

**Nombre Integrante 1:** Liliana Manjarres Villanueva

**Nombre Integrante 2:** Eduardo Arévalo Forero

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?

**Respuesta:**

Esta invocación se realiza a través de RMI por medio de la interfaz local *IServicioVendedoresMockLocal*.

*VendedorBean*

```
36  /**
37   * Relación con la interfaz que provee los servicios necesarios del vendedor
38   */
39   @EJB
40   private IServicioVendedoresMockLocal servicio;
41
118
119  /**
120   * Agrega un nuevo vendedor al sistema
121   * @throws com.losalpes.excepciones OperacionInvalidaException
122   */
123  public void agregarVendedor() throws OperacionInvalidaException
124  {
125      try
126      {
127          servicio.agregarVendedor(vendedor);
128          vendedor=new Vendedor();
129          experiencia=new ExperienciaVendedor();
130      }
131      catch (OperacionInvalidaException ex)
132      {
133          throw new OperacionInvalidaException(ex.getMessage());
134      }
135  }
```

cuando se declara la variable “servicio” de tipo *IServicioVendedoresMockLocal* tiene la anotación *@EJB*, esta anotación indica que ahí es donde se debe inyectar la implementación del método que está en la clase *ServicioVendedoresMock*.

*ServicioVendedoresMock*

```
60  @Override
61  public void agregarVendedor(Vendedor vendedor) throws OperacionInvalidaException
62  {
63      try
64      {
65          persistencia.create(vendedor);
66      }
67      catch (OperacionInvalidaException ex)
68      {
69          throw new OperacionInvalidaException(ex.getMessage());
70      }
71  }
```

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?

**Respuesta:**

El parámetro para identificar el vendedor que debe ser eliminado se pasa mediante el contexto de la sesión que se está ejecutando, se envía por parámetro desde *vendedores.xhtml* y el *VendedorBean* lo toma desde el contexto de la sesión y lo mapea a la variable “vendedorId” de tipo Long.

vendedores.xhtml

```
<h:commandButton id="DDRtn" action="#{vendedorBean.eliminarVendedor}"
                  value="Eliminar" >
    <f:param name="vendedorId" value="#{employee.identificacion}"/>
</h:commandButton>
```

VendedorBean

```
public void eliminarVendedor() throws OperacionInvalidaException
{
    FacesContext context = FacesContext.getCurrentInstance();
    Map map = context.getExternalContext().getRequestParameterMap();

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

    ...
}
```

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

**Respuesta:**

- La interfaz *IServicioVendedoresMockRemote* es remota (tiene la anotación `@Remote`)  
La implementación de sus métodos puede llamarse desde un cliente en la misma máquina o desde otra máquina.
- La interfaz *IServicioVendedoresMockLocal* es local (tiene la anotación `@Local`)  
La implementación de sus métodos puede llamarse sólo desde la misma máquina donde está desplegado el contenedor.

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

**Respuesta:**

Si cambiamos la anotación `@Stateful` a `@Stateless` del session bean `ServicioVendedoresMock` ya no se mantendría el estado durante la sesión del cliente, adicionalmente no habría una instancia reservada para un cliente en su lugar las instancias podrían estar compartidas por los clientes.

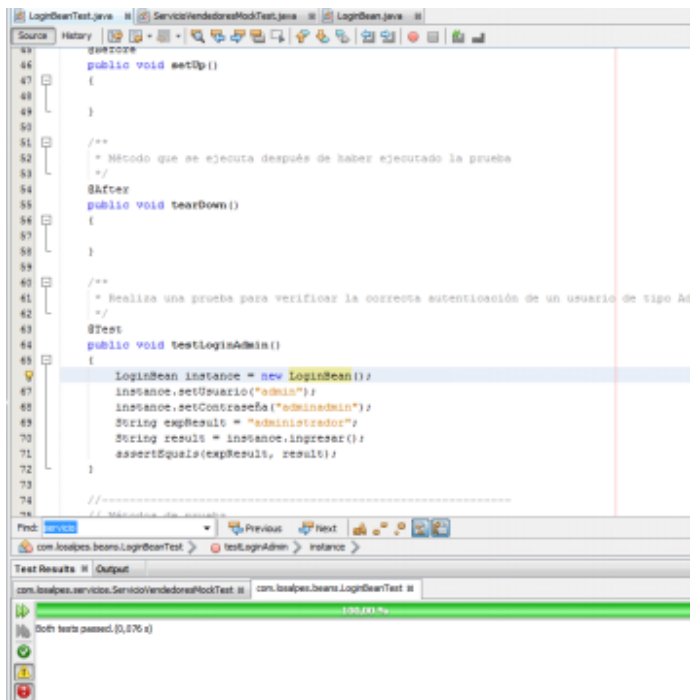
Si usamos la anotación `@Singleton` el session bean `ServicioVendedoresMock` seria instanciado una vez por la aplicación y la instancia seria compartida y accedida por los clientes de la aplicación. La sesión existiría durante todo el ciclo de vida de la aplicación.

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.

**Respuesta:**

La diferencia entre `LoginBeanTest` y `ServicioVendedoresMockTest` es la siguiente:

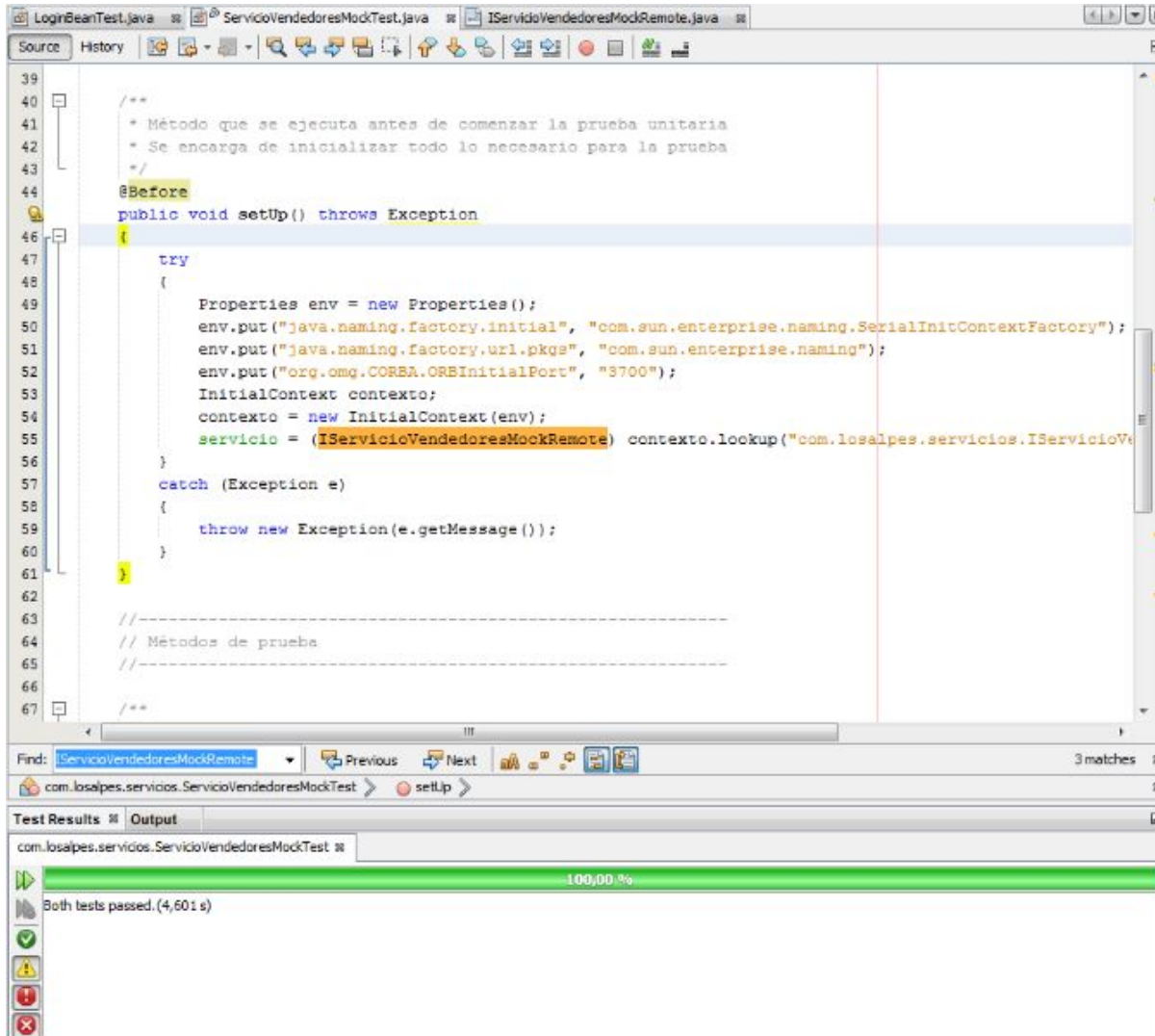
El `LoginBeanTest` instancia la clase que se va a probar se instancia directamente porque están en el mismo paquete.



The screenshot shows an IDE with the following content:

- Source Editor:** Displays the `LoginBeanTest` class. It includes a `setUp()` method, a `tearDown()` method, and a `@Test` method `testLoginAdmin()`. The `testLoginAdmin()` method creates a `LoginBean` instance and performs a login test.
- Test Results:** Shows the execution of the `testLoginAdmin` test. The output indicates that the test passed successfully.

El *ServicioVendedoresMockTest* realiza la inyección de dependencias, localizando la clase a probar dentro del directorio de beans disponibles, en este caso se espera que el servicio esté desplegado. En esta prueba existen dos contextos (el del test y el de los beans existentes).



```
39
40
41  /**
42   * Método que se ejecuta antes de comenzar la prueba unitaria
43   * Se encarga de inicializar todo lo necesario para la prueba
44   */
45  @Before
46  public void setUp() throws Exception
47  {
48      try
49      {
50          Properties env = new Properties();
51          env.put("java.naming.factory.initial", "com.sun.enterprise.naming.SerialInitContextFactory");
52          env.put("java.naming.factory.url.pkgs", "com.sun.enterprise.naming");
53          env.put("org.omg.CORBA.ORBInitialPort", "3700");
54          InitialContext contexto;
55          contexto = new InitialContext(env);
56          servicio = (IServicioVendedoresMockRemote) contexto.lookup("com.losalpes.servicios.IServicioV");
57      }
58      catch (Exception e)
59      {
60          throw new Exception(e.getMessage());
61      }
62  }
63
64  //-----
65  // Métodos de prueba
66  //-----
67  /**
```

Find: *IServicioVendedoresMockRemote* 3 matches

*com.losalpes.servicios.ServicioVendedoresMockTest* *setUp*

Test Results Output

*com.losalpes.servicios.ServicioVendedoresMockTest*

100,00 %

Both tests passed. (4,601 s)