이클립스 플러그인 기반의 확장성 있는 서버 프레임워크

이동호, 김창수○, 박정은
연세대학교 전산학과
고려대학교 전자전기공학과
광운대학교 컴퓨터공학과
ldh0826@naver.com , ssalsue@korea.ac.kr , hehdki@paran.com

Eclipse Plugin based Scalable Server Framework

Dongho Lee, Changsoo Kim○, Jungeun Park
Department of Computer Science, Yonsei University.
Department of Electrical Engineering, Korea University.
Department of Computer Science, Kwang-woon University.

요 약

본 논문에서는 이클립스 플러그인 기반의 확장성 있는 서버 프레임워크에 대해서 기술한다. 플러그인이란 가전제품의 플러그를 전기 콘센트에 연결하여 사용하듯, 프레임워크에 각 기능을 구현하는 플러그인 컴포넌트를 연결하여 그 컴포넌트를 사용할 수 있는 기술이다. 이 플러그인 기술을 이용하여 여러 서비스를 제공 할 수 있는 서버 프레임 워크를 설계, 구현하였다. 본 논문에서는 플러그인 기술을 설명하고 플러그인 기반 서버의 장점과 활용 방안에 대해서 기술한다.

1. 서론

정보 기술의 급속한 발달과 눈부신 인터넷 보급에 따라 우리가 사용하는 컴퓨터나 휴대폰 단말과 같은 정보기기가 어떠한 네트워크에 연결되지 않는 것을 상상하기 어렵게 되었다. 일상 생활에서도 수 많은 서버-클라이언트 모델을 이룬 서비스를 사용하고 있으며, 클라이언트의 정보를 취합하거나 서버공통 기능을 사용하며 제어한다고 할 때 서버의 역할이 필수적이다. 더욱이 요즈음 크게 이슈화 되고 있는 정보보안, 모바일 단말의 프로세싱 능력에 대한 보조를 위해 서버기반 컴퓨팅(Server Based Computing)을 한다고 한다면, 서버의 기능은 무엇보다도 중요 하다고 할 수 있다.

서버는 다양한 기능의 군집으로 이루어져 있고 사용자의 여러 요구를 충족하기 위한 많은 기능이 신속히 추가되거나 재구성 되어야 한다. 따라서 기능이 집중되는 특성에 따라 서버의 종류 및 구조가 복잡해지고 있다.

한편 서버 특성상 클라이언트와 달리 특정 OS가 대표적으로 보급 된 것이 아니라 여러 OS환경에서 구동 될 수 있어야 하며, 정보의 집중을 위해 필수적으로 사용하는 DBMS종류도 다양하여 구축 환경에 따라 같은 기능의 서버를 다시 개발 하는 사례도 있다.

이로 인해 서버 소프트웨어 개발이나 유지 보수에 소모되는 비용이 증가하며, 이는 전체 생산 비용에서 비중 있는 비용을 차지한다.

본 논문에서는 이클립스 플러그인을 기반으로 하여 기동되는 OS에 영향을 받지 않는 서버 프레임워크와 각 기능별로 독립적인 플러그인으로 구성하고, DBMS의 쿼리만을 한번에 관리 할 수 있는 DB프레임워크 중 하나인 iBATIS를 도입하여 비즈니스 로직과 DBMS쿼리를 분리하여 새로운 DBMS에 대해 유연성을 제공하는 등 서버 형상의 재구성과, 기능의 재사용성을 높이고 유지 보수성을 크게 개선시킨 서버프레임워크를 제안하고 기술한다.

2장에서는 이클립스 플러그인 서버 프레임워크에서 사용한 기반기술인 이클립스 플러그인 기술에 대해 설명하고, 3장에서는 플러그인 서버프레임워크의 설계 및 구현 그리고 서버 프레임워크에 의한 기능처리 플러그인 구현에 대해 설명 하였다. 4장에서는 개발된 서버 프레임워크의 유틸리티에 대해 소개 하고, 마지막 5장에서는 본 논문의 결론과 앞으로의 발전 방향에 대해 기술하였다.

2. 관련 연구

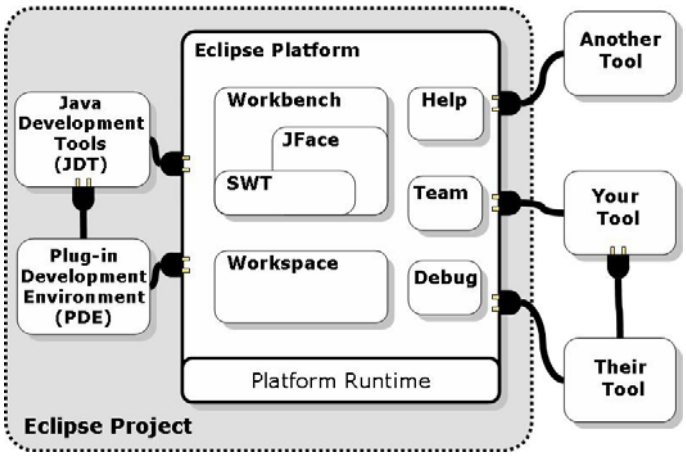
이클립스 플러그인은 대표적인 OSGi 컨테이너인 이클립스의 Equinox 말하며 OSGi는 Java용 동적 모듈

시스템으로서 재사용 가능한 모듈식 Java프로그램을 개발하고 배포하는데, 필요한 프레임워크로 빠르게 확산되고 있다[1]. OSGi 컨테이너를 사용하면 Java모듈(OSGi식으로는 "번들"에 해당)을 jar파일 형식으로 배포 할 수 있으며 독립적인 자바/가상머신 환경에서 제공하지 못하는 완전하며, 동적인 컴포넌트 모델을 구현하며 모듈간의 동적(Dynamic)관계와 의존을 매우 효과적으로 관리 할 수 있다. 게다가 다양한 기능의 번들을 같은 컨테이너로 배포할 수 있다는 점이 OSGi의 매력적인 점이다. OSGi 컨테이너로 배포된 모든 번들은 JVM에서 실행된다[1,2,3,4].

이러한 플러그인 기술은 개발 되어있는 프레임워크에 어떠한 컴포넌트를 연결하면 그것이 적용 된다. 마치 하나의 콘센트에 여러 플러그를 연결하듯 끼워질 플러그인 프레임워크에 맞게 개발된 플러그인을 연결하면 바로 그 프레임워크에 적용되어 하나의 프로그램으로 구성된다[5].

이클립스는 초기 오픈 소스 프로젝트로 하나의 프레임워크인 이클립스 플러그인 로더가 개발됐고 그 후 지속적으로 플러그인들이 개발되어 계속 발전하고 있다. (그림1)과 같이 이클립스는 외부 플러그인 뿐 아니라 내부에서 지원되는 서비스도 플러그인으로 이루어져 있으며, 각 플러그인 하나하나가 내부적으로도 완전한 독립성을 보장 받고 있다[6].

플러그인은 확장점(Extension Point)을 선언하므로써 기존 플러그인 기능을 다른 플러그인이 적절한 방식으로 확장할 수 있게 한다. 각 플러그인은 콘센트에 플러그를 연결 하듯, 플랫폼에 연결 하면 그 플러그인의 역할을 수행 하며, 그 플러그인이 제거되어도 전체 플랫폼의 동작에 영향을 주지 않는다. 이 기술을 사용하여 무한한 확장성과 유연성을 가지게 된다.



(그림1) Eclipse Plugin Architecture

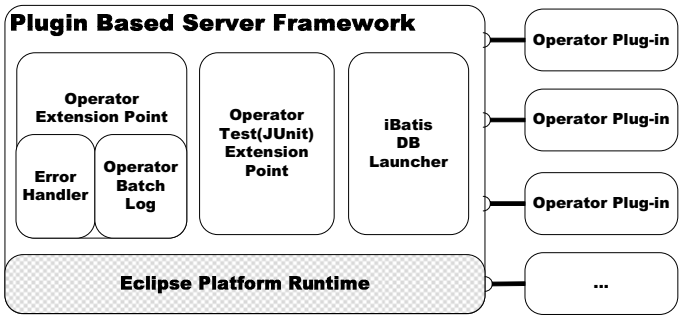
3. 서버프레임워크의 설계 및 구현

3.1 플러그인 기반 서버 프레임워크

본 논문에서 제시한 서버프레임워크는 아래와 같은 기준을 만족하도록 설계되었다.

A. OS Platform에 독립적이어야 한다.
B. 기능에 따른 Operation 구현에 대한 확장성과 독립성이 있어야 한다.
C. DB Platform에 대한 연결과 SQL Query에 대해 독립적이어야 한다.
D. Log에 대한 Leveling 및 특정 관점에 따른 Logging을 지원 하여야 하고, 시스템 구동중인 시점에서 Log에 대한 설정을 할 수 있도록 한다.
E. 중복된 구현에 대해 재차 작성하지 않도록 일괄 처리한다.
F. 각 Operation 구현에 따른 Self-Test및 Tester Tool을 사용한 Test가 가능하여야 한다.

위와 같은 기준을 만족하기 위해 Java언어를 사용하여 JVM기반으로 서버를 실행 하여 OS Platform에 독립적으로 구동되며[4], 이클립스 플러그인 기반으로 서버프레임워크를 구현하고 이에 대한 확장점을 정의하여 추가되는 기능간의 독립성과 서버의 확장성을 보장한다[5]. DB Platform과 기능에 따른 로직의 독립성을 유지하기 위해 iBatis를 이용하여 DB를 관리[7,8]하고 Log의 Leveling관리를 위해 Log4J[9]로 구성하였고, 구동중인 서버에서 Log를 즉시 제어하기 위해 Java Scripting기능을 사용하였다. 또한 매번 중복되는 기능(패킷 분석, 전송할 패킷 작성, Exception, Logging 등) 을 플러그인 마다 재차 작성하지 않도록 AspectJ (Aspect - Oriented Programming Java)[10] 를 도입하고, 각 기능별 Self-Test를 위해 JUnit을 사용하였다.



(그림2) Plugin based Server Framework

위의 내용을 바탕으로 (그림2)와 같이 이클립스 플러그인 기반 서버 프레임워크를 구현하였다. 이 프레임워크는 다음의 두 가지 확장점을 제공한다. 기능에 따른 Operaion 처리에 대한 확장점(Operator Extension Point)과 Self-Test및 Tester를 이용한 Test를 위한 확장점(OperatorTest Extension Point)가 그것이다. 각 확장점에 대한 구현은 이클립스 기반의 플러그인으로 작성되고, 서버구동시 구현된 플러그인을

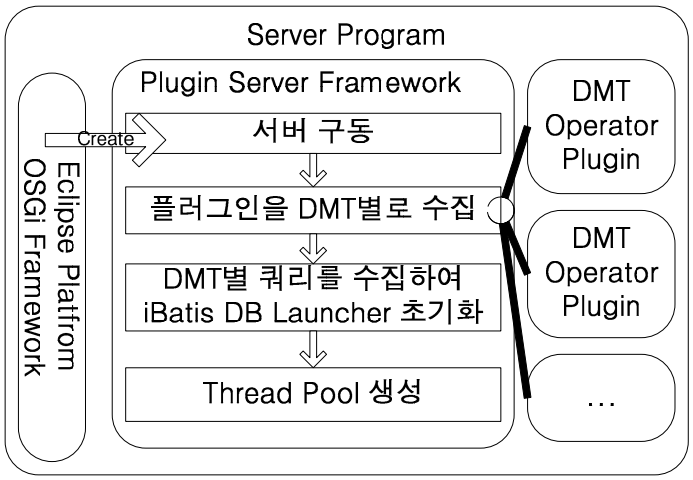
포함하여 클라이언트의 요청을 처리한다.

서버프레임워크에 하나의 단위기능을 Data Message Type (DMT)라고 정의하였으며, 서버프레임워크는 이 DMT를 확인하여 해당 기능을 호출한다.

3.2 서버 프레임워크의 초기화

플러그인 서버프레임 워크는 (그림3)과 같은 초기화 단계를 거친다,.

Server가 구동되면 이클립스 플러그인 로더는 서버프레임워크 플러그인을 로드한다. 이후 서버 프레임워크는 확장점으로 연결된 각 기능별 DMT Operator 플러그인을 수집한다. 이후 수집된 DMT Operator 플러그인을 포함하여 iBatis DB Launcher를 초기화하고, 서버에 제한된 개수 (본 논문에서는 128개로 지정하였다) 만큼의 Thread Pool을 생성한다.

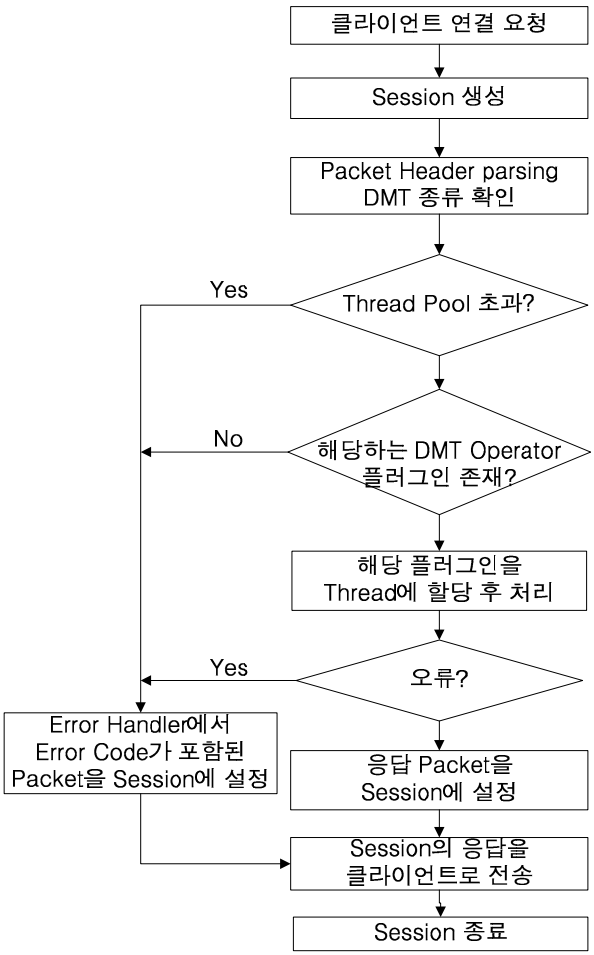


(그림3) 서버프레임워크의 초기화

3.3 Data Message Type(DMT) Operation 처리

서버프레임워크가 초기화 된 후 클라이언트의 요청에 의해 처리되는 순서는 (그림 4)와 같다.

해당 순서도의 선결 상태는 Server Framework 이 구동 및 초기화 된 상태이며, 클라이언트의 요청이 있는 경우 Session 을 생성하고, Packet Header 를 Parsing 하여 어떤 Data Message Type 인지 종류를 확인한다. 이후 해당되는 DMT Operator 플러그인을 Thread 에 Session 과 함께 할당한 후 플러그인의 처리 과정에 따라 처리한다. 이후 해당되는 플러그인은 처리결과를 Session 에 설정하고, 서버프레임워크는 Session 의 응답을 클라이언트로 전송한 뒤 Session 을 종료한다. 만약 Session 에 응답이 설정되지 않은 상태로 Thread 가 종료된 경우는 Error Handler 에서 이를 Catch 하여 해당되는 Error Code 가 포함된 Packet 을 구성하여 Session 을 할당한다.



(그림 4) Operation 처리 순서도

3.4. 서버 프레임워크에 의한 DMT 플러그인의 구현

본 논문의 플러그인 서버프레임 워크는 서버기능(Data Messasge Type Operator Plugin)에 대한 확장점과 Tester(Test Plugin)를 위한 확장점을 제공한다. 제공되는 확장점에 맞도록 플러그인을 제작하고, 서버프레임워크 플러그인에 연결 시켜주면 기능이 동작하게 된다. 본 논문에서 설계한 서버프레임워크의 확장점의 정의는 아래와 같다.

3.4.1 서버기능(DMT Operator Plugin)에 대한 확장점

서버의 실제 기능을 구현하는 확장점으로서 run() method 에 해당 기능의 로직이 구현된다.

이 확장점의 구조는 (표 1)과 같다.

(표 1) DMT Operator 의 확장점 구조

```
public class DMTOperator extends ServerOperator {
  public DMTOperator () {
        … //프레임워크에서 로드할 DMT의 분류 설정
  }
  @Override
```

```
  public String[] getSQLMapResourcesUrl() {
        … //이 기능의 DB쿼리가 담긴 파일의 위치
        return urlString;
  }
  @Override
  public boolean validateRequest() throws Throwable
  {
        … //입력 받은 패킷을 분석하고 유효성 검사
        return true;
  }
  public void run() {
        … //이 기능의 로직이 구현되는 부분
  }
}
```

서버 프레임워크에서 확장된 플러그인을 서버 프레임워크가 로드하고, DMTOperator() Method 에서 이 플러그인의 DMT 명과 분류에 대한 설정을 읽어 들이고, getSQLMapResourceUrl() Method 에서 iBatis DB Launcher 에서 이 플러그인을 위해 사용될 DB 쿼리문을 로드한다. validateRequest() 에서는 이 플러그인으로 넘겨진 패킷의 오류를 검증하고 parsing 하여 Bean 에 넘겨준다. run() method 에서 이 DMT Operator plugin 이 실제 처리할 내용이 구현되도록 하였다.

만약 이 플러그인이 실행 되는 도중 오류가 나면, 3.3 절에서 기술한 바와 같이 Error Handler 에서 이것을 catch 하여 해당되는 Error Code 가 포함된 Packet 을 구성한 뒤 플러그인을 종료한다.

3.4.2 서버 Test 에 대한 확장점

서버의 각 플러그인 단위테스트를 위한 확장점이다. JUnit 을 통한 단위테스트와 서버 프레임워크의 자체 유틸리티를 통한 단위테스트를 위해 구성되는 기능이다.

이 확장점의 구조는 (표 2)와 같다.

(표 2) DMT Tester 의 확장점 구조

```
public class DMTTeser extends OperatorTest {
  public DMTTeser() {
        … //테스터에 입력될 변수를 작성
  }
  @Override
  public PooledByteBuffer buildRequest() throws
Throwable {
        … //DMT Operator로 보낼 패킷 구성.
        return byteBuffer;
  }
}
```

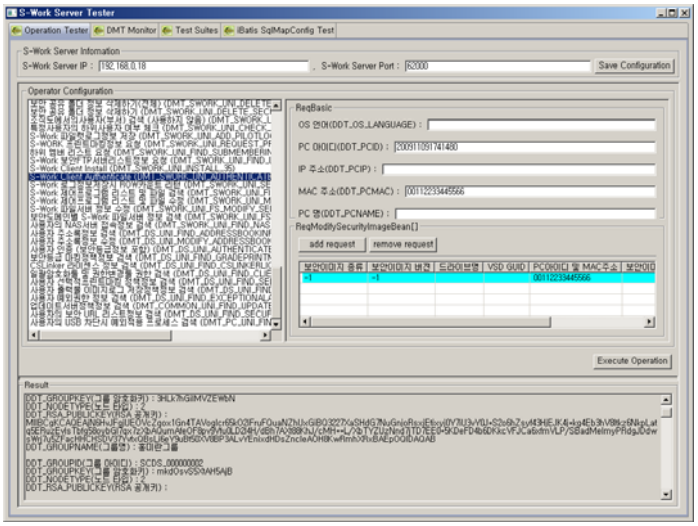
JUnit 으로 테스트하는 경우 서버프레임워크의 JUnit 모듈이 DMTTester() 의 입력될 변수를 읽고 buildRequest() Method 에서 DMT Operator 로 보낼 패킷을 구성하여 테스트를 진행한다. 추가 유틸리티를 사용해서 Test 하는 사례는 5 절에서 기술 하도록 한다.

4. 서버 프레임워크의 유틸리티

본 서버프레임워크를 사용하여 작성된 각 기능별 DMT Operator Plugin 을 단위테스트 하는 도구(그림 5)와 서버가 구동되는 중에 서버에 들어오는 패킷을 캡쳐하여 어떤 DMT Operator 플러그인을 로드하며 어떤 응답을 보내는지 모니터링 할 수 있는 모니터링 도구와 모니터링 도구(그림 6)가 있다. 클라이언트의 요청은 클라이언트의 기능에 따라 여러 번의 요청과 다양한 기 DMT Operator plugin 을 호출할 수 있으므로 모니터링 도구에서 캡쳐한 패킷을 하나의 테스트 군으로 묶어 관리할 수 있는 유틸리티(그림 7)가 구현되어있다.

구현된 테스터 또한 플러그인으로 구성되어있다.

각각의 유틸리티를 살펴 보도록 하자.

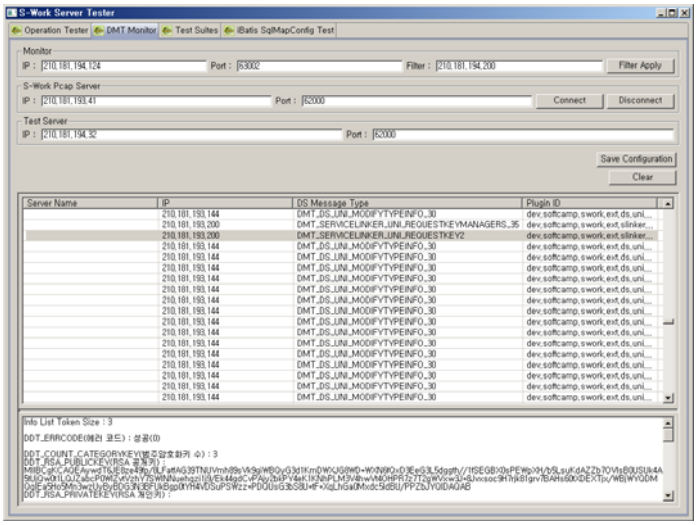


(그림 5) 단위 Tester

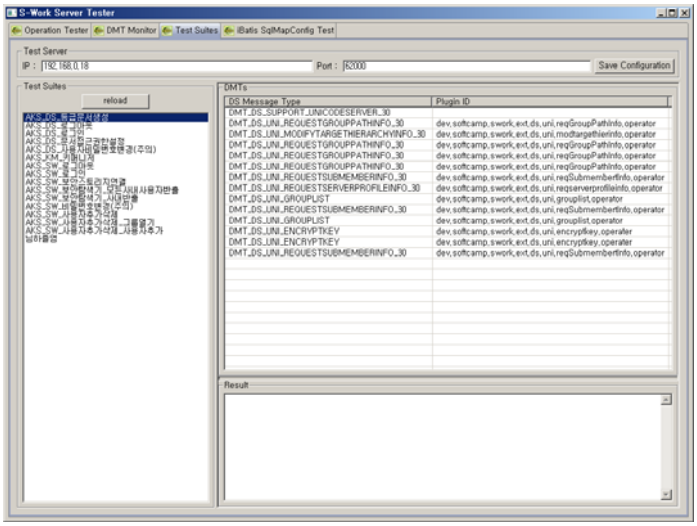
(그림 5)의 단위 Tester 의 테스트단위는 각 DMT Operator Plugin 의 Test 에 대한 확장점에 정의된 내용이 로드되어 있으며, 개별적인 Plugin 에 대한 단위테스트를 하고 그 결과를 바로 볼 수 있도록 구성되었다.

(그림 6)에서의 서버 패킷 모니터링 도구의 경우 서버로 들어오는 모든 패킷이 분석되며, 입력 값을 분석하고, 해당되는 입력을 다시 서버로 보내어 출력되는 값을 볼 수 있도록 구성되어 있다.

(그림 7)에서처럼 이렇게 들어온 입력 값을 구성되는 하나의 기능군을 Test Suites 로 묶어서 관리하고, 기능군 별로 테스트 할 수 있도록 구성 되었다.



(그림 6) 서버 패킷 모니터링 도구



(그림 7) Test Suites 관리툴

5. 결론

본 논문에서는 이클립스 플러스인을 기반으로 하여 서버프레임워크를 설계, 개발하였다. 플러그인 기술을 활용하면서, 서버기능의 확장이 용이 하고, 확장점의 구조만 알면 누구나 쉽게 서버를 개발 할 수 있다. 또한 기능별로 독립적인 플러그인으로 구성되어있으므로, 일부 기능에 문제가 있는 경우 전체 서버를 다시 교체할 필요 없이 문제 있는 기능만을 수정하고, 해당되는 플러그인만 교체하면 완료된다. 또한 서버의 기능 형상 별로 소스를 관리할 필요 없이 릴리즈 시점에서 원하는 기능의 플러그인만 선택하면 되므로 서버의 형상관리 및 유지보수에 매우 용이하다.

현재의 서버프레임워크에서는 패킷 통신 수준의 서버모델을 제시하였지만, 더 나아가 파일 스트리밍이나 파일서버로의 확장도 가능 할 것이다. 또한 기능들의 분산처리를 위한 방법들도 생각 해 볼 수 있을 것이다.

참고문헌

[1]IBM, OSGi 번들을 이용한 웹 서비스 개발 및 전개, http://www.ibm.com/developerworks/kr/library/ws-OSGi/index.html#resources, 2009

[2] OSGi Alliance, Service Platform Core Specification release 4, p. 1, 2006.

[3] OSGi Alliance, Service Platform Specification release 3, pp. 15-21, 2003.

[4] Jon Meyer, Troy Downing, “JAVA Virtual Machine”, O’REILLY, pp. 63-68, 1997.

[5]양석호, 이클립스 실전 플러그인 개발, 에이콘, 2007

[6]http://help.eclipse.org - Eclipse Help Page

[7]Clinton Begin, Ibatis in Action, Manning Publications, 2007

[8]http://ibatis.apache.org - iBatis Site

[9]Samudra Gupta, Pro Apache Log4j SECOND EDITION, Apress, 2005

[10]Laddad, AspectJ in action, Oreilly & Associates Inc, 2009

[11]김성박, 자바 I/O & NIO 네트워크 프로그래밍 한빛미디어 2004

[12]주현태 外 3, OSGi 기반의 확장형 소프트웨어 플랫폼, 한국컴퓨터종합학술대회 논문집, Vol.35, No.1(B), pp. 504-508, 2008