Tutorial 1: Cómo desarrollar una aplicación OpenXava a partir de un modelo UML2 con MOSKitt.

,

Tabla de contenidos

Introducción

Vamos a ver en este tutorial paso a paso, cómo generar una aplicación OpenXava a partir de un modelo UML2.

Primero vamos a describir cómo crear el modelo UML `invoice_model` localizado en el directorio `/moskitt/oxportal/workspace/invoicingModels` que obtenemos al descomprimir el fichero `OXPortal-4.2.X.zip` disponible en la web de MOSKitt.

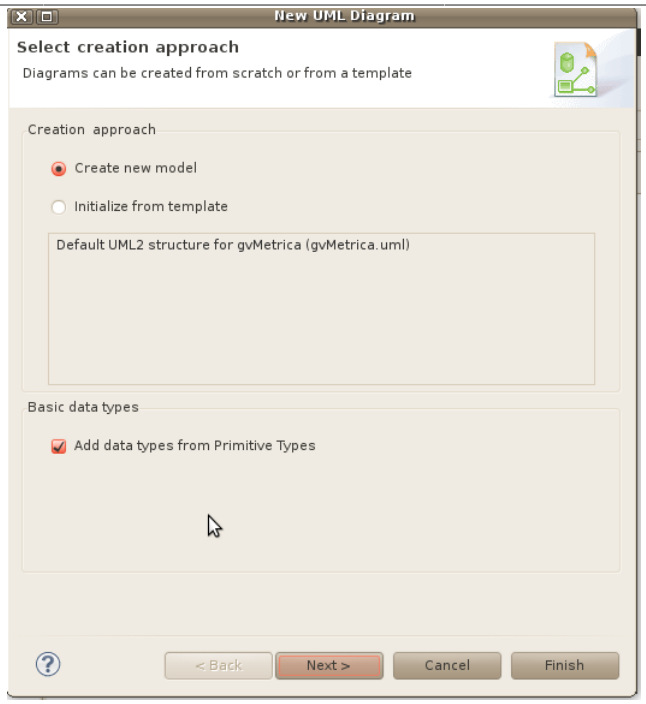
A continuación, sobre el modelo UML ya creado, vamos a lanzar la transformación de código para generar una aplicación en OpenXava.

Si el ususario no quiere crear manualmente el modelo, puede pasar directamente al punto *Tarea 2: Generar el código OpenXava a partir de un modelo UML2*.

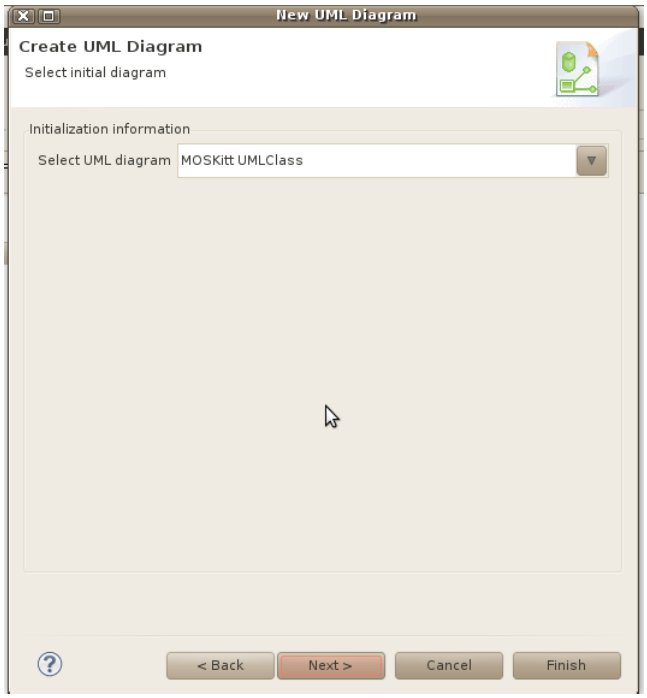
Tarea 1: Crear el Modelo UML2 con MOSKitt

1. Seleccionar la opción `File/New.../MOSKitt UML Diagram`.

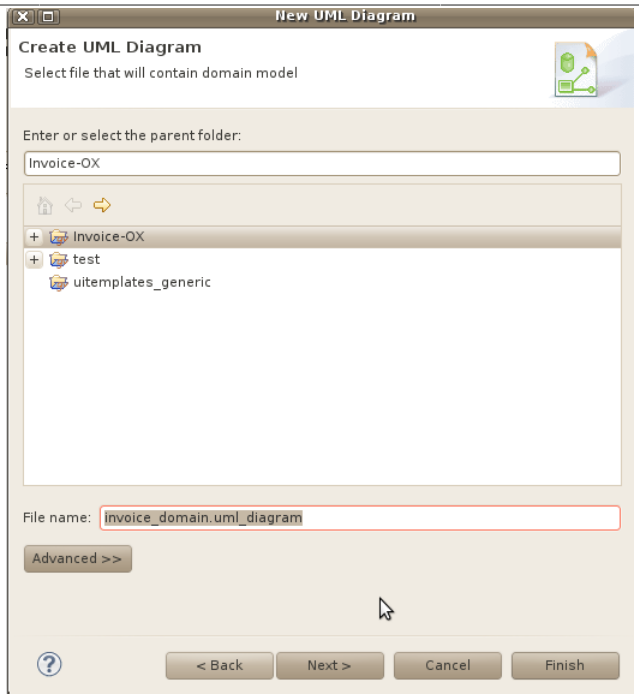
2. Chequear la opción `Create new model`. Indicamos también que el modelo incluya los tipos primitivos de UML2 por defecto (`Add data types from Primitive Types`):



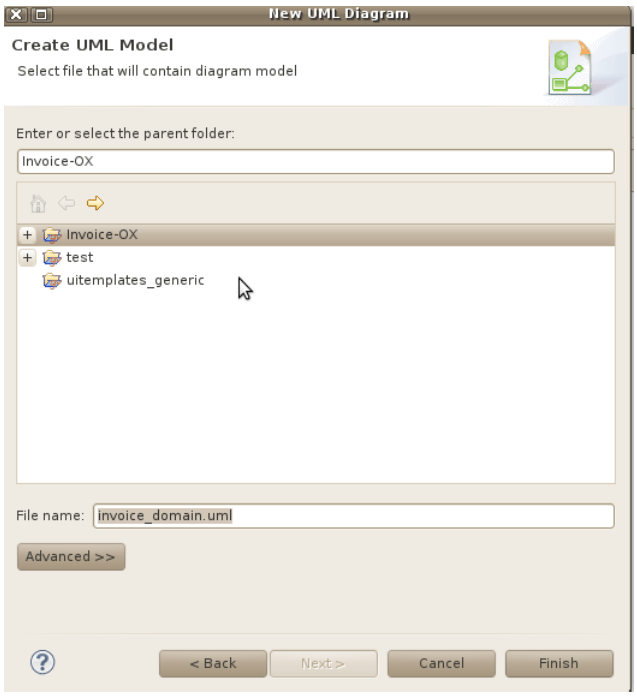
3. Seleccionamos como Diagrama principal un Diagrama de Clases pues sólo vamos a hacer uso de este tipo de diagramas UML2:



4. Seleccionamos el proyecto que debe incluir el diagrama del modelo (`Enter or select the parten folder`) y le asignamos un un nombre (`File name:`):

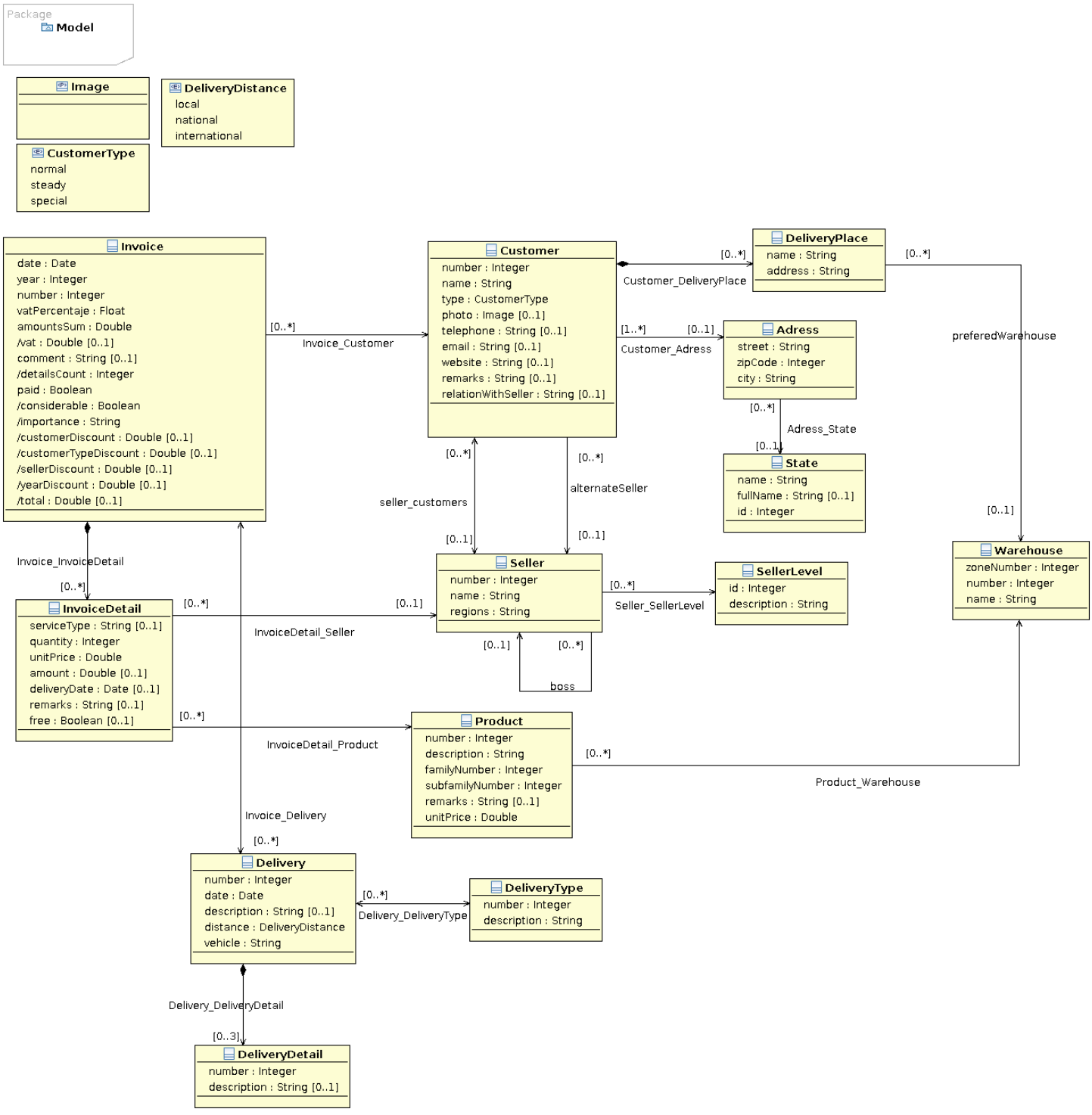


5. Seleccionamos el proyecto que debe incluir el modelo (`Enter or select the parten folder`) y le asignamos un un nombre (`File name:`). Por defecto MOSKitt le asignará el mismo nombre que al diagrama:



A continuación, se abrirá un diagrama de clases vacío para empezar a modelar nuestra aplicación.

Asumimos el siguiente modelo de clases (modelo UML2) para modelar la lógica de negocio de la aplicación de facturación que queremos generar:



Para más información sobre cómo modelar con MOSKitt-UML2 consultar el manual de usuario en la ayuda de MOSKitt.

Tarea 2: Generar el código OpenXava a partir de un modelo UML2

Creación y configuración de un proyecto nuevo OpenXAVA en MOSKitt

Vamos a crear un proyecto OpenXava en el que localizar el código que vamos a generar 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>. Para ello ir a `Window/Open Perspective/Java EE`.
3. Ejecutar la acción de menú: `Run/External Tools/New OpenXava Project`.
4. Introducir el nombre del proyecto en el cuadro de diálogo que aparece.
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.

Lanzar la transformación UML2JPA para obtener el código OpenXava

La transformación UML2JPA de MOSKitt permite generar código Java a partir de elementos de un modelo UML2.

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 y de OpenXava. 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.
- OpenXava: Permite configurar la transforamción para que incorpore las anotaciones @Stererotype, @Hidden en las propiedades de las clases UML. Por defecto, MOSKitt no les aplica estas anotaciones a las propiedades.

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.

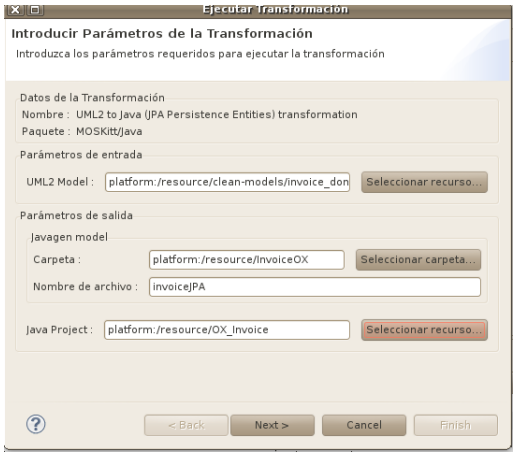
Recursos de Salida:

- Modelo JavaGen con las anotaciones JPA y OpenXava que se van a incluir en las clases Java (`invoiceJPA.javagen`).
- Clases Java con anotaciones JPA y OpenXava.

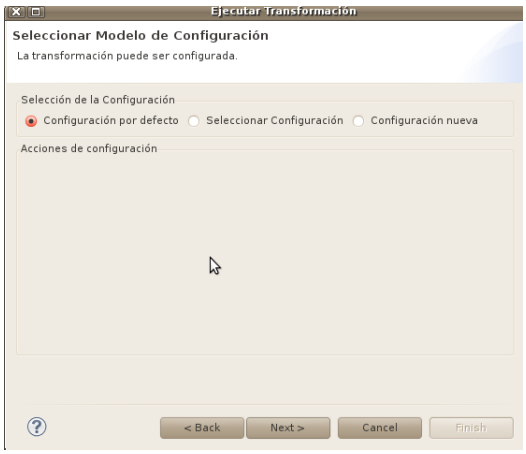
1. Para lanzar la transformación nos debemos situar sobre el modelo UML2 (`invoice_domain.uml`) y seleccionar del menú contextual el submenú *MOSKItt Transformations* y de éste la opción *UML2 to Java (JPA Persistence Entities) transformation*.

   Se debe tener en cuenta que para lanzar la transformación el usuario debe tener activa la perpectiva de MOSKitt (`Window/Open Perspective/ MOSKitt`).

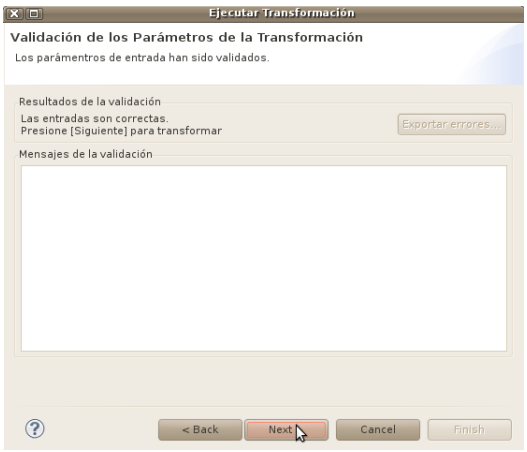
2. El asistente pide como parámetros de entrada el modelo UML2 y como parámetros de salida nos pide una carpeta dónde dejar el modelo JavaGen generado, el nombre que le debe dar y un proyecto Java donde dejar el código fuente generado:



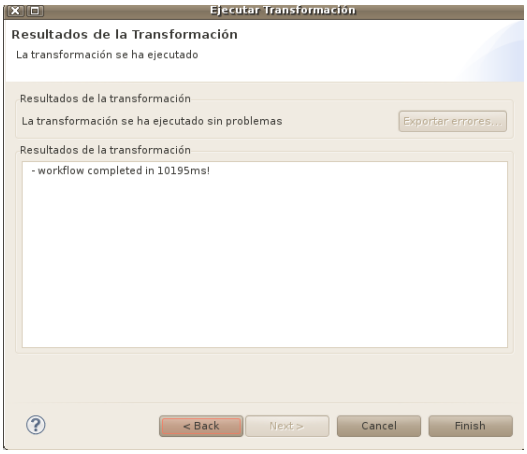
3. Todas las transformaciones MOSKitt se pueden configurar para alterar el resultado en función de la aplicación que nos ocupa en cada momento. Sin embargo, MOSKitt proporciona una configuración por defecto con las reglas de transformación más comunmente utilizadas en cada caso y es esta la que vamos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



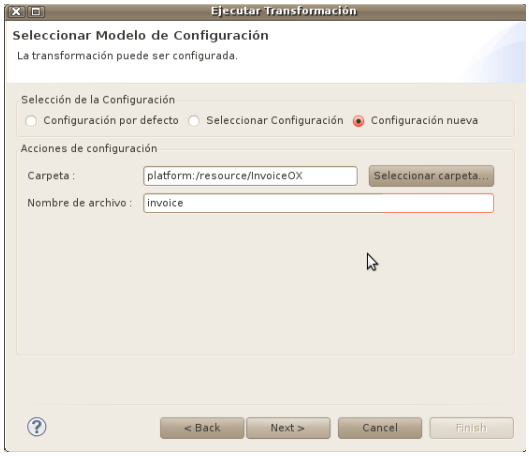
4. Antes de lanzar la transformación se comprueba que los parámetros de entrada son correctos. Es decir, que los modelos de entrada son los que deben de ser y que su contenido es el esperado (cuando se detectan errores en este punto, la transformación sólo continuará si los errores son considerados como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



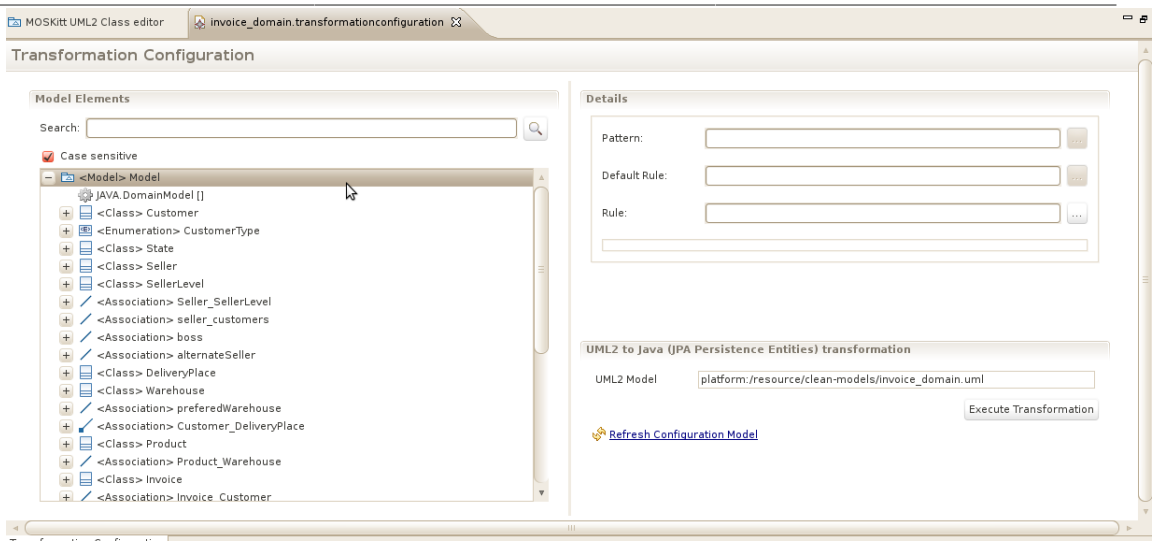
5. Una vez ha finalizado la transformación el asistente nos indica que todo ha ido bien:



Si estuviésemos interesados en configurar alguna de las etiquetas JPA u OpenXava 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 que consideremos necesarias. El editor tiene el aspecto que muestra la siguiente figura:



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

Tarea 3: Lanzar la aplicación OpenXava

Para ejecutar esta tarea seguir los pasos indicados en el tutorial *Tutorial 3: Cómo ejecutar una aplicación OpenXava con OXPortal*.

A partir de las clases Java generadas por MOSKitt, OpenXava aplica sus propias anotaciones para generar una interfaz AJAX JEE por defecto sin necesidad de especificar la Interfaz de Usuario.

Si el analista considera que necesita modificar el diseño de la interfaz de alguna de las pantallas obtenidas, puede incorporar esta información de diseño siguiendo el tutorial *Tutorial 2: Cómo incorporar diseño de interfaz de usuario en aplicaciones OpenXava con MOSKitt*.