Share my trip

Documentación practica3

Changqu Ye

UO232770

Descripción de modificaciones

- Se ha separado el proyecto anterior en tres, uno ear, otro ejb, y otro web.
- Las clases del model implementan la interfaz Serializable, llevan anotación @XmlRootElemet y en los métodos del get llevan @XmlElement.
- Creación de interfaces Local y remote para los EJBs.
- Se ha cambiado SimpleServicesFactory por LocalEjbServiceLocator y RemoteEjbServiceLocator para acceder los EJBs de forma local o remoto usando JNDI.
- Los DAOs se obtienen la conexión através de un Datasource servido por el contendor denominado sdi3-180-ds.xml.
- Nueva package rest para los servicios rest en Web.
- Modificaciones en web.xml para permitir servicio de REST con rest/* por delante.
- Uso del filtro de autentificación BASIC para el servicio REST.
- Nueva package integration para los servicios de mensajería en EJB.
- Las canales que se usan en jms se registran en el fichero sdi3-180-jms.xml.
- Todas las clientes creadas se ejecutan en servidor de WildFly interno: http://localhost:8280/.

Descripción de Mensajería (sdi3-180.Cli-Msg)

En esa parte, la clase EnviarMensajes envia el mensaje desde una canal de cola inicial denominado TripQueue, en la capa de servicio sdi3-180.EJB, la clase MDB TripMessageListener se escucha por esa misma canal TripQueue, una vez recibido el mensaje, decide que el mensaje enviado es válido o no comprobando si el viaje que quiere enviar mensaje existe o no y si el usuario está involucrado al viaje o no.
Si el mensaje enviado es válido, el mensaje se dirige a la clase EjbMessageValido, en esta clase usa una canal de topic que se llama MessageValido para reenviar mensaje, por lo tanto, la clase RecibirMensajes del cliente recibe mensajes por esta misma canal.
Si el mensaje enviado está equivocado, el mensaje se dirige a la clase EjbMessageEquivocado y usa una canal de cola que se llama MessageEquivocado para reenviar el mensaje, pero dado que el mensaje está equivocado, simple lo muestra por la consola en forma de Log.info().
Todas las canales utlizadas están registradas en sdi3-180-jms.xml.
Y para que funcione la canal durable se ha añadido al fichero standalone-full.xml ubicado en WildFly/standalone/Configuration: <permission type="createDurableQueue" roles="guest"/> <permission type="deleteDurableQueue" roles="guest"/>
Se puede comprobar la mensajería usando user3 y user4 que ambos están involucrados en el viaje con id 31, uno manda el mensaje y otro puede recibirlo.

Descripciones de parte opcionales

Opcional1: Cliente SOAP en C#

Con el proyecto lanzado al servidor, usando la herramienta Developer Command Prompt for

VS y el comando wsdl, genera la clase con formato .cs necesario para el proyecto, en la referencia del proyecto añade WebService para eliminar los errores. Por ultimo adapta los códigos de java del Cli-SOAP al C#.

Opcional2: Mantenimiento con Timer Service

Se crea la clase ViajeMantenimiento.java en la capa de servicio para que haga servicio de timer, la clase debe poner la anotación @Singleton del EJB y usando la anotación Schedule(minute="*/3", hour="*"), hacemos que el método updateTrips se ejecuta cada 3 minutos para comprobar el estado del viaje para garantizar que todo fuese correcto.

Otras informaciones

Importar el proyecto sdi3-180.Web

Al importar el proyecto sdi3-180.Web, es posible que el proyecto de sdi3-180.Web se pierde la referencia a la librería **Web App Libraries** y el proyecto no despliegue, para solucionar el problema hay que hacer lo siguiente:

1.Right click al proyecto sdi-180.Web-> Properties

2.Seleccionar Project Facets -> Check Dyanmic Web Module and Java

3.Apply Setting.

Ahora tienes que ver la librería **Web App Libraries** con los jar por dentro y desplegar correctamente el proyecto.

Despliegue del archivo sdi3-180.ear

El archivo sdi3-180.ear se debe depositar en la carpeta S:\wildfly\standalone\deployments para que se despligue correctamente en el servidor.