developer Works

유니버설 미들웨어 프레임워크 – OSGi

이클립스의 핵심, Equinox

OSGi 프레임워크는 홈 게이트웨이의 임베디드 시스템을 위한 미들웨어 프레임워크로서 그 입지를 다지고, 이젠 그 영역을 플랫폼(Platform)으로 확장해 가고 있다. 이 글에서 소개될 Eclipse-Equinox야말로 그 영역 확장의 핵심. Eclipse-Equinox가 무엇이고, 어떤 특성들이 이런 포괄적인 영역 확장을 가능케 했는지 지금부터 살펴보기로 한다.

1

연 재 순 서

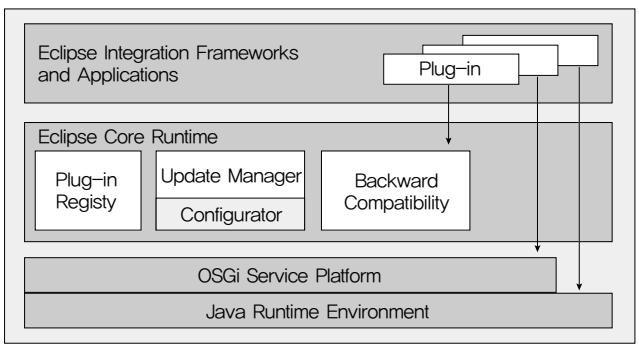
1회 | 2007. 7 | 임베디드를 넘어 엔터프라이즈로!
2회 | 2007. 8 | OSGi 서비스와 활용 사례
3회 | 2007. 9 | OSGi 개발 환경 구현 1 – J2ME & OSGi
4회 | 2007. 10 | OSGi 개발 환경 구현 2 – OSGi Bundle 구현
5회 | 2007. 11 | 이클립스의 핵심, Equinox
6회 | 2007. 12 | 엔터프라이즈로의 확장, Spring-OSGi

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

Equinox는 단순히 구현의 관점에서 본다면, OSGi R4 core framework specification을 근간으로 구현된 코드이다. 이러한 점은 여타의 다른 OSGi 제품들과 다를 바 없지만 이 글을 통해 곧 Equinox가 다른 OSGi와 구별되는 특성을 알게 된다. 그것은 바로 플랫폼으로 확장된 OSGi라는 점이다.

Equinox란?

Equinox는('이퀴넉스' 라고 발음) 기본적으로 다른 OSGi처럼 프레임워크, 번들(Bundle)이 존재하지만 또한 Incubator, Server-Side Project를 통한 다양한 서비스 아키텍처, 서버, 서버 프레임워크 모듈로 확장되어 이클립스 기반의 개발 플랫폼 일부분으로 사용된다. 혹시 IBM의 Expeditor, 어도비의 VersionCue, 래쇼널의 소프트웨어 개발 툴을 본 적이 있는가? 더 나아가 이를 사용해본 적이 있는가? 그들 모두 Eclipse-Equinox 기반의 플랫폼에 최적화된 콘텐츠와 기능들을 가지고 태어난 제품들이다. 마치 윈도우 기반의 애플리케이션과 시스템 아키텍처가 닷넷 프레임워크 위에서 COM 모듈로 쉽게 개발되고 하나의 고유한 프로세스와 GUI 위에서 운영되는 것처럼 이제 이클립스 플랫폼은 MS의 닷넷 프레임워크에 맞서 하나의 플랫폼으로 발전해 나가고 있으며 그 핵심에는 바로 Equinox OSGi가 자리 잡고 있다.



〈그림 1〉 Eclipse-Equinox 아키텍처

Equinox는 2004년 Eclipse Foundation의 요청에 의해 Eclipse 3.0에 탑재되어 세상에 처음으로 등장한다. 이때의 이클립스는 막 버전 2.1에서 3.0으로 업그레이드되었는데 단순히 1.0이 버전업되는 것보다 훨씬 더 많은 변화를 가져왔고, 그 변화의 중심에 Equinox 탑재가 있었다. 이전에는 수많은 기능들이나 모듈들이 플러그인 형태로 다운로드되거나 인스톨되었는데, OSGi 기반의 번들 형식으로 변경되어 재설계되고 Dynamic Module

시스템으로 제공된 것이다. 이미 2003년 초 Eclipse Foundation은 단순히 이클립스를 통합된 하나의 IDE가 아니라 RCP(Rich Client Platform)로의 확장을 목표로 전체적인 시스템 아키텍처의 재설계를 시도한다. 이때 바로 OSGi 프레임워크 기반의 구조가 논의 끝에 설계되었고, IBM의 SMF 3.x를 근간으로 이클립스에 최적화된 Equinox가 탄생하게 된다. 이클립스의 최적화 기간은 약 18개월 정도였으며 이때 하나의 프레임워크로 선택된 SMF-OSGi가 Equinox로 재탄생해 RCP, JDT, PDE 기반 이클립스 플랫폼의 핵심 컴포넌트로 탄생하게 되었다. 초기 Equinox는 OSGi R3를 기반으로 하고 있었으나 현재 R4 스펙으로 버전업되었고, 현재에도 OSGi Alliance의 VEG(Vehicle Expert Group), MEG(Mobile Expert Group)와 협의하에 지속적으로 발전해 나가고 있다. Eclipse Equinox Project에는 몇 가지 특징적인 요소들이 있었는데 프로젝트의 산실인 Incubator 활용, Open Source Community 지원, 그리고 마지막으로 신기술의 개념 도입 등이 그것이다. 우선 Equinox를 구현하기 위해 많은 OSGi 및 이클립스 오픈소스 단체(혹은 동호회)에서 개발 인력을 확보했고 스펙 선정 과정에서 요구사항 등을 수집했으며, 이러한 개발의 집합체 및 통로로 Incubator를 활용했다. Incubator는 말 그대로 '산실'의 의미를 가지는데, Equinox가 개발된 이후에도 여전히 Incubator를 통해 많은 서비스 번들과 프로젝트들이 수행되고 있다. 한편 신기술과 신개념 도입은 AOP(Aspect Oriented Programming)를 자바에 접목한 AspectJ와 Security 부분을 Incubator를 통해 동시에 추진했다. 이러한 특징들은 하나의 회사나 벤더가 독립적으로 개발할 때 나타나는 현상과는 다른 측면이다. 따라서 이런 움직임은 공동의 광범위한 오픈소스 지원과 빠르게 변하는 신기술에 적극적으로 대응하는 데 부합된다고 볼 수 있다.

이클립스의 선택

이클립스는 무엇 때문에 OSGi를 선택하게 되었을까? 또한 왜 이클립스는 2.1에서 3.0으로 급격히 변화해야 했고 당시에 어떤 고민을 가지고 있었는지 살펴볼 필요가 있다.

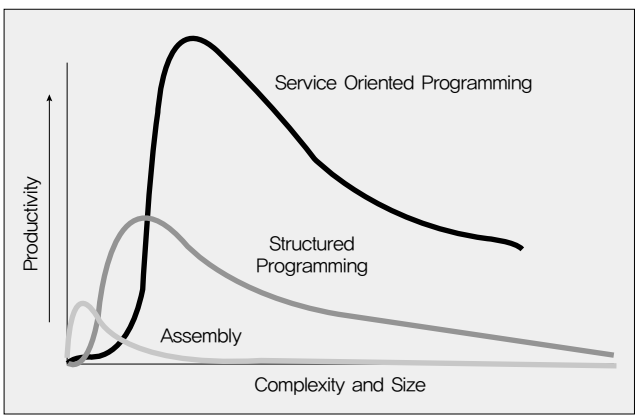
지난 2006년 10월 덴마크에서 열린 'JAOO 컨퍼런스'에서 'The OSGi Foundation for Eclipse'라는 주제로 발표했던 Jeff McAffer는 "그 당시 Eclipse Foundation에서는 전 세계 소프트웨어 시장을 눈 여겨 보고 있었고, 이클립스의 발전을 신기술 도입에 초점을 맞춘 것이 아니라 시장의 흐름과 고객의 요구에 모든 우선순위를 두고 있었다"라고 밝힌 바 있다. 과연 이클립스에서 보았던 소프트웨어 시장의 문제점과 고객들의 불만, 요구사항은 무엇이었을까? 그것은 다음과 같이 몇 가지로 요약할 수 있다.

이식성 및 재사용성의 문제

이식성과 재사용성은 이전부터 많이 회자되는 이슈였고, 또한 그것을 해결하기 위해 컴포넌트, 재사용성에 대한 도구, 툴들이 많이 소개되었으나 여전히 사용자들은 재사용성과 이식성에 대한 불만을 가지고 있었다. 더욱이 과거 PC 기반의 소프트웨어가 대부분이던 시절에서 모바일-임베디드-PC-서버에 이르는 다양한 하드웨어와 플랫폼 환경이 나타나기 시작했고 사용자 환경도 윈도우에서 리눅스, 웹 등으로 점차 다양화되는 실정이었다. 이러한 개발 및 사용자 환경에서 향후 개발된 소프트웨어의 이식성과 재사용성, 다양한 플랫폼의 지원은 더욱 더 중요한 요소로 작용할 것이 분명하다고 예측했다.

소프트웨어의 복잡성

날로 증가하는 소프트웨어 코드는 매우 심각할 정도이다. 일반적인 DVD 플레이어는 백만 라인의 코드를 가지고 있고, BMW의 신규 차량은 50여개의 디바이스가 네트워킹되어 제어 및 관리되고 있다. 심지어 이클립스 2.x에도 250만 라인의 코드가 담겨 있다. 어떻게 보면 이러한 코드의 급격한 증가는 OOAD, OOP의 한계에서 찾을 수 있을 것 같다. Object 개념은 심플하고 재사용성에 뚜렷한 장점을 지니나 최근 두드러지는 네트워크를 통한 연동 및 운영(Collaboration) 환경에서는 오히려 그 장점이 퇴색된다. 다양한 서비스들을 요구하는 시스템에서는 Loose-coupled된 인터페이스가 더 효율적이나 현재의 OOP에서는 오히려 오브젝트간의 복잡성이 증가하고 그 결과로 자연스레 코드의 수가 늘어나게 된다. 또한 오브젝트간의 유연성(Flexibility)도 플러그인 형태의 구조로 개발자가 직접 정의하고 구현해야하는 상황에 이르렀다.



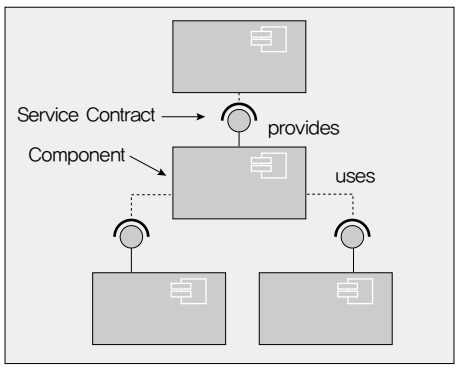
〈그림 2〉 증가하는 소프트웨어 복잡도

효율적인 소프트웨어 관리의 문제

최근 오픈소스와 인터넷의 발달로 과거와는 비교할 수 없을 정

도로 빠르게 버전업과 디버깅, 패치가 이뤄지며, 다양한 플랫폼에서 인터넷을 통해 실시간으로 요구사항과 기능들이 추가되고 수정된다. 따라서 과거의 업그레이드 방식과는 다르게 원격에서 실시간으로 동적인 시스템 업그레이드와 관리가 이뤄지는 형태가 요구되는 시대가 온 것이다.

이러한 문제점들과 사용자들의 요구사항을 분석한 Eclipse Foundation에서는 해결 방안으로 몇 가지 특징과 솔루션, 아키텍처들을 설계하게 되는데 그 당시 새로운 시스템으로 설계할 때 필요한 요구사항은 크게 네 가지였다. 가장 중요한 요소로 Reliability, Large Scale Distribution, Wide Range of devices, Collaborative 등을 정의하고 이것을 구현하기 위한 기술적인 대안으로 SOA, 프레임워크, Dynamic Module를 선택하게 된다.



〈그림 3〉
Eclipse-Equinox
서비스 아키텍처

결국 이는 OSGi의 특징과 많은 부분에서 일치하고 있다. 이러한 분석 결과로 Eclipse Foundation은 1999년부터 개발되고 발전되어 온 OSGi를 이클립스의 기본 프레임워크로 탑재할 것을 결정한다. 그 후로 SMF를 통해 OSGi 솔루션의 선두주자인 IBM에 기술 지원을 요청하게 되고 자연스럽게 이클립스의 통합이 이뤄지게 된다(어쩌면 이것은 당연한 결과였는지도 모른다. 이클립스 역시 IBM에서 개발되어 오픈소스로 공개한 개발 툴이고 오픈소스로 공개되었다 할지라도 여전히 Eclipse Foundation에서 가장 강력한 힘을 발휘하고 있는 것 역시 사실이다. 마치 썬에서 자바를 오픈소스로 공개하고 JCP를 통해 관리하고 있지만 여전히 썬이 자바 세계에서 가지고 있는 무소불위의 힘은 무시할 수 없는 것과 마찬가지인 셈이다). 이렇게 해서 IBM은 이클립스에서 Equinox에 이르는 개발 툴과 프레임워크를 자연스레 통합해 하나의 거대한 플랫폼으로 발전시키는 역할을 담당하게 된다.

한편 좀 더 자세히 이클립스에서 OSGi를 탑재한 이유들을 살펴보면 위에서 언급한 문제점에 대한 해결 방안을 확인할 수 있다. 지난 2005년 IBM Systems Journal에 실린 'The Eclipse 3.0 Platform - Adopting OSGi Technology'에서는 Reliability, Large Scale Distribution, Wide Range of devices, Collaborative의 요구사항에 SOA, 프레임워크, Dynamic Module의 기본적인 아키텍처를 제시하고 OSGi 입장에서의 기능과 장점들을 기술해놓고 있다.

- Extensible : 확장성을 가장 높은 우선순위에 두고 있다. 대단히 정확한 지적이다. 단순한 하나의 개발 툴에서 개발환경(PDE), 플랫폼(RCP)으로 발전하기 위해서는 단순히 플러그인을 다운로드해 기능을 추가하는 것에 그쳐서는 안 된다. 새로운 애플리케이션, 프레임워크, 심지어 서버로서의 확장이 가능해야 할 것이다.

〈화면 1〉
Eclipse-Equinox
플랫폼 빌드

- Light, Dynamic Module : 빠르고 다양한 기능과 콘텐츠, 심지어 서버들을 실시간으로 원격에서 시스템 중단 없이 추가 및 업데이트할 수 있는 관리 기능이 필요하며 또한 가능해야 할 것이다. 네트워크도 유/무선 통합 환경이어야 하므로 가볍고 코드가 작아야 한다.

- Platform Independence : 단순히 윈도우, 리눅스 환경의 문제가 아니다. 모바일-임베디드-PC-서버에 이르는 다양한 하드웨어와 운영체제가 있기에 그 위에서 운영되는 플랫폼에 독립적으로 콘텐츠가 운영되고 서비스가 사용되어야 한다.

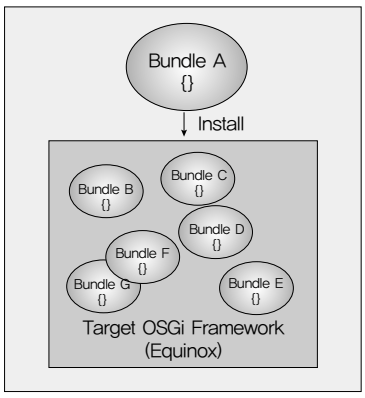
- Rich Eco Systems - Open Source Community : 오픈소스 단체와 지원을 최대한 빠르고 정확하게 수용할 수 있는 구조와 프로세스가 정립되어 있어야 한다. 빠르게 발전하는 기술과 트렌드를 적절히 대응하기 위해서는 오픈소스의 활용을 극대화해야만 한다.

- Native Look and Feel : 윈도우만의 환경이 아니므로 다양한 플랫폼과 디바이스에 적용될 수 있는, 그리고 직관적으로 사용자들에게 경험되고 다가갈 수 있는 GUI가 필요하게 될 것이다. 하나의 프로세스와 직관적인 경험이 플랫폼과 디바이스에 독

립적으로 사용되게 해야 할 것이다.

도약하는 Equinox OSGi

한편 OSGi Alliance 입장에서 OSGi의 발전 가능성과 미래를 예측해 볼 때 이클립스가 OSGi를 선택하면서 가져다 준 장점은 무엇일까? 우리는 지금까지 이클립스가 OSGi를 탑재하면서 얻은 장점들만을 살펴봤다. 그러나 이클립스가 OSGi를 탑재해서 과거와는 완전히 다른 아키텍처를 가지고 하나의 플랫폼으로 확장한 것은 사실이지만, OSGi Alliance 입장에서 본다면 이클립스 못지않게 많은 장점을 가져다 준 것 또한 사실이다. 가장 큰 장점은 바로 브랜드 파워, 즉 마켓에서의 힘이다. 과거 OSGi가 임베디드 시스템에서 안정적인 Dynamic Module 시스템으로 명성(?)을 날렸던 것은 사실이다. 그러나 그것은 어디까지나 한정된 분야(홈게이트웨이, 셋탑박스, 스마트폰 등)였고 또한 임베디드 시스템 전체에서 보면 자바가 탑재되는 일부에 지나지 않았다. 사실 임베디드 시스템에서는 여전히 RTOS와 C 개발 환경에 API 라이브러리, 디바이스 드라이버 그리고 프로세스로 작동되는 구조가 다수인 현실에서 OSGi는 시스템 전체에서 본다면 그야말로 소수의 선택된 개발자들만을 위한 프레임워크였던 것이다. 그러나 이클립스는 다르다. 모바일, 임베디드 등의 다양한 디바이스와 플랫폼이 탄생했다할지라도 여전히 개발 환경에서의 메인 스트림(Main Stream)은 서버 사이드(Server-side)와 PC 클라이언트이다. PC에서 개발하는 개발자들에게 비주얼 스튜디오를 사용하지 않고도 C/C++, Java 등의 개발 환경과 PHP, Perl, Python 등의 스크립트 통합 환경을 제공하고 컴파일-디버깅-버그 트랙킹-소스 관리 등의 개발 프로세스를 무료로 제공한다는 것은 이클립스의 거부할 수 없는 매력임에 틀림없다. 다시 말해 OSGi와는 비교하지 못할 충성스러운 수백만 명의 개발자와 브랜드 파워를 가지고 있는 것이다. 그런 이클립스의 새 버전인 3.0에 핵심으로 탑재된 런타임 환경이었으니 OSGi를 몰랐던 개발자들에게 다가왔던 충격은 대단했다. 물론 개발자들이 갖는 관심과 실제 Equinox OSGi를 사용해 구현하는 것은 다르지만, 이제 Equinox OSGi는 이클립스에 탑재된 단순한 하나의 기능이 아니라 Equinox OSGi를 몰라서는 더 이상 이클립스 플랫폼에서 개발할 수 없을 만한 핵심 요소로 자리 잡았다.



〈그림 4〉
Eclipse-Equinox 타깃 서비스 모델

이로 인해 OSGi Alliance는 전문가 그룹이 3개에서 4개로 확장되고, 스펙 또한 R3에서 R4로 신속하게 업그레이드된다. 이런 변화야말로 Equinox OSGi가 탄생하고 이클립스에 탑재되면서 나타난 대표적인 장점일 것이다. 또 다른 이점은 OSGi가 가지고 있던 시스템 아키텍처와 개발 프로세스들이 소프트웨어 업계에

Equinox's Friends

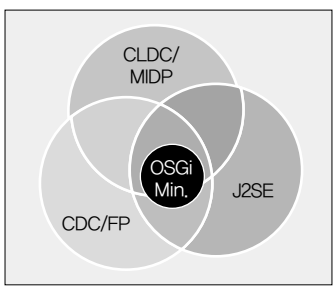
Equinox OSGi를 살펴보면 자주 등장하는 인물들이 있다. BJ Hargrave, Jeff McAffer, Thomas Watson이 바로 그들인데, 모두 IBM 출신이면서 Eclipse Foundation, Equinox Project 그리고 OSGi Alliance에 이르기까지 많은 영향력을 끼치는 사람들이다. 한 가지 아쉬운 것은 모두 IBM 사람들이라는 것인데 이처럼 Eclipse Foundation, Equinox Project, OSGi Alliance 모두에 IBM이 막강한 영향력을 끼친다는 사실은 개발자로서는 아쉬움이 남는 부분이다.

- **Jeff McAffer(IBM Rational Software)** : Equinox 탄생의 주역. Equinox 프로젝트의 총책임자이면서 개발 리더였으며, 또한 이클립스 설계와 개발에도 초기부터 참여해 왔다. 이클립스, Equinox 세미나와 컨퍼런스의 단골 강사이기도 한 그는 캐나다 온타리오에 있는 IBM 연구소에서 근무했고 현재 Equinox를 통한 이클립스 플랫폼 확장에 노력하고 있다. 일본 동경대학에서 박사학위를 받은 특이한 경력의 소유자이기도 하다.

- **Thomas Watson(IBM Lotus)** : 미국 텍사스의 IBM 오스틴 연구소에서 근무하고 있는 그는 Eclipse Equinox 프로젝트에 참가하기 전에 IBM OSGi의 과거 버전인 SMF를 개발한 주역이다. Jeff McAffer와 함께 Eclipse Equinox OSGi 전도사임을 자칭하고 있으며 소프트웨어 아키텍트이면서 엔지니어링 파트에 더 힘을 기울이고 있다. Eclipse Equinox 프로젝트에서 이클립스 환경에 맞는 최적화된 OSGi 디자인을 담당하기도 한 그는 현재 이클립스 플랫폼을 활용해 신규 시스템을 개발했다. 바로 그것이 IBM 협업 환경 Expeditor인데 이는 그의 대표적인 작품으로 꼽힌다.

- **BJ Hargrave(IBM Software Group)** : 미국 텍사스의 IBM 오스틴 연구소에 근무하면서 동시에 2002년부터 OSGi Alliance에서 Lead 엔지니어로 일하다가 현재는 CTO를 맡고 있다. 과거 IBM에서 18년간 소프트웨어 아키텍트와 엔지니어로 근무한 경력이 있으며 주로 임베디드 시스템의 커널과 운영 환경에 주목해 왔다. 이클립스에 OSGi가 탑재되면서부터 자연스레 OSGi Alliance와 Eclipse Foundation 모두에 영향력을 끼치게 되었다.

널리 펴져나갈 계기를 마련했다는 것이다. 이제 이클립스가 보유하고 있었던 다양한 오픈소스 기반의 플러그인들은 Equinox OSGi 번들 형태로 개발된다. Equinox OSGi의 가장 큰 장점인 가볍고(light) 동적인(dynamic) 모듈 시스템 형태의 번들로 개발되면서 Equinox OSGi가 구현된 OSGi R4 스펙은 소프트웨어 업계의 대표적인 표준 요구사항이면서 동시에 개발 프로세스의 트렌드를 주도하게 될 것이다. 물론 Equinox OSGi가 아직 완전한 프레임워크는 아니다. 여전히 발전해야 할 부분이 많이 남아 있다. 특히 모바일 시스템의 지원과 엔터프라이즈 환경에서 서버 사이드 프레임워크로 자리 잡는 부분은 현재 가장 큰 이슈이자 과제이다. 이를 위해 J2ME의 CLDC 환경의 발전과 대표적인 서버 프레임워크인 '스프링' 과의 OSGi 통합 등으로 계속해서 변화하고 있다.



〈그림 5〉
Eclipse-Equinox 실행 환경

Equinox Framework

Equinox Framework Component는 OSGi Core Framework R4 스펙에 충실하게 구현된 코드이다. 크게 세 가지로 구분되는데 번들을 실행하는 Launcher와 bootstrap infrastructure를 수행하는 Boot support, 그리고 번들을 수행하고 관리하는 System Framework로 이뤄져 있다. 아래의 프레임워크 코드들은 모두 Eclipse CVS repository에 정리되어 있다.

- **OSGi R4 Framework(org.eclipse.osgi)** : Eclipse Equinox의 가장 기본이 되는 메인 프레임워크이다. 이 프레임워크를 빌드하면 org.eclispe.osgi.jar가 생성된다.

- **Boot Support(org.eclipse.equinox.boot)** : 이클립스가 구동될 때 framework classloader가 셋업되고 개발 및 운영 환경이 안정적으로 수행하는 것을 지원한다. 현재까지는 org.eclipse. osgi나 org.eclipse.equinox가 아닌 org.eclipse.platform 프로젝트에 속해 있고, 수행 파일은 startup.jar이다.

- **Native Launcher(platform-launcher)** : 실제 환경으로 startup.jar 파일을 수행시키고 그 결과로 프레임워크가 로딩되어 수행된다. 이 모듈은 C로 작성되어 있으며 윈도우, 유닉스, 리눅스, MacOS 등의 다양한 플랫폼과 하드웨어의 사양에 맞게 각각의 버전이 존재한다.

Equinox Framework 파일은 이클립스에서 직접 임포트(Import)해 패치하거나 수정할 수 있는데, 이클립스 메뉴에서 Import -〉 CVS -〉 Projects from CVS를 선택해 CVS에 있는 코드들을 불러올 수 있다.

Equinox Bundles

Equinox Bundles Component들은 OSGi의 다양한 서비스와 기능을 제공하는 모듈로 가장 기본적인 실행 단위이다. 과거 이클립스의 다양한 플러그인 형태의 모듈들이 번들 형태로 변환되어 개발되거나 제공된다. 현재 Equinox Bundles에 대한 스펙 제정과 관리는 OSGi Alliance의 MEG(Mobile Expert Group)와 VEG(Vehicle Expert Group)에서 각각 기능별로 구분해 담당한다. 주요 Bundles은 다음과 같다.

- **Application Container(org.eclipse.equinox.app)**
가장 기본적인 애플리케이션 번들로, MEG에서 제정한 애플리케이션 컨테이너 서비스이다.

- **Common Utility Bundle(org.eclipse.equinox.common)**
이클립스에 의해 자주 호출되어 사용되는 상태 모니터링, Asset 체크 등의 유틸리티 클래스 라이브러리이다.

- **Config Admin(org.eclipse.equinox.config)**
OSGi R4 스펙에 의해 구현되는 Configuration Admin service이다.

- **Device Access Service(org.eclipse.equinox.device)**
OSGi R4 스펙에 의해 구현되는 Device Access Service이다. 자동으로 추가 및 삭제되는 하드웨어 디바이스 탐지 및 그에 해당하는 디바이스 드라이버를 자동으로 탐색하고 다운로드하는 기능을 제공한다.

- **Event Admin Service(org.eclipse.equinox.event)**
번들간의 커뮤니케이션 통신 메커니즘을 제공한다. 구독과 발행의 이벤트 모델을 따라서 수행한다.

- **HTTP Service(org.eclipse.equinox.http)**
OSGi R4 HTTP service. HTML 및 Java servlets을 실행한다.

- **Log Service(org.eclipse.equinox.log)**
OSGi 환경에서 일어나는 모든 로그인 상태 분석 및 관리 기능

그 외에도 Metatype Service, Preferences Service, Extension Registry, Supplemental Bundle/JAR, User Admin Service, OSGi Services API, OSGi Utilities 기능의 번들들이 제공된다.

Equinox Incubator

Equinox Incubator를 실시하는 가장 큰 목적은 이클립스 플랫폼의 확장과 신기술 확보에 대한 실험적 프로젝트 구현이다. 또한 실질적인 프로젝트 수행의 세부적인 가이드라인은 개발하고자 하는 프로젝트의 결과물이 실제 개발자들이 이클립스 플랫폼에서 개발할 때 유용하거나 다양한 수행 환경을 테스트할 수 있어야 한다는 것이다. 무엇보다 신기술의 습득이라고 해서 실제 적용되는 환경에 완전히 무관할 수 없다는 이야기이다. Equinox Incubator를 통해 나온 결과물이 실제적이고(Practical) 유용하고(Applicable) 적용 가능한 형태의 솔루션이나 서비스 기능이 제공되어야 한다는 점이 중요하다. 여기에서 수행되는 모든 프로젝트 결과물은 반드시 Equinox OSGi에서만 사용되는 것은 아니다. 오픈소스이기에 다른 OSGi 환경에서도 사용할 수 있다. 주요 프로젝트를 소개하면 다음과 같다.

- **AspectsJ** : AOP(Aspect-Oriented Programming)를 자바 환경에 적용하는 연구 프로젝트

- **Resource Monitoring** : Equinox OSGi 환경에서 수행되는 모든 번들들의 상태, 메모리, 통신들을 모니터링하는 프레임워크를 구현하는 프로젝트

- **Security** : Eclipse Equinox OSGi하에서 자바 보안 모델과의 유연한 통합을 목표로 하는 프로젝트이다. 지금까지 OSGi 환경에서는 보안 모델을 자바의 메커니즘에 전적으로 의존했는데, 프레임워크를 넘어서 RCP나 서버 플랫폼으로 확장하는 경우에는 보다 강력한 보안 모델이 필요하게 되었다.

- **Server Side OSGi** : Equinox OSGi를 엔터프라이즈 환경으로 확장하기 위한 다양한 시도들 가운데 하나이다. 기존의 서버에 탑재된 프레임워크와의 유연한 연동 및 통합 방안을 이슈로 하고 있다. 그들 중의 하나가 바로 Spring/OSGi이다.

- **Provisioning** : 하드웨어 디바이스와 플랫폼에 독립적으로 다양한 콘텐츠를 실시간 관리할 수 있는 기능을 목표로 한다. 기존의 Update Manager를 대체하면서 보다 강력한 기능으로 선보일 예정이다. IBM에서는 Tivoli Update Manager를 이미 서비스하고 있다.

Equinox Server

Equinox OSGi는 일반적으로 임베디드 시스템이나 PC 애플리케이션에서 사용되지만, 점차 서버로서의 기능을 확장해 나가고 있다. 대표적으로 http와 같은 웹서버나 JSP 또는 Servlet 컨테이너들이 있으며 점차 그 영역을 확장해 애플리케이션 서버로서의 가능성을 보여주고 있다. 많은 애플리케이션 서버들이 Equinox 상에서 구현되고 있으며 대표적으로 Jetty, Jasper들이 있다. 그 외에도 다양한 API 패키지로 구현되어 수행된다. 이렇게 처음부터 설계되고 구현된 번들이 아닌, 다른 써드파티에서 구현된 서버들을 API 패키지로 구현할 경우에는 별도의 전용 repository가 존재해 모든 관리를 담당한다. 바로 Orbit 프로젝트가 그것이다. Orbit 프로젝트의 목적은 다양한 번들의 버전 관리를 담당하는 것인데, 일반적으로 다른 써드파티의 API 라이브러리들을 Equinox 번들화시켜 관리하는 경우를 뜻한다.

Equinox Server Architecture

일반적으로 Equinox Server에는 http 서버와 Servlet, JSP를 수행하는 컨테이너 기능이 있으며 두 가지 방법으로 서버를 구동할 수 있다.

- **Servlet Container와 Equinox 연동** : servletbridge를 통해 Equinox Bundle과 기존의 서블릿 컨테이너간의 연동으로 서블릿 수행

- **Equinox에 내장된 http Server 구동** : 우선적으로 권장하는 방법이다. Equinox에 최적화된 http service 번들을 활용하는 방법으로 가장 적은 메모리를 사용해 서비스한다. 우선 Equinox에서는 2개의 서비스 번들을 제공하는데 org.eclipse.equinox.http와 org.eclipse.equinox.http.jetty이다. Equinox.http는 Servlet API 2.1까지 지원하지만 안정적이면서 가장 작은 메모리를 사용하는 장점이 있고, Equinox.http.jetty는 다소 많은 메모리를 사용하지만 Servlet API 2.4까지 지원하는 장점이 있다.

Equinox server에서 JSP, Servlet을 사용하려면 ❶ org.eclipse.equinox.http, ❷ javax.servlet, ❸ org.eclipse.equinox.

http.registry 등의 3개 번들을 실행하거나 또는 Jetty를 활용해 ❶ org.eclipse.equinox.http.jetty, ❷ org.eclipse.equinox.http.servlet, ❸ org.mortbay.jetty, ❹ org.apache.commons.logging, ❺ javax.servlet, ❻ org.eclipse.equinox.http.registry 6개의 번들을 로딩하면 된다. 따라서 메모리와 하드웨어 성능이 여유가 있다면 jetty를 이용하면 되고 그렇지 않다면 일반적으로 http를 이용해 서버 서비스를 수행하면 된다.

주요 Server-Side Bundles

대부분 HTTP service와 JSP, Servlet을 구동하기 위한 Container 기능을 제공하고 있다. HTTP service, HTTP registry, Servlet Bridge, HTTP Ser vlet, HTTP ServletBridge, Servlet API, Servlet JSP API, Jetty, HTTP Jetty 등이 있으며 특별히 ServletBridge 기능을 통해 기존의 애플리케이션 서버와 인터페이스할 수 있는 확장된 기능을 선보이게 되었다.

Equinox의 미래

Equinox OSGi는 향후해 어떻게 발전되어 나갈 것인가? 현재 OSGi Alliance의 개발 로드맵을 살펴보면 Equinox OSGi는 임베디드에서 모바일과 PC 환경을 넘어 엔터프라이즈 환경으로 성공적인 확장과 발전을 목표로 하고 있다. 우선적으로 Equinox OSGi는 이클립스에 탑재됨으로써 PC 환경으로 확장해 나가는데 성공하고 있는 듯 하지만, 궁극적으로 유니버설 프레임워크(Univesal Framework)나 플랫폼으로 발전하기 위해서는 Eclipse-Equinox가 서버 사이드 플랫폼으로 자리매김해야 가능하다. 현재 이러한 흐름에는 IBM, BEA, Apache, Spring 등의 업체가 참여해 자사의 제품에 Eclipse Equinox를 탑재하는 프로젝트를 진행하고 있거나 이미 부분적으로 상용화해 서비스하고 있다.

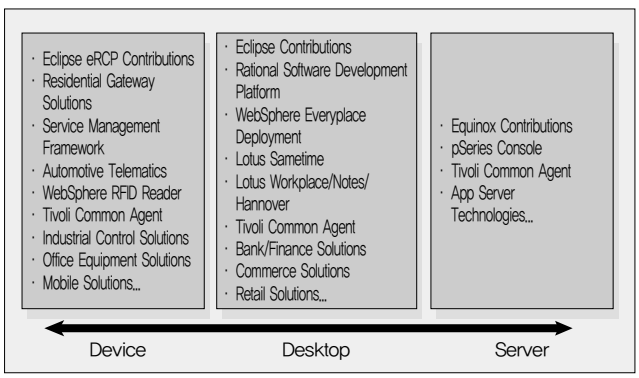
지난 3월에 열린 EclipseCon 컨퍼런스의 기조연설에서 Eclipse-Equinox 개발의 주역인 Jeff McAffer는 "향후에는 원격에서 다양한 콘텐츠들과 서비스 기능들을 쉽게 업데이트하거나 관리 및 운영하려는 요구사항들이 나올 것이다. 시공간을 초월해서 다양한 서비스들을 플랫폼과 하드웨어 디바이스에 무관하게 즉각적으로 제공할 수 있는 기능이 반드시 필요하다"고 말하면서 향후 Eclipse-Equinox의 중요한 개발 이슈로 Provisioning System을 꼽았다. 현재 이러한 Provisiong 기능으로는 이클립스의 Update Manager가 있고, 상용화 솔루션으로는 IBM의 Tivoli Manager를 통해 서비스되고 있다. 하지만 Update Manager 기능을 넘어서는 Provisioning Framework로서의 확장성이 요구되고 있고, 이러한 신규 개발 프로젝트는 Eclipse-Equinox Incubator를 통해 진행 중이란 설명이다. 그 주요 레퍼런스를 소개하면 다음과 같다.

〈화면 2〉
IBM Tivoli – Provisioning Manager

– IBM : IBM은 Eclipse-Equinox가 존재하는 가장 큰 근거를 남긴 업체이다. 또한 반대로 IBM이 추구하는 전략 – Pervasive Computing, Business On-Demand에 가장 큰 수확을 거둔 업체이기도 하다. IBM은 자사가 개발해 키운 두 가지 제품(이클립스, SMF-OSGi)를 오픈소스로 공개했고, 그것에 그치는 것이 아니라 오픈소스 업계에 막강한 영향력을 발휘해 Eclipse-Equinox를 하나의 거대한 플랫폼으로 발전시키는 성과를 거뒀다. IBM은 Eclipse-Equinox를 기반으로 협업 솔루션인 Lotus Domino를 Sametime, Expeditor로 새롭게 변신시켜 선보였고, 대표적인 WAS – WebSphere에 적용해 Websphere Application Server로, 시스템 운영 솔루션인 Tivoli Provising Manager, 그리고 소프트웨어 개발 툴과 프로세스 솔루션인 래쇼널의 전체 소프트웨어 플랫폼에 적용해 고객들에게 제공했다. 이러한 사례는 IBM이라는 업체가 오픈소스와 그를 지원하는 많은 커뮤니티들을 단순히 신기술 발표의 장이나 하나의 트렌드 정도로 보는 것이 아니라 자사의 비즈니스 전략에 핵심 요소로 작용하고 있음을 보여주는 대표적인 사례이다. IBM의 이러한 솔루션 사업 전략은 결과적으로 Eclipse-Equinox의 Universal Platform의 확장에 큰 기여를 할 것으로 예상된다. 임베디드 및 모바일 환경에서는 Eclipse-Equinox run-time 모듈로 별도 공급하고 있고, PC 애플리케이션과 RCP에서는 Eclipse-Equinox를 기반으로 하는 솔루션을, 비즈니스 애플리케이션과 협업 그리고 메시징 시스템에서는 Lotus Expeditor를, 엔터프라이즈 환경에서는 Websphere Server와 Tivoli Solution을 제공하고, 전 개발 프로세스에서는 래쇼널 솔루션을 지원하는 등 전 분야에 걸친 방대한 Eclipse-Equinox 기반의 솔루션과 시스템을 제공하고 있다.

– 어도비 : 포토샵으로 유명한 어도비(Adobe)는 근래 매크로미디어를 인수해 웹과 모바일, PC 환경을 통합해 Imaging, Document, Publishing을 포함한 Digital Content Solution Provider로서의 가치를 높이고 있는 업체다. 따라서 IBM과는 다른 각도

〈화면 3〉
IBM Lotus –
Expeditor



〈그림 6〉 IBM의 Eclipse-Equinox 적용 솔루션

에서 Eclipse-Equinox를 활용하고 있는데, 중점을 두고 적용하는 분야는 바로 Content Management System이다. 이를 위해 어도비는 다양한 포맷의 콘텐츠들을 관리하고 공유하고 콘텐츠 작업들을 효율적으로 팀 내에서 협업하는 데 중점을 둔 솔루션에 Eclipse-Equinox를 적용해 시장에 선보이고 있다. 대표적인 솔루션으로는 Adobe Version Cue가 있다.

– 아파치 : 웹서버로 유명한 오픈소스 단체 아파치(Apache)는 Eclipse-Equinox를 활용하는 것이 아니라 직접 OSGi 스펙을 구현하고 또한 주로 서버 환경에 적용하고 있다. 대표적으로 OSGi 구현 시스템인 Apache-Felix가 있으며 JSE 5.0 기반의 Harmony, Cocoon, James, Geronimo 등이 있는데, 상용화 솔루션으로 개발되기보다는 오픈소스의 성격에 맞게 서버 환경에 적용하

〈화면 4〉 Adobe –
Version Cue

거나 신기술 도입 측면에서 구현되는 것이 특징이다. 따라서 아파치 단체에서 성공적으로 구현된 프로젝트들은 다른 서비스 업체들에게 모듈이나 런타임 기반으로 제공되는 경우가 많다. 아파치 단체가 무엇 때문에 Eclipse-Equinox 같은 플랫폼 또는 런타임 환경에 관심을 가지고 많은 프로젝트들을 진행했는지 등은 주의 깊게 살펴볼 필요가 있다.

JSR 277 vs. JSR 291(OSGi)

Eclipse-Equinox를 포함한 OSGi의 요즘 새로운 이슈는 바로 JSR 277과의 대결이다. JSR 277은 Dynamic Component Model로서 JAVA 7의 주요 스펙 가운데 하나이다. 그 특징적인 요소가 OSGi와 많은 부분에서 유사한 특성을 지녀 OSGi 업계의 많은 관심을 불러일으키고 있다. 실제 JSR 291은 OSGi 스펙으로서 JSR 입장으로 본다면 277과 291 스펙이 충돌하는 것처럼 비쳐진다. 얼마 전 OSGi Alliance의 Peter Kriens은 자신의 블로그를 통해 JSR 277은 이상적인 모델이기는 하나, 실제 환경과는 너무나 동떨어진 모델이며 JAVA 7 스펙에는 포함될지 모르지만 다음 버전인 8 또는 9에서는 곧 사라지게 될 것이라고 직격탄을 날렸다.

그는 또한 JSR 277과 291을 조목조목 항목별로 비교 분석했는데, 대표적으로 Consistency, Optionality, Split Packages 등의 문제점을 예로 들었다. 그는 결론에서 "아직 최종 확정되지는 않았지만 개발 중인 JST 277(Dynamic Com ponent Model)은 자바의 임베디드 버전인 J2ME에서 구동되기 어려우며 지나치게 무겁고, 결국 OSGi 벤더들이 환영하지 않을 것이다"라고 적었다. 또 다른 개발자는 지금의 JSR277이 지속적으로 발전한다면 3~4년 후에는 많은 제품에 탑재될 수도 있겠지만, 오히려 그 때쯤이면 지구상의 모든 시스템에 이미 Eclipse-Equinox가 설치되어 있어 존재 의미가 사라질 것이라고까지 이야기했다. 이렇듯 JSR 277의 등장은 Eclipse-Equinox 업계의 큰 이슈를 몰고 왔는데, 아마도 그것은 Eclipse-Equinox가 근본적으로 자바 환경에서 운영되는 프레임워크이면서 플랫폼이기 때문에 자바를 선도하는 썬에서 OSGi와 비슷한 스펙을 제정했을 때 다가오는 혼란이나 어느 정도 밀려날지도 모르는 불안감들을 나타낸 것이라고도 볼 수 있다.

어느 의견과 주장이 맞는지는 결국 이용자들이 선택하고 판단을 내리게 될 것이다. 그러나 이러한 이슈에서 나타났듯이 가볍고 다이내믹한 컴포넌트 모델을 탑재하고 무장한 Eclipse-Equinox는 향후 모든 시스템에서 사용하게 될 것이고, 널리 확산되는 유니버설 프레임워크나 플랫폼으로 성장해 나갈 것이라는 사실이다.