|  |
| --- |
| **Nombre Integrante 1: Deivis Vergel**  **Nombre Integrante 2: Wilman Rincon**  **Nombre Integrante 3: Zamir Garcia** |

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

1. 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?

Se invoca mediante lenguaje EL con la expresión “#{vendedorBean.agregarVendedor}” que lee el bean de sesión inyectado en algún alcance de contexto, en este caso de sesión , que se encuentra mapeado en el archivo faces-config.xml manualmente.

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

La operación utiliza la variable utilizada en el componente Datalist que tiene como valor una lista “value="#{vendedorBean.vendedores}" , la variable actúa como un for fetch y recorre cada objeto de la lista valor y asigna el valor de un objeto , de esta forma asigna a una variable del contexto faces el atributo “#{employee.identificacion}” con el nombre “vendedorId”. Desde el el managebean se obtiene el contexto externo de faces y mapean los parámetros de la vista en donde se encuentra la variable “vendedorId” con su respectivo valor así :

FacesContext context = FacesContext.getCurrentInstance();

Map map = context.getExternalContext().getRequestParameterMap();

long vendedorId=Long.parseLong((String)map.get("vendedorId"));

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

*IServicioVendedoresMockLocal: es una interfaz Local del EJB es opcional debido a que al instanciar la implementación del Bean por defecto es accesible localmente como un servicio de la aplicación dentro de la misma JVM, esto añade al POJO características de un componente para que su ciclo de vida sea gestionado por el contenedor de aplicaciones y obtener cualidades como manejo administrado de instancias por el servidor web (Escalabilidad), bajo acoplamiento y Rehusó entre otros.*

*IServicioVendedoresMockRemote: Permite acceder el EJB desde una aplicación cliente que no se encuentre dentro del mismo JVM o en un contenedor web distinto , su acceso suele ser más lento y se realiza atreves de esta interfaz exclusivamente, si no se crea esta interfaz no podrá acceder desde otra aplicación.*

1. ¿Si usted cambia la anotación *@Stateful* a *@Stateless* del session bean *ServicioVendedoresMock* qué consecuencias habrían la aplicación? ¿Qué consecuencias genera la anotación *@Singleton*?

* Al modificar el atributo *@Stateful* a *@Stateless el EJB adquiere un ciclo de vida distinto que consiste básicamente en que cada instancia de tipo Stateless es administrada por el pool de forma libre dentro de los usuarios que utilizan la aplicación, anade principalmente dos métodos a su siclo de vida que son PREPASIVE Y POSTACTIVATE para administrar las acciones necesarias antes y despues de entrar a cache, y cada petición es atendida por demanda por los beans disponibles, es decir no guardara un estado conversacional ni les responderá la misma instancia necesariamente como si lo haría un bean Statefull.*
* Un EJB Singlenton crea una única instancia por toda la aplicación asegurando que los datos se mantendrán para todos los usuarios , sin embargo , requiere implementar mecanismos de sincronización de concurrencia para asegurar que solo será accedido por un usuario a la vez y bloquear métodos o propiedades para lectura o escritura.

1. 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.

El proyecto Web realiza pruebas unitarias sobre POJOS convencionales crea instancias de los mismos y analiza los resultados de pruebas unitarias y puede ser ejecutadas independientemente sin ningún problema, mientras que el test del proyecto EJB realiza una búsqueda sobre el contexto del servidor de aplicaciones por lo que requiere que el EJB se encuentre desplegado y disponible en el contenedor web para que sea invocado y permita crear una instancia del objeto.