Qué es el Sotilizator?

Se trata de un plugin para el eclipse (probado en su versión Helios). La finalidad de este plugin es la creación de una aplicación web de forma automática y con unas determinadas características a partir de un esquema de una Base de Datos.

Estas características son:

1 – La aplicación resultante estará realizada empleando un MVC en J2EE con Struts 1.3, inyección de dependencias con spring 3.0, hibernate 3.2.6 para el acceso a datos para la parte de la lógica de negocio. Mientras que para la parte de la vista se empleará jquery junto a las tag-libraries de struts. Además, de DWR para interactuar con el servidor desde el cliente.

2 – Se generará un fichero pom.xml para que a través de MAVEN se puedan obtener las dependencias.

3 – Se generará un fichero build.xml donde estarán las tareas Ant necesarias para el despliegue de la aplicación en Tomcat, JBoss y OAS.

4 – La aplicación resultante será capaz de construir las tareas CRUD de una tabla automáticamente.

5 – Las capas que contendrá la aplicación son las siguientes:

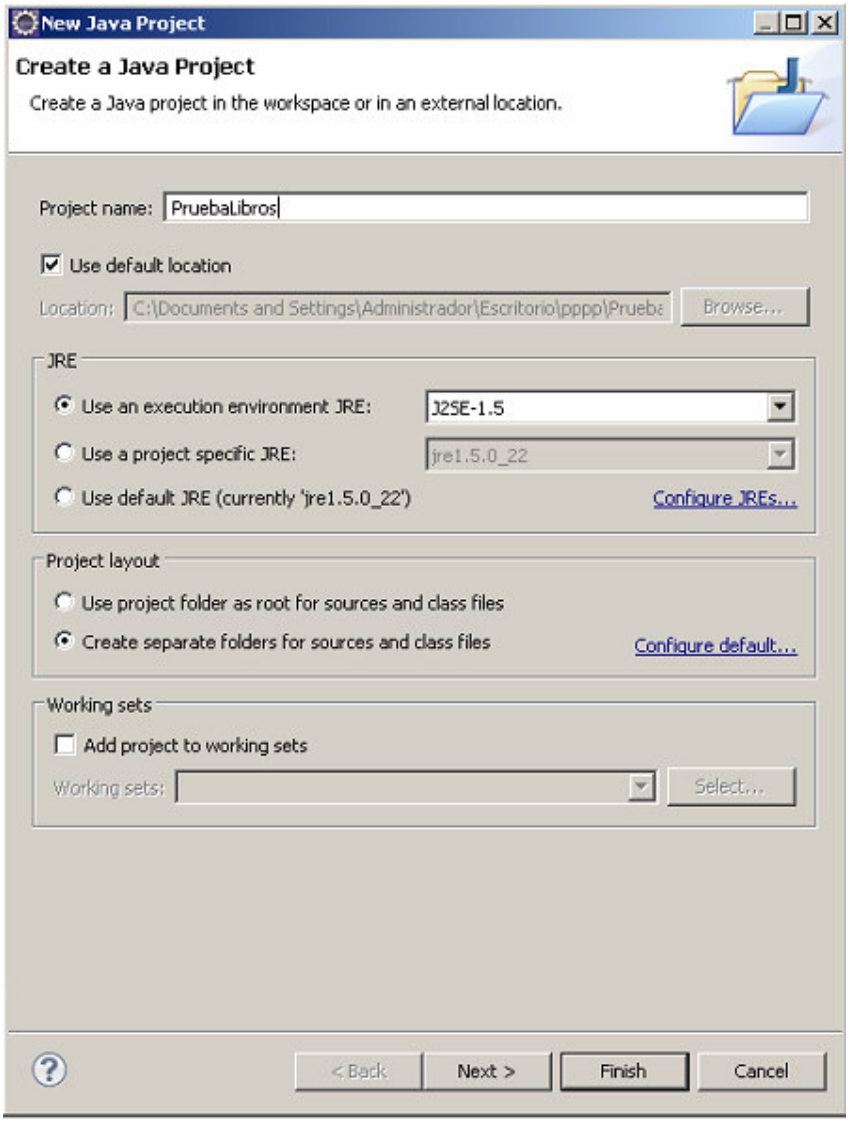
- VO: Objetos que hace referencias a las tablas.
- DAO: Capa que realiza el acceso a la BD.
- Manager: Capa sobre la que se aplican las transacciones.
- Delegate: Capa que se encarga de relacionar la parte de la vista con la lógica de negocio.
- Action: Aquí deberán ir las diferentes acciones que luego el controlador mapea para interactuar con la vista.

Instalación y ejecución del plugin

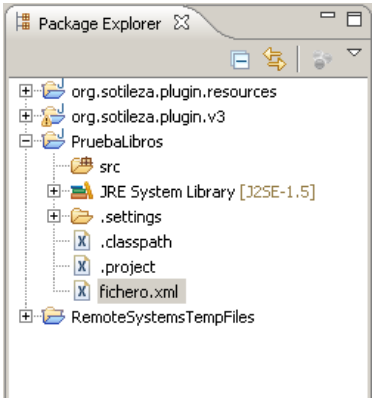
En primer lugar se debe descargar del repositorio lo siguiente:

- org.sotileza.plugin.resources: este directorio deberá colocarse tal cuál en el directorio *plugins* de nuestro eclipse. Contiene el arquetipo de la aplicación a construir.
- org.sotileza.plugin_3.0.0.jar: es el plugin y se debe colocar también en el directorio *plugins* de nuestro eclipse.
- lib: este directorio, contiene las dependencias de la aplicación que crea el Sotilizator. Si se tiene maven instalado no es necesario descargarlo, puesto que las aplicaciones que crea el plugin contienen un fichero pom.xnl a través del cuál se pueden descargar automáticamente con maven.

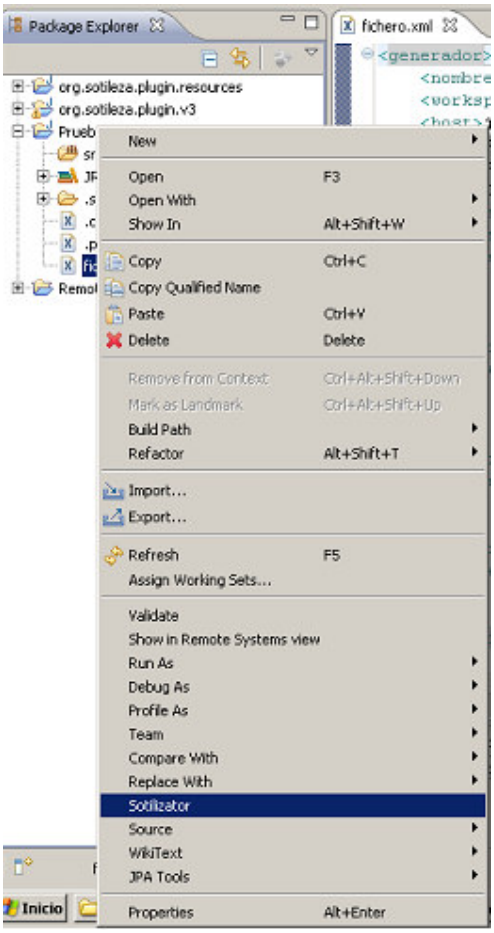
A continuación se arranca el eclipse y se procede a crear un proyecto de java. Ej:



Dentro del proyecto se debe generar un fichero con extensión xml que contendrá la configuración de la aplicación que se desea obtener:



Por último para ejecutar el plugin, basta con hacer click con el botón derecho del ratón sobre el "fichero.xml" y seleccionar la opción sotilizator.



Si todo va bien se generará la aplicación, se puede comprobar situándose en el eclipse en el proyecto y ejecutando F5. Ahora solo faltan las librerías del proyecto. Para esto hay varias formas (explicado anterioremente), o bien se copia la carpeta lib de otro proyecto o bien se bajan de maven con el pom.xml (mvn validate) o se copia la carpeta lib que se adjunta junto a los plugins y que ya tiene exactamente las que se necesitan.

Configuración del fichero.xml

A continuación se va explicar lo que debe contener cada tag del fichero. En general hay que respetar las mayúsculas-minúsculas en todo el fichero, para evitar posibles errores.

nombreApp: Es el nombre de la aplicación creada anteriormente.

workspace: Debe contener la ruta física donde se encuentra el espacio de trabajo donde se encuentra el proyecto.

host: Debe contener la url para acceder a la BD.

user: Usuario de acceso a la BD

pass: Contraseña de acceso a la BD

ds: Debe contener el nombre del jndi que hace referencia al datasource creado en el tomcat, oc4j,…

pakete: Debe contener el paquete raíz para la aplicación.

tablas: Aquí va la configuración de las tablas propias de la aplicación. Un ejemplo para intentar explicarlo:

```
	<tabla nombre="AUTOR" secuencia="SEQ_PK_AUTOR"
esquema="PRUEBA_LIBROS">
		<objeto>Autor</objeto>
		<nivel>maestro</nivel>
		<set>true</set>
	</tabla>
```

nombre: Hace referencia al nombre de la tabla.

secuencia: Hace referencia al nombre de la secuencia que se quiere aplicar a la clave primaria de es tabla.

esquema: Hace referencia al nombre del esquema al que pertenece la tabla.

objeto: Nombre que tendrá esa tabla en la aplicación.

nivel: Indica hasta que capa se quieren generar sus códigos fuente. Posibles valores (cada nivel contiene a los anteriores):

- DAO: Sólo se construirá la capa DAO para ese objeto.
- Manager: Se construirán la capa DAO y la Manager
- Delegate: Se construirá además de las anteriores la capa delegate. Se incluye la creación del ActionForm.
- Maestro: Se construirá todas las capas y además la vista con las acciones de insertar, modificar, borrar y consultar.

set: Si éste tag existe (da igual su valor) significa que es una tabla principal y en los mapeos hbm se le incluirán los sets (tablas que están relacionadas con ella a través de foreign key). Además, los manger y delegate de las tablas a las que hacen referencia las foreign key y los set serán absorbidos por los de esta tabla.

Tipos de Datos Admitidos en la BD

Las claves primarias deben que ser todas de tipo numérico.

Estos son los tipos de datos que de momento son admitidos en la BD para crear los campos:

NUMBER(x ,y): Dependiendo de los valores de x e y, el plugin interpretara que:

- es un **Long**, si tiene una longitud mayor de 10 caracteres.
- un **Integer,** si tiene una longitud menor o igual que 10 caracteres.
- un **Boolean,** si tiene un tamaño de uno.
- un **Double**, si tiene decimales.

NUMBER: Interpretará que es un **Long**.

VARCHAR2: Interpretará que se trata de un **String**.

CLOB: Interpretará que se trata de un **String** (**text** en mapeo de hibernate).

DATE: Interpretará que se trata de un **date**.

Modificaciones en el Arquetipo de la Aplicación

La carpeta org.sotileza.plugin.resources que se descomprimió en la carpeta plugins del eclipse es la que contiene toda la estructura de la aplicación por defecto.

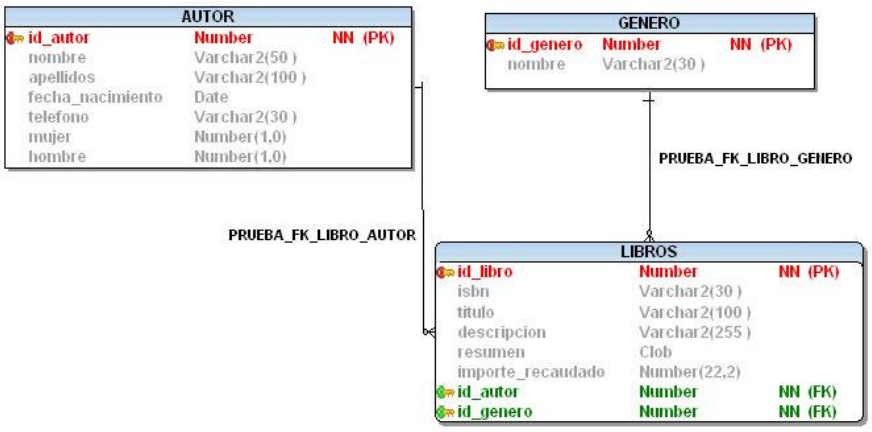
Dentro de ella hay dos carpetas con distintas funciones:

- Resources: Contiene las carpetas, ficheros de configuración y demás archivos necesarios para el funcionamiento de la aplicación, excepto las clases java.

- ResorucesClases: Contiene las clases java fijas en las aplicaciones. Se deben poner con extensión .jav en lugar de .java

Si se quiere modificar el contenido de algún fichero o añadir directorios/ficheros, bastará con realizar las modificaciones ahí y el plugin lo interpretará como corresponda.

Para este esquema:



Este sería el fichero para la aplicación de ejemplo:

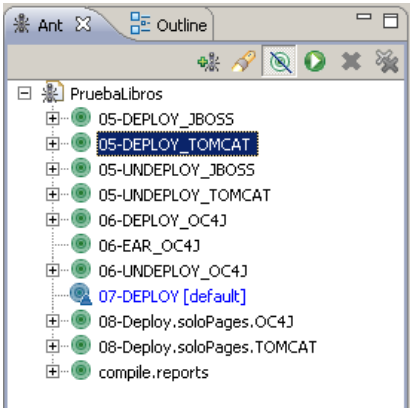
fichero.xml

```
<generador>
    <nombreApp>PruebaLibros</nombreApp>
    <workspace>C:\Documents and Settings\Administrador\Escritorio\pppp</workspace>
    <host>jdbc:oracle:thin:@localhost:1521:XE</host>
    <user>PRUEBA_LIBROS</user>
    <pass>PRUEBA_LIBROS</pass>
    <ds>PRUEBA_LIBROS</ds>
    <pakete>es.com.disastercode.prueba</pakete>
    <tablas>
        <tabla nombre="AUTOR" secuencia="SEQ_PK_AUTOR" esquema="PRUEBA_LIBROS">
            <objeto>Autor</objeto>
            <nivel>maestro</nivel>
            <set>true</set>
        </tabla>
        <tabla nombre="GENERO" secuencia="SEQ_PK_GENERO" esquema="PRUEBA_LIBROS">
            <objeto>Genero</objeto>
            <nivel>maestro</nivel>
            <set>true</set>
        </tabla>
        <tabla nombre="LIBROS" secuencia="SEQ_PK_LIBROS" esquema="PRUEBA_LIBROS">
            <objeto>Libro</objeto>
            <nivel>maestro</nivel>
        </tabla>
    </tablas>


</generador>
```

Nota: Tras ejecutar el plugin en este fichero, añadir las librerías y desplegarlo en el tomcat se obtiene la aplicación con la página de inicio y los maestros funcionando, además de toda la lógica de negocio del resto de tablas.

Para desplegar el proyecto creado se generará un fichero Build.xml con una serie de tareas Ant que permiten desplegar la aplicación en un Tomcat, un JBoss o un Oc4j, según se prefiera. Los nombres de las tareas son bastante descriptivos.



Se debe tener en cuenta que la conexión a la BD se configura en el datasource-beans.xml mediante jndi. Por lo tanto se debería configurar ese jndi en el propio contenedor. Por ejemplo, en el Tomcat consistiría en añadir al Context.xml algo así:

```
<Resource name="jdbc/PRUEBA_LIBROS"
        auth="Container"
        type="javax.sql.DataSource"
        factory="org.apache.tomcat.dbcp.dbcp.BasicDataSourceFactory"
        username="prueba_libros"
        password="prueba_libros"
        driverClassName="oracle.jdbc.driver.OracleDriver"
        url="jdbc:oracle:thin:@localhost:1521:XE"
        maxWait="1000"
        removeAbandoned="true"
        maxActive="30"
        maxIdle="10"
        removeAbandonedTimeout="60"
        logAbandoned="true"/>
```