Tutorial 3: Cómo ejecutar una aplicación OpenXava con OXPortal.

,

Tabla de contenidos

Introducción

A este tutorial podemos haber llegado desde dos puntos distintos:

- Una vez realizado el *Tutorial1: Cómo desarrollar una aplicación OpenXava a partir de un modelo UML2 con MOSKitt*. En este caso ya disponemos de una aplicación OpenXava generada a partir de un modelo UML2. Para poder ejecutar esta aplicación debemos pasar directamente al punto *"3. Descripción del entorno OXportal*" en este tutorial. En él se explica cómo poner en marcha una aplicación OpenXava haciendo uso del entorno de desarrollo proporcionado en el fichero `oxportal-4.2.X.zip`, disponible en la web del proyecto MOSKitt.

- Una vez finalizado el *Tutorial2:Cómo incorporar diseño de interfaz de usuario en aplicaciones OpenXava con MOSKitt*. En este caso dispondremos de un Modelo UML2 a partir del cual ya habremos generado las clases Java correspondientes etiquetadas con anotaciones JPA y OpenXava. También dispondremos de un modelo de Sketcher en el cual el analista ha especificado el diseño de algunas de las ventanas de nuestra aplicación. Si es este nuestro caso, seguiremos por el punto *"2. Tarea 1: Generar el código OpenXava*".

En este tutorial vamos explicar:

- Cómo generar el código Java completo incluyendo también las anotaciones OpenXava relativas al diseño de la interfaz de usuario tal y como ha sido modelado con MOSKitt-Sketcher.

- Cómo instalar el entorno OXPortal y ejecutar en él la aplicación generada con MOSKitt.

Todos los tutoriales están disponibles en la web del proyexto MOSKitt.

Tarea 1: Generar el código OpenXava

Una vez hemos modelado la aplicación ya podemos generar el código con las anotaciones relativas a:

- La persistencia de las entidades definidas en el modelo UML2 según el estándar JPA.
- La interfaz de usuario que nos va a mostrar la información de estas mismas entidades según OpenXava.

Para ello, seguiremos los siguientes pasos:

1. Paso 1: Generar las clases Java con las anotaciones JPA para la persistencia de las entidades UML2.

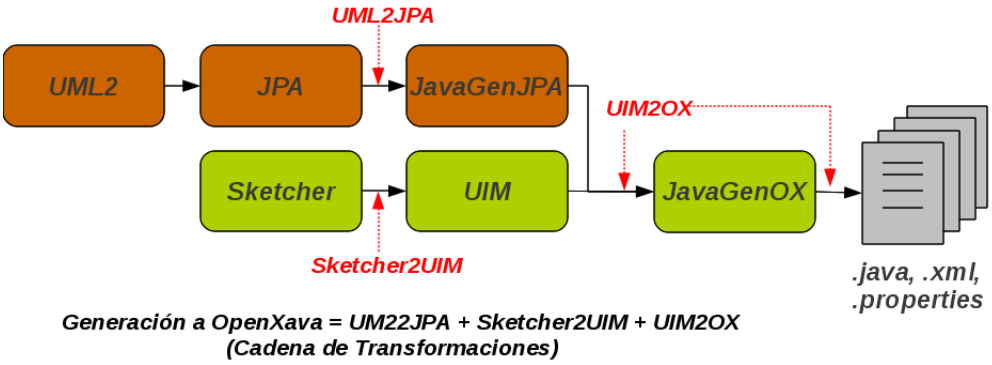
   Para ello lanzaremos la transformación UML2JPA a partir del modelo UML2 (`invoice_domain.uml`).

   Este paso lo hemos realizado al seguir el *"Tutorial 1: Cómo desarrollar una aplicación OpenXava a partir de un modelo UML2con MOSKitt"* y sólo necesitamos repetirlo si hemos hecho alguna modificación en el modelo UML2 durante la especificación de la interfaz de usuario con MOSKitt-Sketcher.

2. Paso 2 [Opcional]: Incorporar a las clases Java las anotaciones OpenXava correspondientes al modelo de Sketcher definido.

   Para ello lanzaremos la transformación UIM2OpenXava a partir del modelo UIM (`invoice.uim`).

En ambos casos se tienen en cuenta más modelos, tal y como se muestra en la figura que se muestra a continuación:



Generación a OpenXava = UM22JPA + Sketcher2UIM + UIM2OX
(Cadena de Transformaciones)

Vamos a ver a continuación con un poco más de detalle estos dos pasos.

Creación y configuración de un proyecto nuevo OpenXava en MOSKitt

Vamos a ver (o recordar) qué pasos tenemos que dar para crear un proyecto nuevo basado en OpenXava y que permita ser empaquetado, desplegado y ejecutado en el portal Liferay de la distribución.

1. Abrir MOSKitt y apuntar el workspace al directorio *oxportal/workspace*
2. Aunque no es necesario se aconseja cambiar a la perspectiva <Java EE>
3. Ejecutar la acción de menú <Run>.<External Tool>.<New OpenXava Project>
4. Introducir el nombre del proyecto en el cuadro de diálogo que aparece (¡Ojo! Este nombre deberá ser igual que el valor de la propiedad 'Id' del elemento Sketcher del modelo de Sketcher o la propiedad 'nombre' del elemento UserInterfaceModel del modelo de UIM).
5. Ya tenemos creado nuestro proyecto en el disco, pero no lo tenemos accesible desde la vista de proyectos. Para hacerlo accesible hacemos lo siguiente:

a. Ejecutar la acción del menú <File>.<Import...>

b. Seleccionar "Existing Projects into Workspace" y pulsar <Next>

c. Seleccionar en la opción "Select root directory" la ruta del workspace: *oxportal/workspace*

d. En la lista de proyectos nos aparecerá nuestro nuevo proyecto seleccionado. Pulsamos <Finish> y nuestro proyecto estará disponible en la vista de Proyectos.

Tarea 1.1: Lanzar la transformación UML2JPA

La transformación UML2JPA de MOSKitt permite generar código Java a partir de elementos de un modelo UML2. Estos últimos son aquellos representados en el diagrama de clases: clases, interfaces, propiedades, operaciones, asociaciones, especializaciones, etc.

El código java contempla, además de la declaración de tipos java estándard, la anotación de las clases y/o sus atributos y métodos. Estas anotaciones permiten expresar y configurar el código generado con elementos específicos de estándares JSR. Los conjuntos de anotaciones soportados actualmente son:

- JSR-220: Enterprise JavaBean 3.0. Java Persistent API. Nos permite definir la configuración de persistencia de nuestras entidades.

- JSR-303: Bean Validation API. Nos permite especificar restricciones de validación en nuestras entidades que luego pueden ser usadas desde distintas capas en la arquitectura de nuestra aplicación.

Si vemos con un poco más de detalle el proceso de transformación del modelo UML2 al código Java correspondiente, nos daremos cuenta de que entran en juego los siguiente recursos:

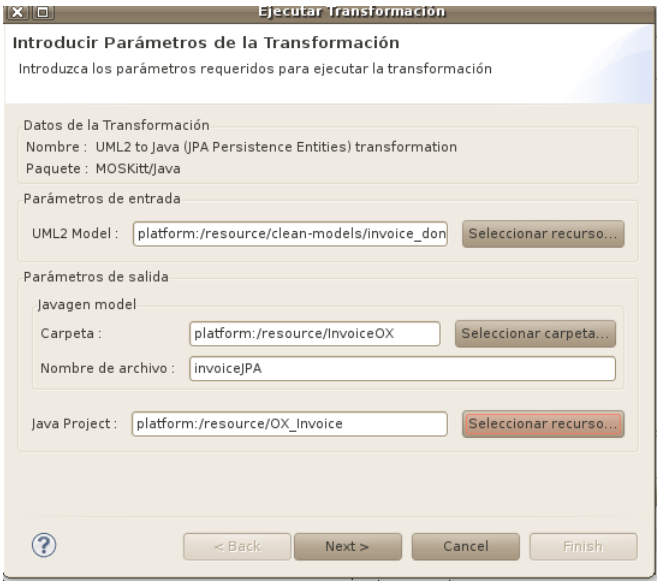
Recusos de Entrada:
- Modelo UML2 (`invoice_domain.uml`).

Recursos Intermedios:
- Modelo de configuración de la transformación (`invoice_domain.transformationconfiguration`) en el cual se indica, para cada entidad UML2 qué anotación JPA se debe aplicar.
- Modelo JPA (`invoice_domain.jpadefinitions`) que complementa al anterior indicándole la información requerida por cada una de las anotaciones seleccionadas.

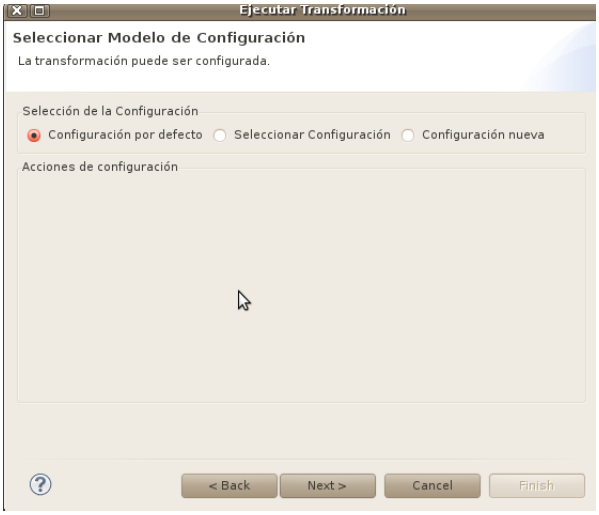
Recursos de Salida:
- Modelo JavaGen con las anotaciones JPA que se van a incluir en las clases Java (`invoiceJPA.javagen`).
- Clases Java con anotaciones JPA.

Para lanzar la transformación nos debemos situar sobre el modelo UML2 (`invoice_domain.uml`) y seleccionar del menú contextual el submenú *MOSKItt Transformations* y de éste la opción *UML2 to Java (JPA Persistence Entities) transformation*.

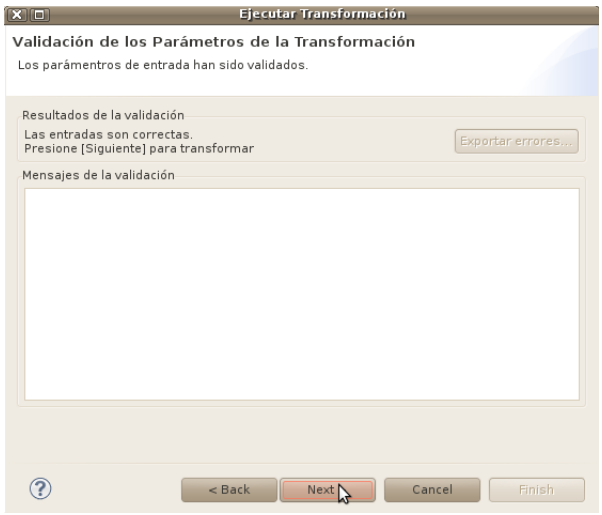
1. El asistente pide como parámetros de entrada el modelo UML2 y como parámetros de salida nos pide una carpeta dónde dejar el modelo JavaGen generado, el nombre que le debe dar y un proyecto Java donde dejar el código fuente generado:



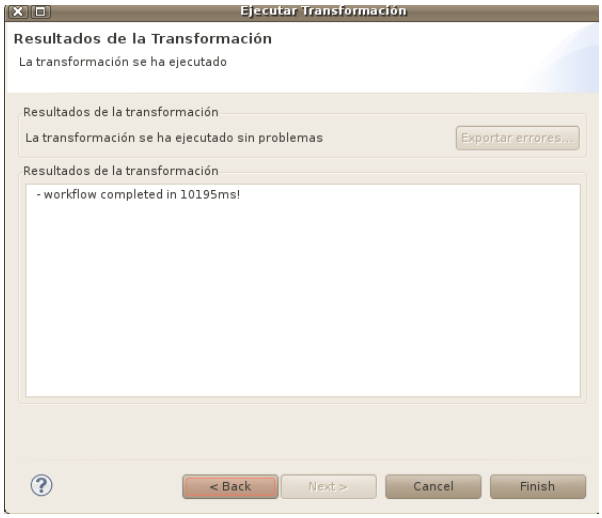
2. Todas las transformaciones MOSKitt se pueden configurar para alterar el resultado en función de la aplicación que nos ocupa en cada momento. Sin embargo, MOSKitt proporciona una configuración por defecto con las reglas de transformación más comúnmente utilizadas en cada caso y es esta la que vamos a utilizar por el momento (más adelante, en secciones posteriores veremos cómo modificar esta configuración):



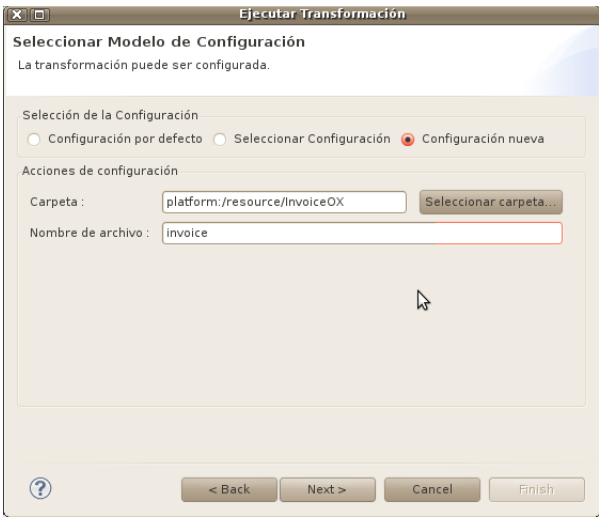
3. Antes de lanzar la transformación se comprueba que los parámetros de entrada son correctos. Es decir, que los modelos de entrada son los que deben de ser y que su contenido es el esperado (cuando se detectan errores en este punto, la transformación sólo continuará si los errores son considerados como warnings):



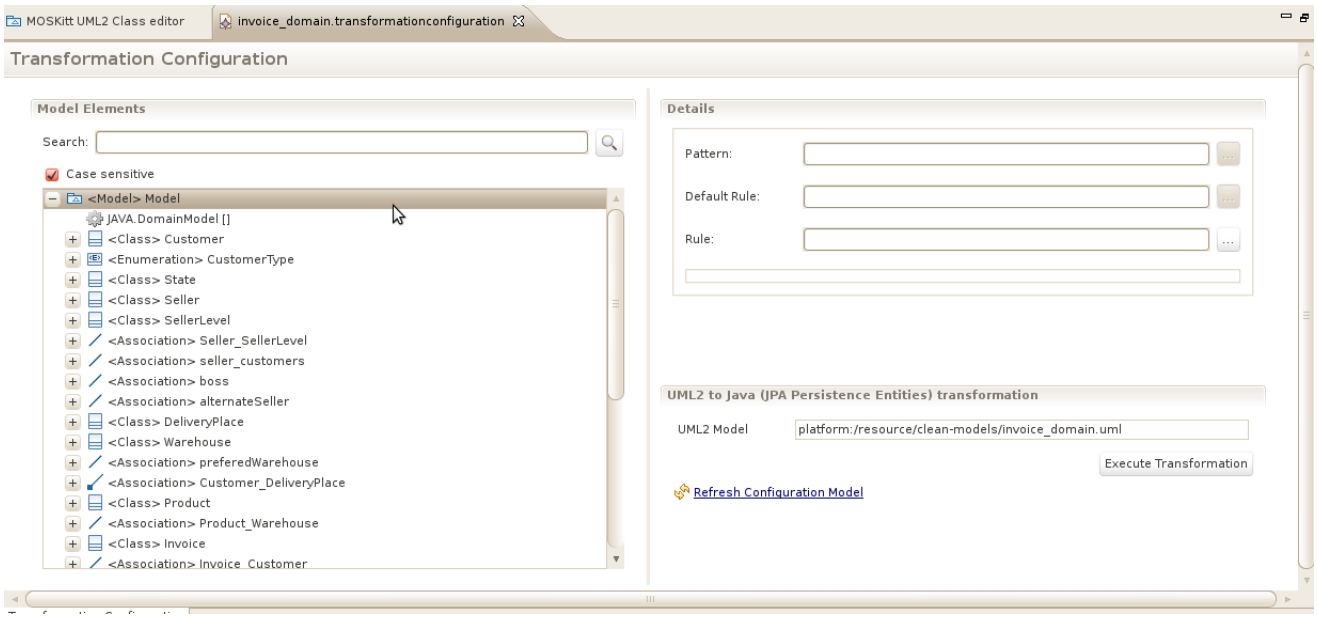
4. Una vez ha finalizado la transformación el asistente nos indica que todo ha ido bien:



Si estuviésemos interesados en configurar alguna de las etiquetas JPA a aplicar en la transformación, en el paso 2 debemos seleccionar la opción Configuración nueva tal y como se muestra en la siguiente figura:



En este caso, al pulsar el botón `Next`, se abrirá el Editor de Configuración de la transformación para que podamos configurar las etiquetas JPA que consideremos necesarias. El editor tiene el aspecto que muestra la siguiente figura:



Para más información sobre las etiquetas JPA configurables ó sobre el funcionamiento del editor de configuración consultar el Manual de Usuario de MOSKitt.

Tarea 1.2: Lanzar la transformación UI2OX

La transformación UI2OX de MOSKitt permite generar código Java a partir de elementos de un modelo abstracto de interfaz de usuario UIM y un modelo JavaGenJPA (un modelo que representa clases Java preparadas para manejar su persistencia con anotaciones JPA) . Las entidades a persistir están modeladas en un modelo UML2 que a su vez está enlazado con el modelo UIM ya que en éste se indica qué propiedades de qué clases van a ser mostradas desde la interfaz.

Por tanto, los recursos utilizados por la transformación son:

Recursos de Entrada:

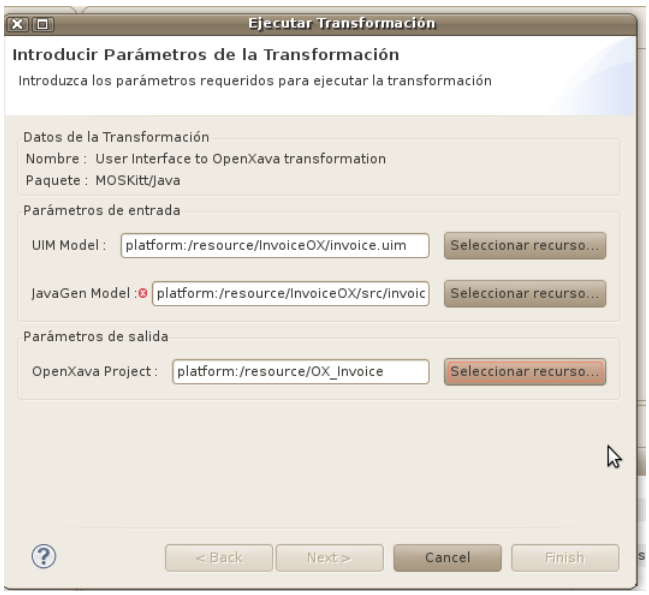
- Modelo UIM (`invoice.uim`).
- Modelo JavaGen obtenido en el paso 1 (`invoice.javagen`).
- A partir del modelo UIM de entrada, MOSKitt alcanzará los modelos Sketcher y UML2 para obtener de ellos cierta información necesaria para completar la transformación.

Recursos de Salida

- Modelo JavaGen en el que se incorpora información sobre las anotaciones OpenXava que se van a incluir en las clases Java (`invoice.javagen`).
- Ficheros con las clases Java con anotaciones JPA y OpenXava (.java), ficheros .xml para definir la aplicación OpenXava y ficheros para la internacionalización de la aplicación (.properties).

Podemos lanzar la transformación desde el modelo de UIM. Para ello, sobre el modelo (`invoice.uim`) seleccionamos del menú contextual el submenú *MOSKItt Transformations* y de éste seleccionamos la opción *User Interface to OpenXava transformation*.

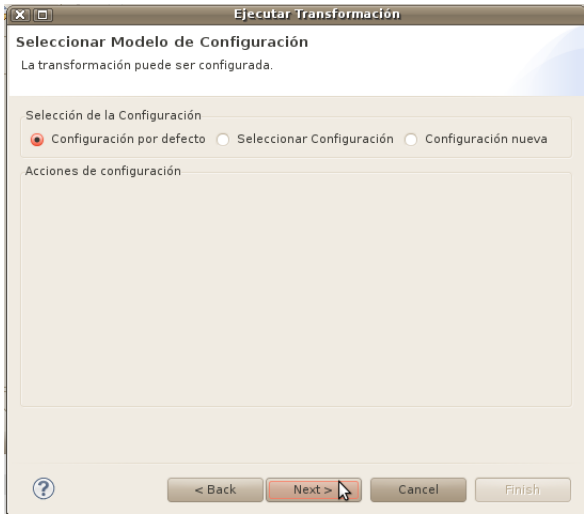
El asistente pide como parámetros de entrada los modelos UIM y JavaGen y un proyecto Java donde dejará el código generado:



Hay que tener en cuenta que hay dos proyectos diferentes:

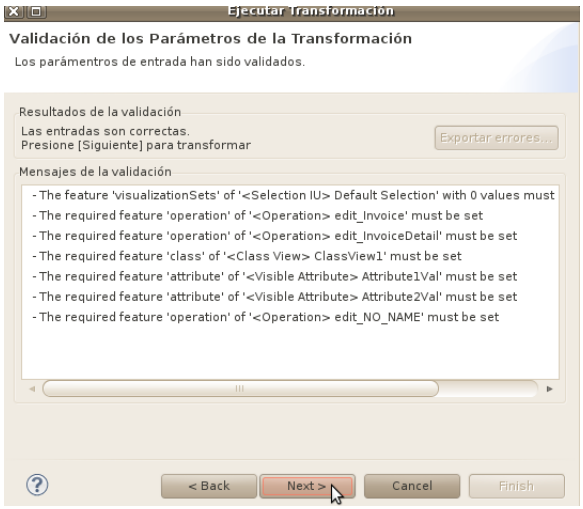
- El proyecto MOSKitt en el cual se localizan nuestros modelos.
- El proyecto Java en el cual se localiza el código generado por la transformación.

Todas las transformaciones MOSKitt se pueden configurar para alterar el resultado en función de la aplicación que nos ocupa en cada momento. Sin embargo, MOSKitt proporciona una configuración por defecto con las reglas de transformación más comunmente utilizadas y es esta la que vamos a utilizar por el momento, más adelante, en secciones posteriores veremos cómo modificar esta configuración:

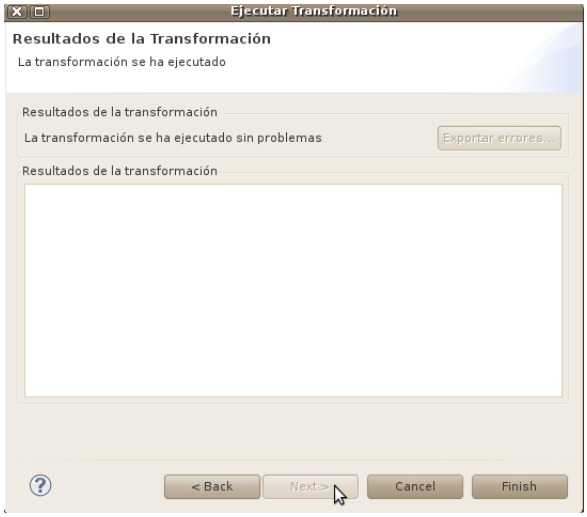


En este caso, la configuración por defecto genera el código sin Zonas Protegidas (para más información sobre qué son y para qué sirven las zonas protegidas ir al apartado correspondiente, más adelante en este mismo manual) y un módulo por componente con los controladores por defecto (para más información sobre los módulos y controladores ir al apartado correspondiente).

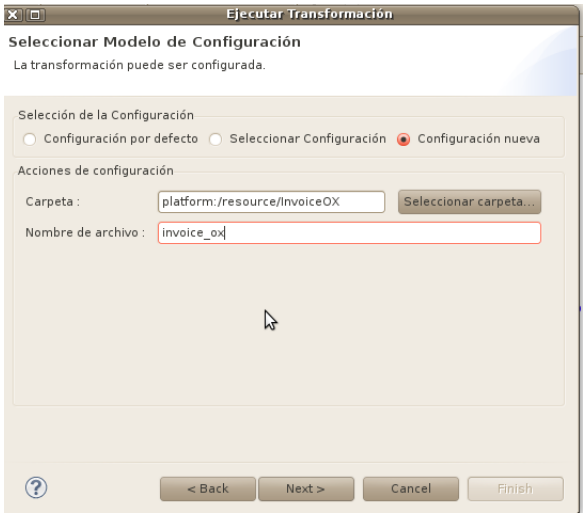
Antes de lanzar la transformación se comprueba que los parámetros de entrada son correctos. Es decir, que los modelos de entrada son los que deben de ser y que su contenido es el esperado. Cuando se detectan errores en este punto, la transformación sólo continuará si los errores son considerados como warnings (como es el caso que se muestra en al figura que aparece a continuación):



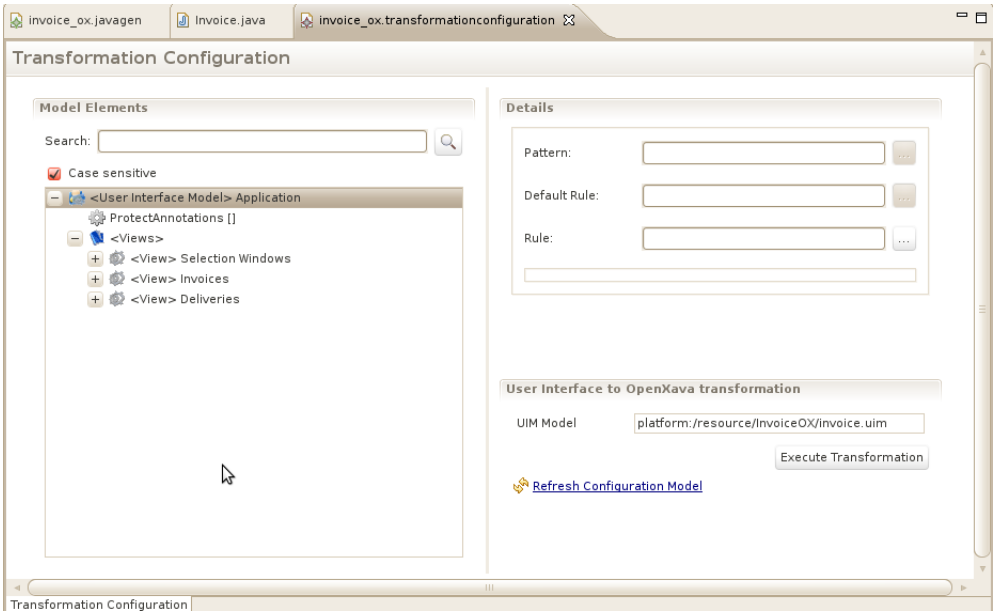
Una vez ha finalizado la transformación el asistente nos indica que todo ha ido bien:



Si estamos interesados en configurar la transformación, en el segundo paso debemos seleccionar la opción `Configuración nueva` tal y como se muestra en la siguiente figura:

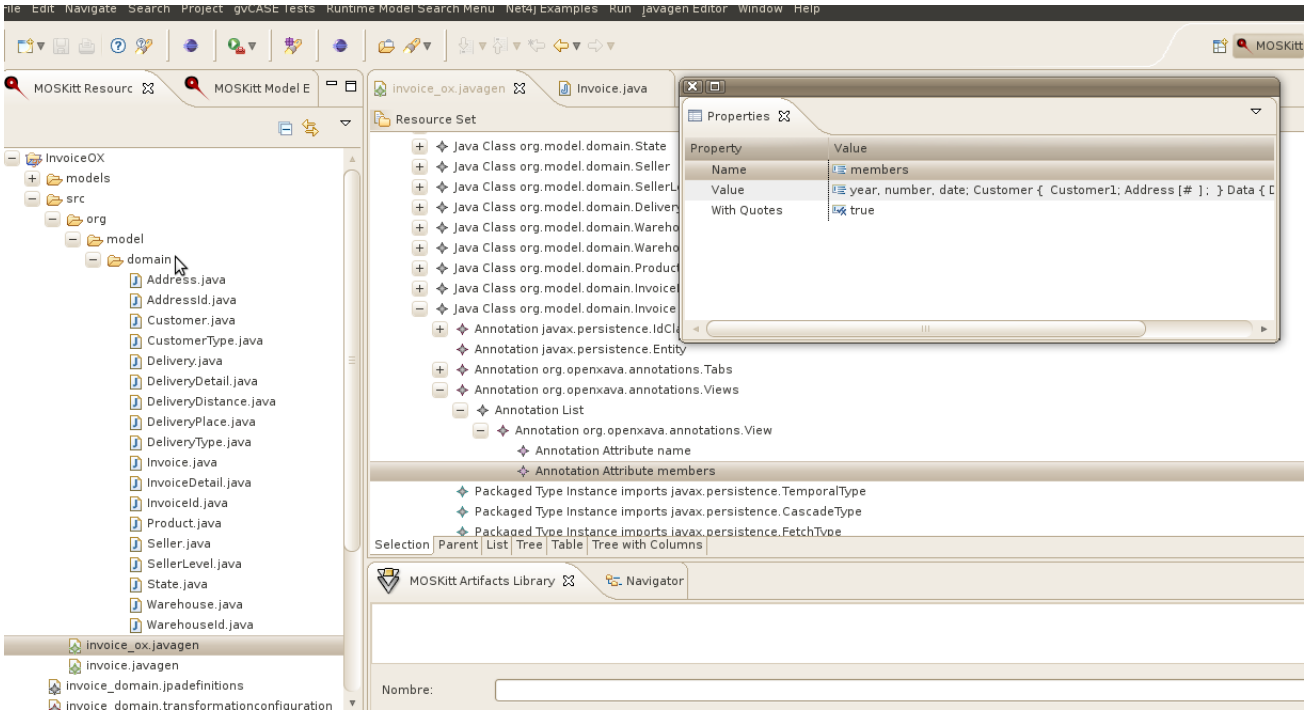


En este caso, al pulsar el botón `Next`, se abrirá el Editor de Configuración de la transformación. El editor tiene es aspecto que muestra la siguiente figura:

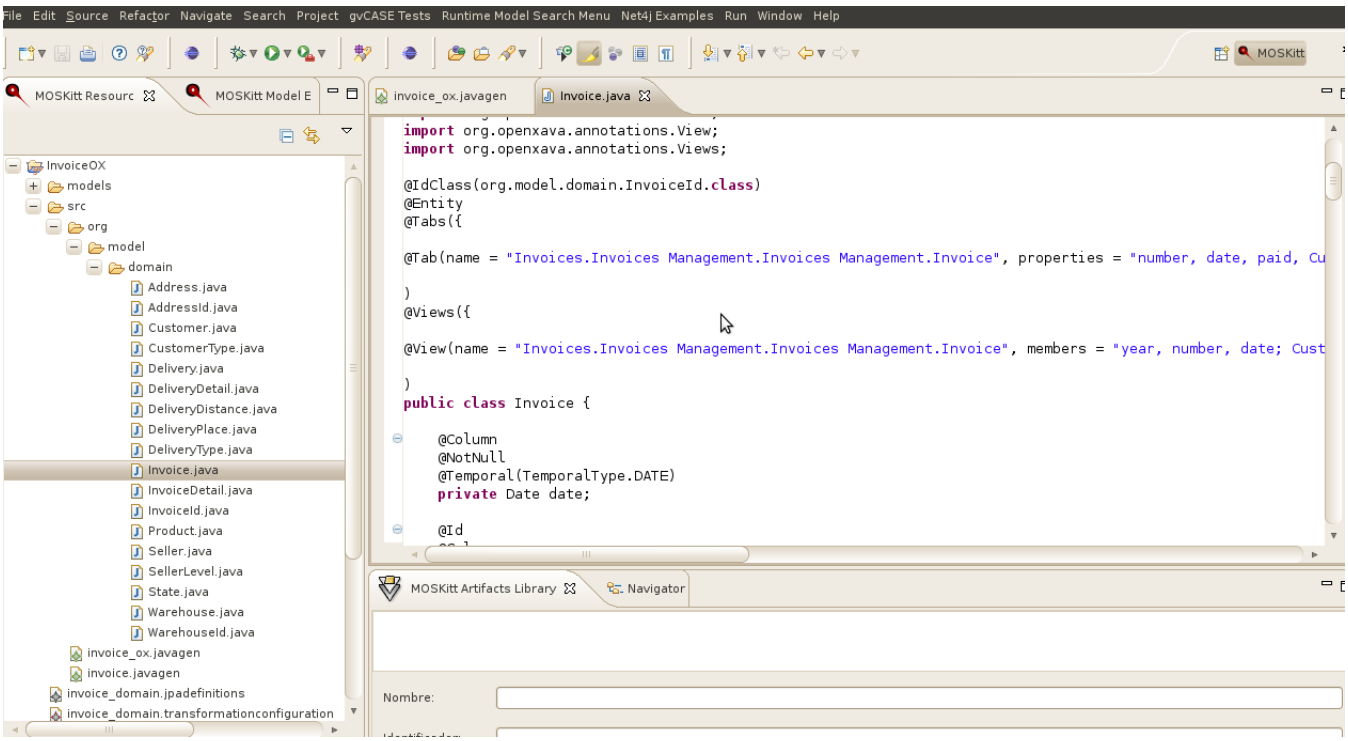


Para más información sobre el funcionamiento del editor de configuración consultar el Manual de Usuario de MOSKitt.

En la siguiente figura se muestra el resultado de la transformación. En concreto el contenido del modelo JavaGen en el cual podemos ver las clases Java que luego aparecen en el código generado:



A continuación podemos ver cómo al pulsar doble click sobre cualquiera de los ficheros Java se muestra su contenido en el editor con las correspondientes anotacioens JPA y OpenXava:



Para más información sobre los recursos generados por la transformación consultar más adelante, en este mismo manual.

Descripción del entorno OXportal

Denominamos oxportal al entorno de desarrollo, despliegue y ejecución de nuestra aplicación OpenXava. Básicamente es un paquete comprimido con los componentes que se describen a continuación:

- **Contenedor de Servlets/JSP Apache Tomcat 6.0.29:** pondrá en ejecución las aplicaciones Web que despleguemos (incluida el propio portal).
  - Ubicación principal: oxportal/liferay/tomcat-6.0.29
  - Ubicación scripts de arranque/parada: oxportal/liferay/tomcat-6.0.29/bin
  - Ubicación ficheros de configuración: oxportal/liferay/tomcat-6.0.29/conf
  - Ubicación directorio despliegue: oxportal/liferay/tomcat-6.0.29/webapps
- **Portal JSR-286 Liferay 6.0.6 Community Edition:** pondrá en ejecución todos los módulos de nuestra aplicación como portlets.
  - Ubicación principal: oxportal/liferay
  - Ubicación directorio despliegue: oxportal/liferay/deploy
- **Base de datos HSQLDB (1.8.1):** mantiene los datos que necesita el portal Liferay además de proporcionarnos almacenamiento para la base de datos de nuestra aplicación.
  - Ubicación del driver JDBC: oxportal/liferay/tomcat-6.0.29/lib/ext/hsql.jar
  - Ubicación base de datos del portal Liferay: oxportal/liferay/data
  - Ubicación base de datos de aplicación: oxportal/liferay/tomcat-6.0.29/data

- **Workspace para MOSKitt con la distribución OpenXava 4.2.x:** entorno que nos permite desarrollar aplicaciones OpenXava. Si desde MOSKitt conectamos con este workspace tendremos a disposición todo lo necesario para generar y desplegar nuestra aplicación en el portal.

  - Ubicación workspace: oxportal/workspace

Para tener listo nuestro entorno sólo tenemos que descomprimir el paquete *oxportal-4.2.x.zip* en un directorio a nuestra elección.

Instalación del entorno OXPortal.

Creación de la instancia de base de datos y configuración de acceso en un proyecto OpenXava.

Ya tenemos nuestro proyecto OpenXava en nuestro workspace listo para ser completado. El siguiente paso es crear una instancia de base de datos HSQLDB en la cual estarán las tablas correspondientes y configurar el proyecto para que pueda acceder a la misma.

Para **crear la instancia de la base de datos** realizaremos los siguientes pasos:

1. Abrir una terminal del intérprete de órdenes (shell).

2. Nos ubicamos en el directorio oxportal/liferay/tomcat-6.0.29/bin

3. Ejecutamos la orden de creación de instancia de base de datos de HSQLDB: *./start-hsqldb.sh <nombre_instancia> <puerto>*

   Debemos tener en cuenta qué nombre damos a nuestra instancia y qué puerto seleccionamos. Ejemplo: *./start-hsqldb.sh invoicingDB 9001*

4. A partir de este momento ya tenemos una instancia de base de datos vacía que podemos acceder a ella vía JDBC usando esta URL: *jdbc:hsqldb:hsql://localhost:<puerto>/<nombre_instancia>*

   Otros parámetros de acceso necesarios son los siguientes:

   - **username:** sa

   - **password:**

Para configurar el **acceso a la base de datos desde MOSKitt para poder crear y actualizar las tablas del esquema de base de datos** realizaremos los siguientes pasos:

1. Editamos el fichero <mi_proyecto>/persistence/META-INF/persistence.xml

2. Buscamos la "<persistence-unit>" llamada "junit". Tenemos que completar la propiedad "hibernate.connection.url". Especificamos la url de acceso JDBC a nuestra base de datos: jdbc:hsqldb:hsql://localhost:<puerto>/<nombre_instancia_db>

3. Guardamos el fichero.

Para configurar el **acceso a la base de datos desde el portal Liferay para que nuestra aplicación acceda a los datos una vez desplegada y en ejecución** realizaremos los siguientes pasos:

1. Verificamos que el servidor Tomcat lo tenemos parado. Si no es así debemos pararlo (`./shutdown.sh`).

2. Editamos el fichero oxportal/liferay/tomcat-6.0.29/conf/context.xml

3. Añadimos al tag <Context> la siguiente entrada:

<Resource name="jdbc/<mi_proyecto>DS" auth="Container" type="javax.sql.DataSource" maxActive="20" maxIdle="5" maxWait="10000" username="sa" password="" driverClassName="org.hsqldb.jdbcDriver" url="jdbc:hsqldb:hsql://localhost:<puerto>/<nombre_instancia_db>"/>

4. Una vez el servidor Tomcat se inicie las aplicaciones en él desplegadas tendrán disponible el recurso JNDI con nombre "jdbc/<mi_proyecto>DS" que hemos registrado en el contexto global del contenedor. Este recurso es un DataSource que nuestra aplicación usará para acceder a la base de datos configurada.

Creación de las tablas de la base de datos a partir de entidades JPA de un proyecto OpenXava.

Una vez tenemos el código de nuestras entidades JPA en nuestra aplicación (ya sea por haber sido generado con MOSKitt o creado a mano) es el momento de crear las tablas en la base de datos. Debemos cerciorarnos que tenemos en marcha la instancia de la base de datos tal y como explica el punto anterior.

Desde MOSKitt ejecutaremos los siguientes pasos:

1. Abrir el fichero <mi_proyecto>/build.xml

2. En la vista "Outline" veremos la lista de targets disponibles en este automatismo Ant. Seleccionamos la target llamada "updateSchema". Abrimos el menú contextual desde ella y ejecutamos la opción <Run As>.<Ant Build>.

3. En la vista "Console" veremos una traza de la ejecución del automatismo seleccionado. En ella veremos cómo se crean nuestras tablas y si el proceso ha ido bien o no. El siguiente paso es inspeccionar dichas tablas. Para ello usaremos un gestor de base de datos compatible JDBC (ejemplo, SquirrelSQL).

Puesta en marcha del entorno de ejecución. Arranque del portal Liferay.

Antes de desplegar nuestra aplicación necesitamos poner en marcha el portal para que una vez se realize el despliegue podamos tener la aplicación en ejecución.

Para ello desde una terminal de órdenes nos ubicaremos en la carpeta o*oxportal/liferay/tomcat-6.0.29/bin y ejecutaremos la orden:*

./startup.sh.

Si queremos ver los mensajes de información que el contenedor Tomcat genera para poder ser informados de posibles errores tanto en el despliegue como en la ejecución de nuestra aplicación, podemos ejecutar la siguiente orden:

tail -f ../logs/catalina.out.

Esta orden muestra por la terminal todo lo que ocurre en el contenedor.

Empaquetado y despliegue de un proyecto OpenXava como portlets

Para poder desplegar una aplicación OpenXava previamente generada tenemos dos opciones:

- **Empaquetado y despliegue como aplicación Web estándar (.WAR):** nos permite empaquetar la aplicación como un archivo .WAR (Web application Archive) el cual puede ser desplegado en cualquier contenedor de Servlets estándard, por ejemplo Apache Tomcat, Jetty, etc. El resultado nos permite acceder a cada módulo de la aplicación especificando una URL concreta, pero esto no resulta muy cómodo de cara al usuario final.

- **Empaquetado y despliegue como aplicación Web con portlets (JSR-286):** nos permite desplegar la aplicación en un portal que soporte el estándard Portlets 2.0 (JSR-286). Cada módulo de la aplicación se transforma en un portlet. El portal nos permite configurar el acceso a todos los portlets estableciendo políticas de acceso a los mismos para los usuarios del portal. Además nos permite utilizar componentes que mejoran la accesibilidad (menús) para que el usuario pueda acceder a todos los módulos de forma intuitiva. Esta es la forma que hemos elegido para desplegar y ejecutar nuestra aplicación de ejemplo.

Vamos a mostrar aquí cómo hacerlo siguiendo la segunda aproximación: Portlets.

Ya tenemos nuestra base de datos disponible y el portal en ejecución. Nuestra aplicación está lista para ser desplegada y ejecutada. En el fichero build.xml que acompaña a nuestro proyecto tenemos un target llamado "**`deployPortlets`**" que hace todas las tareas de una vez. Estas tareas son:

- Compilación del código fuente de la aplicación.
- Generación de los ficheros descriptores necesarios: `web.xml` y `portlet.xml`
- Empaquetado de la aplicación como fichero `.WAR`
- Desplegado de la aplicación en el portal (Liferay).

Desde MOSKitt ejecutaremos los siguientes pasos:

1. Abrir el fichero `<mi_proyecto>/build.xml`
2. En la vista "`Outline`" veremos la lista de targets disponibles en este automatismo `Ant`. Seleccionamos la target llamada "`deployPortlets`". Abrimos el menú contextual desde ella y ejecutamos la opción `<Run As>.<Ant Build>`.
3. En la vista "`Console`" veremos una traza de la ejecución del automatismo seleccionado. En ella veremos cómo compila, se empaqueta la aplicación y se copia a la carpeta de despliegue de aplicaciones con portlets en el portal.

Si tenemos una terminal con los mensajes de salida del contenedor Tomcat tal y como se sugiere en el punto anterior, podremos ver el proceso de despliegue de la aplicación en el portal Liferay. Ello nos ayudara a detectar posibles errores en el despliegue así como a adivinar cuando nuestra aplicación está lista para ser usada.

Acceso a la aplicación en ejecución desde el portal Liferay

Ya tenemos desplegada nuestra aplicación en el portal. Ahora nos toca acceder al mismo y configurarlo para que los portlets de nuestra aplicación OpenXava sean accesibles por el usuario final.

La instancia de Liferay que se distribuye con el paquete `oxportal-4.2.x.zip` ya viene configurada con un usuario administrador con privilegios para crear páginas que nos permitan ejecutar los portlets o módulos de nuestra aplicación. Vamos pues a crear una página en el portal que sea capaz de darnos acceso a los portlets de nuestra aplicación. Para ello seguiremos los siguientes pasos:

1. Abrir nuestro navegador y apuntar a la `URL: http://localhost:8080`
2. En el portlet de Login introduciremos los datos de acceso del usuario administrador del portal:

- Usuario: `test@moskitt.org`
- Password: `test`

3. Desde el menú ubicado en la parte superior, ejecutaremos la acción `<Administrar>.<Panel de Control>`

4. En el panel colapsable localizado a la izquierda, en la sección llamada "`MOSKitt Web Application Generation`", accedemos a la opción "`Páginas`".

5. En el formulario actual vamos a crear una nueva página. Para ello introduciremos el nombre de la página y le diremos que es de tipo "`Panel`". Pulsaremos después el botón "`Añadir página`".

6. Ya tenemos nuestra página creada. Ahora tenemos que configurarla. Para ello seleccionaremos la página en el menú. Después tenemos que seleccionar la opción tabular llamada "`Página`". Entonces nos aparecerá un formulario con la configuración de la misma.

7. En el formulario de configuración de la página veremos una sección llamada "`Seleccione los portlets que estarán disponibles en el panel`.". En ella debemos tener disponible nuestra aplicación con todos sus portlets. Seleccionaremos aquellos que necesitemos sean accesibles por el usuario final. Después pulsaremos el botón "`Guardar`".

8. Ya tenemos nuestra página configurada. Para acceder a la aplicación desde el menú superior pulsaremos la opción "`Volver a MOSKitt Web Application Generation`". En este momento veremos una opción más tabular con el nombre de nuestra página. Si accedemos nos aparecerá a la izquierda un portlet que nos muestra la lista de módulos de nuestra aplicación. Seleccionando cualquiera de ellos nos aparecerá como un portlet a la derecha la ejecución del módulo correspondiente.