ARQUITECTURA PARA APLICACIONES WEB ORIENTADA A ASPECTOS BASADA EN LOS CONCEPTOS DE SEPARACIÓN DE INCUMBENCIAS

ANEXOS

ALEJANDRA MARIA NARVÁEZ CAMAYO

OMAR ALBEIRO TREJO NARVÁEZ

Director: **IE. Javier Alexander Hurtado**

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática

Popayán, Julio de 2.008

TABLA DE CONTENIDO

ANEXO A: INTEGRACIÓN DE JBOSS AOP CON JBOSS AS

Instalación Del Framework Jboss Aop

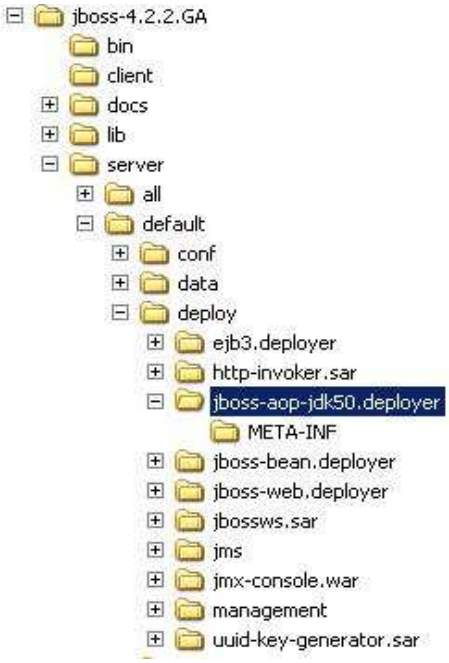
El primer paso para instalar el framework es descargar las herramientas necesarias; se debe tener instalado Jboss Application Server; *jboss-4.2.0.GA.zip*, dicha distribución se puede descargar del sitio oficial de Jboss: http://labs.jboss.com/jbossas/downloads y se debe descargar también una distribución estable de Jboss AOP en este caso *jboss-aop_1.5.6.GA.zip* del sitio http://labs.jboss.com/jbossaop/downloads .
Además se debe tener instalado un JDK apropiado preferiblemente JDK1.4.x o superior.

Instalación Standalone

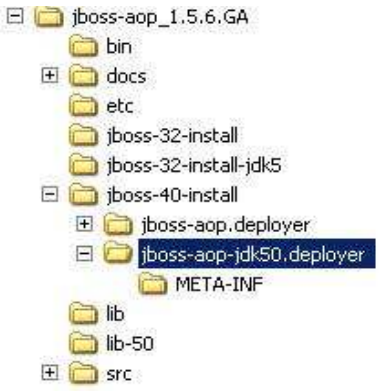
Para instalar Jboss AOP fuera de Jboss Application Server simplemente se utilizan las librerias que se encuentran en los directorios *lib/* y *lib-50/* de la distribución de Jboss AOP utilizando JDK 1.4x y JDK 5 respectivamente.

Instalación con JBoss 4.x Application Server

1. Se debe eliminar el directorio *jboss-aop.deployer* que se encuentra en la ruta *server*/<config-name>/deploy; donde config name hace referencia al tipo de servidor utilizado, que puede ser default, all o minimal.



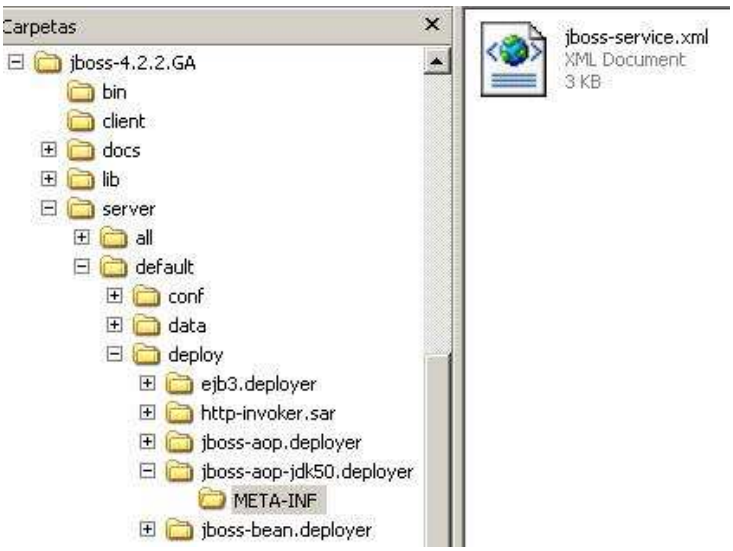
2. Se copia el archivo *jboss-aop.deployer/* o *jboss-aop-jdk50.deployer/* del directorio *jboss-40-install* de la distribución de Jboss AOP, dependiendo del JDK que se tenga instalado.



Integración de Jboss AOP con Jboss AS en una aplicación Web

Jboss 4.x y JDK 5

El primer paso es ingresar a la ruta donde se descomprimió la distribución de Jboss AS y dentro del directorio jboss-aop-jdk50.deployer que es el archivo que se reemplazó, el cual contiene el framewok aop tal como se explicó anteriormente, ver la siguiente gráfica:



Aquí se encuentra un archivo llamado *jboss-service.xml* el cual contiene algunos MBeans que desarrollan y administran el AOP Framework, por defecto Jboss AS no hace manipulación del Byte-code de los archivos AOP en tiempo de carga, se puede habilitar esta caracteristica editando la línea 24 y habilitando el atributo *EnableLoadtimeWeaving* a true, tal como se muestra a continuación:

```
<mbean code="org.jboss.aop.deployment.AspectManagerServiceJDK5"
      name="jboss.aop:service=AspectManager">
      <attribute name="EnableLoadtimeWeaving">true</attribute>
      <!-- only relevant when EnableLoadtimeWeaving is true.
           When transformer is on, every loaded class gets
```

```
            transformed.  If AOP can't find the class, then it
            throws an exception.  Sometimes, classes may not have
            all the classes they reference.  So, the Suppressing
            is needed.  (i.e. Jboss cache in the default
configuration -->
      <attribute
name="SuppressTransformationErrors">true</attribute>
      <attribute name="Prune">true</attribute>
      <attribute name="Include">org.jboss.test,
org.jboss.injbossaop</attribute>
      <attribute name="Exclude">org.jboss.</attribute>
      <!-- This avoids instrumentation of hibernate cglib enhanced
proxies
      <attribute name="Ignore">*$$EnhancerByCGLIB$$*</attribute> -
->
      <attribute name="Optimized">true</attribute>
      <attribute name="Verbose">false</attribute>
   </mbean>
```

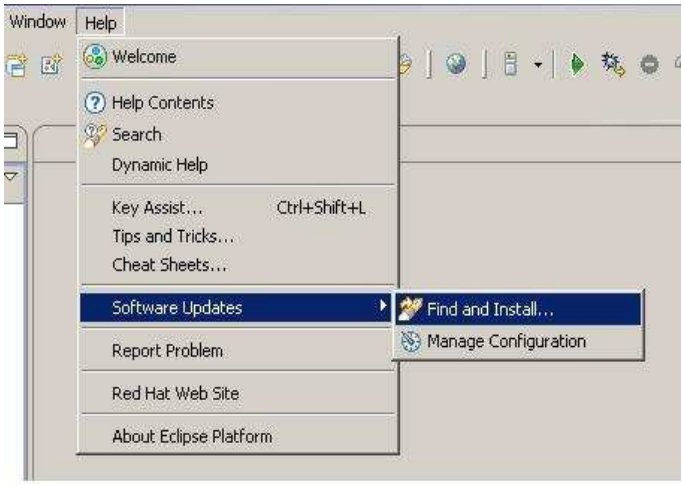
El siguiente paso es copiar el *pluggable-instrumentor.jar* del directorio lib-50 de la distribución Jboss AOP al directorio bin de Jboss AS. Finalmente se debe editar el run.bat de Jboss AS y se debe adicionar la siguiente variable de ambiente a JAVA_OPTS.

```
set JAVA_OPTS=%JAVA_OPTS% -Dprogram.name=%PROGNAME% -
javaagent:pluggable-instrumentor.jar
```

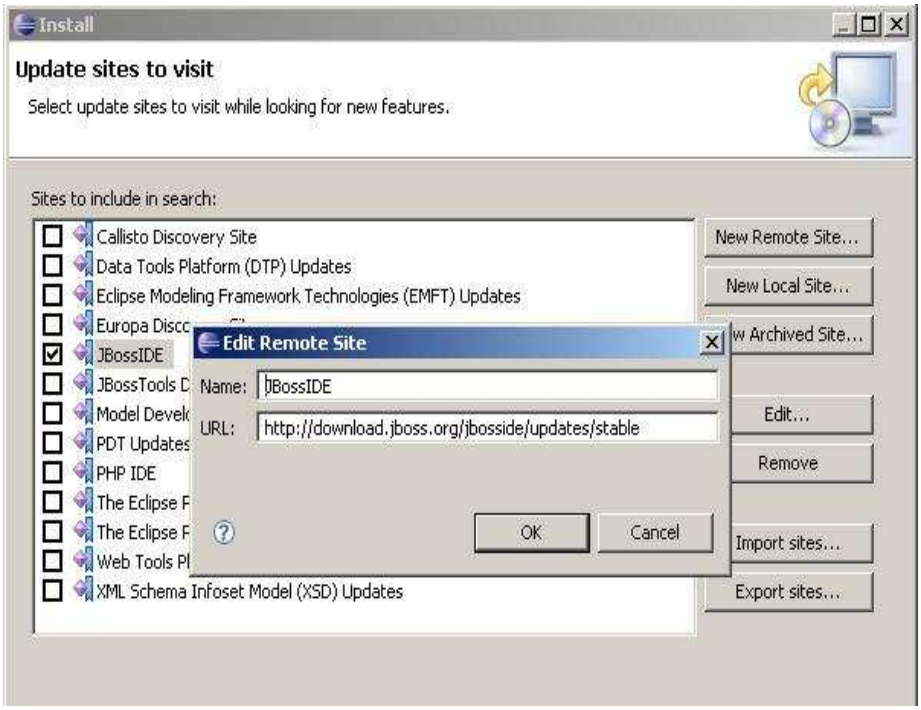
Con esto queda configurado y listo para correr Jboss AOP dentro de Jboss AS.

Instalación del IDE para Eclipse.

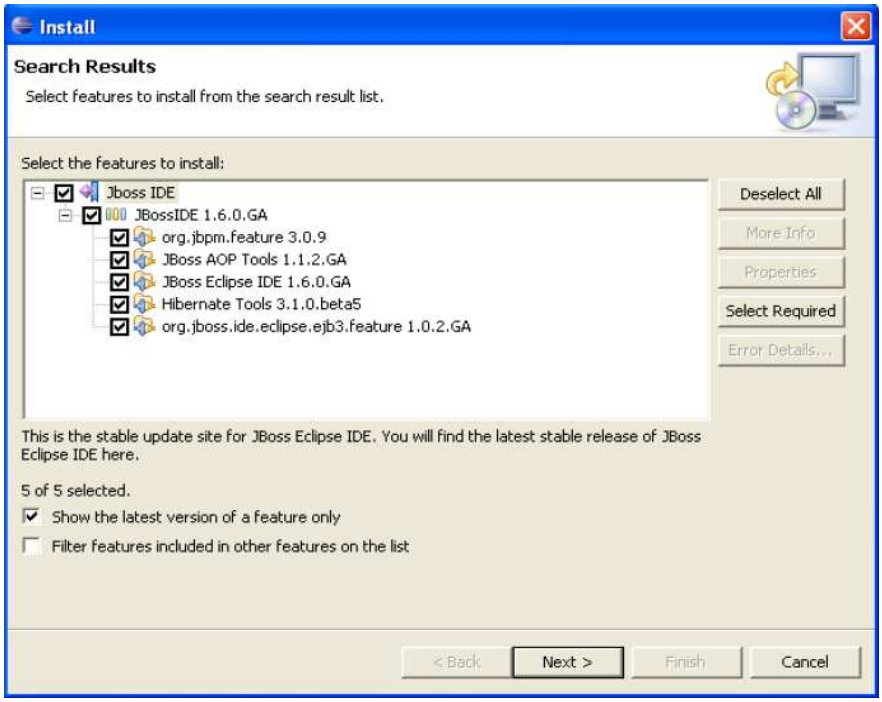
La instalación del entorno para Eclipse se efectúa de la misma forma que se instala un plugin para eclipse, descargando la versión del sitio oficial de Jboss AOP llamada *JBossIDE-AOP-Developer-1.1.2.GA.zip* se descomprime este archivo y se agregan las carpetas contenidas dentro de plugins y features a las respectivas carpetas del Eclipse que se tenga instalado. Tambien se puede descargar el IDE directamente del entorno de Eclipse a través de la opción Help->Softwares Updates->Find And Install.



El wizard permite adicionar una nueva URL de descargas, en esta caso Jboss IDE, como se muestra en la siguiente gráfica:



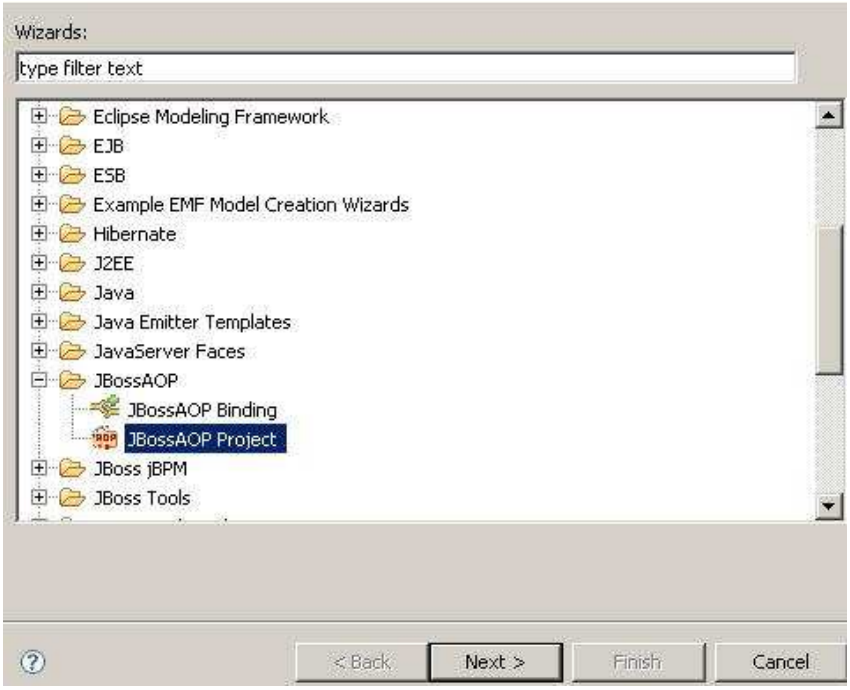
Se presiona siguiente y automáticamente se hacen las búsquedas de las actualizaciones de Jboss AOP IDE.



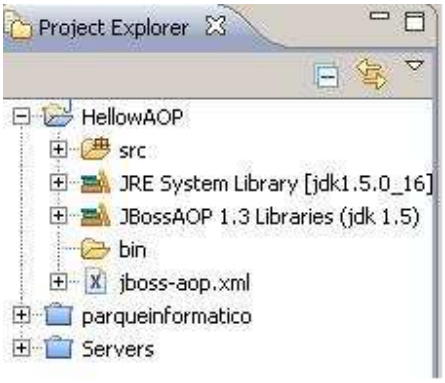
Como se crea un Proyecto con Jboss AOP IDE

En primer lugar se selecciona File->New->Other

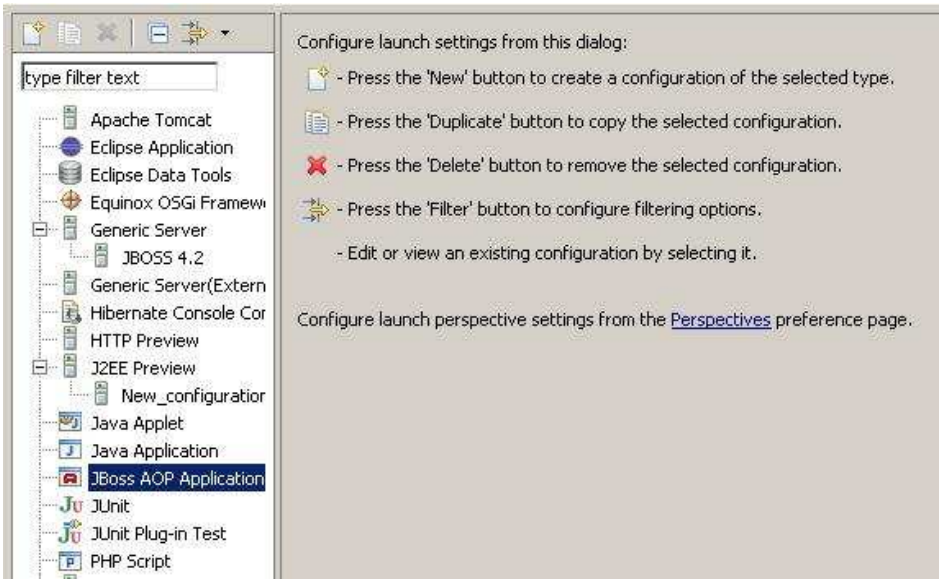
Luego se selecciona del wizard la opción Jboss AOP Project tal como se muestra a continuación.



Se coloca el nombre del proyecto, se selecciona el JDK a utilizar y se da click en finish, automáticamente se muestra en el árbol de directorios de Eclipse el proyecto creado con las librerías cargadas y el archivo jboss-aop.xml, en el cual se definen los pointcuts, advices, clases y métodos afectados, etc.



Una vez creadas las clases del sistema de componentes y las clases interceptoras del sistema de aspectos se puede correr la aplicación como una aplicación java normal, para esto se abre el cuadro de diálogo *Run Dialog* y se selecciona la opción Jboss AOP application tal como se indica a continuación.



Configuración XML

JBoss AOP resuelve la aplicación de advices y Pointcuts en tiempo de ejecución a través del archivo jboss-aop.xml, este archivo define el tejido entre las clases del código fuente con las clases de los aspectos. Cuando se corre una aplicación AOP fuera de Jboss AS, una forma de que el framework Jboss AOP resuelva los archivos XML es colocando un archivo xml con el sufijo *-aop.xml en el directorio deploy, otra forma es encapsulando la aplicación en un archivo .Jar y dentro del directorio META-INF/jboss-aop.xml colocar el archivo xml, a este archivo .Jar se le debe cambiar la extensión por .aop y colocarlo dentro del directorio deploy.

Construyendo y compilando aspectos en Java

Modos de Instrumentación.

Jboss AOP trabaja instrumentando las clases que se van a ejecutar por medio de modificaciones del bytecode para dar información extra a dichas clases sobre las librerias de AOP, Jboss permite dos tipos de instrumentación:

Precompiled – Las clases son instrumentadas en una compilación AOP separada antes de que estas sean ejecutadas.

LoadTime – Las clases son instrumentadas cuando estas se cargan por primera vez.

A continuación se detalla la manera como se precompilan las clases con AOP precompiler.

***Integración mediante ANT Builder*.**

Jboss Aop viene con la herramienta ANT para precompilar las clases con el precompilador aop. La manera para precompilar las clases mediante el builder de ANT es a través de un archivo XML llamado *builder.xml* el cual se describe a continuación:

```
<?xml version="1.0" encoding="UTF-8"?>
<project default="compile" name="JBoss/AOP">
<target name="prepare">
```

Se define el directorio fuente, y el directorio para las clases a compilar. Sin problema se puede indicar el mismo directorio.

```
<property name="src.dir" value="PATH TO YOUR SOURCE DIR">
<property name="classes.dir" value="PATH TO YOUR DIR FOR
COMPILED CLASSES">
```

Se incluyen los path's para las librerias o .jars AOP, estos son comunes a cualquier JDK.

```
<path id="javassist.classpath">
<pathelement path="../../../javassist.jar"/>
</path>
<path id="trove.classpath">
<pathelement path="../../../trove.jar"/>
</path>
<path id="concurrent.classpath">
<pathelement path="../../../concurrent.jar"/>
</path>
<path id="jboss.common.classpath">
<pathelement path="../../../jboss-common.jar"/>
</path>
<path id="lib.classpath">
```

```
<path refid="javassist.classpath"/>
<path refid="trove.classpath"/>
<path refid="jboss.aop.classpath"/>
<path refid="jboss.common.classpath"/>
<path refid="concurrent.classpath"/>
</path>
```

En caso de que se usen Anotaciones Java (JDK 5) se debe tener en cuenta el siguiente fragmento XML.

```
<!-- JDK version 1.5 -->
<!-- Do not include this if using JDK 1.4.2!!!! -->
<path id="jboss.aop.classpath">
<pathelement path="../../../jboss-aop-jdk15.jar"/>
</path>
<!-- JDK version 1.5 - END -->
```

Ahora se fijan los classpath totales de todas las librerías necesarias

```
<path id="classpath">
<path refid="lib.classpath"/>
<path refid="jboss.aop.classpath"/>
</path id="classpath">
```

Se define la herramienta del precompilador de ANT *org.jboss.aop.ant.AopC*

```
<taskdef name="aopc" classname="org.jboss.aop.ant.AopC"
classpathref="jboss.aop.classpath"/>
</target>

<target name="compile" depends="prepare">
```

Compilar los archivos (del directorio fuente hacia el directorio de las clases compiladas:

```
<javac srcdir="${src.dir}"
destdir="${classes.dir}"
debug="on"
deprecation="on"
optimize="off"
includes="**">
<classpath refid="classpath"/>
</javac>
```

Ahora se puede usar la herramienta ANT aop precompiler,

```
<aopc compilerclasspathref="classpath" verbose="true">
<classpath path="${classes.dir}"/>
<src path="${classes.dir}"/>
<include name="**/*.class"/>
<aoppath path="jboss-aop.xml"/>
<aopclasspath path="aspects.jar"/>
</aopc>
</target>
</project>
```

Línea de Comandos

Para ejecutar el aop precompiler en la línea de comandos se necesitan todos los aop jars sobre el classpath, así se esta instrumentando los archivos class sin embargo estos archivos podrían necesitar correr sobre el classpath de java, incluyéndose así mismos o el precompilador no se habilitará para ejecutarse.

ANEXO B: MANUAL DE USUARIO

Configurando Jboss AS para MySQL

Los pasos para cambiar la base (en HSQL) por defecto de JBoss a MySQL son los siguientes:

I. Sustituyendo el "DefaultDS" para que apunte ahora a una base en MySQL

1. mysql-ds.xml
   - Copiar el archivo mysql-ds.xml localizado en [JBOSS_HOME]/docs/examples/jca al folder de deploy.
   - Modificarlo de manera que contenga los parametros de la base de datos para jboss (jbossdb en nuestro ejemplo) así como el nuevo nombre del data source:

```
<jndi-name>DefaultDS</jndi-name>
<connection-
url>jdbc:mysql://localhost:9097/jbossdb</connection-
url>
<driver-class>com.mysql.jdbc.Driver</driver-class>
<user-name>root</user-name>
<password>lolo</password>
```

   - Eliminar el archivo hsqldb-ds.xml del directorio deploy.

2. standardjaws.xml
   - Editar el archivo para cambiar el tipo de mapeo del datasource de Hypersonic SQL a mySQL.

```
<datasource>java:/DefaultDS</datasource>
<type-mapping>mySQL</type-mapping>
```

3. JMS configuration descriptors
   - Eliminar el archivo hsqldb-jdbc2-service.xml del directorio deploy/jms.

- Copiar el archivo mysql-jdbc2-service.xml del folder docs/example/jms al deploy/jms.
- Editar el archivo mysql-jdbc2-service.xml y cambiar el datasource.

```
<mbean code="org.jboss.mq.pm.jdbc2.PersistenceManager"
       name="jboss.mq:service=PersistenceManager">
  <depends optional-attribute-
name="ConnectionManager">jboss.jca:service=DataSourceBindi
ng,name=DefaultDS</depends>
                  <attribute name="SqlProperties">
```

Finlamente se debe descargar el driver *mysql-connector-java-5.1.6-bin.jar* y copiarlo al directorio \jboss-4.2.2.GA\server\default\lib lib.

Una vez configurado Jboss para MySQL se deben crear las tablas utilizadas en la base de datos parqueinformaticodb, para esto se debe correr el siguiente script:

```
# HeidiSQL Dump
#
# --------------------------------------------------------
# Host:                 127.0.0.1
# Database:             parqueinformaticodb
# Server version:       5.0.41-community-nt
# Server OS:            Win32
# Target-Compatibility: Standard ANSI SQL
# HeidiSQL version:     3.2 Revision: 1129
# --------------------------------------------------------

/*!40100 SET CHARACTER SET latin1;*/
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ANSI';*/
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;*/
```

```
#
# Database structure for database 'parqueinformaticodb'
#

CREATE DATABASE /*!32312 IF NOT EXISTS*/ "parqueinformaticodb" /*!40100
DEFAULT CHARACTER SET latin1 */;

USE "parqueinformaticodb";

#
# Table structure for table 'calificaciones'
#

CREATE TABLE /*!32312 IF NOT EXISTS*/ "calificaciones" (
  "codigo" int(10) unsigned NOT NULL auto_increment,
  "nombre" varchar(50) default NULL,
  "calificacion" decimal(50,0) unsigned default NULL,
  PRIMARY KEY  ("codigo")
) /*!40100 DEFAULT CHARSET=latin1*/;

#
# Dumping data for table 'calificaciones'
#

# (No data found.)
#
# Table structure for table 'eventos'
#
CREATE TABLE /*!32312 IF NOT EXISTS*/ "eventos" (
  "id" int(10) unsigned zerofill NOT NULL auto_increment,
```

```
  "autor" varchar(255) default NULL,
  "titulo" varchar(255) default NULL,
  "fecha" varchar(10) default NULL,
  "encabezado" varchar(255) default NULL,
  "contenido" varchar(1000) default NULL,
  PRIMARY KEY  ("id")
) /*!40100 DEFAULT CHARSET=latin1*/;
#
# Dumping data for table 'eventos'
#
# (No data found.)
#
# Table structure for table 'noticias'
#

CREATE TABLE /*!32312 IF NOT EXISTS*/ "noticias" (
  "id" int(10) NOT NULL auto_increment,
  "autor" varchar(255) NOT NULL default '',
  "titulo" varchar(255) NOT NULL default '',
  "fecha" varchar(10) NOT NULL default '',
  "encabezado" varchar(255) NOT NULL,
  "contenido" longtext NOT NULL,
  PRIMARY KEY  ("id")
) /*!40100 DEFAULT CHARSET=latin1*/;

#
# Dumping data for table 'noticias'
#
# (No data found.)
#
# Table structure for table 'usuarios'
```

```
#

CREATE TABLE /*!32312 IF NOT EXISTS*/ "usuarios" (
  "id" int(5) unsigned zerofill NOT NULL auto_increment,
  "nombre" varchar(30) default NULL,
  "apellidos" varchar(30) default NULL,
  "login" varchar(30) default NULL,
  "passwd" varchar(30) default NULL,
  "email" varchar(30) default NULL,
  "tipo" varchar(30) default NULL,
  PRIMARY KEY  ("id")
) /*!40100 DEFAULT CHARSET=latin1*/;

#
# Dumping data for table 'usuarios'
#

LOCK TABLES "usuarios" WRITE;
/*!40000 ALTER TABLE "usuarios" DISABLE KEYS;*/
REPLACE INTO "usuarios" ("id", "nombre", "apellidos", "login", "passwd", "email",
"tipo") VALUES
        ('1235','OMAR','TREJO','omar','omar','omartn@gmail.com','administrador');
REPLACE INTO "usuarios" ("id", "nombre", "apellidos", "login", "passwd", "email",
"tipo") VALUES
        ('1236','Alejandra
Maria','Narvaez','aleja','aleja','drakexle@hotmail.com','usuario');
/*!40000 ALTER TABLE "usuarios" ENABLE KEYS;*/
UNLOCK TABLES;
/*!40101 SET SQL_MODE=@OLD_SQL_MODE;*/
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;*/
```

Estructura de directorios para el prototipo

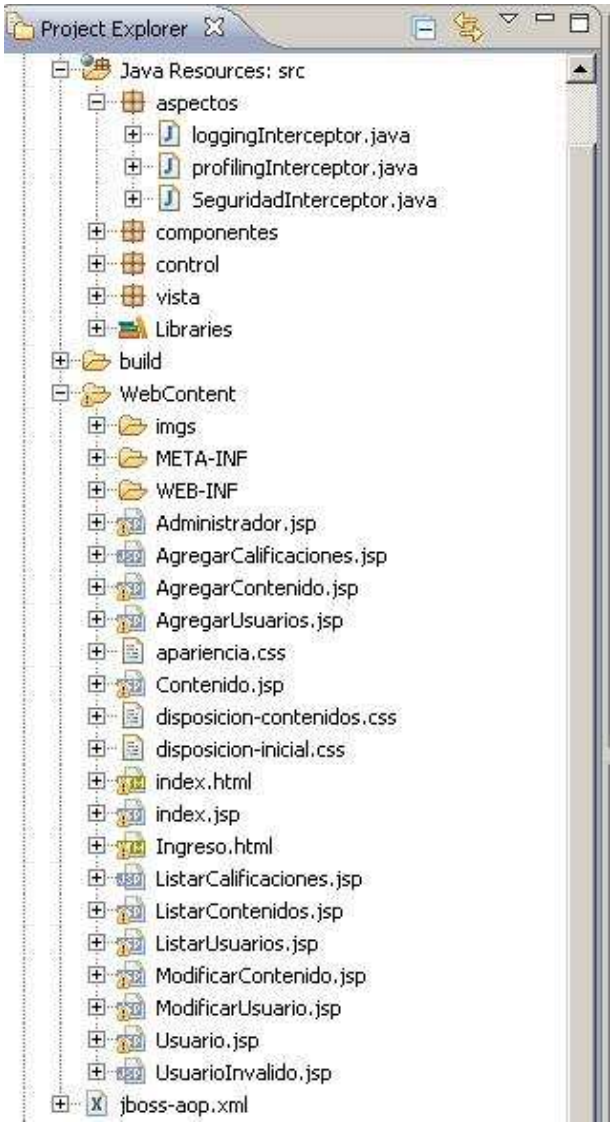
A continuación se muestra la estructura de directorios para poder hacer Builder con ANT y empaquetar la aplicación.

/ parqueinformatico /output/	Es la carpeta donde se guarda la salida después de hacer build de la aplicación por medio de ant.
/ parqueinformatico /src/	Este directorio contiene el código fuente de la aplicación dividido en cuatro capas o paquetes (MVC + paquete de aspectos), mediante este directorio se puede hacer la generación de los archivos .jar, .war, .aop, al ejecutar el build.xml con el ant.
/parqueinformatico/build.xml	Contienen la información de los paths y parámetros para hacer builder con ANT
/ parqueinformatico /deploy.txt	Contiene el siguiente comando **ant deploy-basic-lt-war-in-jar** para hacer build que se ejecuta desde una consola de Windows.

La siguiente tabla muestra la estructura de directorios para el código fuente de la aplicación construida en Eclipse 3.3.

/ parqueinformatico /WebContent /WEB-INF/…	representa la estructura común para una aplicación web, desarrollada en eclipse. Dentro de la carpeta WEB-INF se puede encontrar el archivo web.xml
/ parqueinformatico /jboss-aop.xml	Contiene toda la información de la configuración de los aspectos al interior de la aplicación Web para efectuar el tejido entre aspectos y componentes.

La siguiente figura muestra como se encuentra organizado el prototipo dentro del Project Explorer de Eclipse, el lugar que ocupa el paquete (Capa) de aspectos y el archivo jboss-aop.xml, donde se definen las clases y metodos intervenidos por los aspectos.



Configuración del Archivo build.xml para utilizar la herramienta ANT

```
<?xml version="1.0" encoding="UTF-8"?>
<project default="compile" name="JBoss/AOP">
<target name="prepare">
<property name="src.dir" value="PATH TO YOUR SOURCE DIR">
<property name="classes.dir" value="PATH TO YOUR DIR FOR
COMPILED CLASSES">
<path id="javassist.classpath">
<pathelement path="../../../javassist.jar"/>
</path>
<path id="trove.classpath">
<pathelement path="../../../trove.jar"/>
</path>
<path id="concurrent.classpath">
<pathelement path="../../../concurrent.jar"/>
</path>
<path id="jboss.common.classpath">
<pathelement path="../../../jboss-common.jar"/>
</path>
<path id="lib.classpath">
<path refid="javassist.classpath"/>
<path refid="trove.classpath"/>
<path refid="jboss.aop.classpath"/>
<path refid="jboss.common.classpath"/>
<path refid="concurrent.classpath"/>
</path>
<path id="classpath">
<path refid="lib.classpath"/>
<path refid="jboss.aop.classpath"/>
</path id="classpath">
<taskdef name="aopc" classname="org.jboss.aop.ant.AopC"
classpathref="jboss.aop.classpath"/>
</target>

<target name="compile" depends="prepare">
```

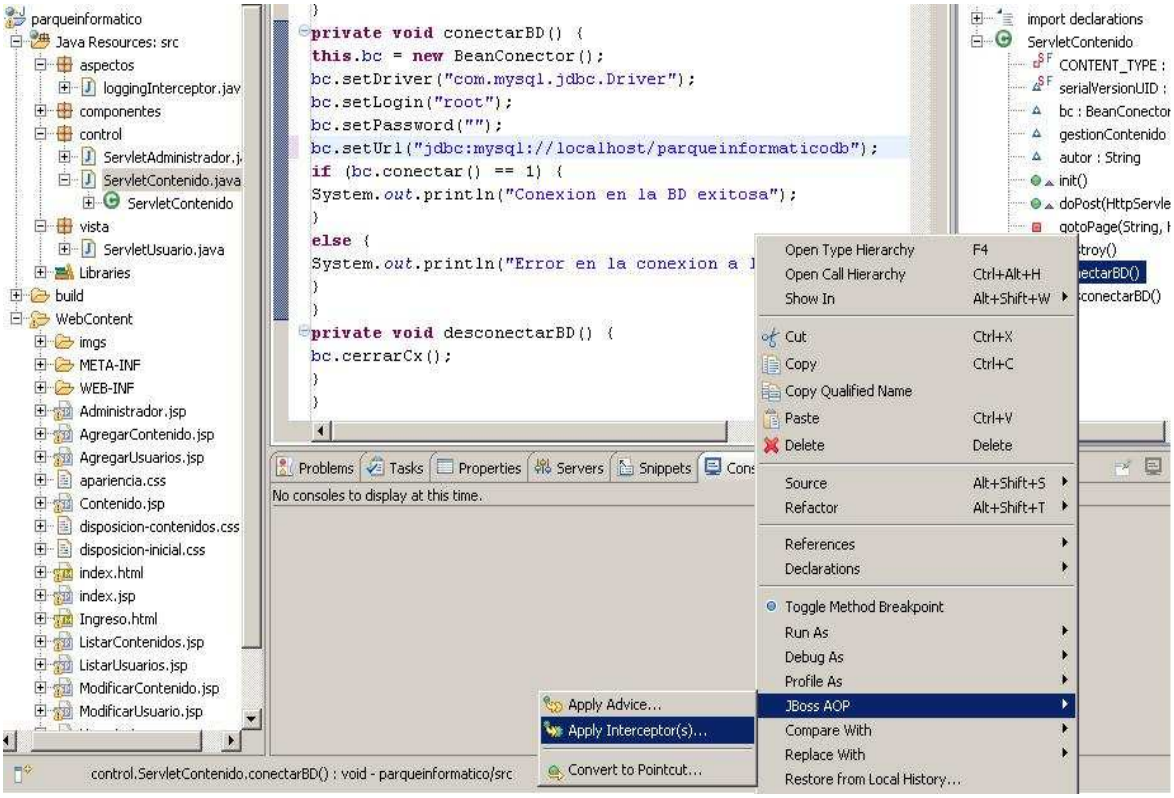
```
<javac srcdir="${src.dir}"
destdir="${classes.dir}"
debug="on"
deprecation="on"
optimize="off"
includes="**">
<classpath refid="classpath"/>
</javac>

<aopc compilerclasspathref="classpath" verbose="true">
<classpath path="${classes.dir}"/>
<src path="${classes.dir}"/>
<include name="**/*.class"/>
<aoppath path="jboss-aop.xml"/>
<aopclasspath path="aspects.jar"/>
</aopc>
</target>
</project>
```

Aplicación empaquetada

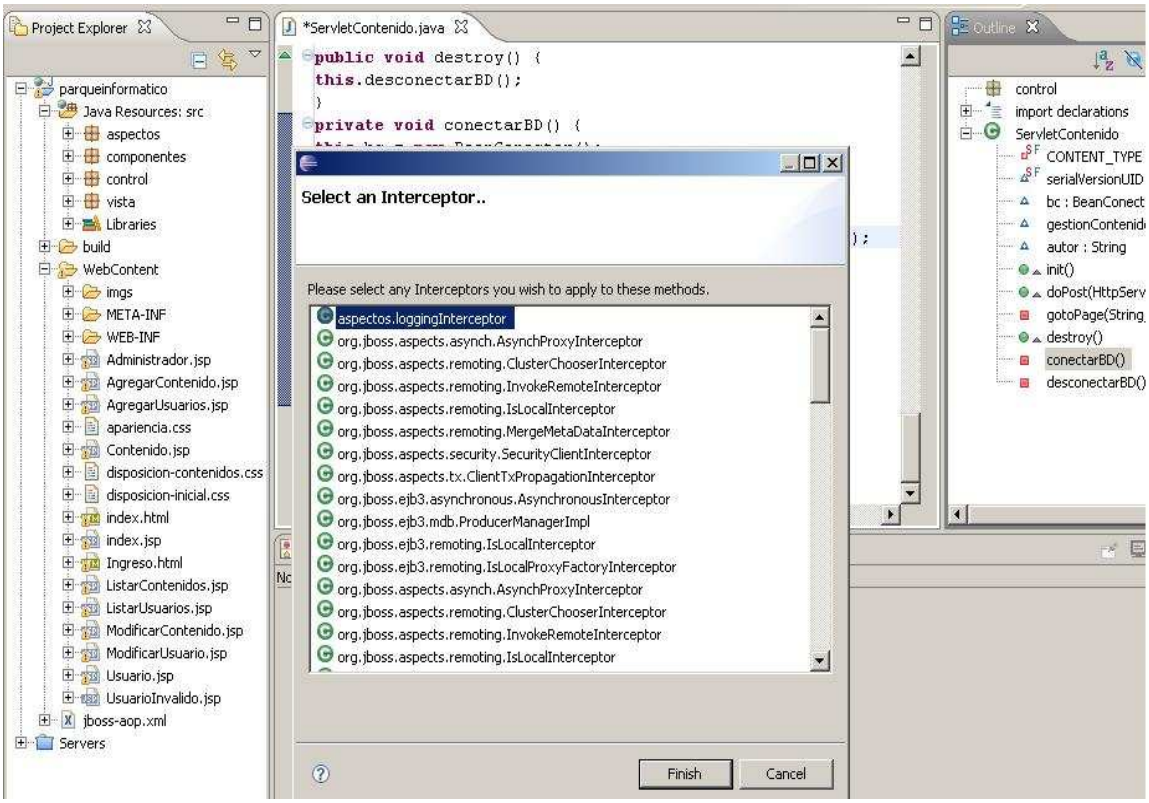
El archivo parqueinformatico.jar contiene un archivo parqueinformatico.war; en el cual se encuentran todos los archivos propios de la aplicación Web en si (JSP's, CSS's y HTML's), un directorio llamado META-INF propio de la estructura de Eclipse y un archivo llamado parqueinformatico.aop, el cual contiene todas las clases compiladas tanto de componentes como de aspectos, además contiene un directorio META-INF al igual que en el archivo .war, la diferencia es que dentro de este directorio esta incluido el archivo jboss-aop.xml, el cual especifica los puntos de corte y la clase que se aplica en dichos puntos.

Ejemplos de cómo aplicar Interceptores dentro del entorno de Eclipse para el caso del Prototipo

La siguiente figura muestra como la manera como se aplica el interceptor al método SerrvletAdministrador.ConectarDB mostrado en el outline de la parte derecha del entorno por medio del IDE de Jboss AOP, esto con el fin de familiarizarse con el entorno de desarrollo.



Una vez se da clic en ApplyInterceptor el entorno muestra una lista con todos los Interceptores implementados y los que vienen junto con Jboss AOP



Se selecciona *aspectos.loggingInterceptor* para el caso del aspecto de logging y se da clic en finish, Eclipse se encarga automáticamente de marcar el método donde se aplicó el Interceptor y de agregar la configuración necesaria al archivo jboss-aop-xml

A continuación se muestran algunos pantallazos del prototipo Web del Parque Informático, es importante aclarar que la ejecución de los aspectos para este caso solo se puede comprobar a través de la consola de Eclipse, para el caso del aspecto de Logging se realiza la impresión de un mensaje de confirmación cuando los servlets de administración se conecten a la base de datos.

Interfaz de ingreso a la zona de administración

Interfaz de administración

Interfaz para agregar Usuario

Interfaz para Listar Usuarios