ESB의 새로운 가능성
베스트 프랙티스를 중심으로

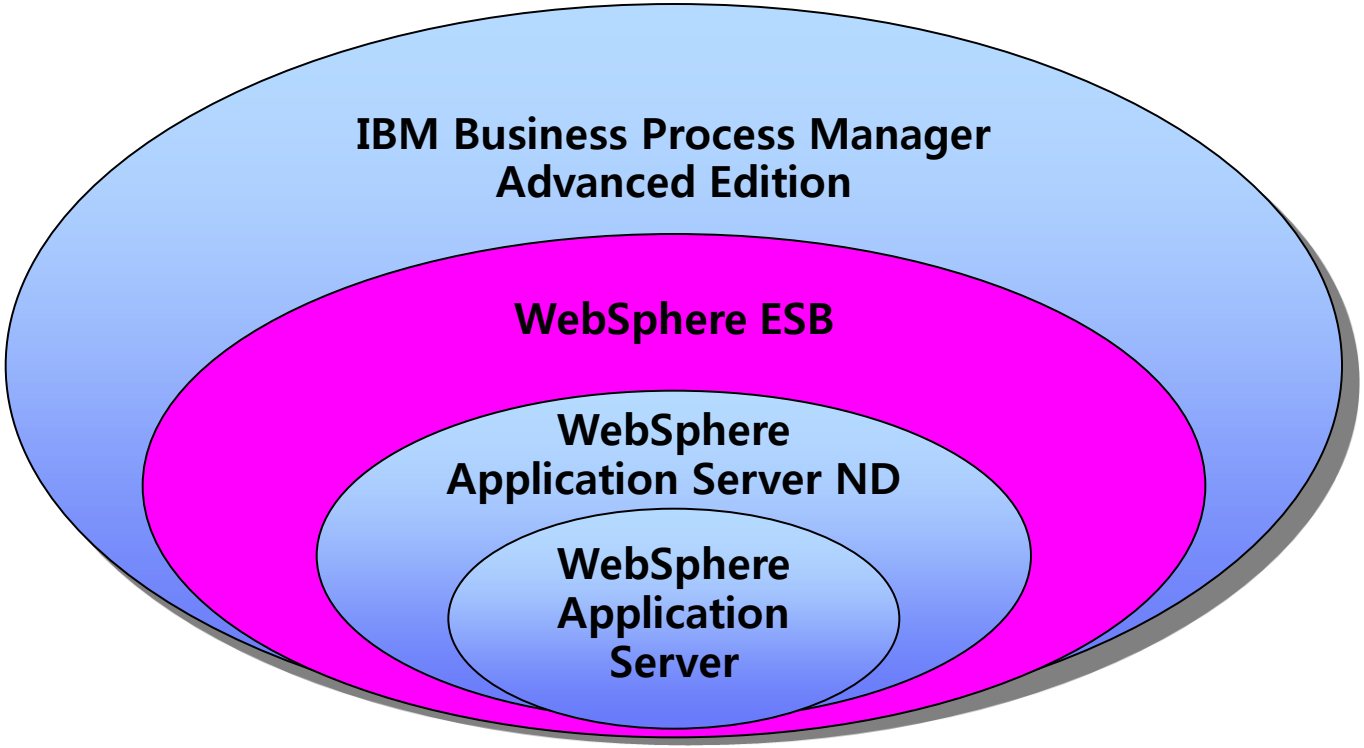
홍창배 차장
한국IBM 소프트웨어 그룹 웹스피어 사업부

Impact Korea 2011
Changing the Way Business and IT Leaders Work

발표 사항

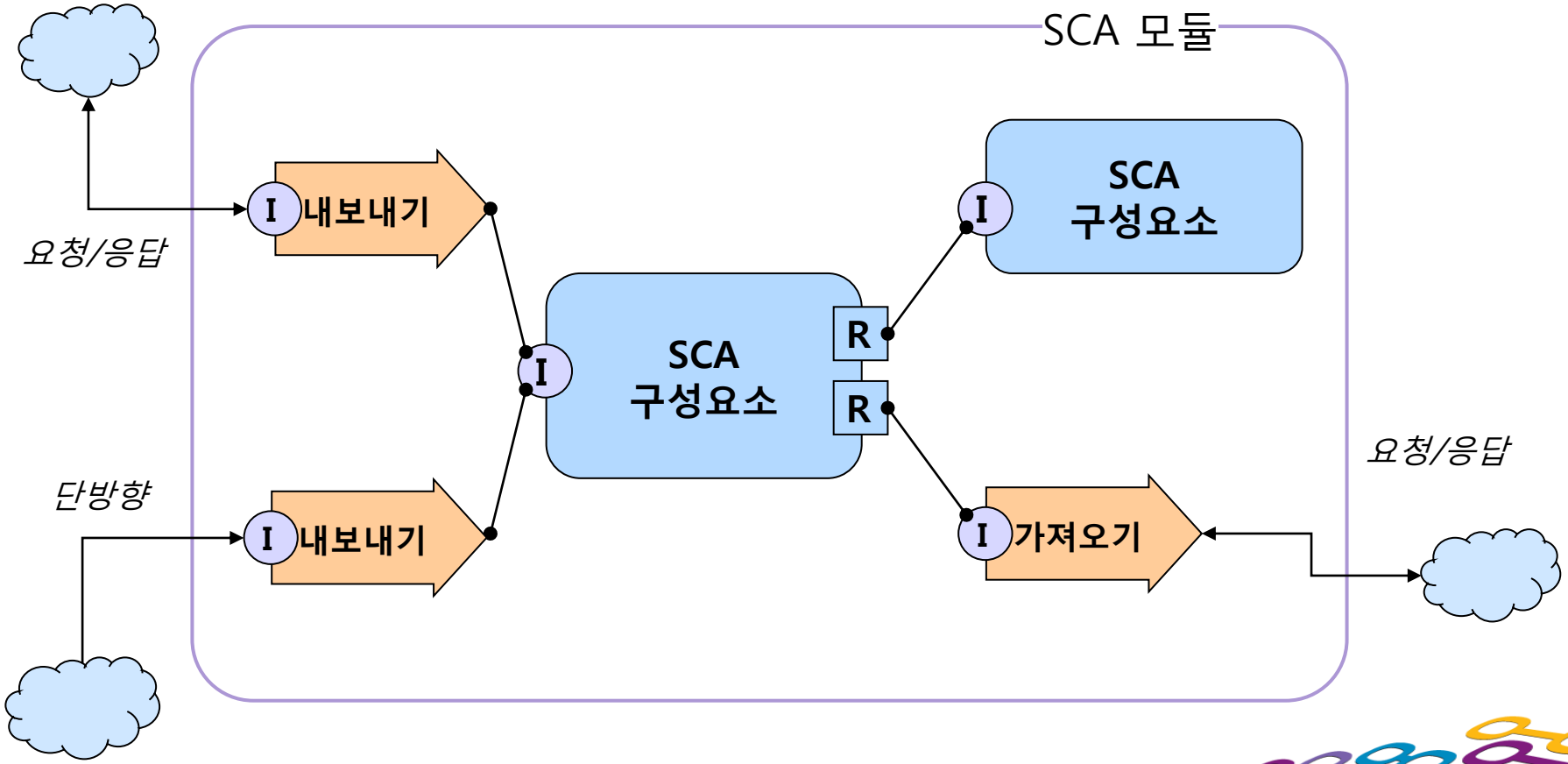
- 간략한 제품 개요
- 베스트 프랙티스
  - 기획, 설계 및 아키텍처
  - 프로세스
  - 개발
  - 성능
- 참고자료 및 추가 정보

WebSphere ESB 핵심: 제품군이 가져다 주는 가치



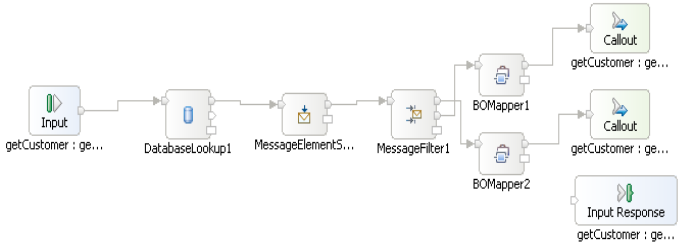
- 구성(choreography) 및 솔루션 관점
- 중개
- 클러스터링
- 애플리케이션 서버

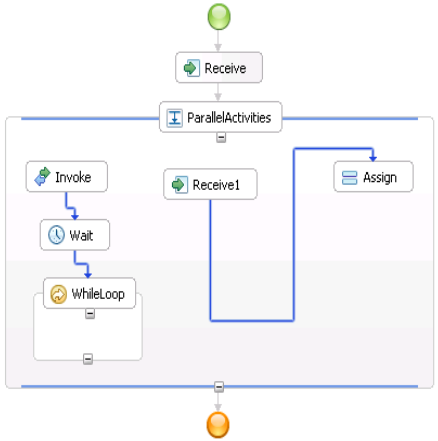
서비스 컴포넌트 아키텍처



모듈 및 컴포넌트 유형

- WebSphere ESB는 다음과 같은 컴포넌트를 가진 중개 모듈(Mediation Module)만을 지원합니다
  - 중개 플로우
  - Java
- IBM BPM Advanced Edition은 다음과 같은 (통합) 모듈을 지원하기도 합니다
  - BPEL (프로세스)
  - Human Task
  - Rule Group (비즈니스 룰)
  - 기타





베스트 프랙티스: 기획, 설계 및 아키텍처

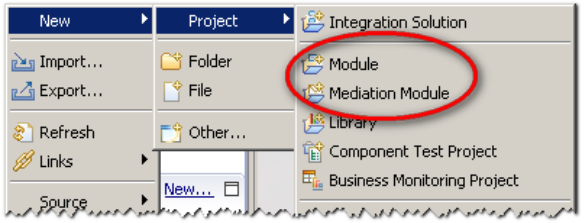
- 불러오기 및 내보내기 노드를 위해 지원되는 연결 바인딩 및 어댑터
  - Web Services (SOAP/HTTP, SOAP/JMS)
  - Messaging (JMS, MQ, JMS MQ, generic JMS((3rd Party) 제공업체)
  - J2EE Applications (EJB 바인딩)
  - Native (SCA 크로스 모듈)
  - HTTP Access (JSON, XML)
  - IBM CICS ECI Resource Adapter
  - IBM IMS Connector for Java
  - IBM WebSphere Adapter for Email
  - IBM WebSphere Adapter for FTP
  - IBM WebSphere Adapter for Flat Files
  - IBM WebSphere Adapter for JDBC
  - IBM WebSphere Adapter for Lotus Domino
  - IBM WebSphere Adapter for Enterprise Content Management
  - IBM WebSphere Adapter for JD Edwards EnterpriseOne® *
  - IBM WebSphere Adapter for Oracle® E-Business Suite *
  - IBM WebSphere Adapter for PeopleSoft *
  - IBM WebSphere Adapter for SAP® Software *
  - IBM WebSphere Adapter for Siebel® Business Applications

* 런타임 라이센스 필요

베스트 프랙티스: 기획, 설계 및 아키텍처

- 간략한 제품 개요
- 베스트 프랙티스
  - 기획, 설계 및 아키텍처
  - 프로세스
  - 개발
  - 실행
- 참고자료 및 추가 정보

모듈 유형 최적화 (1/2)



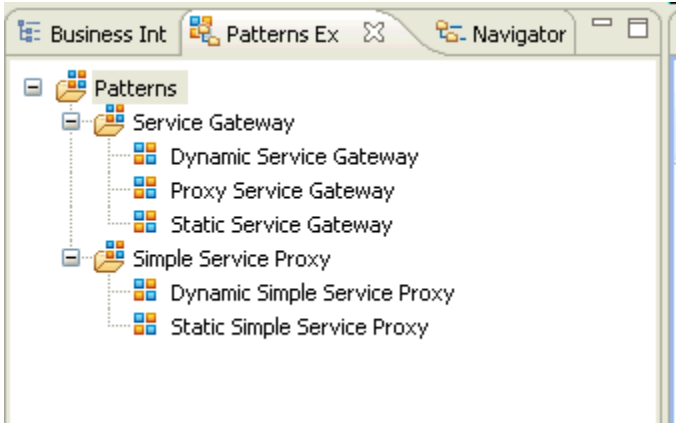
- 디자인 고려사항 –
  중개 로직 vs 프로세스 로직
- 중개 플로우(Mediation Flows)를 위한 중개 모듈(WebSphere ESB & BPM Advanced) 사용 – 통합/중개 로직
  - Short-running, minimal Choreography
  - 헤더 조작 지원
- BPEL을 위한 (통합) 모듈 사용 – 비즈니스/프로세스 로직:
  - Long-running, powerful-choreography 제공
- 더 상세한 정보는 아래 사이트에서 확인 가능
  - http://www.ibm.com/developerworks/websphere/library/techarticles/0803_fasbinder2/0803_fasbinder2.html

모듈 유형 최적화 (2/2)

- V6.2 경우, 중개 플로우 컴포넌트들은 (통합) 모듈에서 사용 가능
- 일부 경우에는, 같은 모듈에 중개 로직과 비즈니스 로직을 혼합해야 할 필요 있음 (성능 고려 경우)
- 일부 제한적 경우에, BPEL을 복잡한 통합을 위한 “비주얼 프로그래밍 언어”로 취급하는 것이 타당할 수 있음
- 하지만 설계 시 중개 로직과 프로세스 로직 간의 구분을 명확히 하도록 유의

통합 패턴 활용

- 가능한 경우, 제공되는 통합 패턴을 활용
  - 중개 개발을 가속화함
  - 생성된 모듈은 Best-Practice 에 근거함.
- 현재 활용 가능한 통합 패턴
  - 단순 정적 서비스 프록시
  - 단순 동적 서비스 프록시
  - 정적 서비스 게이트웨이
  - 단순 동적 게이트웨이
  - 프록시 서비스 게이트웨이
  - 서비스 트랜스레이터(Translator) : 기능팩(Feature Pack)
  - 서비스 셀렉터(Selector) : 기능팩(Feature Pack)

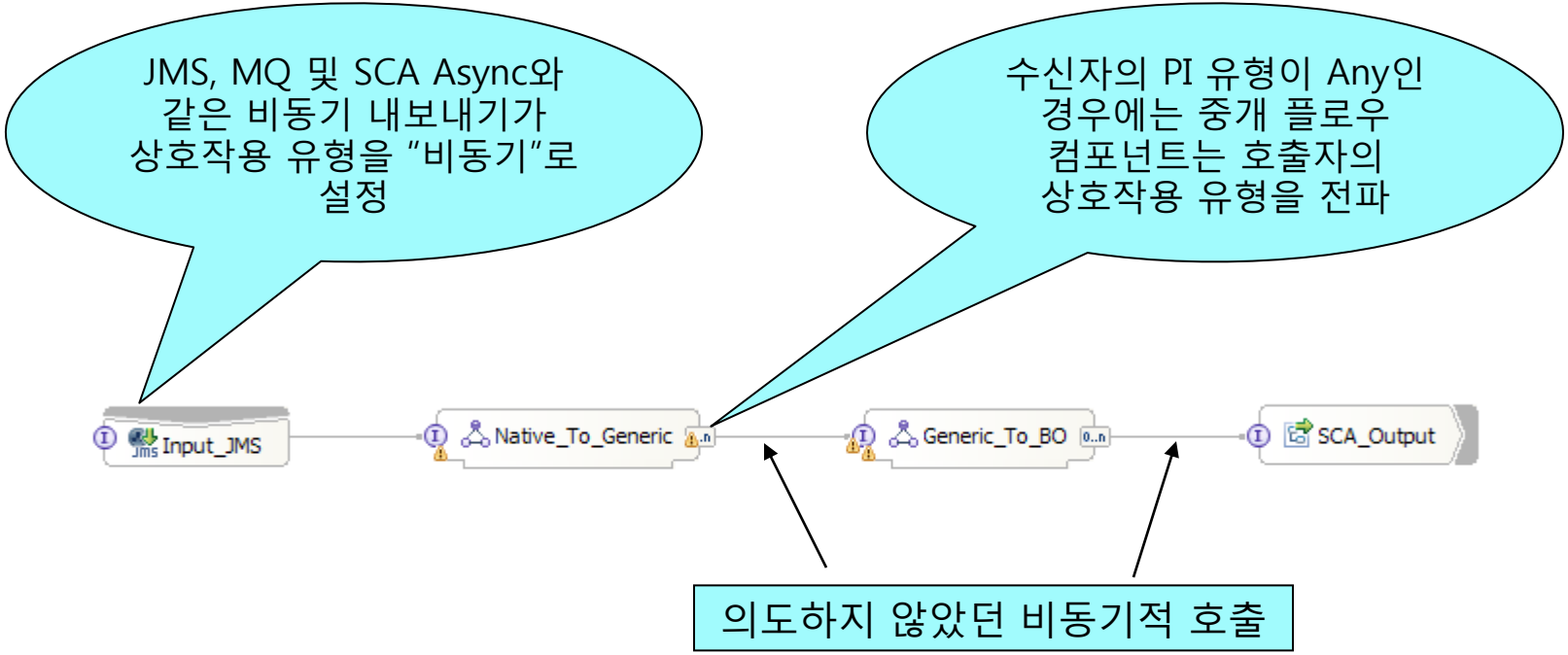


호출(Invocation) 유형

- 통상적으로 바인딩에 의해 정해진 디폴트 "호출" 유형은 일반적으론 바람직하지만, 주의가 필요함.
- (컴포넌트나 모듈 간의) 비동기 호출(Interaction)은 SCA/JMS/MQ 큐(Queue)를 경유해서 지나가며, 이것은 다음과 같은 의미가 있습니다
  - 트랜잭션 범위의 단절
  - 런타임(Runtime) 예외는 롤백(Rollback)을 유발하며 이로 인해 재시도 한도가 넘으면 메시지가 예외 목적지로 전송. (이후 Failed Event Manager를 통해 처리됩니다)
- 호출유형이 비동기임을 예측하는 것은 어려울 수 있음. (성능 최적화의 주요 요소)
- 의심스러운 경우 비동기 호출을 가정하고, 'preferredInteractionStyle'를 사용하는게 바람직함.
- 더 상세한 정보는 아래 사이트에서 확인하시기 바랍니다
  - http://www.ibm.com/developerworks/websphere/library/techarticles/0811_chacko/0811_chacko.html

호출 유형의 이해

- 의도하지 않았던 비동기성의 예
  - 선호 상호작용 유형 = Any



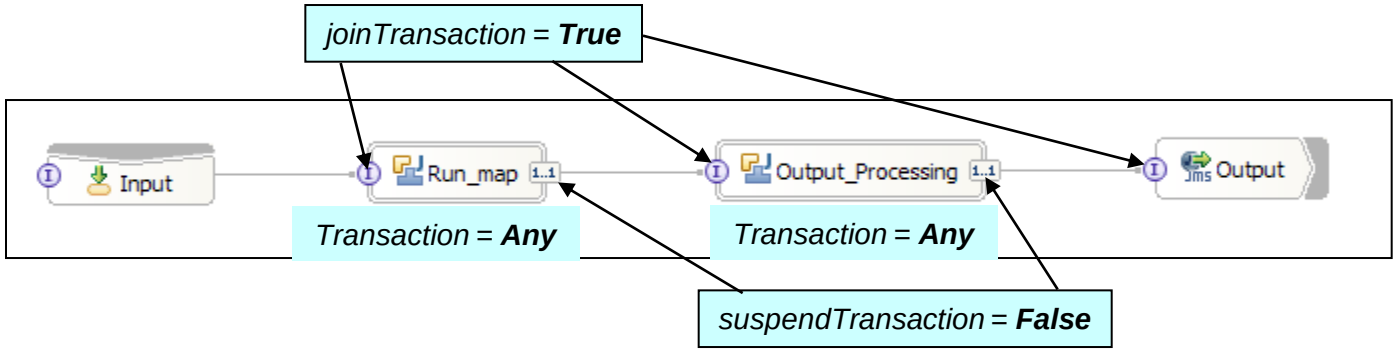
트랜잭션 설계

- 트랜잭션 설계의 중요성
  - 오류 발생시에 일어나는 문제에 영향을 미침
  - Integration Designer의 디폴트 값은 단지 출발점에 불과함.
  - JDBC나 메시징 시스템과 상호작용하고 있다면, 트랜잭션 지원 (transactionality)이 필요할 수 있음.

- 처리 원칙
  - 모든 트랜잭션 확약(Commit)은 오버헤드를 증가시킴
  - 원하는 트랜잭션 동작을 실행하게끔 트랜잭션 확약을 최소화함
  - SCA 트랜잭션 규정자(Qualifier)에 특별히 주의를 기울일 것

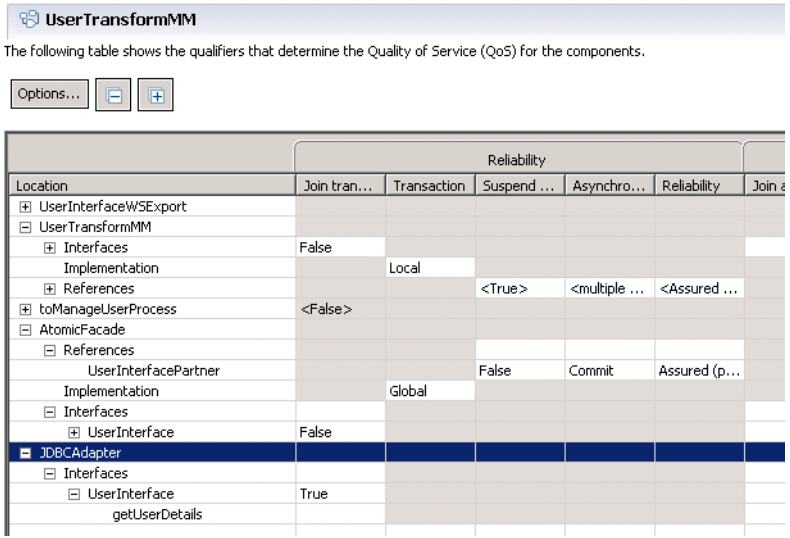
트랜잭션 설계 – SCA Qualifier

- 트랜잭션이 원하는 확약 지점에 이를 때까지 컴포넌트를 통해 전파될 수 있도록 해야 합니다
- 이를 달성하기 위해서는 다음과 같은 원칙을 따라야 합니다
  - suspendTransaction = False
  - Transaction = Any or Global
  - jointTransaction = True



트랜잭션 설계 – SCA Qualifier

- 트랜잭션 Highlighting 및 Qualifiers 편집기를 사용하시기 바랍니다
- 더 상세한 정보는 아래 사이트에서 확인하시기 바랍니다
  - http://soatipsntricks.wordpress.com/2008/07/31/transactionality-in-sca-part-2-refactoring-interfaces/

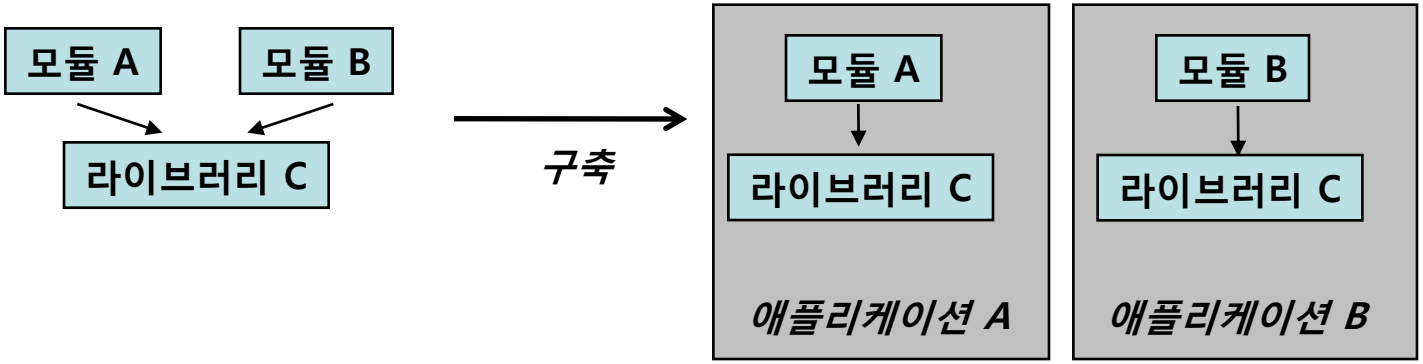


“디폴트” 바인딩 선택

- 종종 바인딩 유형은 상황에 맞게 정해집니다
- 하지만, 결정할 수 있는 여지가 있는 경우에는 다음과 같은 원칙을 따라야 합니다.
  - ESB 내부 또는 ESB 간 통신의 경우에는 SCA default/native 선호: 빠르고, 효율적, 간단함.
  - 동기적 서비스 Export의 경우에는 Web Services (JAX-WS)가 선호: 성숙된 방식이며, SDO 모델에 잘 통합.
  - 비동기적 서비스 Export의 경우에는 JMS가 선호: 애플리케이션 서버 플랫폼과 잘 통합

모듈 분리 방법 고려

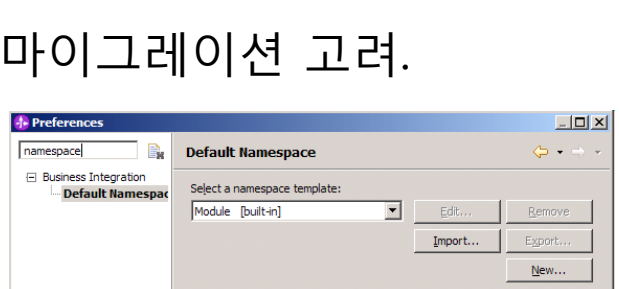
- 각 모듈에 얼마나 많은 내용을 담을 것인가?
  - 많은 수의 모듈은 메모리 소비/배치/장애 복구에 영향을 미칩니다
  - 적은 수의 모듈은 개발 용이성에 영향을 미치며, 일반적으로 모듈 당 1명 이상의 개발자를 두지 않는 것이 바람직합니다
  - Share-by-copy 방식에 의한 라이브러리 공유를 디폴트를 사용하는 경우 대규모 메모리 사용을 초래할 수도 있습니다



  - 참조방식에 의한 공유를 고려하시기 바랍니다 – 모듈 버전 작성에 미치는 영향
    - http://www-01.ibm.com/support/docview.wss?rs=2307&uid=swg21322617

인터페이스 및 비즈니스 오브젝트 고려

- 리팩터링 지원은 임의의 설계 변경을 허용하지 않으므로, 가능한 사전에 정확하게 설계하는 것이 필요.
- 특히, 버전화된 long-running 프로세스 및 프로세스 마이그레이션 고려.
- 명명 규칙 적용
- 제약사항(Constraints) 고려
- 사전정의된 결함 추가 고려
- 네임스페이스 사용: 프로젝트 전체 정책 정의
- 디폴트 네임스페이스 정책을 구성할 것



동적인 중개 제어를 위한 중개 정책 사용

- 중개 정책
  - 가변성 중개 포인트를 외부화함으로써 중개 기능에 대한 제어 가능
- 혜택
  - 중개 구성의 복잡성을 최소화
  - 더 유연한 동적인 중개 제어 가능
- 지원되는 기능
  - 동적 속성
  - WSRR에 저장되고 통제되는 중개 정책
  - Policy Resolution 중개 프리미티브(Primitive)

베스트 프랙티스: 프로세스

- 간략한 제품 개요
- 베스트 프랙티스
  - 기획, 설계 및 아키텍처
  - 프로세스
  - 개발
  - 성능
- 참고자료 및 추가 정보

* 런타임 라이센스 필요

로깅 전략 고려

- 하나의 전략 필요!
- 왜 로깅이 필요할까요? – 진단을 위해? 감사를 위해? 아니면 모니터링을 위해?
- 옵션에는 다음 사항이 포함됩니다
  - Message Logger (Mediations) – java.util.logging 클래스 구현을 통한 로그
  - Trace Primitive (V7 기준) – 운영환경에서는 바람직하지 않음
  - JDBC 또는 Flat File Adapter (개별 모듈에서 사용)
  - 로깅용 Basic Visual Snippets
  - 맞춤형 프리미티브(Custom Primitives)
  - 컴포넌트 간 추적 기능(Cross-Component Trace Facility)
- 통상적인 로깅 로직을 위해서는 서브플로우(Subflow)나 개별 모듈을 사용하시기 바랍니다
- 필요하지 않을 때에는 트레이스 기능을 꺼놓으시기 바랍니다

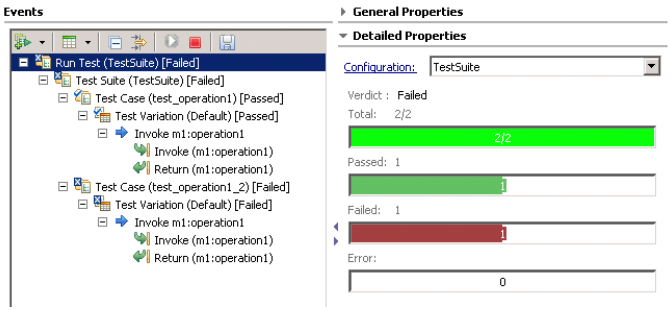
로깅 베스트 프랙티스

- 세분화(Granularity)에 대해서 고려
  - 서비스 진입 및 타 서비스로의 호출 시점 로깅 – "토핑 및 테일링(topping and tailing)" – 일반적으로 바람직한 접근
- 각각의 상호 호출을 위한 고유한 아이디 사용
- 로깅에 대한 속성을 서버에서 제어하는 것이 바람직함.

소스 컨트롤

- 소스 컨트롤 사용 - Integration Designer/Eclipse를 통해 여러 구성이 가능.
- 동시에 모듈당 한 명의 개발자만 허용: 가능하다면 해당 파일을 잠금.
  - 모듈 변경 사항을 통합하려고 하지 말 것: 복잡하고 오류가 발생하기 쉬움
- 너무 많은 체크아웃을 허용하지 말 것.
- 드물게는, 생성된 파일을 변경할 필요가 있을 수 있음
  - 필요한 경우 이러한 변경을 수행하기 위해서 ANT 스크립트 사용이 가능하며, 이러한 작업은 반복될 수 있음

반복적인 단위 테스트 실시

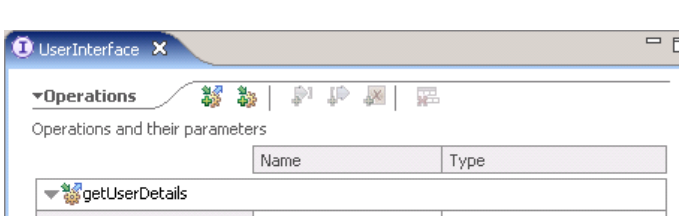


- V6.1 기준으로, WebSphere Integration Developer는 단위 테스트 기능 지원
- IBM Integration Designer V7.5는 통합 테스트 클라이언트 실행을 통하여 테스트 케이스 생성 기능을 추가
- 체크인 전에 테스트 기능 사용이 바람직함.
- 자동화의 일부로서 명령 행(Command Line)으로 실행이 가능함.
- 서버에 배치되고 웹 기반 GUI으로부터 실행될 수 있음
- 테스트가 다음 개발자에 의해서 실행될 수 있도록 할 것: 테스트를 체크인할 것
- 더 많은 정보가 담긴 내용에 대해서는 아래 사이트를 참고할 것
  - http://www.ibm.com/developerworks/websphere/library/techarticles/0806_gregory/0806_gregory.html

베스트 프랙티스: 개발

- 간략한 제품 개요
- 베스트 프랙티스
  - 기획, 설계 및 아키텍처
  - 프로세스
  - 개발
  - 성능
- 참고자료 및 추가 정보

오류의 적절한 처리



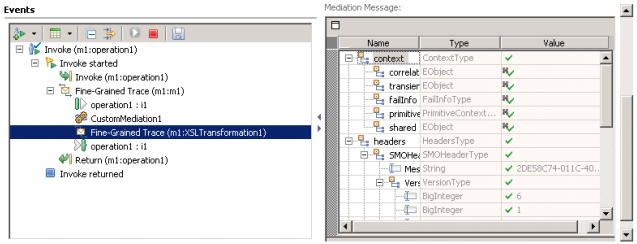
- 두 가지 유형의 오류가 존재
  - 모델링된 오류
    - 인터페이스 상에 나타남
    - 비즈니스, 점검된 오류라고도 알려져 있음
  - 모델링되지 않은 오류는 인터페이스 상에 나타나지 않음
    - 시스템, 런타임 또는 점검되지 않은 오류라고도 알려져 있음
    - 하나의 기술적 목적만을 가지고 있어야 함
- 이 두 가지 유형을 처리하기 위한 준비
  - 모델링되지 않은 오류를 가능한 소스 차원에서 처리할 것
  - 가능한 경우 모델링된 오류를 처리할 것

스키마와 WSDL를 간단하게 만들 것

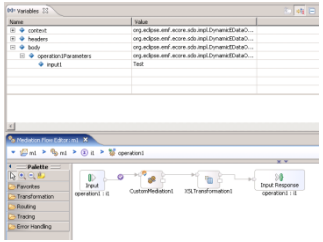
- 익명의 complex type을 피할 것
- 공백이거나 Null 인 namespace를 사용하지 말 것
- 2개의 라이브러리/프로젝트에서 동일한 namespace를 사용하지 말 것
- XSD 유형 전체를 활용할 것: 하나의 xsd:string 만을 사용하지 말 것

활용 가능한 디버깅 도구를 사용할 것

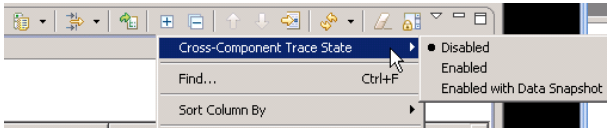
- 통합 테스트 클라이언트(Integration Test Client)



- 디버거(Debugger)
  - breakpoint 지원, 스텝오버(Step Over), SDO/SMO 검사 등



- 서버 로그 보기와 통합된 컴포넌트 교차 추적



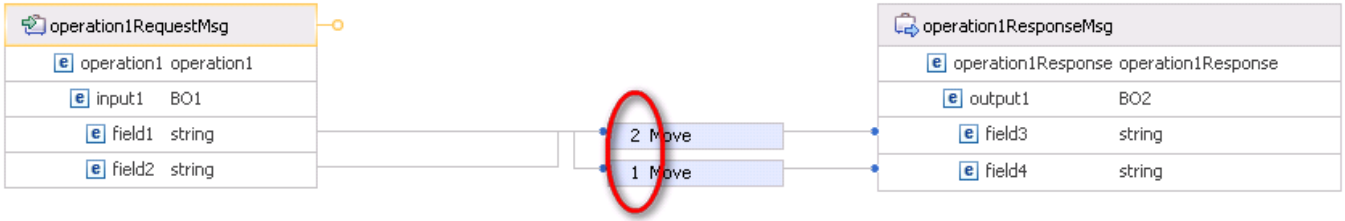
- 로깅

- Generic Service Client (V7.5)

중개 플로우에서 메시지 조작 기능에 대해서 이해할 것

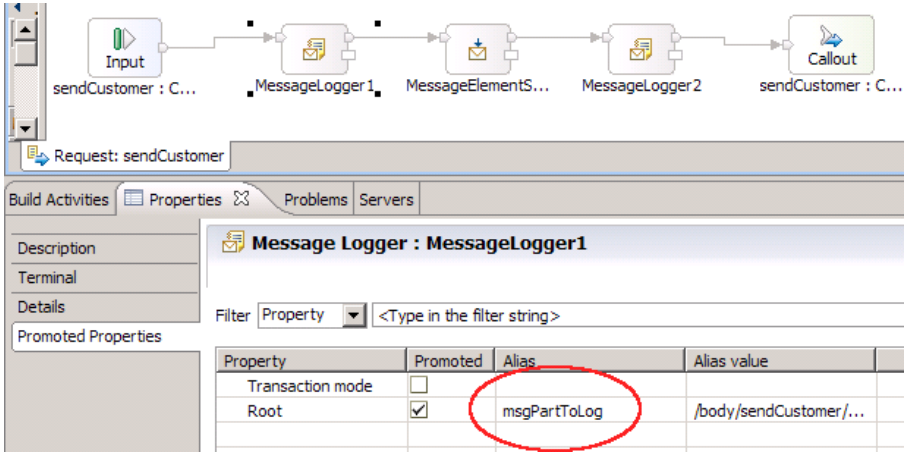
- Message Element Setter
  - 단순하지만 높은 성능을 가지고 있음
  - 메시지 유형을 변경할 수는 없음

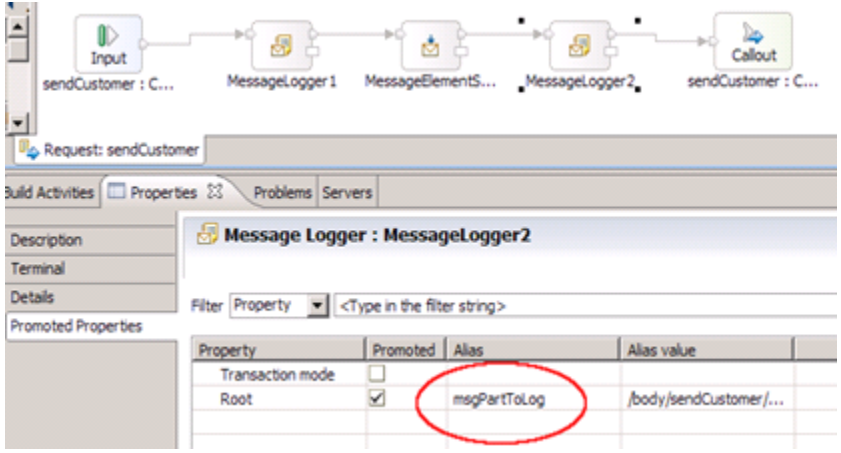
- XSLT
  - XSLT 기능을 사용하거나 XSLT를 사용해서 직접 작업하는 경우에 선호됨
  - 본래 그대로의 XML 데이터를 처리할 때 (즉 일반적으로 웹서비스를 사용해서 작업할 때) 더 성능이 좋음
- BO Map
  - BO 맵을 WPS와 공유하거나, relationship 서비스를 이용하거나, BO 맵퍼(Mapper)의 고유 기능을 필요로 하는 경우



연관된 속성의 경우, 속성 Promote

- 관련되는 경우, 특성들이 함께 변경될 수 있도록 이들 특성들을 동일한 Alias와 연결





기존의 사용자 코드를 새로운 기능으로 대체할 것

- UDDI Endpoint Lookup Primitive (7.0)
- 로깅 및 추적을 위한 Trace (7.0) / Enhanced Message Logger (6.2)
- Message Validator (7.0)
- Flow Order Primitive (7.0)
- Type Casting in XSLT Map (7.0)
- 기타 다양한 기능

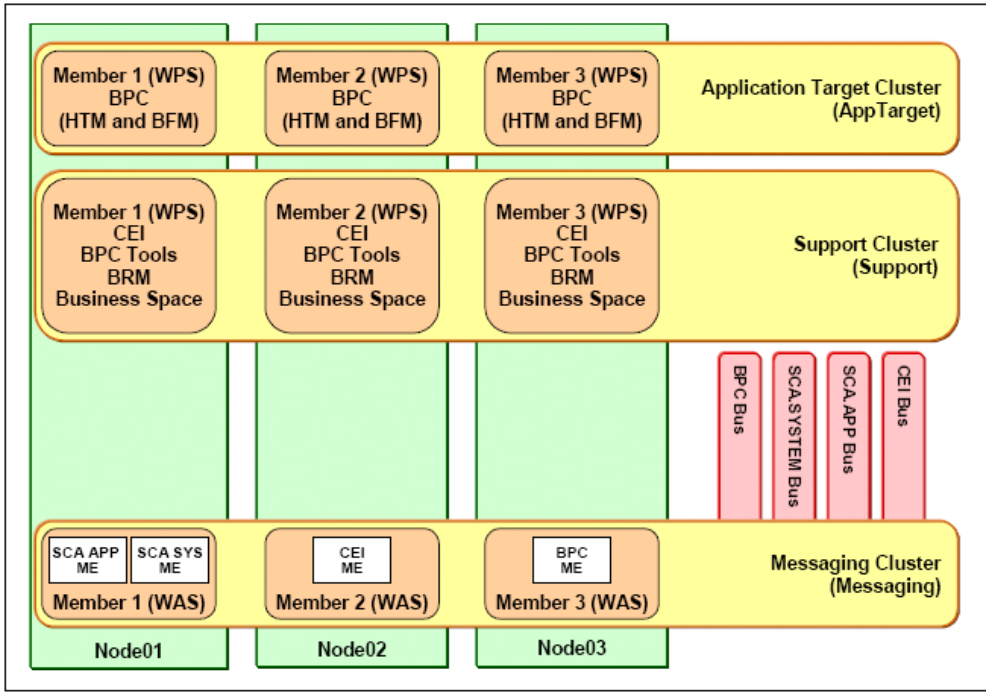
베스트 프랙티스: 성능

- 간략한 제품 개요
- 베스트 프랙티스
  - 기획, 설계 및 아키텍처
  - 프로세스
  - 개발
  - 성능
- 참고자료 및 추가 정보

* 런타임 라이센스 필요

시스템 토폴로지를 초기에 설계하고 적용할 것

- 클러스터 구성
  - 높은 가용성
  - 확장가능성
  - 더 나은 HW 활용
  - 성능 고려
- IBM은 "원격 메시징 및 원격 지원" 토폴로지 패턴을 권장



시스템 토폴로지를 초기에 설계하고 적용할 것

- 개발 및 테스트 환경을 생산 환경처럼 보이게 만들 것 (보안 설정 포함)
- 실제 데이터베이스 및 기타 지원 시스템을 가능한 초기에 사용할 것
- 로드 밸런서(Load Balancer)/HTTP 서버/프록시 서버가 필요한가?
- 연결하고자 하는 대상 시스템은 무엇이며 이들 시스템이 장애복구 및 확장성을 어느 정도로 보장할 것인가?
- 시스템 토폴로지 관련 참고자료
  - http://www.redbooks.ibm.com/redpieces/abstracts/sg247854.html

대형 오브젝트 – 신중히 처리할 것

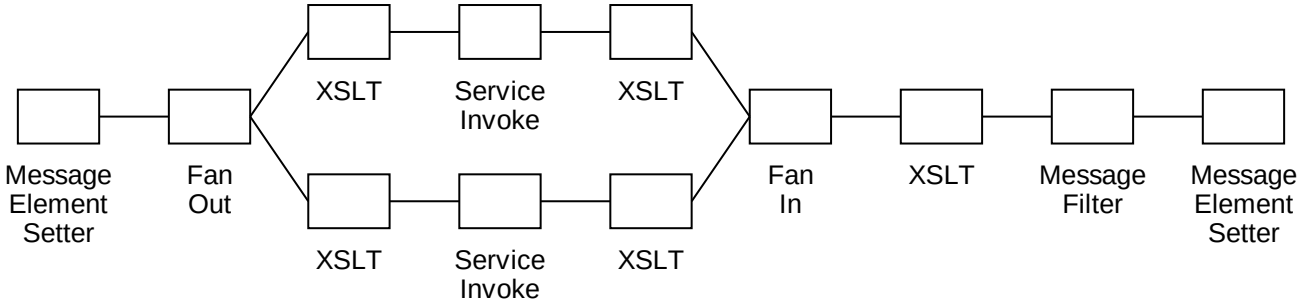
- 처리 원칙
  - IBM J9 JVM 덕분에, 대형 오브젝트 처리 성능이 V6.1 부터 많이 향상
  - 64-비트 지원은 힙 사이즈(Heap Size) 한계를 효과적으로 없애줌
  - 시스템을 통해 처리된 적이 없는 대형 데이터를 이동시키기에는 여전히 불충분
- 베스트 프랙티스
  - 대형 페이로드(Payload)가 배치(Batch)인 경우, 복수의 소형 페이로드로 전송할 것
  - 워크로드(workload)가 입력 메시지의 작은 부분만을 사용하는 경우에는 "Claim Check" 패턴을 사용할 것
    - http://www.ibm.com/developerworks/websphere/library/techarticles/1006_kharlamov/1006_kharlamov.html
  - 필요한 경우, 64비트를 이용할 것
  - 최신 버전을 고려할 것
  - V7.0.0.3부터, Business Object Lazy Parsing 모드를 사용하는 것을 고려

Business Object Lazy Parsing 모드

- XML 처리 및 변환에 맞게 최적화된 새로운 비즈니스 오브젝트 처리
- XML에 더욱 밀접하게 연결된 데이터 표현 방법 사용
- XML 지향적 시나리오에 맞는 중대한 성능상 장점 제공
- WebSphere ESB 7.0.0.3 with interim fix XC70031와 WebSphere Integration Developer 7.0.0.4 Ifix001에서 사용 가능
- IBM Integration Designer의 디폴트 설정

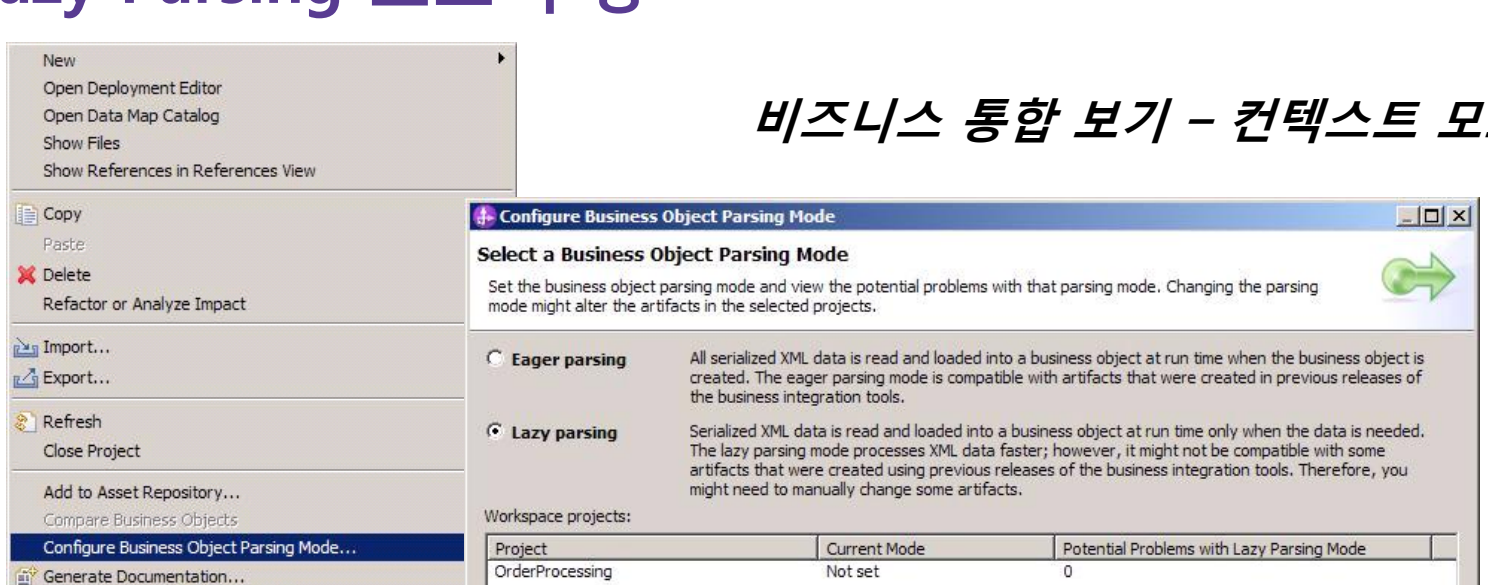
대형 메시지 및 복잡한 플로우를 위한 성능 향상

- Lazy Parsing 모드와 Eager Mode의 상대적 성능은 시나리오에 따라 달라질 수 있습니다
- 다음과 같은 경우에, 성능 이익이 가장 큽니다
  - 대형 XML 문서를 처리할 때
  - 변환 프리미티브와 기타 프리미티브 – 특히, XPath를 사용하는 프리미티브를 결합한 중개 플로우를 사용할 때
- 예를 들어 다음의 뱅킹 플로우는 Lazy Parsing 모드에서 최대 428% 더 빨리 수행됩니다



- 또한 힙(Heap) 점유는 Lazy Parsing 모드에 의해서 줄어들며, 이를 통해 대형 메시지 용량 향상을 가져옵니다

Lazy Parsing 모드 구성



비즈니스 통합 보기 – 컨텍스트 모드

참고 사이트

- 정보 센터(InfoCenter):
  - http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp
- 지원사이트(Support Sites):
  - http://www-01.ibm.com/software/integration/wsesb/support/
- 개발자 작업(developerWorks):
  - http://www.ibm.com/developerworks/websphere/zones/businessintegration/wesb.html
- 개발자 작업 포럼(developerWorks Forum):
  - http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1672
- IBM 교육지원(IBM Education Assistant):
  - http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp
- WebSphere ESB 개발 가이드
  - http://www-01.ibm.com/software/integration/wsesb/library/
- SOA 조언 및 유의사항 블로그:
  - http://soatipsntricks.wordpress.com/

요약

- 간략한 제품 개요
- 베스트 프랙티스
  - 기획, 설계 및 아키텍처
  - 프로세스
  - 개발
  - 성능
- 참고자료 및 추가 정보

감사합니다