Professional Edition v 0.2.x

Primeros pasos

Revisión: 05/03/20

Sumario

1 Introducción

Este documento explica cómo iniciar un proyecto con samsara-software.

2 Creación de proyectos con los arquetipos de Maven.

Los arquetipos de Maven son pequeños programas que nos permiten generar un proyecto con una estructura inicial. Samsara-software dispone de un arquetipo para cada una de las arquitecturas disponibles. Para crear un proyecto con un arquetipo de Maven es necesario haber configurado previamente el profile “samsara-software” en el fichero settings.xml tal y como se explica en el apartado 3.2.2 del manual de instalación: http://www.samsara-software.es/inicio/public/documentacion/manuales/instalacion_v0_1.pdf

Una vez configurado, podemos crear un proyecto para la arquitectura de referencia 1 ejecutando el siguiente comando:

```
mvn -P samsara-archetypes archetype:generate
-DarchetypeGroupId=com.samsarasoftware.archetypes
-DarchetypeArtifactId=reference-architecture-1 -DarchetypeVersion=0.2.0
-DgroupId=[GROUP-ID] -DartifactId=[ARTIFACT-ID] -DoutputDirectory=%cd%
```

Los parámetros GROUP-ID y ARTIFACT-ID se corresponden a los parámetros de Maven. El parámetro outputDirectory es el directorio donde se generará un directorio con el proyecto, cuyo nombre se corresponderá al parámetro ARTIFACT-ID.

3 Importación de proyectos Maven en eclipse

Ahora que ya hemos inicializado un proyecto, vamos a eclipse y seleccionamos las opciones “File”→”Import”→”Maven”→”Existing Maven projects” e importamos el proyecto generado en el paso anterior.

3.1.1 Preparación del entorno

Antes de poder ejecutar por primera vez cualquier generador es necesario instalar las dependencias de los generadores. Para ello, debemos ejecutar el comando *mvn clean install* en el directorio *samsara-software-utils/install-dependencies*.

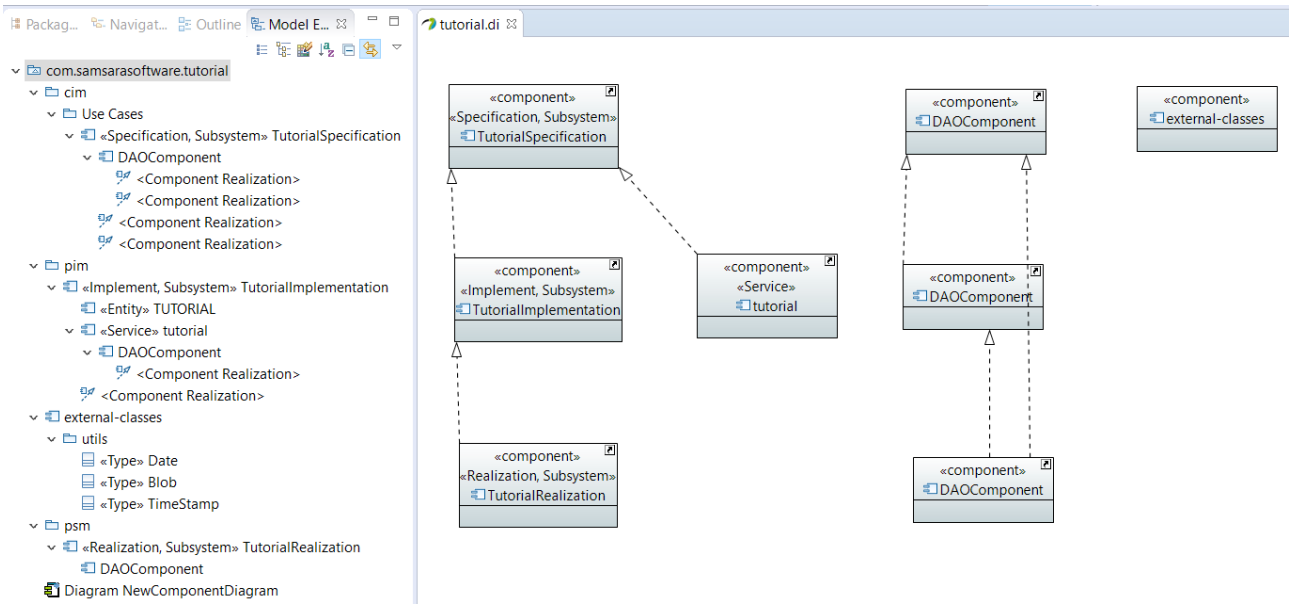
4 Estructura básica de un proyecto

Para iniciar un proyecto, necesitamos dos elementos básicos, el documento UML y las herramientas de Samsara-software. La estructura de directorios generados es la siguiente:

- *[ARTIFACT-ID]:* Directorio raíz del proyecto. El artifactId para este directorio es [ARTIFACT-ID]-root.
  - *Samsara-software-utils:* Directorio de utilizades de samsara-software.
    - *Install-dependencies:* Módulo de soporte para la instalación de las dependencias necesarias para ejecutar las utilidades.
    - *Code-generators:* Agrupa los diferentes generadores de código disponibles para la arquitectura seleccionada.
    - *Documentation-generators:* Agrupa los diferentes generadores de documentación disponibles.
    - *Model-extractors:* Agrupa los diferentes módulos extractores de modelos disponibles.
    - *Model-transformations:* Agrupa las diferentes transformaciones UML disponibles.
  - *src/main/uml:* Directorio donde se encuentran los ficheros UML.

4.1 Conceptos básicos

4.1.1 Elementos básicos del modelo



El documento UML inicial se debe estructurar de la siguiente manera:

- El nombre del modelo debe seguir la nomenclatura *<groupId>.<artifactId>* de maven.
- *Cim* o Componentes de especificación: Contienen la especificación del sistema (computer independent model), definida mediante casos de uso. Estos componentes deben estar anotados con el estereotipo *<<StandardLibrary::Specification>>*. El componente a generar debe estar anotado con el estereotipo *<<StandarLibrary::Subsystem>>*, los demás componentes de especificación que no estén anotados con este estereotipo, se consideran componentes externos, y por tanto, no se generará el código para éstos.
- *Pim* o Componentes de implementación: Contienen la implementación del sistema (platform independent model), definiendo el modelo relacional de negocio, y el modelo de servicios que implementa el sistema. Estos componentes deben estar anotados con el estereotipo *<<StandardLibrary::Implement>>*. El componente a

generar debe estar anotado con el estereotipo *<<StandarLibrary::Subsystem>>*, los demás componentes de especificación que no estén anotados con este estereotipo, se consideran componentes externos, y por tanto, no se generará el código para éstos. El componente debe contener:

  - Un componente anotado con el estereotipo *<<StandardLibrary::Entity>>* que define el modelo relacional de negocio, y **cuyo nombre debe ser el mismo que el del usuario de la base de datos donde se vaya a desplegar**.
  - Un componente anotado con el estereotipo *<<StandardLibrary::Service>>* que define el modelo de servicios del sistema. **Los nombres de los sub-componentes del PIM deben coincidir con los correspondientes nombres de sus sub-componentes representados en el CIM.**

- *Psm* o Componentes de realización: Contiene la arquitectura del sistema (platform specific model), definida mediante componentes maven, y sus relaciones, puertos, interfícies específicas.... Estos componentes deben estar anotados con el estereotipo *<<StandardLibrary::Realization>>*. El componente a generar debe estar anotado con el estereotipo *<<StandarLibrary::Subsystem>>*, los demás componentes de especificación que no estén anotados con este estereotipo, se consideran componentes externos, y por tanto, no se generará el código para éstos.
- *External-classes* o Componente de clases externas: Para poder utilizar más tipos de datos que los tipos básicos del UML Primitive Types Package, debe existir un componente con nombre "external-classes", que contendrá clases propias de Java, o clases utilizadas por samsara-software, que serán las siguientes:
  - Blob: Define un tipo básico que suele representar un array de bytes. Debe estar anotada con el estereotipo *<<StandardLibrary::Type>>*
  - Date: Define un tipo básico para fechas, sin información de tiempo. Debe estar anotada con el estereotipo *<<StandardLibrary::Type>>*
  - TimeStamp: Define un tipo básico para fechas, con información de tiempo. Debe estar anotada con el estereotipo *<<StandardLibrary::Type>>*

4.1.2 Relaciones entre elementos

- Debe existir una relación de realización desde cada componente de realización hacia el componente de implementación correspondiente.

- Debe existir una relación de realización desde cada componente de realización hacia el componente de especificación correspondiente.
- Debe existir una relación de realización desde el componente de implementación hacia el componente de especificación correspondiente.
- Debe existir una relación de realización desde el componente de servicios hacia el componente de especificación correspondiente.

4.1.3 Perfiles UML básicos

- El modelo debe tener aplicados los perfiles propios de especificación de samsara:
  - interface: Perfil de especificación de interfícies de usuario.
  - database: Perfil de especificación del modelo de negocio.
  - security: Perfil de especificación de restricciones de seguridad y roles de usuario.
  - transactionProfile: Perfil de especificación de operaciones transaccionales comunes.
- El modelo debe tener aplicados los perfiles propios de la arquitectura a utilizar, entre los ofrecidos por samsara:
  - dojo: Activa el generador de interfícies de HTML5 con el framework Dojo.
  - ejb3: Activa el generador de EJB3
  - http: Activa el generador de Servlets
  - jpa: Activa el generador de JPA.
  - maven: Activa el generador de Maven.
  - xadisk: Activa el uso de XAdisk para la transaccionalidad de ficheros.
  - jbossEap5: Activa el generador de descriptores de Jboss 5 EAP.