모바일 플랫폼 비교 : IBM Worklight 대 DIY 방식 개발

IBM

내용

이 문서에서는 IBM® Worklight® Developer Edition 플러그인과 "DIY" 개발 방식을 비교하며, 웹, 하이브리드 및 네이티브 모바일 애플리케이션 개발 도구를 중점적으로 설명합니다.

IBM Competitive Project Office에서는 IBM Worklight Developer Edition 6.1과 Eclipse IDE(Integrated Development Environment) , Android SDK(소프트웨어 개발 킷) 및 jQuery, Jersey (REST 서비스용) , Apache Cordova, 기타 프레임워크 등 주요한 프레임워크를 비교하였습니다.

이 솔루션들은 몇 가지 유사점이 있습니다. Worklight 솔루션과 "DIY" 방식 모두 무료로 다운로드하여 사용할 수 있고, 비슷한 오픈 소스 구성 요소를 기반으로 하며, 멀티 플랫폼을 지원하고, 플랫폼 에뮬레이터에 애플리케이션을 배치하고 디버깅할 수 있습니다.

위에 열거한 점들 이외에 IBM Worklight는 "DIY" 방식에 비해 다음과 같은 이점을 제공합니다.

- **개발자 친화적인 설치와 구성:** Worklight는 다양한 솔루션 구성 요소를 설치하고 구성할 수 있는 자세한 설명서를 제공합니다.
- **앱 개발을 가속하는 기획과 구현:** Worklight는 뛰어난 생산성을 보장하는 설계 및 개발 기능을 제공합니다. 공통의 UI 제어를 위해 위저드 기능, 시각적 UI 편집기, 사전에 패키징된 UI 프레임워크, JavaScript API(Application Programming Interface)를 사용합니다. 코드를 계속 재사용하면서 "즉시" 플랫폼을 추가할 수 있습니다.

- **기간 업무 시스템과 연동:** Worklight는 백엔드 기간 업무 시스템과 서비스를 간편하게 검색할 수 있습니다. 또한 프론트엔드/백엔드 통신 소스 및 대상을 중개할 수 있는 어댑터를 간편하게 작성하고 테스트할 수 있습니다.
- **앱 배치와 테스트:** Worklight의 구성요소중 하나인 IBM Worklight Console을 사용하여 모바일 애플리케이션과 어댑터를 임베드된 개발 서버 환경에 배포할 수 있습니다. 개발자들은 테스트 및 운영 환경에 빌드를 배포하고 테스트하기 전에, 테스트를 목적으로 애플리케이션을 신속하게 배치할 수 있습니다.
- **디버깅 및 테스트:** Worklight에서는 디버깅 및 테스트를 위해 모바일 브라우저 시뮬레이터, SDK 에뮬레이터, 실제 모바일 기기를 사용할 수 있습니다. 모바일 브라우저 시뮬레이터는 일반적인 에뮬레이터보다 속도가 빨라 테스트 생산성이 높으며, 테스트를 위해 실제 기기를 구입할 필요가 없으므로 합리적인 비용으로 테스트 환경을 구현할 수 있습니다. Worklight는 또한 에뮬레이터 및 실제 모바일 기기에서 모바일 애플리케이션의 기능을 테스트할 수 있는 Mobile Test Workbench for Worklight를 옵션(무료)으로 제공합니다.

개요

모바일 프로젝트를 시작하기 전에, IBM Worklight로 대표되는 플랫폼 방식과 DIY 방식이 무엇이 같고 다른지 이해할 필요가 있습니다. 모바일 프로젝트를 수행하는 툴로 DIY 방식을 선택할 경우 실제로 개발이 진행되면, 개발을 진행하면서 여러가지 문제를 동시에 해결해야 하므로, DIY 방식을 선택하면 무엇이 필요하고, 어떤 구성요소들을 설치/구성/통합 및 유지보수 해야 하는지 미리 알아둬야 합니다.

IBM Worklight Developer Edition은 DIY 방식과 마찬가지로 오픈 소스 솔루션을 기반으로 만들어졌으며, 모바일 앱 라이프사이클 관리에 필요한 모든 구성요소가 패키징되어 있거나, 추후에 다운로드해 설치/구성할 수 있도록 문서로 잘 정리돼 있습니다.

또한 Worklight는 이러한 오픈 소스 핵심 기능 외에 기업 모바일 애플리케이션의 개발과 배치를 효율적으로 지원할 수 있는 엔드-투-엔드 모바일 플랫폼 기능을 포괄하고 있습니다. 개발용으로 바로 사용할 수 있는 Worklight Developer Edition(Eclipse IDE 플러그인)은 Worklight Studio와 IBM WebSphere® Application Server Liberty Profile Edition으로 구성됩니다. 상기 구성 요소는 모바일 애플리케이션의 설계, 개발, 배포, 디버깅 및 테스트 영역에서 DIY 개발 방식보다 월등한 기능을 제공합니다(자세한 내용은 *IBM Worklight의 이점*에서 살펴보도록 하겠습니다).

솔루션 구성 요소

솔루션 구성 요소를 자세히 살펴보기 전에, 먼저 각 개발 방식의 주요 구성 요소를 살펴보도록 하겠습니다.

그림 1은 DIY 방식의 구성 요소를 보여줍니다. DIY 방식은 Eclipse IDE, Android SDK 및 개발 도구, 백엔드 서비스를 호스트하는 Tomcat 서버, 백엔드 서비스에 액세스하는 REST 라이브러리로 구성됩니다.

그림 2는 Worklight 솔루션의 구성 요소를 보여줍니다. IBM Worklight는 Worklight Studio, 개발용 Android SDK, 디바이스 런타임 라이브러리, Worklight Server 런타임(이중 WAS Liberty Profile은 백엔드 서비스와 연결될 어댑터를 실행하는 데 사용됨) 및 백엔드 서비스를 호스트하는 Tomcat 서버로 구성됩니다.

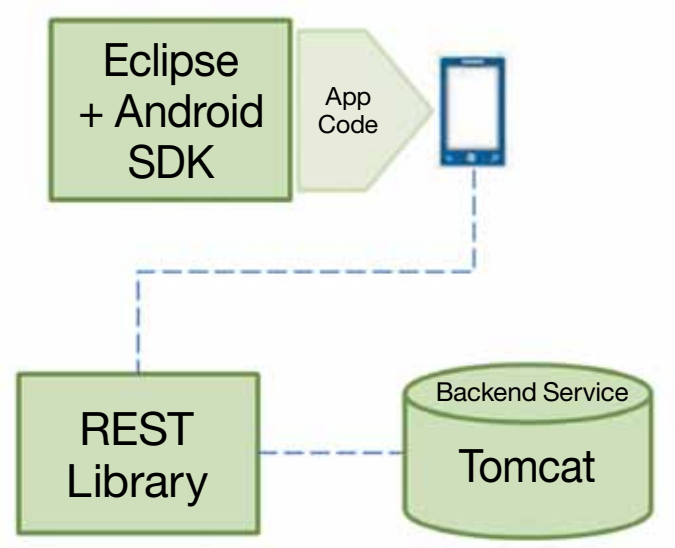

그림 1. DIY 개발 방식의 구성 요소

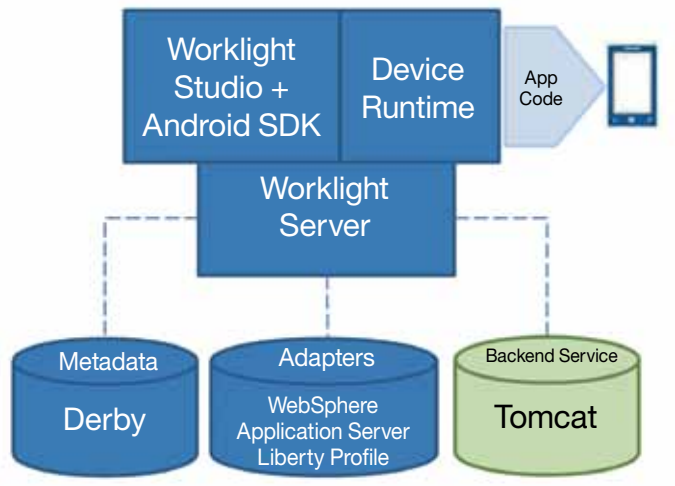

그림 2. IBM Worklight 솔루션의 주요 구성 요소

IBM Worklight Developer Edition 6.1은 IBM Worklight Enterprise Edition을 개발자용으로 경량화한 무료 버전으로, Worklight 엔터프라이즈 버전에서 제공하는 대부분의 구성 요소와 전체 코드를 포함하고 있으며 IBM 기술 지원을 받을 수 있습니다.

참고: Worklight는 개발자 버전 외에 IBM Worklight Consumer Edition과 IBM Worklight Enterprise Edition이 있습니다.

IBM Competitive Project Office 팀에서는 Worklight Developer Edition v6.1을 Eclipse, Android SDK 및 기타 구성 요소를 기반으로 하는 DIY 방식과 비교했습니다. IBM 팀에서 비교한 두 환경의 구성은 아래와 같습니다.

- Eclipse Juno v4.2
- Eclipse v22.3.0용 ADT(Android Development Toolkit) 플러그인
- Android SDK v22.3
- JDK 1.7.0_45 및 JRE7
- Apache Tomcat 7.0.47
- Jersey 1.17.1(JAX-RS 또는 JSR 311 사양 구현)
- JQuery Mobile 1.3.2 라이브러리
- PhoneGap 2.9.0

Worklight Developer Edition은 Eclipse IDE의 플러그인으로 설치됩니다. 개발 환경은 IBM Worklight Studio입니다. Worklight Studio는 Eclipse의 기본 도구를 플러그인에서 제공하는 다양한 기능으로 확장하며, 지원하는 모바일 기기의 다양한 SDK와 통합됩니다.

특히, Worklight는 애플리케이션과 어댑터를 배치하는 데 사용되는 임베드된 개발 서버 IBM WebSphere Application Server Liberty Profile와 함께 제공됩니다.

JQuery 코어는 Worklight Studio 설치 팩에 같이 포함되나, JQuery Mobile은 포함돼 있지 않으므로 다운로드하여 설치해야 합니다. IBM Dojo Mobile 도구도 설치팩에 포함돼 있지만, 여기서는 배제하도록 하겠습니다.

Cordova 프레임워크는 Worklight 설치팩에 함께 제공됩니다. PhoneGap은 DIY 솔루션에 사용되었습니다. 그러나 Cordova와 PhoneGap의 컨텐츠는 유사한데, Cordova는 Apache 조직에 속하는 오픈 소스 프로젝트이고, PhoneGap은 Adobe에서 제공하는 Cordova 배포입니다.

모바일 애플리케이션의 유형

IBM 팀에서는 플랫폼 방식과 DIY 방식을 비교하기 위해 샘플 모바일 애플리케이션을 만들어 개발 관점에서 두 방식을 비교했습니다. 모바일 애플리케이션 앱은 다음과 같이 3가지로 분류할 수 있습니다.

모바일 웹 앱

모바일 웹 앱 유형은 모바일 애플리케이션처럼 보이는 웹 사이트를 말합니다. 모바일 웹 앱은 HTML5에서 단일 페이지 애플리케이션으로 만들 수 있으며, 모바일 기기 페이지에서 앵커 사용 페이지로 이동하는 것을 시뮬레이션할 수 있습니다. 이 유형의 앱은 모바일 기기의 브라우저에서 실행되지만, 브라우저 단추나 표시줄이 보이지 않아 하이브리드 앱이나 네이티브 앱과 구별하기 어렵습니다. 사용자는 URL로 이동하여 앱에 액세스하고 모바일 기기의 브라우저에서 해당 페이지를 책갈피로 추가할 수 있습니다.

장점:

- 네이티브 앱처럼 보이지만 개발 비용이 훨씬 저렴합니다.
- 변경 사항이 적용되면 사용자가 즉시 사용할 수 있습니다.
- 네이티브 앱에 비해 확장성이 우수하고 구축 비용도 합리적인 크로스 플랫폼입니다.

단점:

- 위치 및 미디어 파일과 같이 모바일 기기에서 기본으로 제공하는 기능에만 접근할 수 있습니다.
- 모바일 브라우저가 화면 공간을 차지할 수 있습니다.

모바일 웹 앱 방식으로 앱을 개발하는 것은 일반적인 웹 애플리케이션 개발과는 차이가 있으므로, 이번 비교에서 이 방식은 배제하도록 하겠습니다.

하이브리드 앱

하이브리드 앱은 네이티브 앱과 웹 앱이 결합된 것입니다. 다양한 기기의 기능을 사용할 수 있으며 네이티브 앱과 같이 앱 스토어에서 다운로드할 수 있습니다. 또한 HTML, JavaScript, CSS와 같은 웹 기술을 사용하여 만들 수 있습니다. 하이브리드 방식의 앱은 모바일 기기에 임베드된 브라우저에 표시되므로 참조할 URL이 없습니다.

기업들은 기존에 모바일 웹 앱 방식으로 개발된 앱을 하이브리드 앱에 패키징해(웹 앱에 하이브리드 앱 형식을 덧씌운 것) 앱 스토어에 올립니다. 이렇게 하면 동일한 HTML 코드를 멀티 모바일 운영 체제에서 재사용할 수 있으므로 앱을 다시 개발하느라 애쓸 필요 없이 멀티 플랫폼을 지원할 수 있습니다.

장점:

- 모바일 기기의 고유 기능 및 고급 기능(위치 정보, 카메라 등)을 사용할 수 있습니다.
- 모바일 브라우저에 페이지를 표시하는 데 HTML을 사용할 수 있습니다.

단점:

- 앱 스토어에서 규정한 승인, 품질, 보안 절차를 준수해야 합니다.
- 사용자가 직접 다운로드, 설치 및 앱 관리 등을 해야 합니다.

네이티브 앱

네이티브 앱은 앱 스토어에서 다운로드할 수 있으며 모바일 기기에 설치됩니다. 네이티브 방식의 앱은 특정 플랫폼용으로 개발되므로 카메라, 센서(GPS, 가속도 센서, 나침반 등등), 연락처 등 모바일 기기에서 제공하는 고유 기능을 활용할 수 있습니다. 사용자 인터페이스 제스처도 앱에 통합할 수 있으며, 모바일 기기의 알림 시스템(Notificication)을 사용할 수 있고, 네트워이 끊겨도 오프라인으로 작동할 수 있습니다.

장점:

- 모바일 기기의 고유 기능을 활용할 수 있습니다.
- 모바일 웹 앱보다 빠릅니다.

단점:

- 앱 스토어에서 규정한 승인, 품질, 보안 절차를 준수해야 합니다.
- 사용자가 직접 다운로드, 설치 및 앱 관리 등을 해야 합니다.
- 개발하는 데 시간과 비용이 많이 듭니다.
- 모바일 기기의 파편화(예: 다양한 언어/운영체제 버전)로 비용은 갈수록 늘어납니다.

IBM Worklight 접근 방식

Worklight는 앞에서 설명한 일반적인 모바일 애플리케이션 개발 방식을 지원하며, *하이브리드 앱 – 웹* 및 *하이브리드 앱 – 혼합형* 등의 변형된 하이브리드 앱의 개발도 지원합니다.

*하이브리드 앱 – 혼합형*을 사용할 경우 컨테이너를 통해 모바일 기기의 고유 기능을 활용하는 애플리케이션을 개발할 수 있지만, 다른 플랫폼별 고유 구성 요소(예: 라이브러리 또는 특정 사용자 인터페이스 요소)를 사용하여 모바일 애플리케이션을 개선할 수도 있습니다.

그림 3은 Worklight에서 지원하는 다양한 모바일 애플리케이션 개발 방식을 보여줍니다.

그림 3. Worklight에서 지원하는 모바일 애플리케이션 개발 유형

어떤 개발 방식을 선택할 것인가

어떤 개발 방식으로 모바일 앱을 개발할 것인가. 이에 대한 결정은 기업마다 다를 것입니다. 모바일 프로젝트를 담당하는 팀은 표1의 모바일 앱 개발 방식별 비교표를 참조하여, 기업의 비즈니스 목표, 소구 대상 그리고 모바일 앱 개발에 관련된 내부 개발 역량을 기반으로 우리 기업에 가장 적합한 개발 방식이 무엇인지 선택해야 합니다.

범주	웹 앱	하이브리드 앱	하이브리드 –혼합 앱	네이티브 앱
배움의 난이도	쉬움	중간	중간	어려움
애플리케이션 성능	느림	보통	보통	빠름
모바일 기기에 대한 이해	필요 없음	일부 필요	일부 필요	많이 필요
개발 주기(구현, 테스트, 배포)	짧음	중간	중간	다소 길
이기종 플랫폼 이식성	높음	높음	중간	없음
모바일 기기의 고유 기능 지원	일부	대부분	전부	전부
내장 메커니즘을 통한 배포	지원하지 않음	가능	가능	가능
기기의 고유 기능 확장	안됨	지원	지원	지원

표 1. 모바일 앱 개발 방식 비교

샘플 애플리케이션 시나리오

Worklight 및 DIY 개발 방식을 개발 관점에서 비교하기 위해 IBM 팀은 JKE 라는 가상의 회사를 설정하고, 샘플 모바일 애플리케이션을 만들었습니다. 이 샘플 애플리케이션은 계좌 거래 상세 및 계좌 잔고와 같은 은행의 거래 내역 조회와 근처의 자선단체 검색 및 배당금을 자선 단체에 기부하는 기능들을 제공하도록 구성돼 있습니다.

각각의 방식으로 두 가지 버전의 애플리케이션을 만들었습니다. 하나는 Cordova, JavaScript, HTML5, CSS 등의 기술을 사용한 하이브리드 앱 버전이고, 다른 하나는 Android 플랫폼용으로 Java를 사용한 네이티브 앱입니다. 샘플 애플리케이션 사용자 인터페이스(Android 에뮬레이터)는 그림 4에 나와 있습니다.

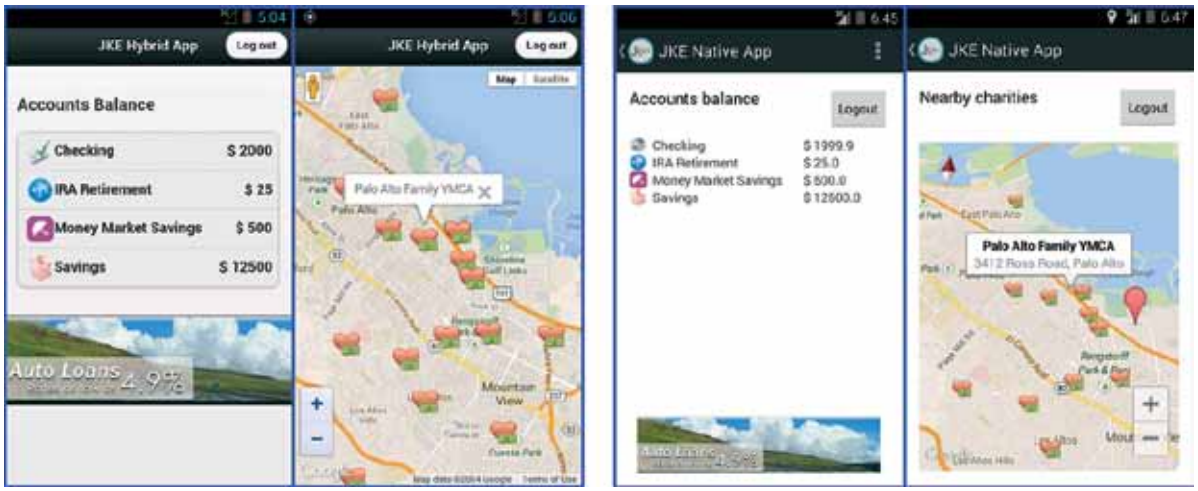

그림 4. Android 에뮬레이터에 표시되는 가상기업의 하이브리드 앱(왼쪽)과 네이티브 앱(오른쪽)

이 앱은 백엔드 REST 서비스(JKE 서비스)와 통신하여 사용자를 인증하고 계좌별 거래 현황 및 잔고 내역을 검색합니다.

이 서비스는 모든 시나리오에서 Tomcat 서버에 배치됩니다. 그러나 Tomcat은 REST 서비스를 지원하는 고유 JAX-RS 구현을 제공하지 않으므로, Jersey 라이브러리 Jar 파일을 수동으로 Tomcat 설치 폴더에 추가했습니다. 또는 Tomcat에 배치할 수 있도록 Jersey Jar 파일이 JKE 서비스 War 파일에 패키징되었을 수 있습니다.

반면, WebSphere Application Server Liberty Profile은 추가 구성 없이 REST 서비스를 호스트할 수 있습니다. Worklight 환경에서는 Tomcat과 달리 Jersey 라이브러리 파일을 해당 솔루션에 추가하지 않고도 JKE 서비스를 Liberty Profile에 직접 배치할 수 있습니다. 이는 Liberty Profile이 기본적으로 REST 서비스를 지원하기 때문입니다. 그러나 IBM 팀에서는 이기종 환경의 시나리오를 구현하고자 했습니다. 이 경우 회사의 어딘가에서 레거시 서비스가 실행 중이고 어댑터를 사용해 기간계 업무 시스템에 플러그인하기 위해 Worklight 솔루션이 사용됩니다.

모바일 앱에서 백엔드 기간계 업무 시스템에 액세스할 수 있도록 하기 위해 IBM 팀에서는 몇 개의 다른 기술을 사용했습니다. DIY 방식의 하이브리드 앱에서는 백엔드 서비스에 액세스하기 위해 JavaScript Ajax 호출을 사용했고, 네이티브 앱에서는 Apache HTTP 호출을 사용했습니다. Worklight 솔루션에서는 백엔드 기간계 업무 시스템 연동을 위해 어댑터를 만들 수 있는데, 하이브리드 앱과 네이티브 앱 환경 모두 이 어댑터를 사용하였습니다.

네이티브 앱도 어댑터를 사용하여 백엔드 기간계 업무 시스템에 액세스할 수 있습니다. IBM 팀은 Worklight를 사용하여 Worklight Library API 프로젝트를 통해 어댑터를 만들고 배치했습니다. 이 API는 네이티브(Java) 코드를 통해 어댑터 프로시저(원래 JavaScript로 작성됨)를 호출할 수 있도록 네이티브 앱용 인터페이스를 제공합니다. IBM 팀에서는 어댑터와 네이티브 앱을 위한 각각의 프로젝트를 만들었습니다. 하이브리드 앱의 경우 어댑터 코드와 앱 코드 모두 JavaScript를 기반으로 하므로, 어댑터 코드와 앱 코드가 같은 Worklight 프로젝트에 있을 수 있습니다. 샘플 애플리케이션 시나리오에서 네이티브 앱은 WebSphere Application Server Liberty Profile 서버에서 실행되는 Worklight 어댑터를 사용하여 Tomcat에서 실행되는 JKE 레거시 서비스에 액세스합니다.

자선 단체를 찾는 데 사용되는 위치 정보 서비스도 환경에 따라 달라집니다. 두 환경 모두에 사용되는 하이브리드 앱의 경우 IBM 팀에서는 Cordova를 사용하여 지도에서 자선 단체(마커)를 찾아서 그리는 좌표를 리턴했습니다. 네이티브 앱의 경우 오픈 소스 환경에는 Android 위치 정보 API를 사용하고, Worklight 환경에는 Worklight 위치 정보 API를 사용했습니다.

표 2는 Worklight와 DIY 방식의 각각의 앱 유형에 사용되는 기술을 요약하여 보여줍니다.

두 개발 방식의 유사점

앞에서 두 개발 방식의 유사점을 확인하였습니다. 우선 두 개발 방식 모두 공개되어 있으며, 무료입니다. 여기서는 Worklight Enterprise Edition을 개발자용으로 경량화한 Worklight Developer Edition을 중점적으로 다루고 있습니다.

두 솔루션 모두 개방형 오픈 소스 구성 요소를 기반으로 합니다(*솔루션 구성 요소* 부분에서 상세 내용을 확인할 수 있습니다).

JKE 샘플 애플리케이션 부분에서 설명한 것과 같이, 두 개발 방식은 비슷한 기술을 사용하여 웹, 하이브리드, 네이티브 앱 개발을 지원하고, 백엔드 기간 업무 시스템에 액세스하는 기술을 제공합니다. 두 개발 방식 모두 애플리케이션 개발과 디버깅 및 테스트 목적으로 에뮬레이터를 배치하는 작업에 플랫폼에서 제공하는 SDK(예: Android 또는 iOS)에 의존합니다.

	DIY 시나리오		Worklight	
	하이브리드 앱	네이티브 앱	하이브리드 앱	네이티브 앱
프론트엔드 기술	HTML, JavaScript, CSS, JQuery Mobile	Java	HTML, JavaScript, CSS, JQuery Mobile	Java
백엔드 서버	Tomcat	Tomcat	Tomcat/ WebSphere Application Server Liberty Profile	Tomcat/ WebSphere Application Server Liberty Profile
백엔드 기간 업무 서비스 액세스	JavaScript Ajax	Apache HTTP API	Worklight 어댑터	Worklight 어댑터
위치 정보	Cordova	Android 위치 정보 API	Cordova	Android 위치 정보 API
지도 및 장소	Maps API(Google)	Maps API(Android)	Maps API(Google)	Maps API(Android)

표 2. 샘플 모바일 앱을 만드는 데 사용되는 기술

IBM Worklight의 이점

Worklight Developer Edition은 DIY 방식에 비해 여러 가지 이점을 제공합니다.

개발자 친화적인 설치와 구성

DIY 방식으로 앱을 개발할 때는 개발자 스스로 어떤 오픈소스 구성요소를 설치하고 구성할지를 먼저 선택해야 합니다. 설치할 오픈소스 구성요소를 선택한 후에는 다운로드 소스를 알고 있어야 하고 사용할 오픈소스 구성요소 버전 간 호환성 및 설명서, 참조 문서, 토론 포럼 등도 스스로 찾고 잘 이해하고 있어야 합니다.

Worklight를 사용할 경우에는 환경 설정 준비 단계로 일부 구성 요소를 다운로드하여 설치해야 하지만, 이 경우 구성 요소를 어떤 순서로 설치할지, 어디서 찾을지가 자세하게 문서화돼 있습니다.

또한 Eclipse의 기존 인스턴스에 Worklight를 플러그인으로 설치할 때 Worklight Studio와 Worklight Server가 같이 구성됩니다. 이를 통해 모바일 애플리케이션의 설계, 개발, 배치를 효율적으로 수행할 수 있습니다. Worklight Server는 어댑터 및 애플리케이션 배치를 위해 WebSphere Application Server Liberty Profile을 사용합니다.

Worklight Developer Edition 설치 및 구성정보에 대해 자세히 알아보십시오.

앱 개발을 가속하는 기획과 구현 환경

Worklight는 모바일 애플리케이션의 기획과 구현을 보다 쉽고 생산적으로 만들어 주는 다음과 같은 기능을 제공합니다.

- 프로젝트 생성 마법사가 UI 및 로직 구성 요소, 백엔드 기간 업무 시스템 연동 어댑터 및 Worklight 서버에 애플리케이션을 배치하는 일련의 구조를 생성합니다.
- WYSIWYG("what you see is what you get") 편집기 및 드래그 앤 드롭 방식의 편집기 등이 제공돼 설계가 용이합니다. 웹 앱 및 하이브리드 앱에서는 HTML 컨트롤과 jQuery Mobile 및 Dojo Mobile 위젯을 지원하며, Android 네이티브 앱에서는 Worklight가 Java UI 컨트롤을 지원합니다.
- 임베드된 Dojo 툴킷 패키지 및 도구를 사용하여 모바일 웹 애플리케이션을 개발할 수 있습니다.
- 한 줄의 코드로 캡슐화된 JQuery를 프로젝트에 추가할 수 있습니다.
- JavaScript API가 환경에 관계없이 일반 UI 컨트롤을 호출하며, 이러한 컨트롤을 각 플랫폼에 고유한 방식으로 표시합니다.
- "스킨" 은 동일한 운영 체제 제품군의 다양한 폼 팩터를 하나의 실행 파일로 지원합니다.
- 프레임워크를 통해 애플리케이션을 다른 언어로 변환할 수 있습니다.
- 환경을 "즉시" 추가할 수 있는 기능을 제공하는데, 이를 통해 지원되는 환경(코드 재사용 지원) 별 코드를 분리할 수 있습니다.
- 환경 최적화 프레임워크를 제공합니다. 앱의 핵심 로직과 설계 지침은 웹 기술(예: HTML, CSS, JavaScript)을 사용하여 작성되며 모든 환경에서 공유됩니다. 필요 시 환경별 최적화 작업을 실행할 수 있습니다.
- Cordova 프레임워크가 모든 모바일 환경에 통합되었습니다. Worklight 프레임워크는 Cordova 라이브러리를 사용합니다. 개발자는 이를 통해 모바일 기기 고유의 기능에 액세스할 수 있습니다.
- 네이티브 UI로 하이브리드 앱을 개선할 수 있습니다. Worklight API를 사용하여 동일한 앱에 모바일 웹 페이지와 네이티브 페이지를 혼합 구성할 수 있습니다. 또는 Cordova를 사용하여 하이브리드 앱에 네이티브 기능을 접목할 수 있습니다.
- Worklight 클라이언트 Java Script API는 네이티브 모바일 플랫폼 API에 연결되며 동적 HTML 로드를 지원합니다.
- "축소" 와 연결(파일 수를 줄여줌) 및 HTML5 애플리케이션 캐시를 사용하여 모바일 웹 앱 성능을 최적화할 수 있습니다.

DIY 방식은 모바일 애플리케이션 개발을 위한 기본적인 기능만 제공할 뿐이며, Worklight에 포함된 Worklight 편집기, 코드 재사용 및 위에 열거한 다양한 기능 등은 제공하지 않습니다.

기간 업무 시스템과 연동

DIY 방식은 오픈 프레임워크를 사용하여 JavaScript Ajax 호출과 같은 백엔드 REST 서비스(웹 앱 및 하이브리드 앱의 경우) 및 Apache HTTP API(네이티브 앱의 경우)에 액세스할 수 있습니다. 따라서 다양한 유형의 애플리케이션을 지원하기 위해서는 필요한 기술을 모두 익혀야 합니다.

또한 백엔드의 기간 업무 서비스 호출을 클라이언트에서 하게 되는데, 이는 애플리케이션 설계 관점에서 권장하는 옵션은 아닙니다. 예를 들어 클라이언트가 서버의 요청 응답을 기다리는 대기 시간을 최소화하기 위해 개발자는 비동기 호출을 구현해야 하고 서비스 가용성을 보장해야 합니다.

Worklight 솔루션도 오픈 기술을 사용하여 백엔드 기간 업무 서비스에 액세스할 수 있지만, 기간계 연동 목적으로 사용할 수 있는 어댑터를 만들 수도 있습니다. Worklight에서 SOAP 서비스에 대해 어댑터를 만드는 작업은 거의 자동으로 이뤄집니다. 서비스 WSDL 파일을 지정하고 마법사가 제시하는 단계를 따르기만 하면 됩니다. 기타 백엔드 서비스(예: REST 서비스)는 백엔드 기간 업무 서비스에 액세스하는 프로시저를 선언하고 구현해야 하지만, Worklight에서 구현하기 쉽게 어댑터 구조를 만들어줍니다.

어댑터는 JavaScript로 정의되지만 Java 코드로 호출할 수도 있습니다. 구현된 어댑터는 웹 앱, 하이브리드 앱 및 네이티브 앱에서 같이 사용할 수 있습니다. 어댑터는 또한 클라이언트와 서버간 통신을 중개하며, 임베드된 서버(WebSphere Application Server Liberty Profile) 또는 지원되는 다른 서버에 배치됩니다.

오픈소스 기술과 비교했을 때 어댑터를 사용하면 다음과 같은 이점이 있습니다.

- 백엔드 서비스 검색(SOAP 또는 SAP 서비스). 서비스 WSDL 파일을 기반으로 하는 프로시저로 어댑터를 자동으로 생성합니다.
- 어댑터 기능을 향상시켜 주는 서버 측 스크립팅 (사전 및 사후 로직, 하나의 트랜잭션으로 처리가 수행됨, 여러 소스의 매시업)
- 어댑터에 Java를 사용할 수 있는 기능(어댑터 프로시저에서 Java 코드를 호출하고, Java 코드에서 어댑터 프로시저를 호출함)
- 인증 및 백엔드 액세스를 관리하고 더 많은 서버 기능을 사용하는 네이티브 Worklight 클라이언트 API
- 암호화된 캐시 메커니즘을 사용하여 클라이언트에 중요한 데이터 저장
- IBM Worklight 네이티브 API 라이브러리를 사용하여 네이티브 Android 애플리케이션이 Worklight Server와 통신할 수 있는 기능

앱 배포와 테스트

Worklight에는 어댑터 및 애플리케이션 배포용 서버(WebSphere Application Server Liberty Profile)가 포함돼 있습니다. 애플리케이션에서 코드를 사용하기 전에 어댑터를 Worklight 서버에 배포하고 어댑터 프로시저를 테스트할 수도 있습니다.

어댑터 및 애플리케이션을 QA(품질보증)및 운영 환경에 배포하기 위해 Apache Tomcat과 같은 서버를 사용할 수 있습니다. 이 옵션을 사용하려면 추가 구성을 직접 수행하거나 Ant 스크립트를 사용해야 합니다. 이 문서의 목적에 따라 IBM 팀에서는 Worklight 솔루션에 기본으로 포함된 WebSphere Application Server Liberty Profile을 사용했습니다.

DIY 방식에서는 디버깅 및 수동 테스트를 위해 플랫폼 에뮬레이터에 애플리케이션을 수동으로 배치(설치)하는 방법으로 앱을 배포할 수 있습니다. 애플리케이션을 전용 앱 스토어에 배치하는 것과 같은 광의의 배포는 여기에서는 언급하고 있지 않으나, 이 용도로 사용할 수 있는 DIY 솔루션도 없습니다. Worklight의 엔터프라이즈 버전에서는 이 기능을 제공합니다. Worklight Application Center가 그것으로, 사전 공개 버전이나 운영 환경에 모바일 앱을 배포 및 관리할 수 있는 기업용 앱 스토어를 만들 수 있습니다.

디버깅 및 테스트

Worklight는 디버거 API를 제공합니다. 로그 메시지를 사용 환경의 로그에 인쇄할 수 있습니다. API는 멀티플랫폼입니다. 따라서 해당 애플리케이션이 실행되는 플랫폼에 따라 출력 대상이 변경됩니다.

Worklight는 앱을 미리 보고 디버깅할 수 있는 모바일 브라우저 시뮬레이터를 제공합니다. 또한 Cordova API도 시뮬레이션합니다. 애플리케이션을 에뮬레이터에서 미리 보는 대신 브라우저에서 미리 볼 수 있으므로 훨씬 신속하게 애플리케이션을 로드하고 미리 볼 수 있습니다.

Worklight에는 또한 Mobile Test Workbench for Worklight를 옵션으로 제공합니다. 이를 이용해 Worklight에서 작성되는 Android 및 iOS 모바일 애플리케이션의 기능을 자동으로 테스트할 수 있습니다.

DIY 방식은 디버거 API를 제공하지 않습니다. 따라서 개발자는 애플리케이션 디버깅을 위해 일반적인 방법으로 메시지를 콘솔 또는 디바이스 화면에 게시해야 합니다. 웹 브라우저 시뮬레이터도 제공되지 않으므로, 플랫폼 에뮬레이터에서 디버깅 및 테스트를 수행해야 합니다. 플랫폼 에뮬레이터는 모바일 브라우저 시뮬레이터보다는 속도가 떨어지지만, 수동 테스트 또는 자동 테스트를 수행할 수 있습니다. 또한 DIY 방식은 즉시 사용 가능한 기능 테스트 도구를 제공하지 않으므로, DIY 방식을 선택한 경우에는 IDE(Integrated Development Environment)와 통합된 테스트 도구를 따로 설치해 기능 테스트를 수행해야 합니다.

결론

이 문서에서는 IBM Worklight Developer Edition이 어떤 점에서 DIY 방식보다 유리한지 살펴보았습니다. 오픈소스 솔루션보다 뚜렷한 이점을 가지고 있음에도 불구하고 IBM Worklight Developer Edition은 무료입니다. 또한 전사적 모바일 애플리케이션 플랫폼이 필요할 경우 기업용 솔루션인 B2C/B2E 버전도 준비되어 있으니, 장기적인 안목으로 IBM Worklight Platform을 검토하시길 바랍니다.

바로 지금 Worklight Developer Edition을 다운로드하시어 직접 사용해 보시기 바랍니다.

부록

IBM Worklight Developer Edition 와 "DIY(do-it-yourself)" 방식의 기능 비교

범주	기능	하위 기능	기능 지원 (Y/N)		범주 순위 별표 4개 = 고급 별표 1개 = 기본		설명
			Worklight	DIY	Worklight	DIY	
대상 플랫폼					****	****	두 솔루션 모두 Android SDK(또는 지원되는 경우 원하는 다른 플랫폼)를 설치하고 구성해야 합니다.
	Android 전화		Y	Y			
	Android 태블릿		Y	Y			
	웹 페이지 임베드		Y	Y			
	모바일 웹 앱		Y	Y			
	표준 앱		Y	Y			
애플리케이션 유형(채널)					****	***	Worklight는 기업용 앱을 개발하는 동시에 기술 세트 및 권한을 분류하는 데 도움이 되는 내부 쉘 접근 방식을 지원합니다.
	웹		Y	Y			
		HTML 5	Y	Y			
		HTML 5 단일 페이지(SPA)	Y	Y			
		기본 HTML	Y	Y			
	모바일 웹		Y	Y			
	하이브리드		Y	Y			
	네이티브		Y	Y			
	웹 사이트(일반)		Y	Y			
	쉘/내부 앱 접근 방식		Y	N			
	하이브리드에 컨테이너 사용		Y	N			
고유 API 지원					****	****	설치된 SDK에 따라 다름
	Android		Y	Y			
도구 플랫폼					****	****	
	Windows		Y	Y			

범주	기능	하위 기능	기능 지원 (Y/N)		범주 순위 별표 4개 = 고급 별표 1개 = 기본		설명
			Worklight	DIY	Worklight	DIY	
	Mac OS X		Y	Y			
	Linux		Y	Y			
IDE					****	**	Worklight는 동일한 프로젝트 내에 여러 개의 대상을 지원하고 손쉬운 GUI 작성을 지원하여 개발자의 생산성을 높여줍니다.
	개발자 레벨		Y	Y			
	Eclipse 기반		Y	Y			
	대상 플랫폼 에뮬레이터		Y	Y			
	통합 디버그		Y	Y			
	단일 프로젝트로 여러 개의 대상 지원		Y	N			
	드래그앤 드롭 GUI 빌더		Y	N			
	Cordova (PhoneGap) 사용 가능		Y	Y			
	버전 관리		Y	Y			Eclipse와 통합되는 타사 도구
	완전한 WYSIWYG		Y	N			DIY방식은 HTML 드래그앤드롭 비주얼 편집기를 제공하지 않지만 (미리 보기는 제공), Java UI 요소에 대해서는 드래그앤드롭 비주얼 편집기를 제공합니다(Android 네이티브 앱).
	OOTB 템플리트		Y	Y			DIY: 플랫폼 SDK에서 사용 가능한 템플릿에 따라 달라집니다.
	코드 재사용		Y	N			Worklight는 플랫폼을 즉시 추가할 수 있으며, 일반 코드와 플랫폼별 코드를 분리할 수 있습니다.
언어					****	****	
	Java		Y	Y			
	JavaScript		Y	Y			

범주	기능	하위 기능	기능 지원 (Y/N)		범주 순위 별표 4개 = 고급 별표 1개 = 기본		설명
			Worklight	DIY	Worklight	DIY	
스크립트 라이브러리					****	***	DIY: 각 라이브러리를 설치해야 합니다. Worklight: 일부 라이브러리가 도구에 사전 패키징되어 있습니다.
	jQuery		Y	Y			Worklight에 포함됨, DIY에 별도 설치
	Dojo Mobile		Y	Y			Worklight에 포함됨, DIY에 별도 설치
	Sencha Touch		Y	Y			별도 설치
	JavaScript로 작성된 플랫폼 라이브러리가 포함됨		Y	N			Worklight는 개발자 생산성을 높이고 플랫폼에 상관없이 앱을 빠르고 쉽게 구현할 수 있도록 JavaScript 라이브러리를 제공합니다.
모바일 타사 프레임워크					****	***	
	JQuery Mobile		Y	Y			별도로 설치해야 합니다.
	Dojo Mobile		Y	Y			별도로 설치해야 합니다.
	Sencha Touch		Y	Y			별도로 설치해야 합니다.
	Node.js		Y	Y			별도로 설치해야 합니다.
	Cordova (PhoneGap)		Y	Y			Cordova는 Worklight에 패키징되어 있으며, DIY 방식에서는 별도로 설치해야 합니다.
시뮬레이션					****	*	Worklight는 개발자 테스트에 대한 로드와 모바일 브라우저 시뮬레이터와 플랫폼 에뮬레이터를 지원합니다.
	브라우저를 통해		Y	N			
	디바이스 인식			N			
	Cordova 시뮬레이션		Y	N			
	디바이스 SDK 에뮬레이터 사용		Y	Y			

범주	기능	하위 기능	기능 지원 (Y/N)		범주 순위 별표 4개 = 고급 별표 1개 = 기본		설명
			Worklight	DIY	Worklight	DIY	
백엔드 연결					****	*	DIY는 타사 라이브러리를 사용하여 백엔드 서비스에 액세스해야 하며, 개발자가 직접 구현해야 합니다.
	직접 어댑터/ 커넥터						
		SAP	Y	N			
		HTTP	Y	N			
		JDBC	Y	N			
		SOAP 웹 서비스	Y	N			
		REST XML	Y	N			
		REST JSON	Y	N			
		JMS/MQ	Y	N			
		데이터베이스	Y	N			
		Cast Iron	Y	N			
		MQTT	Y	N			
	코드가 없는 백엔드 서비스 만들기		Y	N			
	백엔드 서비스용 시뮬레이터		Y	N			
	SMS 지원		Y	N			
	미들웨어 통합			N			
		IIB(WMB)	Y	N			
		MQ	Y	N			
		Cast Iron	Y	N			
		BPM 통합	Y	N			
평가판 또는 무료 개발 라이센스			Y	Y	****	****	

추가 정보

IBM Worklight Developer Edition 6.1에 대한 자세한 내용은 IBM 영업대표 또는 IBM 비즈니스 파트너에게 문의하거나 다음 웹사이트를 방문하시기 바랍니다. **ibm.com**/developerworks/mobile/worklight/

추가적으로, IBM Global Financing은 가장 비용 효율적 방법과 전략적 방법으로 비즈니스에서 필요로 하는 소프트웨어 기능을 취득할 수 있도록 지원합니다. IBM은 신용 있는 고객과 협력하여 귀사의 비즈니스 및 개발 목표에 적합하고 효과적인 현금 관리를 가능하게 하며 귀사의 총소유 비용을 개선하는 맞춤형 재무 솔루션을 제공합니다. IBM Global Financing으로 중대한 IT 투자에 자본을 투입하고 귀사의 비즈니스를 발전시키십시오. 자세한 정보는 다음 웹사이트를 참조하시기 바랍니다.
ibm.com/kr/financing

저자 정보

Ricardo Balduino

Ricardo Balduino는 IBM Competitive Project Office에서 근무중인 수석 소프트웨어 엔지니어로, 전문 분야는 소프트웨어 개발 주기 프로세스와 도구입니다. 그의 목표는 조직의 리더들이 우수한 품질의 소프트웨어를 개발할 수 있도록 최적의 도구와 방법론을 선택하도록 지원하는 것입니다.

© Copyright IBM Corporation 2014

IBM Corporation
Software Group
Route 100
Somers, NY 10589

Produced in the United States of America
2014년 4월

IBM, IBM 로고, ibm.com, Worklight 및 WebSphere는 전 세계 여러 국가에 등록된 International Business Machines Corporation의 상표입니다. 기타 제품 및 서비스 이름은 IBM 또는 타사의 상표일 수 있습니다. 현재 IBM 상표 목록은 웹 "저작권 및 상표 정보" (ibm.com/legal/copytrade.shtml)에 있습니다.

Java 및 모든 Java 기반 상표 및 로고는 Oracle 및/또는 해당 자회사의 상표 또는 등록상표입니다.

이 문서는 처음 발행될 당시의 날짜를 기준으로 업데이트되었으며 IBM은 언제든지 문서 내용을 변경할 수 있습니다. 일부 오퍼링은 IBM 지사가 있는 국가에서 제공되지 않을 수 있습니다.

IBM 제품 및 프로그램과 함께 사용되는 기타 제품 또는 프로그램을 평가 및 검증하는 것은 사용자의 책임입니다.

Please Recycle

WSW14226-KRKO-00