Ingeniería de Sistemas y Computación
CSOF5302 – Desarrollo de Software basado en Componentes
Taller 3 – Lógica de Negocio - EJBs

Nombre Integrante 1: Andres Osorio
Nombre Integrante 2: Sebastian Cardona C
Nombre Integrante 3: Fabian Hernandez

Con respecto a la inspección del código fuente de ejemplo, responda las siguientes preguntas:

I. Ubique la plantilla *vendedores.xhtml*, revise las invocaciones que se realizan al managed bean *VendedorBean*, ubique la acción para agregar un vendedor ¿Cómo se realiza esta invocación?

La funcionalidad de agregar un vendedor la implementa el servicio responsable de Vendedores, mediante el método "agregarVendedor(vendedor)". El servicio se puede invocar gracias a la instanciación que hace el contenedor de componentes del servicio local IServicioVendedoresMockLocal a través de la anotación @EJB presente en el managed bean.

II. En la acción borrar un vendedor en la plantilla *vendedores.xhtml*, ¿Cómo se pasa el parámetro para identificar el vendedor que debe ser eliminado?

En la Vista de la aplicación, tenemos una tabla que nos muestra los datos registrados de los vendedores. Para cada registro de vendedor se tiene un botón "eliminar", cuya acción contiene el valor del ID del vendedor que se desea eliminar. Al ejecutar esta acción para un cierto vendedor, el ID del vendedor viaja como parámetro a través de la petición. Este parámetro llega por un mapa u objeto tipo Map (Key:Value), de donde se extrae el valor del parámetro asociado a la llave "vendedorId". Finalmente, este parámetro es enviado al servicio que se encarga de la eliminación del vendedor.

III. Revise las interfaces *IServicioVendedoresMockLocal* e *IServicioVendedoresMockRemote* ¿Qué tipo de interfaces son?

- La interfaz *IServicioVendedoresMockLocal* es de tipo interfaz de negocio local (ejb.Local)
- La interfaz *IServicioVendedoresMockRemote* es de tipo interfaz de negocio remoto (ejb.Remote)

IV. ¿Si usted cambia la anotación *@Statefull* a *@Stateless* del session bean *ServicioVendedoresMock* qué consecuencias habría la aplicación? ¿Qué consecuencias genera la anotación *@Singleton*?

NOTA: En el código descargado, originalmente tenemos *@Stateless, luego la pregunta que se nos plantea debe leerse, al contrario*, aunque sin ninguna diferencia lógica.

Las consecuencias están relacionadas al ciclo de vida que tendrá la sesión. Al pasarla a @Statefull, el cliente es el que tiene el control y determina cuando terminar la sesión, invocando un método anotado con un @Remove. La sesión posee un estado disponible mientras se encuentre activa. Por otro lado, al tener un estado adicional "pasivo", que no se es posible en unas sesiones de tipo @Stateless y @Singleton, tendremos la oportunidad de ejecutar las dependencias @PrePassive y @PostPassive.

En el @Statefull puedo tener más de una instancia de la sesión, mientras que en el @Singleton solo una mientras corre la aplicación.

Una de los puntos a considerar tiene que ver con el rendimiento: una sesión @Statefull mantiene el estado de conversaciones en memoria -es un estado conversacional entre cliente y servicio- y por lo tanto tiene implicaciones en el uso de la memoria. Pero todo depende de las necesidades y trade-offs que estemos dispuestos a incurrir.

V. Revise las pruebas JUnit implementadas. En el proyecto Web, en el test del servicio de seguridad (*LoginBeanTest*), la inyección de la dependencia se hace por medio de la instanciación del *mock object*

que la implementa. Por otro lado, en el proyecto EJB, la inyección de la dependencia en el test de *VendorServices* (*ServicioVendedoresMockTest*) se hace por medio de JNDI. ¿Cuál es la diferencia? Ejecute ambas pruebas dos veces, una con la aplicación desplegada en el servidor y otra sin dicho despliegue. ¿Qué puede concluir de dichas ejecuciones? Sea claro y concluyente.

Para descubrir un servicio desde un JUNIT, que funciona como un agente externo a la aplicación, se requiere de realizar una búsqueda en el directorio de interfaces usando el JNDI. Es claro que, de todas formas y debido a como está diseñada la prueba, se debe tener el servidor arriba para que se pueda acceder al servicio bajo prueba.

Prueba con servidor arriba:

Test: 21 segundos

Prueba con servidor abajo:

Test: falló. No se pueden ejecutar los métodos del servicio, pues no hay como tal un servicio.