

Editar un Cliente con PrimeFaces y Spring Boot



Guía: Implementación de la Funcionalidad de Edición de Clientes

Introducción

En esta guía, vamos a implementar la funcionalidad de edición de clientes en nuestra aplicación **Zona Fit Gym**. Para ello, agregaremos un botón de **Editar** en cada fila de la tabla de clientes, lo que permitirá abrir la ventana modal con los datos del cliente seleccionado.

Los puntos clave de esta guía son:

- Agregar un botón de **Editar** en la tabla de clientes.
- Cargar la información del cliente en la ventana modal.
- Implementar la funcionalidad de actualización en el backend.
- Actualizar la interfaz para reflejar los cambios sin recargar la página.

1 Modificación de la interfaz en `index.xhtml`

📍 Ubicación del archivo: `src/main/webapp/index.xhtml`

A continuación, agregaremos un botón de edición en la tabla de clientes.

Código:

```
<p:column headerText="Acciones">
    <p:commandButton value="Editar" icon="pi pi-pencil"
        update=":forma-modal:cliente-ventana"
        process="@this"
        oncomplete="PF('ventanaModalCliente').show()"
        class="ui-button-primary"
        style="margin-right: .5rem">
        <f:setPropertyActionListener
            value="#{cliente}"
            target="#{indexControlador.clienteSeleccionado}" />
        <p:resetInput target=":forma-modal:cliente-ventana"/>
    </p:commandButton>
</p:column>
```

Explicación:

- Se añade una **nueva columna** para los botones de acción.
- Se usa un **p:commandButton** con:
 - **Etiqueta:** "Editar"
 - **Ícono:** "pi-pencil" (lápiz)
 - **update:** Refresca la ventana modal antes de mostrarla.
 - **process:** Procesa solo el botón (@this) y sus acciones asociadas, sin afectar ni procesar otros elementos o formularios dentro de la misma página, es una optimización vía Ajax.
 - **oncomplete:** Ejecuta `PF('ventanaModalCliente').show()`, lo que muestra la ventana modal.
- Se usa `<f:setPropertyActionListener>` para asignar el cliente seleccionado al controlador.
- Se usa `<p:resetInput>` para limpiar los valores previos del formulario.

◆ Orden de ejecución de los métodos

Cuando se hace clic en el botón "Editar", se ejecutan los métodos en el siguiente orden:

1. `<f:setPropertyActionListener>`

- Establece el cliente seleccionado en el **IndexControlador** (asigna el objeto `cliente` al atributo `clienteSeleccionado` en el controlador).
- Es lo primero que se ejecuta para que el formulario modal pueda llenarse con los datos del cliente seleccionado.

2. `<p:resetInput>`

- Limpia los valores previos del formulario modal para evitar datos residuales en los campos de entrada.
- Esto se ejecuta **después de que se selecciona el cliente**, pero antes de actualizar y mostrar la ventana modal.

3. `update="" :forma-modal:cliente-ventana"`

- Actualiza el contenido de la ventana modal (`cliente-ventana`) con los nuevos datos del cliente seleccionado.
- Esto asegura que los campos del formulario se actualicen con los datos correctos.

4. `process="@this"`

- Procesa solo el botón que se presionó (`@this`), sin enviar toda la tabla o el formulario.
- Esto evita que se procesen datos innecesarios y optimiza el rendimiento.

5. `oncomplete="PF('ventanaModalCliente').show()"`

- Cuando la actualización del formulario (`update`) ha terminado, este evento **muestra la ventana modal** ejecutando el método `show()` de PrimeFaces.

2 Modificación en `IndexControlador.java`

📌 Ubicación del archivo: `src/main/java/gm/zona_fit/controlador/IndexControlador.java`

Ahora debemos modificar el backend para procesar la actualización de los datos.

Código:

```
public void guardarCliente(){
    logger.info("cliente a guardar: " + this.clienteSeleccionado);
    // Agregar (insert)
    if(this.clienteSeleccionado.getId() == null){
        this.clienteServicio.guardarCliente(this.clienteSeleccionado);
        this.clientes.add(this.clienteSeleccionado);
        FacesContext.getCurrentInstance().addMessage(null,
            new FacesMessage("Cliente Agregado"));
    }
    // Modificar (update)
    else{
        this.clienteServicio.guardarCliente(this.clienteSeleccionado);
        FacesContext.getCurrentInstance().addMessage(null,
```

```

        new FacesMessage("Cliente Actualizado"));

    }
    // Ocultar la ventana modal
    PrimeFaces.current().executeScript("PF('ventanaModalCliente').hide()");
    // Actualizar la tabla usando ajax
    PrimeFaces.current().ajax().update("forma-clientes:mensajes",
        "forma-clientes:clientes-tabla");
    // Reset del objeto cliente seleccionado
    this.clienteSeleccionado = null;
}

```

Explicación:

1. Se verifica si el cliente **tiene un ID**:
 - o Si **no tiene ID**, se está creando un nuevo cliente.
 - o Si **tiene ID**, se está actualizando.
 2. Se usa `clienteServicio.guardarCliente(clienteSeleccionado)` para persistir los cambios.
 3. Se muestra un mensaje de confirmación usando `FacesContext.getCurrentInstance().addMessage(...)`.
 4. Se **oculta** la ventana modal con `PrimeFaces.current().executeScript(...)`.
 5. Se **actualiza** la tabla de clientes con `PrimeFaces.current().ajax().update(...)`.
 6. Se **reinicia** `clienteSeleccionado = null;` para evitar conflictos en futuras ediciones.
-

3 Código Final Completo

🔗 Código final de `index.xhtml`

```

<!DOCTYPE html>
<h:html xmlns:h="http://xmlns.jcp.org/jsf/html"
         xmlns:f="http://xmlns.jcp.org/jsf/core"
         xmlns:p="http://primefaces.org/ui">
    <h:head>
        <title>Zona Fit (GYM)</title>
        <link rel="stylesheet" href="https://unpkg.com/primeflex@latest/primeflex.css"/>
        <link rel="stylesheet" href="https://unpkg.com/primeflex@latest/themes/primeone-dark.css"/>
    </h:head>
    <h:body>
        <div class="card">
            <h:form id="forma-clientes">
                <p:growl id="mensajes" showDetails="true"/>
                <!-- Menubar -->
                <div class="card">
                    <p:menubar>

```

```
<p:menuitem value="Inicio" icon="pi pi-fw pi-user"
             update=":forma-clientes:clientes-tabla"
             actionListener="#{indexControlador.cargarDatos}"/>
<p:menuitem value="Nuevo Cliente" icon="pi pi-fw pi-plus"
             actionListener="#{indexControlador.agregarCliente}"
             update=":forma-modal:cliente-ventana"
             oncomplete="PF('ventanaModalCliente').show()"/>
</p:menubar>
</div>
<!-- DataTable -->
<div class="card">
    <p:dataTable value="#{indexControlador.clientes}" var="cliente"
                  id="clientes-tabla" widgetVar="clientesTabla">
        <f:facet name="header">
            <div class="flex justify-content-center flex-wrap
                    card-container yellow-container">
                <div class="flex align-items-center
                    justify-content-center
                    w-20rem h-4rem bg-yellow-500 font-bold
                    text-gray-900 border-round m-2">
                    <h3>Zona Fit (GYM)</h3>
                </div>
            </div>
        </f:facet>

        <p:column headerText="Id">
            <h:outputText value="#{cliente.id}"/>
        </p:column>

        <p:column headerText="Nombre">
            <h:outputText value="#{cliente.nombre}"/>
        </p:column>

        <p:column headerText="Apellido">
            <h:outputText value="#{cliente.apellido}"/>
        </p:column>

        <p:column headerText="Membresía">
            <h:outputText value="#{cliente.membresia}"/>
        </p:column>

        <p:column>
            <p:commandButton value="Editar" icon="pi pi-pencil"
                            update=":forma-modal:cliente-ventana"
                            process="@this"
```

```
oncomplete="PF('ventanaModalCliente').show()"
    class="ui-button-primary"
    style="margin-right: .5rem">
<f:setPropertyActionListener
    value="#{cliente}"
    target="#{indexControlador.clienteSeleccionado}"/>
<p:resetInput target=":forma-modal:cliente-ventana"/>
</p:commandButton>
</p:column>

</p:dataTable>
</div>
</h:form>
<h:form id="forma-modal">
<p:dialog header="Detalles Cliente" showEffect="fade"
    modal="true" widgetVar="ventanaModalCliente"
    responsive="true">
<p:outputPanel id="cliente-ventana" class="ui-fluid">
<div class="field">
<p:outputLabel for="nombre">Nombre</p:outputLabel>
<p:inputText id="nombre"
    value="#{indexControlador.clienteSeleccionado.nombre}"
    required="true"/>
</div>
<div class="field">
<p:outputLabel for="apellido">Apellido</p:outputLabel>
<p:inputText id="apellido"
    value="#{indexControlador.clienteSeleccionado.apellido}"/>
</div>
<div class="field">
<p:outputLabel for="membresia">Membresía</p:outputLabel>
<p:inputNumber id="membresia"
    value="#{indexControlador.clienteSeleccionado.membresia}"
    required="true"/>
</div>
</p:outputPanel>
<f:facet name="footer">
<p:commandButton value="Guardar" icon="pi pi-check"
    process="cliente-ventana @this"
    actionListener="#{indexControlador.guardarCliente}"/>
<p:commandButton value="Cancelar" icon="pi pi-times"
    onclick="PF('ventanaModalCliente').hide()"
    class="ui-button-secondary" type="button"/>
</f:facet>
</p:dialog>
```

```
</h:form>
</div>
</h:body>
</h:html>
```

📌 Código final de IndexControlador.java

```
package gm.zona_fit.controlador;

import gm.zona_fit.modelo.Cliente;
import gm.zona_fit.servicio.IClienteServicio;
import jakarta.annotation.PostConstruct;
import jakarta.faces.application.FacesMessage;
import jakarta.faces.context.FacesContext;
import jakarta.faces.view.ViewScoped;
import lombok.Data;
import org.primefaces.PrimeFaces;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import java.util.List;
import org.slf4j.Logger;

@Component
@Data
@ViewScoped
public class IndexControlador {

    @Autowired
    IClienteServicio clienteServicio;
    private List<Cliente> clientes;
    private Cliente clienteSeleccionado;
    private static final Logger logger = LoggerFactory.getLogger(IndexControlador.class);

    @PostConstruct
    public void init(){
        cargarDatos();
    }

    public void cargarDatos(){
        this.clientes = this.clienteServicio.listarClientes();
        this.clientes.forEach(cliente -> logger.info(cliente.toString()));
    }
}
```

```
public void agregarCliente(){
    this.clienteSeleccionado = new Cliente();
}

public void guardarCliente(){
    logger.info("cliente a guardar: " + this.clienteSeleccionado);
    // Agregar (insert)
    if(this.clienteSeleccionado.getId() == null){
        this.clienteServicio.guardarCliente(this.clienteSeleccionado);
        this.clientes.add(this.clienteSeleccionado);
        FacesContext.getCurrentInstance().addMessage(null,
            new FacesMessage("Cliente Agregado"));
    }
    // Modificar (update)
    else{
        this.clienteServicio.guardarCliente(this.clienteSeleccionado);
        FacesContext.getCurrentInstance().addMessage(null,
            new FacesMessage("Cliente Actualizado"));
    }
    // Ocultar la ventana modal
    PrimeFaces.current().executeScript("PF('ventanaModalCliente').hide()");
    // Actualizar la tabla usando ajax
    PrimeFaces.current().ajax().update("forma-clientes:mensajes",
        "forma-clientes:clientes-tabla");
    // Reset del objeto cliente seleccionado
    this.clienteSeleccionado = null;
}
}
```

4 Ejecución del Proyecto

Para ejecutar el proyecto y probar la funcionalidad de edición de clientes, sigue estos pasos:

1. Abrir el proyecto en IntelliJ IDEA.
2. Ejecutar la aplicación:
 - o Ir al archivo ZonaFitWeb.java
 - o Clic derecho → Run 'ZonaFitWeb.main()'
3. Abrir el navegador y acceder a la aplicación.
4. Probar la edición:
 - o Ir a la tabla de clientes.
 - o Hacer clic en el botón **Editar** en cualquier cliente.
 - o Modificar los datos en la ventana modal.
 - o Hacer clic en **Guardar**.
 - o Verificar que los datos se actualicen en la tabla y en la base de datos.

The screenshot shows a web application titled "Zona Fit (GYM)" running on "localhost:8080/index.xhtml". A modal dialog titled "Detalles Cliente" is open, prompting for "Nombre *" (Carlos1), "Apellido" (Gomez2), and "Membresía *" (3,003). The background displays a table of client records with columns "Id", "Nombre", and "Apellido". The record for Id 6 (Carlos) is currently selected.

Id	Nombre	Apellido	Membresía
1	Gabriel		
2	Paola		
6	Carlos		

The screenshot shows the same web application after the update. A yellow banner at the top center says "Zona Fit (GYM)". To its right, a blue box displays the message "Cliente Actualizado" with an information icon. The client list table now shows the updated data: Gabriel Flores (Membresía 100), Paola Juarez (Membresía 200), and Carlos1 Gomez2 (Membresía 3003).

Id	Nombre	Apellido	Membresía
1	Gabriel	Flores	100
2	Paola	Juarez	200
6	Carlos1	Gomez2	3003

5 Conclusión

En esta guía, implementamos la funcionalidad para editar clientes en **Zona Fit Gym**. Ahora, cada cliente tiene un botón de **Editar** que permite abrir una ventana modal con sus datos prellenados. Al modificar los datos y guardarlos, la información se actualiza en la base de datos y en la tabla de la interfaz sin necesidad de recargar la página.

🚀 En la siguiente lección, trabajaremos en la funcionalidad de eliminación de clientes. ¡Sigue adelante con tu aprendizaje! 💪

Nota: Recuerda que tienes el archivo .zip en la sección de recursos de este video con todos los archivos del proyecto actualizado según la lección estudiada, el cual puedes utilizar para cualquier problema.

Sigue adelante con tu aprendizaje 🚀, ¡el esfuerzo vale la pena!

¡Saludos! 🙌

Ing. Ubaldo Acosta

Fundador de [GlobalMentoring.com.mx](#)