UNIVERSIDAD DEL QUINDÍO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Información general	
Actualidad por:	Carlos Andrés Flórez Villarraga
Duración estimada en minutos:	120
Docente:	Christian Andrés Candela y Einer Zapata
Guía no.	12
Nombre de la guía:	Delegados

Información de la Guía

OBJETIVO

Lograr hacer uso de la lógica de negocio de la aplicación expuesta a través de EJB de sesión sin estado.

CONCEPTOS BÁSICOS

Manejo de Eclipse, Java, Bases de Datos, DataSource, Entidades, EJB de Sesión y GlassFish.

CONTEXTUALIZACIÓN TEÓRICA

Los delegados son un patrón de diseño usado para exponer funcionalidades de un componente, clase u aplicación. En los delegados se crea una clase que ofrece una o varias funciones. Sin embargo, dicha clase (delegado) no implementa de forma propia las funciones, en su lugar, delega la responsabilidad de dicha funcionalidad en otra clase.

Los delegados hacen las veces de intermediarios entre la clase que necesita el servicio y quien lo presta. Por ejemplo, imagine que contrata un maestro de obra pintar la fachada de su casa. Es como el maestro de obra con quien usted realiza la contratación, sin embargo, el maestro de obra delega la responsabilidad de pintar la fachada de su casa a un ayudante. Es este último quien en realidad realiza la labor encomendada. Algo muy similar pasa con los delegados. Los delegados ofrecen o exponen varias funcionalidades, pero no son ellos quienes las realizan. En su lugar, la ejecución de la tarea es delegada a otra clase o componente.

Los delegados en nuestro caso se usan para ocultar de la capa de presentación la forma en la que se accede a los EJB de negocio. Esto con el fin de que exista un único punto en el cual se deba realizar esta comunicación, para que si por algún motivo la tecnología o forma en que se invoca la funcionalidad cambia, solo deberá ser modificada en este punto. Adicionalmente al ocultar con un delegado la forma de acceder a los métodos de negocio, la capa de presentación verá las funcionalidades de la capa de negocio como funcionalidades trabajadas de forma local incluso dentro del mismo proyecto aunque las funcionalidades se estén invocando de forma remota.

PRECAUCIONES Y RECOMENDACIONES

Recuerde verificar que el servidor de aplicaciones esté en ejecución. De igual forma debe verificar que su aplicación esté siendo ejecutada en el servidor de aplicaciones.

ARTEFACTOS

Se requiere tener instalado el JDK y un IDE para el desarrollo de aplicaciones (Eclipse JEE en su última versión), un servidor de aplicaciones que cumpla con las especificaciones de JEE, para esta práctica Glassfish y el motor de base de datos MySQL.

EVALUACIÓN O RESULTADO

Se espera que el alumno pueda usar exitosamente los delegados para comunicar la capa de presentación con la capa de negocio.

Procedimiento

1. Para el desarrollo de esta guía necesitará una base de datos en mysql, un proyecto de tipo Maven – POM, el cual debe contener un proyecto Maven con soporte para el uso de JPA, uno más con soporte para EJB y otro proyecto Maven configurado para la realización de pruebas. Además debe haber una conexión a la base de datos para ser usada en la generación de las tablas.

2. Asegúrese de tener al menos una entidad y un EJB en sus proyectos, de no tenerlos deberá crearlos.

3. Verifique que su EJB tenga una interfaz remota para facilitar el acceso remoto a los métodos del EJB. No olvide adicionar las firmas de los métodos de negocio en la interfaz remota. La interfaz remota determina los servicios (métodos) que se expondrán y podrán ser invocados desde otras aplicaciones. Ejemplo:

```
@Remote
public interface PruebaEJBRemote {
	public void registrarUsuario(Usuario usuario) throws Exception;
	public Usuario autenticarUsuario(String login, String password) throws Exception;
}
```

   NOTA: Si lo prefiere al momento de crear el EJB puede seleccionar la interface remota (como se hizo en la anterior guía) lo que obligará a crear de forma automática la interface para el EJB.

4. Despliegue la aplicación para verificar lo desarrollado hasta ahora. **IMPORTANTE**: Al desplegar la aplicación acceda al log del servidor de aplicaciones y verifique el nombre bajo el cual se está publicando el EJB. Este nombre deberá ser usado más adelante. Para facilitar el desarrollo se sugiere adicionar a la interfaz remota un atributo de tipo String que esté marcado como **public**, **static** y **final,** al cual se le asigne como valor la cadena de conexión remota al EJB (la cual encontrará en el log del servidor).

```
String JNDI =
"java:global/ProyectoEAR/ProyectoNegocio/PruebaEJB!co.edu.uniquindio.prueba.ejb.PruebaEJ
BRemote";
```

   NOTA: Como podrá ver, la cadena de conexión al EJB lleva el nombre de la interfaz remota. Esta cadena se puede observar en el log del servidor glassfish, generalmente inicia con **Portable JNDI names for EJB Nombre_EJB** y a continuación entre corchetes encontrará dos cadenas separadas por coma, aquella que lleve el nombre de su interfaz remota es la que debe usar para la conexión.

5. En su EJB desarrolle al menos un método de negocio (uno de su predilección acorde con el proyecto desarrollado). No olvide desarrollar las pruebas necesarias para verificar su correcto funcionamiento. Si ya posee algún método de negocio desarrollado puede usarlo.

6. Adicione un nuevo módulo a su Proyecto padre. El nuevo módulo Maven será el Proyecto escritorio o GUI. Para ello debe dar clic derecho sobre el Proyecto padre y en la opción Maven seleccionar new Maven Module Project. Para este módulo asígnele el nombre ProyectoEscritorio y se usará la configuración por defecto, seleccione como tipo de empaquetamiento jar. Cree los paquetes vista, modelo y controlador.

7. Adicione al POM de su proyecto de escritorio la siguiente dependencia.

```
<dependency>
	<groupId>org.glassfish.main.extras</groupId>
	<artifactId>glassfish-embedded-all</artifactId>
	<version>5.1.0</version>
	<scope>provided</scope>
</dependency>
```

Además, adicione como dependencia el Proyecto de Negocio EJB.

8. Ahora al interior del paquete `modelo` del Proyecto Escritorio cree una clase delegado (PruebaDelegado) y declare como atributo de clase la interfaz de negocio remota que se creó previamente. Los delegados son fachadas que se encargan de encapsular la lógica referente a la creación e invocación de los objetos de negocio. Los delegados son usados principalmente con el patrón Layer, evitando así que las capas de presentación principalmente se contaminen con la lógica de la capa de negocio.

```
public class PruebaDelegado {
	private PruebaEJBRemote pruebaEJB;
}
```

En este punto, acceda al menú Source, y allí seleccione la opción Generate Delegate Methods. Seleccione los métodos de la capa de negocio.

Para finalizar, después de crear los métodos delegados, ponga su delegado a implementar la interfaz remota. Este paso no es necesario, sin embargo, nos permitirá obtener alertas cuando creemos nuevos métodos de negocio y no hayan sido creados en los delegados.

```
public class PruebaDelegado implements PruebaEJBRemote{
	private PruebaEJBRemote pruebaEJB;
}
```

9. Adicione un constructor privado y sin parámetros a su delegado.

10. En el constructor obtenga una instancia de su EJB. Para ello deberá obtener los EJB de forma remota.

```
private PruebaDelegado() {
	pruebaEJB = (PruebaEJBRemote) new InitialContext().lookup( PruebaEJBRemote.JDNI );
}
```

Debe tener en cuenta que el `InitialContext` hace parte del paquete `javax.naming`, verifique su correcta importación. Adicionalmente el `lookup` puede llegar a arrojar una excepción por lo que requiere que lo ponga en un `try-catch`.

11. Dado que no será necesario tener más de una instancia de su delegado es una buena práctica aplicar en el patrón singleton. Esto puede lograrlo de varias formas, una de ellas consiste en:

- Cambiar el alcance del constructor de la clase de **`public`** a **`private`** (lo cual ya se hizo en el punto 10).

- Adicionar un atributo del mismo tipo de la clase, con los modificadores **`public`** y **`static`**. El cual sea inicializado a través de un método de instancia así:

```
public class PruebaDelegado implements PruebaEJBRemote{

	public static PruebaDelegado pruebaDelegado = instancia();

	private static PruebaDelegado instancia() {

		if(pruebaDelegado==null) {
			pruebaDelegado = new PruebaDelegado();
			return pruebaDelegado;
		}

		return pruebaDelegado;
	}
}
```

12. Ahora, debe crear una ventana que muestre un formulario con los campos necesarios para hacer uso de su método de negocio, como es un proyecto de escritorio puede hacer uso de `JavaFX` o `Swing`. Por ejemplo, si su método de negocio es autenticar usuario, deberá crear un formulario con los campos usuario y contraseña, además de un botón que le permite ingresar al sistema.

13. Use una las entidades creadas previamente en su proyecto persistencia para crear en la capa de negocio un EJB con las funciones (con las validaciones necesarias) para gestionar dicha entidad (insertar, actualizar, listar, buscar y borrar).

14. En su proyecto de escritorio cree un nuevo delegado para su nuevo EJB.

15. Ahora en su aplicación de escritorio cree una interfaz gráfica que le permita la interacción con el usuario para la implementación de un CRUD (creación, actualización, búsqueda y borrado) de una de sus entidades. **Construya** la GUI para los requisitos de la persona de tipo administrador del proyecto.

16. Ejecute su aplicación y compruebe el correcto funcionamiento de la misma.