developer Works

유니버설 미들웨어 프레임워크 – OSGi

임베디드를 넘어 엔터프라이즈로!

OSGi는 운영체제, 플랫폼에 독립적으로 운영되는 미들웨어 프레임워크이다. 또한 표준화된 스펙, 컴포넌트 구조 그리고 분산 네트워크 서비스에 최적화된 컴퓨팅 환경을 제공해주는 서비스 플랫폼이기도 하다. 이러한 특성으로 최초의 적용 분야인 모바일과 임베디드 시스템을 넘어 클라이언트/서버와 데스크탑 애플리케이션의 RCP(Rich Client Platform), 나아가 엔터프라이즈 환경의 프레임워크(Spring-OSGi)에 이르기까지 점차 그 영역을 넓혀 가고 있다.

2

연 재 순 서
1회 | 2007. 7 | 임베디드를 넘어 엔터프라이즈로!
2회 | 2007. 8 | OSGi의 개발 환경 구현
3회 | 2007. 9 | 임베디드 환경에서의 OSGi
4회 | 2007. 10 | 이클립스의 핵심, Equinox
5회 | 2007. 11 | 엔터프라이즈로의 확장, Spring-OSGi
6회 | 2007. 12 | OSGi 베스트 프랙티스 소개

김석우 suhgoo.kim@samsung.com | Polytech 전산학 석사. 뉴욕의 IBM 연구소를 거쳐 현재 삼성전자 선행개발팀에 근무 중이다. 빌딩 컨트롤 네트워크(Building Control Network)와 임베디드 시스템이 주요 개발 분야로, 현재는 OSGi 기반의 WSN 컨트롤러를 구상하고 있다.

무엇이 모바일/임베디드 환경의 OSGi를 엔터프라이즈까지 확장시켰을까? 이제부터 유니버설 미들웨어 서비스 플랫폼(Universal Middleware Service Platform)으로 자리매김한 OSGi에 대해 알아본다.

OSGi 탄생과 배경

OSGi는 Open Service Gateway Initiatives의 줄임말로 일종의 미들웨어 프레임워크(Middleware Framework)라 할 수 있다. 또한 OSGi는 기본적으로 자바 환경에서 구현되며 자바를 위한 다이내믹 모듈 시스템(Dynamic Module System이라고도 불린다.

이런 OSGi가 탄생하게 된 배경은 다음과 같다. 지난 1990년대 중반에 시작된 인터넷의 폭발적인 성장세는 IT뿐만 아니라 우리 생활 곳곳의 많은 부분을 변화시켰다. 이러한 변화 가운데 새롭게 각광받은 분야가 바로 홈 네트워크이다. 홈 네트워크는 기존의 전통적인 가정과 주거환경을 인터넷과 결합시켜 우리에게 보다 더 편리하고 쾌적한 주거 및 가정환경을 제공해줄 것으로 기대됐고, 이에 따라 많은 사람과 기업이 주목하기 시작했다. 또한 수많은 기업들이 홈 네트워크에 대규모 투자를 감행했고, 여기에는 마침내 기존 IT 기업뿐만 아니라 통신, 가전, 건설, 환경 업체들까지 뛰어들게 됐다. 하지만 이렇게 많은 업체들이 홈 네트워크에 뛰어들면서 인터넷에 연결되는 정보 및 가전기기들의 호환성 문제가 대두된다.

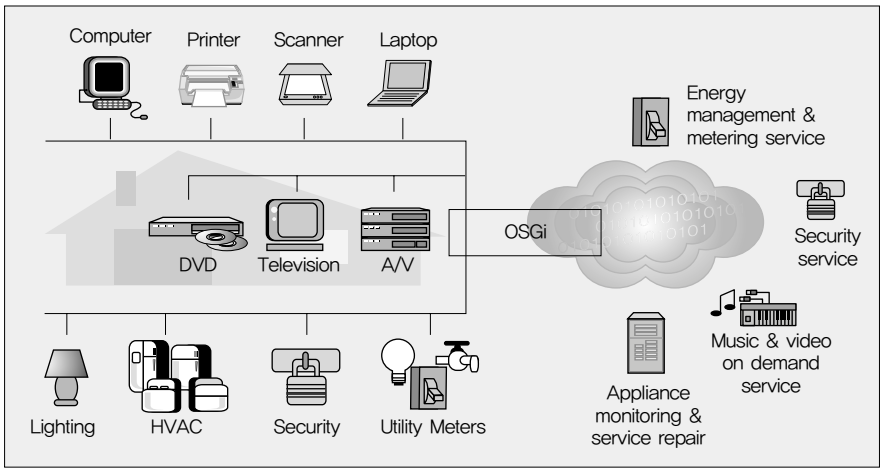
이에 IT, 가전, 통신, 주택, 환경 분야를 대표하는 업체들이 모

OSGi 얼라이언스

1999년 3월 설립된 비영리 단체로 OSGi 공개 표준의 스펙을 제정하는 것이 주 활동이다. 2000년 5월에 OSGi 1.0을 릴리즈(Release)한 후 4.x까지 제정되어 있다. 주요 멤버를 분야별로 살펴보면 통신 분야에서는 노키아, 모토롤라, 도이치텔레콤, 프랑스텔레콤, KT, 보다폰 등이 참가했고, 가전 분야에서는 삼성전자와 필립스, 월풀, 지멘스, 샤프, 도시바, 히타치 등이 회원으로 가입해 있다. 또한 IT 분야의 IBM, 썬, 인텔, 오라클, HP, ETRI, Prosyst와 BMW 등의 자동차 제조사 외에 현재 80여개 기업이 이 단체에 가입되어 있다.

한편 OSGi 얼라이언스는 네 개의 전문가 그룹으로 구성되어 각 해당 분야의 요구 사항을 분석하고 그에 따른 스펙을 제정하고 있다. 그 그룹을 차례로 나열하면 CPEG(Core Platform Expert Group), VEG(Vehicle Expert Group), MEG(Mobile Expert Group), EEG(Enterprise Expert Group)이다.

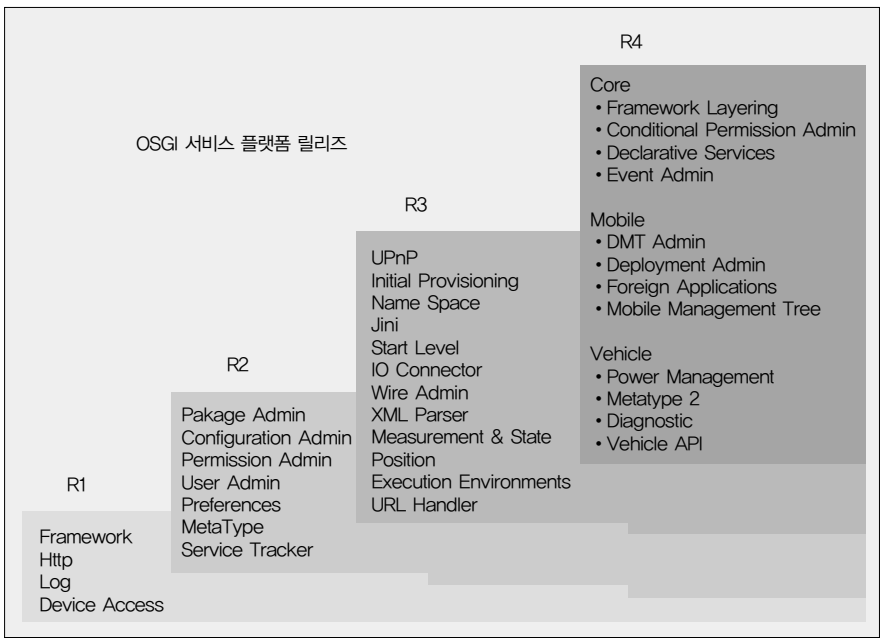
여 로컬 네트워크(Local Network)상에서 상호 호환성을 보장하고, 각 디바이스에서 관리되는 서비스들의 배포 및 공유에 대한 공개 스펙을 제정하게 된다. 이러한 배경을 가지고 탄생한 것이 바로 OSGi 기술이다. OSGi는 스펙이 공개된 기술로, 모든 개발 및 스펙 제정 활동은 OSGi 얼라이언스(Alliance)에 의해 승인되고 진행된다. 국내에서는 삼성전자가 OSGi 얼라이언스 창립 때부터 주요 이사로 참가해 활동해 왔고, 현재는 KT와 ETRI도 참여하고 있다.



〈그림 1〉 OSGi와 홈 솔루션 서비스 구성도

현재 홈 네트워크의 기능은 크게 가전기기의 상태 정보/모니터링, 기기의 원격제어 컨트롤, A/V 및 주방 가전의 홈 솔루션 통합으로 구분하고 있는데, 가정의 네트워크에 속한 단말이나 가전에 접근하기 위해서는 디바이스에 대한 표준화된 네트워킹(이더넷, 블루투스, 무선 LAN, IEEE1394, PLC)과 범용 미들웨어(Universal Middleware)가 필요하다. 이런 이유에서 나온 것이 바로 UPnP, HAVi, JINI 등의 표준이다.

〈그림 1〉은 이러한 장비 연결 및 제어로 얻을 수 있는 유효 서비스들을 나타낸 것이다. 이러한 서비스의 배포 문제를 해결하고 서비스가 작동하기 위한 제반 환경을 제공하는 것이 바로 OSGi의 기본 목표라고 할 수 있다.



〈그림 2〉 OSGi – 릴리즈 버전

초기의 OSGi R1.0에서는 기초적인 정보기기의 연동과 상태 모니터링 기능 탑재, 그리고 표준화에 주력했다. 이어서 2.0은 운영과 관리, 보안 기능 등을 추가했다. 본격적인 컨텐츠 서비스 플랫폼으로의 모습은 3.0에 이르러 등장한다. 외부의 오픈소스를 이용해 처리하던 XML 파싱, Wire Admin, URL Handler 등의 기능이 표준 서비스로 채택되었고, UPnP와 JINI 표준이 기본 시스템 서비스로 올라오면서 OSGi는 명실상부하게 모바일, 임베디드, 데스크탑 애플리케이션, 클라이언트/서버 환경을 모두 아우르는 미들웨어 프레임워크로 자리매김한다.

4.0에 이르러서는 좀 더 분야가 세분화되면서 카테고리별로 디바이스 특성에 맞는 컨텐츠와 시스템 서비스들이 발전하게 된다. 특히 휴대폰과 스마트홈 컨트롤러, 빌딩 자동화 컨트롤러, 자동차용 내비게이션과 텔레매틱스 솔루션 등의 폭발적인 성장에 힘입어 모바일, 임베디드 시스템에 대한 많은 요구 사항들이 직접 기본 서비스에 선택되어 탑재된다.

여기서 나오는 시스템 기본 서비스는 OSGi 얼라이언스에서 승인해 기본 OSGi 프레임워크에 탑재된다. 그러나 여기에 빠져 있더라도 벤더가 자신들이 원하는 기능을 OSGi 스펙에 맞춰 개발하면 얼마든지 OSGi 프레임워크에 탑재해 사용할 수 있다. 이를테면 노키아에서 개발한 응용 서비스 번들과 서비스 등은 모토롤라나 소니에릭슨의 휴대폰에 탑재해 사용할 수 있다.

OSGi 특징

OSGi는 서비스가 작동하고 운영되는 서비스 환경에 관한 표

준 기술이다. 과거 OSGi R1.0이 발표될 때만 하더라도 OSGi는 홈 네트워크의 네트워크 표준에 국한된 부분이 많았으나, 지금의 4.0에 이르러서는 기존의 홈 네트워크와 모바일, 임베디드, 텔래매틱스, PC 애플리케이션(이클립스 기반의 RCP)은 물론이고, 엔터프라이즈 환경의 프레임워크에까지 확장되고 있다. 그만큼 기존의 미들웨어에 비해 많은 장점을 가지고 있는 셈이다.

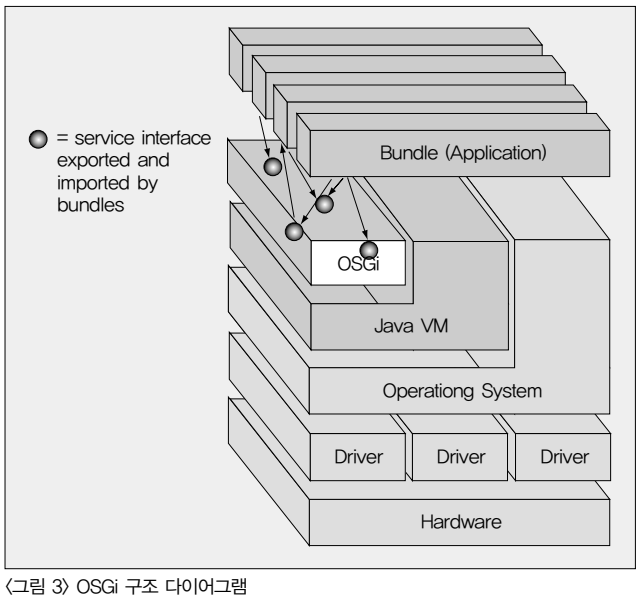
또한 과거 초기 단계의 홈 네트워킹 및 빌딩 제어, 자동화 시장에는 디바이스 제어 기술이 주로 쓰였지만 현재는 그보다 훨씬 다양한 디바이스간의 상호작용을 요구하고 있고, 나아가 컨텐츠 및 솔루션 서비스가 점차 고도됨에 따라 OSGi와 같은 서비스 플랫폼 환경이 꼭 필요하게 될 것이다.

현재 다양한 디바이스들간의 상호 운용성을 위해 UPnP, HAVi, JINI 같은 표준이 만들어져 주로 장비의 제어나 데이터 전달 등을 처리하므로, 이는 OSGi 기술과 상호 보완적인 위치에 있다고 할 수 있다. OSGi를 사용하거나 구현하는 쪽에서는 이러한 미들웨어 지원이 또 다른 고려 사항임을 유의해야 한다. 그럼 OSGi의 주요 특징을 살펴보자.

S/W Component Management

OSGi는 자바 기반의 컴포넌트(Component) 구조로 설계되어 있다. 따라서 OSGi는 자바 런타임 환경하(J2ME, J2SE, J2EE)에서 작동하도록 만들어진 표준이다. 자바 VM은 이질적인 임베디드 운영체제와 임베디드 CPU에서 발생하는 차이점에 대한 완충 역할을 수행한다. 또한 OSGi는 서비스 기반의 구조를 지향한다. 서비스는 모두 번들(Bundle)이라 불리는 물리적 묶음에 포함된다. 복수 개의 OSGi 서비스가 하나의 번들에 포함될 수도 있으며, 번들은 배포와 관리의 기본 단위를 형성한다. 이 번들을 관리하는 것이 바로 프레임워크(Framework)이다. 프레임워크는 서비스에 대한 등록/관리기(Service Registry)를 가지고 있어 서비스에 대한 등록, 조회, 실행, 삭제 등을 수행한다.

또한 이벤트와 그에 따른 이벤트 탐지 및 대응도 처리한다. 여기서 말하는 이벤트는 장비에서 생성된 물리적 이벤트와는 관계가 없고 서비스와 번들, 프레임워크라는 세 가지 이벤트 산출자를 근간으로 하는 논리적 이벤트라는 점에 유의해야 한다. OSGi의 가장 기본적이고 중요한 요소들인 번들, 서비스, 프레임워크는 차후에 좀 더 자세히 설명한다.



〈그림 3〉 OSGi 구조 다이어그램

원격 컴포넌트 관리

OSGi는 원격관리를 지원한다. OSGi는 번들 단위로 서비스를 형성하고 운영하는데, 번들을 업데이트하거나 원격에서 업데이트를 관리 및 제어할 수 있다. 자바 VM(Virtual Machine)을 재부팅할 필요 없이 사용 중에 원격으로 업데이트(업그레이드)할 수 있는 것은 매우 큰 장점임에 틀림없다. 이를 위해 OSGi는 원격관리 표준 프로토콜을 제정했으므로, OSGi가 탑재되고 운영되는 환경에 맞춰 이용하면 된다.

원격관리 프로토콜로는 대표적으로 OMA-DM, SNMP, CMISE 그리고 Telnet/SSH 등이 사용된다. OSGi가 탑재된 컨트롤러나 디바이스가 인터넷이나 로컬 네트워크에서 운영된다면 주로 과거에는 SNMP, Telnet/SSH 등을 사용했다. 그러나 최근 무선 환경이 급격히 확산되고 모바일/무선 디바이스의 사용이 늘어나면서 새롭게 탄생한 프로토콜이 바로 OMA-DM (Open Mobile Alliance for Device Management)이다. 다시 말해 OMA-DM은 모바일 디바이스를 위한 유무선 통합 관리 프로토콜이다.

물리적인 레이어(Physical Layer)에서 유선모드라면 USB,

〈그림 4〉 필립스의 홈 컨트롤러 iPronto

RS-232C를 사용하고, 무선모드에서는 GSM, CDMA, IrDA, 블루투스를 사용해 데이터를 전송한다. 또한 전송 레이어(Transport Layer)에서는 HTTP, WAP, OBEX(Object Exchange)를 사용한다. 데이터 전송 언어로는 SyncML을 사용해 데이터 호환 및 규격을 통일했다.

마지막으로 Binary Transmission과 Session Management를 통해 SyncML의 텍스트 포맷(Text format)을 압축/암호화하고 HTTP/WAP의 세션을 관리한다. 필립스의 가정용 컨트롤러인 iPronto, 노키아의 스마트폰, BMW 5/6 시리즈에 탑재된 텔레매틱스 등이 OMA-DM을 활용해 컨텐츠 번들을 관리하고 업그레이드 하는 대표적인 제품들이다.

애플리케이션간의 협업

많은 자바 애플리케이션 서버 환경에서 구동하는 자바 애플리케이션들은 독립성을 보장하기 위해 극히 폐쇄적인 컨테이너 환경에서 작동된다. 따라서 다른 자바 애플리케이션과의 연동이나 통합이 이뤄지려면 라이브러리 코드를 각각 가져와 구동해야 하므로 오버헤드가 필연적으로 발생한다. 예를 들어 엔터프라이즈 환경에서 사용되는 JMS 서비스 API는 백앤드 서버 라이브러리에 위치하며 그 크기가 대략 30~40KB 정도다.

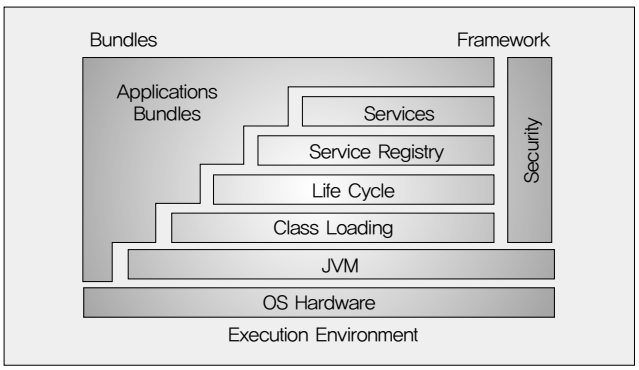
만약 MIDP 애플리케이션이 JMS를 이용해 메시징 서비스를 하려한다면, 백엔드 서버의 라이브러리 코드를 사용해야 한다. 이때 다수의 MIDP 애플리케이션이 구동된다면 n개의 JMS-API가 동작하는 오버헤드가 생긴다. 이런 문제점들을 해결하기 위해 나온 솔루션 서비스가 바로 SOA(Service Oriented Architecture)이다. OSGi는 이러한 SOA의 기본적인 구조, 즉 공통적으로 사용되기 위한 서비스 또는 라이브러리 API를 서버나 공간의 어느 디바이스에 등록해(Registry) 배포 및 공유하면(Contribute) 접근 가능한 어떤 애플리케이션이라도 사용할 수 있는 연결 구조를(Loosely Coupled) 지향한다.

특히 OSGi는 엔터프라이즈 서버 환경보다 매우 가볍고, 빠르게 접근해 바인딩할 수 있는 OSGi Framework Service Registry 구조를 가지고 있다. 이러한 OSGi 서비스간의 협업 구조는 리소스가 한정적인 모바일이나 임베디드 환경에서 매우 강력한 위력을 발휘했다. 현재는 데스크탑 애플리케이션에까지 그 영역이 확장되어 이클립스 기반의 RCP와 IBM Expeditor 솔루션으로 나타나고 있다.

아키텍처

OSGi 구조는 다음과 같이 몇 개의 계층으로 구성되어 있다. OSGi는 우선 자바 런타임 환경에서 구동된다. 자바 런타임은 하드웨어 환경에 의해 J2ME(CDC/CLDC), J2SE, J2EE로 구성되며, 하나의 VM에서 서로 다른 복수 개의 클래스 로더들에 의해 각각의 OSGi 애플리케이션을 실행한다. 자바 런타임 환경 위에서는 OSGi 프레임워크가 실행된다.

OSGi 프레임워크는 크게 번들의 실행주기(설치, 시작, 중단, 제거, 업데이트)와 OSGi 기본 실행 단위인 번들(Bundle)과 서비스에 대한 운영 관리, 이어서 리소스와 서비스 레지스트리 등을 담당하게 된다. 복수 개의 클래스 로더에 의해 각기 다른 OSGi 애플리케이션이 독립성을 가지고 실행되지만, OSGi Framework Service Registry에 등록된 Sharing Code와 Address에 의해 서로 다른 번들간의 리소스 공유와 연동/통합으로 무수히 많은 서비스들을 생성하고 실행할 것이다. 이러한 OSGi 프레임워크 구조는 JES(Java Embedded Server)에 의해 많은 영향을 받아 왔다.



〈그림 5〉 OSGi 아키텍처

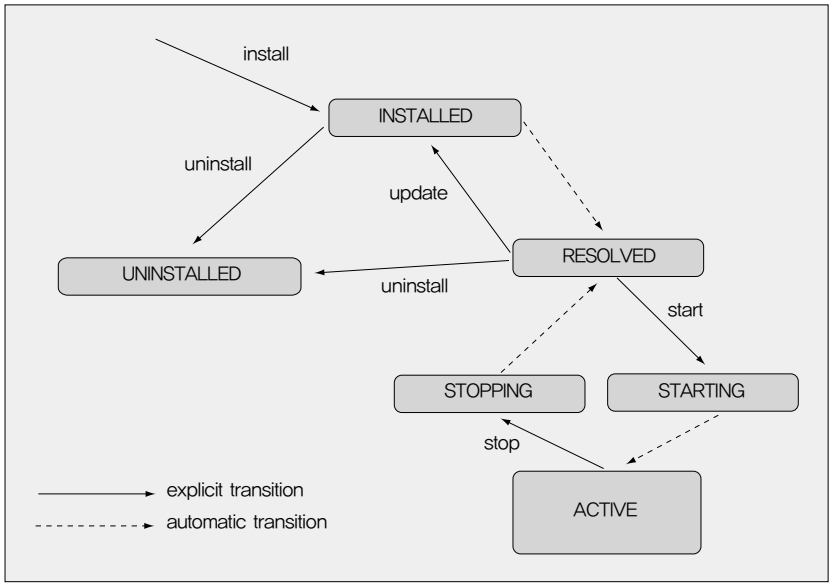
OSGi 핵심, 번들과 서비스

OSGi의 핵심으로 고려할 만한 것은 바로 번들과 서비스(Service)이다.

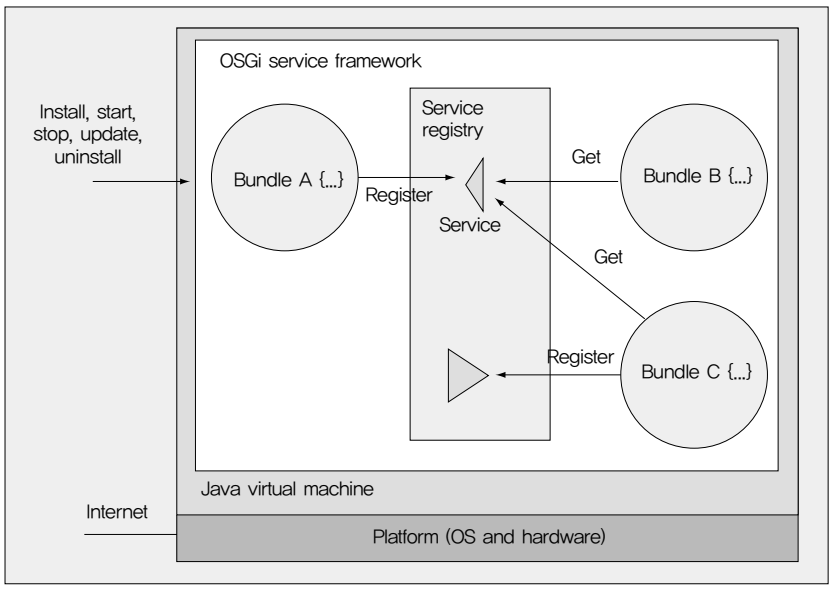
번들

OSGi의 가장 기본적인 실행 단위인 번들은 OSGi 프레임워크에서 수행되는 어떤 S/W 컴포넌트의 resources, 동작을 위한 Java classes, 번들 정보를 담고 있는 Manifest file, service를 포함하는 JAR 파일 등이다. OSGi는 단 하나의 VM 인스턴스 위에서 동작하고, 복수 개의 클래스 로더를 수행해 독립된 namespace를 가진다. 또한 번들은 동적 라이프 사이클(dynamic Install, Start, Stop, Delete, Update)을 갖고 수행하며(Without a JVM restart), 성능을 위해 여러 리소스와 인터페이스를 지원한다(Images, GUI look & Feel, Native-JNI).

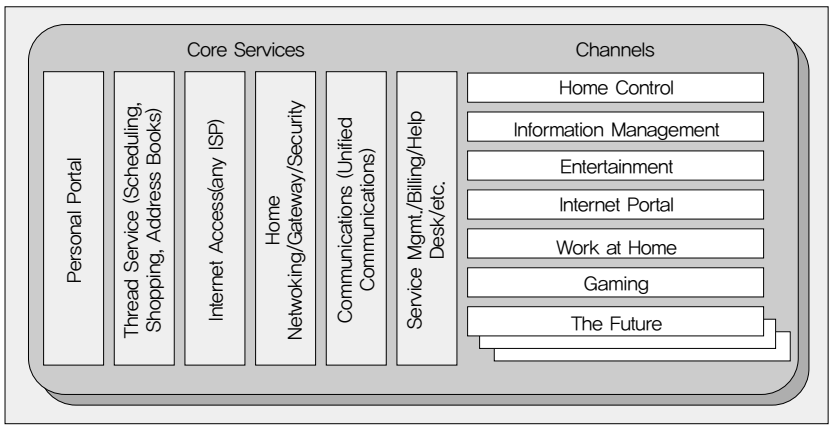
자바 애플릿처럼 서버에서 다운로드하는 게 아니라 로컬 디바이스에 상주하는 방식인 것도 하나의 특징이다. 〈그림 6〉은 번들



〈그림 6〉 OSGi 번들 – 다이내믹 라이프 사이클



〈그림 7〉 OSGi 번들과 서비스 생성



〈그림 8〉 다양한 OSGi 서비스 모델

의 동적 라이프 사이클을 나타낸다. 번들이 OSGi 프레임워크에 등록되면(Active Status) Bundle Context가 생성되는데, OSGi 프레임워크와 번들 사이의 인터페이스 역할을 수행하거나 다른 서비스 검색 및 등록이 이뤄질 때 사용된다.

서비스

OSGi 서비스는 자바 오브젝트로서 하나의 번들에 의해 등록되며, 다른 번들에 의해 사용되거나 복수 개의 번들이 연동 및 통합해 독자적인 서비스를 만들기도 한다. 명시된 형태는 자바 인터페이스 스타일로 표현되며 서비스 스펙과 구현이 완벽하게 분리되어 사용된다. 이렇게 되면 서로 다른 번들이 동일한 서비스 기능을 가졌더라도 서로 다른 구현(Implementation)을 서비스 레지스트리에 등록할 수 있다.

지금까지 OSGi의 기본적인 개요와 탄생 배경에 대해 살펴봤다. OSGi의 기본적인 개념은 새로운 개념이 아니지만, 하루가 다르게 탄생하는 새로운 플랫폼, 표준 그리고 프레임워크를 감안할 때 1999년부터 꾸준히 발전해온 OSGi가 분명 독특한 장점을 지녔음에 틀림없다. 그렇기에 계속해서 발전할 수 있고, 나아가 홈 네트워크용 서비스 플랫폼이나 모바일/임베디드 프레임워크 분야에서 점점 더 영향력을 확대해 나가는 것으로 생각된다.

특히 이클립스 플랫폼에 기본 탑재되고 RCP로의 확장에 핵심 요소로 지목되는 것은 OSGi가 단순한 미들웨어나 프레임워크가 아님을 증명하는 좋은 예일 것이다. 물론 OSGi의 단점이 전혀 없는 것은 아니다. 자바 환경에서 하나의 VM 인스턴스로 작동되는 것은 어떤 운영체제에 종속되지 않는 장점으로 이어지지만, 디바이스와 하드웨어에 영향을 많이 받는 모바일/임베디드 환경에서는 오히려 제한된 리소스와 성능에 대한 오버헤드로 나타날 수 있다.

그러나 문제점이 없는 서비스, 플랫폼, 솔루션이란 존재할 수 없고 단점을 하나둘 극복

해 장점을 더욱 강화해 나가는 것이 치열한 경쟁에서 살아남을 수 있는 유일한 길이라고 생각할 때, 기존의 모바일/임베디드 환경에서 RCP와 데스크탑 애플리케이션, 나아가 엔터프라이즈 환경의 프레임워크로까지 영역을 넓혀가는 OSGi야말로 이 연재의 제목에 표현된 것처럼 유니버설 미들웨어 플랫폼(또는 프레임워크)이라고 불려도 전혀 손색이 없을 것이다.

특히 디바이스-디바이스 M2M 환경이나 센서 네트워크, 로봇 등과 같은 새로운 솔루션과 서비스가 속속 등장하고 있는 걸 보면, 서비스를 더 쉽게 생성해 운영 및 관리할 수 있고 표준화된 스펙을 제공하는 OSGi가 미래에는 더 큰 영향력을 가지게 되리라 확신한다. 따라서 다가올 M2M이나 마이크로 센서 네트워크(Micro Sensor Networks) 시대에는 비로소 OSGi가 진정한 유니버설 미들웨어 프레임워크로 자리 잡게 될 것이다.

다음 호에서는 실제 OSGi 솔루션의 성공적인 활용 사례를 소개하고, 임베디드 개발 환경의 구축에 대해 설명한다.

참고 자료

1. OSGi Service Platform Core SPEC R4 - www.osgi.org
2. About the OSGi Service Platform - www.osgi.org
3. IBM - SMF (Service Management Framework) - www.ibm.com/software/wireless/smf
4. 'The Fundamental of OSGi' ? 한국산업기술대학교, 서대영 교수