반입**을 선택하십시오.
      가져오기 마법사가 열립니다.
   c. **반입 소스 선택** 목록에서 **Zip 파일**을 선택하고 **다음**을 누르십시오.
   d. **찾아보기(Zip 파일** 필드 옆에 있는)를 누르고 견본 코드를 탐색하십시오. 이 파일은 다음 위치에 지정됩니다.
      yourDirectory\WC_SAMPLE_55.zip
      여기서, *yourDirectory*는 패키지를 다운로드한 디렉토리입니다.
   e. **모두 선택 취소**를 누른 후 MyNewJSPTemplate_All.jsp 파일을 선택하십시오.
   f. **폴더** 필드에서는 이미 Stores/Web Content/*FashionFlow_name* 폴더가 지정되어 있습니다. 이 값을 변경하지 마십시오.
   g. **완료**를 누르십시오.
5. 다음을 수행하여 학습용 특성을 해당 디렉토리로 반입하십시오.
   a. J2EE 계층 보기에서 다음 디렉토리를 펼치십시오.
      Stores > Web Content > WEB-INF > classes > *FashionFlow_name*
   b. ***FashionFlow_name*** 디렉토리에서 마우스 오른쪽 버튼을 누르고 **반입**을 선택하십시오.
      가져오기 마법사가 열립니다.
   c. **반입 소스 선택** 목록에서 **Zip 파일**을 선택하고 **다음**을 누르십시오.

d. **찾아보기(Zip 파일** 필드 옆에 있는)를 누르고 견본 코드를 탐색하십시오. 이 파일은 다음 위치에 지정됩니다.
yourDirectory\WC_SAMPLE_55.zip
여기서, *yourDirectory*는 패키지를 다운로드한 디렉토리입니다.

e. **모두 선택 취소**를 누른 후 Tutorial_All_en_US.properties 파일을 선택하십시오.

f. **폴더** 필드에는 이미 `Stores/Web Content/WEB-INF/classes/`*`FashionFlow_name`* 폴더가 지정되어 있습니다. 이 값을 변경하지 마십시오.

g. **완료**를 누르십시오.

6. 학습 과정에 작성하는 새 자원에 해당하는 액세스 제어 정책을 로드하는데 사용되는 네 개의 파일이 있습니다. 파일은 다음과 같습니다.

- MyNewViewACPolicy.xml -- 이 XML 파일에는 새 뷰를 작성할 때 사용되는 액세스 제어 정책이 있습니다.
- MyNewControllerCmdACPolicy.xml-- 이 XML 파일에는 새 컨트롤러 명령을 작성할 때 사용되는 액세스 제어 정책이 있습니다.
- SampleACPolicy_template.xml-- 이 XML 파일에는 새 엔터프라이즈 bean을 작성할 때 사용되는 액세스 제어 정책이 있습니다.
- SampleACPolicy_template_en_US.xml-- 이 XML 파일에는 새 엔터프라이즈 bean을 작성할 때 사용되는 액세스 제어 정책 설명이 있습니다.

다음을 수행하여 앞에 표시된 파일을 해당 디렉토리로 복사하십시오.

a. 파일 시스템에서
yourDirectory\WC_SAMPLE_55.zip 디렉토리로 이동하십시오.

b. Zip 파일을 펼치고 앞에 표시된 네 개의 파일을 다음 디렉토리에 추출하십시오.
WCStudio_installdir\Commerce\xml\policies\xml

7. 다음을 수행하여 학습을 시작할 준비가 되었는지 확인하기 위해 사용자 환경을 테스트하십시오.

a. WebSphere Studio Application Developer에서 서버 perspective로 전환하십시오.

b. Payment Server를 시작하십시오. 로컬 Payment Server를 실행 중인 경우, **WebSphereCommercePaymentsServer** 서버에서 마우스 오른쪽 버튼을 누르고 **시작**(또는 **다시 시작**)을 선택하십시오.

c. **WebSphereCommerceServer**에서 마우스 오른쪽 버튼을 누르고 **시작**(또는 **다시 시작**)을 선택하십시오.

d. WebSphereCommerceServer 서버가 그의 시작 프로세스를 언제 완료했는지 참조하기 위해 콘솔을 살펴보십시오. 서버가 시작되었다면 다음과 유사한 정보를 볼 수 있습니다.

```
[4/2/03 12:56:06:286 EST] 66adf8d1 ApplicationMg A WSVR0221I:
Application started: WebSphereCommerceServer
[4/2/03 12:56:06:777 EST] 66adf8d1 HttpTransport A SRVE0171I:
Transport http is listening on port 9,080.
[4/2/03 12:56:10:742 EST] 66adf8d1 HttpTransport A SRVE0171I:
Transport https is listening on port 9,443.
[4/2/03 12:56:10:762 EST] 66adf8d1 HttpTransport A SRVE0171I:
Transport http is listening on port 8,080.
[4/2/03 12:56:10:762 EST] 66adf8d1 HttpTransport A SRVE0171I:
Transport http is listening on port 80.
[4/2/03 12:56:11:123 EST] 66adf8d1 HttpTransport A SRVE0171I:
Transport https is listening on port 443.
[4/2/03 12:56:11:183 EST] 66adf8d1 HttpTransport A SRVE0171I:
Transport https is listening on port 9,043.
[4/2/03 12:56:11:653 EST] 66adf8d1 RMIConnectorC A ADMC0026I:
RMI Connector available at port 2809
[4/2/03 12:56:12:575 EST] 66adf8d1 WsServer
A WSVR0001I: Server server1 open for e-business
```

e. J2EE 계층 보기에서 **Stores** 프로젝트를 펼치십시오. 그 다음에 **Web Content** > ***FashionFlow_name***를 펼치십시오.

f. **index.jsp** 파일에서 마우스 오른쪽 버튼을 누르고 **서버에서 실행**을 선택하십시오.
   견본 상점이 열립니다.

g. 상품을 선택하고 구입할 수 있는지 확인하십시오.

이제 학습을 진행할 준비가 되었습니다.

새 뷰 작성

이 학습의 첫 번째 단계는 새 뷰를 작성하는 것입니다. 이 새 뷰의 이름은 MyNewView이며 해당되는 JSP 템플리트인 MyNewJSPTemplate.jsp를 갖습니다.

이 절에서는 다음에 대해 배웁니다.

- 특정 상점에 적용하는 JSP 템플리트 및 그래픽 파일을 놓을 위치
- WebSphere Studio Application Developer를 사용하여 JSP 템플리트를 작성하는 방법
- JSP 템플리트의 텍스트를 포함하는 특성 파일 작성 방법
- 뷰 레지스트리(VIEWREG 테이블)를 새 ″MyNewView″로 갱신하는 방법
- 새 뷰의 액세스 제어를 설정하는 방법
- WTE(WebSphere 개발 환경)를 사용하여 새 뷰를 테스트하는 방법

새 뷰 작성에는 다음과 같은 단계가 포함됩니다.

1. 뷰에 이름 지정 후 뷰 레지스트리에 등록
2. JSP 템플리트의 번역 가능 텍스트가 저장될 새 특성 파일 작성
3. 새 뷰의 새 JSP 템플리트 작성
4. 뷰의 액세스 제어 정책 작성 및 로드

MyNewView 등록

이 시나리오에서는 사용자가 작성하는 뷰를 MyNewView라고 합니다. 이 뷰는 명령 레지스트리의 일부인 VIEWREG 테이블에 등록해야 합니다. 새 뷰를 등록하려면 간단한 SQL문을 사용하여 VIEWREG 테이블에 새 항목을 작성하면 됩니다.

이 단계를 진행하기 전에 상점의 고유 식별자를 알아야 합니다. 개발 데이터베이스에 대해 다음의 SQL 조회를 실행하여 이를 판별할 수 있습니다.

```
select STOREENT_ID from STOREENT where IDENTIFIER = 'FashionFlow_name'
```

여기서, *FashionFlow_name*은 상점의 이름입니다. 다음 절에서 필요하므로 이 값을 기록해 두십시오.

DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 MyNewView를 등록하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령행 도구 > 명령 센터**)를 여십시오.
2. **도구** 메뉴에서 **도구 설정**을 선택하십시오.
3. **명령문 종료 문자 사용** 선택란을 선택한 후 지정된 문자가 세미콜론(**;**)인지 확인하십시오.
4. 도구 설정을 닫으십시오.
5. 스크립트 탭을 선택하고 스크립트 창에 다음 정보를 입력하여 VIEWREG 테이블에 필수 항목을 작성하십시오.

```
connect to developmentDB user dbuser using dbpassword;
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
values ('MyNewView', -1, FF_storeent_ID,
   'com.ibm.commerce.command.ForwardViewCommand',
   'com.ibm.commerce.command.HttpForwardViewCommandImpl',
'docname=MyNewJSPTemplate.jsp', 'This is my new view for tutorial one',
0, null)
```

   여기서,

   - *developmentDB*는 개발 데이터베이스의 이름입니다.
   - *dbuser*는 데이터베이스 사용자입니다.
   - *dbpassword*는 데이터베이스 사용자의 암호입니다.
   - *FF_storeent_ID*는 FashionFlow 견본 상점을 기초로 하는 사용자 상점의 고유 식별자입니다.

   실행 아이콘을 누르십시오.

   SQL문이 성공적으로 완료되었음을 나타내는 메시지가 표시되어야 합니다.

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 사용자 뷰를 데이터베이스에 등록하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작 > 프로그램 > Oracle > Application Development > SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.

3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 SQL문을 입력하십시오.

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
values ('MyNewView',-1, FF_storeent_ID,
   'com.ibm.commerce.command.ForwardViewCommand',
   'com.ibm.commerce.command.HttpForwardViewCommandImpl',
   'docname=MyNewJSPTemplate.jsp','This is my new view for tutorial 1',
   0, null);
```

   여기서,

   - *FF_storeent_ID*는 FashionFlow 견본 상점을 기초로 하는 사용자 상점의 고유 식별자입니다.

   SQL문을 실행하려면 Enter를 누르십시오.
6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

   Enter를 눌러 SQL문을 실행하십시오.

이제 MyNewView가 등록되었습니다.

학습용 특성 파일 작성

이 단계에서는 JSP 템플리트에 사용되는 모든 변환 가능 텍스트를 보유하기 위한 새 특성 파일을 작성합니다. 변환 가능 텍스트를 JSP 템플리트 자체와 구분하면 변환 태스크가 단순하게 되고 또한 글로벌 웹 사이트를 갖기 위한 키가 됩니다.

특성 파일을 작성하려면 WebSphere Studio Application Developer에서 다음을 수행하십시오.

1. 웹 perspective를 여십시오(**창 > Perspective 열기 > 웹**).
2. **상점** 웹 프로젝트에서 **Web Content > WEB-INF > 클래스 > *FashionFlow_name*** 폴더를 펼치십시오. Tutorial_All_en_US.properties 파일을 찾을 수 있습니다.
3. **Tutorial_All_en_US.properties**에서 마우스 오른쪽 버튼을 누르고 **열기 프로그램 > 특성 파일 편집기**를 선택하십시오.

4. ***FashionFlow_name*** 폴더에서 마우스 오른쪽 버튼을 누르고 **새로 만들기 > 기타 > 단순 > 파일 > 다음**을 선택하여 새 특성 파일을 작성하십시오.
   새 파일 창이 열립니다.
5. **파일 이름** 필드에 TutorialNLS_en_US.properties를 입력한 후 **완료**를 누르십시오.
   비어 있는 새 파일이 열립니다.
6. Tutorial_All_en_US.properties 파일에서 새 TutorialNLS_en_US.properties 파일로 section 1을 복사하십시오. 그러면 다음의 이름 값 쌍이 TutorialNLS_en_US.properties 파일에 삽입됩니다.

```
# -- SECTION 1 -- #

ProgrammerGuide=Programmer's Guide
Tutorial=Tutorial: Creating new business logic
ParametersFromCmd= List of parameter-value pairs sent from the controller command
CalledByControllerCmd=MyNewView was called by a controller command
CalledByWhichControllerCmd=MyNewView was called by the controller command which is -
ControllerParm1=ControllerParm1=
ControllerParm2=ControllerParm2=
Example=This is an example of using the <if>
tag from JSP Standard Tag Library (JSTL)
UserName=UserName=
Points=Points=
Greeting=Greeting=
UserId=UserId=
FirstInput=Your first input parameter
RegisteredUser=is a registered user
ReferenceNumber=The member refernce number of this user is
NotRegisteredUser=is not a registered user
BonusAdmin=Bonus Administration
PointBeforeUpdate=The bonus point before update is
PointAfterUpdate=The bonus point after update is
EnterPoint=Please enter the points, then submit it to the controller
command

# -- END OF SECTION 1 -- #
```

   값의 행 바꾸기는 표시 목적일 뿐입니다.
7. TutorialNLS_en_US.properties 파일을 저장하십시오(Ctrl+S).

새 TutorialNLS_en_US.properties 파일은 Stores\Web Content\WEB-INF\classes***FashionFlow_name*** 디렉토리에 저장되며 새 JSP 템플리트에서 자원 번들로 사용됩니다.

주: 특성 파일의 내용이 변경될 경우, 테스트에서 변경사항이 표시되기 전에 서버를 다시 시작해야 합니다.

MyNewJSPTemplate 작성

이 단계에서는 WebSphere Studio에서 Page Designer 도구를 사용하여 새 JSP 템플리트를 작성합니다. 특히, MyNewView에서 사용되는 MyNewJSPTemplate.jsp를 작성합니다. 이 템플리트를 작성할 때, 새로운 공백 JSP 템플리트를 작성한 후 또 다른 JSP 템플리트(MyNewJSPTemplate_All.jsp)에서 해당 섹션을 복사합니다.

새 JSP 템플리트를 작성하려면 다음을 수행하십시오.

1. 웹 perspective에서 J2EE 네비게이터 보기로 전환하십시오.
2. **Stores** 웹 프로젝트에서 마우스 오른쪽 버튼을 누른 후 **특성**을 선택하십시오.
3. 왼쪽 분할창에서 **웹**을 선택한 후 사용 가능한 웹 프로젝트 특징 목록에서 **JSP 표준 태그 라이브러리 포함**을 선택하십시오. **적용**을 누르십시오. 갱신이 완료되면 **확인**을 눌러 특성 편집기를 닫으십시오.
4. **Web Content*FashionFlow_name*** 디렉토리를 펼치십시오.
5. **MyNewJSPTemplate_All.jsp** 파일에서 마우스 오른쪽 버튼을 누르고 **열기 프로그램 > Page Designer**를 선택하십시오.
6. ***FashionFlow_name*** 폴더를 펼치고 **새로 만들기 > JSP 파일**을 선택하여 이 폴더에서 새 JSP 템플리트를 작성하십시오.
   새 JSP 파일 창이 열립니다.
7. 다음과 같이 새 파일의 값을 지정하십시오.
   a. **파일 이름** 필드에 MyNewJSPTemplate.jsp를 입력하십시오.
   b. **마크업 언어** 드롭 다운 목록에서 **XHTML**을 선택한 후 **다음**을 누르십시오.
   c. **태그 라이브러리 추가**를 누르십시오.
      태그 라이브러리 선택 창이 열립니다.
   d. 다음의 태그 라이브러리를 선택하십시오.
      - http://java.sun.com/jstl/core
      - http://java.sun.com/jstl/fmt

      확인을 누른 후 **다음**을 누르십시오.
   e. **다음**을 누르십시오.

f. **(워크벤치 기본값 사용) 워크벤치 인코딩** 선택란을 지우십시오.

g. **인코딩** 드롭 다운 목록에서 **ISO Latin -1**을 선택하십시오.

h. **문서 유형** 드롭 다운 목록에서 **XHTML 1.0 변환**을 선택하십시오.

i. **완료**를 누르십시오.

MyNewJSPTemplate.jsp 파일이 열립니다. 파일의 여러 뷰마다 설계, 소스 및 미리보기 탭을 누르십시오.

8. 설계 탭을 선택하고 **여기에 MyNewJSPTemplate.jsp 내용을 배치하십시오.** 텍스트를 누르십시오. 이 텍스트를 `Hello world!`로 바꾸십시오.

9. 소스로 전환한 후 미리보기 탭으로 전환하십시오. 텍스트가 변경되었다는 점에 유의하십시오.

10. 이제 MyNewJSPTemplate_All.jsp 파일에서 새 MyNewJSPTemplate.jsp 파일로 준비 섹션을 복사해야 합니다. 이 섹션은 파일에 작성할 갱신사항의 위치 지정자를 설정합니다. `<%--PREPARATION SECTION`과 `END OF PREPARATION SECTION --%>` 사이의 텍스트를 새 JSP 템플리트에 복사하십시오. 이 텍스트를 JSP 템플리트에 복사할 때 다음 텍스트를 겹쳐쓰십시오.

```
<title> MyNewJSPTemplate.jsp </title>
</head>
<body>
<p> Hello World! </p>
</body>
</html>
```

주: `<%--PREPARATION SECTION` 및 `END OF PREPARATION SECTION --%>`는 새 JSP 템플리트에 복사하지 마십시오. 이 표시 사이에 포함된 텍스트만 복사하십시오.

11. MyNewJSPTemplate_All.jsp 파일에서 새 MyNewJSPTemplate.jsp 파일로 섹션 1A 및 2를 복사하십시오. `<!-- SECTION 1A -->`, `<!-- END OF SECTION 1A -->`, `<!-- SECTION 2 -->` 및 `<!-- END OF SECTION 2 -->` 사이에 새 텍스트를 넣으십시오. 그러면 MyNewJSPTemplate.jsp에 다음 텍스트가 삽입됩니다.

```
<!-- SECTION 1A -->

   <%@ include file="include/EnvironmentSetup.jsp"%>

<!-- END OF SECTION 1A -->
```

```
<!-- SECTION 2 -->

<fmt:setLocale value="${CommandContext.locale}" />
<fmt:setBundle basename="${sdb.directory}/TutorialNLS" var="tutorial" />

<!-- END OF SECTION 2 -->
```

처음 섹션에는 환경 변수를 설정하는 데 사용되는 EnvironmentSetup.jsp 파일이 포함됩니다. 두 번째 섹션은 특성 파일에서 정보를 검색하기 위해 사용되는 자원 번들 오브젝트를 작성하는 데 사용되며 로케일을 설정합니다.

12. 이제 그래픽과 텍스트를 JSP 템플리트에 추가하십시오. 이 단계는 MyNewJSPTemplate_All.jsp 파일에서 MyNewJSPTemplate.jsp 파일로 텍스트를 복사하여 수행됩니다. 이번에는 MyNewJSPTemplate_All.jsp 파일에서 MyNewJSPTemplate.jsp 파일로 섹션 3을 복사하십시오. 그러면 JSP 템플리트에 다음 텍스트가 삽입됩니다.

```
<!-- SECTION 3 -->

<table cellpadding="0" cellspacing="0" border="0">
 <tr>
  <td bgcolor="#ff2d2d" >
   <img src="<c:out value="${sdb.jspStoreDirFilePath}
                           images/male_blueshirt.gif"
                           />" border="0"/>
  </td>
 </tr>
</table>

<h1><fmt:message key="ProgrammerGuide" bundle="${tutorial}"
/> </h1>

<h2><fmt:message key="Tutorial" bundle="${tutorial}"
/> </h2>

<!-- END OF SECTION 3 -->
```

섹션 3은 상점 특정의 이미지 서브폴더(Stores\Web Content*FashionFlow_name*\images)에 있는 이미지를 삽입합니다. 또한 특성 파일에서 텍스트를 검색합니다.

13. MyNewJSPTemplate.jsp 파일 변경사항을 저장하십시오(Ctrl+S).

MyNewView의 액세스 제어 정책 작성 및 로드

새 뷰에 명령 레벨 액세스 제어를 지정해야 합니다. 이 경우, 명령 레벨 액세스 제어 정책은 모든 사용자가 뷰를 실행할 수 있도록 지정합니다. 이 유형의 액세스 제어 정책은 개발 환경에는 적합하지만 다른 상황에는 적합하지 않을 수 있다는 것을 유의하십시오. 고급 액세스 제어 요구사항에 대해서는 *WebSphere Commerce 보안 안내서*를 참조하십시오.

액세스 제어 정책은 MyNewViewACPolicy.xml 파일에서 정의하며 준비 단계의 일부는 다음 디렉토리에 있습니다.
WCStudio_installdir\Commerce\xml\policies\xml

새 정책을 로드하려면 다음을 수행하십시오.

1. 명령 프롬프트에서 다음 디렉토리를 탐색하십시오.
   WCStudio_installdir\Commerce\bin
2. 다음 양식의 acpload 명령을 실행하십시오.

   ```
   acpload db_name db_user db_password inputXMLFile
   ```

   여기서,

   - *db_name*은 개발 데이터베이스의 이름입니다.
   - *db_user*는 데이터베이스 사용자 이름입니다.
   - *password*는 데이터베이스 사용자의 암호입니다.
   - *inputXMLFile*은 액세스 제어 정책 스펙을 포함하는 XML 파일입니다. 이 경우, `MyNewViewACPolicy.xml`을 지정하십시오.

   다음은 변수가 지정된 명령의 예입니다.

   ```
   acpload Demo_Dev db2user db2user MyNewViewACPolicy.xml
   ```

MyNewView 테스트

새 뷰를 작성하는 마지막 단계는 WebSphere 개발 환경에서 새 뷰를 테스트하는 것입니다. 새 뷰를 테스트할 때(그리고 나중에 새 명령을 테스트할 때) 상점의 홈페이지를 먼저 실행해야 한다는 점에 유의하십시오. 상점 실행이 필요한 이유는 새 뷰가 명확히 사용자 상점에 등록되기 때문입니다. 결과적으로 새 뷰에 액세스를 시도하기 전에 먼저 상점 홈페이지를 실행하여 명령 컨텍스트에서 상점 ID 값을 설

정합니다. 마찬가지로 나중에 작성하는 컨트롤러 명령도 사용자 상점에 특별히 등록되며 명령 컨텍스트로부터 상점 ID를 요구합니다.

새 뷰를 테스트하려면 다음을 수행하십시오.

1. WebSphere Studio Application Developer에서 서버 Perspective로 전환하십시오(**창 > Perspective 열기 > 서버**).
2. **WebSphereCommerceServer** 서버에서 마우스 오른쪽 버튼을 누르고 **시작**(또는 **다시 시작**)을 선택하십시오.
3. Stores\Web Content*FashionFlow_name* 디렉토리에 있는 **index.jsp**에서 마우스 오른쪽 버튼을 누르고 **서버에서 실행**을 선택하십시오.
   상점 홈페이지가 웹 브라우저에 표시됩니다.
4. 웹 브라우저에서 다음 URL을 입력하십시오.

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewView
   ```

   몇 초 후에, 다음 화면에 표시된 것처럼 새 JSP 템플리트가 표시됩니다.


Hello World!.
Programmer's Guide
Tutorial: Creating new business logic

그림 31.

새 컨트롤러 명령 작성

이 단계에서는 MyNewControllerCmd라고 하는 새 컨트롤러 명령을 작성합니다. 초기에 이 명령은 MyNewView 뷰만 리턴합니다.

이 절에서는 다음에 대해 배웁니다.

- 컨트롤러 명령에 포함되는 코드의 최소 요구사항
- 새 컨트롤러 명령 인터페이스 및 구현 클래스 작성 방법
- 뷰를 리턴하기 위해 컨트롤러 명령 설정 방법
- 명령 레지스트리에서 컨트롤러 명령을 등록하는 방법
- 컨트롤러 명령의 액세스 제어를 설정하는 방법

새 컨트롤러 명령 작성에는 다음과 같은 단계가 포함됩니다.

1. 명령 레지스트리에 새 명령 등록
2. 명령의 인터페이스 작성
3. 명령의 구현 클래스 작성
4. 명령의 액세스 제어 정책 작성 및 로드
5. 명령 테스트

MyNewControllerCmd 등록

학습의 이 부분에서는 MyNewControllerCmd라고 하는 새 컨트롤러 명령을 작성합니다. 이 명령은 명령 레지스트리에 등록해야 합니다. 특히 인터페이스는 URLREG 테이블에 등록해야 하고, 인터페이스와 해당 구현 클래스 사이의 연관은 CMDREG 테이블에 등록됩니다.

DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 MyNewControllerCmd를 등록하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령행 도구 > 명령 센터**)를 여십시오.
2. 스크립트 탭을 선택하고 스크립트 창에 다음 정보를 입력하여 URLREG 테이블에 필수 항목을 작성하십시오.

```
connect to developmentDB user
dbuser using dbpassword;
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS, DESCRIPTION,
   AUTHENTICATED) values ('MyNewControllerCmd',FF_storeent_ID,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   0, 'This is a new controller command for tutorial one.',null);
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FF_storeent_ID,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   'This is a new controller command for tutorial one.',
   'com.ibm.commerce.sample.commands.MyNewControllerCmdImpl',
   'local');
```

여기서,

- *developmentDB*는 개발 데이터베이스의 이름입니다.
- *dbuser*는 데이터베이스 사용자입니다.
- *dbpassword*는 데이터베이스 사용자의 암호입니다.
- *FF_storeent_ID*는 FashionFlow 견본 상점을 기초로 하는 사용자 상점의 고유 식별자입니다.

실행 아이콘을 누르십시오.

SQL문이 성공적으로 완료되었음을 나타내는 메시지가 표시되어야 합니다.

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 MyNewControllerCmd를 등록하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 SQL문을 입력하십시오.

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS, DESCRIPTION,
   AUTHENTICATED) values ('MyNewControllerCmd',FF_storeent_ID,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   0, 'This is a new controller command for tutorial one.',null);
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FF_storeent_ID,
```

```
'com.ibm.commerce.sample.commands.MyNewControllerCmd',
'This is a new controller command for tutorial one.',
'com.ibm.commerce.sample.commands.MyNewControllerCmdImpl','local');
```

여기서,

- *FF_storeent_ID*는 FashionFlow 견본 상점을 기초로 하는 사용자 상점의 고유 식별자입니다.

SQL문을 실행하려면 Enter를 누르십시오.

6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

Enter를 눌러 SQL문을 실행하십시오.

CMDREG 테이블로의 두 번째 insert 문은 반드시 필요한 것이 아님을 주의하십시오. 이 시나리오에서 인터페이스는 기본 구현을 사용하고 인터페이스와 구현 클래스 사이의 연관은 명령 레지스트리에 실제로 지정할 필요는 없습니다. 여기서,는 완성 목적으로 포함된 것입니다.

MyNewControllerCmd 인터페이스 작성

WebSphere Commerce 프로그래밍 모델에 따라, 모든 새 컨트롤러 명령은 인터페이스와 구현 클래스를 가지고 있어야 합니다. 이 학습의 경우, 견본 코드에 인터페이스 기본이 제공되어 있습니다. 이는 현재 코드에 설명으로 표시되어 여러 개의 섹션으로 분할됩니다. 학습을 진행하면서 코드의 다양한 섹션에 대해 주석을 제거합니다.

MyNewControllerCmd 인터페이스를 작성하려면 다음을 수행하십시오.

1. WebSphere Studio Application Developer에서 Java perspective를 여십시오 (**Window > Perspective 열기 > Java**).
2. **WebSphereCommerceServerExtensionsLogic** 프로젝트를 펼치십시오.
3. **src** 디렉토리를 탐색한 후 **com.ibm.commerce.sample.commands** 패키지를 펼치십시오.
4. **MyNewControllerCmd.java** 인터페이스를 두 번 눌러서 파일을 여십시오.
5. 소스에서 섹션 1 주석을 제거하십시오(섹션 이전의 "/*" 및 섹션 이후의 "*/"를 제거하십시오.). 그러면 다음 코드가 인터페이스에 삽입됩니다.

```
/// Section 1 ////////////////////////////////////////////

// set default command implement class

 static final String defaultCommandClassName =
     "com.ibm.commerce.sample.commands.MyNewControllerCmdImpl";


/// End of section 1/////////////////////////////////////
```

이 코드 섹션은 기본적으로 인터페이스가 MyNewControllerCmdImpl 구현 클래스를 사용해야 함을 지정합니다.

6. 변경사항을 인터페이스에 저장하십시오(Ctrl+S).

MyNewControllerCmdImpl 구현 클래스 작성

인터페이스가 작성되고 나면 다음 단계는 명령의 구현 클래스를 작성하는 것입니다. 이 학습의 경우, 견본 코드에 구현 클래스의 기본이 제공되어 있습니다. 이는 현재 코드에 설명으로 표시되어 여러 개의 섹션으로 분할됩니다. 학습을 진행하면서 코드의 다양한 섹션에 대해 주석을 제거합니다.

MyNewControllerCmdImpl 클래스를 작성하려면 다음을 수행하십시오.

1. **MyNewControllerCmdImpl.java** 클래스를 두 번 눌러서 여십시오.
2. 아웃라인 보기에서 **performExecute** 메소드를 선택하여 해당 소스 코드를 보십시오.
3. performExecute 메소드의 소스 코드에서 Section 1의 주석을 제거하십시오. 메소드에 다음 코드가 삽입됩니다.

```
/// Section 1 ////////////////////////////////////////////

    /// create a new TypedProperties for output purpose.

 TypedProperty rspProp = new TypedProperty();

/// End of section 1 /////////////////////////////////////
```

이것으로 인해 명령의 응답 특성을 보유하기 위해 사용되는 새 TypedProperty 오브젝트가 작성됩니다.

4. performExecute 메소드의 소스 코드에서 Section 5의 주석을 제거하십시오. 메소드에 다음 코드가 삽입됩니다.

```
/// Section 5 ////////////////////////////////////////////////

    /// see how controller command call a JSP

 rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyNewView");
 setResponseProperties( rspProp );

/// End of section 5/////////////////////////////////////////
```

   이 코드 섹션은 두 가지의 기본 태스크를 수행합니다. 먼저, 이는 모든 컨트롤러 명령이 뷰를 리턴하는 WebSphere Commerce 프로그래밍 모델의 요구사항입니다. 이 섹션에서 리턴할 뷰는 이전에 작성한 MyNewView임을 지정합니다. 또한 명령의 응답 특성이 새 rspProp 오브젝트가 되도록 설정합니다.

5. 작업을 저장하십시오(ctrl+s).
6. 코드에 작성한 변경사항을 컴파일하려면 **WebSphereCommerceServerExtensionsLogic** 프로젝트에서 마우스 오른쪽 버튼을 누르고 **프로젝트 빌드**를 선택하십시오.

명령의 액세스 제어 정책 작성 및 로드

새 명령에 명령 레벨 액세스 제어를 지정해야 합니다. 이 경우, 명령 레벨 액세스 제어 정책은 모든 사용자가 명령을 실행할 수 있도록 지정합니다. 이 유형의 액세스 제어 정책은 개발 환경에는 적합하지만 다른 상황에는 적합하지 않을 수 있다는 것을 유의하십시오. 고급 액세스 제어 요구사항에 대해서는 *WebSphere Commerce 보안 안내서*를 참조하십시오.

액세스 제어 정책은 MyNewControllerCmdACPolicy.xml 파일에서 정의하며 준비 단계의 일부로 다음 디렉토리에 둡니다.
WCStudio_installdir\Commerce\xml\policies\xml

새 정책을 로드하려면 다음을 수행하십시오.

1. 명령 프롬프트에서 다음 디렉토리를 탐색하십시오.
   WCStudio_installdir\Commerce\bin
2. 다음 양식의 acpload 명령을 실행해야 합니다.

```
acpload db_name db_user db_password inputXMLFile
```

여기서,

- *db_name*은 개발 데이터베이스의 이름입니다.
- *db_user*는 데이터베이스 사용자 이름입니다.
- *password*는 데이터베이스 사용자의 암호입니다.
- *inputXMLFile*은 액세스 제어 정책 스펙을 포함하는 XML 파일입니다. 이 경우, `MyNewControllerCmdACPolicy.xml`을 지정하십시오.

다음은 변수가 지정된 명령의 예입니다.

```
acpload Demo_Dev db2user db2user MyNewControllerCmdACPolicy.xml
```

MyNewControllerCmd 테스트

이제 인터페이스, 구현 클래스, 명령 등록 및 액세스 제어 정보가 모두 작성되었으므로, 새 컨트롤러 명령을 테스트할 수 있습니다.

Java 코드를 수정한 경우, 변경사항을 인식하려면 먼저 테스트 서버를 다시 시작해야 합니다.

새 코드를 테스트하려면 다음을 수행하십시오.

1. 서버 perspective로 전환하십시오(**창 > Perspective 열기 > 서버**).
2. **WebSphereCommerceServer** 서버에서 마우스 오른쪽 버튼을 누르고 **시작**(또는 **다시 시작**)을 선택하십시오.
3. Stores\Web Content*FashionFlow_name* 디렉토리에 있는 **index.jsp**에서 마우스 오른쪽 버튼을 누르고 **서버에서 실행**을 선택하십시오.
   상점 홈페이지가 웹 브라우저에 표시됩니다.
4. 웹 브라우저에서 다음 URL을 입력하십시오.

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd
   ```

   몇 초 후에, 다음 화면에 표시된 것처럼 새 JSP 템플리트가 표시됩니다.

그림 32.

MyNewControllerCmd에서 MyNewView로 정보 전달

이 단계에서는 정보를 MyNewView에 전달하도록 MyNewControllerCmd를 수정합니다. 정보를 뷰에 전달하는 두 가지 다른 방법이 표시됩니다. 먼저, 응답 특성에 TypedProperties 오브젝트를 사용하는 방법과 이 오브젝트에서 JSP 템플리트로 정보를 추출하는 방법을 학습하게 됩니다. 두 번째로, 정보를 JSP 템플리트에 전달하기 위해 사용하는 새 데이터 bean을 작성하는 방법을 학습합니다.

TypedProperties 오브젝트를 사용하여 정보 전달

이 절에서는 JSP 템플리트에 정보를 전달하도록 MyNewControllerCmdImpl을 수정합니다. 특히, 명령에서 응답 특성에 사용되는 기존 rspProp TypedProperties 오브젝트로 이름 값 쌍을 추가하도록 명령을 수정합니다. JSP 템플리트 내에서 JSTL 표현식 언어를 사용하여 응답 특성에서 정보를 추출합니다.

이 절에서는 다음에 대해 배웁니다.

- 추가 응답 특성을 TypedProperty에 포함하도록 컨트롤러 명령을 수정하는 방법
- JSTL 표현식 언어를 사용하여 응답 특성에서 정보를 검색하도록 JSP 템플리트를 수정하는 방법

JSP 템플리트에서 TypedProperties 오브젝트로부터 정보가 표시될 수 있도록 하려면 다음을 수행하십시오.

1. 첫 번째 단계는 다음과 같이 MyNewControllerCmdImpl 클래스를 수정하는 것입니다.
   a. Java perspective로 전환하십시오.
   b. **WebSphereCommerceServerExtensionsLogic > src > com.ibm.commerce.sample.commands** 디렉토리를 펼치십시오.
   c. **MyNewControllerCmdImpl.java**를 두 번 누르고 아웃라인 보기에서 **performExecute** 메소드를 선택하십시오.
   d. performExecute 메소드의 소스 코드에서 Section 2의 주석을 제거하십시오. 메소드에 다음 코드가 삽입됩니다.

```
/// Section 2 ////////////////////////////////////////////////

    /// see how the controller command pass in variables to JSP

 /// add additional parameters in controller command to rspProp
  ///  for response
    String message1 = "Hello from IBM!";

 rspProp.put("controllerParm1", message1);
 rspProp.put("controllerParm2", "Have a nice day!");

/// End of section 2//////////////////////////////////////////
```

앞의 코드 부분은 응답 특성 오브젝트에 놓이는 두 개의 새 매개변수를 작성합니다. 이 오브젝트는 결국 뷰에 전달됩니다.

e. 변경사항을 저장하십시오.

f. **WebSphereCommerceServerExtensionsLogic** 프로젝트에서 마우스 오른쪽 버튼을 누르고 **프로젝트 빌드**를 선택하여 명령을 컴파일하십시오.

2. 그리고 나서 다음을 수행하여 MyNewJSPTemplate.jsp 파일을 갱신해야 합니다.

   a. JSP 템플리트 파일이 아직 열려 있지 않으면 다음을 수행하십시오.

      1) 웹 perspective에서 J2EE 네비게이터 보기로 전환한 후 **Stores** 웹 프로젝트를 펼치십시오.

      2) **Web Content*FashionFlow_name*** 디렉토리를 탐색하십시오.

      3) **MyNewJSPTemplate_All.jsp**에서 마우스 오른쪽 버튼을 누르고 **열기 프로그램 > Page Designer**를 선택하십시오.

      4) **MyNewJSPTemplate.jsp**에서 마우스 오른쪽 버튼을 누르고 **열기 프로그램 > Page Designer**를 선택하십시오.

   b. MyNewJSPTemplate_All.jsp 파일에서 새 MyNewJSPTemplate.jsp 파일로 섹션 4를 복사하십시오. `<!-- SECTION 4 -->`와 `<!-- END OF SECTION 4 -->` 사이에 새 텍스트를 놓으십시오. 그러면 MyNewJSPTemplate.jsp에 다음 텍스트가 삽입됩니다.

```
<!-- SECTION 4 -->

<h3><fmt:message key="ParametersFromCmd" bundle="${tutorial}"
/> </h3>

<fmt:message key="ControllerParm1"
bundle="${tutorial}" />
<c:out value="${controllerParm1}"
/> <br />

<fmt:message key="ControllerParm2"
bundle="${tutorial}" />
<c:out value="${controllerParm2}"/> <br
/> <br />

<!-- END OF SECTION 4 -->
```

이 섹션은 JSTL 표현식 언어를 사용하여 컨트롤러 명령에서 전달된 값을 확보하여 표시합니다.

c. 변경사항을 저장하십시오.

3. 다음 단계는 다음을 수행하여 컨트롤러 명령과 JSP 템플리트의 수정사항을 테스트하는 것입니다.

   a. 서버 Perspective로 전환하십시오(**창 > Perspective 열기 > 서버**).

   b. **WebSphereCommerceServer** 서버에서 마우스 오른쪽 버튼을 누르고 **시작**(또는 **다시 시작**)을 선택하십시오.

   c. Stores\Web Content*FashionFlow_name* 디렉토리에 있는 **index.jsp**에서 마우스 오른쪽 버튼을 누르고 **서버에서 실행**을 선택하십시오.
   상점 홈페이지가 웹 브라우저에 표시됩니다.

   d. 웹 브라우저에서 다음 URL을 입력하십시오.

      ```
      http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd
      ```

      몇 초 후에, 다음 화면에 표시된 것처럼 새 JSP 템플리트가 표시됩니다.

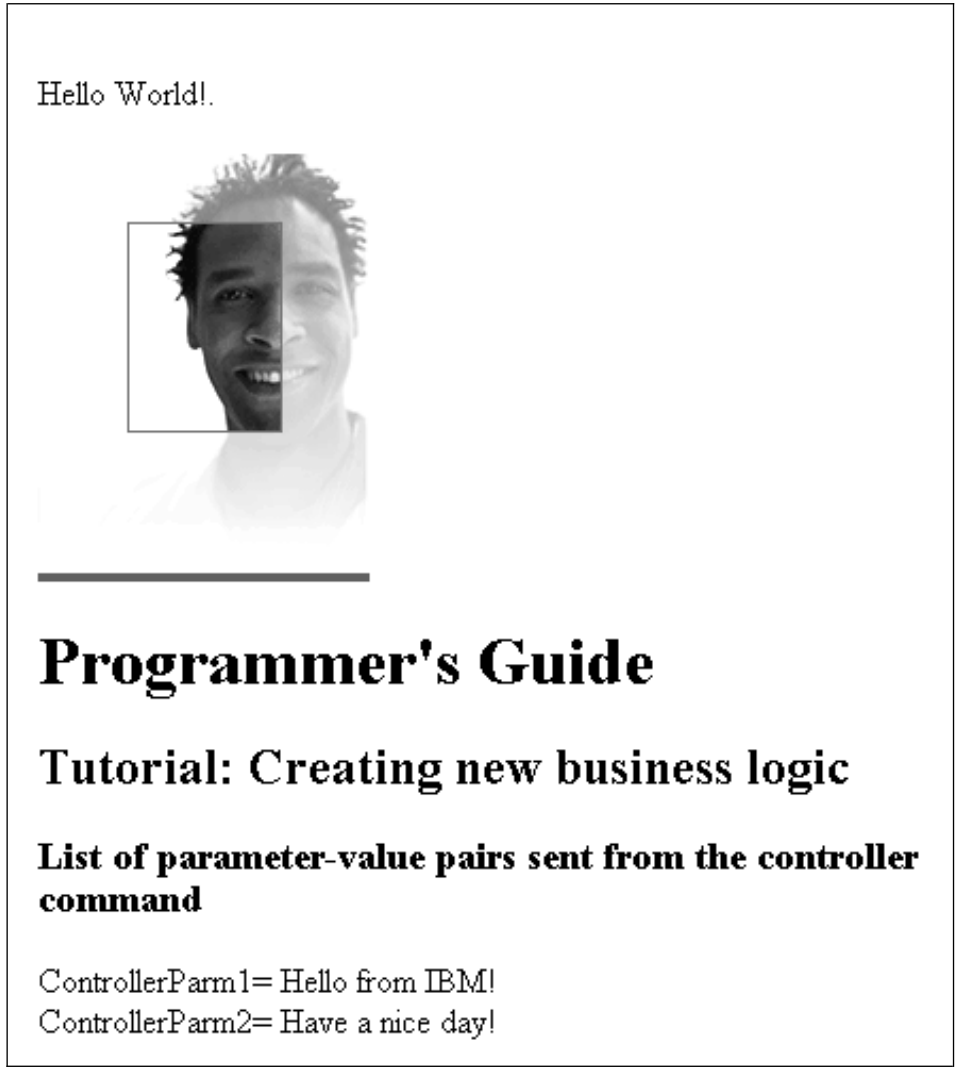


그림 33.

데이터 bean을 사용하여 정보 전달

이 절에서는, 뷰가 직접 호출되었는지 아니면 컨트롤러 명령으로 호출되었는지를 판별하는 코드를 추가합니다. 나중 경우, JSP 템플리트는 호출한 명령의 이름도 표시해야 합니다.

이 정보를 뷰에 전달하기 위해, MyNewDataBean이라고 하는 새 데이터 bean이 작성됩니다. 또한 MyNewJSPTemplate가 새 정보를 표시하도록 수정됩니다.

컨트롤러 명령에서 JSP 템플리트로 정보를 사용 가능하게 하기 위해 MyNewDataBean을 엄격하게 사용합니다. 데이터베이스로부터 정보를 사용 가능하게 하려면 이와 반대로 하십시오. 학습의 나중 부분에서 데이터베이스로부터 JSP 템플리트로 정보를 사용 가능하게 하는 것이 목적인 새 데이터 bean을 작성하는 방법을 학습하게 됩니다.

이 학습에서 컨트롤러 명령으로부터 단지 정보를 사용할 수 있도록 하기 위해 데이터 bean을 사용하는 이유에 대해 궁금할 수도 있습니다. 이 bean을 작성하는 데는 두 가지 이유가 있습니다. 첫째는 이것이 속성의 논리적 그룹화를 허용하는 좋은 프로그래밍 연습이고, 둘째로는 웹 페이지 개발자가 TypedProperties 응답 특성 오브젝트를 사용하는 것보다 데이터 bean을 통해 웹 페이지에 정보를 추가하는 것이 더 간편하다는 것입니다.

이 절에서는 다음에 대해 배웁니다.

- 새 데이터 bean 작성 방법
- 데이터 bean 인스턴스를 작성하기 위해 컨트롤러 명령을 수정하는 방법
- 컨트롤러 명령을 사용하여 데이터 bean에서 속성을 설정하는 방법
- 인스턴스화된 데이터 bean을 JSP 템플리트에 전달하는 방법
- 데이터 bean에서 정보를 검색하도록 JSP 템플리트를 수정하는 방법
- JSP 템플리트에서 <if> 태그를 사용하는 예 보기

MyNewDataBean 작성

MyNewDataBean은 정보를 MyNewJSPTemplate.jsp 페이지에 전달하기 위해 사용됩니다. 학습의 다른 섹션처럼, 견본 코드에 데이터 bean의 기본이 제공되어 있습니다. 이는 현재 코드에 설명으로 표시되어 여러 개의 섹션으로 분할됩니다. 학습을 진행하면서 코드의 다양한 섹션에 대해 주석을 제거합니다.

MyNewDataBean을 작성하려면 다음을 수행하십시오.

1. Java perspective를 열고 패키지 탐색기 뷰를 사용하십시오.
2. **WebSphereCommerceServerExtensionsLogic > src > com.ibm.commerce.sample.databeans** 디렉토리를 펼치십시오.
3. **MyNewDataBean.java**를 두 번 눌러서 소스 코드를 보십시오.

4. 기본 클래스의 소스 코드에서 Section 1의 주석을 제거하십시오. 그러면 다음 코드가 클래스에 삽입됩니다.

```
/// Section 1 ////////////////////////////////////////////
/// create fields and accessors (setter/getter methods)

 private java.lang.String callingCommandName = null;
 private boolean calledByControllerCmd = false;


 public java.lang.String getCallingCommandName() {
  return callingCommandName;
 }

 public void setCallingCommandName(java.lang.String newCallingCommandName) {
  callingCommandName = newCallingCommandName;
 }

 public boolean getCalledByControllerCmd() {
  return calledByControllerCmd;
 }

 public void setCalledByControllerCmd(boolean newCalledByControllerCmd) {
  calledByControllerCmd = newCalledByControllerCmd;
 }

/// End of Section 1 ////////////////////////////////////
```

   앞에 표시된 코드는 뷰의 URL에 의해 직접 호출하기 보다는 컨트롤러 명령에서 뷰를 리턴할 때 정보를 표시하기 위해 사용하는 두 변수를 도입합니다.

5. 변경사항을 저장하십시오.

MyNewControllerCmd를 사용하여 MyNewDataBean 인스턴스 작성 및 속성 설정

이 단계에서는 MyNewControllerCmdImpl을 수정하여 MyNewDataBean 인스턴스를 작성하고 bean 속성을 설정합니다.

MyNewControllerCmdImpl을 수정하려면 다음을 수행하십시오.

1. Java perspective에서 다음 디렉토리를 펼치십시오.
   WebSphereCommerceServerExtensionsLogic > src > com.ibm.commerce.sample.commands.
2. **MyNewControllerCmdImpl.java** 클래스를 두 번 눌러서 소스 코드를 보십시오.

3. 기본 클래스의 소스 코드에서 Import Section 1의 주석을 제거하여 이 클래스에서 새 데이터 bean을 사용할 수 있도록 하십시오. 그러면 다음 코드가 클래스에 삽입됩니다.

```
/// Import Section 1 ////////////////////////////
import com.ibm.commerce.sample.databeans.*;
/// End of Import Section 1 ////////////////////////
```

4. 아웃라인 보기에서 **performExecute** 메소드를 선택하십시오.
5. performExecute 메소드의 소스 코드에서 Section 3A의 주석을 제거하십시오. 메소드에 다음 코드가 삽입됩니다.

```
/// Section 3A/////////////////////////////////////

 /// instantiate the MyNewDataBean databean and set the properties,
  ///  then add the instance to resProp for response

 MyNewDataBean mndb = new MyNewDataBean();
 mndb.setCallingCommandName(this.getClass().getName());
 mndb.setCalledByControllerCmd(true);

/// end of section 3A/////////////////////////////////////

/// Section 3B/////////////////////////////////////
 rspProp.put("mndbInstance", mndb);

/// end of section 3B/////////////////////////////////////
```

   앞에 표시된 코드 부분은 MyNewDataBean 오브젝트 인스턴스를 작성하고, 오브젝트에 두 개의 매개변수(컨트롤러 명령으로 호출하였음을 표시하고 이를 호출한 명령을 표시하는)를 설정한 후 데이터 bean 오브젝트를 JSP 템플리트에 사용 가능하도록 응답 특성에 놓습니다.

6. 변경사항을 저장하십시오.
7. **WebSphereCommerceServerExtensionsLogic** 프로젝트에서 마우스 오른쪽 버튼을 누르고 **프로젝트 빌드**를 선택하십시오.

MyNewJSPTemplate에서 MyNewDataBean 사용

이 섹션에서는 뷰가 컨트롤러 명령에 의해 리턴되었는지 여부를 표시하도록 MyNewJSPTemplate를 수정합니다. 뷰가 컨트롤러 명령에 의해 리턴되었으면 컨트롤러 명령의 이름도 표시해야 합니다. JSP 템플리트는 조건부 로직에 JSTL 태그를 사용하고 MyNewDataBean의 값을 근거로 이를 판별합니다.

JSP 템플리트를 수정하려면 다음을 수행하십시오.

1. JSP 템플리트 파일이 아직 열려 있지 않으면 다음을 수행하십시오.
   a. 웹 perspective에서 J2EE 네비게이터 보기로 전환한 후 **Stores** 웹 프로젝트를 펼치십시오.
   b. **Web Content*FashionFlow_name*** 디렉토리로 이동하십시오.
   c. **MyNewJSPTemplate_All.jsp**와 **MyNewJSPTemplate.jsp** 파일을 둘 다 강조표시하고 마우스 오른쪽 버튼을 누른 후 **열기 프로그램 > Page Designer**를 선택하십시오.
2. MyNewJSPTemplate_All.jsp 파일에서 MyNewJSPTemplate.jsp 파일로 Section 5를 복사하십시오. 그러면 JSP 템플리트에 다음 텍스트가 삽입됩니다.

```
<!-- SECTION 5 -->

<c:if test="${mndbInstance.calledByControllerCmd}">
   <fmt:message key="Example" bundle="${tutorial}" /> <br />
   <fmt:message key="CalledByControllerCmd" bundle="${tutorial}" />
      <br />
   <fmt:message key="CalledByWhichControllerCmd"
   bundle="${tutorial}" />
   <b><c:out value="${mndbInstance.callingCommandName}"
   /></b> <br />
      <br />
</c:if>

<!-- END OF SECTION 5 -->
```

   이 코드 섹션은 JSTL <if> 태그를 사용하여 호출하는 컨트롤러 명령에 대한 정보의 표시 여부를 판별합니다. 또한 학습의 자원 번들에서 변환 가능 텍스트를 검색합니다.
3. 변경사항을 저장하십시오.

수정한 JSP 템플리트 테스트

수정한 JSP 템플리트를 테스트하려면 다음을 수행하십시오.

1. 서버 perspective로 전환하십시오(**창 > Perspective 열기 > 서버**).
2. **WebSphereCommerceServer** 서버에서 마우스 오른쪽 버튼을 누르고 **시작**(또는 **다시 시작**)을 선택하십시오.

3. Stores\Web Content*FashionFlow_name* 디렉토리에 있는 **index.jsp**에서 마우스 오른쪽 버튼을 누르고 **서버에서 실행**을 선택하십시오.
   상점 홈페이지가 웹 브라우저에 표시됩니다.
4. 웹 브라우저에서 다음 URL을 입력하십시오.

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd
   ```

   몇 초 후에 다음 화면에 표시된 대로 JSP 템플리트가 표시됩니다.

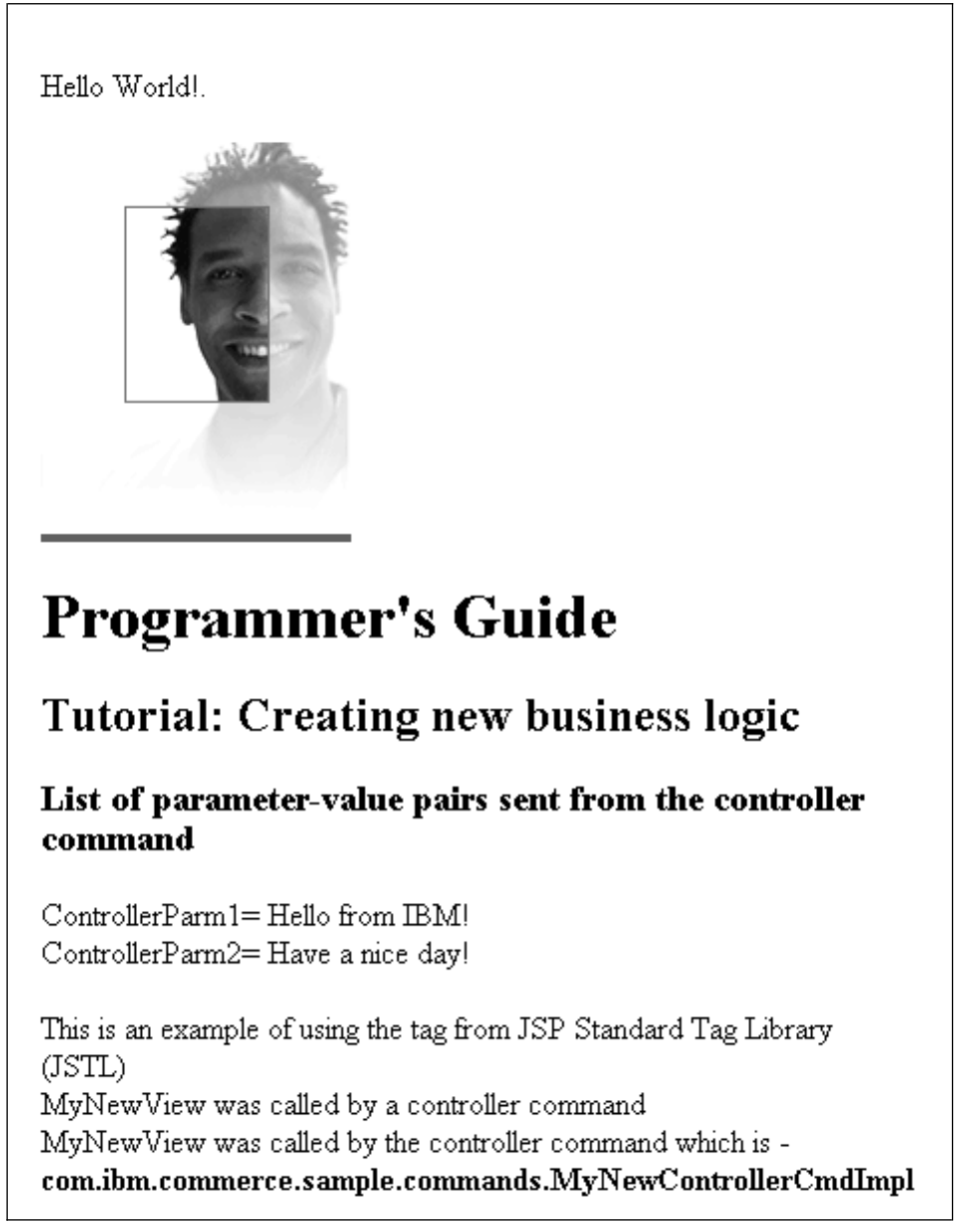


그림 34.

5. 다음 URL을 입력하여 뷰를 직접 호출하고 표시된 정보의 차이점을 알아 보십시오.

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewView
   ```

MyNewControllerCmd에서 URL 매개변수 구문 분석 및 유효성 확인

이 단계에서는 컨트롤러 명령을 호출하는 URL을 통해 전달되는 매개변수를 사용할 수 있도록 컨트롤러 명령을 수정합니다. 필수 매개변수가 포함되고 매개변수에 해당 값을 사용하는지 확인하기 위해 유효성 확인 로직도 명령에 포함됩니다.

현재 명령에 있는 validateParameters 메소드는 실제로는 명령 스텁(stub)일 뿐입니다. 이 메소드는 다음 코드로 구성됩니다.

```
public void validateParameters() throws ECApplicationException {
}
```

이제는 사용자 정의 매개변수 확인을 명령에 추가하고 URL 매개변수를 JSP 템플리트에 전달해야 합니다. validateParameters 메소드를 수정할 때는 URL 매개변수에 해당하는 새 필드를 추가합니다.

이 절에서는 다음에 대해 배웁니다.

- URL 매개변수에 해당하는 새 필드를 명령에 추가하는 방법
- getRequestProperties 메소드를 사용하여 이러한 필드에 URL 입력 매개변수 자료를 대량 반입하는 방법
- 누락 매개변수 예외를 포착하는 방법
- URL 매개변수를 뷰에 전달하는 적절하는 방법
- 누락되거나 올바르지 않은 매개변수 값의 다양한 테스트 경우 예 보기

MyNewControllerCmd에 새 필드 추가

이 단계에서는 URL 매개변수에 해당하는 두 개의 새 필드를 작성합니다. 명령 인터페이스와 구현 클래스 둘 다에 추가됩니다. 한 URL 매개변수는 사용자 이름을 보유하는 문자열 값이 되고, 두 번째는 보너스 점수의 입력 값을 승인하기 위해 사용할 정수가 됩니다.

새 필드를 추가하려면 다음을 수행하십시오.

1. Java perspective로 전환하십시오.
2. **WebSphereCommerceServerExtensionsLogic > src > com.ibm.commerce.sample.commands** 디렉토리를 펼치십시오.

3. **MyNewControllerCmd.java** 인터페이스를 두 번 눌러서 소스 코드를 보십시오.
4. Section 2 주석을 제거하여 새 필드와 해당되는 getter 메소드를 인터페이스에 추가하십시오. 그러면 다음 코드가 클래스에 삽입됩니다.

```
/// Section 2 ////////////////////////////////////////////////

// set interface methods


 public java.lang.Integer getPoints() ;

 public java.lang.String getUserName() ;

 public void setPoints(java.lang.Integer newPoints) ;

 public void setUserName(java.lang.String newUserName)  ;


/// End of section 2//////////////////////////////////////////
```

5. 변경사항을 저장하십시오.
6. **MyNewControllerCmdImpl.java** 클래스를 두 번 눌러서 소스 코드를 보십시오.
7. 기본 클래스에서 Section 1 주석을 제거하여 새 필드와 해당되는 getter 및 setter 메소드를 클래스에 추가하십시오. 그러면 다음 코드가 클래스에 삽입됩니다.

```
/// Section 1 ////////////////////////////////////////////////

/// create and implement controller command's fields and accessors
/// (setter/getter methods)

 private java.lang.String userName = null;
 private java.lang.Integer points;


 public java.lang.Integer getPoints() {
  return points;
 }

 public java.lang.String getUserName() {
  return userName;
 }

 public void setPoints(java.lang.Integer newPoints) {
```

```
 points = newPoints;
}

public void setUserName(java.lang.String newUserName) {
 userName = newUserName;
}

/// End of Section 1 ////////////////////////////////////////
```

8. 변경사항을 저장하십시오.

뷰에 URL 매개변수 전달

이 단계에서는 입력 매개변수를 JSP 템플리트에 전달하기 위해 코드를 포함시킵니다. 이는 입력 매개변수 값으로 데이터 bean의 필드를 설정하여 수행합니다.

URL 매개변수를 전달하려면 다음을 수행하십시오.

1. **MyNewControllerCmdImpl.java**를 두 번 누르십시오.
2. 아웃라인 보기에서 **performExecute** 메소드를 선택하십시오.
3. performExecute 메소드의 소스 코드에서 Section 3C의 주석을 제거하십시오. 메소드에 다음 코드가 삽입됩니다.

   ```
   /// Section 3C////////////////////////////////////////

   // pass the input information to the databean
   mndb.setUserName(this.getUserName());
   mndb.setPoints(this.getPoints());

   /// end of section 3C////////////////////////////////////////
   ```

   이 코드는 데이터 bean 오브젝트의 값이 JSP 템플리트에 사용 가능하게 되도록 설정합니다.
4. 변경사항을 저장하십시오.

누락 매개변수 포착 및 값 유효성 확인

이 단계에서는 오류 확인 및 매개변수 유효성 확인 로직을 삽입하기 위해 validateParameters 메소드를 수정합니다. 수정하면 코드는 다음을 확인합니다.

- 첫 번째 매개변수를 제공하지 않으면 “매개변수를 찾을 수 없음” 예외가 발생합니다. 이는 첫 번째 입력 매개변수가 필수 매개변수이기 때문입니다. 이러한 경우 일반 오류 페이지가 고객에게 표시됩니다.
- 두 번째 입력 매개변수는 선택적입니다. 마찬가지로 두 번째 입력 매개변수를 제공하지 않으면 “매개변수를 찾을 수 없음”이 발생하지 *않습니다*. 대신, 두 번째 입력 매개변수의 값은 기본값 0으로 설정되고 고객에게는 오류가 발생하지 않습니다. 처리는 계속됩니다.

이 오류 확인을 추가하려면 다음을 수행하십시오.

1. **MyNewControllerCmdImpl.java**를 두 번 누르십시오
2. 아웃라인 보기에서 **validateParameters** 메소드를 선택하십시오.
3. validateParameters 메소드의 소스 코드에서 Section 1의 주석을 제거하십시오. 메소드에 다음 코드가 삽입됩니다.

```
/// Section 1 ////////////////////////////////////////////////

/// uncomment to check parameters

    final String strMethodName = "validateParameters";

 TypedProperty prop = getRequestProperties();

 /// retrieve required parameters
 try {
  setUserName(prop.getString("input1"));

 } catch (ParameterNotFoundException e) {
  /// the next exception uses
  _ERR_CMD_MISSING_PARAM ECMessage object
     /// defined in ECMessage class
  throw new ECApplicationException(ECMessage
  ._ERR_CMD_MISSING_PARAM, this.getClass()
  .getName(), strMethodName,
               ECMessageHelper.generateMsgParms
               (e.getParamName()));
 }

 /// retrieve optional Integer
 // set input2 = 0 if no input value
 setPoints(prop.getInteger("input2",0));

/// End of section 1/////////////////////////////////////////////
```

앞에 표시된 코드 부분은 두 입력 매개변수를 확인합니다. try 블록은 첫 번째 매개변수의 존재 여부를 판별하며 없는 경우 예외가 발생합니다. 두 번째 매개변수는 선택적이므로 이 코드는 매개변수가 누락되거나 잘못된 유형인 경우 매개변수의 값을 0으로 설정합니다.

4. 변경사항을 저장하십시오.

MyNewDataBean에 새 필드 추가

이 단계에서는 새 필드와 이에 연관되는 getter 메소드를 MyNewDataBean 데이터 bean에 추가하여 URL 매개변수가 JSP 템플리트에 사용 가능하도록 만듭니다.

MyNewDataBean을 수정하려면 다음을 수행하십시오.

1. **MyNewDataBean.java**를 두 번 눌러서 소스 코드를 보십시오.
2. 클래스에 다음 코드가 삽입되도록 Section 2 주석을 제거하십시오.

```
/// Section 2 ////////////////////////////////////////

 private java.lang.String userName = null;
 private java.lang.Integer points;


 public String getUserName() {
  return userName;
 }

 public void setUserName(java.lang.String newUserName) {
  userName = newUserName;
 }


 public Integer getPoints() {
  return points;
 }

 public void setPoints(java.lang.Integer newPoints) {
  points = newPoints;
 }


/// End of Section 2 ////////////////////////////////////
```

3. 변경사항을 저장하십시오.

4. **WebSphereCommerceServerExtensionsLogic**에서 마우스 오른쪽 버튼을 누르고 **프로젝트 빌드**를 선택하여 코드 변경사항을 컴파일하십시오.

URL 매개변수를 표시하도록 MyNewJSPTemplate 수정

이 단계에서는 다음을 수행하여 URL 입력 매개변수를 표시하는 새 섹션을 추가하기 위해 MyNewJSPTemplate.jsp 파일을 수정합니다.

1. JSP 템플리트 파일이 아직 열려 있지 않으면 다음을 수행하십시오.
   a. 웹 perspective에서 J2EE 네비게이터 보기로 전환한 후 **Stores** 웹 프로젝트를 펼치십시오.
   b. Web Content*FashionFlow_name* 서브폴더를 탐색하십시오.
   c. **MyNewJSPTemplate_All.jsp**와 **MyNewJSPTemplate.jsp** 파일을 둘 다 강조표시하고 마우스 오른쪽 버튼을 누른 후 **열기 프로그램 > Page Designer**를 선택하십시오.
2. MyNewJSPTemplate_All.jsp 파일에서 MyNewJSPTemplate.jsp 파일로 Section 6을 복사하십시오. 그러면 JSP 템플리트에 다음 텍스트가 삽입됩니다.

```
<!-- SECTION 6 -->

<fmt:message key="UserName" bundle="${tutorial}" />
<c:out value="${mndbInstance.userName}"/> <br>

<fmt:message key="Points" bundle="${tutorial}" />
<c:out value="${mndbInstance.points}"/> <br>

<!-- END OF SECTION 6 -->
```

3. 변경사항을 저장하십시오.

URL 매개변수 값 테스트

다음 단계는 다음을 수행하여 새 오류 확인이 제대로 작동하는지 테스트하는 것입니다.

1. 서버 perspective로 전환하십시오(**창 > Perspective 열기 > 서버**).
2. **WebSphereCommerceServer** 서버에서 마우스 오른쪽 버튼을 누르고 **시작**(또는 **다시 시작**)을 선택하십시오.

3. Stores\Web Content*FashionFlow_name* 디렉토리에 있는 **index.jsp**에서 마우스 오른쪽 버튼을 누르고 **서버에서 실행**을 선택하십시오.
   상점 홈페이지가 웹 브라우저에 표시됩니다.
4. 경우 1: 첫 번째 테스트 경우는 두 매개변수 모두를 URL에서 제외하는 경우입니다. 상점 홈페이지가 표시되면 다음 URL을 입력하십시오.

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd
   ```

   명령에 어떤 매개변수도 전달되지 않았으므로, 일반 응용프로그램 오류가 표시됩니다.

그림 35.

WebSphere Studio Application Developer의 콘솔은 다음과 유사한 정보를 표시합니다.

```
timeStamp 6730e546 CommerceSrvr  E com.ibm.commerce.sample.
```

commands.MyNewControllerCmdImpl validateParameters CMN0206E 모든 필드를 확인하십시오. "input1"은 필수 필드입니다.

5. 경우 2: 다음 테스트 경우는 올바른 첫 번째 매개변수를 사용하지만 두 번째 매개변수는 생략합니다. 이러한 경우, 어떤 오류도 검출되지 않을 것으로 예상합니다. 기본적으로 0 값이 누락된 두 번째 매개변수에 사용되기 때문입니다. 다음 URL을 입력하십시오.

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?input1=abc
   ```

   이 명령의 결과는 MyNewJSPTemplate 페이지가 표시되고 input2에 0 값이 표시되는 것입니다.

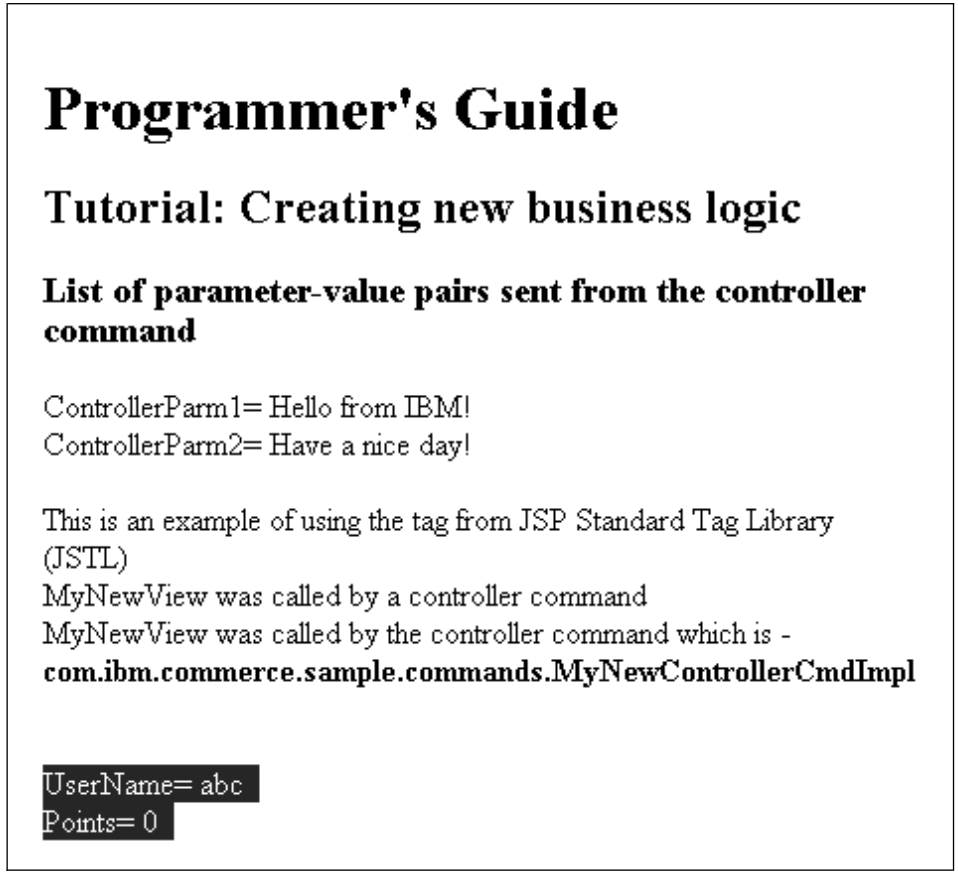


그림 36.

6. 경우 3: 이 테스트 경우에서는 첫 번째 입력 매개변수로 올바른 매개변수가 제공되고, 두 번째 매개변수에 대해서는 올바르지 않은 매개변수가 제공됩니다(정수가 아닌 문자열이 사용됩니다). 마지막 경우처럼, 사용자는 오류를 알아서는 안됩니다. 이는 오류가 발생하면 두 번째 입력 매개변수가 0으로 변경되기 때문입니다. 다음 URL을 입력하십시오.

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
      input1=abc&input2=abc
   ```

   이 명령의 결과는 MyNewJSPTemplate 페이지가 표시되고 input2에 0 값이 표시되는 것입니다.

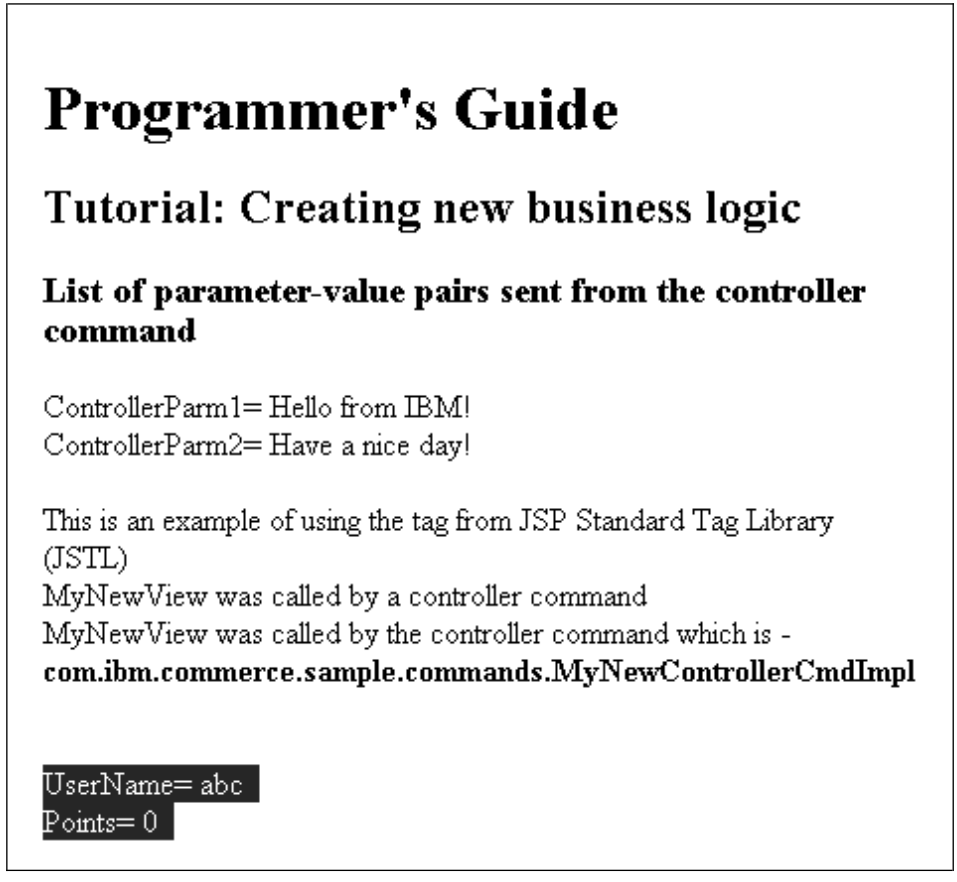


그림 37.

7. 경우 4: 두 URL 입력 매개변수에 올바른 매개변수가 사용되는 경우입니다. 다음 URL을 입력하십시오.

```
http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

이 명령의 결과는 MyNewJSPTemplate 페이지가 표시되고 사용자에게 1000 포인트가 표시되는 것입니다.

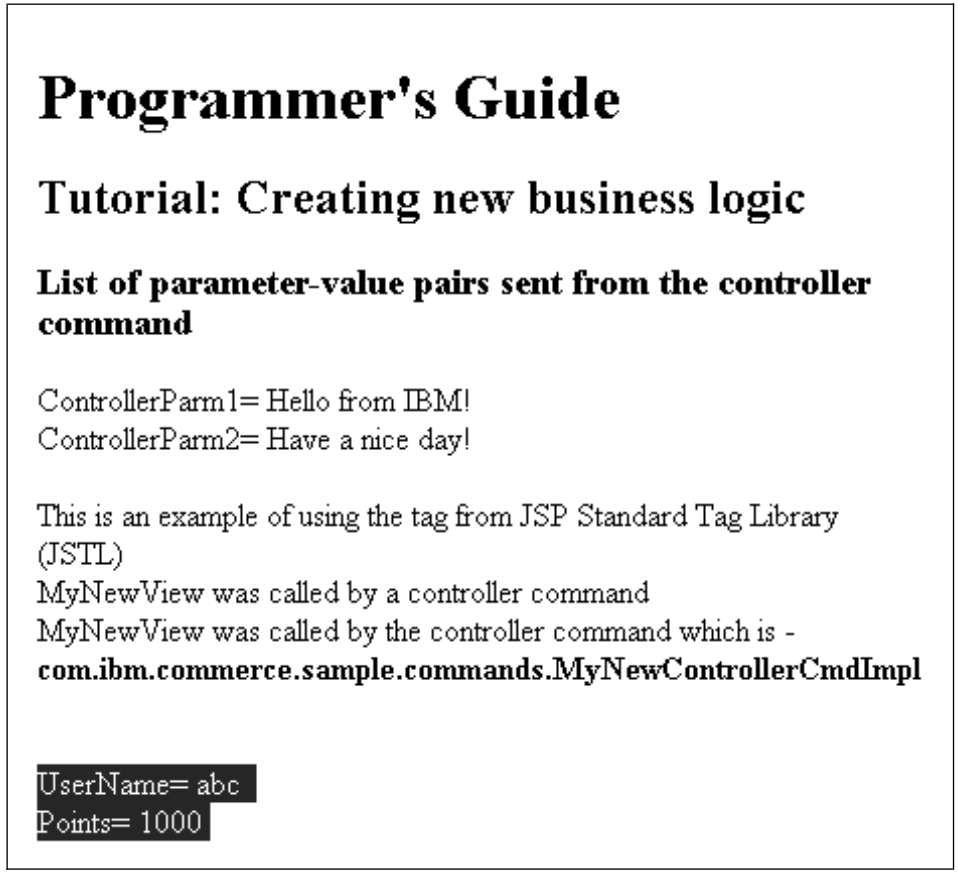


그림 38.

새 태스크 명령 작성

일반적으로 컨트롤러 명령은 비즈니스 처리나 복잡한 기능을 표시합니다. 예를 들어, 주문 처리와 관련된 모든 비즈니스 로직은 OrderProcessCmd 컨트롤러 명령에서 캡슐화됩니다. 비즈니스 프로세스는 종종 보다 작고 더욱 구체적인 태스크로 나눌 수 있습니다. 예를 들어, OrderProcessCmd 컨트롤러 명령 내에는 개별 작업 단위를 수행하기 위해 호출되는 몇 가지 태스크 명령이 있습니다.

MyNewControllerCmdImpl은 현재 어떤 태스크 명령도 호출하지 않습니다. 이 섹션은 두 단계로 나뉩니다. 첫 번째 단계에서 새 태스크 명령을 작성하게 됩니다. 두 번째 단계에서는 컨트롤러 명령의 performExecute 메소드를 수정하여 새 태스크 명령을 호출합니다.

이 단계에서는 새 태스크 명령 인터페이스와 이에 연관되는 구현 클래스를 작성합니다. 초기에 새 태스크 명령은 뷰 매개변수를 처리하는 것을 제외하고 아주 사소한 것을 수행합니다. 이 명령은 defaultCommandClassName, URL 매개변수 및 현재 보너스 점수 값에 해당하는 필드만 가지고 있습니다. 이 명령에는 현재 보너스 점수 값을 가져오기 위한 메소드가 있습니다.

학습의 이 단계에서 다음을 배우게 됩니다.

- 새 태스크 명령 인터페이스와 이에 연관되는 구현 클래스를 작성하는 방법
- 태스크 명령에 필요한 최소한의 코드
- 필드 및 메소드를 태스크 명령에 추가하는 방법

MyNewTaskCmd 작성

이 단계에서는 새 태스크 명령 작성 방법을 표시합니다. 태스크 명령을 작성하려면 인터페이스 및 구현 클래스를 작성해야 합니다. 태스크 명령을 작성할 때 해당 인터페이스는 `com.ibm.commerce.commands.TaskCommand`를 확장해야 합니다. 구현 클래스는 `com.ibm.commerce.command.TaskCommandImpl`을 확장해야 합니다.

이 연습이 완료되면 태스크 새 명령인 MyNewTaskCmd가 작성됩니다. 이 명령은 FashionFlow 견본에 기초하는 상점에서 사용됩니다.

MyNewTaskCmd를 작성하려면 다음을 수행하십시오.

1. Java perspective로 전환한 후 **WebSphereCommerceServerExtensionsLogic** 프로젝트를 선택하십시오.
2. **src** 디렉토리를 펼친 후 **com.ibm.commerce.sample.commands** 디렉토리를 펼치십시오.
3. **MyNewTaskCmd.java** 인터페이스를 두 번 눌러서 소스 코드를 보십시오.
4. 인터페이스의 소스 코드에서 Section 1 주석을 제거하여 인터페이스에서 사용할 기본 구현 클래스를 지정하는 필드를 작성하십시오. 그러면 다음 코드가 인터페이스에 삽입됩니다.

   ```
   /// Section 1 ////////////////////////////////////////////

   // set default command implement class

    static final String defaultCommandClassName=
             "com.ibm.commerce.sample.commands.MyNewTaskCmdImpl";

   /// End of section 1/////////////////////////////////////////
   ```

   전체 사이트에 동일한 구현 클래스가 사용되며 기본 특성이 명령에 전달되지 않으므로 코드에 기본 구현을 바로 지정할 수 있습니다. 복수 구현이 있거나 기본 특성(CMDREG 테이블에 저장)을 가진 명령이 있는 경우, CMDREG 테이블에 명령을 등록하여 인터페이스 및 구현 클래스 간의 맵핑을 작성해야 합니다.
5. 다음은 Section 2 주석을 제거하여 MyNewTaskCmdImpl 구현 클래스에 사용할 getter 및 setter 메소드를 작성하십시오. 이 메소드는 다음 유형의 정보에 해당하는 필드용입니다.
   - 고객의 사용자 ID
   - 보너스 점수 값
   - 인사 메시지

   Section 2의 주석을 제거하면 다음 코드가 인터페이스에 삽입됩니다.

   ```
   /// Section 2 ////////////////////////////////////////////
   // set interface methods

    public void setInputUserName(java.lang.String inputUserName);
    public void setInputPoints(Integer inputPoints);
    public void setGreetings(java.lang.String greeting);
   ```

```
public java.lang.String getInputUserName();
public java.lang.Integer getInputPoints();
public java.lang.String getGreetings();


/// End of section 2/////////////////////////////////////////
```

6. 변경사항을 저장하십시오.
7. **MyNewTaskCmdImpl.java** 구현 클래스를 두 번 눌러 해당 소스 코드를 보십시오.
8. Sections 1A 및 1B의 주석을 제거하여 필드와 해당되는 getter 및 setter 메소드를 구현 클래스에서 작성하십시오. 그러면 다음 코드가 클래스에 삽입됩니다.

```
//// Section 1A ////////////////////////////////////////////

private java.lang.String inputUserName;
private java.lang.String greetings;
private java.lang.Integer inputPoints;


////End of Section 1A ////////////////////////////////////////


//// Section 1B ////////////////////////////////////////////

public void setInputUserName(java.lang.String newInputUserName) {
 inputUserName = newInputUserName;
}

public void setInputPoints(Integer newInputPoints) {
 inputPoints = newInputPoints;
}

public void setGreetings(java.lang.String newGreetings) {
 greetings = newGreetings;
}

public java.lang.String getInputUserName() {
 return inputUserName;
}

public Integer getInputPoints() {
 return inputPoints;
}
```

```
public java.lang.String getGreetings() {
 return greetings;
}


////End of Section 1B ////////////////////////////////////////
```

작업을 저장하십시오.

9. 아웃라인 보기에서 MyNewTaskCmdImpl 클래스의 **performExecute** 메소드를 선택하십시오.
10. performExecute 메소드의 소스 코드에서 Section 1의 주석을 제거하여 메소드로 다음 코드를 삽입하십시오.

```
/// Section 1 ////////////////////////////////////////////////

/// modify the greetings and see it in the NVP list

 setGreetings( "Hello ! " + getInputUserName() );

/// End of section 1 ////////////////////////////////////////
```

이것으로 인해 greetings 값이 갱신됩니다. 이제 greetings 값은 getGreetings() 메소드를 통해 다른 오브젝트에 사용 가능하게 됩니다. 이것은 NVP(이름-값 쌍) 목록에 추가됩니다.

11. 작업을 저장하십시오.

태스크 명령 호출

일단 태스크 명령을 작성하면 컨트롤러 명령으로부터 명령을 호출해야 합니다. 다음 단계에서는 이러한 방식으로 컨트롤러 명령을 수정하는 방법에 대해 설명합니다.

1. Java perspective에서 **MyNewControllerCmdImpl.java** 클래스를 두 번 누르십시오.
2. 아웃라인 보기에서 **performExecute** 메소드를 선택하십시오.
3. performExecute 메소드의 소스 코드에서 Area 4의 주석을 제거하십시오.
4. 메소드에 다음 코드가 삽입되도록 Section 4A, 4B, 4C 주석을 제거하십시오.

```
/// Section 4A
    /// see how the controller command call a task command
```

```
    MyNewTaskCmd cmd = null;

 try {

    cmd = (MyNewTaskCmd) CommandFactory.createCommand(
        "com.ibm.commerce.sample.commands.MyNewTaskCmd", getStoreId());

   // this is required for all commands
         cmd.setCommandContext(getCommandContext());

  /// set input parameters to task command
  cmd.setInputUserName(getUserName());
  cmd.setInputPoints(getPoints()); // change to Integer

/// End Section 4A ////////////////////////////////////

/// Section 4B ////////////////////////////////////

  /// invoke the command's performExecute method
   cmd.execute();

  /// retrieve output parameter from task command, then put it to
     /// response properties
  rspProp.put("taskOutputGreetings", cmd.getGreetings());

/// End Section 4B ////////////////////////////////////

/// Start Section 4C ////////////////////////////////////
 } catch (ECException ex) {
  /// throw the exception as is
         throw (ECException) ex;
 }

/// End Section 4C ////////////////////////////////////
```

Section 4A는 명령 팩토리를 사용하여 새 태스크 명령 오브젝트를 작성합니다. 그런 다음 명령 컨텍스트를 설정하고, 태스크 명령의 입력 매개변수를 설정합니다. Section 4B는 태스크 명령의 execute 메소드를 호출하기 전에 액세스 제어 목적으로 validateParameters 메소드를 호출합니다. 그리고 나서 태스크 명령에서 greetings 값을 검색합니다. Section 4C는 단순한 예외 포착 블록입니다.

5. 변경사항을 저장하십시오.
6. **WebSphereCommerceServerExtensionsLogic** 프로젝트에서 마우스 오른쪽 버튼을 누르고 **프로젝트 빌드**를 선택하여 코드 변경사항을 컴파일하십시오.

greetings 메시지를 추가하기 위해 MyNewJSPTemplate 수정

이 단계에서는 다음을 수행하여 greetings 메시지를 표시하는 새 섹션을 추가하기 위해 MyNewJSPTemplate.jsp 파일을 수정합니다.

1. JSP 템플리트 파일이 아직 열려 있지 않으면 다음을 수행하십시오.
   a. 웹 perspective에서 J2EE 네비게이터 보기로 전환한 후 **Stores** 웹 프로젝트를 펼치십시오.
   b. Web Content*FashionFlow_name* 서브폴더를 탐색하십시오.
   c. **MyNewJSPTemplate_All.jsp**와 **MyNewJSPTemplate.jsp** 파일을 둘 다 강조표시하고 마우스 오른쪽 버튼을 누른 후 **열기 프로그램 > Page Designer**를 선택하십시오.
2. MyNewJSPTemplate_All.jsp 파일에서 MyNewJSPTemplate.jsp 파일로 Section 7을 복사하십시오. 그러면 JSP 템플리트에 다음 텍스트가 삽입됩니다.

```
<!-- SECTION 7 -->

<fmt:message key="Greeting" bundle="${tutorial}" />
<c:out value="${taskOutputGreetings}"/> <br /> <br />

<!-- END OF SECTION 7 -->
```

3. 변경사항을 저장하십시오.

MyNewTaskCmd 테스트

다음 단계는 다음을 수행하여 새 태스크 명령이 제대로 작동하는지 테스트하는 것입니다.

1. 서버 perspective로 전환하십시오(**창 > Perspective 열기 > 서버**).
2. **WebSphereCommerceServer** 서버에서 마우스 오른쪽 버튼을 누르고 **시작**(또는 **다시 시작**)을 선택하십시오.
3. Stores\Web Content*FashionFlow_name* 디렉토리에 있는 **index.jsp**에서 마우스 오른쪽 버튼을 누르고 **서버에서 실행**을 선택하십시오.
   상점 홈페이지가 웹 브라우저에 표시됩니다.
4. 이어서, 다음 URL을 입력하십시오.

```
http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

MyNewJSPTemplate가 표시됩니다. 여기에 태스크 명령이 작성한 greeting 메시지가 포함됩니다.

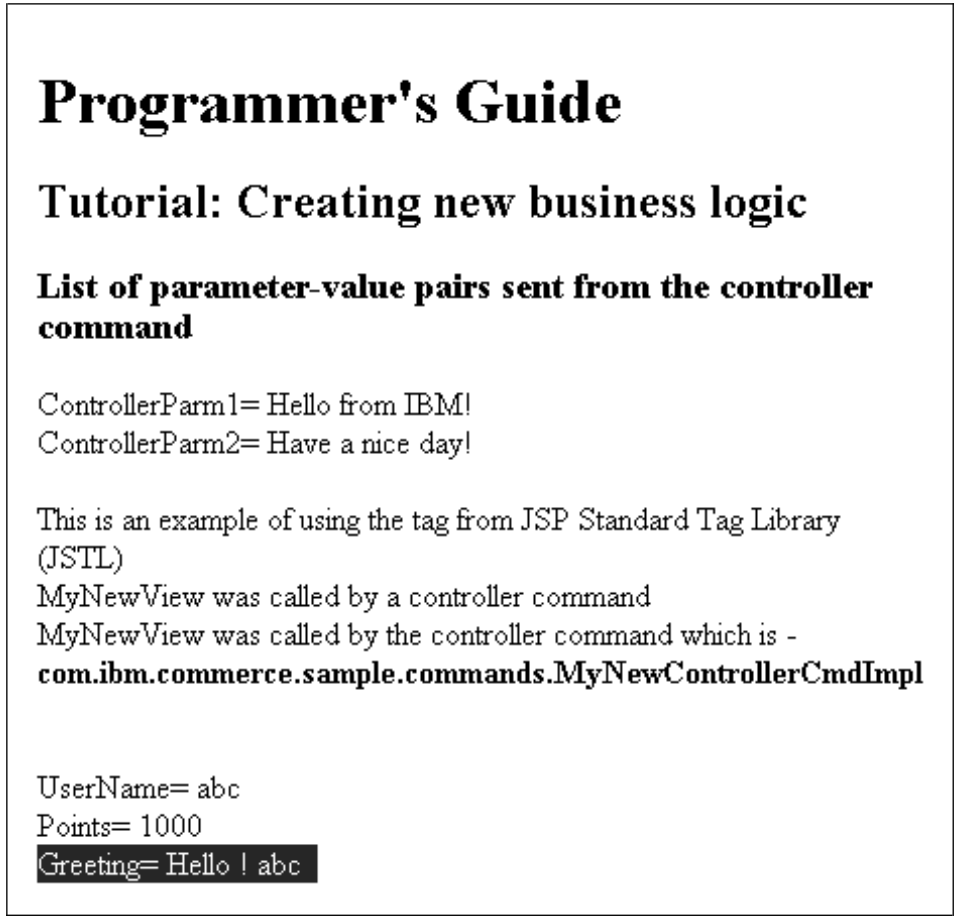


그림 39.

MyNewTaskCmd 수정

이 학습 단계에서는 URL에 포함된 사용자 이름이 등록된 사용자의 이름인지 판별하도록 MyNewTaskCmd를 수정합니다. MyNewDataBean은 또한 태스크 명령에서 필드(예: 사용자 이름)를 처리하도록 수정됩니다. 이에 따라 MyNewJSPTemplate는 사용자의 등록 여부를 표시하도록 수정됩니다.

이 절에서는 다음에 대해 배웁니다.

- 사용자 고유의 사용자 정의 코드에서 URL 매개변수를 사용하고 기존 WebSphere Commerce 정보에 액세스하는 방법

태스크 명령 오브젝트를 작성하도록 MyNewControllerCmdImpl 수정

오브젝트를 보다 효과적으로 사용하기 위해 컨트롤러 명령은 UserRegistryAccessBean 오브젝트 인스턴스 변수를 작성합니다. 결과적으로 이 오브젝트는 태스크 명령에도 사용할 수 있습니다. 이러한 접근 방법을 통해 태스크 명령은 개별 오브젝트 인스턴스를 작성하지 않아도 됩니다. 이 UserRegistryAccessBean 오브젝트는 나중에 구매자가 등록된 사용자인지 여부를 판별하기 위해 태스크 명령에서 사용됩니다.

MyNewControllerCmdImpl을 수정하려면 다음을 수행하십시오.

1. Java perspective에서 **MyNewControllerCmdImpl.java** 클래스를 두 번 눌러 소스 코드를 보십시오.
2. 클래스의 기본 본문에서 Section 2 주석을 제거하여 클래스에 다음 코드를 삽입하십시오.

```
/// Section 2 ////////////////////////////////////////////
/// create a user registry accessbean resource instance variable

    private UserRegistryAccessBean rrb = null;

/// End of Section 2 /////////////////////////////////////////
```

3. 아웃라인 보기에서 **performExecute** 메소드를 선택하십시오.
4. performExecute 메소드의 소스 코드에서 Sections 4D 및 4F의 주석을 제거하여 인스턴스 변수를 태스크 명령에 전달한 후 리턴된 사용자 ID가 응답 특성에서 사용 가능하도록 하십시오. 메소드에 다음 코드가 삽입됩니다.

```
// Section 4D ///////////////////////////////////
/// pass rrb instance variable to the task command

  cmd.setUserRegistryAccessBean(rrb);

// End of section 4D ////////////////////////////


// Section 4F ///////////////////////////////////////
```

```
///using access bean to get information from database
  if (cmd.getFoundUserId() != null) {
   rspProp.put("taskOutputUserId", cmd.getFoundUserId());
  }
// End of section 4F ////////////////////////////
```

일부 메소드가 정의되지 않았음을 표시하는 오류를 수신할 수도 있지만, 이는 태스크 명령을 수정할 때 다음 단계에서 해결됩니다.

5. 작업을 저장하십시오.

사용자 이름 유효성 확인을 위해 새 태스크 명령 수정

다음은 URL의 사용자 이름 입력이 등록된 사용자 이름인지를 확인하기 위해 다음과 같이 새 태스크 명령을 수정해야 합니다.

1. **MyNewTaskCmd.java** 인터페이스를 두 번 눌러서 소스 코드를 보십시오.
2. 인터페이스에 다음 코드가 삽입되도록 Section 3 주석을 제거하십시오.

```
/// Section 3 ////////////////////////////////////////////////

 public void setFoundUserId(java.lang.String inputUserId);
 public java.lang.String getFoundUserId();

 public void setUserRegistryAccessBean(UserRegistryAccessBean rrb);

/// End of section 3/////////////////////////////////////////
```

3. 작업을 저장하십시오.
4. **MyNewTaskCmdImpl.java** 인터페이스를 두 번 눌러서 소스 코드를 보십시오. Import section 1의 주석을 제거하고 다음의 두 import 문을 코드에 삽입하십시오.

```
/// Import section 1 ////////////////////////////////////////
import com.ibm.commerce.user.objects.*;
import com.ibm.commerce.sample.databeans.*;
///  End of Import section 1 ////////////////////////////////
```

5. Sections 2A 및 2B의 주석을 제거하여 인터페이스에 추가된 메소드에 해당하는 getter 및 setter 메소드와 새 필드를 작성하십시오. 그러면 다음 코드가 클래스에 삽입됩니다.

```
//// Section 2A ////////////////////////////////////////////

 private java.lang.String foundUserId = null;
```

```
 private UserRegistryAccessBean rrb = null;

////End of Section 2A ////////////////////////////////////////

//// Section 2B ////////////////////////////////////////////

 public void setUserRegistryAccessBean(UserRegistryAccessBean newRRB) {
  rrb = newRRB;
 }

 public void setFoundUserId(java.lang.String newFoundUserId) {
  foundUserId = newFoundUserId;
 }

 public java.lang.String getFoundUserId() {
  return foundUserId;
 }


/// End of section 2B ////////////////////////////////////////
```

6. 아웃라인 보기에서 **validateParameters** 메소드를 선택하고 해당 소스 코드를 검사하십시오. 메소드에 다음 코드가 삽입되도록 Section 1 주석을 제거하십시오.

```
// section 1 ////////////////////////////////////////////////

// use UserRegistryAccessBean to check user Id

 try {

   if (rrb!=null){
     setFoundUserId(rrb.getUserId());
     } else {
     rrb =new UserRegistryAccessBean();
     rrb=rrb.findByUserLogonId(getInputUserName());
     setFoundUserId(rrb.getUserId());
   }

} catch (javax.ejb.FinderException e) {
    return;

} catch (java.rmi.RemoteException e) {
throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
    this.getClass().getName(), "validateParameters");
} catch (javax.naming.NamingException e) {
throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
    this.getClass().getName(), "validateParameters");
```

```
} catch (javax.ejb.CreateException e) {
throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
    this.getClass().getName(), "validateParameters");
  }

// end of section 1 ////////////////////////////////////////////
```

7. 작업을 저장하십시오.
8. **WebSphereCommerceServerExtensionsLogic** 프로젝트에서 마우스 오른쪽 버튼을 누르고 **프로젝트 빌드**를 선택하여 코드 변경사항을 컴파일하십시오.

사용자 이름 유효성 확인을 위해 MyNewJSPTemplate 수정

사용자 이름 유효성 확인 정보를 표시하도록 현재 JSP 템플리트를 수정해야 합니다. 이 파일을 수정하려면 다음을 수행하십시오.

1. 웹 perspective로 전환하십시오.
2. **MyNewJSPTemplate_All.jsp**와 **MyNewJSPTemplate.jsp** 파일을 모두 여십시오.
3. MyNewJSPTemplate_All.jsp 파일에서 MyNewJSPTemplate.jsp 파일로 Section 8을 복사하십시오. 그러면 JSP 템플리트에 다음 텍스트가 삽입됩니다.

```
<!-- SECTION 8 -->

<c:if test="${!empty taskOutputUserId}">
   <fmt:message key="UserId" bundle="${tutorial}" />
   <c:out value="${taskOutputUserId}"/> <br />
   <fmt:message key="FirstInput" bundle="${tutorial}" />
   <b><c:out value="${userName}"/></b>
   <fmt:message key="RegisteredUser" bundle="${tutorial}" /> <br />
   <fmt:message key="ReferenceNumber" bundle="${tutorial}" />
   <b><c:out value="${taskOutputUserId}"/></b> <br /> <br />
</c:if>

<c:if test="${empty taskOutputUserId}">
   <fmt:message key="FirstInput" bundle="${tutorial}" />
   <b><c:out value="${userName}"/></b>
   <fmt:message key="NotRegisteredUser" bundle="${tutorial}" /> <br />
</c:if>


<!-- END OF SECTION 8 -->
```

4. MyNewJSPTemplate.jsp 파일 변경사항을 저장하십시오.

사용자 이름 유효성 확인 테스트

사용자 이름 유효성 확인 로직을 테스트하려면 다음을 수행하십시오.

1. 서버 perspective로 전환하십시오(**창 > Perspective 열기 > 서버**).
2. **WebSphereCommerceServer** 서버에서 마우스 오른쪽 버튼을 누르고 **시작**(또는 **다시 시작**)을 선택하십시오.
3. Stores\Web Content*FashionFlow_name* 디렉토리에 있는 **index.jsp**에서 마우스 오른쪽 버튼을 누르고 **서버에서 실행**을 선택하십시오.
   상점 홈페이지가 웹 브라우저에 표시됩니다.
4. 다음을 수행하여 새 등록된 사용자를 작성하십시오.
   a. **등록**을 누르십시오.
   b. **등록**을 다시 눌러 새 고객을 작성하십시오.
   c. 등록 양식에서 모든 필수 필드에 해당 값을 입력하십시오. 예를 들어, 전자 우편 필드에 `tester@mycompany`를 입력하십시오. 전자 우편 주소의 값을 쓰십시오: _____.
   d. 값이 입력되면 **제출**을 누르십시오.
5. 다음, input1의 값으로 올바른 사용자 이름을 입력하십시오. 다음 URL을 입력하십시오.

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
      input1=new_e-mail&input2=1000
   ```

   여기서, *new_e-mail*은 4단계에서 작성된 사용자의 전자 우편 주소입니다. MyNewJSPTemplate가 표시됩니다. 이것은 이제 input1의 값이 올바른 사용자 이름임을 표시해야 합니다.

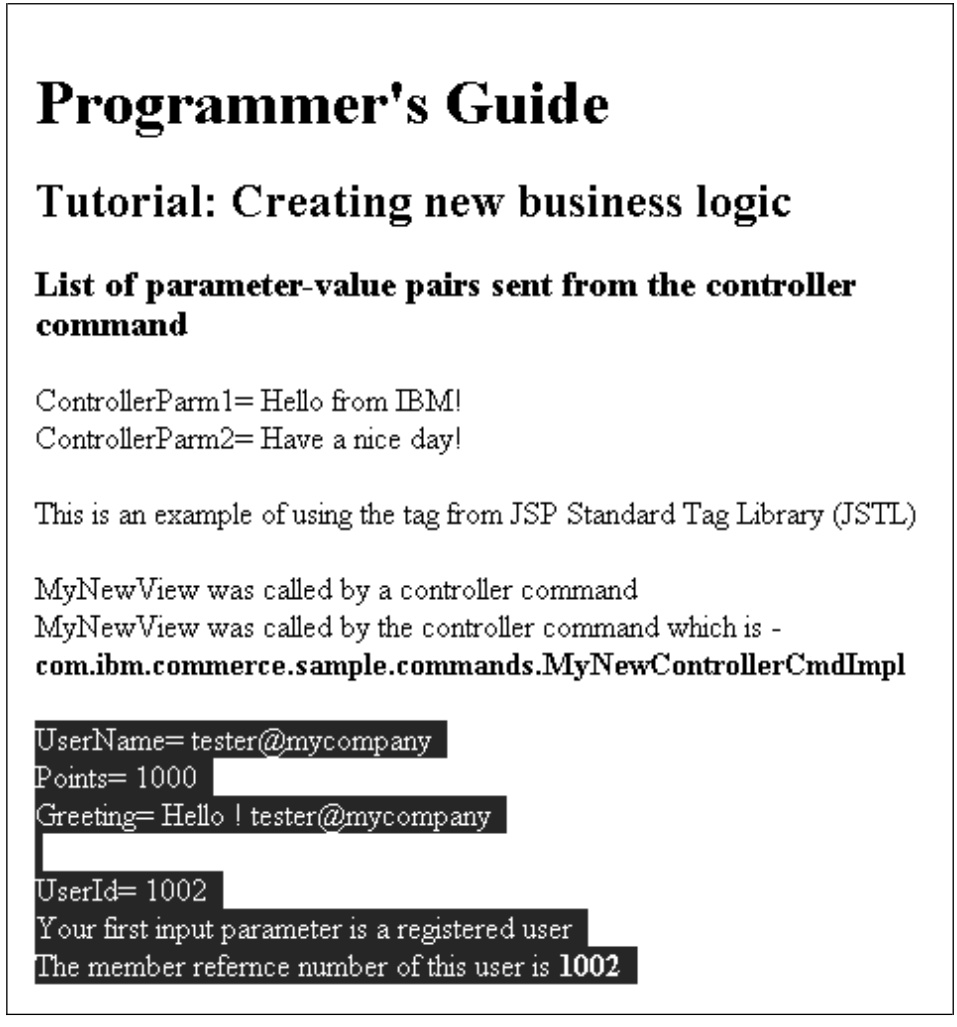


그림 40.

6. 이어서, 사용자 이름의 값이 올바르지 않은 다음 URL을 입력하십시오.

```
http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

일반 예외가 표시됩니다. 페이지의 소스를 보면 _ERR_FINDER_EXCEPTION이 발생한 것을 알 수 있습니다. 이것은 제공된 사용자 이름이 등록된 사용자에 해당하지 않는다는 사실 때문입니다.

새 엔티티 bean 작성

이 절에서는 새 엔티티 bean을 작성하는 방법에 대해 설명합니다. 이 예제 시나리오에는 상거래 응용프로그램의 각 사용자마다 보너스 점수 기록이 점차 올라가는 비즈니스 요구사항이 있습니다. WebSphere Commerce 데이터베이스 스키마에는 정보가 없으므로,WebSphere Commerce 프로그래밍 모델에 따라 일단 데이터베이스 테이블을 작성하면 데이터에 액세스하기 위한 엔티티 bean을 작성해야 합니다

XBONUS 테이블 작성

엔티티 bean 작성 준비 단계에서 우선 데이터베이스 테이블을 작성해야 합니다. 작성할 테이블은 XBONUS입니다.

DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 테이블을 작성하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령행 도구 > 명령 센터**)를 열고 **스크립트** 탭을 누르십시오.
2. 스크립트 창에서 다음을 입력하십시오.

```
connect to developmentDB user dbuser using dbpassword;
create table XBONUS (MEMBERID BIGINT NOT NULL,
    BONUSPOINT INTEGER NOT NULL,
           constraint p_xbonus primary key (MEMBERID),
    constraint f_xbonus foreign key (MEMBERID)
    references users (users_id) on delete cascade)
```

여기서,

- *developmentDB*는 개발 데이터베이스의 이름입니다.
- *dbuser*는 데이터베이스 사용자입니다.
- *dbpassword*는 데이터베이스 사용자의 암호입니다.

실행 아이콘을 누르십시오.

SQL문이 성공적으로 완료되었음을 나타내는 메시지가 표시되어야 합니다.

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 테이블을 작성하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 SQL문을 입력하십시오.

```
create table XBONUS (MEMBERID NUMBER NOT NULL,
    BONUSPOINT INTEGER NOT NULL,
           constraint p_xbonus primary key (MEMBERID),
    constraint f_xbonus foreign key (MEMBERID)
    references users (users_id) on delete cascade);
```

   Enter를 눌러 SQL문을 실행하십시오. XBONUS 테이블이 작성됩니다.

6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

   Enter를 눌러 SQL문을 실행하십시오.

BonusBean 엔티티 bean 작성

테이블이 작성되면 새 엔티티 bean을 작성할 수 있습니다. 다음 단계에서는 WebSphere Studio Application Developer를 사용하여 이 bean을 작성합니다.

다음을 수행하여 새 Bonus bean을 작성합니다.

1. WebSphere Studio Application Developer에서 J2EE perspective로 전환하십시오.
2. J2EE 계층 보기에서 **EJB 모듈**을 펼치십시오.
3. **WebSphereCommerceServerExtensionsData** 모듈에서 마우스 오른쪽 버튼을 누르고 **새로 만들기** > **(기타 >) 엔터프라이즈 bean**을 선택하십시오.
   엔터프라이즈 bean 작성 마법사가 열립니다.
4. **EJB 프로젝트** 드롭 다운 목록에서 **WebSphereCommerceServerExtensionsData**를 선택하고 **다음**을 누르십시오.
5. 엔터프라이즈 bean 작성 창에서 다음을 수행하십시오.

a. **CMP(container-managed persistence) 필드가 있는 엔티티 bean**을 선택하십시오.

b. **Bean 이름** 필드에 Bonus를 입력하십시오.

c. **소스 폴더** 필드에서는 지정된 기본값(ejbModule)을 그대로 두십시오.

d. **기본 패키지** 필드에 com.ibm.commerce.extension.objects를 입력하십시오.

e. **다음**을 누르십시오.

6. 엔터프라이즈 bean 정보 창에서 다음을 수행하십시오.

   a. **추가**를 눌러 BONUS 테이블의 MEMBERID 및 BONUSPOINT 열에 대한 새 CMP 속성을 추가하십시오.
   CMP 속성 작성 창이 열립니다. 이 창에서 다음을 수행하십시오.

      1) **이름** 필드에 memberId를 입력하십시오.

      2) **유형** 필드에 java.lang.Long을 입력하십시오.

         주: *long* 데이터 유형이 아닌 *java.lang.Long* 데이터 유형을 사용해야 합니다.

      3) **키 필드** 선택란을 선택하십시오.

      4) **적용**을 누르십시오.

      5) **이름** 필드에 bonusPoint를 입력하십시오.

      6) **유형** 필드에 java.lang.Integer를 입력하십시오.

         주: *integer* 데이터 유형이 아닌 *java.lang.Integer* 데이터 유형을 사용해야 합니다.

      7) **getter 및 setter 메소드로 액세스** 선택란을 선택하십시오.

      8) **getter 및 setter를 원격 인터페이스로 승격** 선택란을 지우십시오. **getter 읽기 전용으로 만들기** 선택란은 사용할 수 없게 됩니다.

      9) **적용**을 누르십시오.

      10) **닫기**를 눌러 창을 닫으십시오.

   b. **키 클래스에 단일 키 속성 유형 사용** 선택란을 지우고 **다음**을 누르십시오.

7. EJB Java 클래스 정보 창에서 다음을 수행하십시오.

a. Bean의 상위 클래스를 선택하려면 **찾아보기**를 누르십시오.
유형 선택사항 창이 열립니다.

b. **사용하는 클래스 선택: (임의)** 필드에 ECEntityBean을 입력하고 **확인**을 누르십시오. 그러면 상위 클래스로 com.ibm.commerce.base.objects.ECEntityBean이 선택됩니다.

c. **추가**를 눌러서 원격 인터페이스를 확장해야 하는 인터페이스를 지정하십시오. 유형 선택 창이 열립니다.

d. **사용하는 클래스 선택: (임의)** 필드에 Protectable을 입력하고 **확인**을 누르십시오. com.ibm.commerce.security.Protectable이 선택됩니다. 이 인터페이스는 액세스 제어 하에 새 자원을 보호하기 위해 필요합니다.

e. **완료**를 누르십시오.

다음을 수행하여 새 bean의 분리 레벨을 설정하십시오.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼치십시오.
2. **WebSphereCommerceServerExtensionsData** 프로젝트를 두 번 눌러 전개 설명자 편집기에서 여십시오. (다른 방법으로 이 파일을 이 편집기에서 열려면 다음을 수행하십시오.

   a. J2EE 계층 보기에서 **WebSphereCommerceServerExtensionsData** > **ejbModule** > **META-INF**를 펼치십시오.

   b. **ejb-jar.xml**에서 마우스 오른쪽 버튼을 누르고 **열기 프로그램** > **전개 설명자 편집기**를 선택하십시오.)
3. **액세스** 탭을 누르십시오.
4. 분리 레벨 텍스트 상자 옆에 있는 **추가**를 누르십시오.
분리 레벨 추가 창이 열립니다.
5. DB2 **반복 가능 읽기**를 선택한 후 **다음**을 누르십시오.

   Oracle **읽기 확약**을 선택한 후 **다음**을 누르십시오.
6. **발견된 Bean** 목록에서 **Bonus** bean을 선택한 후 **다음**을 누르십시오.
7. **발견된 Bean** 목록에서 **Bonus**를 선택하여 해당되는 모든 메소드를 선택하고 **완료**를 누르십시오.
8. 작업을 저장하고(ctrl+s) 편집기는 열린 상태로 두십시오.

이어서 다음을 수행하여 bean의 보안 동일성을 설정하십시오.

1. 전개 설명자 편집기에서 액세스 탭이 선택되어 있는지 확인하십시오.
2. 보안 동일성 텍스트 상자 옆에 있는 **추가**를 누르십시오.
   보안 동일성 추가 창이 열립니다.
3. **EJB 서버의 동일성 사용**을 선택한 후 **다음**을 누르십시오.
4. **발견된 Bean** 목록에서 **Bonus** bean을 선택한 후 **다음**을 누르십시오.
5. **발견된 Bean** 목록에서 **Bonus**를 선택하여 해당되는 모든 메소드를 선택하고 **완료**를 누르십시오.
6. 작업을 저장하고(ctrl+s) 편집기를 열린 상태로 두십시오.

이번에는 다음을 수행하여 bean의 메소드에 대해 보안 역할을 설정하십시오.

1. 전개 설명자 편집기에서 어셈블리 설명자 탭을 선택하십시오.
2. 메소드 권한 절에서 **추가**를 누르십시오.
3. 보안 역할로 **WCSecurityRole**을 선택하고 **다음**을 누르십시오.
4. 발견된 bean 목록에서 **Bonus**를 선택하고 **다음**을 누르십시오.
5. 메소드 요소 페이지에서 **모두에 적용**을 누른 후 **완료**를 누르십시오.
6. 작업을 저장하고(ctrl+s) 전개 설명자 편집기를 닫으십시오.

다음 단계는 WebSphere Studio Application Developer가 작성한 엔티티 컨텍스트에 관련된 필드 및 메소드 일부를 제거하는 것입니다. 필드를 삭제해야 하는 이유는 ECEntityBean 기본 클래스가 해당 메소드의 구현을 제공하기 때문입니다. 작성된 엔티티 컨텍스트 필드와 메소드를 삭제하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **WebSphereCommerceServerExtensionsData** 프로젝트를 펼치십시오.
2. **Bonus** bean을 펼친 후 **BonusBean** 클래스를 두 번 누르십시오.
3. 아웃라인 보기에서 다음을 수행하십시오.
   a. **myEntityCtx** 필드에서 마우스 오른쪽 버튼을 누르고 **삭제**를 선택하십시오.
   b. **getEntityContext()** 메소드에서 마우스 오른쪽 버튼을 누르고 **삭제**를 선택하십시오.

c. **setEntityContext(EntityContext)** 메소드에서 마우스 오른쪽 버튼을 누르고 **삭제**를 선택하십시오.

d. **unsetEntityContext()** 메소드에서 마우스 오른쪽 버튼을 누르고 **삭제**를 선택하십시오.

4. 작업을 저장하십시오(Ctrl+S). BonusBean 클래스를 열린 상태로 두십시오.

다음을 수행하여 새 getMemberId 메소드를 엔터프라이즈 bean에 추가하십시오.

1. **BonusBean** 클래스의 소스 코드를 보십시오.
2. 다음 코드를 이 클래스 끝(클래스 내에서)에 추가하십시오.

```
public java.lang.Long getMemberId() {
   return memberId;
}
```

3. 다음을 수행하여 새 메소드를 원격 인터페이스에 추가해야 합니다.

   a. 아웃라인 보기에서 **getMemberId** 메소드를 마우스 오른쪽 버튼으로 누르고 **엔터프라이즈 bean > 원격 인터페이스로 승격**을 선택하십시오. 이 작업이 완료되면 메소드 옆에 원격 인터페이스로 승격되었음을 표시하는 작은 R 아이콘이 표시됩니다.

4. 작업을 저장하십시오.
5. BonusBean 편집기를 닫으십시오.

다음을 수행하여 BonusHome 인터페이스에 새 FinderHelper 메소드를 추가하십시오.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼치십시오.
2. **WebSphereCommerceServerExtensionsData** 프로젝트를 두 번 누르고 EJB 전개 설명자 편집기를 여십시오.
3. **Bean** 탭을 누르십시오.
4. Beans 분할창에서 **Bonus** bean을 선택한 후 오른쪽에 있는 분할창에서 아래로 화면이동하여 **WebSphere Extensions**를 펼치십시오.
5. **파인더** 텍스트 상자 옆에 있는 **추가**를 누르십시오.
   파인더 설명자 추가 창이 열립니다.
6. **새로 만들기**를 선택한 후 **이름** 필드에서 `findByMemberId`를 입력하십시오.

7. **매개변수** 텍스트 상자 옆에 있는 **추가**를 누르고 다음을 수행하십시오.
   a. **이름** 필드에 memberId를 입력하십시오.
   b. **유형** 필드에 java.lang.Long을 입력하십시오.
   c. **확인**을 누르십시오.
8. **리턴 유형** 드롭 다운 목록에서 **com.ibm.commerce.extension.objects.Bonus**를 선택한 후 **다음**을 누르십시오.
9. **파인더 유형** 드롭 다운 목록에서 **WhereClauseFinderDescriptor**를 선택하십시오.
10. **파인더 명령문** 필드에서 T1.MEMBERID = ?를 입력한 후 **완료**를 누르십시오.
11. 작업을 저장하고 EJB 전개 설명자 편집기를 닫으십시오.

다음을 수행하여 새 ejbCreate 메소드를 Bonus bean에 추가하십시오.

1. J2EE 계층 보기에서 **BonusBean** 클래스를 두 번 눌러서 열고 소스 코드를 보십시오.
2. 다음 코드를 클래스에 추가하여 새 ejbCreate(Long, Integer) 메소드를 작성하십시오.

```
public com.ibm.commerce.extension.objects.BonusKey ejbCreate(
   java.lang.Long memberId,java.lang.Integer bonusPoint)
   throws javax.ejb.CreateException {
   _initLinks();
      this.memberId=memberId;
      this.bonusPoint=bonusPoint;
return null;
}
```

3. 코드 변경사항을 저장하십시오.
4. 홈 인터페이스에 새 ejbCreate(Long, Integer) 메소드를 추가해야 합니다. 그러면 메소드가 작성된 액세스 bean에서 사용할 수 있게 됩니다. 홈 인터페이스에 메소드를 추가하려면 다음을 수행하십시오.
   a. 아웃라인 보기에서 **ejbCreate(Long, Integer)** 메소드를 마우스 오른쪽 버튼으로 누르고 **엔터프라이즈 bean > 홈 인터페이스로 승격**을 선택하십시오.

다음을 수행하여, ejbCreate(Long, Integer) 메소드와 같은 매개변수를 갖도록 새 ejbPostCreate(Long, Integer) 메소드를 작성하십시오.

1. **BonusBean** 클래스를 두 번 눌러서 열고 해당 소스 코드를 보십시오.
2. 다음 코드를 클래스에 추가하여 새 ejbPostCreate(Long, Integer) 메소드를 작성하십시오.

```
public void ejbPostCreate(java.lang.Long memberId,
   java.lang.Integer bonusPoint)
   throws javax.ejb.CreateException
   {
   }
```

3. 코드 변경사항을 저장하십시오.

다음 단계에서는 WebSphere Commerce 액세스 제어 시스템에 의해 보호될 수 있도록 Bonus bean에 새 메소드를 추가합니다. 다음을 수행하여 getOwner 및 fulfills 메소드를 bean에 추가해야 합니다.

1. **BonusBean** 클래스를 두 번 눌러서 열고 해당 소스 코드를 보십시오.
2. 다음 코드를 클래스 끝에 추가하여 새 getOwner 메소드를 BonusBean 클래스에 추가하십시오.

```
public java.lang.Long getOwner()
   throws java.lang.Exception {
      return getMemberId();
   }
```

3. 작업을 저장하십시오.
4. 다음 코드를 클래스 끝에 추가하여 새 fulfills 메소드를 추가하십시오.

```
public boolean fulfills(Long member, String relationship)
   throws java.lang.Exception {
      if (relationship.equalsIgnoreCase("creator")) {
         return member.equals(getMemberId());
      }
      return false;
   }
```

5. 작업을 저장하고 bonus bean 편집기를 닫으십시오.

다음 단계는 XBONUS 테이블을 BonusBean 엔티티 bean에 맵핑하는 것입니다. Meet-in-the-middle 맵핑이 사용됩니다. 맵핑을 작성하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **WebSphereCommerceServerExtensionsData**에서 마우스 오른쪽 버튼을 누르고 **작성 > EJB에서 RDB로 맵핑**을 선택하십시오. EJB에서 RDB로 맵핑 창이 열립니다.

2. **중간 맞춤**을 선택하고 **다음을** 누르십시오.
3. 데이터베이스 연결 창에서 다음을 수행하십시오.
   a. **연결 이름** 필드에 WebSphereCommerceServerExtensionsData를 입력하십시오.
   b. **데이터베이스 필드**에 전개 데이터베이스의 이름을 입력하십시오.
   c. **사용자 ID** 필드에 데이터베이스 사용자 ID를 입력하십시오.
   d. **암호** 필드에 데이터베이스 사용자의 암호를 입력하십시오.
   e. **데이터베이스 공급업체 유형** 드롭 다운 목록에서 개발 데이터베이스의 데이터베이스 공급업체 유형을 선택하십시오.
      - DB2 DB2 Universal Database 8.1
      - Oracle Oracle 9i
   f. Oracle **호스트** 필드에 데이터베이스 서버의 완전한 호스트 이름을 입력하십시오. 예를 들어, dbserver.yourcompany.com을 입력하십시오.
   g. Oracle 클래스 위치 필드에서 classes12.zip 파일의 위치를 입력하십시오. 예를 들어, D:\oracle\ora92\jdbc\lib\classes12.zip을 입력하십시오.
   h. **다음**을 누르십시오. 연결되면 데이터베이스의 테이블 목록이 표시됩니다.데이터 perspective의 데이터베이스 서버 보기를 참조하여 나중에 연결 문서를 볼 수도 있습니다.
4. **XBONUS**를 선택하고 **다음**을 누르십시오.
5. **이름 및 유형 기준 일치**를 선택한 후 **완료**를 누르십시오. 맵핑 편집기가 열립니다.
6. Oracle XBONUS 테이블을 마우스 오른쪽 버튼으로 누르고 **테이블 편집기 열기**를 선택하십시오. 테이블 편집기에서 다음을 수행하십시오.
   a. **열** 탭을 선택하십시오.
   b. BONUSPOINT 열을 선택하고 열 유형을 NUMBER에서 INTEGER로 변경하십시오.
   c. 변경사항을 저장하십시오.

7. 엔터프라이즈 bean 분할창에서 **Bonus** bean을 펼치십시오. 테이블 분할창에서 **XBONUS** 테이블을 펼치십시오.
8. 다음을 수행하여 Bonus bean의 필드를 XBONUS 테이블 열에 맵핑하십시오.
   a. Bonus bean에서 마우스 오른쪽 버튼을 누르고 **이름 기준 일치**를 선택합니다.
9. Ctrl + S를 눌러서 Map.mapxmi 파일을 저장하십시오. 파일을 닫으십시오.
10. Oracle 다음과 같이 텍스트 편집기를 사용하여 테이블 정의를 편집해야 합니다.
    a. 텍스트 편집기에서 XBONUS.xmi 파일을 여십시오.
    b. 모든 `SQLNumeric6` 어커런스를 `SQLNumeric3`으로 바꾸십시오.
    c. 변경사항을 저장하십시오.

다음 단계는 새 bean이 다른 데이터베이스에 이식 가능하도록 스키마 이름을 수정하는 것입니다. 다음과 같이 수정하십시오.

1. J2EE Perspective에서 J2EE 계층 보기로 전환하십시오.
2. **데이터베이스**를 펼친 후 **WebSphereCommerceServerExtensionsData**를 펼치십시오.
3. 스키마 노드(예: DB2USER)를 마우스 오른쪽 버튼으로 누르고 **이름 바꾸기**를 선택하십시오.
4. 값을 `NULLID`로 설정하십시오.

일단 Bonusbean 엔티티가 작성되고 스키마가 올바르게 맵핑되면 엔티티 bean의 액세스 bean을 작성해야 합니다. 이 액세스 bean은 응용프로그램이 Bonus 엔티티 bean에 들어 있는 정보에 더 간단하게 액세스하도록 합니다. WebSphere Studio Application Developer의 도구는 이미 작성한 엔티티 bean에 기초하여 액세스 bean을 작성하는 데 사용됩니다(특히 액세스 bean은 원격 인터페이스로 승격된 메소드만 사용합니다). Bonus 엔티티 bean의 액세스 bean을 작성하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼친 후 **WebSphereCommerceServerExtensionsData**에서 마우스 오른쪽 버튼을 누르고 **새로 만들기 > 액세스 bean**을 선택하십시오.
   액세스 bean 추가 창이 열립니다.
2. **복사 헬퍼**를 선택한 후 **다음**을 누르십시오.
3. **Bonus** bean을 선택하고 **다음**을 누르십시오.
4. 생성자 메소드 드롭 다운 목록에서 **findByPrimaryKey(com.ibm.commerce.extension.objects.BonusKey)**를 생성자 메소드로 선택하십시오.
5. 속성 헬퍼 섹션에서 모든 속성을 선택하십시오.
6. **완료**를 누르십시오.

다음을 펼친 후 J2EE 네비게이터 탭으로 전환하여 새로 작성된 코드를 볼 수 있습니다.

WebSphereCommerceServerExtensionsData >ejbModule >com.ibm.commerce.extension.objects

BonusAccessBean이라는 새 클래스와 BonusAccessBeanData라는 새 인터페이스가 패키지 내에서 작성되어 표시됩니다.

다음 단계는 전개 코드를 작성하는 것입니다.

코드 작성 유틸리티는 bean을 분석하여 Sun Microsystems의 EJB 스펙에 맞는지 확인하고 EJB 서버 고유 규칙을 따랐는지 확인합니다. 또한 선택된 엔터프라이즈 bean마다 코드 작성 도구는 CMP bean용 JDBC 지속 및 finder 클래스뿐 아니라 홈 및 EJBObject(원격) 구현, 홈 및 원격 인터페이스용 구현 클래스를 작성합니다. 또한 홈 및 원격 인터페이스용 스텁(stub)뿐 아니라, IIOP를 통한 RMI 액세스에 필요한 Java ORB, 스텁, 타이 클래스를 작성합니다.

전개 코드를 작성하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼친 후 **WebSphereCommerceServerExtensionsData**에서 마우스 오른쪽 버튼으로 누르고 **작성 > 전개 및 RMIC 코드**를 선택하십시오.
2. **Bonus** bean을 선택하고 **완료**를 누르십시오.

J2EE 네비게이터 보기로 전환하여 새로 작성된 코드를 볼 수 있습니다. 다음을 볼 수 있습니다.

표 11.

코드 유형	클래스 이름
컨테이너 구현 작성 코드	EJSCMPBonusHomeBean.java
	EJSRemoteCMPBonus.java
	EJSRemoteCMPBonusHome.java
	EJSFinderBonusBean.java
JDBC 액세스 코드	EJSJDBCPersisterCMPBonusBean.java
RMI 타이 및 스텁 코드	_EJSRemoteCMPBonus_Tie.java
	_Bonus_Stub.java
	_EJSRemoteCMPBonusHome_Tie.java
	_BonusHome_Stub.java

다음 단계는 다음을 수행하여 새 엔터프라이즈 bean을 테스트하기 위해 범용 테스트 클라이언트를 사용하는 것입니다.

1. 서버 perspective로 전환하십시오.
2. 서버 구성 보기에서 **WebSphereCommerceServer** 서버를 두 번 누르고 **구성** 탭을 누르십시오.
3. **범용 테스트 클라이언트 사용**을 선택하십시오. 변경사항을 저장하십시오.
4. 서버 보기에서 **WebSphereCommerceServer** 서버를 마우스 오른쪽 버튼으로 누른 후 **시작**을 선택하십시오.
5. J2EE 계층에서 **EJB 모듈 > WebSphereCommerceServerExtensionsData**을 펼치십시오.
6. **Bonus** bean에서 마우스 오른쪽 버튼을 누르고 **서버에서 실행**을 선택하십시오.
   IBM 범용 테스트 클라이언트가 열립니다.
7. 왼쪽 분할창에서 **Bonus**를 누른 후 **Bonus 홈**을 누르십시오.
8. **Bonus create(Long, Integer)** 메소드를 누르십시오.
9. 오른쪽 분할창에 있는 **Long** 필드에 `-1000`을 입력하고 **Integer** 필드에 `1000`을 입력하십시오.

10. **Invoke**를 누르십시오. 결과는 맨 아래 분할창에 표시됩니다.
11. **오브젝트에 대한 작업**을 눌러서 원격 인터페이스를 참조 분할창에 추가하면 EJB 참조 아래에 입력한 값이 표시됩니다. BONUS 테이블에 새 레코드가 작성되었습니다.
12. **getMemberId** 메소드를 선택하고 **호출**을 누르십시오. 결과 -1000이 맨 아래 분할창에 표시됩니다.
13. 테스트 클라이언트를 닫고 서버를 중지하십시오.

Bonus 엔티티 bean과 MyNewControllerCmd 통합

이전 절에서는 WebSphere Studio Application Developer 내에서 작성된 테스트 클라이언트를 사용하여 Bonus 엔티티 bean을 테스트했습니다. 테스트 중, 데이터베이스 정보를 성공적으로 갱신할 수 있음을 판별했습니다. 이제는 Bonus 엔티티 bean을 MyNewControllerCmd 로직과 통합합니다. Java 코드가 갱신되면 MyNewJSPTemplate.jsp 파일이 고객의 보너스 점수 대차 대조를 갱신하게 하는 인터페이스를 작성하도록 갱신됩니다.

Bonus 엔티티 bean 통합에는 다음과 같은 고급 단계가 수반됩니다.

1. 보너스 점수를 위한 필드 및 메소드를 포함하도록 MyNewTaskCmd 태스크 명령 수정, validateParameters 메소드 갱신 및 사용자의 보너스 점수 잔고를 갱신하도록 로직 추가
2. getResources 메소드를 MyNewControllerCmdImpl 클래스에 추가하여 명령이 사용하는 자원 목록 리턴. 이 메소드는 액세스 제어 목적으로 포함됩니다.
3. JSP 템플리트에 보너스 점수를 표시할 수 있도록 새 BonusDataBean 작성
4. 새 자원에 대한 새 액세스 제어 정책 작성
5. 사용자의 보너스 점수를 입력하고 사용자의 새 보너스 점수 잔고를 표시할 수 있도록 MyNewJSPTemplate.jsp 템플리트를 수정

보너스 점수를 포함하도록 MyNewTaskCmd 인터페이스 수정

이 단계에서는 다음을 수행하여 보너스 점수에 필요한 필드 및 메소드를 지정하도록 MyNewTaskCmd 인터페이스를 수정합니다.

1. Java perspective로 전환한 후 **WebSphereCommerceServerExtensionsLogic** 프로젝트를 펼치십시오.

2. **com.ibm.commerce.sample.commands\src** 디렉토리를 펼치십시오.
3. **MyNewTaskCmd** 인터페이스를 두 번 눌러서 소스 코드를 보십시오.
4. 다음 패키지를 포함하도록 Import section 2의 주석을 제거하십시오.

```
/// Import section 2 ////////////////////////////////////
import com.ibm.commerce.extension.objects.*;
/// End of import section 2 ////////////////////////////////
```

5. 메소드에 다음 코드가 삽입되도록 Section 4 주석을 제거하십시오.

```
/// Section 4 ////////////////////////////////////////////////

 public java.lang.Integer getOldBonusPoints();
 public Integer getTotalBonusPoints();

 public void setBonusAccessBean(BonusAccessBean bb);
 public BonusAccessBean getBonusAccessBean();


/// End of section 4/////////////////////////////////////////
```

6. 변경사항을 저장하십시오.

보너스 점수를 계산하도록 MyNewTaskCmdmpl 수정

MyNewTaskCmdImpl은 Bonus 엔티티 bean과 MyNewControllerCmd 사이의 통합 점수로 사용됩니다(MyNewControllerCmd가 MyNewTaskCmd를 호출하므로).

보너스 점수를 계산하도록 MyNewTaskCmdImpl을 수정하려면 다음을 수행하십시오.

1. **MyNewTaskCmdImpl** 클래스를 선택하여 소스 코드를 보십시오.
2. 다음 패키지를 삽입하도록 Import section 2의 주석을 제거하십시오.

```
/// Import section 2 ////////////////////////////////////
import com.ibm.commerce.extension.objects.*;
///  End of Import section 2 ////////////////////////////////
```

3. 클래스에 다음 코드가 삽입되도록 Section 3A 및 3B 주석을 제거하십시오.

```
//// Section 3A ////////////////////////////////////////////

 private java.lang.Integer oldBonusPoints;
 private java.lang.Integer totalBonusPoints;


 private BonusAccessBean bb = null;
```

```
////End of Section 3A ////////////////////////////////////////

//// Section 3B ////////////////////////////////////////////

 public void setBonusAccessBean(BonusAccessBean newBB) {
  bb = newBB;
 }

 public BonusAccessBean getBonusAccessBean(){
  return bb;
 }

    public java.lang.Integer getOldBonusPoints() {
  return oldBonusPoints;
 }

 public Integer getTotalBonusPoints(){
  return totalBonusPoints;
 }


/// End of section 3B ////////////////////////////////////////
```

4. 아웃라인 보기에서 **validateParameters** 메소드를 선택하고 다음 코드가 메소드에 삽입되도록 섹션 2의 주석을 제거하십시오.

```
// section 2 ////////////////////////////////////////////////

 try {
    oldBonusPoints = bb.getBonusPoint();
} catch (javax.ejb.FinderException e) {
  try {
   // If bb is null, create a new instance
    bb = new BonusAccessBean(new Long(foundUserId), new Integer(0));
    oldBonusPoints = new Integer(0);
} catch (javax.ejb.CreateException ec) {
throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
     this.getClass().getName(), "validateParameters");
} catch (javax.naming.NamingException ec) {
throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
     this.getClass().getName(), "validateParameters");
} catch (java.rmi.RemoteException ec) {
throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
     this.getClass().getName(), "validateParameters");
   }
} catch (javax.naming.NamingException e) {
throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
    this.getClass().getName(), "validateParameters");
} catch (java.rmi.RemoteException e) {
```

```
throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
    this.getClass().getName(), "validateParameters");
} catch (javax.ejb.CreateException e) {
throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
    this.getClass().getName(), "validateParameters");
  }

// end of section 2 ////////////////////////////////////////////
```

5. 아웃라인 보기에서 **performExecute** 메소드를 선택하십시오.
6. performExecute 메소드의 소스 코드에서 Section 2 주석을 제거하십시오. 메소드에 다음 코드가 삽입됩니다.

```
/// use BonusAccessBean to update new bonus point
/// Section 2 ////////////////////////////////////////////////

 int newBP =  oldBonusPoints.intValue() + getInputPoints().intValue();
 totalBonusPoints = new Integer (newBP);
 bb.setBonusPoint(totalBonusPoints)   ;

 try {
   bb.commitCopyHelper();
} catch (javax.ejb.FinderException e) {
throw new ECSystemException(ECMessage._ERR_FINDER_EXCEPTION,
          this.getClass().getName(), "performExecute");
} catch (javax.naming.NamingException e) {
throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
          this.getClass().getName(), "performExecute");
} catch (java.rmi.RemoteException e) {
throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
          this.getClass().getName(), "performExecute");
} catch (javax.ejb.CreateException e) {
throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
          this.getClass().getName(), "performExecute");
 }

/// End of section 2 ////////////////////////////////////////////
```

7. 변경사항을 저장하십시오.

getResources 메소드를 MyNewControllerCmdImpl 클래스에 추가

이 절에서는 새 getResources 메소드를 MyNewControllerCmdImpl에 추가합니다. 이 메소드는 처리 중 명령이 사용하는 자원 목록을 리턴합니다. 이 메소드는 자원 레벨 액세스 제어를 위해 필요합니다.

getResources 메소드를 추가하려면 다음을 수행하십시오.

1. **MyNewControllerCmdImpl** 클래스를 두 번 눌러서 열고 해당 소스 코드를 보십시오.

2. 소스 코드에서 Import Section 1의 주석을 제거하여 클래스에 다음 패키지를 삽입하십시오.

```
/// Import Section 2 ////////////////////////////////
import com.ibm.commerce.extension.objects.*;
/// End of Import Section 2 ////////////////////////
```

3. 클래스에 다음 코드가 삽입되도록 Section 3 주석을 제거하십시오.

```
/// Section 3 ////////////////////////////////////////////////
/// Create an instance variable of type AccessVector to hold
/// the resources and a BonusAccessBean instance variable for
/// access control purposes.

private AccessVector resources = null;
    private BonusAccessBean bb = null;

/// End of Section 3 ////////////////////////////////////////
```

4. 소스 코드에서 액세스 제어 절의 주석을 제거하십시오. 이 절은 다음과 같은 코드 단편에 표시된 것처럼 표시됩니다.

```
/// AccessControl Section ////////////////////////////////////

public AccessVector getResources() throws ECException {

   if (resources == null) {

    /// use UserRegistryAccessBean to check user reference number

 String refNum = null;
 String methodName = "getResources";

  rrb = new UserRegistryAccessBean();

 try {
   rrb = rrb.findByUserLogonId(getUserName());
          refNum = rrb.getUserId();
} catch (javax.ejb.FinderException e) {
throw new ECSystemException(ECMessage._ERR_FINDER_EXCEPTION,
        this.getClass().getName(),methodName,e);
} catch (javax.naming.NamingException e) {
throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
        this.getClass().getName(), methodName,e);
} catch (java.rmi.RemoteException e) {
throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
        this.getClass().getName(), methodName,e);
} catch (javax.ejb.CreateException e) {
```

```
throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
        this.getClass().getName(), methodName,e);
  }


 /// find the bonus bean for this registered user

    bb = new com.ibm.commerce.extension.objects.BonusAccessBean();
 try {
if (refNum != null) {
    bb.setInitKey_memberId(new Long(refNum));
    bb.refreshCopyHelper();
    resources = new AccessVector(bb);
   }
} catch (javax.ejb.FinderException e) {

   ///doesn't have a bonus object so return the container that
     ///will hold the bonus object when it's created
  resources = new AccessVector(rrb);
   return resources;

} catch (javax.naming.NamingException e) {
throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
      this.getClass().getName(), methodName);
} catch (java.rmi.RemoteException e) {
throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
      this.getClass().getName(), methodName);
} catch (javax.ejb.CreateException e) {
throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
      this.getClass().getName(), methodName);
  }

 }
   return resources;
}

/// End of AccessControl Section ////////////////////////////////
```

5. 변경사항을 저장하십시오.

MyNewControllerCmdImpl 클래스의 performExecute 메소드 수정

이 단계에서는 다음과 같이 새 보너스 bean에 관련된 코드를 포함하도록 MyNewControllerCmdImpl의 performExecute 메소드에서 코드를 수정합니다.

1. **MyNewControllerCmdImpl** 클래스를 두 번 누르십시오.
2. 아웃라인 보기에서 **performExecute** 메소드를 선택하십시오.

3. 소스 코드에서 Section 4E, 4G, 4H의 주석을 제거하여 메소드에 다음 코드를 삽입하십시오.

```
// Section 4E ////////////////////////////////////
/// pass bb instance variable to the task command
    cmd.setBonusAccessBean(bb);
// End of section 4E ////////////////////////////

// Section 4G ////////////////////////////////////
if (cmd.getOldBonusPoints() != null) {
      rspProp.put("oldBonusPoints", cmd.getOldBonusPoints());
}
// End of section 4G ////////////////////////////

// Section 4H ////////////////////////////////////
///Instantiate the bonus data bean , then put it to response properties
   BonusDataBean bdb  = new com.ibm.commerce.sample.databeans.BonusDataBean(
      cmd.getBonusAccessBean());
  rspProp.put("bdbInstance", bdb );
// End of section 4H ////////////////////////////
```

4. 변경사항을 저장하십시오.

주: BonusDataBean이 아직 정의되지 않았으므로 오류를 보게 됩니다. 다음 섹션에서 정정됩니다.

BonusDataBean 데이터 bean 작성

프로그래밍 모델에 따라, 새 Bonus 엔티티 bean에 해당하는 새 데이터 bean을 작성해야 합니다. 모든 엔티티 bean이 해당되는 데이터 bean을 가지고 있어야 하는 것은 아니지만, JSP 템플리트에서 엔티티 bean으로부터 정보를 표시할 수 있으려면 이 목적의 새 데이터 bean을 작성해야 합니다.

이 시나리오에서는 BonusAccessBean을 확장하는 새 BonusDataBean 데이터 bean을 작성해야 합니다. 학습의 다른 부분처럼 기본 코드가 제공되므로 코드의 다양한 섹션에 대해 주석을 제거해야 합니다.

BonusDataBean을 코드에 삽입하려면 다음을 수행하십시오.

1. 첫 번째 단계는 다음과 같이 새 데이터 bean의 기본 코드를 반입하는 것입니다.
   a. Java Perspective에서 J2EE 네비게이터로 전환하십시오.
   b. **WebSphereCommerceServerExtensionsLogic** 프로젝트를 펼치십시오.

c. **src** 폴더에서 마우스 오른쪽 버튼을 누르고 **반입**을 선택하십시오.
가져오기 마법사가 열립니다.

d. **반입 소스 선택** 목록에서 **Zip 파일**을 선택하고 **다음**을 누르십시오.

e. **찾아보기(Zip 파일** 필드 옆에 있는)를 누르고 견본 코드를 탐색하십시오.
이 파일은 다음 위치에 지정됩니다.
yourDirectory\WC_SAMPLE_55.zip
여기서, *yourDirectory*는 패키지를 다운로드한 디렉토리입니다.

f. **모두 선택 취소**를 누른 후 디렉토리를 펼쳐 반입할 다음 파일을 선택하십시오.

- com\ibm\commerce\sample\databeans\BonusDataBean.java

g. **폴더** 필드에서는 이미 WebSphereCommerceServerExtensionsLogic/src 폴더가 지정되어 있습니다. 이 값을 변경하지 마십시오.

h. **완료**를 누르십시오.

2. **BonusDataBean** 클래스를 두 번 누르고 해당 소스 코드를 보십시오.
3. 소스 코드에서 Section 1의 주석을 제거하여 bean에 다음 코드를 삽입하십시오.

```
/// Section 1 ////////////////////////////////////////////////

// create fields and accessors (setter/getter methods)

 private java.lang.String userId;
 private java.lang.Integer totalBonusPoints;


 public java.lang.String getUserId() {
  return userId;
 }

 public void setUserId(java.lang.String newUserId) {
  userId = newUserId;

  ////////////////////////////////////
  /// Section A : instantiate BonusAccessbean

  if (userId != null)
   this.setInitKey_memberId(new Long(newUserId));

  ////////////////////////////////////
 }
```

```
    public java.lang.Integer getTotalBonusPoints() {
  return totalBonusPoints;
 }
 public void setTotalBonusPoints(java.lang.Integer newTotalBonusPoints) {
  totalBonusPoints= newTotalBonusPoints;
 }


//// End of section 1 ////////////////////////////////////////////
```

4. Bean에 다음 코드 섹션이 삽입되도록 Section 2 주석을 제거하십시오.

```
/// Section 2/////////////////////////////////////////////////

// create a new constructor for passing access bean into databean
// so that JSP can work with the access bean

public BonusDataBean(BonusAccessBean bb) throws com.ibm.commerce
.exception.ECException {
 try {
  super.setEJBRef(bb.getEJBRef());
} catch (javax.ejb.FinderException e) {
throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
     "BonusDataBean", "BonusDataBean(bb)");
} catch (javax.naming.NamingException e) {
throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
     "BonusDataBean", "BonusDataBean(bb)");
} catch (java.rmi.RemoteException e) {
throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
     "BonusDataBean", "BonusDataBean(bb)");
} catch (javax.ejb.CreateException e) {
throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
     "BonusDataBean", "BonusDataBean(bb)");
  }
}

//// End of section 2 ////////////////////////////////////////////
```

5. Bean에 다음 코드가 삽입되도록 Section 3 주석을 제거하십시오.

```
//// Section 3 ////////////////////////////////////////////////

// set additional data field that is used for instantiating BonusAccessbean

 try {

  setUserId(getRequestProperties().getString("taskOutputUserId"));

  try {
   super.refreshCopyHelper();
} catch (javax.ejb.FinderException e) {
throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
     "BonusDataBean", "populate");
} catch (javax.naming.NamingException e) {
throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
```

```
        "BonusDataBean", "populate");
    } catch (java.rmi.RemoteException e) {
    throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
        "BonusDataBean", "populate");
    } catch (javax.ejb.CreateException e) {
    throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
        "BonusDataBean", "populate");
      }


     }
    catch (ParameterNotFoundException e){}


    ///// End of Section 3 ////////////////////////////////////////
    }
```

6. Bean에 다음 코드가 삽입되도록 Section 4 주석을 제거하십시오.

```
/// Section 4 ////////////////////////////////////////

// copy input TypedProperteis to local

 requestProperties = aParam;

/// End of section 4 ///////////////////////////////////
```

7. 변경사항을 저장하십시오.
8. **WebSphereCommerceServerExtensionsLogic** 프로젝트에서 마우스 오른쪽 버튼을 누르고 **프로젝트 재빌드**를 선택하여 변경된 코드를 컴파일하십시오.

새 엔티티 bean의 액세스 제어 정책 작성

견본 액세스 제어 정책이 제공됩니다. 이 정책은 다음과 같은 액세스 제어 오브젝트를 작성합니다.

조치 작성되는 조치는 com.ibm.commerce.sample.commands.MyNewControllerCmd 입니다

조치 그룹

작성되는 조치 그룹은 MyNewControllerCmdActionGroup입니다. 이 조치 그룹에는 하나의 조치만이 들어 있습니다. com.ibm.commerce.sample.commands.MyNewControllerCmd

자원 카테고리

작성되는 자원 카테고리는 `com.ibm.commerce.sample.objects.BonusResourceCategory`입니다. 이 자원 카테고리는 Bonus 엔티티 bean용입니다.

자원 그룹

작성된 자원 그룹은 `BonusResourceGroup`입니다. 이 자원 그룹에는 선행 자원 카테고리만이 들어 있습니다.

정책 작성되는 정책은 `AllUsersUpdateBonusResourceGroup`입니다. 이 정책은 사용자가 보너스 오브젝트의 "소유자"인 경우에만 Bonus bean에서 MyNewControllerCmd 조치를 수행할 수 있게 합니다. 예를 들어, 사용자가 `tester@mycompany` 사용자로 로그온하는 경우 사용자는 자신의 보너스 점수만을 수정할 수 있습니다.

`AllUsersUpdateBonusResourceGroup` 정책 설정은 다음 단계와 관련됩니다.

1. 사용자 환경을 반영하도록 액세스 제어 정책 수정
2. `acpload` 명령을 사용하여 SampleACPolicy.xml 파일 로드
3. `acpnlsload` 명령을 사용하여 SampleACPolicy_en_US.xml 설명 로드

사용자 환경의 액세스 제어 정책을 사용자 정의하려면 다음을 수행하십시오.

1. 다음과 같이 FashionFlow 상점의 구성원 ID 값을 판별하십시오.
   - DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하십시오.
     a. 개발 데이터베이스에 연결하십시오.
     b. 다음 SQL문을 실행하십시오.

        ```
        select member_id from storeent where storeent_id=FF_storeent_ID
        ```

        여기서, *FF_storeent_ID*는 FashionFlow 상점의 상점 엔티티 ID입니다. 예를 들어, 다음을 입력할 수 있습니다.

        ```
        select member_id from storeent where storeent_id=10001
        ```

        구성원 ID 값을 쓰십시오: _____
   - Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하십시오.

a. Oracle SQL Plus 명령 창을 열어 전개 데이터베이스에 연결하십시오.

b. 다음을 입력하십시오.

```
column member_id format
99999999999999999999999999999999999(39개의 9)
```

이것은 전체 값을 뷰에 충분한 member_id 열 너비 표시를 설정합니다.

c. 다음 SQL문을 실행하십시오.

```
select member_id from storeent where storeent_id=FF_storeent_ID
```

여기서, *FF_storeent_ID*는 FashionFlow 상점의 상점 엔티티 ID입니다. 예를 들어, 다음을 입력할 수 있습니다.

```
select member_id from storeent where storeent_id=10001
```

구성원 ID 값을 쓰십시오: _____

d. 다음을 입력하여 member_id 열의 표시 너비를 재설정하십시오.

```
column member_id clear
```

2. 텍스트 편집기를 사용하여 SampleACPolicy_template.xml 파일을 열고 (*WCStudio_installdir*\Commerce\xml\policies\xml 디렉토리에서) *FashionFlowMemberId*를 1단계에서 결정한 Fashion Flow 상점의 구성원 ID 값으로 바꾸십시오. 수정한 파일을 SampleACPolicy.xml로 저장하십시오(동일한 디렉토리에).

3. 텍스트 편집기를 사용하여 SampleACPolicy_template_en_US.xml 파일을 열고(*WCStudio_installdir*\Commerce\xml\policies\xml 디렉토리에서) *FashionFlowMemberId*를 1단계에서 결정한 Fashion Flow 상점의 구성원 ID 값으로 바꾸십시오. 수정된 파일을 SampleACPolicy_en_US.xml로 저장하십시오(동일한 디렉토리에).

4. 명령 프롬프트에서 다음 디렉토리로 전환하십시오.
WCStudio_installdir\commerce\bin

5. SampleACPolicy.xml 파일을 로드하려면 다음 양식의 acpload 명령을 실행해야 합니다.

```
acpload db_name db_user db_password inputXMLFile
```

여기서,

- *db_name*은 데이터베이스의 이름입니다.
- *db_user*는 데이터베이스 사용자 이름입니다.
- *db_password*는 데이터베이스 암호입니다.
- *inputXMLFile*은 정책을 포함하는 XML 파일의 이름입니다. 이 경우, SampleACPolicy.xml을 입력하십시오.

예를 들어, 다음과 같은 명령을 발행할 수 있습니다.

```
acpload Demo_dev db2user db2user SampleACPolicy.xml
```

6. 정책 설명을 로드하려면 다음 양식의 acpnlsload 명령을 실행해야 합니다.

   ```
   acpnlsload db_name db_user db_password inputXMLFile
   ```

   예를 들어, 다음과 같은 명령을 발행할 수 있습니다.

   ```
   acpnlsload Demo_dev db2user db2user SampleACPolicy_en_US.xml
   ```

7. 테스트 환경용 서버가 현재 실행 중일 경우, WebSphere Commerce 관리 콘솔에서 레지스트리 최신 정보로 고침 옵션을 사용하여 다음과 같이 액세스 제어 레지스트리를 갱신할 수 있습니다.
   a. 웹 브라우저를 열고 다음 URL을 입력하십시오.

      ```
      https://localhost/webapp/wcs/admin/servlet/ToolsLogon?XMLFile=
         adminconsole.AdminConsoleLogon
      ```

   b. 프롬프트가 표시되면 사이트 운영자 ID를 사용하여 로그인하십시오.
   c. **사이트**에서 작업할 것을 선택하고 **확인**을 누르십시오.
   d. **구성** 메뉴에서 **레지스트리**를 선택하십시오.
   e. **모두 갱신**을 누르고, 잠시 후에 **최신 정보로 고침**을 눌러 갱신이 수행되었는지 검증하십시오.
   f. 로그아웃하고 관리 콘솔 창을 닫으십시오.

보너스 점수를 포함하도록 MyNewJSPTemplate.jsp 템플리트 수정

표시 페이지를 수정하려면 다음을 수행하십시오.

1. WebSphere Studio Application Developer에서 웹 perspective로 전환하십시오.
2. **MyNewJSPTemplate_All.jsp** 및 **MyNewJSPTemplate.jsp** 파일 모두를 여십시오.

3. MyNewJSPTemplate_All.jsp 파일에서 MyNewJSPTemplate.jsp 파일로 Section 9를 복사하십시오. 그러면 JSP 템플리트에 다음 텍스트가 삽입됩니다.

```
<!-- SECTION 9 -->

<h2><fmt:message key="BonusAdmin" bundle="${tutorial}" /> </h2>

<c:if test="${!empty taskOutputUserId}">
 <ul>
  <li>
   <b>
   <fmt:message key="PointBeforeUpdate" bundle="${tutorial}" />
   <c:out value="${oldBonusPoints}"/>
   </b>
  </li>
  <li>
   <b>
   <fmt:message key="PointAfterUpdate" bundle="${tutorial}" />
      <c:out value="${bdbInstance.bonusPoint}" />
      </b>
     </li>
 </ul>
</c:if>


      <br />
<b><fmt:message key="EnterPoint" bundle="${tutorial}" /></b><p />


<form name="Bonus" action="MyNewControllerCmd">
<table>
 <tr>
  <td>
   <b>Logon ID </b>
  </td>
  <td>
   <input type="text" name="input1" value="<c:out
    value="${userName}"/>" />
  </td>
 </tr>
 <tr>
  <td>
   <b>Bonus Point</b>
  </td>
  <td>
   <input type="text" name="input2" />
  </td>
 </tr>
```

```
 <tr>
  <td colspan="2">
   <input type="submit" />
  </td>
 </tr>
</table>
</form>

<!-- END OF SECTION 9 -->
```

4. MyNewJSPTemplate.jsp 파일을 저장하십시오.

통합된 Bonus bean 테스트

새 Bonus bean은 액세스 제어 하에서 보호되며 사용자는 소유한 bean에 대해 MyNewControllerCmd 조치만 수행할 수 있으므로 사용자는 로그인해야 합니다. 이 같은 이유로 사용자는 견본 상점에서 로그인 기능을 사용하여 해당 사용자에게 로그인을 허용합니다.

새 로직을 테스트하려면 다음을 수행하십시오.

1. 서버 보기로 전환하십시오.
2. **WebSpherCommerceServer** 서버에서 마우스 오른쪽 버튼을 누르고 **시작** (또는 **다시 시작**)을 선택하십시오.
3. 사용자 상점의 **index.jsp** 파일에서 마우스 오른쪽 버튼을 누르고 **서버에서 실행**을 선택하십시오.
   상점 홈 페이지가 표시됩니다.
4. 다음을 수행하여 등록된 사용자로 로그온하십시오.
   a. **등록** 링크를 누르십시오.
      등록 페이지가 표시됩니다.
   b. **전자 우편 주소** 필드에 305 페이지의 『사용자 이름 유효성 확인 테스트』에서 작성한 사용자의 전자 우편 주소를 입력하십시오.
   c. **암호** 필드에 사용자의 암호를 입력하고 **로그인**을 누르십시오.
   d. 로그인이 완료되면 같은 브라우저에 다음 URL을 입력하십시오.

      ```
      http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
         input1=user_e-mail&input2=1000
      ```

여기서, *user_e-mail*은 305 페이지의 『사용자 이름 유효성 확인 테스트』에서 작성한 사용자의 전자 우편 주소입니다. 사용자가 보너스 포인트 대차 대조를 갱신할 수 있는 새 양식과 이전의 모든 출력 매개변수가 들어 있는 페이지가 함께 표시됩니다.

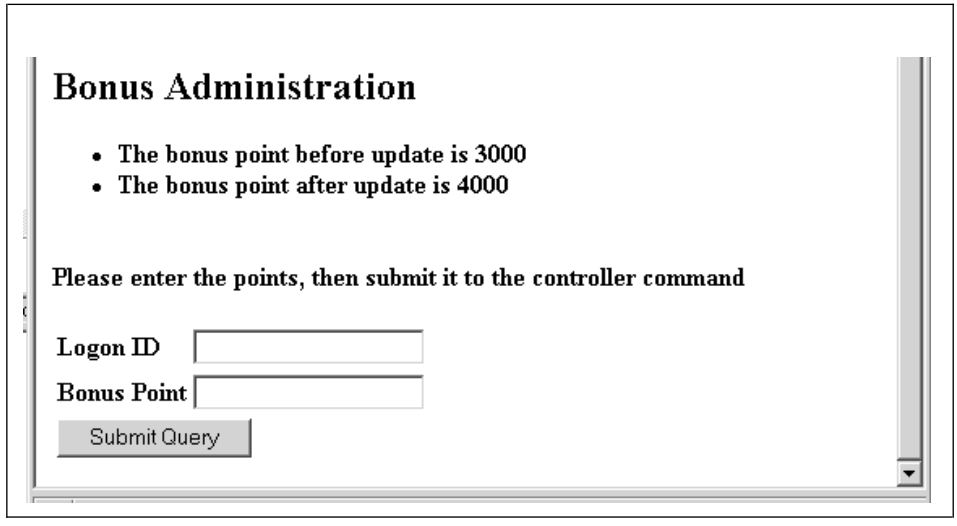


그림 41.

e. 이제 **로그온 ID** 필드에 사용자의 전자 우편 주소를 입력하고, **보너스 점수** 필드에 `500`을 입력하십시오. **제출**을 누르십시오. 보너스 점수 대차 대조가 갱신되었음을 표시하는 다음과 유사한 페이지가 표시됩니다.

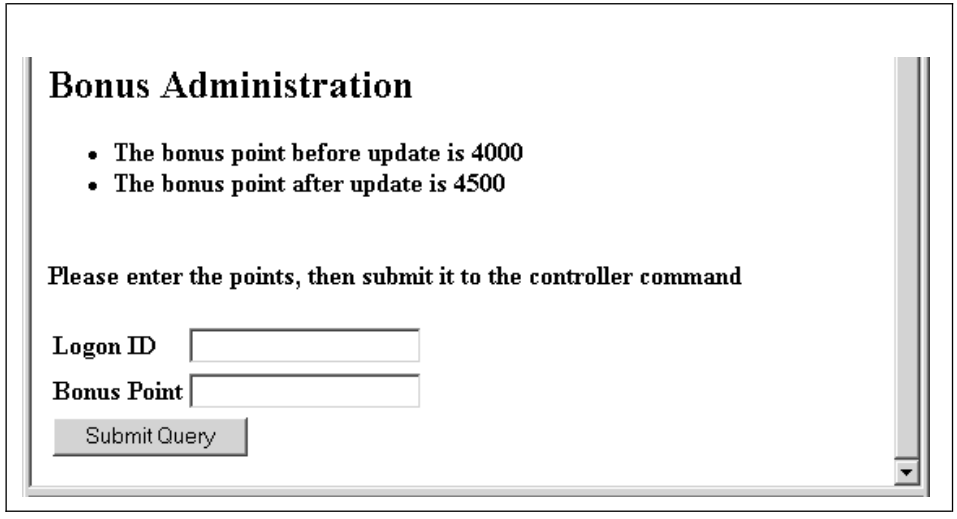


그림 42.

보너스 점수 로직 전개

이 절에서는 원격 WebSphere Commerce Server에서 실행 중인 상점으로 새 비즈니스 로직을 전개하는 방법에 대해 설명합니다. 전개 단계를 시작하려면 먼저 원격 WebSphere Commerce Server에 상점을 작성해 놓아야 합니다(InFashion 견본 상점을 기초로).

개발 처리에는 대상 WebSphere Commerce Server에서 수행되는 단계 뿐만 아니라 개발 시스템에서 수행되는 단계도 포함됩니다.

대상 WebSphere Commerce Server에 전개해야 하는 여러 가지 유형의 자원이 있습니다. 자원 유형은 다음과 같습니다.

- 컨트롤러 명령, 태스크 명령 및 데이터 bean 로직
- 엔터프라이즈 bean 로직
- JSP 템플리트 및 이미지 파일
- 특성 파일 및 자원 번들
- 스키마 갱신(새 테이블)과 명령 레지스트리 갱신을 포함한 데이터베이스 갱신사항
- 액세스 제어 갱신사항

이 절에서는 이 모든 자원을 대상 WebSphere Commerce Server에 *증분적으로* 전개하는 방법에 대해 설명합니다. 이는 EAR 파일 전체를 전개하는 것과는 반대로 점층적으로 수행됩니다.

명령 및 데이터 bean JAR 파일 작성

이 절에서는 컨트롤러 명령, 태스크 명령 및 데이터 bean 로직을 포함하는 JAR 파일을 작성하는 방법에 대해 설명합니다.

JAR 파일을 작성하려면 개발 시스템에서 다음 단계를 수행하십시오.

1. 로컬 파일 시스템에 *drive*:\ExportTemp라고 하는 디렉토리를 작성하십시오.
2. WebSphere Studio Application Developer에서 J2EE 네비게이터 보기로 전환하십시오.
3. **WebSphereCommerceServerExtensionsLogic** 프로젝트에서 마우스 오른쪽 버튼을 누른 후 **반출**을 선택하십시오.
   반출 마법사가 열립니다.
4. 반출 마법사에서 다음을 수행하십시오.
   a. **JAR 파일**을 선택하고 **다음**을 누르십시오.
   b. **반출할 자원 선택?** 아래에 있는 왼쪽 분할창이 프로젝트 이름으로 채워집니다. 이 값은 있는 그대로 두십시오.
   c. 오른쪽 분할창에서 다음 자원만이 선택되었음을 확인하십시오.
      - .classpath
      - .project
      - .serverPreference
   d. **작성된 클래스 파일 및 자원 반출**이 선택되어 있는지 확인하십시오.
   e. Java 소스 파일 및 자원 **반출**을 선택하지 *마십시오*.
   f. **반출 대상 선택** 필드에서 사용할 완전한 JAR 파일 이름을 입력하십시오. 이 경우에는 *drive*:\ExportTemp\WebSphereCommerceServerExtensionsLogic.jar을 입력하십시오. JAR 파일 이름은 WebSphereCommerceServerExtensionsLogic.jar이어야 합니다.

g. **완료**를 누르십시오.

EJB JAR 파일 작성

EJB JAR 파일을 작성하려면 다음을 수행하십시오.

1. WebSphere Studio Application Developer를 열고 J2EE 네비게이터 보기로 전환하십시오.
2. **WebSphereCommerceServerExtensionsData** 프로젝트를 펼치십시오.
3. **EJB 전개 설명자**를 두 번 누르십시오.
4. 개요 탭을 선택한 상태에서 분할창의 맨 아래로 화면이동하여 **WebSphere 바인딩** 섹션을 찾으십시오.
5. **DataSource JNDI 이름** 필드에 대상 WebSphere Commerce Server의 데이터소스 JNDI 이름을 입력하십시오. 다음은 예제 값입니다.

   DB2 `jdbc/WebSphere Commerce DB2 DataSource demo`
   여기서, 대상 WebSphere Commerce Server는 DB2 데이터베이스를 사용 중이고 WebSphere Commerce 인스턴스 이름은 “demo”입니다.

   Oracle `jdbc/WebSphere Commerce Oracle DataSource demo`
   여기서, 대상 WebSphere Commerce Server는 Oracle 데이터베이스를 사용 중이고 WebSphere Commerce 인스턴스 이름은 “demo”입니다.

DataSource JNDI 이름에 대한 값은 “jdbc/”를 대상 WebSphere Commerce Server의 데이터 소스 이름에 추가하여 작성됩니다. 대상 WebSphere Commerce Server에서 *instanceName*.xml 파일을 열고 파일에 있는 DatasourceName=을 검색하여 데이터 소스 이름을 확인할 수 있습니다.

6. 전개 설명자 변경사항을 저장하십시오(ctrl+s).
7. J2EE 네비게이터 보기에서 **WebSphereCommerceServerExtensionsData** 프로젝트에서 마우스 오른쪽 버튼을 누른 후 **반출**을 선택하십시오.
   반출 마법사가 열립니다.
8. 반출 마법사에서 다음을 수행하십시오.
   a. **EJB JAR 파일**을 선택하고 **다음**을 누르십시오.
   b. **반출할 자원은?** 값은 EJB 프로젝트의 이름으로 채워집니다. 이 값은 있는 그대로 두십시오.

c. **자원 반출 위치** 필드에서 사용할 완전한 JAR 파일 이름을 입력하십시오. 이 경우에는 *drive*:\ExportTemp\WebSphereCommerceServerExtensionsData.jar을 입력하십시오.

d. **완료**를 누르십시오.

9. JAR 파일을 작성하였으면 5단계에서 수행한 로컬 전개 설명자 변경사항을 실행 취소하여 로컬 테스트 서버에 필요한 설정을 복원하십시오.

주: 이 학습은 전개 데이터베이스와 대상 WebSphere Commerce Server에 사용된 데이터베이스가 동일한 유형이라고 가정합니다. 다른 데이터베이스 유형에 전개하고 있다면 236 페이지의 『변환으로 EJB JAR 파일 작성』의 지시사항에 따라야 할 것입니다.

상점 자원 반출

상점 자원을 반출하려면 다음을 수행하십시오.

1. J2EE 네비게이터 보기로 전환하십시오.
2. **Stores** 폴더를 펼치십시오.
3. **Web Content** 폴더에서 마우스 오른쪽 버튼을 누르고 **반출**을 선택하십시오. 반출 마법사가 열립니다.
4. 반출 마법사에서 다음을 수행하십시오.
   a. **파일 시스템**을 선택하고 **다음**을 누르십시오.
   b. **모두 선택 취소**를 누르십시오.
   c. 다음 자원을 반출하도록 선택하십시오.
      - Web Content*FashionFlow_name*\MyNewJSPTemplate.jsp
      - Web Content*FashionFlow_name*\images\male_blueshirt.gif
      - Web Content\WEB-INF\classes*FashionFlow_name*\Tutorial_NLS_en_US.properties
      - Web Content\WEB-INF\lib\jstl.jar
      - Web Content\WEB-INF\lib\standard.jar
   d. **파일의 디렉토리 구조 작성**을 선택하십시오.

e. 디렉토리 필드에 자원을 놓을 임시 디렉토리를 입력하십시오. 예를 들어, C:\ExportTemp를 입력하십시오.

f. **완료**를 누르십시오.

액세스 제어 정책 패키징

이 절에서는 새 자원에 대해 작성된 액세스 제어 정책을 다음과 같이 *drive*:\ExportTemp 디렉토리에 복사합니다.

1. *WCStudio_installdir*\Commerce\xml\policies\xml 디렉토리를 탐색하십시오.
2. 다음 파일을 *drive*:\ExportTemp\ACPolicies 디렉토리에 복사하십시오.
   - MyNewViewACPolicy.xml
   - MyNewControllerCmdACPolicy.xml
   - SampleACPolicy_template.xml
   - SampleACPolicy_template_en_US.xml

대상 WebSphere Commerce Server로 자원 전송

이 단계에서는 대상 WebSphere Commerce Server에서 임시 디렉토리를 작성하고 보너스 점수 자원을 이 디렉토리에 복사합니다. 그 다음 단계에서는 여러 유형의 코드를 WebSphere Commerce 응용프로그램 내의 적정 위치에 배치할 것입니다.

개발 시스템에서 대상 WebSphere Commerce Server로 파일을 복사하려면 다음을 수행하십시오.

1. 대상 WebSphere Commerce Server에서 *drive*:\ImportTemp라고 하는 임시 디렉토리를 작성하십시오.
2. 컴퓨터 사이에 파일을 복사할 방법을 판별하십시오. 대상 WebSphere Commerce Server의 드라이브를 개발 시스템에 맵핑하거나 FTP 응용프로그램(구성된 경우)을 사용하여 수행할 수 있습니다.
3. 개발 시스템에서 *drive*:\ExportTemp 내용을 대상 WebSphere Commerce Server의 *drive*:\ImportTemp에 복사하십시오.

대상 WebSphere Commerce Server 중지

전개 단계를 시작하기 전에 대상 WebSphere Commerce Server를 중지해야 합니다. WebSphere Commerce Server 중지에 대한 자세한 정보는 *WebSphere Commerce Studio 설치 안내서*를 참조하십시오.

대상 WebSphere Commerce Server에서 데이터베이스 갱신

대상 데이터베이스를 갱신하기 전에 사용자 정의된 로직을 전개하고 있는 상점의 상점 엔티티 ID를 검증하십시오. 이 값을 결정하는 데 다음 SQL문을 사용할 수 있습니다.

```
select STOREENT_ID from STOREENT where IDENTITY='FashionFlow_name'
```

여기서, *FashionFlow_name*은 코드를 전개하고 있는 상점의 이름입니다.

뷰 등록

DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 MyNewView를 등록하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령행 도구 > 명령 센터**)를 여십시오.
2. **도구** 메뉴에서 **도구 설정**을 선택하십시오.
3. **명령문 종료 문자 사용** 선택란을 선택한 후 지정된 문자가 세미콜론(**;**)인지 확인하십시오.
4. 도구 설정을 닫으십시오.
5. 스크립트 탭을 선택하고, 스크립트 창에 다음 정보를 입력하여 VIEWREG 테이블에 필수 항목을 작성하십시오.

```
connect to targetDB user dbuser
using dbpassword;
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
values ('MyNewView',-1, FF_storeent_ID,
   'com.ibm.commerce.command.ForwardViewCommand',
   'com.ibm.commerce.command.HttpForwardViewCommandImpl',
   'docname=MyNewJSPTemplate.jsp','This is my new view for tutorial 1',
   0, null);
```

여기서,

- *targetDB*는 대상 데이터베이스의 이름입니다.
- *dbuser*는 데이터베이스 사용자입니다.
- *dbpassword*는 데이터베이스 사용자의 암호입니다.
- *FF_storeent_ID*는 FashionFlow 견본 상점을 기초로 하는 사용자 상점의 고유 식별자입니다.

실행 아이콘을 누르십시오. 명령 센터를 열린 상태로 두십시오.

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 사용자 뷰를 데이터베이스에 등록하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작 > 프로그램 > Oracle > Application Development > SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 SQL문을 입력하십시오.

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
values ('MyNewView',-1, FF_storeent_ID,
   'com.ibm.commerce.command.ForwardViewCommand',
   'com.ibm.commerce.command.HttpForwardViewCommandImpl',
   'docname=MyNewJSPTemplate.jsp','This is my new view for tutorial 1',
   0, null);
```

   여기서,

   - *FF_storeent_ID*는 FashionFlow 견본 상점을 기초로 하는 사용자 상점의 고유 식별자입니다.

   SQL문을 실행하려면 Enter를 누르십시오.
6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

   Enter를 눌러 SQL문을 실행하십시오.

이제 MyNewView가 등록되었습니다.

새 컨트롤러 명령 등록

MyNewControllerCmd를 등록하려면 다음을 수행하십시오.

1. DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 MyNewControllerCmd를 등록하십시오.
   a. 명령 센터에서 스크립트 탭을 선택하고 스크립트 창에 다음 정보를 입력하여 URLREG 테이블에 필수 필드를 작성하십시오.

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS, DESCRIPTION,
   AUTHENTICATED) values ('MyNewControllerCmd',FF_storeent_ID,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   0, 'This is a new controller command for tutorial one.',null);
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FF_storeent_ID,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   'This is a new controller command for tutorial one.',
   'com.ibm.commerce.sample.commands.MyNewControllerCmdImpl',
   'local');
```

   여기서,

   - *targetDB*는 대상 데이터베이스의 이름입니다.
   - *dbuser*는 데이터베이스 사용자입니다.
   - *dbpassword*는 데이터베이스 사용자의 암호입니다.
   - *FF_storeent_ID*는 FashionFlow 견본 상점을 기초로 하는 사용자 상점의 고유 식별자입니다.

   실행 아이콘을 누르십시오. 명령 센터를 열린 상태로 두십시오.

2. Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 MyNewControllerCmd를 등록하십시오.
   a. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).
   b. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
   c. **암호** 필드에 Oracle 암호를 입력하십시오.
   d. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
   e. SQL Plus 창에서 다음 SQL문을 입력하십시오.

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS, DESCRIPTION,
   AUTHENTICATED) values ('MyNewControllerCmd',FF_storeent_ID,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   0, 'This is a new controller command for tutorial one.',null);
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FF_storeent_ID,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   'This is a new controller command for tutorial one.',
   'com.ibm.commerce.sample.commands.MyNewControllerCmdImpl','local');
```

여기서,

- *FF_storeent_ID*는 FashionFlow 견본 상점을 기초로 하는 사용자 상점의 고유 식별자입니다.

SQL문을 실행하려면 Enter를 누르십시오.

f. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

Enter를 눌러 SQL문을 실행하십시오.

XBONUS 테이블 작성

이 단계에서는 대상 WebSphere Commerce Server에서 사용하는 데이터베이스에 XBONUS 테이블을 작성합니다.

DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 테이블을 작성하십시오.

1. 스크립트 창에서 다음을 입력하십시오.

```
create table XBONUS (MEMBERID BIGINT NOT NULL,
    BONUSPOINT INTEGER NOT NULL, constraint p_xbonus
           primary key (MEMBERID),
    constraint f_xbonus foreign key (MEMBERID)
    references users (users_id) on delete cascade)
```

여기서,

- *targetDB*는 대상 데이터베이스의 이름입니다.
- *dbuser*는 데이터베이스 사용자입니다.
- *dbpassword*는 데이터베이스 사용자의 암호입니다.

실행 아이콘을 누르십시오.

XBONUS 테이블이 작성됩니다.

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 테이블을 작성하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 SQL문을 입력하십시오.

```
create table XBONUS (MEMBERID NUMBER NOT NULL,
    BONUSPOINT INTEGER NOT NULL, constraint p_xbonus
           primary key (MEMBERID),
    constraint f_xbonus foreign key (MEMBERID)
    references users (users_id) on delete cascade);
```

   Enter를 눌러 SQL문을 실행하십시오. XBONUS 테이블이 작성됩니다.

6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

   Enter를 눌러 SQL문을 실행하십시오.

대상 WebSphere Commerce Server에서 액세스 제어 정책 로드

이 단계에서는 새 자원의 액세스 제어 정책을 대상 WebSphere Commerce Server에 로드합니다.

새 정책을 로드하려면 다음을 수행하십시오.

1. 다음과 같이 대상 WebSphere Commerce Server 특정 값을 반영하도록 액세스 제어 정책을 갱신하십시오.
   a. 다음과 같이 FashionFlow 상점의 구성원 ID 값을 판별하십시오.
      - DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하십시오.
        1) 개발 데이터베이스에 연결하십시오.

2) 다음 SQL문을 실행하십시오.

```
select member_id from storeent where storeent_id=FF_storeent_ID
```

여기서, *FF_storeent_ID*는 FashionFlow 상점의 상점 엔티티 ID입니다. 예를 들어, 다음을 입력할 수 있습니다.

```
select member_id from storeent where storeent_id=10001
```

구성원 ID 값을 쓰십시오: _____

- Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하십시오.

1) Oracle SQL Plus 명령 창을 열어 전개 데이터베이스에 연결하십시오.

2) 다음을 입력하십시오.

```
column member_id format
999999999999999999999999999999999999(39개의 9)
```

이것은 전체 값을 뷰에 충분한 member_id 열 너비 표시를 설정합니다.

3) 다음 SQL문을 실행하십시오.

```
select member_id from storeent where storeent_id=FF_storeent_ID
```

여기서, *FF_storeent_ID*는 FashionFlow 상점의 상점 엔티티 ID입니다. 예를 들어, 다음을 입력할 수 있습니다.

```
select member_id from storeent where storeent_id=10001
```

구성원 ID 값을 쓰십시오: _____

4) 다음을 입력하여 member_id 열의 표시 너비를 재설정하십시오.

```
column member_id clear
```

b. 다음 디렉토리를 탐색하십시오.
drive:\ImportTemp\ACPolicies

c. 텍스트 편집기를 사용하여 SampleACPolicy_template.xml 파일을 열고 *FashionFlowMemberId*를 1a단계에서 결정한 Fashion Flow 상점의 구성원 ID 값으로 바꾸십시오. 수정한 파일을 SampleACPolicy.xml로 저장하십시오.

d. 텍스트 편집기를 사용하여 SampleACPolicy_template_en_US.xml 파일을 열고 *FashionFlowMemberId*를 1a단계에서 결정한 Fashion Flow 상점의 구성원 ID 값으로 바꾸십시오. 수정한 파일을 SampleACPolicy_en_US.xml 로 저장하십시오.

2. 다음의 액세스 제어 정책을 *WC_installdir*\xml\policies\xml 디렉토리로 복사하십시오.
   - MyNewViewACPolicy.xml
   - MyNewControllerCmdACPolicy.xml
   - SampleACPolicy.xml
   - SampleACPolicy_en_US.xml
3. 명령 프롬프트에서
   WC_installdir\bin 디렉토리로 이동하십시오.
4. 다음 양식의 acpload 명령을 실행해야 합니다.

   ```
   acpload targetDB dbuser dbpassword inputXMLFile
   ```

   여기서,
   - *targetDB*는 개발 데이터베이스의 이름입니다.
   - *dbuser*는 데이터베이스 사용자의 이름입니다.
   - *dbpassword*는 데이터베이스 사용자의 암호입니다.
   - *inputXMLFile*은 액세스 제어 정책 스펙을 포함하는 XML 파일입니다. 이 경우, `MyNewViewACPolicy.xml`을 지정하십시오.

   다음은 변수가 지정된 명령의 예입니다.

   ```
   acpload Demo_Dev db2admin db2admin MyNewViewACPolicy.xml
   ```
5. 다음 액세스 제어 정책 각각에 대해 4단계를 반복하십시오.
   - MyNewControllerCmdACPolicy.xml
   - SampleACPolicy.xml
6. 정책 설명(SampleACPolicy_en_US.xml 파일에 포함된)을 로드하려면 다음 양식의 acpnlsload 명령을 실행해야 합니다.

   ```
   acpnlsload db_name db_user db_password inputXMLFile
   ```

예를 들어, 다음과 같은 명령을 발행할 수 있습니다.

```
acpnlsload Demo_dev user password SampleACPolicy_en_US.xml
```

7. 액세스 제어 정책이 로드되는 동안 작성되었을 오류 로그에 대한 *WC_installdir*를 확인하십시오.

대상 WebSphere Commerce Server에서 상점 자원 갱신

이 단계에서는 다음과 같이 상점을 수정한 상점 자원으로 갱신합니다.

1. *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear\Stores.war 디렉토리를 백업하십시오(여기서, *cellName*는 대개 사용자 시스템의 호스트 이름이며 *instanceName*은 WebSphere Commerce 인스턴스의 이름입니다).
2. *drive*:\ImportTemp\Stores\Web Content 디렉토리로 이동하십시오.
3. *FashionFlow_name* 및 WEB-INF 폴더를 다음 디렉토리로 복사하십시오. *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear\Stores.war

대상 WebSphere Commerce Server에서 명령 및 데이터 bean JAR 파일 갱신

이 단계에서는 다음과 같이 새 명령과 데이터 bean JAR 파일을 사용하도록 대상 WebSphere Commerce Server를 갱신합니다.

1. 다음과 같이 기존 JAR 파일의 백업 사본을 작성해야 합니다.
   a. *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 디렉토리로 이동하십시오.
   b. WebSphereCommerceServerExtensionsLogic.jar 파일의 사본을 작성하고 백업 위치에 저장하십시오.
2. *drive*:\ImportTemp 디렉토리의 새 WebSphereCommerceServerExtensionsLogic jar 파일을 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 디렉토리에 복사하십시오.

여기서, *instanceName*은 WebSphere Commerce 인스턴스 이름입니다.

대상 WebSphere Commerce Server에서 EJB JAR 파일 갱신

이 단계에서는 다음과 같이 새 EJB JAR 파일을 사용하도록 대상 WebSphere Commerce Server를 갱신합니다.

1. 다음과 같이 기존 JAR 파일의 백업 사본을 작성해야 합니다.
   a. *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 디렉토리로 이동하십시오.
   b. WebSphereCommerceServerExtensionsData.jar 파일의 사본을 작성하고 백업 위치에 저장하십시오.
2. *drive*:\ImportTemp 디렉토리의 WebSphereCommerceServerExtensionsData.jar 파일을 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 디렉토리에 복사하십시오.
3. 다음과 같이 EJB 전개 설명자 정보를 수정해야 합니다.
   a. WebSphere Application Server 셀의 전개 저장소(META-INF 디렉토리)를 찾으십시오. 일반적으로 양식은 다음과 같습니다.
      WAS_installdir\config\cells*cellName* \applications\WC_*instance_name*.ear\deployments\ WC_*instance_name**EJBModuleName*.jar\META-INF
      특정 예는
      D:\WebSphere\AppServer\config\cells\myCell\applications\WC_demo.ear\deployments\WC_demo\WebSphereCommerceServerExtensionsData.jar\META-INF입니다.
   b. 디렉토리에는 다음 파일이 있습니다.
      - ejb-jar.xml
      - ibm-ejb-access-bean.xmi
      - ibm-ejb-jar-bnd.xmi
      - ibm-ejb-jar-ext.xmi
      - MANIFEST.MF

      이러한 모든 파일의 백업
   c. 도구를 사용하여 새 WebSphereCommerceServerExtensionsData.jar 파일을 열고 그 내용을 보십시오.
   d. 이 WebSphereCommerceServerExtensionsData.jar 파일에서 meta-inf 디렉토리(앞에 표시된 파일)의 내용을 3a단계의 디렉토리로 추출하십시오. 파일을 추출한 후에 디렉토리 구조가 올바르게 되어있는지 확인하십시오.

4. 명령행에서 WebSphere Application Server startServer 명령을 사용하여 WebSphere Commerce 인스턴스를 다시 시작하십시오. 이 인스턴스의 시작 및 중지에 대한 정보는 사용자 플랫폼 및 데이터베이스 *WebSphere Commerce 설치 안내서*를 참조하십시오.

대상 WebSphere Commerce Server에서 보너스 점수 로직 검증

이 단계에서는 다음을 수행하여 보너스 점수 로직이 대상 WebSphere Commerce Server로 전개되었는지를 검증합니다.

1. 웹 브라우저를 열고 URL을 입력하여 FashionFlow 견본 상점을 기반으로 하는 사용자 상점을 시작하십시오.
2. 다음을 수행하여 새 등록된 사용자를 작성하십시오.
   a. **등록**을 누르십시오.
   b. **등록**을 다시 눌러 새 고객을 작성하십시오.
   c. 등록 양식에서 모든 필수 필드에 해당 값을 입력하십시오. 예를 들어, 전자 우편 필드에 `tester@mycompany`를 입력하십시오. 전자 우편 주소의 값을 쓰십시오: _____.
   d. 값이 입력되면 **제출**을 누르십시오.
3. 로그인이 완료되면 같은 브라우저에 다음 URL을 입력하십시오.

   ```
   http://hostname/webapp/wcs/stores/servlet/MyNewControllerCmd?
      input1=user_e-mail&input2=1000
   ```

   여기서, *hostname*은 호스트 이름이며 *user_e-mail*는 2단계에서 작성한 사용자의 전자 우편 주소입니다. 사용자가 보너스 포인트 대차 대조를 갱신할 수 있는 새 양식과 이전의 모든 출력 매개변수가 들어 있는 페이지가 함께 표시됩니다.

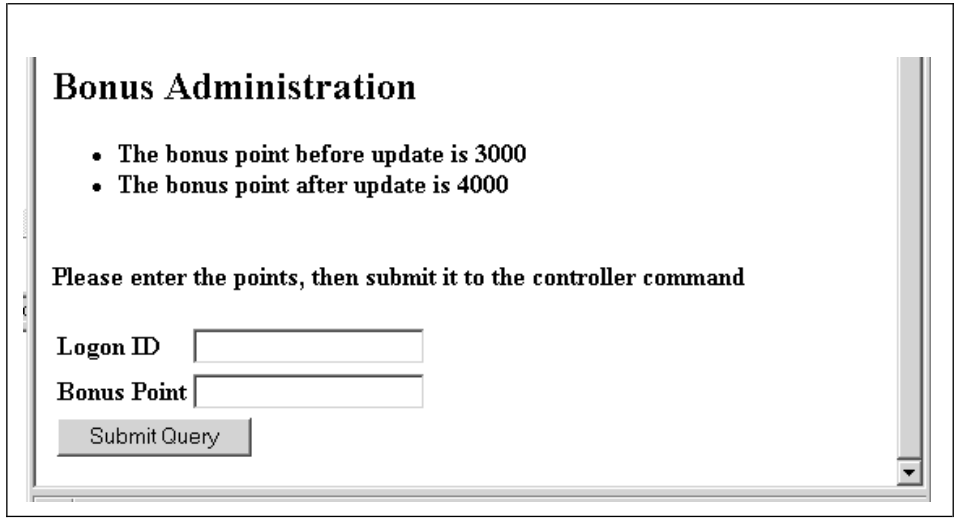


그림 43.

4. 이제 **로그온 ID** 필드에 사용자의 전자 우편 주소를 입력하고, **보너스 점수** 필드에 500을 입력하십시오. **제출**을 누르십시오. 보너스 점수 잔고가 갱신되었음을 표시하는 다음과 유사한 페이지가 표시됩니다.

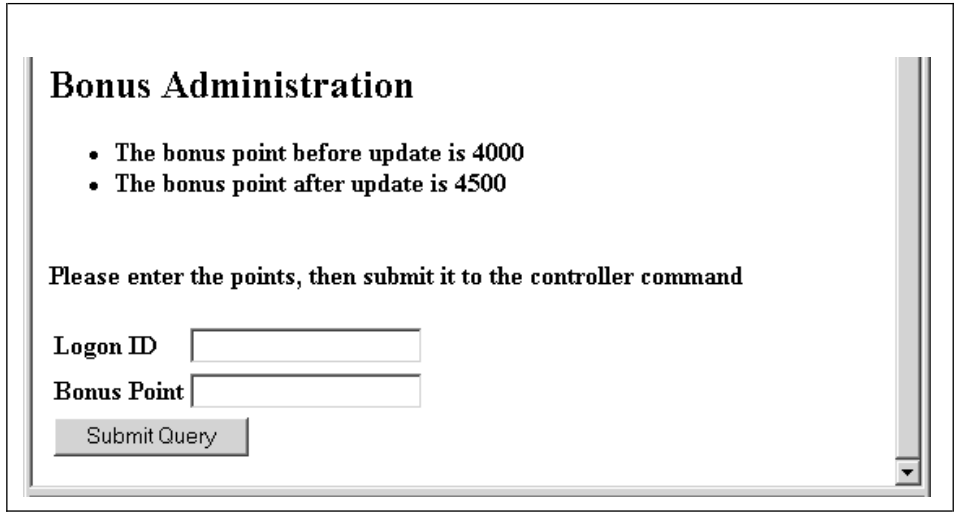


그림 44.

제 11 장 학습: 기존 컨트롤러 명령 수정

이 학습의 목적은 기존 컨트롤러 명령을 수정하는 데 사용되는 프로세스를 보여주는 것입니다.

이 학습에서 고객 장바구니의 항목 수는 5개 이하로 제한됩니다. 이 솔루션을 구현하려면 장바구니의 항목 수를 확인하기 위한 로직을 포함하는 사용자 고유의 구현으로 OrderItemAddCmdImpl을 대체합니다. 고객이 6번째의 항목을 장바구니에 추가하려고 할 경우, 예외가 발생합니다. 이 예외에는 새 오류 메시지가 사용됩니다.

이 학습의 목적은 기존 명령 로직을 수정하는 데 사용되는 전개 프로세스를 설명하는 것임을 유의하십시오. 장바구니에서 제한되는 항목의 모든 예를 설명하지는 않습니다. 이 학습에 사용된 로직은 학습 목적으로 단순화되었습니다.

이 학습에서는 다음을 학습하게 됩니다.

- 기존 컨트롤러 명령의 새 구현을 작성하는 방법
- 새 구현이 응용프로그램에서 사용되도록 명령 레지스트리를 갱신하는 방법
- 수정한 컨트롤러 명령을 기존 WebSphere Commerce 응용프로그램에 전개하는 방법

전제 조건

이 학습에서는 251 페이지의 제 10 장 『학습: 새 비즈니스 로직 작성』을 완료하지 않아도 됩니다. 학습을 완료한 경우, 코드를 작업 영역에 두어도 이 학습과 충돌하지 않으므로 문제가 되지 않습니다.

이 학습을 시작하기 전에 FashionFlow 견본 상점을 기초로 상점을 공개해야 합니다. 이 상점 내에서 구매를 완료할 수 있어야 합니다(예를 들어, 카탈로그를 찾아보고 항목을 장바구니에 추가한 후 주문을 시작하여 주문 확인이 표시되어야 합니다).

전개 단계를 완료하려면 상점이 대상 WebSphere Commerce Server에도 있어야 합니다.

새 MyOrderItemAddCmdImpl 클래스 작성

학습의 이 단계에서 새 MyOrderItemAddCmdImpl 클래스를 작성합니다. 이 클래스를 작성하려면 다음을 수행하십시오.

1. WebSphere Studio Application Developer에서 Java perspective를 여십시오 (**창 > Perspective 열기 > Java**).
2. **WebSphereCommerceServerExtensionsLogic** 프로젝트를 탐색하십시오.
3. **src** 디렉토리를 탐색하십시오.
4. 251 페이지의 제 10 장『학습: 새 비즈니스 로직 작성』을 완료하지 않은 경우, **src** 디렉토리에서 마우스 오른쪽 버튼을 누르고 **새로 만들기 > 패키지**를 선택하십시오.
   새 Java 패키지 마법사가 열립니다. 이 마법사에서 다음을 수행하십시오.
   a. **이름** 필드에 com.ibm.commerce.sample.commands를 입력하십시오.
   b. **완료**를 누르십시오.
5. **com.ibm.commerce.sample.commands** 패키지에서 마우스 오른쪽 버튼을 누르십시오. **새로 만들기 > 클래스**를 선택하십시오.
   새 Java 클래스 마법사가 열립니다.
6. 새 Java 클래스 마법사에서 다음을 수행하십시오.
   a. **이름** 필드에 MyOrderItemAddCmdImpl을 입력하십시오.
   b. 상위 클래스를 지정하려면 상위 클래스 필드 옆에 있는 **찾아보기** 버튼을 누르고 OrderItemAddCmdImpl을 입력하십시오. **확인**을 누르십시오.
   c. 구현할 인터페이스를 지정하려면 **추가**를 누르고 OrderItemAddCmd를 입력한 후 **확인**을 누르십시오.
   d. **완료**를 누르십시오.

MyOrderItemAddCmdImpl 클래스의 소스 코드가 표시됩니다.

다음 단계는 다음과 같이 이 클래스에서 반입 명령문을 갱신하는 것입니다.

1. 아웃라인 보기에서 선택하여 **반입 선언**을 여십시오. 두 개의 반입 명령문이 이미 작성되어 있음을 알게 됩니다.
2. 반입 명령문의 소스 코드에 다음 반입 명령문을 추가하십시오.

```
import com.ibm.commerce.exception.ECApplicationException;
import com.ibm.commerce.exception.ECException;
import com.ibm.commerce.exception.ECSystemException;
import com.ibm.commerce.order.objects.OrderAccessBean;
import com.ibm.commerce.ras.ECMessage;import com.
ibm.commerce.sample.messages.MyNewMessages;
```

3. 변경사항을 저장하십시오.

다음 단계는 비즈니스 로직 및 예외 처리를 추가하여 주문 항목을 더 추가하기 전에 고객 장바구니에 6개 이상의 항목이 있는지 판별하는 것입니다. 다음과 같이 코드를 갱신하십시오.

1. 아웃라인 보기에서 MyOrderItemAddCmdImpl 클래스를 선택하고 해당되는 소스 코드를 보십시오. 소스 코드에는 현재 다음 사항만 있습니다.

```
public class MyOrderItemAddCmdImpl
 extends OrderItemAddCmdImpl
 implements OrderItemAddCmd {

}
```

2. 이 클래스에 새 performExecute 메소드를 추가해야 합니다. 이 메소드에는 장바구니의 항목 수를 확인하기 위한 로직이 포함되며 항목 수가 5 미만이면 정상적으로 상위 클래스(OrderItemAddCmdImpl)의 performExecute 메소드가 호출됩니다. 항목이 5개 이상이면 예외가 발생하고 사용자는 더이상 항목을 장바구니에 추가할 수 없습니다. 이 메소드를 추가하려면 다음 소스 코드를 클래스에 복사하십시오(클래스 끝을 표시하는 마지막 종료 중괄호 "}" 이전에 포함되도록 하십시오).

```
public void performExecute() throws ECException {
  // Get a list of order ids
  String[] orderIds = getOrderId();

  // Check to make sure that an id exists at all
  // if order id exists then get number of items in the order
  // else if no order id exists then execute normal code
  if (orderIds != null && orderIds.length > 0) {
   // An exception should be thrown when trying to add a sixth item
   // to the cart.  Since this code is run before any items are added
   // throw and exception if there are 5 or more items in the cart
```

```
   if (itemsInOrder(orderIds[0]) >= 5) {
          throw new ECApplicationException(
     MyNewMessages._ERR_TOO_MANY_ITEMS,
     this.getClass().getName(),
      "performExecute");
   }
   // else perform normal flow
  }
  super.performExecute();
 }
 // get number of items in the order
 protected int itemsInOrder(String orderId) throws ECException {
  try {
   OrderAccessBean order = new OrderAccessBean();
   order.setInitKey_orderId(orderId);
   order.refreshCopyHelper();
   return order.getOrderItems().length;
} catch (javax.ejb.FinderException e) {
   throw new ECSystemException(
    ECMessage._ERR_FINDER_EXCEPTION,
    this.getClass().getName(),
    "itemsInOrder");
} catch (javax.naming.NamingException e) {
   throw new ECSystemException(
    ECMessage._ERR_NAMING_EXCEPTION,
    this.getClass().getName(),
    "itemsInOrder");
} catch (java.rmi.RemoteException e) {
   throw new ECSystemException(
    ECMessage._ERR_REMOTE_EXCEPTION,
    this.getClass().getName(),
    "itemsInOrder");
} catch (javax.ejb.CreateException e) {
   throw new ECSystemException(
    ECMessage._ERR_CREATE_EXCEPTION,
    this.getClass().getName(),
    "itemsInOrder");
  }
 }
```

새 코드를 클래스에 붙여넣은 다음 이 소스 코드에서 오른쪽 마우스 버튼을 누른 후 **포맷**을 눌러 코드를 포맷하십시오.

3. 작업을 저장하십시오.

   주: 메시지 정보가 누락되었다는 경고입니다. 이것은 다음 단계에서 수정됩니다.

메시지 정보 작성

새 명령 구현에는 새 오류 메시지, _ERR_TOO_MANY_ITEMS가 사용됩니다. 이 절에서는 새 메시지의 코드와 이와 관련된 특성 파일을 작성합니다. 이 코드를 반입하려면 다음을 수행하십시오.

1. **WebSphereCommerceServerExtensionsLogic** 프로젝트를 펼치십시오.
2. **src** 디렉토리를 오른쪽 마우스 버튼으로 누른 후 **새로 만들기 > 패키지**를 선택하십시오.
   새 Java 패키지 마법사가 열립니다. 이 마법사에서 다음을 수행하십시오.
   a. **이름** 필드에 com.ibm.commerce.sample.messages를 입력하십시오.
   b. **완료**를 누르십시오.
3. **com.ibm.commerce.sample.messages** 패키지에서 마우스 오른쪽 버튼을 누르십시오. **새로 만들기 > 클래스**를 선택하십시오.
   새 Java 클래스 마법사가 열립니다.
4. 새 Java 클래스 마법사에서 다음을 수행하십시오.
   a. **이름** 필드에 MyNewMessages를 입력하십시오.
   b. **완료**를 누르십시오.
5. **MyNewMessages** 클래스를 두 번 눌러 이 클래스의 소스 코드를 보십시오.
6. 코드의 public class MyNewMessages 행 바로 앞에 다음 반입 명령문을 추가하십시오.

```
import com.ibm.commerce.ras.ECMessage;import
com.ibm.commerce.ras.ECMessageSeverity;
import com.ibm.commerce.ras.ECMessageType;
```

7. 클래스에 다음 코드를 추가하십시오.

```
// Resouce bundle used to extract the text for an exception
 static final String errorBundle = "MyNewErrorMessages";

 // An ECMessage is used to describe an ECException and is passed
 // into the ECException when thrown
 public static final ECMessage _ERR_TOO_MANY_ITEMS =
      new ECMessage(ECMessageSeverity.ERROR, ECMessageType.USER,
      MyNewMessageKeys._ERR_TOO_MANY_ITEMS, errorBundle);
```

8. 변경사항을 저장하십시오.

9. **com.ibm.commerce.sample.messages** 패키지에서 마우스 오른쪽 버튼을 누르십시오. **새로 만들기 > 클래스**를 선택하십시오.
   새 Java 클래스 마법사가 열립니다.
10. 새 Java 클래스 마법사에서 다음을 수행하십시오.
    a. **이름** 필드에 MyNewMessageKeys를 입력하십시오.
    b. **완료**를 누르십시오.
11. 다음과 같이 클래스에 코드를 정의하십시오.

```
public class MyNewMessageKeys {
// This class defines the keys used to create new exceptions that are
// thrown by customized code.
 public static final String _ERR_TOO_MANY_ITEMS = "_ERR_TOO_MANY_ITEMS";
}
```

12. 변경사항을 저장하십시오.
13. **WebSphereCommerceServerExtensionsLogic** 프로젝트에서 마우스 오른쪽 버튼을 누르고 **프로젝트 빌드**를 선택하여 코드를 컴파일하십시오.
14. 다음과 같이 메시지 정보를 포함할 새 특성 파일을 작성하십시오.
    a. 웹 perspective를 여십시오(**창 > Perspective 열기 > 웹**).
    b. **상점** 웹 프로젝트에서 **웹 컨텐츠 > WEB-INF > 클래스** 폴더를 펼치십시오.
    c. **클래스** 폴더에서 마우스 오른쪽 버튼을 누른 후 **새로 만들기 > 기타 > 단순 > 파일 > 다음**을 선택하여 새 특성 파일을 작성하십시오.
       새 파일 창이 열립니다.
    d. **파일 이름** 필드에 MyNewErrorMessages.properties를 입력한 후 **완료**를 누르십시오.
       비어 있는 새 파일이 열립니다.
    e. 새 파일로 다음 텍스트를 복사하십시오.

```
_ERR_TOO_MANY_ITEMS=You are trying to place too many different items
   in one shopping cart.
```

       주: 앞의 코드의 행 바꾸기는 표시 목적일 뿐입니다. 단일 행에 텍스트를 입력하십시오.
    f. 변경사항을 저장하십시오.

명령 레지스트리 수정

이 단계에서는 새 MyOrderItemAddCmdImpl 구현 클래스가 원래 OrderItemAddCmdImpl 구현 클래스 대신 사용되도록 명령 레지스트리를 수정합니다. 명령 레지스트리에서 수정해야 하는 유일한 테이블은 CMDREG 테이블입니다. 이 경우, 새 구현 클래스는 모든 상점에 사용됩니다.

명령 레지스트리를 수정하려면 다음을 수행하십시오.

1. DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 MyOrderItemAddCmdImpl을 등록하십시오.
   a. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령 센터**)를 여십시오.
   b. **도구** 메뉴에서 **도구 설정**을 선택하십시오.
   c. **명령문 종료 문자 사용** 선택란을 선택한 후 지정된 문자가 세미콜론(**;**)인지 확인하십시오.
   d. 스크립트 탭을 선택하고 스크립트 창에 다음 정보를 입력하여 URLREG 테이블에 필수 항목을 작성하십시오.

      ```
      connect to developmentDB user dbuser using dbpassword;
      update CMDREG
      set CLASSNAME='com.ibm.commerce.sample.commands.MyOrderItemAddCmdImpl'
      WHERE INTERFACENAME='com.ibm.commerce.orderitems.commands.OrderItemAddCmd'
      and storeent_Id=0;
      ```

      여기서,

      - *developmentDB*는 개발 데이터베이스의 이름입니다.
      - *dbuser*는 데이터베이스 사용자입니다.
      - *dbpassword*는 데이터베이스 사용자의 암호입니다.

      실행 아이콘을 누르십시오.

2. Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 MyOrderItemAddCmdImpl을 등록하십시오.
   a. Oracle SQL Plus 명령창을 여십시오(**시작 > 프로그램 > Oracle > Application Development > SQL Plus**).
   b. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
   c. **암호** 필드에 Oracle 암호를 입력하십시오.

d. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.

e. SQL Plus 창에서 다음 SQL문을 입력하십시오.

```
update CMDREG
set CLASSNAME='com.ibm.commerce.sample.commands.MyOrderItemAddCmdImpl'
WHERE INTERFACENAME='com.ibm.commerce.orderitems.commands.OrderItemAddCmd'
and storeent_Id=0;
```

SQL문을 실행하려면 Enter를 누르십시오.

f. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

Enter를 눌러 SQL문을 실행하십시오.

MyOrderItemAddCmdImpl 명령 테스트

다음 단계는 새 로직이 제대로 작동하는지 테스트하는 것입니다. 이를 테스트하려면 5개의 항목을 성공적으로 장바구니에 추가할 수 있어야 하지만, 6번째 항목을 장바구니에 추가하려고 하면 오류가 발생합니다.

새 비즈니스 로직을 테스트하려면 다음을 수행하십시오.

1. 서버 perspective로 전환하십시오(**창 > Perspective 열기 > 서버**).
2. **WebSphereCommerceServer** 서버에서 마우스 오른쪽 버튼을 누르고 **시작**(또는 **다시 시작**)을 선택하십시오.
3. Stores\Web Content*FashionFlow_name* 디렉토리에 있는 **index.jsp**에서 마우스 오른쪽 버튼을 누르고 **서버에서 실행**을 선택하십시오.
   상점 홈페이지가 웹 브라우저에 표시됩니다.
4. 상점에서 쇼핑하고 5개의 항목을 장바구니에 추가하십시오. 5번째 항목을 추가하고 나면 장바구니는 다음과 유사한 상태가 됩니다.

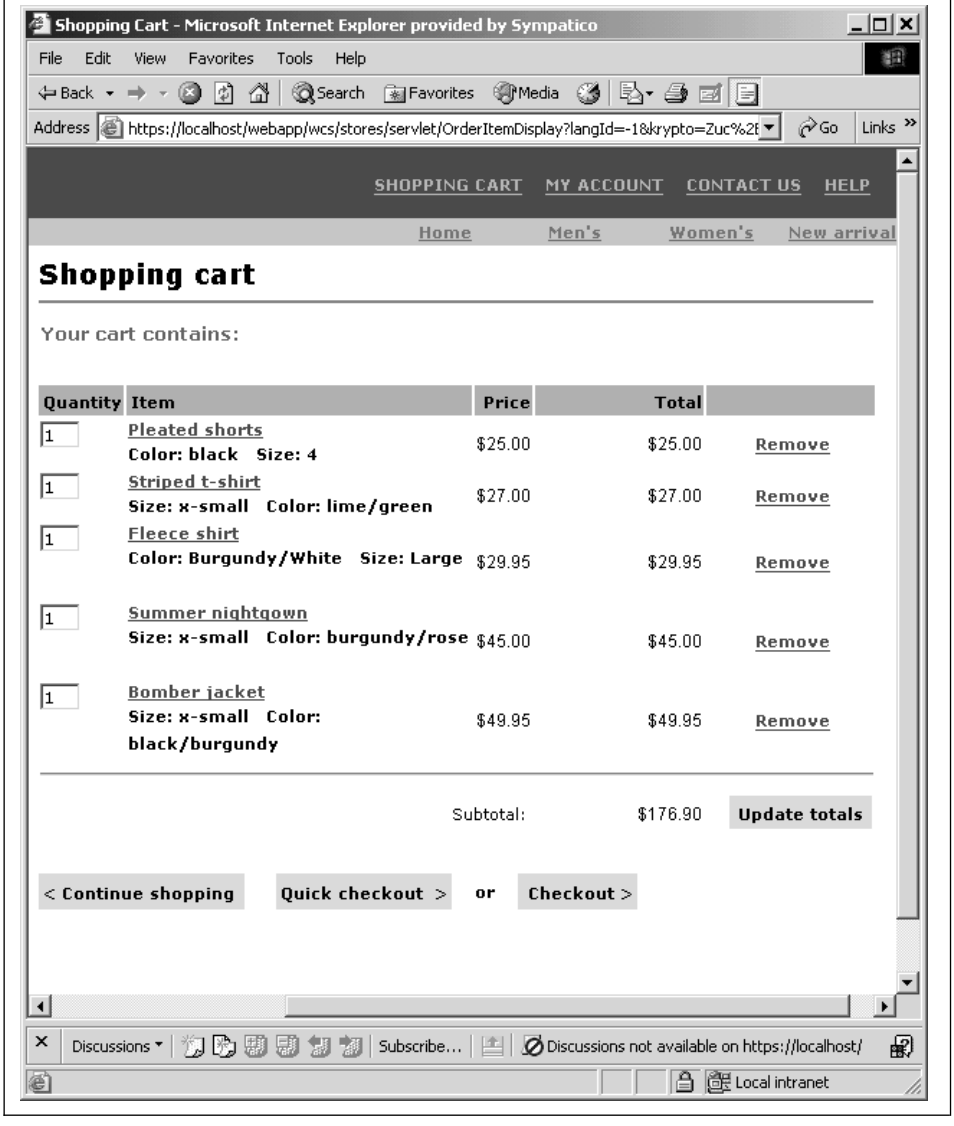


그림 45.

5. **계속 쇼핑하기**를 누르고 다른 항목을 선택하십시오. 이 선택 항목에 대해 **장바구니에 추가**를 누르십시오. 다음과 같은 오류 페이지가 표시됩니다.

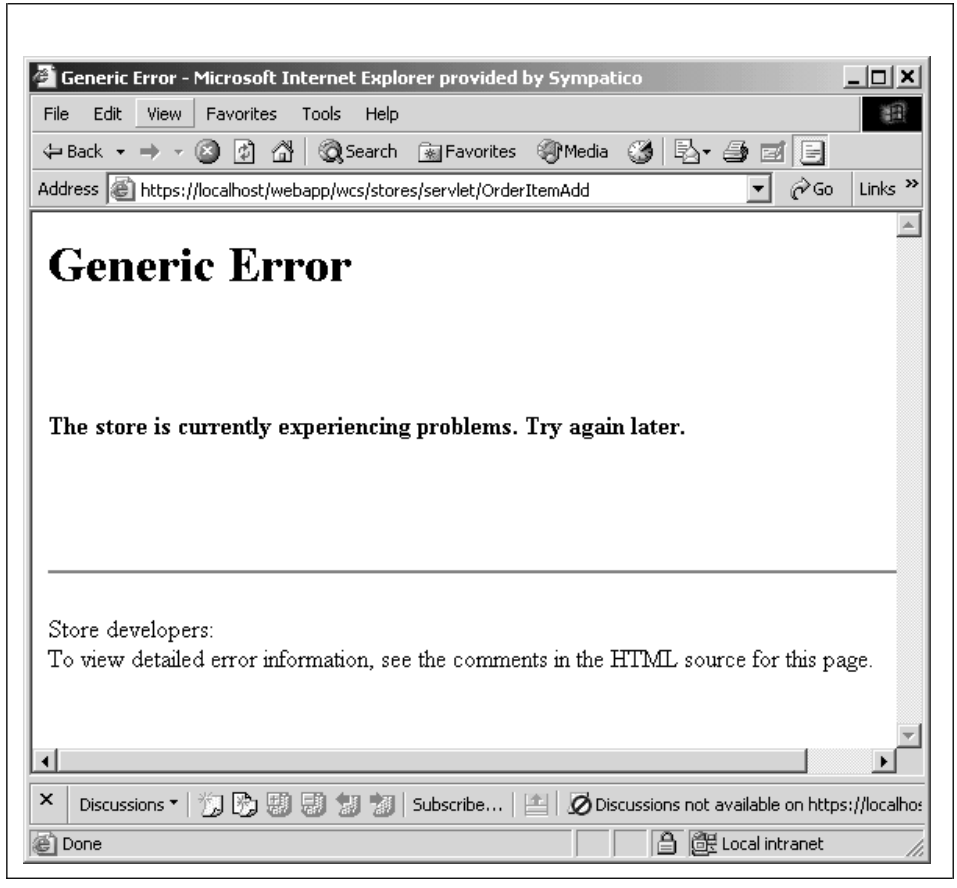


그림 46.

오류 페이지에 대한 소스를 보십시오. 소스에서 화면이동하여 다음 오류 정보를 찾으십시오.

```
Message Key: _ERR_TOO_MANY_ITEMS
Message: You are trying to place too many different items in one shopping
         cart
```

MyOrderItemAddCmdImpl 전개

이 단계에서는 수정한 비즈니스 로직을 대상 WebSphere Commerce Server에 전개합니다. 이 경우 전개는 다음의 상위-레벨 단계로 구성됩니다.

1. 명령 로직과 오류 클래스를 포함하는 JAR 파일 작성
2. 오류 메시지 특성 파일 반입

3. 대상 WebSphere Commerce Server로 자원 전송
4. 대상 WebSphere Commerce Server의 명령 레지스트리 갱신
5. 대상 WebSphere Commerce Server에서 새 로직 유효성 확인

명령 JAR 파일 작성

WebSphere Commerce 코드 사용자 정의 전략에 따라 사용자 정의한 명령 및 데이터 bean은 WebSphereCommerceServerExtensionsLogic 프로젝트에 놓입니다. 결과적으로 사용자 정의한 코드를 전개할 시간이 되면 이전에 사용자 정의한 코드를 JAR 파일에 포함시킨 것을 알게됩니다. 예를 들어, 251 페이지의 제 10 장『학습: 새 비즈니스 로직 작성』을 완료하면 MyOrderItemAddCmdImpl 클래스를 전개할 JAR 파일을 작성할 때 이전에 작성한 MyNewControllerCmd와 기타 클래스를 포함시킨 것을 알게됩니다.

MyOrderItemAddCmdImpl 클래스를 포함하는 JAR 파일을 작성하려면 개발 시스템에서 다음 단계를 수행하십시오.

1. 로컬 파일 시스템에 *drive*:\ExportTemp2라고 하는 디렉토리를 작성하십시오.
2. WebSphere Studio Application Developer를 열고 J2EE 네비게이터 보기로 전환하십시오.
3. **WebSphereCommerceServerExtensionsLogic** 프로젝트에서 마우스 오른쪽 버튼을 누른 후 **반출**을 선택하십시오.
   반출 마법사가 열립니다.
4. 반출 마법사에서 다음을 수행하십시오.
   a. **JAR 파일**을 선택하고 **다음**을 누르십시오.
   b. **반출할 자원 선택?** 아래에 있는 왼쪽 분할창이 프로젝트 이름으로 채워집니다. 이 값은 있는 그대로 두십시오.
   c. 오른쪽 분할창에서 다음 자원만이 선택되었음을 확인하십시오.
      - .classpath
      - .project
      - .serverPreference
   d. **작성된 클래스 파일 및 자원 반출**이 선택되어 있는지 확인하십시오.
   e. Java 소스 파일 및 자원 **반출**을 선택하지 *마십시오*.

f. **반출 대상 선택** 필드에서 사용할 완전한 JAR 파일 이름을 입력하십시오. 이 경우에는 *drive*:\ExportTemp2\WebSphereCommerceServerExtensionsLogic.jar을 입력하십시오. JAR 파일 이름은 WebSphereCommerceServerExtensionsLogic.jar이어야 합니다.

g. **완료**를 누르십시오.

메시지 특성 파일 반입

이 절에서 다음과 같이 새 메시지에 대한 텍스트를 포함하는 특성 파일을 반입합니다.

1. J2EE 네비게이터 보기로 전환하십시오.
2. **Stores** 폴더를 펼치십시오.
3. **Web Content** 폴더에서 마우스 오른쪽 버튼을 누르고 **반출**을 선택하십시오. 반출 마법사가 열립니다.
4. 반출 마법사에서 다음을 수행하십시오.
   a. **파일 시스템**을 선택하고 **다음**을 누르십시오.
   b. **모두 선택 취소**를 누르십시오.
   c. 다음 자원을 반출하도록 선택하십시오.
      - Web Content\WEB-INF\classes\MyNewErrorMessages.properties
   d. **파일의 디렉토리 구조 작성**을 선택하십시오.
   e. 디렉토리 필드에 자원을 놓을 임시 디렉토리를 입력하십시오. 예를 들어, C:\ExportTemp2를 입력하십시오.
   f. **완료**를 누르십시오.

대상 WebSphere Commerce Server로 자원 전송

이 단계에서는 대상 WebSphere Commerce Server에서 임시 디렉토리를 작성하고 MyOrderItemAddCmdImpl 자원을 이 디렉토리에 복사합니다.

개발 시스템에서 대상 WebSphere Commerce Server로 파일을 복사하려면 다음을 수행하십시오.

1. 대상 WebSphere Commerce Server에서 *drive*:\ImportTemp2라고 하는 임시 디렉토리를 작성하십시오.
2. 컴퓨터 사이에 파일을 복사할 방법을 판별하십시오. 대상 WebSphere Commerce Server의 드라이브를 개발 시스템에 맵핑하거나 FTP 응용프로그램(구성된 경우)을 사용하여 수행할 수 있습니다.
3. 개발 시스템에서 *drive*:\ExportTemp2 내용을 대상 WebSphere Commerce Server의 *drive*:\ImportTemp2로 복사하십시오.

대상 WebSphere Commerce Server 중지

전개 단계를 시작하기 전에 대상 WebSphere Commerce Server를 시작해야 합니다. 대상 WebSphere Commerce Server 중지에 대한 자세한 정보는 *WebSphere Commerce Studio 설치 안내서*를 참조하십시오.

대상 WebSphere Commerce Server에서 데이터베이스 갱신

이 단계에서는 새 MyOrderItemAddCmdImpl 구현 클래스를 사용하도록 명령 레지스트리를 수정합니다.

DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 MyOrderItemAddCmdImpl을 등록하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령 센터**)를 여십시오.
2. **도구** 메뉴에서 **도구 설정**을 선택하십시오.
3. **명령문 종료 문자 사용** 선택란을 선택한 후 지정된 문자가 세미콜론(**;**)인지 확인하십시오.
4. 도구 설정을 닫으십시오.
5. 스크립트 탭을 선택하고 스크립트 창에 다음 정보를 입력하여 CMDREG 테이블에 필수 항목을 작성하십시오.

```
connect to targetDB user dbuser using dbpassword;
update CMDREG
set CLASSNAME='com.ibm.commerce.sample.commands.MyOrderItemAddCmdImpl'
WHERE INTERFACENAME='com.ibm.commerce.orderitems.commands.OrderItemAddCmd'
and storeent_Id=0;
```

여기서,

- *targetDB*는 대상 WebSphere Commerce Server에서 사용하는 데이터베이스의 이름입니다.
- *dbuser*는 데이터베이스 사용자입니다.
- *dbpassword*는 데이터베이스 암호입니다.

실행 아이콘을 누르십시오.

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 MyOrderItemAddCmdImpl을 등록하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작 > 프로그램 > Oracle > Application Development > SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 SQL문을 입력하십시오.

```
update CMDREG
set CLASSNAME='com.ibm.commerce.sample.commands.MyOrderItemAddCmdImpl'
WHERE INTERFACENAME='com.ibm.commerce.orderitems.commands.OrderItemAddCmd'
and storeent_Id=0;
```

   SQL문을 실행하려면 Enter를 누르십시오.

6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

   Enter를 눌러 SQL문을 실행하십시오.

대상 WebSphere Commerce Server에서 명령 JAR 파일 갱신

이 단계에서는 다음과 같이 새 MyOrderItemAddCmdImpl을 포함하는 JAR 파일을 사용하도록 대상 WebSphere Commerce Server를 갱신합니다.

1. 명령행에서 WebSphere Application Server stopServer 명령을 사용하여 WebSphere Commerce 인스턴스를 중지하십시오. 필요한 경우, 이 인스턴스의 시작 및 중지에 대한 정보는 사용자 플랫폼 및 데이테베이스 *WebSphere Commerce 설치 안내서*를 참조하십시오.
2. 다음과 같이 기존 JAR 파일의 백업 사본을 만들어야 합니다.

a. *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 디렉토리로 이동하십시오. 여기서, *cellName*은 일반적으로 시스템의 호스트 이름입니다.

b. WebSphereCommerceServerExtensionsLogic.jar 파일의 사본을 작성하고 백업 위치에 저장하십시오.

3. *drive*:\ImportTemp2 디렉토리의
새 WebSphereCommerceServerExtensionsLogic.jar 파일을
WAS_installdir\installedApps*cellName*\WC_*instanceName*.ear 디렉토리에 복사하십시오.

여기서, *instanceName*은 WebSphere Commerce 인스턴스 이름입니다.

대상 WebSphere Commerce Server에서 메시지 특성 파일 갱신

이 단계에서는 다음과 같이 메세지 특성 파일을 응용프로그램에 추가합니다.

1. *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear\Stores.war 디렉토리를 백업하십시오(여기서, *cellName*은 대개 시스템의 호스트 이름이며 *instanceName*은 WebSphere Commerce 인스턴스 이름입니다.).
2. *drive*:\ImportTemp\Stores\Web Content 디렉토리로 이동하십시오.
3. WEB-INF 폴더를 다음 디렉토리에 복사하십시오.
WAS_installdir\installedApps*cellName*\WC_*instanceName*.ear\Stores.war
4. 명령행에서 WebSphere Application Server startServer 명령을 사용하여 WebSphere Commerce 인스턴스를 다시 시작하십시오.

여기서, *instanceName*은 WebSphere Commerce 인스턴스 이름입니다.

대상 WebSphere Commerce Server에서 MyOrderItemAddCmdImpl 로직 확인

이 단계에서는 전개된 코드가 제대로 작동하는지 검증하기 위해 빠른 확인을 수행합니다.

코드를 검증하려면 다음을 수행하십시오.

1. 웹 브라우저를 열고 FashionFlow 상점을 실행하십시오. 예를 들어, 다음 URL을 입력하면 상점이 실행됩니다.

```
http://hostname/webapp/wcs/stores/servlet/FashionFlow/index.jsp
```

여기서, *hostname*은 인스턴스의 호스트 이름입니다.

2. 상점에서 쇼핑하고 5개의 항목을 장바구니에 추가하십시오. 5번째 항목을 추가하고 나면 장바구니는 다음과 유사한 상태가 됩니다.

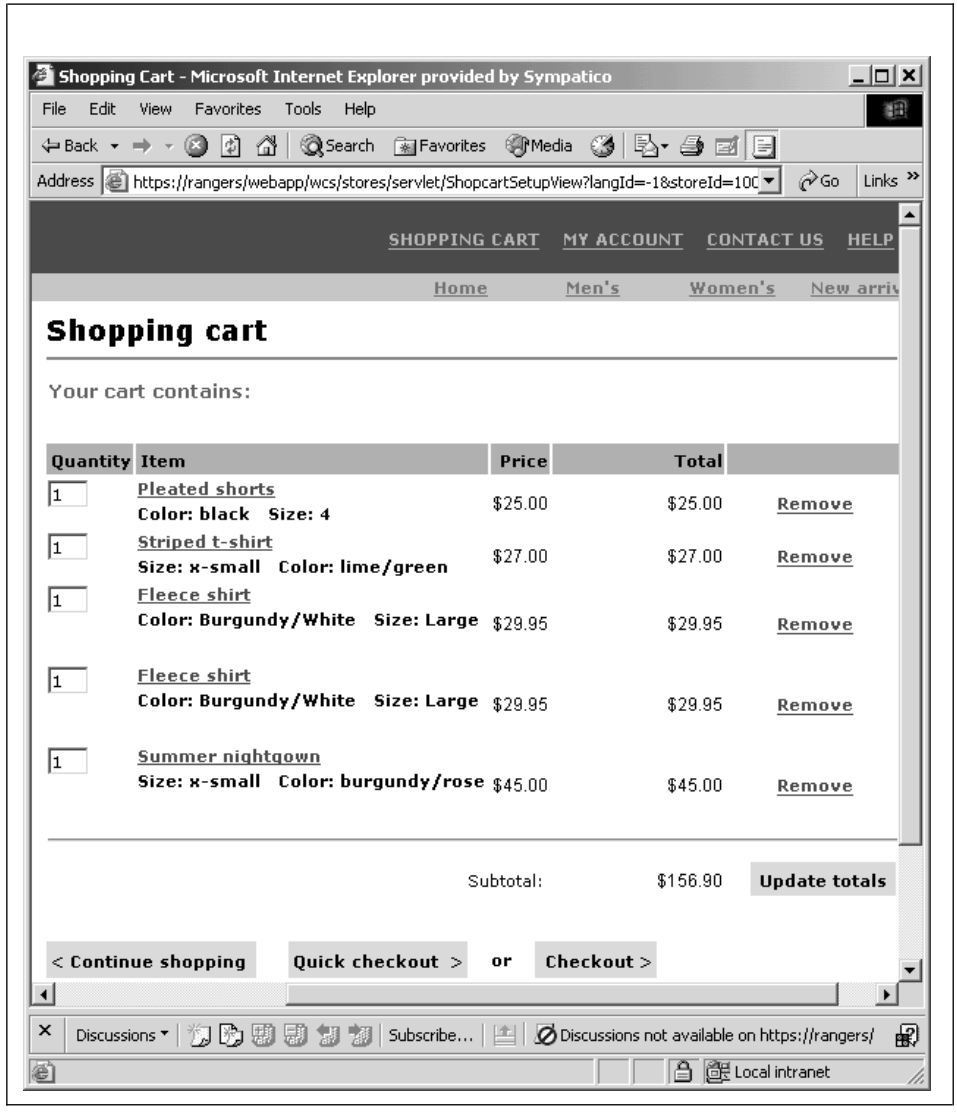


그림 47.

3. **계속 쇼핑하기**를 누르고 다른 항목을 선택하십시오. 이 선택 항목에 대해 **장바구니에 추가**를 누르십시오. 다음과 같은 오류 페이지가 표시됩니다.

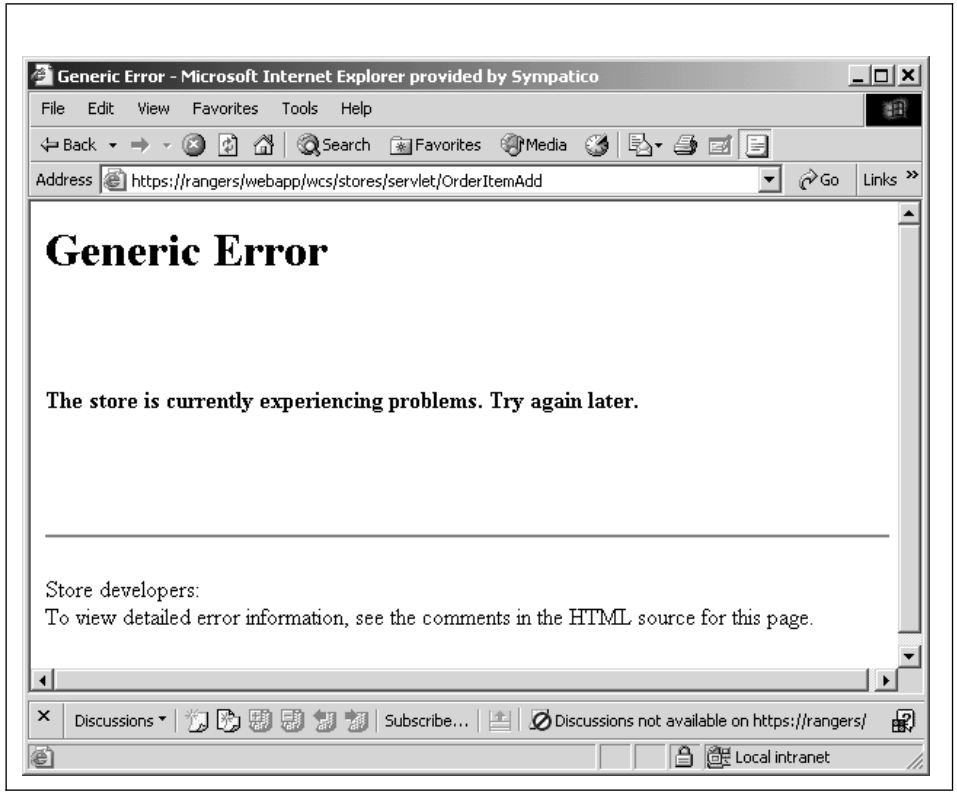


그림 48.

오류 페이지에 대한 소스를 보십시오. 소스에서 화면이동하면 오류의 메시지 키가 _ERR_TOO_MANY_ITEMS임을 알 수 있습니다.

제 12 장 학습: 오브젝트 모델 확장 및 기존 태스크 명령 수정

이 학습에서는 주문에 대한 선물 카드 정보 수집의 요구사항을 제시합니다. 수집해야 하는 정보에는 받는 사람의 이름, 보낸 사람의 이름 및 두 가지의 메시지가 포함됩니다. 정보는 고객이 주문을 제출할 때 수집됩니다.

선물 카드 정보는 데이터베이스에 저장해야 하므로, 당연히 새 데이터베이스 테이블이 필요합니다. 이는 주문 프로세스의 확장으로 두 가지 방법으로 오브젝트 모델을 수정할 수 있습니다. 일반적으로 기존 WebSphere Commerce 공용 엔티티 bean을 수정하고 수정한 bean의 새 필드를 새 테이블에 맵핑하거나, 새 테이블에 직접 맵핑하는 새 엔티티 bean을 작성할 수 있습니다. 이러한 방식에서는 주문 엔티티 bean을 확장하고 이 bean이 "find for update" 유형의 SQL 조회를 사용해야 하므로, 이 방식으로는 bean을 확장할 수 없습니다. 그런 것으로 이것은 새 테이블에 맵핑하는 새 엔티티 bean을 작성해야하는 경우입니다.

새 엔티티 bean 외에도, 기존 ExtOrderProcessCmdImpl 태스크 명령이 확장됩니다. 확장은 새 테이블에 대응하는 새 데이터 bean 인스턴스를 작성하고 데이터베이스에서 선물 정보를 갱신하는 데 사용됩니다.

이 학습에는 다음과 같은 상위-레벨 태스크가 포함됩니다.

1. 새 XORDGIFT 테이블 작성 및 대량 자료 반입
2. 새 OrderGift 엔티티 bean 작성
3. XORDGIFT 스키마 작성
4. 테이블 정의 작성 및 OrderGift 엔티티 bean 필드를 XORDGIFT 테이블 열에 맵핑
5. OrderGift bean의 전개 코드 및 액세스 bean 작성
6. 새 OrderGiftDataBean 작성
7. 새 MyExtOrderProcessCmdImpl 태스크 명령 구현 작성
8. 메시지 정보 수집을 위해 OrderSubmitForm.jsp를 수정하고 메시지 정보 표시를 위해 OrderDetailDisplayForm.jsp 수정

9. 수정한 코드 테스트

전제 조건

이 학습을 완료하지 않아도 됩니다. 학습을 완료한 경우, 코드를 작업 영역에 두어도 이 학습과 충돌하지 않으므로 문제가 되지 않습니다.

이 학습을 시작하기 전에 FashionFlow 견본 상점을 기초로 상점을 공개해야 합니다. 이 상점 내에서 구매를 완료할 수 있어야 합니다(예를 들어, 카탈로그를 찾아보고 항목을 장바구니에 추가한 후 주문을 시작하여 주문 확인이 표시되어야 합니다).

XORDGIFT 테이블 작성 및 대량 자료 반입

이 단계에서는 XORDGIFT 테이블을 작성합니다.

DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 테이블을 작성하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령행 도구 > 명령 센터**)를 열고 **스크립트** 탭을 누르십시오.
2. 스크립트 창에서 다음을 입력하십시오.

```
connect to developmentDB user dbuser using dbpassword;
create table XORDGIFT (ORDERSID bigint not null, RECEIPTNAME varchar(50),
SENDERNAME varchar(50), MSGFIELD1 varchar(50), MSGFIELD2 varchar(50),
constraint p_xordgift primary key (ORDERSID),
constraint f_xordgift foreign key (ORDERSID) references ORDERS(ORDERS_ID)
on delete cascade);
insert into XORDGIFT (ORDERSID) (select ORDERS_ID from ORDERS);
```

여기서,

- *developmentDB*는 개발 데이터베이스의 이름입니다.
- *dbuser*는 데이터베이스 사용자입니다.
- *dbpassword*는 데이터베이스 사용자의 암호입니다.

실행 아이콘을 누르십시오.
XORDGIFT 테이블이 작성됩니다.

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 테이블을 작성하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작 > 프로그램 > Oracle > Application Development > SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 SQL문을 입력하십시오.

```
create table XORDGIFT (ORDERSID NUMBER NOT NULL, RECEIPTNAME VARCHAR(50),
SENDERNAME VARCHAR(50), MSGFIELD1 VARCHAR(50), MSGFIELD2 VARCHAR(50),
constraint p_xordgift primary key (ORDERSID),
constraint f_xordgift foreign key (ORDERSID) references ORDERS(ORDERS_ID)
on delete cascade);
insert into XORDGIFT (ORDERSID) (select ORDERS_ID from ORDERS);
```

   Enter를 눌러 SQL문을 실행하십시오. XORDGIFT 테이블이 작성됩니다.

   주: 누군가 이전에 이 데이터베이스를 사용하여 예를 실행한 경우, XORDGIFT 테이블을 작성하기 전에 다음 명령을 실행해야 합니다.

```
drop table XORDGIFT;
```

6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

   Enter를 눌러 SQL문을 실행하십시오.

OrderGift 엔티티 bean 작성

일단 데이터베이스가 작성되면 새 엔티티 bean 작성을 시작할 준비가 된 것입니다. 다음 단계에서는 WebSphere Studio Application Developer를 사용하여 이 bean을 작성합니다.

다음을 수행하여 새 OrderGift bean을 작성합니다.

1. WebSphere Commerce Studio를 시작하십시오(**시작 > 프로그램 > IBM WebSphere Commerce Studio > WebSphere Commerce Development Environment**).

2. J2EE Perspective를 여십시오.
3. J2EE 계층 보기에서 **EJB 모듈**을 펼치십시오.
4. **WebSphereCommerceServerExtensionsData** 모듈에서 마우스 오른쪽 버튼을 누르고 **새로 만들기 > 엔터프라이즈 bean**을 선택하십시오.
   엔터프라이즈 bean 작성 마법사가 열립니다.
5. **EJB 프로젝트** 드롭 다운 목록에서 **WebSphereCommerceServerExtensionsData**를 선택하고 **다음**을 누르십시오.
6. 엔터프라이즈 bean 작성 창에서 다음을 수행하십시오.
   a. **CMP(container-managed persistence) 필드가 있는 엔티티 bean**을 선택하십시오.
   b. **Bean 이름** 필드에 `OrderGift`를 입력하십시오.
   c. **소스 폴더** 필드에서는 지정된 기본값(ejbModule)을 그대로 두십시오.
   d. **기본 패키지** 필드에 `com.ibm.commerce.extension.objects`를 입력하십시오.
   e. **다음**을 누르십시오.
7. CMP 속성 창에서 다음을 수행하십시오.
   a. **추가**를 눌러 테이블의 다음 열에 대한 새 필드를 추가하십시오.
      - ORDERSID
      - RECEIPTNAME
      - SENDERNAME
      - MSGFIELD1
      - MSGFIELD2

      CMP 속성 작성 창이 열립니다. 이 창에서 다음을 수행하십시오.
      1) 다음과 같이 ordersId 필드를 작성하십시오.

표 12.

매개변수 이름	매개변수 값
이름	`ordersId`

표 12. (계속)

매개변수 이름	매개변수 값
유형	java.lang.Long 주: *long* 데이터 유형이 아닌 *java.lang.Long* 데이터 유형을 사용해야 합니다.
키 필드	Select

적용을 누르십시오.

2) 다음과 같이 receiptName 필드를 작성하십시오.

표 13.

매개변수 이름	매개변수 값
이름	receiptName
유형	java.lang.String
getter 및 **setter** 메소드로 액세스	Select
getter 및 **setter** 메소드를 원격 인터페이스로 승격	Clear

적용을 누르십시오.

3) 다음과 같이 senderName 필드를 작성하십시오.

표 14.

매개변수 이름	매개변수 값
이름	senderName
유형	java.lang.String
getter 및 **setter** 메소드로 액세스	Select
getter 및 **setter** 메소드를 원격 인터페이스로 승격	Clear

적용을 누르십시오.

4) 다음과 같이 msgField1 필드를 작성하십시오.

표 15.

매개변수 이름	매개변수 값
이름	msgField1
유형	java.lang.String
getter 및 **setter** 메소드로 액세스	Select
getter 및 **setter** 메소드를 원격 인터페이스로 승격	Clear

적용을 누르십시오.

5) 다음과 같이 msgField2 필드를 작성하십시오.

표 16.

매개변수 이름	매개변수 값
이름	msgField2
유형	java.lang.String
getter 및 setter 메소드로 액세스	Select
getter 및 setter 메소드를 원격 인터페이스로 승격	Clear

적용을 누르십시오.

6) **닫기**를 눌러 창을 닫으십시오.

b. **키 클래스에 단일 키 속성 유형 사용** 선택란을 지우고 **다음**을 누르십시오.

8. EJB Java 클래스 정보 창에서 다음을 수행하십시오.
   a. Bean의 상위 클래스를 선택하려면 **찾아보기**를 누르십시오.
   유형 선택사항 창이 열립니다.
   b. **사용하는 클래스 선택: (임의)** 필드에 ECEntityBean을 입력하고 **확인**을 누르십시오. 그러면 상위 클래스로 com.ibm.commerce.base.objects.ECEntityBean이 선택됩니다.
   c. **완료**를 누르십시오.

다음을 수행하여 새 bean의 분리 레벨을 설정하십시오.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼치십시오.
2. **WebSphereCommerceServerExtensionsData** 프로젝트를 두 번 누르고 전개 설명자 편집기에서 프로젝트를 여십시오.
3. **액세스** 탭을 누르십시오.
4. 분리 레벨 텍스트 상자 옆에 있는 **추가**를 누르십시오.
   분리 레벨 추가 창이 열립니다.
5. DB2 **반복 가능 읽기**를 선택한 후 **다음**을 누르십시오.
   Oracle **읽기 확약**을 선택한 후 **다음**을 누르십시오.
6. **OrderGift** bean을 선택한 후 **다음**을 누르십시오.
7. **OrderGift**를 선택하여 해당되는 모든 메소드를 선택하고 **완료**를 누르십시오.
8. 작업을 저장하십시오(Ctrl+S).

다음을 수행하여 bean의 보안 동일성을 설정하십시오.

1. 전개 설명자 편집기에서 액세스 탭을 선택하십시오.
2. 보안 동일성(매소드 레벨) 텍스트 상자 옆에 있는 **추가**를 누르십시오. 보안 동일성 추가 창이 열립니다.
3. **EJB 서버의 동일성 사용**을 선택한 후 **다음**을 누르십시오.
4. **OrderGift** bean을 선택한 후 **다음**을 누르십시오.
5. **OrderGift**를 선택하여 해당되는 모든 메소드를 선택하고 **완료**를 누르십시오.
6. 작업을 저장하고(ctrl+s) 편집기를 열린 상태로 두십시오.

이번에는 다음을 수행하여 bean의 메소드에 대해 보안 역할을 설정하십시오.

1. 전개 설명자 편집기에서 어셈블리 설명자 탭을 선택하십시오.
2. 메소드 권한 절에서 **추가**를 누르십시오.
3. 보안 역할로 **WCSecurityRole**을 선택하고 **다음**을 누르십시오.
4. 발견된 bean 목록에서 **OrderGift**를 선택하고 **다음**을 누르십시오.
5. 메소드 요소 페이지에서 **모두에 적용**을 누른 후 **완료**를 누르십시오.
6. 작업을 저장하고(ctrl+s) 전개 설명자 편집기를 닫으십시오.

다음 단계는 WebSphere Studio Application Developer가 작성한 엔티티 컨텍스트에 관련된 필드 및 메소드를 제거하는 것입니다. 필드를 삭제해야 하는 이유는 ECEntityBean 기본 클래스가 해당 메소드의 구현을 제공하기 때문입니다. 작성된 엔티티 컨텍스트 필드와 메소드를 삭제하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **WebSphereCommerceServerExtensionsData** 프로젝트를 펼치십시오.
2. **OrderGift** EJB 모듈을 펼친 후 **OrderGiftBean**을 두 번 누르십시오.
3. 아웃라인 보기에서 다음을 수행하십시오.
   a. **myEntityCtx** 필드에서 마우스 오른쪽 버튼을 누르고 **삭제**를 선택하십시오.
   b. **getEntityContext()** 메소드에서 마우스 오른쪽 버튼을 누르고 **삭제**를 선택하십시오.

c. **setEntityContext(EntityContext)** 메소드에서 마우스 오른쪽 버튼을 누르고 **삭제**를 선택하십시오.

d. **unsetEntityContext()** 메소드에서 마우스 오른쪽 버튼을 누르고 **삭제**를 선택하십시오.

4. 작업을 저장하십시오(Ctrl+S).

새 OrderGift bean이 작성될 때 모든 매개변수가 명시적으로 설정되도록 작성된 ebjCreate 메소드를 다음과 같이 수정해야 합니다.

1. J2EE 계층 보기에서 **OrderGiftBean** 클래스를 두 번 눌러 열고 해당 소스 코드를 보십시오.
2. 아웃라인 보기에서 ejbCreate(Long) 메소드를 선택하십시오. 해당 소스 코드가 다음과 같이 표시됩니다.

```
public com.ibm.commerce.extension.objects.OrderGiftKey
   ejbCreate(java.lang.Long ordersId)
   throws javax.ejb.CreateException {
   _initLinks();
      this.ordersId = ordersId;
return null;
}
```

3. 코드가 다음과 같이 나타나도록 수정하십시오.

```
public com.ibm.commerce.extension.objects.OrderGiftKey
   ejbCreate(java.lang.Long ordersId)
   throws javax.ejb.CreateException {
   _initLinks();
      this.ordersId = ordersId;
      this.senderName = null;
      this.receiptName = null;
      this.msgField1 = null;
      this.msgField2 = null;
      return null;
}
```

4. 코드 변경사항을 저장하십시오.

다음을 수행하여 새 ejbCreate 메소드를 OrderGift bean에 추가하십시오.

1. J2EE 계층 보기에서 **OrderGiftBean** 클래스를 두 번 눌러 열고 해당 소스 코드를 보십시오.

2. 다음 코드를 클래스에 추가하여 새 ejbCreate(Long, String, String, String, String) 메소드를 작성하십시오.

```
public com.ibm.commerce.extension.objects.OrderGiftKey
   ejbCreate(java.lang.Long ordersId,
   java.lang.String receiptName,
   java.lang.String senderName,
   java.lang.String msgField1,
   java.lang.String msgField2)
   throws javax.ejb.CreateException {
   _initLinks();
      this.ordersId = ordersId;
      this.senderName = senderName;
      this.receiptName = receiptName;
      this.msgField1 = msgField1;
      this.msgField2 = msgField2;
return null;
 }
```

3. 코드 변경사항을 저장하십시오.
4. 이 새 ejbCreate 메소드를 홈 인터페이스에 추가해야 합니다. 그러면 메소드가 작성된 액세스 bean에서 사용할 수 있게 됩니다. 홈 인터페이스에 메소드를 추가하려면 다음을 수행하십시오.
   a. 아웃라인 보기에서 **ejbCreate(Long, String, String, String, String)** 메소드에서 마우스 오른쪽 버튼을 누르고 **엔터프라이즈 bean > 홈 인터페이스로 승격**을 선택하십시오.

이제 다음을 수행하여 ejbCreate(Long, String, String, String, String) 메소드와 동일한 매개변수를 갖도록 새 ejbPostCreate(Long, String, String, String, String) 메소드를 작성하십시오.

1. **OrderGiftBean** 클래스를 두 번 눌러 열고 해당 소스 코드를 보십시오.
2. 다음 코드를 클래스에 추가하여 새 ejbPostCreate(Long ordersId, String receiptName, String senderName, String msgField1, String msgField2) 메소드를 작성하십시오.

```
public void ejbPostCreate(
   java.lang.Long ordersId,
   java.lang.String receiptName,
   java.lang.String senderName,
```

```
java.lang.String msgField1,
java.lang.String msgField2)
throws javax.ejb.CreateException {
}
```

3. 코드 변경사항을 저장하십시오.

다음은 XORDGIFT 테이블의 정의를 작성해야 합니다. 251 페이지의 제 10 장 『학습: 새 비즈니스 로직 작성』을 완료하지 않은 경우, 다음을 수행하여 XORDGIFT 테이블 정의를 작성하십시오.

1. 데이터 perspective를 열고 데이터 정의 보기로 전환하십시오.
2. **WebSphereCommerceServerExtensionsData** > **ejbModule** > **META-INF** 디렉토리를 탐색하십시오.
3. **META-INF**에서 마우스 오른쪽 버튼을 누르고 **새 데이터베이스 정의**를 선택하십시오.
   새 데이터베이스 정의 마법사가 열립니다.
4. **데이터베이스 이름** 필드에 개발 데이터베이스의 이름을 입력하십시오. 예를 들어,Demo_Dev를 입력하십시오.
5. 데이터베이스 공급업체 유형 드롭 다운 목록에서 전개 환경에 적절한 데이터베이스 유형을 선택하십시오.

   DB2 **DB2 Universal Database V8.1**

   Oracle **Oracle 9i**
6. **완료**를 누르십시오. 새 데이터베이스 정의가 작성됩니다.
7. **WebSphereCommerceServerExtensionsData** > **ejbModule** > **META-INF** > **Schema** > *developmentDB* 디렉토리를 탐색하십시오.
8. *developmentDB*를 마우스 오른쪽 버튼으로 누르고 **새로 만들기** > **새 스키마 정의**를 선택하십시오.
   새 스키마 정의 마법사가 열립니다.
9. **스키마 이름** 필드에 NULLID를 입력하고 **완료**를 누르십시오.
10. **WebSphereCommerceServerExtensionsData** > **ejbModule** > **META-INF** > **Schema** > *developmentDB* > **NULLID** > **Tables**를 탐색하십시오.

11. **Tables**에서 마우스 오른쪽 버튼을 누르고 **새 테이블 정의**를 선택하십시오. 새 테이블 정의 마법사가 열립니다.
12. **테이블 이름** 필드에 XORDGIFT을 입력한 후 **다음**을 누르십시오.
13. 다음과 같이 키 열을 테이블 정의에 추가해야 합니다.
    a. **추가하기**를 누르십시오.
    b. **열 이름** 필드에 ORDERSID를 입력하십시오.
    c. **열 유형** 드롭 다운 목록에서 다음을 선택하십시오.
       DB2 BIGINT
       Oracle NUMBER
    d. **키 열**을 선택하십시오.
    e. Oracle **수치 정밀도** 필드에 38을 입력하십시오.
    f. Oracle **수치 스케일**의 값을 0으로 두십시오.
14. 다음과 같이 테이블에 열을 더 추가하십시오.
    a. **추가하기**를 누르고 다음 특성으로 열을 작성하십시오.

표 17.

특성	값
열 이름	RECEIPTNAME
열 유형	DB2 VARCHAR Oracle VARCHAR2
널 가능	Select
문자열 길이	50
비트 데이터의 경우	Clear

    b. **추가하기**를 누르고 다음 특성으로 열을 작성하십시오.

표 18.

특성	값
열 이름	SENDERNAME
열 유형	DB2 VARCHAR Oracle VARCHAR2

표 18. (계속)

특성	값
널 가능	Select
문자열 길이	50
비트 데이터의 경우	Clear

c. **추가하기**를 누르고 다음 특성으로 열을 작성하십시오.

표 19.

특성	값
열 이름	MSGFIELD1
열 유형	DB2 VARCHAR Oracle VARCHAR2
널 가능	Select
문자열 길이	50
비트 데이터의 경우	Clear

d. **추가하기**를 누르고 다음 특성으로 열을 작성하십시오.

표 20.

특성	값
열 이름	MSGFIELD2
열 유형	DB2 VARCHAR Oracle VARCHAR2
널 가능	Select
문자열 길이	50
비트 데이터의 경우	Clear

e. **완료**를 누르십시오.

f. Oracle 다음과 같이 텍스트 편집기를 사용하여 테이블 정의를 편집해야 합니다.

1) J2EE 네비게이터 보기로 전환하십시오.

2) **WebSphereCommerceServerExtensionsData** 프로젝트를 펼치십시오.

3) **ejbModule** > **META-INF** > **Schema**를 펼치십시오.

4) **WebSphereCommerceServerExtensionsData_NULL_XORDGIFT xmi** 파일을 누르고 **열기** > **텍스트 편집기**를 선택하십시오.

5) 모든 SQLNumeric_6 어커런스를 SQLNumeric_3으로 바꾸십시오.

6) 변경사항을 저장하고 텍스트 편집기를 닫으십시오.

251 페이지의 제 10 장 『학습: 새 비즈니스 로직 작성』을 완료한 경우, 다음을 수행하여 XORDGIFT 테이블 정의를 작성하십시오.

1. 데이터 perspective를 열고 데이터 정의 보기로 전환하십시오.
2. **WebSphereCommerceServerExtensionsData** > **ejbModule** > **META-INF** > **Schema** > *developmentDB* > **NULLID** > **Tables** 디렉토리를 탐색하십시오.
3. **Tables** 디렉토리를 마우스 오른쪽 버튼으로 누르고 **새로 만들기** > **새 테이블 정의**를 선택하십시오.
   새 테이블 정의 마법사가 열립니다.
4. **테이블 이름** 필드에 XORDGIFT을 입력한 후 **다음**을 누르십시오.
5. 다음과 같이 키 열을 테이블 정의에 추가해야 합니다.
   a. **추가하기**를 누르십시오.
   b. **열 이름** 필드에 ORDERSID를 입력하십시오.
   c. **열 유형** 드롭 다운 목록에서 다음을 선택하십시오.
      DB2 BIGINT
      Oracle NUMBER
   d. **키 열**을 선택하십시오.
   e. Oracle **수치 정밀도** 필드에 38을 입력하십시오.
   f. Oracle **수치 스케일**의 값을 0으로 두십시오.
6. 다음과 같이 테이블에 열을 더 추가하십시오.

a. **추가하기**를 누르고 다음 특성으로 열을 작성하십시오.

표 21.

특성	값
열 이름	RECEIPTNAME
열 유형	DB2 VARCHAR Oracle VARCHAR2
널 가능	Select
문자열 길이	50
비트 데이터의 경우	Clear

b. **추가하기**를 누르고 다음 특성으로 열을 작성하십시오.

표 22.

특성	값
열 이름	SENDERNAME
열 유형	DB2 VARCHAR Oracle VARCHAR2
널 가능	Select
문자열 길이	50
비트 데이터의 경우	Clear

c. **추가하기**를 누르고 다음 특성으로 열을 작성하십시오.

표 23.

특성	값
열 이름	MSGFIELD1
열 유형	DB2 VARCHAR Oracle VARCHAR2
널 가능	Select
문자열 길이	50
비트 데이터의 경우	Clear

d. **추가하기**를 누르고 다음 특성으로 열을 작성하십시오.

표 24.

특성	값
열 이름	MSGFIELD2
열 유형	DB2 VARCHAR Oracle VARCHAR2
널 가능	Select
문자열 길이	50
비트 데이터의 경우	Clear

e. **완료**를 누르십시오.

7. Oracle 다음과 같이 텍스트 편집기를 사용하여 테이블 정의를 편집해야 합니다.
   a. J2EE 네비게이터 보기로 전환하십시오.
   b. **WebSphereCommerceServerExtensionsData** 프로젝트를 펼치십시오.
   c. **ejbModule** > **META-INF** > **Schema**를 펼치십시오.
   d. **WebSphereCommerceServerExtensionsData_NULL_XORDGIFT.xmi** 파일을 마우스 오른쪽 버튼으로 누르고 **열기** > **텍스트 편집기**를 선택하십시오.
   e. 모든 SQLNumeric_6 어커런스를 SQLNumeric_3으로 바꾸십시오.
   f. 변경사항을 저장하고 텍스트 편집기를 닫으십시오.

다음 단계는 XORDGIFT 테이블을 OrderGiftBean 엔티티 bean에 맵핑하는 것입니다. 개발 데이터베이스와 XORDGIFT 테이블이 이미 존재하기 때문에 중간 맞춤 맵핑이 사용됩니다.

251 페이지의 제 10 장 『학습: 새 비즈니스 로직 작성』을 완료하지 않은 경우, 다음을 수행하여 맵핑을 작성하십시오.

1. J2EE 계층 보기에서 **WebSphereCommerceServerExtensionsData** 프로젝트에서 마우스 오른쪽 버튼을 누르고 **작성** > **EJB에서 RDB로 맵핑**을 선택하십시오.
   새 EJB/RDB 맵핑 작성 마법사가 열립니다.

2. **중간 맞춤**을 선택하고 **다음을** 누르십시오.
3. **이름 및 유형 기준 일치**를 선택하고 **완료**를 누르십시오.
   Map.mapxmi 편집기가 열립니다.
4. 엔터프라이즈 bean 분할창에서 **OrderGift** bean을 펼치십시오. 테이블 분할창에서 **XORDGIFT**를 펼치십시오.
5. 다음을 수행하여 Bonus bean의 필드를 XORDGIFT 테이블 열에 맵핑하십시오.
   a. **OrderGift** bean을 마우스 오른쪽 버튼으로 누르고 **Match By Name**을 선택하십시오.
6. Ctrl+S를 눌러서 Map.mapxmi 파일을 저장하십시오. 파일을 닫으십시오.

251 페이지의 제 10 장『학습: 새 비즈니스 로직 작성』을 완료한 경우, 다음을 수행하여 맵핑을 작성하십시오.

1. J2EE 계층 보기에서 **WebSphereCommerceServerExtensionsData** 프로젝트에서 마우스 오른쪽 버튼을 누르고

   작성 **>** **EJB**에서 **RDB**로 맵핑을 선택하십시오.
   Map.mapxmi
   편집기가 열립니다.
2. 엔터프라이즈 bean 분할창에서 **OrderGift** bean을 펼치십시오. 테이블 분할창에서 **XORDGIFT**를 펼치십시오.
3. 다음을 수행하여 Bonus bean의 필드를 XORDGIFT 테이블 열에 맵핑하십시오.
   a. **OrderGift** bean을 마우스 오른쪽 버튼으로 누르고 **Match By Name**을 선택하십시오.
4. Ctrl+S를 눌러서 Map.mapxmi 파일을 저장하십시오. 파일을 닫으십시오.

일단 Bonusbean 엔티티가 작성되고 스키마가 올바르게 맵핑되면 엔티티 bean의 액세스 bean을 작성할 수 있습니다. 이 액세스 bean은 응용프로그램이 OrderGift 엔티티 bean에 들어 있는 정보에 더 간단하게 액세스하도록 합니다. WebSphere Studio Application Developer의 도구는 이미 작성한 엔티티에 기초하여 액세스 bean을 작성하는 데 사용됩니다(특히 액세스 bean은 원격 인터페이스로 승격된 메소드만 사용합니다). OrderGift 엔티티 bean의 액세스 bean을 작성하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼친 후 **WebSphereCommerceServerExtensionsData**에서 마우스 오른쪽 버튼을 누르고 **새로 만들기 > 액세스 bean**을 선택하십시오.
   액세스 bean 추가 창이 열립니다.
2. **복사 헬퍼**를 선택한 후 **다음**을 누르십시오.
3. **OrderGift** bean을 선택하고 **다음**을 누르십시오.
4. **생성자 메소드** 드롭 다운 목록에서 **findByPrimaryKey(com.ibm.commerce.extension.objects.OrderGiftKey)**를 선택하십시오.
5. 속성 헬퍼 섹션에서 모든 속성을 선택하십시오.
6. **완료**를 누르십시오.

J2EE 네비게이터 보기로 전환하여 새로 작성된 코드를 보고, **WebSphereCommerceServerExtensionsData** 프로젝트, **ejbModule** 서브폴더 및 **com.ibm.commerce.extension.objects**를 차례로 펼칠 수 있습니다. OrderGiftAccessBean이라고 하는 새 클래스와 OrderGiftAccessBeanData라고 하는 새 인터페이스가 작성되어 패키지 내에 표시됩니다.

다음 단계는 전개 코드를 작성하는 것입니다.

코드 작성 유틸리티는 bean을 분석하여 Sun Microsystems의 EJB 스펙에 맞는지 확인하고 EJB 서버 고유 규칙을 따랐는지 확인합니다. 또한 선택된 엔터프라이즈 bean마다 코드 작성 도구는 CMP bean용 JDBC 지속 및 finder 클래스뿐 아니라 홈 및 EJBObject(원격) 구현, 홈 및 원격 인터페이스용 구현 클래스를 작성합니다. 또한 홈 및 원격 인터페이스용 스텁(stub)뿐 아니라, IIOP를 통한 RMI 액세스에 필요한 Java ORB, 스텁, 타이 클래스를 작성합니다.

전개 코드를 작성하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼친 후 **WebSphereCommerceServerExtensionsData**에서 마우스 오른쪽 버튼을 누르고 **새로 만들기 > 전개 및 RMIC 코드**를 선택하십시오.
   전개 및 RMIC 코드 창이 열립니다.
2. **OrderGift**를 선택하고 **완료**를 누르십시오.

J2EE 네비게이터 보기로 전환하여 새로 작성된 코드를 볼 수 있습니다. 다음을 볼 수 있습니다.

표 25.

코드 유형	클래스 이름
컨테이너 구현 작성 코드	EJSCMPOrderGiftHomeBean.java
	EJSRemoteCMPOrderGift.java
	EJSRemoteCMPOrderGiftHome.java
	EJSFinderOrderGiftBean.java
JDBC 액세스 코드	EJSJDBCPersisterCMPOrderGiftBean.java
RMI 타이 및 스텁 코드	_EJSRemoteCMPOrderGift_Tie.java
	_OrderGift_Stub.java
	_EJSRemoteCMPOrderGiftHome_Tie.java
	_OrderGiftHome_Stub.java

쇼핑 플로우에 OrderGift 엔티티 bean 통합

이 절에서는 다음을 수행하여 견본 상점의 정기 쇼핑 플로우에 OrderGift 엔티티 bean을 통합합니다.

1. 새 OrderGiftDataBean 데이터 bean 작성
2. 새 MyExtOrderProcessCmdImpl 태스크 명령 작성
3. OrderSubmitForm.jsp 표시 페이지 수정

각각의 앞 단계에 대해 후속 절에서 더 자세히 설명합니다.

OrderGiftDataBean 작성

OrderGift 엔티티 bean의 속성이 JSP 표시 페이지에 표시될 수 있도록 OrderGiftDataBean을 작성해야 합니다. 학습의 다른 부분처럼 기본 코드가 제공되므로 코드의 다양한 섹션에 대해 주석을 제거해야 합니다.

OrderGiftDataBean을 작성하려면 다음을 수행하십시오.

1. 첫 번째 단계는 다음과 같이 새 데이터 bean의 기본 코드를 반입하는 것입니다.
   a. 252 페이지의 『견본 코드 찾기』의 단계를 완료했는지 확인하십시오.

b. J2EE Perspective로 전환한 후 J2EE 네비게이터 보기를 선택하십시오.

c. **WebSphereCommerceServerExtensionsLogic** 프로젝트를 펼치십시오.

d. **src** 폴더에서 마우스 오른쪽 버튼을 누르고 **반입**을 선택하십시오. 가져오기 마법사가 열립니다.

e. **반입 소스 선택** 목록에서 **Zip 파일**을 선택하고 **다음**을 누르십시오.

f. **찾아보기(Zip 파일** 필드 옆에 있는)를 누르고 견본 코드를 탐색하십시오. 이 파일은 다음 위치에 지정됩니다.
yourDirectory\WC_SAMPLE_55.zip
여기서, *yourDirectory*는 패키지를 다운로드한 디렉토리입니다.

g. **모두 선택 취소**를 누른 후 디렉토리를 펼쳐 반입할 다음 파일을 선택하십시오.

- com\ibm\commerce\sample\databeans\OrderGiftDataBean.java

h. **폴더** 필드에서는 이미 `WebSphereCommerceServerExtensionsLogic/src` 폴더가 지정되어 있습니다. 이 값을 변경하지 마십시오.

i. **완료**를 누르십시오.

bean에 대한 코드를 반입한 후에 코드를 검사하십시오.

MyExtOrderProcessCmdImpl 클래스 작성

이 절에서는 선물 주문에 관련된 정보가 XORDGIFT 데이터베이스 테이블에서 갱신되도록 OrderProcess 비즈니스 프로세스 끝에서 새 로직을 추가합니다. 이 비즈니스 프로세스의 확장 지점으로 ExtOrderProcessCmdImpl 명령이 제공됩니다. 프로그래밍 모델에 따라 로직을 확장하려면 태스크 명령의 새 구현 클래스를 작성하고 새 로직을 이 클래스에 포함시킵니다. 그런 다음 새 구현 클래스가 ExtOrderProcessCmd 인터페이스와 연관되도록 명령 레지스트리를 갱신해야 합니다.

MyExtOrderProcessCmdImpl 클래스를 작성하려면 다음을 수행하십시오.

1. 첫 번째 단계는 다음과 같이 새 명령의 기본 코드를 반입하는 것입니다.

   a. **WebSphereCommerceServerExtensionsLogic** 프로젝트를 펼치십시오.

   b. **src** 폴더에서 마우스 오른쪽 버튼을 누르고 **반입**을 선택하십시오. 가져오기 마법사가 열립니다.

c. 반입 소스로 **Zip 파일**을 선택하고 **다음**을 누르십시오.

d. **반입 소스 선택** 목록에서 **Zip 파일**을 선택하고 **다음**을 누르십시오.

e. **찾아보기(Zip 파일** 필드 옆에 있는)를 누르고 견본 코드를 탐색하십시오. 이 파일은 다음 위치에 지정됩니다.
yourDirectory\WC_SAMPLE_55.zip
여기서, *yourDirectory*는 패키지를 다운로드한 디렉토리입니다.

f. **모두 선택 취소**를 누른 후 디렉토리를 펼쳐 반입할 다음 파일을 선택하십시오.

- com\ibm\commerce\sample\commands\ MyExtOrderProcessCmdImpl.java

g. **폴더** 필드에서는 이미 `WebSphereCommerceServerExtensionsLogic/src` 폴더가 지정되어 있습니다. 이 값을 변경하지 마십시오.

h. **완료**를 누르십시오.

이런 새 명령의 코드가 반입되면 소스를 검사하여 명령이 수행하는 작업을 알 수 있습니다. 명령이 상위 클래스의 performExecute() 메소드를 호출하여 명령이 실행되는 프로세스를 확인한다는 것을 유의하십시오. 이것에는 선물 주문 정보를 새 데이터 bean으로 설정하는 로직이 포함됩니다.

이 단계에서는 새 MyExtOrderProcessCmdImpl 구현 클래스가 원래 ExtOrderProcessCmdImpl 구현 클래스 대신 사용되도록 명령 레지스트리를 수정합니다. 명령 레지스트리에서 수정해야 하는 유일한 테이블은 CMDREG 테이블입니다. 이 경우, 새 구현 클래스는 모든 상점에 사용됩니다.

명령 레지스트리를 수정하려면 다음을 수행하십시오.

DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 MyExtOrderProcessCmdImpl을 등록하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령행 도구 > 명령 센터**)를 여십시오.
2. **도구** 메뉴에서 **도구 설정**을 선택하십시오.
3. **명령문 종료 문자 사용** 선택란을 선택한 후 지정된 문자가 세미콜론(**;**)인지 확인하십시오.

4. 도구 설정을 닫으십시오.
5. 스크립트 탭을 선택하고 스크립트 창에 다음 정보를 입력하여 URLREG 테이블에 필수 항목을 작성하십시오.

```
connect to developmentDB user dbuser using dbpassword;
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FashionFlow_storeent_ID,
   'com.ibm.commerce.order.commands.ExtOrderProcessCmd',
   'This is a new task command for tutorial two.',
   'com.ibm.commerce.sample.commands.MyExtOrderProcessCmdImpl',
   'local');
```

여기서,

- *developmentDB*는 개발 데이터베이스의 이름입니다.
- *dbuser*는 데이터베이스 사용자입니다.
- *dbpassword*는 데이터베이스 사용자의 암호입니다.
- *FashionFlow_storeent_ID*는 견본 상점의 상점 식별자입니다.

실행 아이콘을 누르십시오.

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 MyOrderItemAddCmdImpl을 등록하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작 > 프로그램 > Oracle > Application Development > SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 SQL문을 입력하십시오.

```
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FashionFlow_storeent_ID,
   'com.ibm.commerce.order.commands.ExtOrderProcessCmd',
   'This is a new task command for tutorial two.',
   'com.ibm.commerce.sample.commands.MyExtOrderProcessCmdImpl',
   'local');
```

SQL문을 실행하려면 Enter를 누르십시오.

6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

   ```
   commit;
   ```

   Enter를 눌러 SQL문을 실행하십시오.

변경사항 컴파일

중요: 상점 웹 프로젝트를 다시 빌드하지 않도록 하십시오.

이 절에서는 다음과 같이 코드에 작성한 변경사항을 컴파일합니다.

1. J2EE 네비게이터 보기에서 다음 프로젝트를 선택하십시오.
   - WebSphereCommerceServerExtensionsData
   - WebSphereCommerceServerExtensionsLogic
2. 앞에 표시된 프로젝트를 강조표시한 상태에서 마우스 오른쪽 버튼을 누른 후 **프로젝트 빌드**를 선택하십시오.

선물 메시지용 표시 페이지 수정

이 단계에서는 고객이 요약 페이지에서 정보를 볼 수 있을 뿐만 아니라 선문 주문에 관한 메시지 정보를 입력할 수 있도록 OrderSubmitForm 및 OrderDetailDisplay 템플리트를 수정합니다. 이런 페이지 수정 방법은 페이지의 새 정보를 지정하는 추가 JSP 템플리트를 포함하는 것입니다. 이런 새 페이지(OrderSubmitFormInclude.jsp 및 OrderDetailDisplayInclude.jsp)는 JSTL을 사용하여 새 정보를 표시합니다.

표시 페이지를 수정하려면 다음을 수행하십시오.

1. 웹 perspective로 전환하고 J2EE Navigator 보기로 이동하십시오.
2. 251 페이지의 제 10 장『학습: 새 비즈니스 로직 작성』을 완료하지 않았다면 다음과 같이 웹 프로젝트 저장 특성을 수정해야 합니다.
   a. **Stores** 웹 프로젝트에서 마우스 오른쪽 버튼을 누른 후 **특성**을 선택하십시오.

b. 왼쪽 분할창에서 **웹**을 선택한 후 사용 가능한 웹 프로젝트 특징 목록에서 **JSP 표준 태그 라이브러리 포함**을 선택하십시오. **적용**을 누르십시오. 갱신이 완료되면 **확인**을 눌러 특성 편집기를 닫으십시오.

3. 다음 디렉토리로 펼치십시오.
   Stores\Web Content*FashionFlow_name*
4. 다음을 수행하여 OrderSubmitForm.jsp 파일의 백업 사본을 작성하십시오.
   a. **ShoppingArea>CheckoutSection>StandardCheckoutSubsection** 디렉토리를 펼치십시오.
   b. **OrderSubmitForm.jsp** 파일에서 마우스 오른쪽 버튼을 누르고 **이름 바꾸기**를 선택하십시오.
   c. 이름 바꾸기 창에서 `OrderSubmitForm_bak.jsp`를 입력하고 **확인**을 누르십시오.
   d. 이 파일로 링크를 갱신하려는지 여부를 묻는 프롬프트에서 **아니오**를 누르십시오.
5. 다음을 수행하여 OrderDetailDisplay.jsp 파일의 백업 사본을 작성하십시오.
   a. **UserArea>ServiceSection>TrackOrderStatusSubsection** 디렉토리를 펼치십시오.
   b. **OrderDetailDisplay.jsp** 파일에서 마우스 오른쪽 버튼을 누르고 **이름 바꾸기**를 선택하십시오.
   c. 이름 바꾸기 창에서 `OrderDetailDisplay_bak.jsp`를 입력하고 **확인**을 누르십시오.
   d. 이 파일로 링크를 갱신하려는지 여부를 묻는 프롬프트에서 **아니오**를 누르십시오.
6. *FashionFlow_name* 디렉토리에서 마우스 오른쪽 버튼을 누르고 **반입**을 선택하십시오.
   가져오기 마법사가 열립니다.
7. **반입 소스 선택** 목록에서 **Zip 파일**을 선택하고 **다음**을 누르십시오.
8. **찾아보기(Zip 파일** 필드 옆에 있는)를 누르고 견본 코드를 탐색하십시오. 이 파일은 다음 위치에 지정됩니다.

yourDirectory\WC_SAMPLE_55.zip
여기서, *yourDirectory*는 패키지를 다운로드한 디렉토리입니다.

9. **모두 선택 취소**를 누른 후 디렉토리를 펼쳐 반입할 다음 파일을 선택하십시오.
   - ShoppingArea\CheckoutSection\StandardCheckoutSubsection\OrderSubmitForm.jsp
   - ShoppingArea\CheckoutSection\StandardCheckoutSubsection\OrderSubmitFormInclude.jsp
   - UserArea\ServiceSection\TrackOrderStatusSubsection\OrderDetailDisplay.jsp
   - UserArea\ServiceSection\TrackOrderStatusSubsection\OrderDetailDisplayInclude.jsp
10. **폴더** 필드에는 이미 Stores/Web Content/*FashionFlow_name* 폴더가 지정되어 있습니다. 이 값을 변경하지 마십시오.
11. **완료**를 누르십시오.

OrderSubmitForm.jsp 파일을 검사할 때 다음이 추가된 것을 알게 될 것입니다.

```
<!-- tutorial start-->
   <jsp:include page="OrderSubmitFormInclude.jsp" flush="true" />
<!-- tutorial done -->
```

이것은 새 OrderSubmitFormInclude.jsp 파일이 기존 JSP 템플리트로 상호작용하는 방법입니다. 템플리트의 JSTL 사용 방법의 예제에 대한 OrderSubmitFormInclude.jsp를 검사하십시오. 이와 유사하게, OrderDetailDisplay.jsp 및 OrderDetailDisplayInclude.jsp 파일을 검사하십시오.

또한 수정된 JSP 템플리트에서 사용된 문자열 값이 들어 있는 특성 파일을 반입해야 합니다. 이 파일의 이름은 ordergift.properties입니다. 파일을 반입하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 다음 디렉토리를 펼치십시오.
   Stores > Web Content > WEB-INF > classes > *FashionFlow_name*

2. *FashionFlow_name* 디렉토리에서 마우스 오른쪽 버튼을 누르고 **반입**을 선택하십시오.
   가져오기 마법사가 열립니다.
3. **반입 소스 선택** 목록에서 **Zip 파일**을 선택하고 **다음**을 누르십시오.
4. **찾아보기(Zip 파일** 필드 옆에 있는)를 누르고 견본 코드를 탐색하십시오. 이 파일은 다음 위치에 지정됩니다.
   yourDirectory\WC_SAMPLE_55.zip
   여기서, *yourDirectory*는 패키지를 다운로드한 디렉토리입니다.
5. **모두 선택 취소**를 누른 후 디렉토리를 펼쳐 반입할 다음 파일을 선택하십시오.
   - ordergift.properties
6. **폴더** 필드에는 이미 `Stores/Web Content/WEB-INF/classes/`*`FashionFlow_name`* 폴더가 지정되어 있습니다. 이 값을 변경하지 마십시오.
7. **완료**를 누르십시오.

새 선물 메시지 기능 테스트

이 절에서는 다음과 같이 새 선물 메시지 기능을 테스트합니다.

1. 서버 perspective로 전환하십시오(**창 > Perspective 열기 > 서버**).
2. Payment Server를 시작하십시오.
3. **WebSphereCommerceServer** 서버에서 마우스 오른쪽 버튼을 누르고 **시작**(또는 **다시 시작**)을 선택하십시오.
4. 웹 브라우저를 열고 다음 URL을 입력하십시오.

   ```
   http://localhost/webapp/wcs/stores/servlet/FashionFlow/index.jsp
   ```

5. 새 사용자로 로그온하십시오. 예를 들어, **등록** 다음에 새 사용자 작성을 누르거나 기존 사용자로 로그온하십시오.
6. 새로 등록한 사용자로서 상점을 찾아보고 항목을 장바구니에 추가한 후 구매를 완료하십시오. 다음 화면에 표시된 것처럼 주문에 선물 메시지를 추가할 수 있습니다.

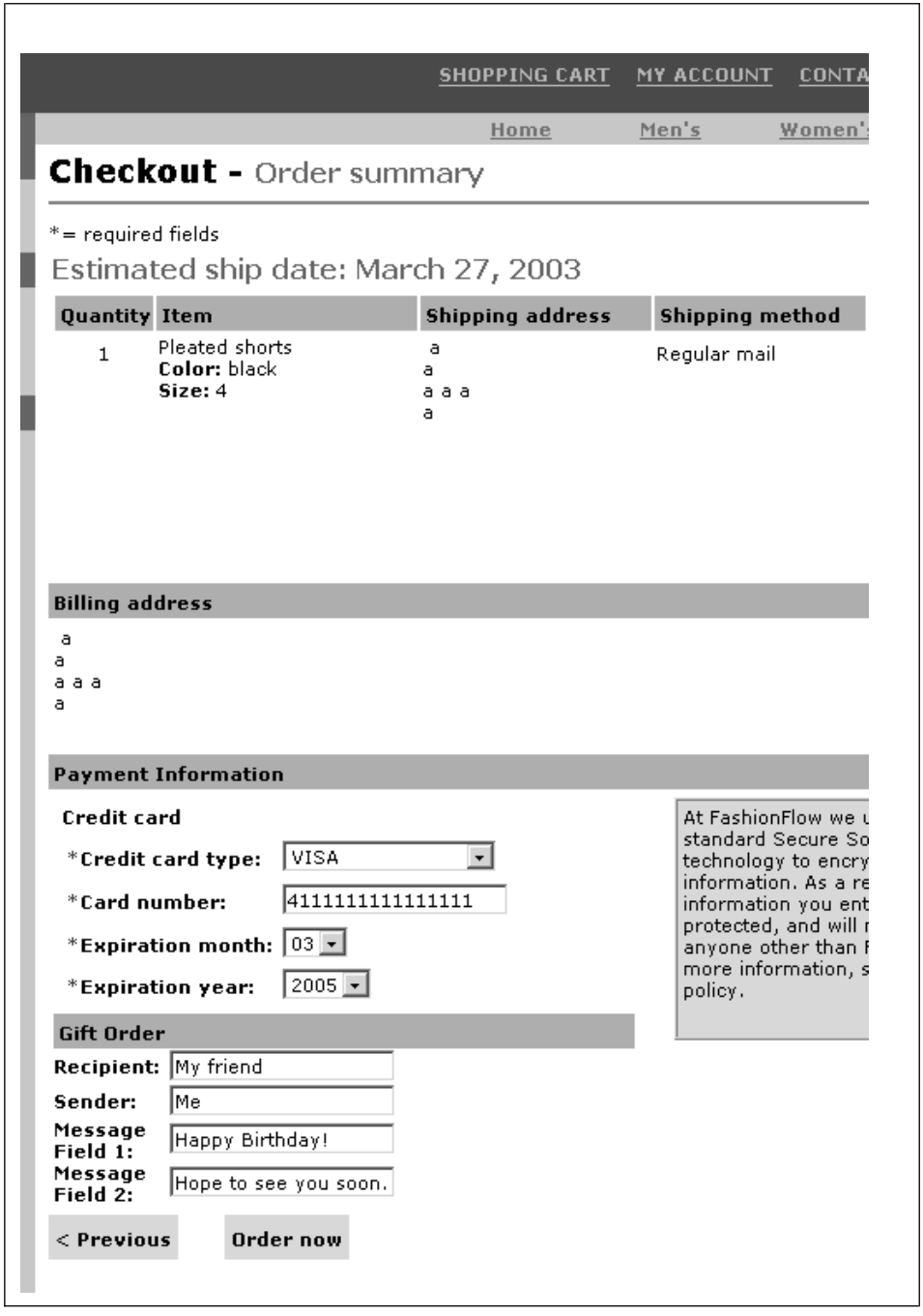


그림 49.

이 주문과 연관되는 주문 번호를 기록하십시오.

7. **회원 정보**를 누르십시오.
8. **주문 보기**를 누르십시오.
9. 6단계에서 작성한 주문을 선택하십시오. 다음과 유사한 화면이 표시됩니다.

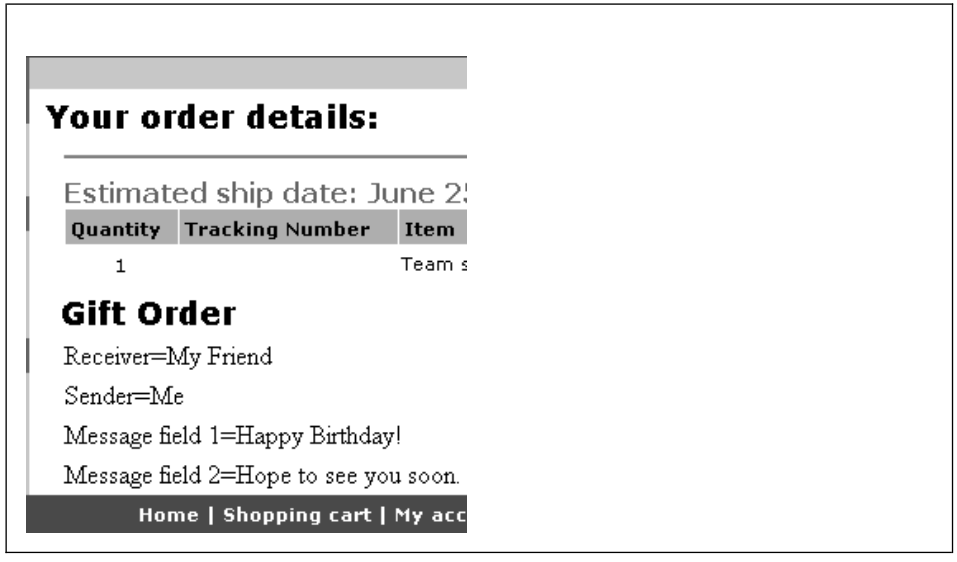


그림 50.

선물 메시지 기능 전개

이 절에서는 원격 WebSphere Commerce Server에서 실행 중인 상점으로 새 비즈니스 로직을 전개하는 방법에 대해 설명합니다. 전개 단계를 시작하려면 먼저 원격 WebSphere Commerce Server에 상점을 작성해 놓아야 합니다(InFashion 견본 상점을 기초로). 해당 상점 내에서 구매를 완료할 수 있어야 합니다.

개발 처리에는 대상 WebSphere Commerce Server에서 수행되는 단계 뿐만 아니라 개발 시스템에서 수행되는 단계도 포함됩니다.

대상 WebSphere Commerce Server에 전개해야 하는 여러 가지 다른 유형의 자원이 있습니다. 자원 유형은 다음과 같습니다.

- 태스크 명령 및 데이터 bean 로직

- 엔터프라이즈 bean 로직
- 갱신된 JSP 템플리트
- 스키마 갱신(새 테이블)과 명령 레지스트리 갱신을 포함한 데이터베이스 갱신사항

이 절에서는 이 모든 자원을 대상 WebSphere Commerce Server에 *증분적으로* 전개하는 방법에 대해 설명합니다. 이는 EAR 파일 전체를 전개하는 것과는 대조적으로 점층적 전개로 수행됩니다.

명령 및 데이터 bean JAR 파일 작성

이 절에서는 컨트롤러 명령, 태스크 명령 및 데이터 bean 로직을 포함하는 JAR 파일을 작성하는 방법에 대해 설명합니다.

JAR 파일을 작성하려면 개발 시스템에서 다음 단계를 수행하십시오.

1. 로컬 파일 시스템에 *drive*:\ExportTemp3라고 하는 디렉토리를 작성하십시오.
2. WebSphere Studio Application Developer를 열고 J2EE 네비게이터 보기로 전환하십시오.
3. **WebSphereCommerceServerExtensionsLogic** 프로젝트에서 마우스 오른쪽 버튼을 누른 후 **반출**을 선택하십시오.
   반출 마법사가 열립니다.
4. 반출 마법사에서 다음을 수행하십시오.
   a. **JAR 파일**을 선택하고 **다음**을 누르십시오.
   b. **반출할 자원 선택?** 아래에 있는 왼쪽 분할창이 프로젝트 이름으로 채워집니다. 이 값은 있는 그대로 두십시오.
   c. 오른쪽 분할창에서 다음 자원만이 선택되었음을 확인하십시오.
      - .classpath
      - .project
      - .serverPreference
   d. **작성된 클래스 파일 및 자원 반출**이 선택되어 있는지 확인하십시오.
   e. Java 소스 파일 및 자원 **반출**을 선택하지 *마십시오*.

f. **반출 대상 선택** 필드에서 사용할 완전한 JAR 파일 이름을 입력하십시오. 이 경우에는 *drive*:\ExportTemp3\WebSphereCommerceServerExtensionsLogic.jar을 입력하십시오. JAR 파일 이름은 WebSphereCommerceServerExtensionsLogic.jar이어야 합니다.

g. **완료**를 누르십시오.

EJB JAR 파일 작성

EJB JAR 파일을 작성하려면 다음을 수행하십시오.

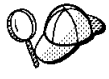
1. WebSphere Studio Application Developer를 열고 J2EE 네비게이터 보기로 전환하십시오.
2. **WebSphereCommerceServerExtensionsData** 프로젝트를 펼치십시오.
3. **EJB 전개 설명자**를 두 번 누르십시오.
4. 개요 탭을 선택한 상태에서 분할창의 맨 아래로 화면이동하여 **WebSphere 바인딩** 섹션을 찾으십시오.
5. **DataSource JNDI 이름** 필드에 대상 WebSphere Commerce Server의 데이터소스 JNDI 이름을 입력하십시오. 다음은 예제 값입니다.

   DB2 `jdbc/WebSphere Commerce DB2 DataSource demo`

   여기서, 대상 WebSphere Commerce Server는 DB2 데이터베이스를 사용 중이고 WebSphere Commerce 인스턴스 이름은 “demo”입니다.

   Oracle `jdbc/WebSphere Commerce Oracle DataSource demo`

   여기서, 대상 WebSphere Commerce Server는 Oracle 데이터베이스를 사용 중이고 WebSphere Commerce 인스턴스 이름은 “demo”입니다.



DataSource JNDI 이름에 대한 값은 “jdbc/”를 대상 WebSphere Commerce Server의 데이터 소스 이름에 추가하여 작성됩니다. 대상 WebSphere Commerce Server에서 *instanceName*.xml 파일을 열고 파일에 있는 DatasourceName=을 검색하여 데이터 소스 이름을 확인할 수 있습니다.

6. 전개 설명자 변경사항을 저장하십시오(ctrl+s).

7. J2EE 네비게이터 보기에서 **WebSphereCommerceServerExtensionsData** 프로젝트에서 마우스 오른쪽 버튼을 누른 후 **반출**을 선택하십시오.
   반출 마법사가 열립니다.
8. 반출 마법사에서 다음을 수행하십시오.
   a. **EJB JAR 파일**을 선택하고 **다음**을 누르십시오.
   b. **반출할 자원은?** 값은 EJB 프로젝트의 이름으로 채워집니다. 이 값은 있는 그대로 두십시오.
   c. **자원 반출 위치** 필드에서 사용할 완전한 JAR 파일 이름을 입력하십시오. 이 경우에는
   drive:\ExportTemp3\WebSphereCommerceServerExtensionsData.jar을 입력하십시오.
   d. **완료**를 누르십시오.
9. JAR 파일을 작성하였으면 5단계에서 수행한 로컬 전개 설명자 변경사항을 실행 취소하여 로컬 테스트 서버에 필요한 설정을 복원하십시오.

상점 자원 반출

WebSphere Studio Application Developer에서 OrderSubmitForm 및 OrderDetailDisplay 표시 템플리트를 반출하려면 다음을 수행하십시오.

1. WebSphere Studio Application Developer를 열고 J2EE 네비게이터 보기로 전환하십시오.
2. **Stores** 폴더를 펼치십시오.
3. **Web Content** 폴더에서 마우스 오른쪽 버튼을 누르고 **반출**을 선택하십시오.
   반출 마법사가 열립니다.
4. 반출 마법사에서 다음을 수행하십시오.
   a. **파일 시스템**을 선택하고 **다음**을 누르십시오.
   b. 전개할 다음 자원을 선택하십시오.
      - Web Content*FashionFlow_name*\ShoppingArea\ CheckoutSection\StandardCheckoutSubsection\ OrderSubmitForm.jsp

- Web Content*FashionFlow_name*\ShoppingArea\CheckoutSection\StandardCheckoutSubsection\OrderSubmitFormInclude.jsp
- Web Content*FashionFlow_name*\UserArea\ServiceSection\TrackOrderStatusSubsection\OrderDetailDisplay.jsp
- Web Content*FashionFlow_name*\UserArea\ServiceSection\TrackOrderStatusSubsection\OrderDetailDisplayInclude.jsp
- Web Content\WEB-INF\lib\jstl.jar
- Web Content\WEB-INF\lib\standard.jar
- Web Content\WEB-INF\classes*FashionFlow_name*\ordergift.properties

c. **파일의 디렉토리 구조 작성**을 선택하십시오.

d. 디렉토리 필드에 자원을 놓을 임시 디렉토리를 입력하십시오. 예를 들어, `C:\ExportTemp3`를 입력하십시오.

e. **완료**를 누르십시오.

대상 WebSphere Commerce Server로 자원 전송

이 단계에서는 대상 WebSphere Commerce Server에서 임시 디렉토리를 작성한 후 이 디렉토리에 선물 주문 자원을 복사합니다. 그 다음 단계에서는 여러 유형의 코드를 WebSphere Commerce 응용프로그램 내의 적정 위치에 배치합니다.

개발 시스템에서 대상 WebSphere Commerce Server로 파일을 복사하려면 다음을 수행하십시오.

1. 대상 WebSphere Commerce Server에서 *drive*:\ImportTemp3라고 하는 임시 디렉토리를 작성하십시오.
2. 컴퓨터 사이에 파일을 복사할 방법을 판별하십시오. 대상 WebSphere Commerce Server의 드라이브를 개발 시스템에 맵핑하거나 FTP 응용프로그램(구성된 경우)을 사용하여 이를 수행할 수 있습니다.
3. 개발 시스템에서 *drive*:\ExportTemp3 내용을 대상 WebSphere Commerce Server의 *drive*:\ImportTemp3로 복사하십시오.

대상 WebSphere Commerce Server 중지

전개 단계를 시작하기 전에, 명령행에서 stopServer 명령을 실행하여 대상 WebSphere Commerce Server를 중지해야 합니다. 이 명령에 대한 자세한 정보는 *WebSphere Commerce Studio 설치 안내서*를 참조하십시오.

대상 WebSphere Commerce Server에서 데이터베이스 갱신

태스크 명령 구현 등록

명령 레지스트리를 수정하려면 다음을 수행하십시오.

1. DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 MyExtOrderProcessCmdImpl을 등록하십시오.
   a. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령행 도구 > 명령 센터**)를 여십시오.
   b. **도구** 메뉴에서 **도구 설정**을 선택하십시오.
   c. **명령문 종료 문자 사용** 선택란을 선택한 후 지정된 문자가 세미콜론(**;**)인지 확인하십시오.
   d. 도구 설정을 닫으십시오.
   e. 스크립트 탭을 선택하고 스크립트 창에 다음 정보를 입력하여 URLREG 테이블에 필수 항목을 작성하십시오.

```
connect to targetDB user dbuser using dbpassword;
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FashionFlow_storeent_ID,
   'com.ibm.commerce.order.commands.ExtOrderProcessCmd',
   'This is a new task command for tutorial two.',
   'com.ibm.commerce.sample.commands.MyExtOrderProcessCmdImpl',
   'local');
```

   여기서,

   - *targetDB*는 대상 데이터베이스의 이름입니다.
   - *dbuser*는 데이터베이스 사용자입니다.
   - *dbpassword*는 데이터베이스 사용자의 암호입니다.
   - *FashionFlow_storeent_ID*는 견본 상점의 상점 식별자입니다.

실행 아이콘을 누르십시오. 명령 센터를 열린 상태로 두십시오.

2. Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 MyOrderItemAddCmdImpl을 등록하십시오.
   a. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).
   b. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
   c. **암호** 필드에 Oracle 암호를 입력하십시오.
   d. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
   e. SQL Plus 창에서 다음 SQL문을 입력하십시오.

```
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FashionFlow_storeent_ID,
   'com.ibm.commerce.order.commands.ExtOrderProcessCmd',
   'This is a new task command for tutorial two.',
   'com.ibm.commerce.sample.commands.MyExtOrderProcessCmdImpl',
   'local');
```

   SQL문을 실행하려면 Enter를 누르십시오.
   f. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

   Enter를 눌러 SQL문을 실행하십시오.

XORDGIFT 테이블 작성

이 단계에서는 XORDGIFT 테이블을 작성합니다.

DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 테이블을 작성하십시오.

1. 스크립트 창에서 다음을 입력하십시오.

```
create table XORDGIFT (ORDERSID bigint not null, RECEIPTNAME varchar(50),
SENDERNAME varchar(50), MSGFIELD1 varchar(50), MSGFIELD2 varchar(50),
constraint p_xordgift primary key (ORDERSID),
constraint f_xordgift foreign key (ORDERSID) references ORDERS(ORDERS_ID)
on delete cascade);
insert into XORDGIFT (ORDERSID) (select ORDERS_ID from ORDERS);
```

여기서,

- *targetDB*는 대상 데이터베이스의 이름입니다.
- *dbuser*는 데이터베이스 사용자입니다.
- *dbpassword*는 데이터베이스 사용자의 암호입니다.

실행 아이콘을 누르십시오.
XORDGIFT 테이블이 작성됩니다.

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 테이블을 작성하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 SQL문을 입력하십시오.

```
create table XORDGIFT (ORDERSID NUMBER NOT NULL, RECEIPTNAME VARCHAR(50),
SENDERNAME VARCHAR(50), MSGFIELD1 VARCHAR(50), MSGFIELD2 VARCHAR(50),
constraint p_xordgift primary key (ORDERSID),
constraint f_xordgift foreign key (ORDERSID) references ORDERS(ORDERS_ID)
on delete cascade);
insert into XORDGIFT (ORDERSID) (select ORDERS_ID from ORDERS);
```

   Enter를 눌러 SQL문을 실행하십시오. XORDGIFT 테이블이 작성됩니다.
6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

   Enter를 눌러 SQL문을 실행하십시오.

대상 WebSphere Commerce Server에서 상점 자원 갱신

이 단계에서는 다음과 같이 상점을 수정한 상점 자원으로 갱신합니다.

1. *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear\ Stores.war 디렉토리를 백업하십시오.
2. *drive*:\ImportTemp3\Stores\Web Content 디렉토리로 이동하십시오.

3. *FashionFlow_name* 폴더를 다음 디렉토리에 복사하십시오.
   WAS_installdir\installedApps*cellName*\WC_*instanceName*.ear\ Stores.war

여기서, *instanceName*은 WebSphere Commerce 인스턴스 이름입니다.

대상 WebSphere Commerce Server에서 명령 및 데이터 bean JAR 파일 갱신

이 단계에서는 다음과 같이 새 명령과 데이터 bean JAR 파일을 사용하도록 대상 WebSphere Commerce Server를 갱신합니다.

1. 다음과 같이 기존 JAR 파일의 백업 사본을 작성해야 합니다.
   a. *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 디렉토리로 이동하십시오.
   b. WebSphereCommerceServerExtensionsLogic.jar 파일의 사본을 작성하고 백업 위치에 저장하십시오.
2. *drive*:\ImportTemp3 디렉토리의
   새 WebSphereCommerceServerExtensionsLogic.jar 파일을
   WAS_installdir\nstalledApps*cellName*\WC_*instanceName*.ear 디렉토리에 복사하십시오.

여기서, *instanceName*은 WebSphere Commerce 인스턴스 이름입니다.

대상 WebSphere Commerce Server에서 EJB JAR 파일 갱신

이 단계에서는 다음과 같이 새 EJB JAR 파일을 사용하도록 대상 WebSphere Commerce Server를 갱신합니다.

1. 다음과 같이 기존 JAR 파일의 백업 사본을 작성해야 합니다.
   a. *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 디렉토리로 이동하십시오.
   b. WebSphereCommerceServerExtensionsData.jar 파일의 사본을 작성하고 백업 위치에 저장하십시오.

   여기서, *instanceName*은 WebSphere Commerce 인스턴스 이름입니다.

2. *drive*:\ImportTemp3 디렉토리의
   새 WebSphereCommerceServerExtensionsData.jar 파일을
   WAS_installdir\installedApps*cellName*\WC_*instanceName*.ear 디렉토리에 복사하십시오.
3. 다음과 같이 EJB 전개 설명자 정보를 수정해야 합니다.
   a. WebSphere Application Server 셀의 전개 저장소(META-INF 디렉토리)를 찾으십시오. 일반적으로 다음과 같은 양식을 사용합니다.
      WAS_installdir\config\cells*cellName* \applications\WC_*instance_name*.ear\deployments\ WC_*instance_name**EJBModuleName*.jar\META-INF
      다음은 그 예입니다.
      D:\WebSphere\AppServer\config\cells\myCell\applications\WC_demo.ear\deployments\WC_demo\WebSphereCommerceServerExtensionsData.jar\META-INF입니다.
      여기서,
      - mycell은 WebSphere Application Server 셀 이름이고,
      - demo는 WebSphere Commerce 인스턴스 이름입니다.

   b. 디렉토리에는 다음 파일이 있습니다.
      - ejb-jar.xml
      - ibm-ejb-access-bean.xmi
      - ibm-ejb-jar-bnd.xmi
      - ibm-ejb-jar-ext.xmi
      - MANIFEST.MF

      이러한 모든 파일의 백업

   c. 도구를 사용하여 새 WebSphereCommerceServerExtensionsData.jar 파일을 열고 그 내용을 보십시오.
   d. 이 WebSphereCommerceServerExtensionsData.jar 파일에서 meta-inf 디렉토리의 내용을 3a단계의 디렉토리로 추출하십시오.
4. 명령행에서 WebSphere Application Server startServer 명령을 사용하여 WebSphere Commerce 인스턴스를 다시 시작하십시오.

대상 WebSphere Commerce Server에서 선물 메시지 기능 검증

이 절에서는 다음을 수행하여 선물 메시지 로직이 대상 WebSphere Commerce Server에서 올바르게 기능하는지 검증합니다.

1. 브라우저를 열고 FashionFlow 견본 상점을 기반으로 하는 상점의 URL을 입력하십시오.
2. 새 사용자로 로그온하십시오. 예를 들어, ″등록″을 누른 후 사용자 ″shopper″를 작성하십시오.
3. 새로 등록한 사용자로서 상점을 찾아보고 항목을 장바구니에 추가한 후 구매를 완료하십시오. 다음 화면에 표시된 것처럼 주문에 선물 메시지를 추가할 수 있습니다.

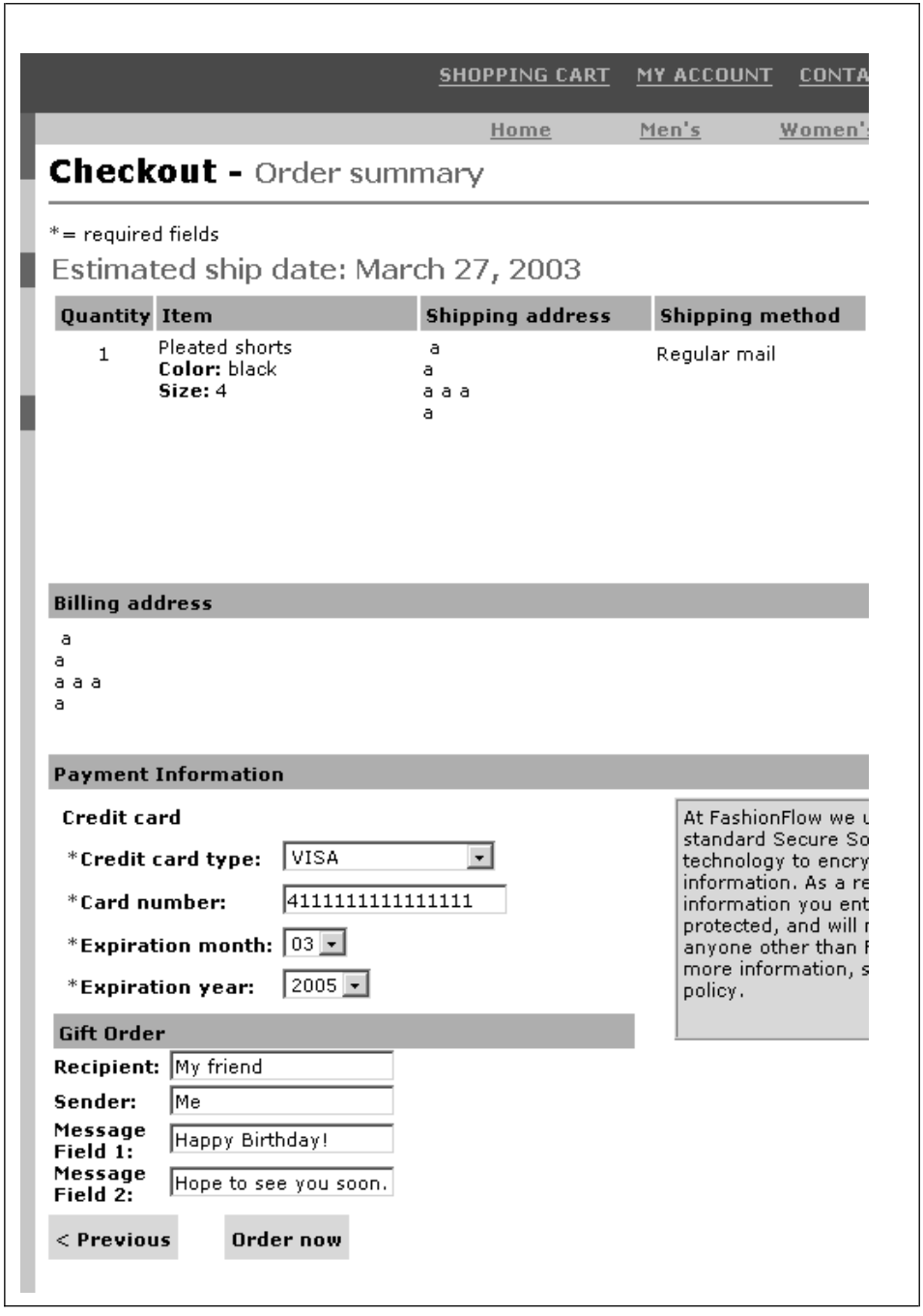


그림 51.

이 주문과 연관되는 주문 번호를 기록하십시오.

4. **회원 정보**를 누르십시오.
5. **주문 보기**를 누르십시오.
6. 3단계에서 작성한 주문을 선택하십시오.다음과 유사한 화면이 표시됩니다.

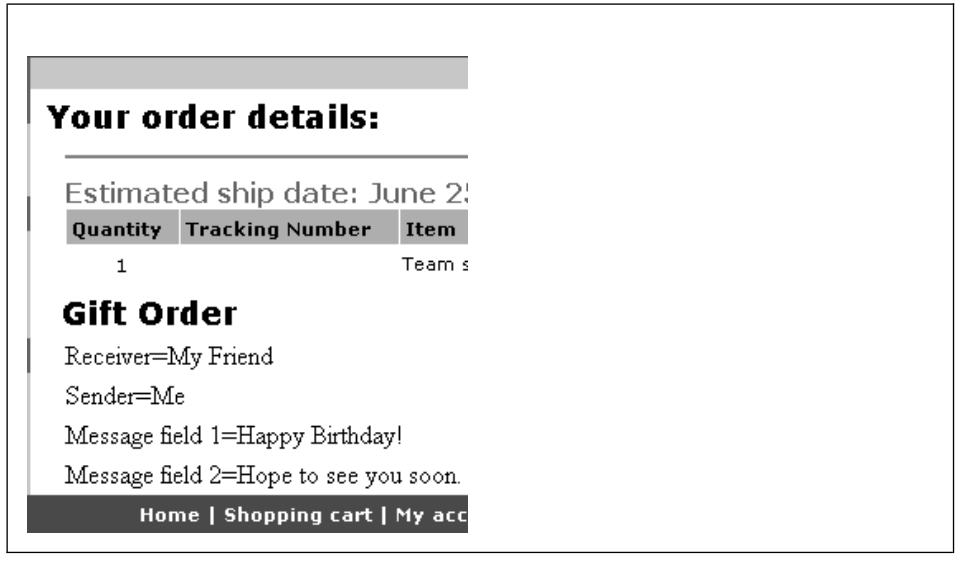


그림 52.

제 13 장 학습: 기존 WebSphere Commerce 엔티티 bean 확장

이 학습에서는 기존 WebSphere Commerce 엔티티 bean 수정에 사용되는 프로세스를 학습합니다. 이것은 사용자 등록 프로세스에 추가 주택 정보 조사를 추가하는 시나리오를 사용합니다. 이 경우에 사용자 엔티티 bean은 사용자의 주택 유형 값과 위치 값을 저장하는 추가 필드를 포함하도록 수정됩니다. 새 데이터베이스 테이블이 작성되고(XHOUSING이라고 부름) 사용자 bean에 대한 기존 맵핑 정보가 이 새 테이블을 포함하도록 수정됩니다.

새 테이블 및 엔티티 bean 변경과 더불어, 새로운 MyPostUserRegistrationAddCmdImpl 구현 클래스가 작성됩니다. 이것은 주택 조사 정보를 처리하고 데이터베이스를 새 정보로 갱신하는 로직을 포함합니다.

주택 정보를 수집하고 후에 그 결과를 표시할 수 있도록, UserRegistrationAddForm.jsp 및 UserRegistrationUpdateForm.jsp 파일이 수정됩니다.

전제 조건

이 학습을 완료하지 않아도 됩니다. 학습을 완료한 경우, 코드를 작업 영역에 두어도 이 학습과 충돌하지 않으므로 문제가 되지 않습니다.

이 학습을 시작하기 전에 FashionFlow 견본 상점을 기초로 상점을 공개해야 합니다. 이 상점 내에서 구매를 완료할 수 있어야 합니다(예를 들어, 카탈로그를 찾아보고 항목을 장바구니에 추가한 후 주문을 시작하여 주문 확인이 표시되어야 합니다).

XHOUSING 테이블 작성 및 대량 자료 반입

엔티티 bean 작성 준비 시, 먼저 새 데이터베이스 테이블을 작성하고 대량 자료 반입해야 합니다. 작성할 테이블을 XHOUSING이라고 합니다.

DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 테이블을 작성하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령행 도구 > 명령 센터**)를 열고 **스크립트** 탭을 누르십시오.
2. 스크립트 창에서 다음을 입력하십시오.

```
connect to developmentDB user dbuser using dbpassword;
create table XHOUSING (MEMBERID bigint not null,
    HOUSINGTYPE integer, LOCATION integer,
   constraint p_xhousing primary key (MEMBERID),
   constraint f_xhousing foreign key (MEMBERID)
    references USERS (USERS_ID) on delete cascade);
insert into XHOUSING (MEMBERID) (select USERS_ID from USERS);
```

여기서,

- *developmentDB*는 개발 데이터베이스의 이름입니다.
- *dbuser*는 데이터베이스 사용자입니다.
- *dbpassword*는 데이터베이스 사용자의 암호입니다.

실행 아이콘을 누르십시오.
SQL문이 성공적으로 완료되었음을 표시하는 메시지가 표시되어야 합니다.

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 테이블을 작성하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작 > 프로그램 > Oracle > Application Development > SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 SQL문을 입력하십시오.

```
create table XHOUSING (MEMBERID number not null,
    HOUSINGTYPE integer, LOCATION integer,
   constraint p_xhousing primary key (MEMBERID),
   constraint f_xhousing foreign key (MEMBERID)
    references USERS (USERS_ID) on delete cascade);
insert into XHOUSING (MEMBERID) (select USERS_ID from USERS);
```

Enter를 눌러 SQL문을 실행하십시오. XHOUSING 테이블이 작성됩니다.

6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

   ```
   commit;
   ```

   Enter를 눌러 SQL문을 실행하십시오.

User 엔티티 bean에 새 필드 추가

이 절에서는 주택 정보를 캡처하기 위해 사용하는 사용자 엔티티 bean에 두 개의 새 필드를 추가합니다. 새 필드는 housingType과 location입니다. 이 두 필드는 결국 XHOUSING 테이블의 HOUSINGTYPE 및 LOCATION 열에 맵핑됩니다.

새 필드를 추가하려면 다음을 수행하십시오.

1. 이미 열려 있지 않은 경우 WebSphere Commerce Studio를 시작하십시오(**시작 > 프로그램 > IBM WebSphere Commerce Studio > WebSphere Commerce Development Environment**).
2. 서버 perspective로 전환하고 WebSphereCommerceServer 테스트 서버가 중지되었는지 확인하십시오.
3. J2EE perspective로 전환하고 J2EE 계층 보기를 선택하십시오.
4. **EJB 모듈**을 펼친 후 **Member-MemberManagementData** EJB 프로젝트를 두 번 누르십시오. 그러면 EJB 전개 설명자 편집기가 열립니다.
5. **Bean** 탭을 누르고 표시되는 bean 목록에서 **사용자** bean을 선택하십시오.
6. CMP 필드 텍스트 상자 옆에 있는 **추가**를 누르십시오.
   CMP 속성 작성 창이 열립니다.
7. 다음 특성을 갖는 CMP 필드를 작성하십시오.
   a. **이름** 필드에 `housingType`을 입력하십시오.
   b. 유형 필드에 `java.lang.Integer`를 입력하십시오.
   c. **getter 및 setter 메소드로 액세스**를 사용하십시오.
   d. **getter 및 setter를 원격 인터페이스로 승격** 선택란을 지우십시오(그러면 getter를 읽기 전용 옵션으로 만들기 위한 옵션이 자동으로 지워집니다).
   e. **확인**을 누르십시오.

8. 다시 **추가**를 눌러서 다음 특성을 가진 다른 CMP 필드를 작성하십시오.
   a. **이름** 필드에 location을 입력하십시오.
   b. 유형 필드에 java.lang.Integer를 입력하십시오.
   c. **getter 및 setter 메소드로 액세스**를 사용하십시오.
   d. **getter 및 setter를 원격 인터페이스로 승격** 선택란을 지우십시오(그러면 getter를 읽기 전용 옵션으로 만들기 위한 옵션이 자동으로 지워집니다).
   e. **확인**을 누르십시오.

   CMP 필드 목록에 새 필드가 표시됩니다.
9. 변경사항을 저장하고 EJB 전개 설명자 편집기를 닫으십시오.

스키마 및 테이블 맵핑 정보 갱신

다음 절에서는 새 XHOUSING 테이블로 사용자 스키마를 갱신하고, 새 테이블의 foreign key 관계를 작성하며 User 엔티티 bean의 필드와 XHOUSING 테이블 열 사이의 테이블 맵을 작성합니다. 이런 접근 방식으로 오브젝트 모델을 확장하여 새 필드가 직접 USERS 테이블에 추가된 것처럼 코드에 표시됩니다.

XHOUSING 테이블의 테이블 정의 작성

XHOUSING 테이블 정의를 작성하고 USERS 테이블과 XHOUSING 테이블 사이의 foreign key 관계를 작성하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **데이터베이스**를 펼치십시오.
2. **Member-MemberManagementData** 데이터베이스를 펼친 후 **NULLID** 스키마를 펼치십시오.
3. **테이블**에서 마우스 오른쪽 버튼을 누르고 **새로 만들기 > 새 테이블 정의**를 선택하십시오.
   테이블 정의 창이 열립니다.
4. **테이블 이름** 필드에 XHOUSING을 입력한 후 **다음**을 누르십시오.
5. 다음과 같이 키 열을 테이블 정의에 추가해야 합니다.
   a. **추가하기**를 누르십시오.
   b. **열 이름** 필드에 MEMBERID를 입력하십시오.

c. DB2 **열 유형** 드롭 다운 목록에서 BIGINT를 선택하십시오.

   Oracle **열 유형** 드롭 다운 목록에서 NUMBER를 선택하십시오.

d. **키 열**을 선택하십시오.

e. Oracle **수치 정밀도** 필드에 38을 입력하십시오.

f. Oracle **수치 스케일**의 값을 0으로 두십시오.

6. 다음과 같이 테이블에 열을 더 추가하십시오.

   a. **추가하기**를 누르고 다음 특성으로 열을 작성하십시오.

표 26.

특성	값
열 이름	HOUSINGTYPE
열 유형	INTEGER
널 가능	Select

   b. **추가하기**를 누르고 다음 특성으로 열을 작성하십시오.

표 27.

특성	값
열 이름	LOCATION
열 유형	INTEGER
널 가능	Select

   c. **다음**을 누르십시오.

7. **1차 키 이름** 필드에 p_xhousing을 입력하고 **다음**을 누르십시오.
8. **추가하기**를 눌러 foreign key를 추가하십시오. 다음 값을 지정하십시오.

   a. **foreign key 이름** 필드에 f_xhousing을 입력하십시오.

   b. **삭제** 드롭 다운 목록에서 **CASCADE**를 선택하십시오.

   c. **대상 테이블** 드롭 다운 목록에서 **NULLID.USERS**를 선택하십시오.

   d. 소스 열 분할창에서 **MEMBERID**를 누른 후 >를 눌러 foreign key를 추가하십시오.

9. **완료**를 누르십시오.

10. Oracle 다음과 같이 텍스트 편집기를 사용하여 테이블 정의를 편집해야 합니다.
    a. J2EE 네비게이터 보기로 전환하십시오.
    b. **Member-MemberManagementData** 프로젝트를 펼치십시오.
    c. **ejbModule** > **META-INF** > **Schema**를 펼치십시오.
    d. **Member-MemberManagementData_NULL_XHOUSING.xmi** 파일에서 마우스 오른쪽 버튼을 누르고 **열기** > **텍스트 편집기**를 선택하십시오.
    e. 모든 `SQLNumeric_6` 어커런스를 `SQLNumeric_3`으로 바꾸십시오.
    f. 변경사항을 저장하고 텍스트 편집기를 닫으십시오.

Member-MemberManagementData/NULLID 아래에 있는 Tables 폴더를 펼치면 새 Member-MemberManagementData_NULL_XHOUSING 테이블 정의를 볼 수 있습니다.

XHOUSING 테이블 맵 작성

이 절에서는 XHOUSING 테이블의 두 열과 사용자 엔티티 bean의 두 필드(housingType 및 location) 사이의 맵핑을 작성합니다.

다음 맵핑을 작성하려면 다음을 수행하십시오.

1. J2EE 네비게이터 보기로 전환하십시오.
2. **Member-MemberManagementData** > **ejbModule** > **META-INF** 폴더를 펼치십시오.
3. **Map.mapxmi** 파일을 두 번 누르십시오.
4. 테이블 분할창에서 다음 테이블을 강조표시하십시오.
    - **MEMBER**
    - **USERS**
    - **XHOUSING**

    (Ctrl 키를 누른 상태에서 여러 테이블을 한번에 선택하십시오.)
5. 엔터프라이즈 bean 분할창(편집기 맨 위에 있는)에서 **구성원** 그룹을 펼친 후 **사용자** 엔티티 bean에서 마우스 오른쪽 버튼을 누르고 **맵핑 작성**을 선택하십시오.

6. **User** 엔티티 bean을 펼치십시오.
7. 테이블 분할창에서 해당 열을 볼 수 있도록 **XHOUSING** 테이블을 펼치십시오.
8. **housingType** bean 속성을 강조표시하고 이를 **HOUSINGTYPE** 열로 끌어서 맵핑을 작성하십시오.
9. **location** bean 속성을 강조표시하고 이를 **LOCATION** 열로 끌어서 맵핑을 작성하십시오.
10. 변경사항을 저장한 후 Map.mapxmi 파일을 닫으십시오.

맵핑 파일 갱신

1. **Member-MemberManagementData > ejbModule >META-INF** 폴더를 펼치십시오.
2. **Map.mapxmi** 파일에서 마우스 오른쪽 버튼을 누르고 **열기 프로그램 > 텍스트 편집기**를 선택하십시오.
3. 다음 문자열을 찾으십시오.

   ```
   User_EJB
   ```

   이 문자열 두 행 아래에서 다음을 볼 수 있습니다.

   ```
   <discriminatorValues>User</discriminatorValues>
   ```

   앞의 행에서 찾았다면 그 다음 이를 다음과 같이 변경합니다.

   ```
   <discriminatorValues>'U'</discriminatorValues>
   ```
4. 변경사항을 저장하십시오.

액세스 bean 및 전개 코드 작성

User 엔티티 bean을 수정했으므로, 해당 액세스 bean 및 전개 코드를 다시 작성해야 합니다.

액세스 bean을 다시 작성하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 **EJB 모듈**을 펼치십시오.
2. **Member-MemberManagementData** EJB 모듈에서 마우스 오른쪽 버튼을 누르고 **액세스 bean > 액세스 bean 편집**을 선택하십시오.

3. **CopyHelper**를 선택하고 **다음**을 누르십시오.
4. EJB 프로젝트 선택 창에서 **다음**을 누르십시오.
5. 액세스 Bean 복사 헬퍼 창에서 다음을 수행하십시오.
   a. 엔터프라이즈 bean 드롭 다운 목록에서 **사용자**를 선택하십시오.
   b. **생성자 메소드** 드롭 다운 목록에서 **findByPrimaryKey(com.ibm.commerce.user.objects.MemberKey)**를 선택하십시오.
   c. 속성 헬퍼 목록에서 **housingType** 및 **location** 속성을 나머지 속성과 함께 선택했는지 확인하십시오.
6. **완료**를 누르십시오.
7. **Member-MemberManagementData** EJB 모듈을 마우스 오른쪽 버튼으로 누르고 **액세스 bean> 액세스 bean 다시 작성**을 선택하십시오.
8. **모두 선택**을 누르고 **완료**를 누르십시오.

전개된 코드를 다시 작성하려면 다음을 수행하십시오.

1. **Member-MemberManagementData** EJB 모듈에서 마우스 오른쪽 버튼을 누르고 **작성 > 전개 및 RMIC 코드**를 선택하십시오.
2. **모두 선택**을 누르고 **완료**를 누르십시오.

MyPostUserRegistrationAddCmdImpl 구현 작성

이 단계에서는 UserRegistrationAddCmd 컨트롤러 명령을 확장하여 등록 양식에서 수집한 새 주택 정보를 구문 분석하기 위해 사용하는 새 로직을 포함시킵니다. PostUserRegistrationCmd 인터페이스를 구현하는 새 MyPostUserRegistrationAddCmdImpl 구현 클래스를 작성하여 확장됩니다. 이 새 구현 클래스를 작성하고 나면 명령 레지스트리를 갱신하여 이러한 변경을 반영해야 합니다. 다른 학습처럼, 이 새 명령에 해당하는 코드가 제공됩니다.

새 MyPostUserRegistrationAddCmdImpl 구현 클래스를 작성하려면 다음을 수행하십시오.

1. 252 페이지의 『견본 코드 찾기』의 단계를 완료했는지 확인하십시오.
2. WebSphere Studio Application Developer에서 Java perspective를 여십시오(**Window > Perspective 열기 > Java**).

3. **WebSphereCommerceServerExtensionsLogic** 프로젝트를 펼치십시오.
4. **src** 폴더에서 마우스 오른쪽 버튼을 누르고 **반입**을 선택하십시오.
   가져오기 마법사가 열립니다.
5. **반입 소스 선택** 목록에서 **Zip 파일**을 선택하고 **다음**을 누르십시오.
6. **찾아보기(Zip 파일** 필드 옆에 있는)를 누르고 견본 코드를 탐색하십시오. 이 파일은 다음 위치에 지정됩니다.
   yourDirectory\WC_SAMPLE_55.zip
   여기서, *yourDirectory*는 패키지를 다운로드한 디렉토리입니다.
7. **모두 선택 취소**를 누른 후 디렉토리를 펼쳐 다음 파일을 반입하도록 선택하십시오.
   - com\ibm\commerce\sample\commands\
     MyPostUserRegistrationAddCmdImpl.java
8. **폴더** 필드에서는 이미 WebSphereCommerceServerExtensionsLogic/src 폴더가 지정되어 있습니다. 이 값을 변경하지 마십시오.
9. **완료**를 누르십시오.
10. **com.ibm.commerce.sample.commands** 패키지를 펼치십시오.
11. 새 **MyPostUserRegistrationAddCmdImpl** 클래스를 두 번 누르십시오.
12. Section 1 주석을 제거하십시오. 이 코드는 변수를 설정하고 이 변수에 대한 getter 및 setter를 작성합니다.

```
/// Section 1 ////////////////
 Integer housingType = null;
 Integer location = null;

 public Integer getHousingType() {
  return housingType;
 }

 public Integer getLocation() {
  return location;
 }

 public void setHousingType(Integer newHousingType) {
  housingType = newHousingType;
 }
```

```
 public void setLocation(Integer newLocation) {
  location = newLocation;
 }
/// End of Section 1 ////////////////
```

13. Section 2 주석을 제거하십시오. 이것은 다음 코드를 클래스로 도입합니다.

```
/// Section 2 ////////////////
public void performExecute() throws ECException {
  super.performExecute();

  // Set the needed fields before processing the survey
  setHousingType(requestProperties.getInteger("housingType", 0));
  setLocation(requestProperties.getInteger("location", 0));

  processSurvey();
 }

/// End of Section 2 ////////////////
```

앞의 코드는 상위 클래스의 performExecute 메소드를 호출하여 PostUserRegistrationAddCmdImpl의 정규 로직이 실행되도록 합니다. 완료되면 새 processSurvey 메소드가 호출됩니다.

14. 클래스에 다음 코드가 도입되도록 3 주석을 제거하십시오.

```
/// Section 3 ////////////////

 private void processSurvey() throws ECException {

  try {
   // load up the user data
   UserAccessBean abUser = new UserAccessBean();
   abUser.setInitKey_MemberId(commandContext.getUserId().toString());
   abUser.refreshCopyHelper();

   // store the new attributes
   abUser.setHousingType(getHousingType());
   abUser.setLocation(getLocation());

   abUser.commitCopyHelper();
} catch (javax.ejb.FinderException e) {
   throw new ECSystemException(
    ECMessage._ERR_FINDER_EXCEPTION,
    this.getClass().getName(),
    "processSurvey");
} catch (javax.naming.NamingException e) {
   throw new ECSystemException(
    ECMessage._ERR_NAMING_EXCEPTION,
    this.getClass().getName(),
```

```
        "processSurvey");
    } catch (java.rmi.RemoteException e) {
       throw new ECSystemException(
        ECMessage._ERR_REMOTE_EXCEPTION,
        this.getClass().getName(),
        "processSurvey");
    } catch (javax.ejb.CreateException e) {
       throw new ECSystemException(
        ECMessage._ERR_CREATE_EXCEPTION,
        this.getClass().getName(),
        "processSurvey");
      }

     }
    /// End of Section 3 ////////////////
```

15. 변경사항을 저장하십시오.
16. **WebSphereCommerceServerExtensionsLogic** 프로젝트에서 마우스 오른쪽 버튼을 누른 후 **프로젝트 빌드**를 선택하십시오.

명령 레지스트리 수정

새 구현 클래스가 쇼핑 플로우에서 사용되도록 명령 레지스트리를 수정해야 합니다.

명령 레지스트리를 수정하려면 다음을 수행하십시오.

1. DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 MyPostUserRegistrationAddCmdImpl을 등록하십시오.
   a. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령 센터**)를 여십시오.
   b. **도구** 메뉴에서 **도구 설정**을 선택하십시오.
   c. **명령문 종료 문자 사용** 선택란을 선택한 후 지정된 문자가 세미콜론(**;**)인지 확인하십시오.
   d. 스크립트 탭을 선택하고 스크립트 창에 다음 정보를 입력하여 URLREG 테이블에 필수 항목을 작성하십시오.

      ```
      connect to developmentDB user dbuser using dbpassword;
      insert into CMDREG values (FashionFlow_storeent_Id,
      'com.ibm.commerce.usermanagement.commands.PostUserRegistrationAddCmd'
      'Command for modified user bean tutorial',
      'com.ibm.commerce.sample.commands.MyPostUserRegistrationAddCmdImpl',
      null, null, 'Local');
      ```

      여기서,

- *developmentDB*는 개발 데이터베이스의 이름입니다.
- *dbuser*는 데이터베이스 사용자입니다.
- *dbpassword*는 데이터베이스 사용자의 암호입니다.
- *FashionFlow_storeent_Id*는 상점의 고유한 상점 엔티티 식별자입니다.

실행 아이콘을 누르십시오.

2. Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 MyPostUserRegistrationAddCmdImpl을 등록하십시오.
   a. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).
   b. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
   c. **암호** 필드에 Oracle 암호를 입력하십시오.
   d. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
   e. SQL Plus 창에서 다음 SQL문을 입력하십시오.

```
insert into CMDREG values (FashionFlow_storeent_Id,
'com.ibm.commerce.usermanagement.commands.PostUserRegistrationAddCmd'
'Command for modified user bean tutorial',
'com.ibm.commerce.sample.commands.MyPostUserRegistrationAddCmdImpl',
null, null, 'Local');
```

   SQL문을 실행하려면 Enter를 누르십시오.
   f. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

   Enter를 눌러 SQL문을 실행하십시오.

주택 정보를 수집 및 표시하도록 JSP 템플리트 수정

이 단계에서는 고객이 로그인할 때 주택 정보를 입력하고 요약 페이지에서 주택 정보를 볼 수 있도록 UserRegistrationAddForm 및 UserRegistrationUpdateForm 템플리트를 수정합니다. 이런 페이지 수정 방법은 페이지의 새 정보를 지정하는 추가 JSP 템플리트를 포함하는 것입니다. 이런 새 페이지 (UserRegistrationAddFormInclude.jsp 및 UserRegistrationUpdateFormInclude.jsp)는 JSTL을 사용하여 새 정보를 표시합니다.

페이지를 수정하려면 다음을 수행하십시오.

1. 웹 perspective로 전환하십시오.
2. 251 페이지의 제 10 장『학습: 새 비즈니스 로직 작성』을 완료하지 않았다면 다음과 같이 웹 프로젝트 저장 특성을 수정해야 합니다.
   a. **Stores** 웹 프로젝트에서 마우스 오른쪽 버튼을 누른 후 **특성**을 선택하십시오.
   b. 왼쪽 분할창에서 **웹**을 선택한 후 사용 가능한 웹 프로젝트 특징 목록에서 **JSP 표준 태그 라이브러리 포함**을 선택하십시오. **적용**을 누르십시오. 갱신이 완료되면 **확인**을 눌러 특성 편집기를 닫으십시오.
3. 다음 디렉토리로 펼치십시오.
   Stores\Web Content*FashionFlow_name*.
4. 다음을 수행하여 UserRegistrationAddForm.jsp 파일의 백업 사본을 작성하십시오.
   a. **UserArea > AccountSection > RegistrationSubsection** 디렉토리를 펼치십시오.
   b. **UserRegistrationAddForm.jsp** 파일에서 마우스 오른쪽 버튼을 누르고 **이름 바꾸기**를 선택하십시오.
   c. 이름 바꾸기 창에서 `UserRegistrationAddForm_bak.jsp`를 입력하고 **확인**을 누르십시오.
   d. 이 파일로 링크를 갱신하려는지 여부를 묻는 프롬프트에서 **아니오**를 누르십시오.
5. 다음을 수행하여 UserRegistrationUpdateForm.jsp 파일의 백업 사본을 작성하십시오.
   a. **UserArea>AccountSection>RegistrationSubsection** 디렉토리를 펼치십시오.
   b. **UserRegistrationUpdateForm.jsp** 파일에서 마우스 오른쪽 버튼을 누르고 **이름 바꾸기**를 선택하십시오.
   c. 이름 바꾸기 창에서 `UserRegistrationUpdateForm_bak.jsp`를 입력하고 **확인**을 누르십시오.

d. 이 파일로 링크를 갱신하려는지 여부를 묻는 프롬프트에서 **아니오**를 누르십시오.

6. *FashionFlow_name* 디렉토리에서 마우스 오른쪽 버튼을 누르고 **반입**을 선택하십시오.
   가져오기 마법사가 열립니다.
7. **반입 소스 선택** 목록에서 **Zip 파일**을 선택하고 **다음**을 누르십시오.
8. **찾아보기(Zip 파일** 필드 옆에 있는)를 누르고 견본 코드를 탐색하십시오. 이 파일은 다음 위치에 지정됩니다.
   yourDirectory\WC_SAMPLE_55.zip
   여기서, *yourDirectory*는 패키지를 다운로드한 디렉토리입니다.
9. **모두 선택 취소**를 누른 후 디렉토리를 펼쳐 반입할 다음 파일을 선택하십시오.
   - UserArea\AccountSection\RegistrationSubsection\UserRegistrationAddForm.jsp
   - UserArea\AccountSection\RegistrationSubsection\UserRegistrationAddFormInclude.jsp
   - UserArea\AccountSection\RegistrationSubsection\UserRegistrationUpdateForm.jsp
   - UserArea\AccountSection\RegistrationSubsection\UserRegistrationUpdateFormInclude.jsp
10. **폴더** 필드에는 이미 Stores/Web Content/*FashionFlow_name* 폴더가 지정되어 있습니다. 이 값을 변경하지 마십시오.
11. **완료**를 누르십시오.

UserRegistrationAddForm.jsp 파일을 검사할 때 다음 절이 이 학습에 추가된 것을 알게 될 것입니다.

```
<%-- Add for tutorial --%>
 <tr>
      <td colspan="3">
         <jsp:include page="UserRegistrationAddFormInclude.jsp" flush="true" />
  </td>
 </tr>
<%-- End of tutorial --%>
```

또한 UserRegistrationAddFormInclude.jsp 파일을 검사하여 JSTL을 이용한 새 정보 수집 방법을 알 수 있습니다. 이와 유사하게 UserRegistrationAddForm.jsp 및 UserRegistrationAddFormInclude.jsp 파일을 검사하십시오.

또한 수정된 JSP 템플리트에서 사용된 문자열 값이 들어 있는 특성 파일을 반입해야 합니다. 이 파일의 이름은 Housing.properties입니다. 파일을 반입하려면 다음을 수행하십시오.

1. J2EE 계층 보기에서 다음 디렉토리를 펼치십시오.
   Stores > **Web Content** > **WEB-INF** > **classes** > *FashionFlow_name*
2. *FashionFlow_name* 디렉토리에서 마우스 오른쪽 버튼을 누르고 **반입**을 선택하십시오.
   가져오기 마법사가 열립니다.
3. **반입 소스 선택** 목록에서 **Zip 파일**을 선택하고 **다음**을 누르십시오.
4. **찾아보기(Zip 파일** 필드 옆에 있는)를 누르고 견본 코드를 탐색하십시오. 이 파일은 다음 위치에 지정됩니다.
   yourDirectory\WC_SAMPLE_55.zip
   여기서, *yourDirectory*는 패키지를 다운로드한 디렉토리입니다.
5. **모두 선택 취소**를 누른 후 디렉토리를 펼쳐 반입할 다음 파일을 선택하십시오.
   - Housing.properties
6. **폴더** 필드에는 이미 `Stores/Web Content/WEB-INF/classes/`*`FashionFlow_name`* 폴더가 지정되어 있습니다. 이 값을 변경하지 마십시오.
7. **완료**를 누르십시오.

수정한 코드 테스트

이제 다음을 수행하여 수정된 등록 정보를 테스트해야 합니다.

1. 서버 perspective로 전환하십시오.
2. **WebSphereCommerceServer** 테스트 서버를 마우스 오른쪽 버튼으로 누르고 **시작**을 선택하십시오.
3. Stores\WebContent*FashionFlow_name* 디렉토리의 **index.jsp**를 마우스 오른쪽 버튼으로 누르고 **서버에서 실행**을 선택하십시오.

4. 상점의 홈 페이지가 열리면 **등록**을 누르십시오.
5. **등록**을 다시 눌러 새 사용자를 작성하십시오.
6. 다음 화면에 표시되는 것과 같이 수정된 등록 페이지(주택 조사가 포함됨)가 표시됩니다.

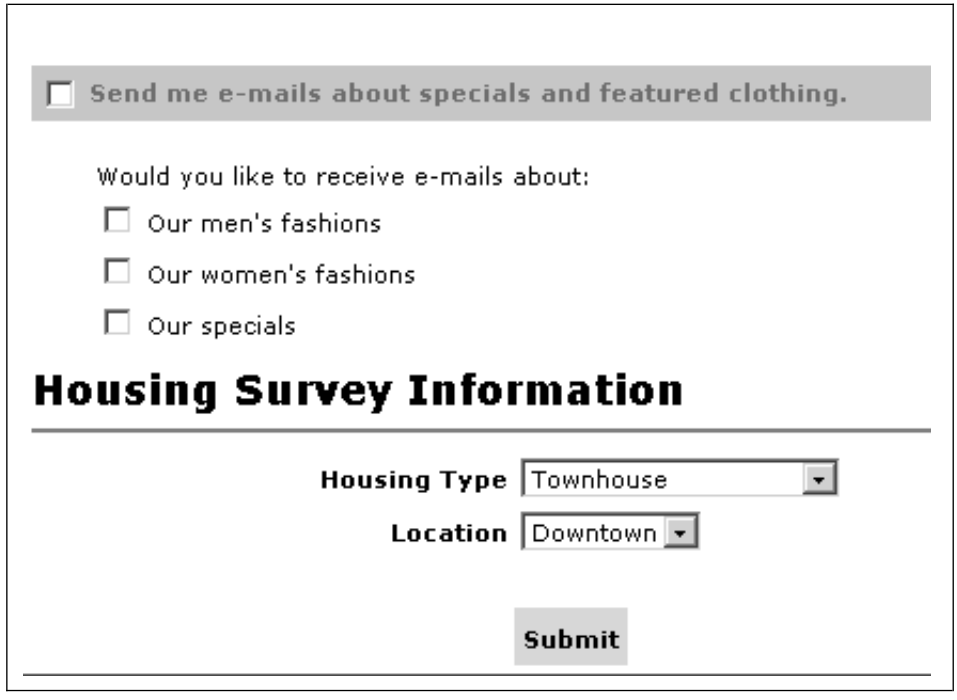


그림 53.

7. 새 사용자에 대한 적절한 정보를 입력하고 **제출**을 누르십시오.
8. 정보가 제출되면 **개인 정보 변경**을 눌러 주택 정보를 캡처했는지 검증하십시오. 다음과 같이 조사 응답의 요약을 표시하는 화면이 표시됩니다.

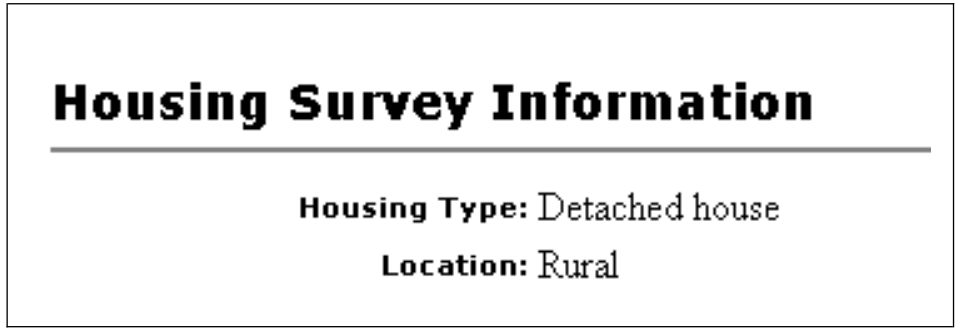


그림 54.

주택 조사 로직 전개

이 절에서는 원격 WebSphere Commerce Server에서 실행 중인 상점으로 새 비즈니스 로직을 전개하는 방법에 대해 설명합니다. 전개 단계를 시작하려면 먼저 원격 WebSphere Commerce Server에 상점을 작성해 놓아야 합니다(InFashion 견본 상점을 기초로).

개발 처리에는 대상 WebSphere Commerce Server에서 수행되는 단계 뿐만 아니라 개발 시스템에서 수행되는 단계도 포함됩니다.

대상 WebSphere Commerce Server에 전개해야 하는 여러 가지 유형의 자원이 있습니다. 자원 유형은 다음과 같습니다.

- 명령 로직
- 수정된 엔터프라이즈 bean 로직
- JSP 템플리트
- 특성 파일
- 스키마 갱신(새 테이블)과 명령 레지스트리 갱신을 포함한 데이터베이스 갱신사항

이 절에서는 이 모든 자원을 대상 WebSphere Commerce Server에 *증분적으로* 전개하는 방법에 대해 설명합니다.

명령 JAR 파일 작성

이 절에서는 새 MyPostUserRegistrationAddCmdImpl 로직이 들어 있는 JAR 파일 작성 방법에 대해 설명합니다.

JAR 파일을 작성하려면 개발 시스템에서 다음 단계를 수행하십시오.

1. 로컬 파일 시스템에 *drive*:\ExportTemp4 디렉토리를 작성하십시오.
2. WebSphere Studio Application Developer를 열고 J2EE 네비게이터 보기로 전환하십시오.
3. **WebSphereCommerceServerExtensionsLogic** 프로젝트에서 마우스 오른쪽 버튼을 누른 후 **반출**을 선택하십시오.
   반출 마법사가 열립니다.
4. 반출 마법사에서 다음을 수행하십시오.
   a. **JAR 파일**을 선택하고 **다음**을 누르십시오.
   b. **반출할 자원 선택?** 아래에 있는 왼쪽 분할창이 프로젝트 이름으로 채워집니다. 이 값은 있는 그대로 두십시오.
   c. 오른쪽 분할창에서 다음 자원만이 선택되었음을 확인하십시오.
      - .classpath
      - .project
      - .serverPreference
   d. **작성된 클래스 파일 및 자원 반출**이 선택되어 있는지 확인하십시오.
   e. Java 소스 파일 및 자원 **반출**을 선택하지 *마십시오*.
   f. **반출 대상 선택** 필드에서 사용할 완전한 JAR 파일 이름을 입력하십시오. 이 경우에는 *drive*:\ExportTemp4\WebSphereCommerceServerExtensionsLogic.jar을 입력하십시오. JAR 파일 이름은 WebSphereCommerceServerExtensionsLogic.jar이어야 합니다.
   g. **완료**를 누르십시오.

EJB JAR 파일 작성

EJB JAR 파일을 작성하려면 다음을 수행하십시오.

1. WebSphere Studio Application Developer를 열고 J2EE 네비게이터 보기로 전환하십시오.
2. **Member-MemberManagementData** 프로젝트를 펼치십시오.
3. **EJB 전개 설명자**를 두 번 누르십시오.
4. 개요 탭을 선택한 상태에서 분할창의 맨 아래로 화면이동하여 **WebSphere 바인딩** 섹션을 찾으십시오.
5. **DataSource JNDI 이름** 필드에 대상 WebSphere Commerce Server의 데이터소스 JNDI 이름을 입력하십시오. 다음은 예제 값입니다.

   DB2 `jdbc/WebSphere Commerce DB2 DataSource demo`

   여기서, 대상 WebSphere Commerce Server는 DB2 데이터베이스를 사용 중이고 WebSphere Commerce 인스턴스 이름은 “demo”입니다.

   Oracle `jdbc/WebSphere Commerce Oracle DataSource demo`

   여기서, 대상 WebSphere Commerce Server는 Oracle 데이터베이스를 사용 중이고 WebSphere Commerce 인스턴스 이름은 “demo”입니다.
6. 전개 설명자 변경사항을 저장하십시오(ctrl+s).
7. J2EE 네비게이터 보기에서 **Member-MemberManagementData** 프로젝트를 마우스 오른쪽 버튼으로 누르고 **반출**을 선택하십시오.
   반출 마법사가 열립니다.
8. 반출 마법사에서 다음을 수행하십시오.
   a. **EJB JAR 파일**을 선택하고 **다음**을 누르십시오.
   b. **반출할 자원은?** 값은 EJB 프로젝트의 이름으로 채워집니다. 이 값은 있는 그대로 두십시오.
   c. **자원 반출 위치** 필드에 사용할 완전한 JAR 파일 이름을 입력하십시오. 이 경우에는 *drive*:\ExportTemp4\Member-MemberManagementData.jar을 입력하십시오.
   d. **완료**를 누르십시오.
9. JAR 파일을 작성하였으면 5단계에서 수행한 로컬 전개 설명자 변경사항을 실행 취소하여 로컬 테스트 서버에 필요한 설정을 복원하십시오.

상점 자원 반출

수정된 JSP 템플리트 및 새 특성 파일을 반출하려면 다음을 수행하십시오.

1. WebSphere Studio Application Developer를 열고 J2EE 네비게이터 보기로 전환하십시오.
2. **Stores** 폴더를 펼치십시오.
3. **Web Content** 폴더에서 마우스 오른쪽 버튼을 누르고 **반출**을 선택하십시오. 반출 마법사가 열립니다.
4. 반출 마법사에서 다음을 수행하십시오.
   a. **파일 시스템**을 선택하고 **다음**을 누르십시오.
   b. 전개할 다음 자원을 선택하십시오.
      - Web Content*FashionFlow_name*\UserArea\AccountSection\RegistrationSubsection\UserRegistrationAddForm.jsp
      - Web Content*FashionFlow_name*\UserArea\AccountSection\RegistrationSubsection\UserRegistrationAddFormInclude.jsp
      - Web Content*FashionFlow_name*\UserArea\AccountSection\RegistrationSubsection\UserRegistrationUpdateForm.jsp
      - Web Content*FashionFlow_name*\UserArea\AccountSection\RegistrationSubsection\UserRegistrationUpdateFormInclude.jsp
      - Web Content\WEB-INF\lib\jstl.jar
      - Web Content\WEB-INF\lib\standard.jar
      - Web Content\WEB-INF\classes*FashionFlow_name*\Housing.properties
   c. **선택한 파일의 디렉토리 구조 작성**을 선택하십시오.
   d. 디렉토리 필드에 자원을 놓을 임시 디렉토리를 입력하십시오. 예를 들어, `C:\ExportTemp4`를 입력하십시오.
   e. **완료**를 누르십시오.

대상 WebSphere Commerce Server로 자원 전송

이 단계에서는 대상 WebSphere Commerce Server에서 임시 디렉토리를 작성한 후 이 디렉토리에 주택 조사 자원을 복사합니다. 그 다음 단계에서는 여러 유형의 코드를 WebSphere Commerce 응용프로그램 내의 적정 위치에 배치합니다.

개발 시스템에서 대상 WebSphere Commerce Server로 파일을 복사하려면 다음을 수행하십시오.

1. 대상 WebSphere Commerce Server에서 임시 디렉토리 *drive*:\ImportTemp4를 작성하십시오.
2. 컴퓨터 사이에 파일을 복사할 방법을 판별하십시오. 대상 WebSphere Commerce Server의 드라이브를 개발 시스템에 맵핑하거나 FTP 응용프로그램(구성된 경우)을 사용하여 이를 수행할 수 있습니다.
3. 개발 시스템에서 *drive*:\ExportTemp4의 컨텐츠를 대상 WebSphere Commerce Server의 *drive*:\ImportTemp4로 복사하십시오.

대상 WebSphere Commerce Server 중지

전개 단계를 시작하기 전에, 명령행에서 stopServer 명령을 실행하여 대상 WebSphere Commerce Server를 중지해야 합니다. 이 명령에 대한 자세한 정보는 *WebSphere Commerce Studio 설치 안내서*를 참조하십시오.

대상 WebSphere Commerce Server에서 데이터베이스 갱신

XHOUSING 테이블 작성

DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 테이블을 작성하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령행 도구 > 명령 센터**)를 열고 **스크립트** 탭을 누르십시오.
2. 스크립트 창에서 다음을 입력하십시오.

```
connect to developmentDB user dbuser using dbpassword;
create table XHOUSING (MEMBERID bigint not null,
    HOUSINGTYPE integer, LOCATION integer,
   constraint p_xhousing primary key (MEMBERID),
```

```
   constraint f_xhousing foreign key (MEMBERID)
    references USERS (USERS_ID) on delete cascade);
insert into XHOUSING (MEMBERID) (select USERS_ID from USERS);
```

여기서,

- *developmentDB*는 개발 데이터베이스의 이름입니다.
- *dbuser*는 데이터베이스 사용자입니다.
- *dbpassword*는 데이터베이스 사용자의 암호입니다.

실행 아이콘을 누르십시오.

SQL문이 성공적으로 완료되었음을 표시하는 메시지가 표시되어야 합니다.

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 테이블을 작성하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 SQL문을 입력하십시오.

   ```
   create table XHOUSING (MEMBERID number not null,
       HOUSINGTYPE integer, LOCATION integer,
      constraint p_xhousing primary key (MEMBERID),
      constraint f_xhousing foreign key (MEMBERID)
       references USERS (USERS_ID) on delete cascade);
   insert into XHOUSING (MEMBERID) (select USERS_ID from USERS);
   ```

   Enter를 눌러 SQL문을 실행하십시오. XHOUSING 테이블이 작성됩니다.
6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

   ```
   commit;
   ```

   Enter를 눌러 SQL문을 실행하십시오.

태스크 명령 등록

명령 레지스트리를 수정하려면 다음을 수행하십시오.

1. DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 MyPostUserRegistrationAddCmdImpl을 등록하십시오.
   a. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령 센터**)를 여십시오.
   b. **도구** 메뉴에서 **도구 설정**을 선택하십시오.
   c. **명령문 종료 문자 사용** 선택란을 선택한 후 지정된 문자가 세미콜론(**;**)인지 확인하십시오.
   d. 스크립트 탭을 선택하고 스크립트 창에 다음 정보를 입력하여 URLREG 테이블에 필수 항목을 작성하십시오.

```
connect to developmentDB user dbuser using dbpassword;
insert into CMDREG values (FashionFlow_storeent_Id,
'com.ibm.commerce.usermanagement.commands.PostUserRegistrationAddCmd'
'Description',
'com.ibm.commerce.sample.commands.MyPostUserRegistrationAddCmdImpl',
null, null, 'Local');
```

   여기서,
   - *developmentDB*는 개발 데이터베이스의 이름입니다.
   - *dbuser*는 데이터베이스 사용자입니다.
   - *dbpassword*는 데이터베이스 사용자의 암호입니다.
   - *FashionFlow_storeent_Id*는 상점의 고유한 상점 엔티티 식별자입니다.

   실행 아이콘을 누르십시오.

2. Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 MyPostUserRegistrationAddCmdImpl을 등록하십시오.
   a. Oracle SQL Plus 명령창을 여십시오(**시작 > 프로그램 > Oracle > Application Development > SQL Plus**).
   b. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
   c. **암호** 필드에 Oracle 암호를 입력하십시오.
   d. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
   e. SQL Plus 창에서 다음 SQL문을 입력하십시오.

```
insert into CMDREG values (FashionFlow_storeent_Id,
'com.ibm.commerce.usermanagement.commands.PostUserRegistrationAddCmd'
'Description',
'com.ibm.commerce.sample.commands.MyPostUserRegistrationAddCmdImpl',
null, null, 'Local');
```

SQL문을 실행하려면 Enter를 누르십시오.

f. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

Enter를 눌러 SQL문을 실행하십시오.

대상 WebSphere Commerce Server에서 상점 자원 갱신

이 단계에서는 다음과 같이 상점을 수정한 상점 자원으로 갱신합니다.

1. *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear\ Stores.war 디렉토리를 백업하십시오. 여기서, *cellName*은 대부분 시스템의 호스트 이름입니다.
2. *drive*:\ImportTemp4\Stores\Web Content 디렉토리로 이동하십시오.
3. *FashionFlow_name* 및 WEB-INF 폴더를 다음 디렉토리에 복사하십시오. *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear\Stores.war

여기서, *instanceName*은 WebSphere Commerce 인스턴스 이름입니다.

대상 WebSphere Commerce Server에서 명령 JAR 파일 갱신

이 단계에서는 다음과 같이 새 명령 JAR 파일을 사용하기 위해 대상 WebSphere Commerce Server를 갱신합니다.

1. 다음과 같이 기존 JAR 파일의 백업 사본을 작성해야 합니다.
   a. *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 디렉토리로 이동하십시오.
   b. WebSphereCommerceServerExtensionsLogic.jar 파일의 사본을 작성하고 백업 위치에 저장하십시오.

2. *drive*:\ImportTemp4 디렉토리의
   새 WebSphereCommerceServerExtensionsLogic.jar 파일을
   WAS_installdir\installedApps*cellName*\WC_*instanceName*.ear 디렉토리에 복사하십시오.

여기서, *instanceName*은 WebSphere Commerce 인스턴스 이름입니다.

대상 WebSphere Commerce Server에서 EJB JAR 파일 갱신

이 단계에서는 다음과 같이 새 EJB JAR 파일을 사용하도록 대상 WebSphere Commerce Server를 갱신합니다.

1. 다음과 같이 기존 JAR 파일의 백업 사본을 작성해야 합니다.
   a. *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 디렉토리로 이동하십시오.
   b. Member-MemberManagementData.jar 파일의 사본을 작성하고 백업 위치에 저장하십시오.
2. *drive*:\ImportTemp4 디렉토리의 새 Member-MemberManagementData.jar 파일을 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 디렉토리에 복사하십시오.
3. 다음과 같이 EJB 전개 설명자 정보를 수정해야 합니다.
   a. WebSphere Application Server 셀의 전개 저장소(META-INF 디렉토리)를 찾으십시오. 일반적으로 그 양식은
      WAS_installdir\config\cells*cellName* \applications\WC_*instance_name*.ear\deployments\ WC_*instance_name**EJBModuleName*.jar\META-INF와 같습니다.
      예는 다음과 같습니다.
      D:\WebSphere\AppServer\config\cells\myCell\applications\
      WC_demo.ear\deployments\WC_demo\
      Member-MemberManagementData.jar\META-INF
      여기서,
      - myCell은 WebSphere Application Server 셀 이름이고,
      - demo는 WebSphere Commerce 인스턴스 이름이며

- Member-MemberManagementData는 사용자 정의 EJB 모듈 이름입니다.

b. 디렉토리에는 다음 파일이 있습니다.

- ejb-jar.xml
- ibm-ejb-access-bean.xmi
- ibm-ejb-jar-bnd.xmi
- ibm-ejb-jar-ext.xmi
- MANIFEST.MF

이러한 모든 파일의 백업

c. 도구를 사용하여 새 Member-MemberManagementData.jar 파일을 열고 그 내용을 보십시오.

d. 이 Member-MemberManagementData.jar 파일에서 meta-inf 디렉토리의 내용을 3a단계의 디렉토리로 추출하십시오.

4. 명령행에서 WebSphere Application Server startServer 명령을 사용하여 WebSphere Commerce 인스턴스를 다시 시작하십시오.

대상 WebSphere Commerce Server에서 주택 조사 로직 검증

이 단계에서는 다음을 수행하여 주택 조사 로직이 대상 WebSphere Commerce Server로 전개되었는지를 검증합니다.

1. 웹 브라우저를 열고 URL을 입력하여 FashionFlow 견본 상점을 기반으로 하는 사용자 상점을 시작하십시오.
2. 상점의 홈 페이지가 열리면 **등록**을 누르십시오.
3. **등록**을 다시 눌러 새 사용자를 작성하십시오.
4. 다음 화면에 표시되는 것과 같이 수정된 등록 페이지(주택 조사가 포함됨)가 표시됩니다.

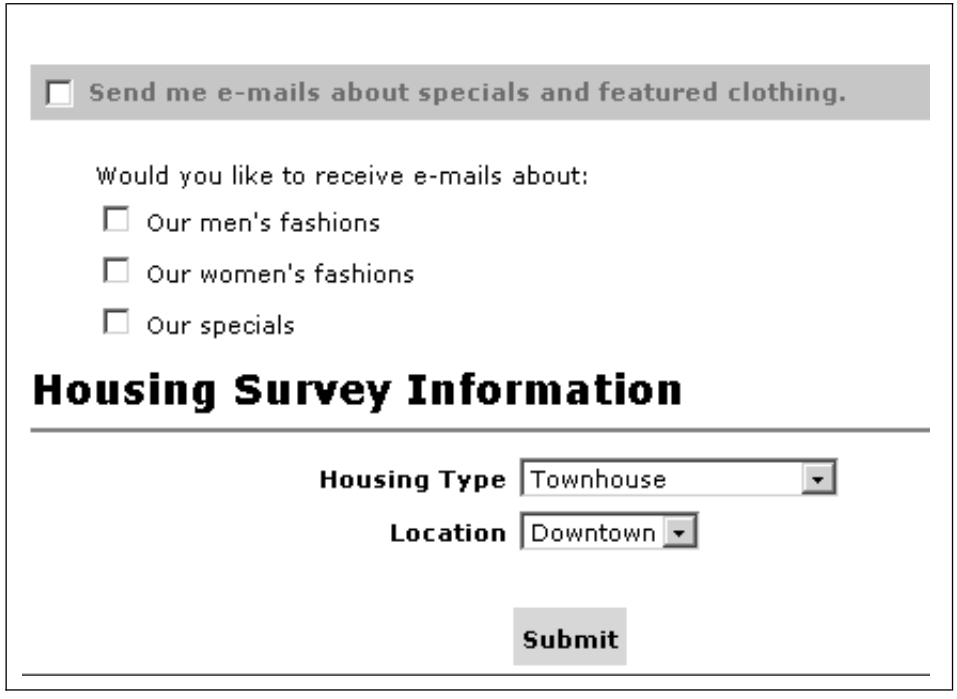


그림 55.

5. 새 사용자에 대한 적절한 정보를 입력하고 **제출**을 누르십시오.
6. 정보가 제출되면 **개인 정보 변경**을 눌러 주택 정보를 캡처했는지 검증하십시오. 다음과 같이 조사 응답의 요약을 표시하는 화면이 표시됩니다.

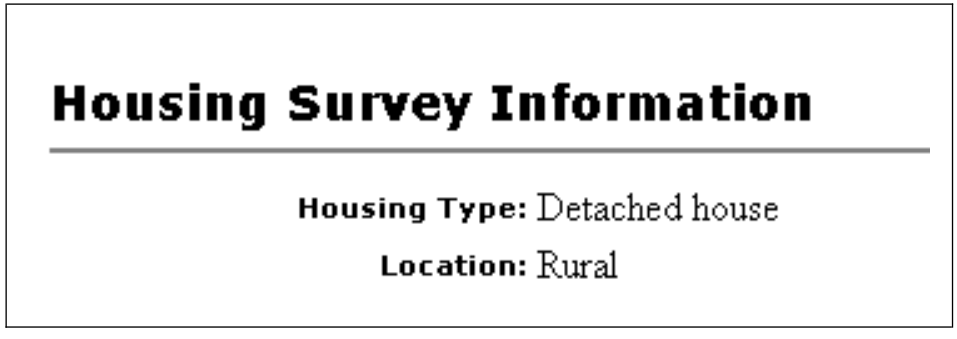


그림 56.

제 5 부 부록

부록 A. WebSphere Commerce Studio에 WebSphere Commerce 구성요소 추적 구성

이 부록에서는 WebSphere Studio Application Developer 개발 환경에서 여러 WebSphere Commerce 구성요소 실행 시 이에 대한 추적을 사용하는 방법에 대해 설명합니다. 구성요소 추적을 사용하려면 다음을 수행하십시오.

1. 필요한 경우, WebSphere Commerce 개발 환경(**시작 > 프로그램 > IBM WebSphere Commerce Studio > WebSphere Commerce 개발 환경**)을 여십시오.
2. 서버 perspective로 전환하십시오.
3. 서버의 보기에서 **WebSphereCommerceServer**를 마우스 오른쪽 버튼을 누르고 **중지**(서버가 현재 실행 중인 경우)를 선택하십시오.
4. 서버 구성 보기에서 **Server Configurations** 폴더를 펼치십시오.
5. **WebSphereCommerceServer**를 두 번 누르십시오.
   WebSphereCommerceServer 편집기가 열립니다.
6. 추적 탭을 선택하십시오.
7. **추적 사용**을 선택하십시오.
8. **문자열 추적** 텍스크 상자에서 추적이 사용될 구성요소를 지정하십시오. WebSphere JRas 확장자 추적 로그 작성 식별자 값을 다음에 "=all=enabled"가 오게 하여 사용하십시오. 이러한 값의 모든 목록은 *WebSphere Commerce 관리 안내서*의 "구성"을 참조하십시오. 복수 구성요소는 콜론(:)으로 분리해야 합니다. 예를 들어, SERVER 및 RAS 구성요소 모두에 추적을 사용하려면 다음과 같이 추적 문자열을 지정하십시오.

   ```
   com.ibm.websphere.commerce.WC_SERVER=all=enabled:
      com.ibm.websphere.commerce.WC_RAS=all=enabled
   ```

   행 바꾸기는 표시 용도일 뿐임에 유의하십시오.
9. 변경사항을 저장하십시오(ctrl+s).

출력 파일

출력 로그 파일의 이름은 기본값으로 activity.log입니다. 이 파일은 *workspace_dir*\.metadata\.plugin\com.ibm.etools.server.core\tmp0\logs 디렉토리에 있습니다.

activity.log 파일은 2진 파일이므로 이 파일은 로그 분석기를 사용하여 읽습니다. 구성요소 추적을 사용하면 WebSphere JRas가 추적 항목과 함께 로그 항목도 일반 텍스트 포맷으로 추적 출력 파일에 작성합니다.

WebSphere Commerce Studio의 로그 분석기 도구 구성에 대한 정보는, *WebSphere Commerce Studio 설치 안내서*를 참조하십시오.

또한 WebSphere Studio Application Developer의 콘솔 보기에 메시지가 표시됩니다.

부록 B. 추가 정보

WebSphere Commerce Studio 시스템 및 그 구성요소에 대한 자세한 정보는 여러 포맷의 다양한 소스에서 사용 가능합니다. 다음 절은 사용 가능한 정보의 내용과 액세스 방법을 표시합니다.

WebSphere Commerce Studio 정보

다음은 WebSphere Commerce Studio 정보의 소스입니다.

- 『WebSphere Commerce Studio 온라인 도움말』
- 442 페이지의 『WebSphere Commerce 웹 사이트』
- 442 페이지의 『WebSphere Developer Domain』
- 442 페이지의 『IBM 레드북』

WebSphere Commerce Studio 온라인 도움말

WebSphere Commerce Studio 온라인 정보는 WebSphere Commerce Studio에 상점을 작성 및 공개하는 정보의 기본 소스입니다.

WebSphere Commerce Studio 온라인 도움말을 보려면 다음을 수행하십시오.

1. **시작 → 프로그램 → IBM WebSphere Commerce Studio → WebSphere Commerce 개발 환경**을 선택하여 WebSphere Commerce Studio를 시작하십시오.
2. **도움말** 메뉴에서 **도움말 목차**를 선택하십시오.

주: 『WebSphere Commerce Studio 온라인 도움말』의 복수 플랫폼에 대한 지시사항을 참조하는 경우, WebSphere Commerce Studio를 위한 지시사항을 따르고 있는지 확인하십시오. 도움말 페이지에 복수 플랫폼에 대한 정보가 있는 경우, WebSphere Commerce Studio 특정 정보는 다음 아이콘으로 표시됩니

다.

WebSphere Commerce Studio 특정 정보가 사용 불가능한 경우, Windows 특정 지시사항을 따르고 있는지 확인하십시오. 도움말 페이지에 복수 플랫폼에 대한 지시사항이 있으면 Windows용 지시사항은 다음 아이콘으로 표시됩니다.

WebSphere Commerce 웹 사이트

WebSphere Commerce Studio 제품 정보는 WebSphere Commerce 웹 사이트에서 사용 가능합니다. 보다 자세한 제품 정보는 다음 URL을 참조하십시오.

`http://www.ibm.com/software/webservers/commerce/library/`

WebSphere Developer Domain

WebSphere Commerce Studio 및 WebSphere Commerce의 추가 정보는 다음 WebSphere Developer Domain의 WebSphere Commerce Zone에서 사용 가능합니다.

`http://www.ibm.com/websphere/developer/zones/commerce/`

IBM 레드북

WebSphere Commerce Studio 및 WebSphere Commerce 정보는 다음의 IBM 레드북™ 웹 사이트에서 사용 가능합니다.

`http://www.ibm.com/redbooks`

WebSphere Studio Application Developer 정보

다음은 WebSphere Studio Application Developer에 대한 정보의 소스입니다.

- 『WebSphere Studio Application Developer 온라인 도움말』
- 『WebSphere Studio Application Developer 웹 사이트』
- 『WebSphere Developer Domain』
- 444 페이지의 『IBM 레드북』

WebSphere Studio Application Developer 온라인 도움말

WebSphere Studio Application Developer 온라인 도움말은 WebSphere Studio Application Developer에서 태스크를 수행하는 방법에 대한 정보의 기본 소스입니다.

WebSphere Studio Application Developer 온라인 도움말을 보려면 다음을 수행하십시오.

1. **시작 → 프로그램 → IBM WebSphere Studio → Application Developer 5.0**을 선택하여 WebSphere Commerce Studio를 시작하십시오.
2. **도움말** 메뉴에서 **도움말 목차**를 선택하십시오.

WebSphere Studio Application Developer 웹 사이트

WebSphere Studio Application Developer 제품 정보는 다음의 WebSphere Studio Application Developer 웹 사이트에서 사용 가능합니다.

`http://www.ibm.com/software/ad/studioappdev/library/`

WebSphere Developer Domain

WebSphere Studio Application Developer에 대한 추가 정보는 다음 WebSphere Developer Domain의 WebSphere Studio Zone에 있는 WebSphere Studio Application Developer 페이지에서 사용 가능합니다.

`http://www.ibm.com/websphere/developer/zones/studio/appdev/`

IBM 레드북

WebSphere Studio Application Developer 정보는 다음의 IBM 레드북 웹 사이트에서 사용 가능합니다.

`http://www.ibm.com/redbooks`

주의사항

이 정보는 미국에서 제공되는 제품 및 서비스용으로 작성된 것입니다. IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이센스까지 부여하는 것은 아닙니다. 라이센스에 대한 의문사항은 다음으로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2바이트(DBCS) 정보와 관련된 라이센스 문의는 한국 IBM 고객만족센터에 문의하거나 다음의 주소로 서면 문의하시기 바랍니다.

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

다음의 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다.

IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여(단, 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증없

이 이 책을 현상태대로 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 암시적인 보증의 면책사항을 허용하지 않으므로 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며 이 변경사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통고없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 비IBM의 웹 사이트는 단지 편의상 제공된 것으로 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 갖게 됩니다.

IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

(1) 독립적으로 작성된 프로그램 및 기타 프로그램(이 프로그램 포함) 간의 정보 교환 (2) 교환된 정보의 상호 이용을 목적으로 정보를 원하는 프로그램 라이센스 사용자는 다음 주소로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터

이러한 정보는 해당 조항 및 조건에 따라(예를 들면, 라이센스 지불 포함) 사용할 수 있습니다.

이 정보에 기술된 라이센스 프로그램 및 사용 가능한 모든 라이센스 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

이 책에 포함된 성능 데이터는 제한된 환경에서 산출된 것입니다. 그러므로 다른 운영 환경에서의 결과와 상당히 다를 수도 있습니다. 일부 측정은 개발 단계의 시스템에서 이루어진 것이므로 그 측정치가 일반적으로 사용 가능한 시스템에서도 동

일하다고 보장할 수 없습니다. 또한 일부 측정치는 보외법을 통해 이루어졌으므로 실제 결과는 다를 수도 있습니다. 이 책의 사용자는 자신의 고유 환경에 적합한 데이터를 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 제품들을 테스트하지 않았으므로, 비IBM 제품과 관련된 성능의 정확성, 호환성 또는 배상 청구에 대해서는 확신할 수 없습니다. 타사 제품의 기능에 대한 질문은 해당 제품의 제공업체로 해야 합니다.

IBM의 진행 예정이나 의도에 대한 모든 진술은 통지 없이 변경되거나 취소될 수 있으며 목적과 목표만을 나타낼 뿐입니다.

표시된 모든 IBM 가격은 IBM이 제안하는 소매가이며 최신 가격으로 통지 없이 변경될 수 있습니다. 소매업체 가격들은 다양할 수 있습니다.

이 정보는 계획 목적으로만 사용됩니다. 여기의 정보는 설명된 제품이 사용 가능해지기 전에 변경될 수 있습니다.

이 책에서는 일상적인 비즈니스 운영에 사용되는 데이터 및 보고서 예가 수록되어 있습니다. 가능한 한 완벽하게 표시하기 위해 예제에는 개인, 회사, 브랜드, 상품의 이름이 포함됩니다. 그러한 이름들은 모두 가공의 이름으로서 실제 회사에서 사용하는 이름 및 주소와 유사하다 하더라고 전적으로 우연입니다.

저작권

이러한 견본 프로그램의 각 사본이나 일부 또는 여기에서 파생된 저작물에는 다음과 같이 저작권 주의사항이 포함되어야 합니다.

이 정보를 소프트카피로 보는 경우, 사진과 컬러 그림은 나타나지 않을 수 있습니다.

상표

IBM 로고 및 다음 용어는 미국 또는 기타 국가에서 사용되는 IBM Corporation의 상표 또는 등록상표입니다.

400
AIX
AS/400
DB2
DB2 Universal Database
@server
IBM
iSeries
WebSphere

Windows는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 등록상표입니다.

Java 및 모든 Java 기반 상표와 로고는 미국 또는 기타 국가에서 사용되는 Sun Microsystems, Inc.의 상표 또는 등록상표입니다.

기타 회사, 제품 및 서비스 이름은 해당 회사의 상표 또는 서비스표입니다.

색인

IBM 한글 지원에 관한 설문

IBM WebSphere Commerce
프로그래밍 안내서 및 학습서
버전 5.5

FAX : (02) 3787-0123

보내 주시는 의견은 더 나은 고객 지원 체제를 위한 귀중한 자료가 됩니다. 독자 여러분의 좋은 의견을 기다립니다.

성 명		직위/담당업무	
회 사 명		부 서 명	
주 소			
전화번호		팩스번호	
전자우편 주소			
사용중인 시스템	o 중대형 서버 o UNIX 서버 o PC 및 PC 서버		

1. IBM에서 제공하는 한글 책자와 영문 책자 중 어느 것을 더 좋아하십니까?
   그 이유는 무엇입니까?
   ☐ 한글 책자 ☐ 영문 책자
   (이유:)

2. 본 책자와 해당 소프트웨어에서 사용된 한글 용어에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

3. 본 책자와 해당 소프트웨어에서 번역 품질에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

4. 본 책자의 인쇄 상태에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

5. 한글 소프트웨어 및 책자가 지원되는 분야에 대해 귀하는 어떻게 생각하십니까?
   ☐ 한글 책자를 늘려야 함 ☐ 현재 수준으로 만족
   ☐ 그다지 필요성을 느끼지 않음

6. IBM은 인쇄물 형식(hardcopy)과 화면 형식(softcopy)의 두 종류로 책자를 제공합니다.
   어느 형식을 더 좋아하십니까?
   ☐ 인쇄물 형식(hardcopy) ☐ 화면 형식(softcopy) ☐ 둘 다

☜ IBM 한글 지원 서비스에 대해 기타 제안사항이 있으시면 적어주십시오.

_____

_____

_____

_____

☺ 설문에 답해 주셔서 감사합니다.
귀하의 의견은 저희에게 매우 소중한 것이며, 고객 여러분들께 보다 좋은 제품을 제공해 드리기 위해 최선을 다하겠습니다.