Persistencia

Índice

1. Hibernate en el SIGEM

Para analizar el funcionamiento de SIGEM vamos a analizar su flujo de llamadas y los métodos declarados en los distintos paquetes y módulos que lo componen. Para ello vamos utilizar las funciones de búsqueda de Eclipse y comandos de búsqueda de Linux.

Comenzamos copiando en la MV linux el código fuente de SIGEM. Se encuentra en este enlace. Guardamos el fichero en la MV y lo descomprimimos:

```
tar xvf SIGEM_Fuentes.tar
```

Aparecerá la carpeta `SIGEM_Fuentes` en la que se encuentran todos los ficheros `*.java`, `*.properties` y `*.xml`. Ahora podemos buscar ficheros por su nombre utilizando la instrucción Unix `find`:

```
> cd SIGEM_Fuentes
> find . -name <patron>
```

El patrón puede ser cualquier expresión regular que contiene caracteres libres como `*` o `?`. Hay que tener en cuenta que el texto que escribamos es sensible a mayúsculas y minúsculas.

De los distintos módulos que forman parte del SIGEM, sólo se utiliza Hibernate en el proyecto `SIGEM_RegistroPresencialIEJar`. Para comprobarlo, recordemos que el fichero de configuración de Hibernate siempre tiene que llamarse `hibernate.cfg.xml`. Lo buscamos en los fuentes:

```
> find . -name hibernate*
./SIGEM_RegistroPresencialIEJar/src/hibernate.cfg.xml
```

Podemos buscar también todos los ficheros de mapeado con la siguiente instrucción (si se ha seguido la convención de terminar sus nombres con la extensión `.hbm.xml`):

```
> find . -name *hbm.xml
./SIGEM_RegistroPresencialIEJar/src/com/ieci/tecdoc/common/invesdoc/Idocxnextid.hbm.x
./SIGEM_RegistroPresencialIEJar/src/com/ieci/tecdoc/common/invesdoc/Idoccompusg.hbm.x
./SIGEM_RegistroPresencialIEJar/src/com/ieci/tecdoc/common/invesdoc/Ivolvolhdr.hbm.xm
...
./SIGEM_RegistroPresencialIEJar/src/com/ieci/tecdoc/common/invesicres/ScrNackrecvreg.
./SIGEM_RegistroPresencialIEJar/src/com/ieci/tecdoc/common/invesicres/ScrNackrecvmsg.
./SIGEM_RegistroPresencialIEJar/src/com/ieci/tecdoc/common/invesicres/ScrSendmsg.hbm.
...
```

SIGEM sigue la convención de colocar los ficheros de mapeado en los mismos directorios en los que se encuentran las entidades Java. Vemos que se encuentran en los paquetes `com.ieci.tecdo.common.invesdoc` y `com.ieci.tecdoc.common.invesicres`. Se tratan de dos proyectos desarrollados por Informática El Corte Inglés, en su división Inves. Por los comentarios en distintos ficheros, se puede comprobar que el proyecto SICRES fue desarrollado en 2004. En esa

fecha el Ministerio de Administraciones Públicas define el proyecto SICRES para el desarrollo de un sistema informático para los registros de entrada-salidad (SICREM = Sistema de Información Común de Registros de Entrada Salida).

Lanzamos Eclipse y abrimos alguno de los ficheros de mapeo y de su clase Java correspondiente. Por ejemplo, los ficheros `ScrTypedoc.hbm.xml` y `ScrTypedoc.java` en el subpaquete `common.invesicres`:

Fichero **`com/ieci/tecdoc/common/invesicres/ScrTypedoc.hbm.xml`**:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
   "-//Hibernate/Hibernate Mapping DTD 2.0//EN"
   "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd" >

<hibernate-mapping>
   <!--
      Created by the Middlegen Hibernate plugin

      http://boss.bekk.no/boss/middlegen/
      http://hibernate.sourceforge.net/
   -->

   <class name="com.ieci.tecdoc.common.invesicres.ScrTypedoc"
      table="SCR_TYPEDOC">
      <meta attribute="class-description" inherit="false">
         @hibernate.class table="SCR_TYPEDOC"
      </meta>

      <id name="id" type="java.lang.Integer" column="ID">
         <meta attribute="field-description">
            @hibernate.id generator-class="assigned"
            type="java.lang.Integer" column="ID"
         </meta>
         <generator class="assigned" />
      </id>

      <property name="description" type="java.lang.String"
         column="DESCRIPTION" not-null="true" length="50">
         <meta attribute="field-description">
            @hibernate.property column="DESCRIPTION" length="50"
            not-null="true"
         </meta>
      </property>

      <property name="typePerson" type="int" column="TYPE_PERSON"
         not-null="true" length="10">
         <meta attribute="field-description">
            @hibernate.property column="TYPE_PERSON" length="10"
            not-null="true"
         </meta>
      </property>

      <property name="code" type="java.lang.String" column="CODE"
         not-null="true" length="1">
         <meta attribute="field-description">
            @hibernate.property column="CODE" length="1"
```

```
            not-null="true"
         </meta>
      </property>

      <!-- associations -->
      <!-- bi-directional one-to-many association to ScrPjur -->
      <set name="scrPjurs" lazy="true" inverse="true">
         <meta attribute="field-description">
            @hibernate.set lazy="true" inverse="true"
            @hibernate.collection-key column="TYPE_DOC"
            @hibernate.collection-one-to-many
            class="com.ieci.tecdoc.common.invesicres.ScrPjur"
         </meta>
         <key>
            <column name="TYPE_DOC" />
         </key>
         <one-to-many
            class="com.ieci.tecdoc.common.invesicres.ScrPjur" />
      </set>

      <!-- bi-directional one-to-many association to ScrPfi -->
      <set name="scrPfis" lazy="true" inverse="true">
         <meta attribute="field-description">
            @hibernate.set lazy="true" inverse="true"
            @hibernate.collection-key column="TYPE_DOC"
            @hibernate.collection-one-to-many
            class="com.ieci.tecdoc.common.invesicres.ScrPfi"
         </meta>
         <key>
            <column name="TYPE_DOC" />
         </key>
         <one-to-many
            class="com.ieci.tecdoc.common.invesicres.ScrPfi" />
      </set>

   </class>
</hibernate-mapping>
```

Clase **`com.ieci.tecdoc.common.invesicres.ScrTypedoc.java`**:

```
package com.ieci.tecdoc.common.invesicres;

import java.io.Serializable;
import java.util.Set;
import org.apache.commons.lang.builder.EqualsBuilder;
import org.apache.commons.lang.builder.HashCodeBuilder;
import org.apache.commons.lang.builder.ToStringBuilder;

/**
 *        @hibernate.class
 *         table="SCR_TYPEDOC"
 *
 */
public class ScrTypedoc implements Serializable {
   private Integer id;
   private String description;
   private int typePerson;
```

```
   private String code;
   private Set scrPjurs;
   private Set scrPfis;

   /** full constructor */
   public ScrTypedoc(Integer id, String description, int
typePerson,
         String code, Set scrPjurs, Set scrPfis) {
      this.id = id;
      this.description = description;
      this.typePerson = typePerson;
      this.code = code;
      this.scrPjurs = scrPjurs;
      this.scrPfis = scrPfis;
   }

   /** default constructor */
   public ScrTypedoc() {
   }

   /**
    *            @hibernate.id
    *             generator-class="assigned"
    *             type="java.lang.Integer"
    *             column="ID"
    *
    */
   public Integer getId() {
      return this.id;
   }

   public void setId(Integer id) {
      this.id = id;
   }

   /**
    *            @hibernate.property
    *             column="DESCRIPTION"
    *             length="50"
    *             not-null="true"
    *
    */
   public String getDescription() {
      return this.description;
   }

   public void setDescription(String description) {
      this.description = description;
   }

   /**
    *            @hibernate.property
    *             column="TYPE_PERSON"
    *             length="10"
    *             not-null="true"
    *
    */
   public int getTypePerson() {
      return this.typePerson;
```

```
}

public void setTypePerson(int typePerson) {
   this.typePerson = typePerson;
}

/**
 *            @hibernate.property
 *             column="CODE"
 *             length="1"
 *             not-null="true"
 *
 */
public String getCode() {
   return this.code;
}

public void setCode(String code) {
   this.code = code;
}

/**
 *            @hibernate.set
 *             lazy="true"
 *             inverse="true"
 *                @hibernate.collection-key
 *                 column="TYPE_DOC"
 *            @hibernate.collection-one-to-many
 *             class="com.ieci.tecdoc.common.invesicres.ScrPjur"
 *
 */
public Set getScrPjurs() {
   return this.scrPjurs;
}

public void setScrPjurs(Set scrPjurs) {
   this.scrPjurs = scrPjurs;
}

/**
 *            @hibernate.set
 *             lazy="true"
 *             inverse="true"
 *                @hibernate.collection-key
 *                 column="TYPE_DOC"
 *            @hibernate.collection-one-to-many
 *             class="com.ieci.tecdoc.common.invesicres.ScrPfi"
 *
 */
public Set getScrPfis() {
   return this.scrPfis;
}

public void setScrPfis(Set scrPfis) {
   this.scrPfis = scrPfis;
}

public String toString() {
   return new ToStringBuilder(this).append("id",
```

```
getId()).toString();
   }

   public boolean equals(Object other) {
      if (!(other instanceof ScrTypedoc))
         return false;
      ScrTypedoc castOther = (ScrTypedoc) other;
      return new EqualsBuilder().append(this.getId(),
castOther.getId())
            .isEquals();
   }

   //**********************************
   // Incluido pos ISicres-Common Oracle 9i

   public String toXML() {
      String className = getClass().getName();
      className = className.substring(className.lastIndexOf(".") +
1,
            className.length()).toUpperCase();
      StringBuffer buffer = new StringBuffer();
      buffer.append("<");
      buffer.append(className);
      buffer.append(">");
      try {
         java.lang.reflect.Field[] fields =
getClass().getDeclaredFields();
         java.lang.reflect.Field field = null;
         String name = null;
         int size = fields.length;
         for (int i = 0; i < size; i++) {
            field = fields[i];
            name = field.getName();
            buffer.append("<");
            buffer.append(name.toUpperCase());
            buffer.append(">");
            if (field.get(this) != null) {
               buffer.append(field.get(this));
            }
            buffer.append("</");
            buffer.append(name.toUpperCase());
            buffer.append(">");
         }
      } catch (Exception e) {
         e.printStackTrace(System.err);
      }
      buffer.append("</");
      buffer.append(className);
      buffer.append(">");
      return buffer.toString();
   }

   //**********************************

   public int hashCode() {
      return new HashCodeBuilder().append(getId()).toHashCode();
   }

}
```

Vemos que la entidad `ScrTypedoc` define una entidad con 6 atributos:

- **Integer id**: identificador único.
- **String description**: descripción del tipo de documento.
- **int typePerson**: tipo de persona que lleva ese tipo de documento.
- **String code**: código del tipo de documento.
- **Set scrPjurs**: conjunto resultante de una relación uno-a-mucho compuesto por objetos de la clase `ScrPjur`, personas jurídicas.
- **Set scrPfils**: conjunto que define una relación uno-a-muchos entre entidades `ScrTypedoc` y entidades `ScrPfil`, personas físicas.

Las entidades `ScrPfil` (persona física) y `ScrPjur` (persona jurídica) se definen con las siguientes clases Java:

Clase **`com.ieci.tecdoc.common.invesicres.ScrPfi`**:

```
package com.ieci.tecdoc.common.invesicres;

public class ScrPfi implements Serializable {
    private Integer id;
    private String nif;
    private String firstName;
    private String secondName;
    private String surname;
    private com.ieci.tecdoc.common.invesicres.ScrTypedoc
scrTypedoc;

    /** full constructor */
    public ScrPfi(Integer id, String nif, String firstName,
                  String secondName, String surname,
                  com.ieci.tecdoc.common.invesicres.ScrTypedoc
scrTypedoc) {
        this.id = id;
        this.nif = nif;
        this.firstName = firstName;
        this.secondName = secondName;
        this.surname = surname;
        this.scrTypedoc = scrTypedoc;
    }

    public ScrPfi() {
    }

    public ScrPfi(Integer id, String firstName,
                  com.ieci.tecdoc.common.invesicres.ScrTypedoc
scrTypedoc) {
        this.id = id;
        this.firstName = firstName;
        this.scrTypedoc = scrTypedoc;
    }

    public Integer getId() {
        return this.id;
    }
    public void setId(Integer id) {
```

```
        this.id = id;
    }

    public String getNif() {
        return this.nif;
    }

    public void setNif(String nif) {
        this.nif = nif;
    }
    public String getFirstName() {
        return this.firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getSecondName() {
        return this.secondName;
    }

    public void setSecondName(String secondName) {
        this.secondName = secondName;
    }

    public String getSurname() {
        return this.surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public com.ieci.tecdoc.common.invesicres.ScrTypedoc
getScrTypedoc() {
        return this.scrTypedoc;
    }

    public void
setScrTypedoc(com.ieci.tecdoc.common.invesicres.ScrTypedoc
scrTypedoc) {
        this.scrTypedoc = scrTypedoc;
    }

    public String toString() {...}
    public boolean equals(Object other) {...}
    public String toXML() {...}
    public int hashCode() {...]
```

La clase **`com.ieci.tecdoc.common.invesicres.ScrPjur`** es muy similar. La consultamos.

Vamos a comprobar la definición SQL de estas entidades. En el fichero de mapeo se definen las tablas `SCR_TYPEDOC`, `SCR_PFIS` y `SCR_PJUR`. Analizamos la base de datos definida en el fichero de configuración de Hibernate:

Fichero **`hibernate.cfg.xml`**:

```
<!-- Fichero de configuracion de hibernate -->
<hibernate-configuration>
   <session-factory name="java:/hibernate/HibernateFactory">

   <!-- Cambiar para poner el nombre de la fuente de datos -->
   <property
name="connection.datasource">java:comp/env/jdbc/registroDS_</property>
   <!-- property
name="dialect">net.sf.hibernate.dialect.OracleDialect</property>
-->
   <property
name="dialect">net.sf.hibernate.dialect.PostgreSQLDialect</property>
   <property name="show_sql">false</property>
   <property name="use_outer_join">false</property>
   <property name="use_query_cache">true</property>
   <property
name="cache.provider_class">net.sf.hibernate.cache.OSCacheProvider</property>

   <mapping
resource="com/ieci/tecdoc/common/invesdoc/Idocbtblctlg.hbm.xml" />
   <mapping
resource="com/ieci/tecdoc/common/invesdoc/Iuserldapgrphdr.hbm.xml"
/>
   <mapping
resource="com/ieci/tecdoc/common/invesdoc/Idocrpt.hbm.xml" />
   <mapping
resource="com/ieci/tecdoc/common/invesdoc/Idocglbdoch.hbm.xml" />
   ...
```

Vemos que utiliza el *datasource* definido con el nombre `java:comp/env/jdbc/registroDS_`. El servidor Tomcat proporciona este recurso.

Vamos a explorar esta base de datos utilizando el volcado de la base de datos que hicimos en la primera sesión de integración. Por si no lo tenemos, recordemos que se puede utilizar la herramienta PgAdmin3:

Ponemos en marcha la base de datos entrando como usuario `postgres` y ejecutando `pg_ctl start`:

```
> su postgres
Contraseña: postgres
> pg_ctl start -D /usr/local/pgsql/data
```

Volcamos la base de datos `registro_000` al fichero `registro.sql`:

```
pg_dump -U postgres -cif registro.sql registro_000
```

Y, por último, abrimos el fichero `registro.sql` con Eclipse y buscamos las tablas `SCR_TYPEDOC`, `SCR_PFIS` y `SCR_PJUR`:

```
CREATE TABLE scr_typedoc (
    id integer NOT NULL,
    description character varying(50) NOT NULL,
    type_person integer NOT NULL,
```

```
    code character varying(1) NOT NULL
);
ALTER TABLE ONLY scr_typedoc
    ADD CONSTRAINT pk_scr_typedoc0 PRIMARY KEY (id);

CREATE TABLE scr_pfis (
    id integer NOT NULL,
    type_doc integer,
    nif character varying(25),
    first_name character varying(25) NOT NULL,
    second_name character varying(25),
    surname character varying(20)
);
ALTER TABLE ONLY scr_pfis
    ADD CONSTRAINT fk_scr_pfis0 FOREIGN KEY (type_doc) REFERENCES
scr_typedoc(id);


CREATE TABLE scr_pjur (
    id integer NOT NULL,
    type_doc integer,
    cif character varying(17),
    name character varying(72) NOT NULL
);
ALTER TABLE ONLY scr_pjur
    ADD CONSTRAINT fk_scr_pjur0 FOREIGN KEY (type_doc) REFERENCES
scr_typedoc(id);

COPY scr_typedoc (id, description, type_person, code) FROM stdin;
1 CIF 2 C
2 NIF 1 N
3 Pasaporte 1 P
4 Documento de identificación de extranjeros 1 E
5 Otros 0 X
\.
```

2. Probando las entidades

Vamos a crear un proyecto para probar las entidades Hibernate. Lo llamamos `SIGEM_TestsHibernate`.

Enlazamos el proyecto `SIGEM_RegistroPresencialIEJar` y sus librerías. Copiamos el fichero `hibernate.cfg.xml` en el directorio `src` para configurar una conexión a la BD que no utilice la fuente de datos:

Fichero **`hibernate.cfg.xml`** en `SIGEM_TestsHibernate`:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 2.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-2.0.dtd">
<!-- Fichero de configuracion de hibernate -->
<hibernate-configuration>
   <session-factory>
    <property
```

```
name="connection.url">jdbc:postgresql://localhost/registro_000</property>
    <property name="connection.username">postgres</property>
    <property name="connection.password">postgres</property>
    <property
name="connection.driver_class">org.postgresql.Driver</property>
   <!-- Cambiar para poner el nombre de la fuente de datos -->
   <!-- <property
name="connection.datasource">java:comp/env/jdbc/regiDS_</property>-->
   <!-- property
name="dialect">net.sf.hibernate.dialect.OracleDialect</property>
-->
   <property
name="dialect">net.sf.hibernate.dialect.PostgreSQLDialect</property>
   <property name="show_sql">true</property>
   ...
```

Una vez modificado el fichero de configuración, creamos el paquete `test.entity` y en él la clase `TestScrPfi.java` en la que definimos un método `main` con una prueba de las entidades de Hibernate. El esqueleto de la clase es el siguiente:

```
public class TestScrPfi {
        public static void main(String[] args) {
            try{
                    SessionFactory sessionFactory = new
Configuration().
configure().buildSessionFactory();
                    Session session = sessionFactory.openSession();
                    Transaction tx = session.beginTransaction();

                    // Código

                    tx.commit();

                        System.out.println("Terminado");
                        }
            } catch (HibernateException e) {
                    System.out.println("Error");
                    System.exit(1);
            }
        }
}
```

Rellenamos un fragmento de código para buscar el tipo de documento 1, creamos una persona física, la hacemos persistente y relacionamos el tipo de documento y la persona. Después imprimimos todas las personas con ese tipo de documento.

Podemos buscar consultas de Hibernate en `com.ieci.tecdoc.isicres.session.reports`.

3. Flujo de la aplicación

Utilizamos la utilidad de Hibernate de *Búsqueda Java* para entender cómo están relacionadas las distintas capas de SIGEM y cómo se llaman unos a otras.

Vamos a ir hacia atrás, a partir de la entidad `ScrPfi`. Vamos a buscar, por ejemplo, en qué métodos se obtiene el NIF de una persona. Abrimos la clase, colocamos el cursor en el método `getFirstName()` y escogemos la opción *Search* > *Java* y buscamos todas las referencias a ese método.

Vemos que la clase `BookSession` en su método estático `addScrPfi` utiliza una entidad de tipo `ScrPfi` y realiza una llamada para obtener el nombre de la persona física. El método completo es el siguiente:

```
public static int addScrPfi(String sessionID, ScrPfi pfi, List
doms, String entidad)
        throws BookException, SessionException,
ValidationException {
     Validator.validate_String_NotNull_LengthMayorZero(sessionID,
ValidationException.ATTRIBUTE_SESSION);

     Date currentDate = null;¡¡
     Transaction tran = null;
     try {
         Session session = HibernateUtil.currentSession(entidad);
         tran = session.beginTransaction();

         // Recuperamos la sesión
         CacheBag cacheBag =
CacheFactory.getCacheInterface().getCacheEntry(sessionID);

         AuthenticationUser user = (AuthenticationUser)
cacheBag.get(HIBERNATE_Iuseruserhdr);
                 ScrOfic scrOfic = (ScrOfic)
cacheBag.get(HIBERNATE_ScrOfic);

         currentDate = BBDDUtils.getDateFromTimestamp(
DBEntityDAOFactory.getCurrentDBEntityDAO().getDBServerDate(entidad));

         int pfisId =
DBEntityDAOFactory.getCurrentDBEntityDAO().getContador4PERSONS(user.getId(),
entidad);

         if (log.isDebugEnabled()) {
             log.debug("addScrPfi.pfisId => " + pfisId);
         }

         pfi.setId(new Integer(pfisId));

         if (log.isDebugEnabled()) {
             log.debug("addScrPfi.pfi =>" + pfi.getFirstName() +
"," + pfi.getSecondName());
         }

         ScrTypedoc typeDoc = (ScrTypedoc)
session.load(ScrTypedoc.class, pfi.getScrTypedoc().getId());

         if (log.isDebugEnabled()) {
             log.debug("addScrPfi.typeDoc =>" + typeDoc);
         }

         pfi.setScrTypedoc(typeDoc);
```

```
            session.save(pfi);

            ScrPinfo scrPinfo = new ScrPinfo();
            scrPinfo.setId(new Integer(pfisId));
            scrPinfo.setOptype(0);
            scrPinfo.setOfficeid(scrOfic.getId().intValue());
            scrPinfo.setUsername(user.getName());
            scrPinfo.setOpdate(currentDate);
            session.save(scrPinfo);

            ScrAddress scrAddress = null;
            ScrDom scrDom = null;
            for (Iterator it = doms.iterator(); it.hasNext();) {
                scrDom = (ScrDom) it.next();

                if (scrDom.getId().intValue() == 0) {
                    scrAddress = new ScrAddress();
                    scrAddress.setId(new Integer(DBEntityDAOFactory
.getCurrentDBEntityDAO().getContador4SCRADDRESS(user.getId(),
entidad)));
                    scrAddress.setIdPerson(pfisId);
                    scrAddress.setType(0);
                    session.save(scrAddress);

                    scrDom.setId(scrAddress.getId());
                    session.save(scrDom);
                }
            }

            HibernateUtil.commitTransaction(tran);

            return pfisId;
        } catch (BookException bE) {
            HibernateUtil.rollbackTransaction(tran);
            throw bE;
        } catch (SessionException sE) {
            HibernateUtil.rollbackTransaction(tran);
            throw sE;
        } catch (Exception e) {
            HibernateUtil.rollbackTransaction(tran);
            log.error("Impossible to add a ScrPfis for the session ["
+ sessionID + "]", e);
            throw new
BookException(BookException.ERROR_CANNOT_ADD_PFIS);
        } finally {
            HibernateUtil.closeSession(entidad);
        }
    }
```

Vemos que se trata de un método de negocio que abre y cierra una sesión de Hibernate y realiza la acción de añadir una entidad en la base de datos. Podemos comprobar también que se utiliza una capa más antigua de acceso a BD no implementada con Hibernate, sino con DAOs.

Miramos quién utiliza este método, de la misma forma que antes. Y comprobamos que se trata de la clase `com.ieci.tecdoc.isicres.usecase.book.BookUseCase`, en su

método estático `vldInterSave`.

Si miramos quién usa este método, veremos que por fin hemos llegado a la capa web. Se trata del método `VldInterSave` en el paquete `com.ieci.tecdoc.isicres.servlets`.

Si hacemos este seguimiento para distintos métodos, vemos que siempre se repite el mismo patrón:

- **Capa web**: La capa web está formada por los servlets que se encuentran en el `com.ieci.tecdoc.isicres.servlets`. Cada servlet llama a un caso de uso de la aplicación, definido en la capa intermedia.
- **Capa intermedia**: Define los casos de uso de la aplicación. Clase `BookUsecase` en el paquete `com.ieci.tecdoc.isicres.usecase.book`. Llama a la lógica de negocio, que es quien utiliza las sesiones de Hibernate, y gestiona las transacciones.
- **Lógica de negocio**: Clases `BookSession`, `ReportSession`, etc. en el paquete `com.ieci.tecdoc.isicres.session.book`. Define un método estático por cada caso de uso. Los métodos reciben y devuelven entidades desconectadas que hacen de transfer objects.
- **Capa de persistencia**: Clases de entidad en el paquete `com.ieci.tecdoc.common.invesicres`, clases DAO en el paquete `com.ieci.tecdoc.isicres.entity.dao` y clases EJB de entidad refactorizadas en el paquete `com.ieci.tecdoc.isicres.entity`.