Práctica 5: Servicios Web

Objetivo

En esta práctica vamos a separar la lógica de negocio en servicios REST. Esto nos permitiría consumir los servicios desde diferentes aplicaciones e incluso mover esa lógica de negocio a un servidor diferente.

Configuración

Vamos a seguir trabajando en un entorno no Java EE, con lo que tenemos que realizar los siguientes pasos para poder utilizar JAX-RS:

1. Descargamos el zip con la librería Jersey[1] y lo descomprimimos en una carpeta del disco duro. Comprobamos que incluye varios ficheros *.jar dentro de los subdirectorios api, ext y lib.
2. Desde Eclipse creamos un *Dynamic Web Project* y habilitamos la creación del fichero web.xml (aparece en el último diálogo y se llega a él clickando sobre *Next* en lugar de sobre *Finish*).
3. Modificamos el web.xml para referenciar al servlet de Jersey.
4. Configuramos el *Build Path* y el *Web Deployment Assembly* para incluir todos los *.jar del primer punto.

Proveedor

Empezamos migrando a servicio REST el servlet que devuelve el listado de productos en formato JSON para la librería D3:

1. Substituir el servlet por una clase que ofrezca un servicio REST que siga devolviendo los datos en formato JSON.
2. Probad que el servicio funcione correctamente llamándolo directamente desde el navegador.
3. Actualizad la llamada correspondiente en el código JavaScript para que apunte a la URL del servicio REST.
4. Probad que el servicio funcione correctamente accediendo al dendograma (ya que a su vez lo llamará desde JavaScript).

1 http://repo1.maven.org/maven2/org/glassfish/jersey/bundles/jaxrs-ri/2.6/jaxrs-ri-2.6.zip

Práctica 5: Servicios Web

A continuación vamos a ofrecer la lógica de negocio de la aplicación como servicios REST:

1. Copiad (la borraremos más adelante) la lógica de negocio del bean creado en la práctica 4 a una nueva clase.
2. Incluid las anotaciones correspondientes para que esta clase sea un servicio REST.
3. Debe permitir obtener el listado de productos de la tienda por GET y devolverlo en formato XML. Probadlo llamando directamente al servicio desde el navegador.
4. Debe permitir dar de alta un nuevo producto recibido en formato XML por POST. De momento esta parte no podemos probarla.

Consumidor

Finalmente vamos a hacer que nuestra propia aplicación consuma los servicios REST con la lógica de negocio:

1. Eliminad del bean creado en la práctica 4 la parte correspondiente a la lógica de negocio inicial. De esta forma la lógica de negocio estará únicamente en los servicios REST de la clase implementada en el apartado anterior.
2. Modificad el servlet del usuario para que obtenga el listado de productos accediendo por GET al recurso REST correspondiente. Probad que funcione correctamente.
3. Modificad el servlet de administrador para que guarde el nuevo producto invocando por POST al servicio REST correspondiente. Comprobad que el producto se de de alta adecuadamente.

Ampliación de la Práctica

Modificad los servicios REST para que los datos ofrecidos/consumidos estén en formato JSON en lugar de en formato XML. Este código JSON no se debe generar de forma manual, sino de forma automática al estilo de JAX-RS. Para ello puede resultar de utilidad incluir en el proyecto la librería genson[2].

2 https://code.google.com/p/genson/