Trabajo Final de Máster 19 de enero 2014

Ampliación de funcionalidades para KPAX

ANEXO: Descripción del servicio web Kpax

Descripción del servicio web y su funcionalidad general

La siguiente imagen es una vista del entorno de desarrollo Eclipse, con los componentes del servicio web srvKpax:

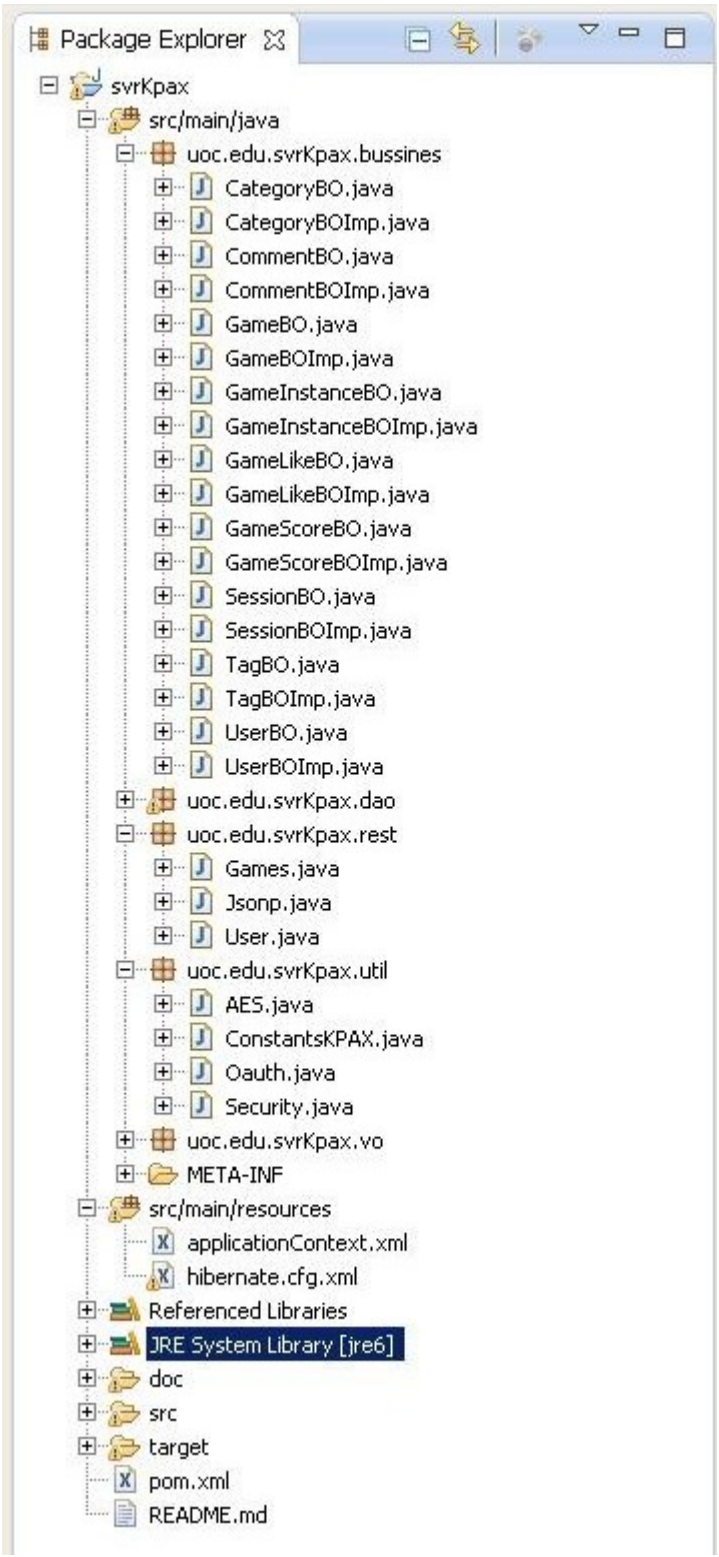

Figura 1 Vista del proyecto Java svrKpax

El servicio web está definido bajo la carpeta target. Hay además dos ficheros auxiliares que debemos considerar:

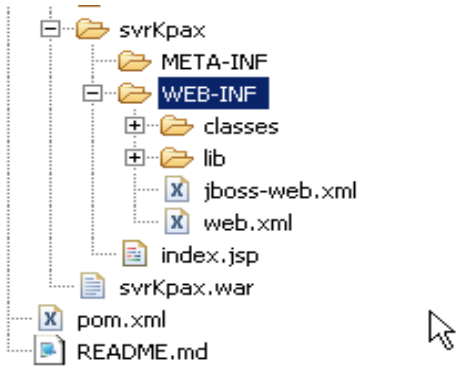

Figura 2 Vista del directorio WEB-INF

por una parte pom.xml, fichero de configuración de Maven en el que se define el proyecto. http://Maven.Apache.org/guides/introduction/introduction-to-the-pom.html#Minimal_POM Este fichero se va modificando a medida que se incorporan dependencias al proyecto, pero requiere una configuración mínima. En nuestro caso la configuración destacada tiene los siguientes puntos:

Definición del nombre del proyecto. Esta sería la configuración mínima: stacada genera cuando se define el proyecto a medida que se

```
<project                              xmlns="http://Maven.Apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                      xsi:schemaLocation="http://Maven.Apache.org/POM/4.0.0
http://Maven.Apache.org/Maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>uoc.edu</groupId>
    <artifactId>svrKpax</artifactId>
    <packaging>war</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>svrKpax Maven Webapp</name>
    <url>http://Maven.Apache.org</url>
```

Se define el repositorio público del que se descargarán las dependencias:

```
    <repositories>
          <repository> <id>Maven2-repository.java.net</id> <name>Java.net
Repository
             for Maven</name> <url>http://download.java.net/Maven/2/</url>
<layout>default</layout>
        </repository>
     </repositories>

    <dependencies>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-annotations</artifactId>
            <version>3.5.6-Final</version>
            <scope>provided</scope>
        </dependency>

        <dependency>
              <groupId>com.sun.xml.security</groupId>
              <artifactId>Apache-xmlsec</artifactId>
```

```
            <version>1.0.1</version>
        </dependency>
…….<!--resto de dependencias -->
 </dependencies>
```

Se define el tipo de perfil utilizado, en nuestro caso el proyecto sólo usara un repositorio local:

```
    <profiles>
        <profile>
            <id>local</id>
            <activation>
                <property>
                    <name>env</name>
                    <value>local</value>
                </property>
            </activation>
            <properties>
                <!-- web.xml parameters -->
                                                    <configFilePath>${basedir}/src/main/resources/config/config.properties</configFilePath>
```

Se definen los parámetros de despliegue utilizados, así como la ruta del servidor de aplicaciones, el directorio donde se deja el fichero war de la aplicación y los plugins para generar el fichero de despliegue:

```
                <!--deploy parameters -->
                        <JBossdeployhome>\Archivos de programa\JBoss-4.2.3.GA\server\default\deploy</JBossdeployhome>
            </properties>
            <build>
                <plugins>
                    <plugin>
                        <groupId>org.Apache.Maven.plugins</groupId>
                        <artifactId>Maven-antrun-plugin</artifactId>
                        <executions>
                            <execution>
                                <phase>package</phase>
                                <configuration>
                                    <tasks>
                                                                <copy file="${basedir}/target/svrKpax.war" todir="${JBossdeployhome}">
                                        </copy>
                                    </tasks>
                                </configuration>
                                <goals>
                                    <goal>run</goal>
                                </goals>
                            </execution>
                        </executions>
                    </plugin>
                </plugins>
            </build>
            <modules>
            </modules>
        </profile>
                                                        </profiles>
<build>
        <finalName>svrKpax</finalName>
    </build>
    <reporting>
        <plugins>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>cobertura-Maven-plugin</artifactId>
```

```
                <version>2.5.1</version>
            </plugin>
        </plugins>
    </reporting>
</project>
```

Por otra parte ApplicationContext.xml en el directorio src/main/sources, fichero de definición de dependencias de clases para la gestión de los objetos Bean por parte de Spring.
A continuación mostramos fragmentos de este fichero, donde podemos ver :

una zona de cabecera:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
    xmlns:context="http://www.springframework.org/schema/context"
xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
    http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">
```

la zona donde se definen las clases que se ocuparán de gestionar los beans leyendo las java annotations:

```
    <!-- post-processors for all standard config annotations -->
    <bean class="org.springframework.orm.jpa.support.
PersistenceAnnotationBeanPostProcessor" />

    <bean id="sessionFactory" class="org.springframework.orm.
hibernate3.annotation.AnnotationSessionFactoryBean">
```

la ubicación del fichero de configuración para hibernate:

```
     <property name="configLocation" value="classpath:hibernate.cfg.xml" />
     <property name="packagesToScan" value="uoc.edu.svrKpax.vo" />
    </bean>
```

la declaración del componente administrador de transacciones que permitirá enviar las peticiones desde los objetos bussines a los objetos DAO y que gestionará las aperturas y cierres de sesión hibernate:

```
<!-- Spring's hibernate transaction manager -->
    <bean id="transactionManager"
    class="org.springframework.orm.hibernate3.HibernateTransactionManager">
        <property name="sessionFactory" ref="sessionFactory" />
    </bean>
```

las relaciones entre los objetos o beans instanciados a partir de las clases. Existen dos tipos de declaraciones, para objetos de acceso a datos Dao y para objetos de la capa Bussines:

```
    <!-- DAOS -->
    <bean id="sDao" class="uoc.edu.svrKpax.dao.SessionDaoImpl">
        <property name="sessionFactory" ref="sessionFactory" />
    </bean>
```

```
     <bean id="gDao" class="uoc.edu.svrKpax.dao.GameDaoImpl">
        <property name="sessionFactory" ref="sessionFactory" />
    </bean>
…..................<!-- Resto de declaraciones -->
<!-- GAMES -->
    <bean id="gBo" class="uoc.edu.svrKpax.bussines.GameBOImp">
        <property name="sBo" ref="sBo" />
        <property name="gDao" ref="gDao" />
        <property name="gvDao" ref="gvDao" />
        <property name="cDao" ref="cDao" />
    </bean>
…..................<!-- Resto de declaraciones -->
</beans>
```

Si tomamos como ejemplo la declaración de la clase GameBoImp, vemos que algunos de sus métodos hacen referencia a las clases listadas como propiedades, por lo que al crearla o modificarla, es necesario, si procede, incluir estas dependencias.

La información de este fichero se completa con anotaciones java, por ejemplo, la clase Games.java del paquete REST, en el que se definen los métodos ofrecidos por el servicio Web, lleva numerosos @inject con las clases necesarias para cada método definido.

Para “ver” el sistema en acción, tal cual se nos proporciona y a medida que vamos realizando cambios en el código y compilando, el único método que nos da información precisa es la consola del servidor JBoss. Hemos obtenido bastantes datos acerca del comportamiento de los componentes del sistema del fichero server.log, situado en el directorio C:\JBoss\server\default\log. Este log se sobreescribe a cada nueva ejecución del servidor y es editable con el servidor bajado. Su extensión es considerable, por lo que mostraremos pequeños fragmentos con detalles relevantes.

En las fases finales del arranque de JBoss, cuando los componentes gestores ya están cargados y funcionando, comenzamos a ver el proceso de despliegue de los beans:

Primero se preinstancian, a continuación se crea el bean raíz, sessionfactory como instancia compartida. La instancia sessionfactory resolverá los fallos de referecias circulares entre las clases o beans en caso de producirse y registrará cada uno de los beans desplegados :

```
2013-12-19 14:36:28,812 INFO
[org.springframework.beans.factory.support.DefaultListableBeanFactory] Pre-
instantiating singletons in
org.springframework.beans.factory.support.DefaultListableBeanFactory@92c11b
:  defining beans
[org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor
#0,sessionFactory,
transactionManager,uDao,sDao,rDao,gDao,lDao,iDao,scDao,cDao,comDao,gvDao,tD
ao,uBo,sBo,gBo,lBo,iBo,scBo,tagBo,catBo,comBo,oUser]; root of factory
hierarchy

2013-12-19 14:36:28,812 DEBUG
[org.springframework.beans.factory.support.DefaultListableBeanFactory]
Creating instance of bean 'sessionFactory'
2013-12-19 14:36:29,031 DEBUG
[org.springframework.beans.factory.support.DefaultListableBeanFactory]
Eagerly caching bean 'sessionFactory' to allow for resolving potential
circular references
```

Se produce la carga de componentes hibernate. Se ejecuta un escaneo de los beans leyendo las anotaciones java de los ficheros VO para tener en cuenta los tipos de columnas, si son valores únicos o no, etc.

```
2013-12-19 14:36:29,187 INFO  [org.hibernate.cfg.annotations.Version]
Hibernate Annotations 3.2.1.GA

2013-12-19 14:36:29,234 INFO  [org.hibernate.cfg.Environment] Hibernate
3.2.4.sp1

2013-12-19 14:36:31,140 DEBUG [org.hibernate.cfg.AnnotationConfiguration]
Execute first pass mapping processing
```

Este es el procesado del campo idcategory de la tabla Category:

```
2013-12-19 14:36:31,296 DEBUG [org.hibernate.cfg.AnnotationBinder]
Processing uoc.edu.svrKpax.vo.Category property annotation

2013-12-19 14:36:31,328 DEBUG [org.hibernate.cfg.AnnotationBinder]
Processing annotations of uoc.edu.svrKpax.vo.Category.idCategory

2013-12-19 14:36:31,343 DEBUG [org.hibernate.cfg.AnnotationBinder]
idCategory is an id

2013-12-19 14:36:31,343 DEBUG
[org.hibernate.cfg.annotations.SimpleValueBinder] building SimpleValue for
idCategory

2013-12-19 14:36:31,343 DEBUG
[org.hibernate.cfg.annotations.PropertyBinder] Building property idCategory

2013-12-19 14:36:31,359 DEBUG [org.hibernate.cfg.AnnotationBinder] Bind @Id
on idCategory
```

En una segunda pasada, se leen las foreing keys y las relaciones entre tablas como en el caso de la tabla realm que contiene una relación manytomany con la tabla idgameinstance

```
2013-12-19 14:36:31,671 DEBUG [org.hibernate.cfg.CollectionSecondPass]
Second pass for collection: uoc.edu.svrKpax.vo.User.realm
2013-12-19 14:36:31,671 DEBUG
[org.hibernate.cfg.annotations.CollectionBinder] Binding as ManyToMany:
uoc.edu.svrKpax.vo.User.realm
2013-12-19 14:36:31,687 DEBUG [org.hibernate.cfg.CollectionSecondPass]
Mapped collection key: idUser, element: idRealm
```

Posteriormente se lanzan consultas sql de prueba sobre todas las tablas analizadas:

```
2013-12-19 14:36:33,750 DEBUG
[org.hibernate.persister.entity.AbstractEntityPersister] Static SQL for
entity: uoc.edu.svrKpax.vo.GameInstance

2013-12-19 14:36:33,750 DEBUG
[org.hibernate.persister.entity.AbstractEntityPersister]  Version select:
select idGameInstance from GameInstance where idGameInstance =?
```

Se crean las instancias de las clases DAO y cada una se registra con sessionfactory para que no existan referencias circulares.

```
2013-12-19 14:36:34,125 DEBUG
[org.springframework.beans.factory.support.DefaultListableBeanFactory]
Creating shared instance of singleton bean 'uDao'

2013-12-19 14:36:34,125 DEBUG
[org.springframework.beans.factory.support.DefaultListableBeanFactory]
Eagerly caching bean 'uDao' to allow for resolving potential circular
references
```

```
2013-12-19 14:36:34,156 DEBUG
[org.springframework.beans.factory.support.DefaultListableBeanFactory]
Finished creating instance of bean 'uDao'
```

Se detectan las clases que proporcionan los recursos de la aplicación:

```
2013-12-19 14:36:35,078 ERROR [STDERR] dic 19, 2013 2:36:35 PM
com.sun.jersey.api.core.ScanningResourceConfig logClasses
Información: Root resource classes found:
  class uoc.edu.svrKpax.rest.Jsonp
  class uoc.edu.svrKpax.rest.User
  class uoc.edu.svrKpax.rest.Games
```

Finalmente se arranca la implementación Jersey para servicios web tipo rest y los beans correspondiente a las clases de la capa bussines.

```
2013-12-19 14:36:35,328 ERROR [STDERR] dic 19, 2013 2:36:35 PM
com.sun.jersey.server.impl.application.WebApplicationImpl _initiate
Información: Initiating Jersey application, version 'Jersey: 1.8 06/24/2011
12:17 PM'

2013-12-19 14:36:38,921 DEBUG
[org.springframework.beans.factory.support.DefaultListableBeanFactory]
Returning cached instance of singleton bean 'gBo'
```

Este es el mensaje de servidor arrancado con la aplicación svrKpax.war correctamente desplegada:

```
2013-12-19 14:36:43,140 INFO  [org.JBoss.system.server.Server] JBoss (MX
MicroKernel) [4.2.3.GA (build: SVNTag=JBoss_4_2_3_GA date=200807181439)]
Started in 54s:62ms
```

Durante la presentación en video volveremos a la consola de JBoss para observar las respuestas a las peticiones hechas desde los plugins de Elgg Kpax y Gameserver al servicio web.