developer Works

유니버설 미들웨어 프레임워크 – OSGi

OSGi 개발 환경 구현 1 – J2ME & OSGi

OSGi가 가지고 있는 특성상 OSGi 개발 환경을 구축하기에 앞서 자바 환경에 대한 이해가 필요하다. 자바는 자바 애플리케이션이 동작하는 하드웨어나 디바이스에 맞게 J2ME, J2SE, J2EE로 구분되어 있다. 이 가운데 특히 주목해야 하는 부분이 바로 J2ME이다. 지난 회에서도 이야기했듯이 OSGi는 기본적으로 홈 네트워크 환경이나 디바이스 컨트롤러로 많이 활용 및 발전되어 왔기에 임베디드 환경에 최적화된 미들웨어 프레임워크라 할 수 있다. 따라서 J2SE나 J2EE보다 임베디드 자바 환경에 맞는 J2ME 환경에서 작동되어 왔다.

1

연 재 순 서

1회 | 2007. 7 | 임베디드를 넘어 엔터프라이즈로!
2회 | 2007. 8 | OSGi 서비스와 활용 사례
3회 | 2007. 9 | OSGi 개발 환경 구현 1 – J2ME & OSGi
4회 | 2007. 10 | OSGi 개발 환경 구현 2 – OSGi Bundle 구현
5회 | 2007. 11 | 이클립스의 핵심, Equinox
6회 | 2007. 12 | 엔터프라이즈로의 확장, Spring-OSGi

김석우 suhgoo.kim@samsung.com | Polytech 전산학 석사. 현재 삼성전자 선행개발팀에 근무 중이며 센서 네트워크를 활용한 빌딩의 폐적제어 시스템을 개발하고 있다. 또한 OSGi 기반의 센서 네트워크 컨트롤러를 구상하고 있다.

여러분은 곧 J2ME의 이해가 OSGi 개발에 첫걸음임을 알게 될것이다. 임베디드, 모바일 솔루션의 폭발적인 성장에 따라서 CDC, CLDC로 구분된 J2E 플랫폼은 MSA(JSR248), MSA-A(JSR249) 그리고 OSGi의 다이내믹 기능을 전폭적으로 수용한 MOM(JSR232), MJSP 등으로 끊임없이 발전하고 있다. J2ME에 대한 자세한 설명은 썬의 개발자 사이트를 활용하길 바라며 여기서는 J2ME의 기본 개요와 이해 그리고 OSGi 개발 환경 셋업에 중점을 두고 설명하기로 한다.

OSGi 개발 환경 설치에 앞서

OSGi는 여러 가지 버전이 있는데 여기서는 IBM 솔루션을 사용하기로 한다. OSGi라는 말 대신에 솔루션이라는 말을 사용한 것은 단순히 OSGi 하나의 툴을 설치하는 것이 아니라 이클립스 개발 환경 및 도구, 자바 런타임 모듈, JVM과 SDK 등을 모두 합한 개념으로 접근해야 하기 때문이다. IBM은 J9이라는 훌륭한 VM을 보유했을 뿐 아니라 썬과 더불어 웹스피어라는 강력한 솔루션을 가지고 있다. 또한 이클립스 역시 IBM에서 태동된 오픈소스인 만큼 현재 대부분의 IBM 솔루션들은 이클립스를 기반으로 해서 구현되고 있다. 이런 점을 바탕으로 많은 회사들에서 사용되고 검증된 IBM 솔루션을 통해 OSGi를 구현하고 경험해 보기로 한다. 과거에는 IBM의 OSGi 솔루션 SMF를 설치하기 위해 VM(J9)과 SMF, 개발툴 WSDD를 별도로 설치해야 했지만, 이클립스에 OSGi가 탑재된 이후부터는 이클립스를 통해 간단하게 인스톨할 수 있게 되었다. 따라서 먼저 이클립스를 다운로드해 설치해야만 한다. 이클립스를 설치했다면 이제부터 OSGi 개발 환경을 구축해 보자(현재 IBM은 SMF를 별도의 모듈이 아닌 Lotus Expeditor 솔루션을 통해 공급하고 있다. 그러나 여전히 별도의 런타임 모듈로 설치해 개발할 수 있으며 여기서 우리는 데모 버전을 통해 실습하고자 한다).

이클립스 자바 개발 환경(JDT) 설치

OSGi가 자바 환경에서 구동되기에 자바 개발 환경(JDT)은 기본적으로 설치해야 한다.

이클립스 메뉴 중에서 Help → Software Updates → Find and Install을 찾는다.

다운로드 사이트를 지정하고 다음 단계로 넘어간다. 여기서는 'Callisto Discovery Site'를 지정했지만, 다른 미러사이트를 정해도 무방하다.

〈화면 1〉 Eclipse의 Install/Update 화면

〈화면 2〉 Eclipse에서의 JDT 관련 모듈 선택

해당 사이트를 확장한 후에 다음의 항목들을 다운로드해 플러그인한다.

● Graphical Editors and Frameworks

- Visual Editor
- Graphical Editing Framework

● Java Development

- Visual Editor

● Models and Model Development

- Eclipse Modeling Framework

J9 SDK 플러그인 설치

자바 개발 환경을(JDT)를 설치했다면 이제 본격적으로 들어가 보자. 먼저 해야 할 것은 VM 설치이다. IBM은 J9이라는 강력한 VM과 SDK를 제공하는데 다음의 주소에서 찾을 수 있다. 이클립스를 잠시 닫아 두고 해당 사이트로 이동한다.

http://wiki.eclipse.org/index.php/J9

J9의 위키(Wiki) 사이트로 이를 천천히 읽어보기를 바라며, 가장 마지막의 'the JDT Debug Page'를 클릭한다(http://www.eclipse.org/eclipse/debug/j9/j9_launching.php).

〈화면 3〉 J9 WiKi 웹사이트 화면

〈화면 4〉 J9 JRE 다운로드

해당 페이지에서 Eclipse J9 JRE Support를 클릭해 받은 JAR 파일을 Eclipse Plugin folder에 삽입한다.

J9 Java Runtime Environments 설치

이제는 J9의 런타임 환경을 설치할 차례이다. 다음의 사이트에서 자신의 환경(윈도우나 리눅스 등)에 맞는 런타임 모듈을 다운로드한다.

http://www-128.ibm.com/developerworks/websphere/zones/

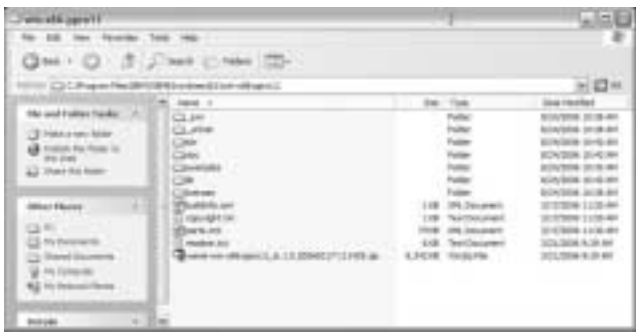
wireless/weme_eval_runtimes.html

〈화면 5〉 J9 Java Runtime Environments 다운로드(WSEE)

여기서는 윈도우(x86) 개발 환경을 기준으로 설명한다. 해당 페이지로 넘어가면 IBM의 웹스피어 솔루션 다운로드 사이트로 이동하는데, 타깃 디바이스의 플랫폼과 하드웨어 사양에 따라 런타임 모듈을 정한 후 다운로드해야 한다. 또한 해당 디바이스에 운영되는 J2ME의 스펙(SPEC)이 어떤가도 살펴봐야 한다. 즉, 타깃 디바이스의 자바 환경이 J2ME의 CDC 기반인지 CLDC 기반인지, 그리고 프로파일도 F/P, P/P, MIDP 등의 기반 환경과 프로파일의 스펙을 고려해 선택하고 다운로드한다. 여기서 말한 타깃 디바이스란 지금 개발하는 PC가 아니라 OSGi 기반으로 작동되는 디바이스를 지칭한다. 일반적으로 모바일 환경에서 PDA나 휴대폰, 컨트롤러들을 지칭한다.

〈화면 6〉 타깃 디바이스 런타임 환경 선택

타깃 디바이스의 운영 환경과 스펙을 결정하고 런타임 환경을 인스톨하면 최종적으로 Zip이 작성되는데(예: weme-win-x86-ppro11_6.1.0.20060317-111429.zip) 해당 파일은 타깃 디바이스로 복사하기 전에 풀어서 내용 파일들을 그대로 복사해 주면 된다.



〈화면 7〉 타깃 디바이스에 맞는 런타임 모듈이 Zip 파일로 생성

이클립스에서 J9 런타임 환경 셋업

이제 이클립스에서 PC에 설치된 J9과 런타임 환경을 셋업하기로 한다. 이클립스 메뉴에서 Window → Preferences… → Installed JREs를 선택한다.

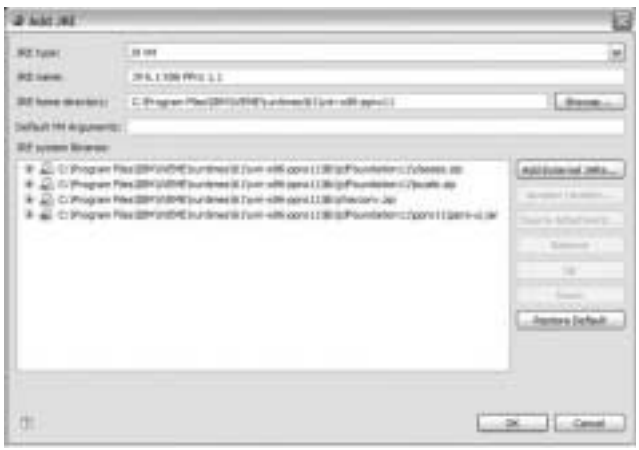
〈화면 8〉 다양한 J9 JRE 설정

설치된 JREs properties의 오른편 Add를 눌러 환경을 셋업한다.

〈화면 9〉 J9 프로퍼티 설정

JRE Type에는 'J9 VM' 이라고 적고 'JRE Home Directory' 필드에는 좀 전에 설치했던 런타임 모듈의 디렉토리를 정한다 (예: C:\Program Files\IBM\WEME\61\runtimes\61\win-x86-ppro11).

'JRE System Libraries' 필드에는 default Java Class Libraries 패스를 설정하고, 또한 앞으로 구현할 타깃 디바이스의 프로파일에 대한 패스도 함께 정의한다. 예를 들어 Personal Profile Applications를 구현할 거라면 external JAR reference로 'ppro-ui.jar' 파일을 추가해야 한다.



〈화면 10〉 Class Libraries 패스 설정

이클립스에 J2SE, J2ME와 같은 다양한 JRE들을 설치했다면 앞으로 자바 프로젝트를 개발할 때는 기본 런타임 모듈을 어느 것으로 해야 할지 결정해야 한다. 만약 J2ME에 Personal Profile 1.1의 애플리케이션을 개발한다면 기본적인 JRE 타입은 'J9 PPro 1.1' 을 설정하면 된다.

〈화면 11〉 기본 J9의 환경 설정

이제 마지막으로 해야 하는 설정은 바로 컴파일러이다. 타깃 디바이스와 스펙에 맞게 JRE를 설정했다면 역시 그에 맞는 JDK Compiler Compliance Level도 설정해야만 한다. JDK Compiler Compliance Level은 'Compiler' 탭(tab)에서 설정할 수 있는데, JavaME CDC/Foundation 1.0 & Personal Profile 1.0 applications을 설정했다면 JDK Compiler Compliance Level은 버전 1.3으로 하고, 또한 JavaME CDC/Foundation 1.1 & Personal Profile 1.1 applications를 개발할 것이라면 버전 1.4로 설정해야 한다.

〈화면 12〉 Compiler Compliance Level 설정

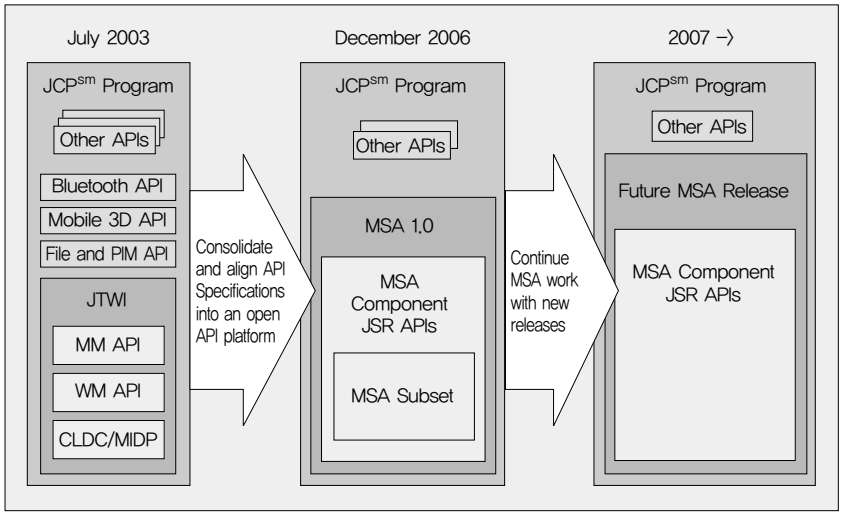
지금까지 이클립스에서 임베디드 자바 개발 환경을(J2ME) 설치했다. J2ME에 최적화된 개발 환경에서 실제로 OSGi 모듈이 탑재되고 운영되는 사례는 다음 시간에 살펴보도록 한다.

J2ME (Java 2 micro Edition)

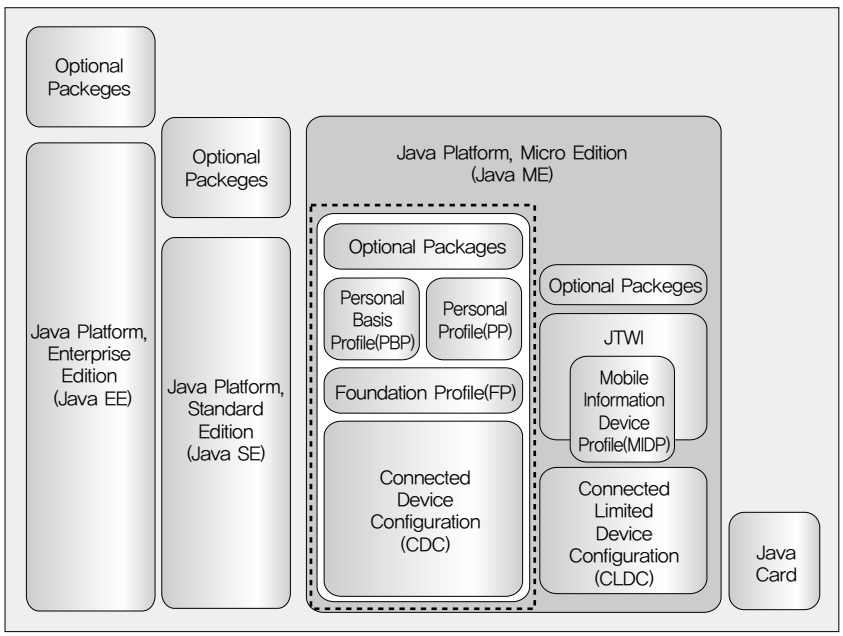
Java 2 플랫폼과 Micro Edition(J2ME)은 자바 플랫폼에서 가장 작은 부분으로, 스마트 카드나 호출기와 같은 매우 작은 장치에서부터 TV 세톱박스, 컴퓨터 데스크톱 등의 강력한 장치에 이르는 소비자용 내장형 장치에 사용된다.

J2ME의 주요 요소로는 소비자용 내장형 장치를 위한 자바 솔루션을 제공하는 여러 도구 및 기술뿐만 아니라 CDC(Con

〈그림 1〉 J2ME 플랫폼에서 운영되는 노키아 S60 GUI



〈그림 2〉 MSA로 발전해 나가는 J2ME의 CDC & CLDC 플랫폼



〈그림 3〉 Java2 Platform 구성도

nected Device Configurations) & CLDC(Connected Limited Device Configurations)라 불리는 핵심 API 셋, 각 디바이스 제품군에 맞는 API 셋 스펙을 정의한 프로파일인 F/P(Foundation Profile), P/P(Personal Profile), MIDP(Mobile Information Device Profile) 등으로 구성된다. 특히 소비자 가전(CE) 제품용으로 최적화된 Java Runtime Environment도 여기에 포함된다.

이렇듯 J2ME 기술은 다양한 범위의 극소형 제품에 사용되며 스마트카드, 호출기, 세톱박스, 그리고 기타 소형 제품 내에서 보안이나 연결 또는 유용한 유틸리티 프로그램을 사용 가능케 한다. 결국 J2ME 기술은 자바 제품군의 일부인데 관련된 자바 플랫폼에는 J2SE(Java 2 Standard Edition)와 J2EE(Java 2 Enterprise Edition)가 있다. J2ME 플랫폼은 실행 및 개발 환경으로 CVM & KVM, 애플리케이션 프로그래밍 API 라이브러리, Deployment와 Configuration tool 등을 제공한다.

J2ME 플랫폼이 지향하는 제품군들을 두 개의 그룹으로 나누면 첫 번째로는 개인 휴대가 가능하며 네트워크를 통한 정보전달이 가능한 셀룰러폰, 페이저 등의 디바이스들을 말할 수 있으며(KVM-CLDC-MIDP), 두 번째로는 구체적인 기능을 가지고 고정되어 쓰이는 정보전달 기기군인 세톱박스, 인터넷 TV, 자동차 내비게이션 시스템 등(CVM-CDC-F&PP)을 꼽을 수 있다.

셀룰러폰이나 페이저 등은 제품의 모양과 성능, 특성이 제각각이기 때문에 각각의 기계장치들의 공통적인 핵심 부분을 모아 버추얼 머신의 Configuration과 API들을 제공하고 있다. 실행환경 측면에서의 J2ME Configuration은 메모리 용량과 프로세스의 처리 용량이 비슷한 것들을 같은 범주로 하는 API들로 CLDC를 정의하고 있다.

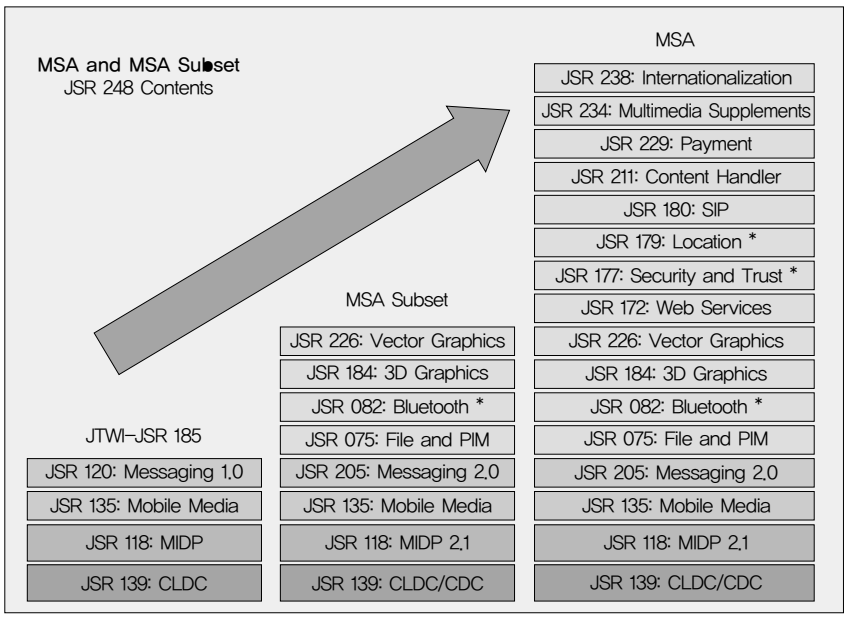
Configuration의 스펙을 살펴보면, 프로그램 언어인 자바, 자바 버추얼 머신 실행 환경, 개발 도구로서의 자바 라이브러리와 API를 제공하고 있다. 하이-레벨 아키텍처는 기계장치의 운영체제 위에 버추얼 머신이 존재하고, 그 위에 Configuration과 프로파일이 수직적 구조로 이뤄져 있다.

현재 J2ME 플랫폼에서는 CLDC와 CDC라는 두 가지 표준 Configuration이 있으나 작년에 열린 '자바원 2006'에서 발표된 향후 J2ME의 개발 로드맵은 MSA(Mobile Service Architecture) 1.0으로의 발전을 전망하고 있다(JSR 248, 249).

J2ME 플랫폼은 사용되는 기계장치별로 자바 플랫폼을 정의하는 프로파일을 제공함으로써 개발과 실행을 가능케 한다. 프로파일은 Configuration 위에 위치하는 기계장치별로 구분되는 API들로, 현재는 MIDP(Mobile Information Device Profile)만이 제공되어 실행되며 차후에 PDA 프로파일, RMI 프로파일 등이 포함될 것이다. 썬은 JCP(자바에 대한 라이선스와 표준을 관장하는 기구, Java Community Process의 약자)를 만들면서 전

체 자바 구조를 새로운 기틀을 바탕으로 재편성했다. 다시 말해 업무 및 적용 영역에 따라 다음과 같은 세 가지 섹션으로 나눈 것이다 .

- J2SE : 업무용 서버/시스템 전용
- J2EE : 표준적인 구성으로 데스크톱, 워크스테이션 전용
- J2ME : 적은 자원으로 동작하는 편입 기기용



〈그림 4〉 MSA 주요 제공 기능

J2ME 플랫폼은 J2SE나 J2EE에 비해 가장 늦게(?) 스펙이 정의되었으나, 실제로는 그 이전부터 PersonalJava, PICOJava, eJava 등의 솔루션으로부터 발전되어 왔던 개념이다. 자바의 탄생 배경이 가전제품의 표준화된 제어와 네트워크 솔루션이라고 한다면 어쩌면 J2ME는 자바의 목적에 가장 잘 맞는 규격인지도 모르겠다.

21세기에 들어서면서 급격하게 임베디드, 모바일 기술이 발전하고 있고, 이른바 유비쿼터스 환경이 도래하면서 J2ME 플랫폼이 휴대폰, MP3 플레이어, 디지털 카메라, 차량용 내비게이션, 홈 게이트웨이, 그 밖의 각종 디바이스 컨트롤러 등에서 많은 수요를 창출하고 있다. 또한 J2ME 환경의 인기 급상승과 더불어 OSGi의 수요도 급증하고 있다는 사실은 매우 고무적인 일이 아닐 수 없다.