Ⅰ. 개요

FUSE™ ARCHITECTURE 2.0 는 Spring의 경량 컨테이너 기술을 사용하여 이중화된 컨텍스트로 구성된다. Bootstrap application context 는 코어 모듈들이 존재하며 Business application context 에는 비즈니스 모듈들이 존재한다. Bootstrap application context 는 Business application context 의 부모 컨텍스트로 Business application context 에 존재하는 모듈은 부모에 존재하는 모듈에 접근이 허용된다.



FUSE™ ARCHITECTURE 2.0 기반의 응용프로그램은 추상화 3 계층 서비스 응용프로그램으로 논리적으로 구성된 객체 지향 응용프로그램이다. 데이터베이스 관점에서 볼 때 이 응용프로그램은 OLTP 처리 모델을 기반으로 한다. 인프라 관점에서 보면 이 응용 프로그램은 하드웨어 및 네트워크 아키텍처는 웹 서버 및 응용 프로그램 서버 기능에 대해 개별적 물리적 계층을 요청하는 4개 계층 분산을 기반으로 한다. 마지막으로 배포 관점에서 볼 때는

공개 소프트웨어

Spring

자바 개발자들에게 가장 널리 알려진 공개 소프트웨어 Spring은 Rod Johnson 의 저서 “Expert One-on-One J2EE Design and Development, 2002” 의 코드에 기반하는Layered Java/J2EE 애플리케이션 플랫폼이다. FUSE™ ARCHITECTURE 2.0 은 Spring 의 경량 컨테이너, 트랜잭션 관리를 위한 추상화 계층(Abstract Layer), JDBC 추상화 계층(Abstract Layer), AOP 기능들을 사용하고 있다.

Ⅱ. 설치

# 1. 사전 준비 사항

|  |  |  |  |
| --- | --- | --- | --- |
| **환 경** | | | **검증 여부** |
| H/W | IBM p780 | | ○ |
| S/W | IBM AIX 5.3 Oracle Weblogic 10.3 with Java 1.6.x Oracle 10g | |
| H/W | | HP DL380 G7 | ○ |
| S/W | | Tomcat 6.0.30 with Java 1.6.0 Redhat Enterprise Linux Server release 5.5  Oracle 11g(11.2.0) |

# 2. 설치

## 2.1 어플리케이션 환경 설정

웹 어플리케이션을 배포할 때는 WEB-INF/classes 경로에 아래와 같이 XML 파일(applicatioin-init.xml)을 생성하여 어플리케이션 동작을 위하여 필요한 환경 설정 정보 및 데이터들을 저장하게 될 경로를 설정 한다.

<?xml version="1.0" encoding="UTF-8"?>  
<application>

<home>file://C:\TOOLS\workspace\architecture\_v2\architecture-ee\profile\default</home>  
</application>

## 2.2 어플리케이션 서버 환경 설정

## 2.3 클러스터 환경 설정

## 2.4 싱글사인온(Single Sign On) 환경 설정

# 3. 클러스터 환경 구축

Ⅲ. 주요 기능

# 1. 개요

# 2. 사용자 & 그룹

# 3. 보안

# 4. 다국어 & 지역화

## 4.1 소개

소프트웨어에서 여러 언어를 지원하기 위한 체계를 갖추는 것을 Internationalization 또는 줄여서 I18n 라고 하고 특정 언어에 맞추는 과정을 Localization 또는 줄여서 L10n 라고 한다. 완벽한 I18n과 L10n 을 구현하기 위해서는 특정 지역(Locale)에 적절하게 문자들의 집합을 모우고, 프로그램 전반에 걸쳐 적절한 위치에서 문자들을 사용할 수 있어야 한다.

자바에서는 특정 Locale 에 대한 자원 또는 정보들의 집합체인 ResourceBundle 클래스를 제공하고 있다. 특정 지역을 표현하는 자바의 Locale 클래스의 언어와 국가 코드는 2문자이다. 언어 코드는 ISO-639[[1]](#footnote-1) 표준에 정의되어 있으며 소문자로 표현된다. 국가코드는 ISO-3166 [[2]](#footnote-2)표준에 정의되어 있으며 대문자로 표현된다. 자바는 특정 Locale 에 해당하는 ResourceBundle 을 아래와 같은 순서로 찾는다.

1. bundleName\_localeLanguage\_localeCountry\_localeVariant
2. bundleName\_localeLanguage\_localeCountry
3. bundleName\_localeLanguage
4. bundleName\_defaultLanguage\_defaultCountry\_defaultVariant
5. bundleName\_defaultLanguage\_defaultCountry
6. bundleName\_defaultLanguage

예를 들어 Locale.KOREA 에 “ApplicaitonResoueceBundle” 를 을 찾는 경우 아래와 같은 순서로 찾게 된다.

ResourceBundle.getBundle( “ApplicaitonResoueceBundle”, Locale.KOREA );

1. ApplicaitonResoueceBundle\_ko\_KR
2. ApplicaitonResoueceBundle\_ko
3. ApplicaitonResoueceBundle\_defaultLanguage\_defaultCountry\_defaultVariant
4. ApplicaitonResoueceBundle\_defaultLanguage\_defaultCountry
5. ApplicaitonResoueceBundle\_defaultLanguage

자바는 ResourceBundle 클래스를 확장한 PropertyResourceBundle 과 ListResourceBundle 클래스를 제공한다. PropertyResourceBundle 는 ISO-8859-1 인코딩을 따르는 .properties 파일에서 특정 Locale 에 따른 문자들을 가져온다. PropertyResourceBundle 는 위에서 검색 규칙에 따라 아래와 같은 순서로 찾게 된다.

디폴트 Locale 이 en\_US 인 경우 :

1. ApplicaitonResoueceBundle\_ko\_KR.properties
2. ApplicaitonResoueceBundle\_ko.properties
3. ApplicaitonResoueceBundle\_en\_US.properties
4. ApplicaitonResoueceBundle\_en.properties

ListResourceBundle 는 추상클래스로 자바 클래스를 사용하여 ResourceBundle 을 구현하는 경우에 사용된다.

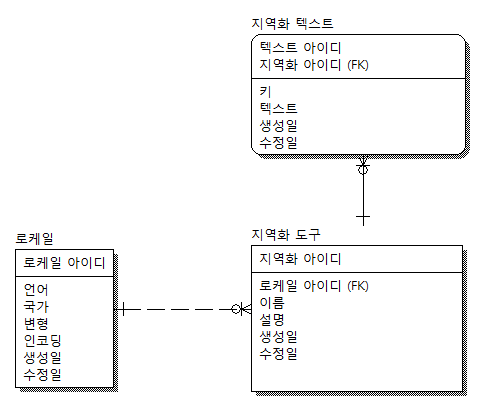
자바에서 제공하는 ResourceBundle 클래스는 I18n과 L10n 을 구현하는 좋은 방법이지만 다음과 같은 어려움이 있다.

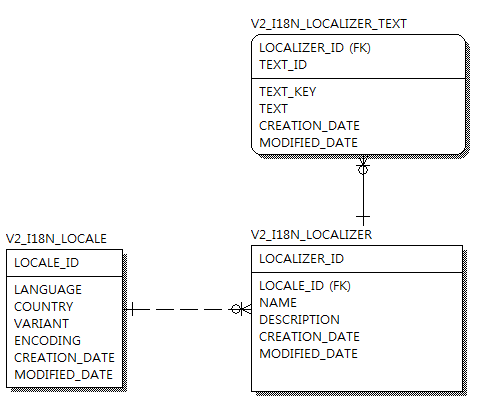
* 동적으로 변경된 내용이 반영되지 않는다.
* ISO-8859-1 인코딩을 따르는 properties 파일의 특성으로 한국어와 같은 유니코드 문자를 사용하는 경우는 변환이 필요하다. (eclipse 와 같은 도구를 사용하면 손쉽게 해결)

## 4.2 I18nTextManager

I18nTextManager 는 기본 컴포넌트로 데이터베이스를 기반으로 동작하는 I18n과 L10n 구현을 위하여 제공된다. 또한 Data Access Object(DAO) 디자인 패턴[[3]](#footnote-3)을 적용하여 데이터 접근 로직과 프로그램 로직을 분리하고 있어 향후 확장이 용의하다.







## 2.2 다국어 지원을 위한 유틸리티

직접 I18nTextManager 를 사용하는 것은 여러 가지 어려움이 존재하기 때문에 특정 Locale 에 해당하는 ResourceBundle 형태로 제공하는 I18nTextUtils 클래스가 제공된다.

I18nTextUtils 클래스를 사용하면 아래의 코드와 같이 쉽게 특정 Locale 에 해당하는 ResourceBundle 객체를 데이터베이스 정보를 기반으로 가져올 수 있다.

ResourceBundle resourceBundle = I18nTextUtils.getResourceBundle("framework", Locale.CANADA);  
  
/\*\*

\* 데이터 베이스에서 Locale.CANADA 해당하는 ResourceBundle 를 찾지 못하는 경우 디폴트 Locale

\* “en\_US”  에 해당하는 ResourceBundle 를 검색하고 존재하지 않는 경우 로컬 파일 시스템에서 검색한다.

\*/

## 2.3 Spring 지원

Spring 의 다국어 메시지 지원을 위하여 제공되는 MessageSource 인터페이스를 확장하는 I18nTextResourceBundleMessageSource 클래스가 제공되며 이를 통하여 Spring 에서 제시하는 방법인 MessageSourceAware 인터페이스를 비즈니스 클래스에서 구현하여 사용하거나 ApplicationHelper.getMessage() 함수를 사용하여 국제화를 처리할 수 있다.

ApplicationHelper.*getMessage*(“main.page.title”, **new** Object[]{}, **null**);

## 2.4 Struts2 지원

# 5. UI

## 5.1 MENU

데이터베이스

테이블 물리 관계도

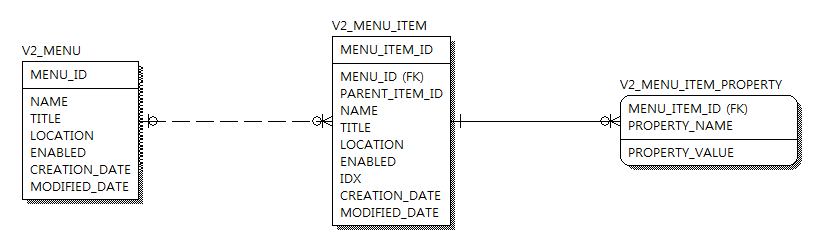
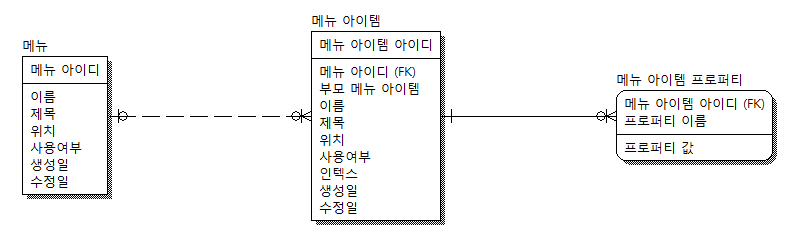


그림 ‑ 테이블 물리 관계도

테이블 논리 관계도



# 6. 메시지 전송

# 7. 데이터 동기화

# 8. 백업

# 9. 데이터베이스 접근

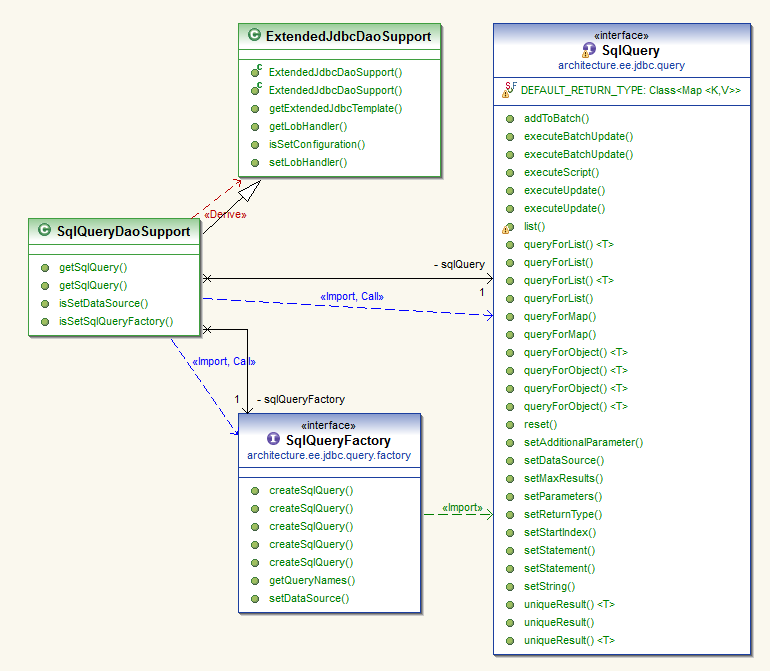
## 3.1 SqlQuery

SqlQuery 는 소스 코드와 SQL(structured query language)을 분리하여 전통적 JDBC 프로그래밍의 생산성 및 유지보수성을 개선하였다.



1. SQL(Structured query language)들을 XML(Extensible Markup Language) 형식으로 관리하여 소스 코드와 SQL을 분리하여 개발하는 것을 지원.
2. XML 파일 디플로이 성능 개선.
3. SQL 상의 기법을 사용하지 않고 특정 범위 내의 쿼리 결과조회 지원. (예를 들면 조회 결과 중 처음 1000 번째 부터 2000 째 까지 형식으로)
4. 대용량 데이터 Update, Insert, Delete 작업을 위한 Batch 작업 지원.
5. SQL들이 정의된 XML 파일들은 로컬 디스크를 포함하여 HTTP, HTTPS, FTP, FTPS, RAM 등 여러 파일 시스템 지원한다.  추가로 JAR, ZIP,  TGZ, TBZ2,  GZIP, BZIP2 압축 파일들을 지원.
6. 파라메터 또는 임의로 주어진 값에 따라 동적으로 여러 형태의 쿼리를 생성을 지원한다. 템플릿 언어 기술을 사용하여 강력한 동적 쿼리 지원.
7. SQL 을 실행할 때, 스크립트 모드를 지원. (여러 쿼리를 한꺼번에 실행할 수 있다. 테이블 생성 및 데이터 입력 작업이 가능)
8. 새로운 데이터타입 Recordset 추가.
9. 보다 손쉬운 데이터베이스 작업을 지원하기 위하여 SqlQueryClient 제공.
10. [Core J2EE 패턴](http://java.sun.com/blueprints/corej2eepatterns/index.html)에서 소개하는 DAO(Data Access Object) 디자인 패턴 지원.

DAO(Data Access Object) 디자인패턴 적용을 위하여 SqlQueryDaoSupport 클래스를 제공하고 있다. SqlQueryDaoSupport를  상속받는 클래스는 getSqlQuery() 함수를 호출하여 SqlQuery 객체를 생성하여 사용할 수 있다.

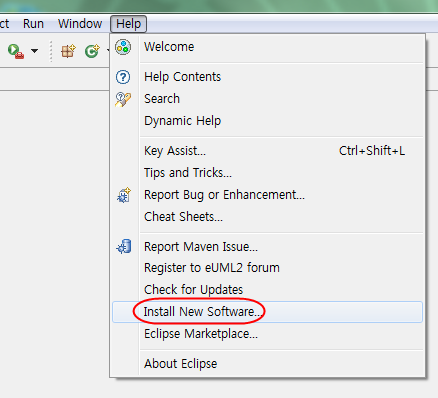


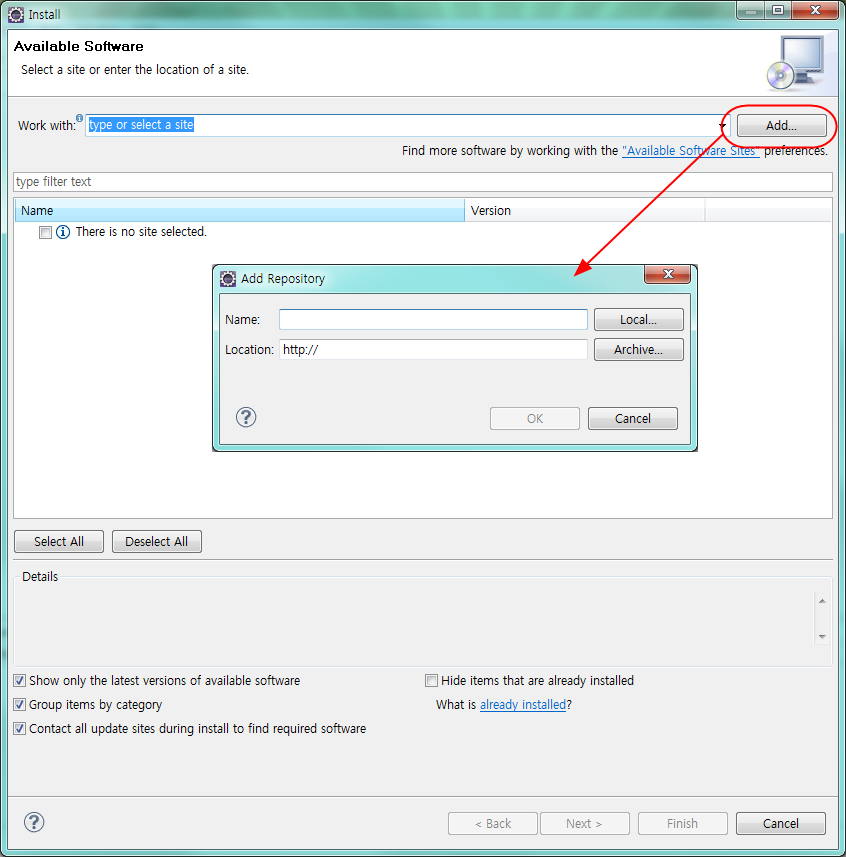
**1.1.1 Samples**  
SqlQuery 는 단일함수 호출 또는 분리된 함수 호출을 통한 데이터베이스 조작을 지원한다.  
  
**Method 1**  
복잡한 처리에 적합하다.  
  
SqlQuery query = getSqlQuery("COMMON.SELECT\_ALL\_ENT\_APP" );  
List rows = query.list();  
  
또는   
  
SqlQuery query = getSqlQuery( );  
List rows = query.setStatement("COMMON.SELECT\_ALL\_ENT\_APP").list();  
  
**Method 2**   
단일 함수 호출을 통하여 처리되며 간단한 처리에 적합하다. (Method1 과 비교하여 고성능)   
  
SqlQuery query = getSqlQuery( );  
query.queryForList("COMMON.SELECT\_ALL\_ENT\_APP");  
  
**setStatement**   
setStatement 함수를 사용하면, SqlQuery 객체를 다시 사용할 수 있다.  
  
query.setStatement("COMMON.SELECT\_ALL\_ENT\_APP2");  
List rows = query.list();  
  
queryForObject, queryForList, queryForMap 함수들은 연속하여 사용하는 경우 명시적으로 reset() 명령을 호출해야 한다.  
  
Integer count = query.reset().queryForObject("COMMON.COUNT\_ENT\_APP\_PROPERTY", new Integer[]{1}, new int []{Types.INTEGER}, Integer.class);  
  
**Batch Update**

SqlQuery query = getSqlQuery();  
query.setStatement("COMMON.INSERT\_ENT\_APP\_PROPERTY");  
for( int i = 0 ; i < 100 ; i ++ ){  
 query.setParameters(new Object[]{ "1" , "name-" + i , "value-" + i }, new int []{Types.INTEGER, Types.VARCHAR, Types.VARCHAR});  
 query.addToBatch();  
}  
int sum = query.executeUpdate();   
  
  
**Pagging**  
  
SqlQuery query = getSqlQuery("COMMON.SELECT\_ALL\_ENT\_APP" );  
List list = query.reset().setMaxResults(10).list();  
  
**1.1.2 Dynamic Query**  
인자로 넘겨지는 값에 따라 동적으로 Sql 을 생성하여 사용할 수 있다. 과거의 기능과 비교하면 다이나믹 언어인 오픈소스 freemarker 엔진을 적용하여 더욱 강력하게 다양한 형식의 쿼리를 동적으로 처리할 수 있다. 다이나믹 쿼리는 부분은 반듯이 <dynamic> 태그로 지정을 해주여야 한다.  
  
SqlQuery query = getSqlQuery("COMMON.SELECT\_ALL\_USER\_ID\_BY\_STATUS");  
List<Integer> list = query.setReturnType(Integer.class).setAdditionalParameter( "parameters", new Object[]{0,1,2,3}).list();

|  |
| --- |
| *common-queryset.xml* |
| <sql-query name="SELECT\_ALL\_USER\_ID\_BY\_STATUS" comment="테이블 데이터 조회" >        SELECT USER\_ID FROM V2\_USERS        <dynamic><![CDATA[                <#assign max\_idx = parameters?size >        <#assign idx = 0 >          <#if 0 < max\_idx >WHERE</#if>              <#list parameters as parameter>        <#assign idx = idx + 1 >         STATUS = ${parameter} <#if idx < max\_idx >OR</#if>        </#list>            ]]></dynamic></sql-query> |

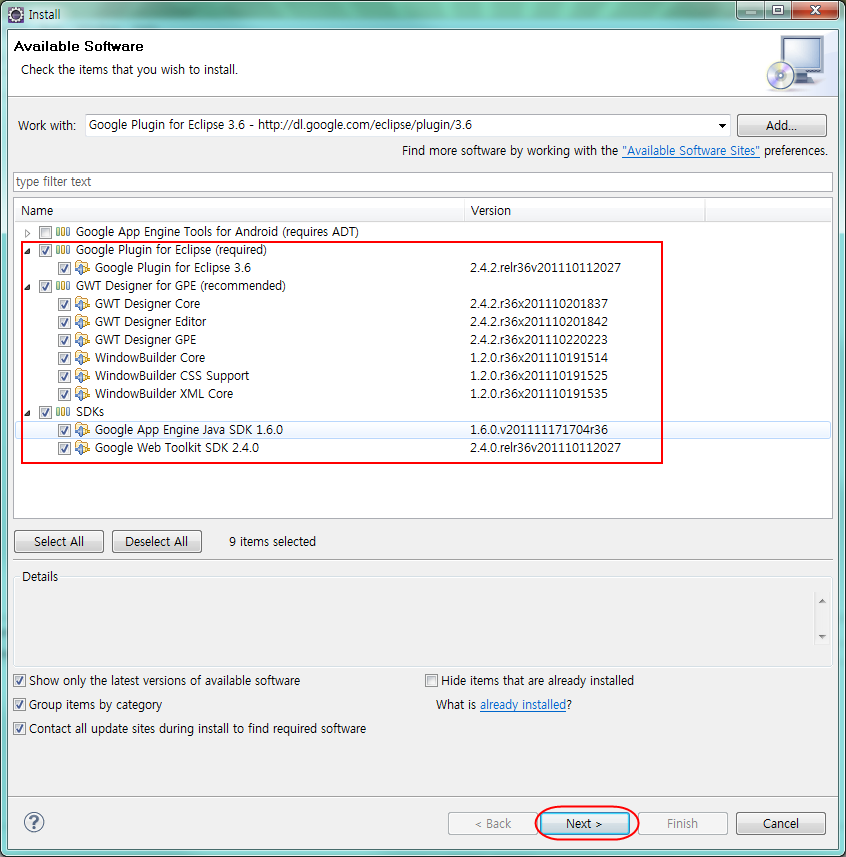
위의 경우 다음과 같은 쿼리가 실행된다.  
  
   SELECT USER\_ID FROM V2\_USERS   
   WHERE  STATUS = 0 OR  
             STATUS = 1 OR  
             STATUS = 2 OR  
             STATUS = 3

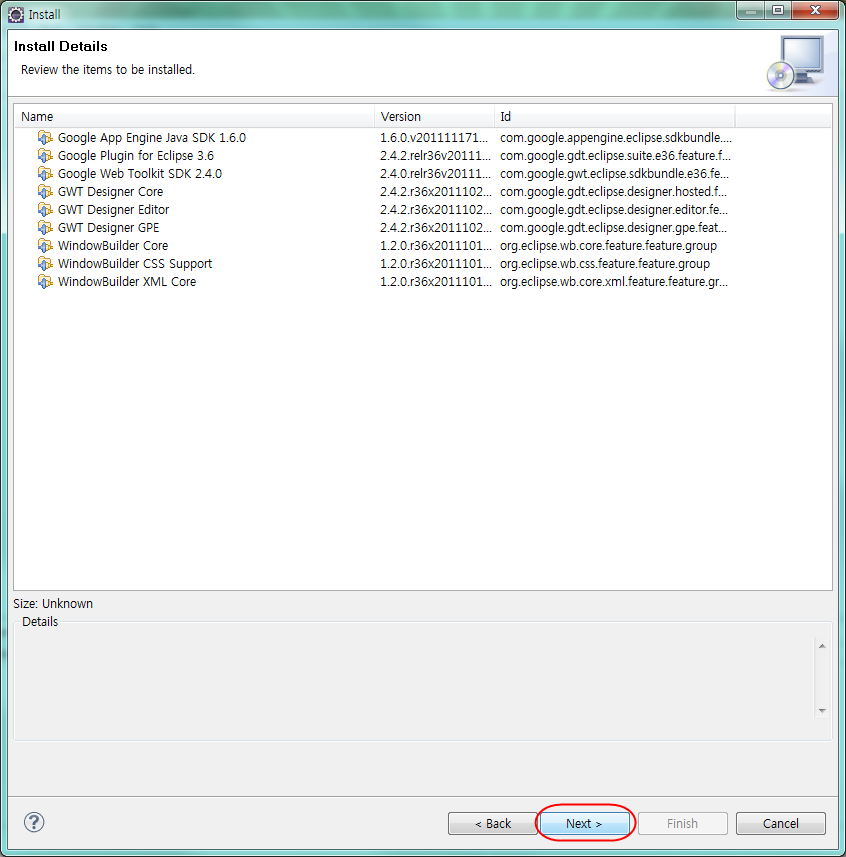


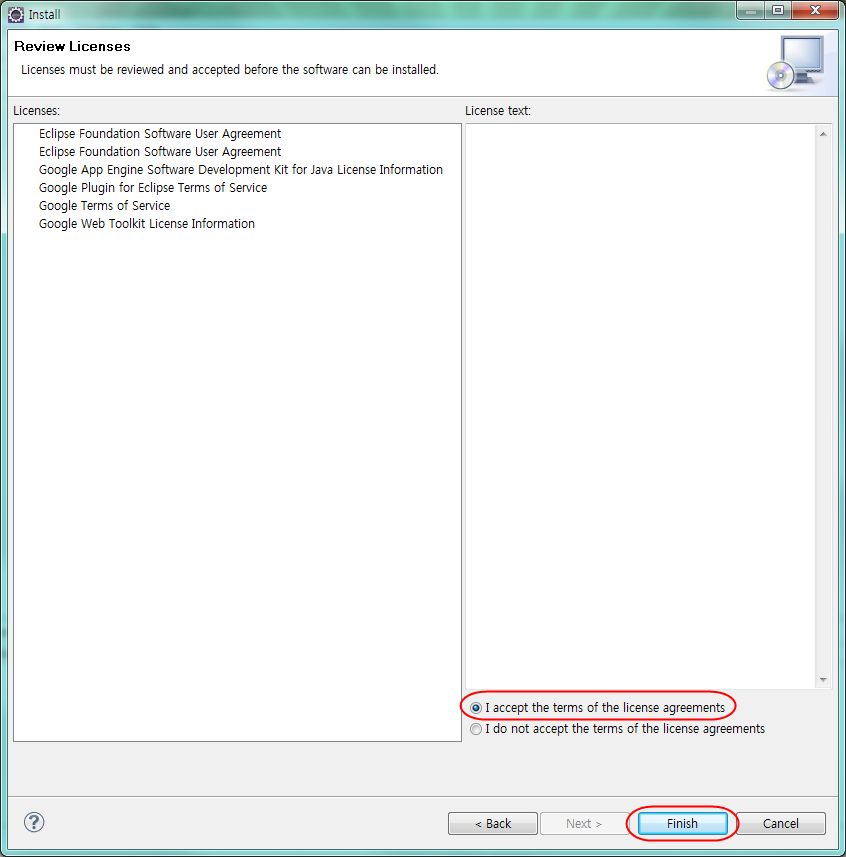


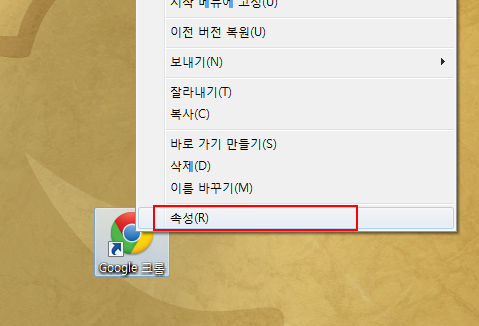
Google Plugin for Eclipse 3.6

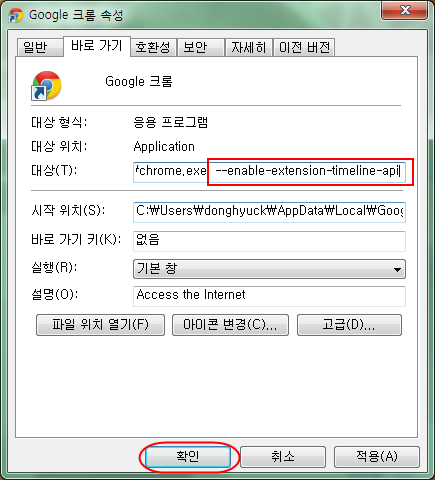
http://dl.google.com/eclipse/plugin/3.6

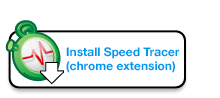




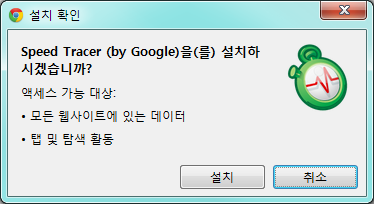






[](https://clients2.google.com/service/update2/crx?response=redirect&x=id=ognampngfcbddbfemdapefohjiobgbdl&uc&lang=en-US)

https://clients2.google.com/service/update2/crx?response=redirect&x=id%3Dognampngfcbddbfemdapefohjiobgbdl%26uc%26lang%3Den-US



Ⅳ. 서비스 구현

1. ISO-639 : <http://www.w3.org/WAI/ER/IG/ert/iso639.htm> [↑](#footnote-ref-1)
2. ISO-3166 : <http://userpage.chemie.fu-berlin.de/diverse/doc/ISO_3166.html> [↑](#footnote-ref-2)
3. DAO (Data Access Objects) : 객체 지향적 설계 패턴의 일종이다. DAO는 애플리케이션에 대하여 하나 이상의 데이터 저장 장치 혹은 관련 소프트웨어에 대한 공통적인 인터페이스를 제공하는 컴포넌트를 의미한다. 즉, 애플리케이션에 대해서는 일관성 있는 데이터 접근을 확보해주는 것이다.  [↑](#footnote-ref-3)