Práctica No. 5. Desplegando y Leyendo de Elementos de Control (Vistas Dinámicas)

Preparación del Entorno

- Descargar de la página web http://academicos.azc.uam.mx/jfg/pags/tarea_taller_web.html los archivos del Servidor ***Apache Tomcat***
- Instalar, agregar y arrancar el servidor
- Descargar e importar el proyecto contenido en el archivo **NavegacionDinamicaBase.zip**

Se tiene el controlador para validar al usuario a través de las clases de apoyo en el paquete negocio. De momento solo se tiene la parte que captura la invocación y presenta la vista.

Se agregará el método que reciba los datos y adicionalmente presente en pantalla la información del usuario en la pantalla de bienvenido.

Como primera opción se creará una clase que represente los datos a mostrar en la pantalla de bienvenida, en este caso los datos de un usuario y una colección de viajes. Esto será en el paquete ***uam.tdaw.dinamica.formas***

LoginFrm.java

```
package uam.tdaw.dinamica.formas;

import java.util.LinkedList;
import uam.tdaw.dinamica.clases.Cliente;
import uam.tdaw.dinamica.clases.Viaje;

public class LoginFrm {

	private LinkedList<Viaje>viajes;
	private Cliente cliente;

	public Cliente getCliente() {
		return cliente;
	}
	public void setCliente(Cliente cliente) {
		this.cliente = cliente;
	}

	public LinkedList<Viaje> getViajes() {
		return viajes;
	}
	public void setViajes(LinkedList<Viaje> viajes) {
		this.viajes = viajes;
	}
}
```

ValidacionController.java

```
@RequestMapping(value="validarUsuario", method=RequestMethod.POST)
      public ModelAndView
validarUsuario(@ModelAttribute("usuarioFrm")Usuario usuario){

            ValidarUsuario validarUsuario = new ValidarUsuario();
            Cliente cliente = validarUsuario.validarUsuario(usuario);

            System.out.println("Entró el cliente " + cliente.getUsuario());

            BienvenidaFrm bienvenidaFrm = new BienvenidaFrm();
            bienvenidaFrm.setCliente(cliente);

            ModelAndView modelo = new
ModelAndView("bienvenido","bienvenidaFrmJSP",bienvenidaFrm);

            return modelo;
      }
```

En la vista de bienvenida se colocará la instrucción para desplegar los datos del usuario.

bienvenido.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>Bienvenido ${bienvenidaFrmJSP.cliente}</h1>

<a href="nuevoViaje">Seleccionar un Nuevo Viaje</a>
<br><br>

<h2>Edita tus Viajes Pendientes</h2>
</body>
</html>
```

La siguiente instrucción permite desplegar los datos del usuario.

```
<h1>Bienvenido ${bienvenidaFrmJSP .cliente}</h1>
```

En este caso **bienvenidaFrmJSP** representa el nombre con el que se le pasa un parámetro y **cliente**

representa el atributo de la clase que se quiere desplegar, en este caso presenta el método **toString**, sin embargo se puede presentar solo un atributo, por ejemplo:

```
<h1>Bienvenido ${bienvenidaFrmJSP .cliente.nombre}</h1>
```

Despliege de una Tabla

A continuación se presentará una tabla llenada de forma dinámica, lo primero es llenar una lista con los objetos de la tabla, en este caso objetos de tipo Viaje

ValidacionController.java

```
@RequestMapping(value="validarUsuario", method=RequestMethod.POST)
      public ModelAndView
validarUsuario(@ModelAttribute("usuarioFrm")Usuario usuario){

            ValidarUsuario validarUsuario = new ValidarUsuario();
            AdministrarListas administrarListas = new AdministrarListas();

            Cliente cliente = validarUsuario.validarUsuario(usuario);
            LinkedList<Viaje>listaViajes =
administrarListas.listasViajes();

            System.out.println("Entró el cliente " + cliente.getUsuario());

            BienvenidaFrm bienvenidaFrm = new BienvenidaFrm();
            bienvenidaFrm.setCliente(cliente);
            bienvenidaFrm.setViajes(listaViajes);

            ModelAndView modelo = new
ModelAndView("bienvenido","bienvenidaFrmJSP",bienvenidaFrm);

            return modelo;
      }
```

Para desplegar la tabla de manera dinámica, se agregará el uso de una nueva *tag*

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
```

bienvenido.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
    <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>Bienvenido ${bienvenidaFrmJSP.cliente}</h1>

<a href="nuevoViaje">Seleccionar un Nuevo Viaje</a>
<br><br>

<h2>Edita tus Viajes Pendientes</h2>

<table>
      <tr>
            <th>No.</th>
            <th>Destino</th>
            <th>Preferencias</th>
            <th>Tipo de Viaje</th>
            <th>Editar</th>
            <th>Eliminar</th>
      </tr>

      <c:forEach items="${bienvenidaFrmJSP.viajes}" var="viajeHTML"
varStatus="conteo">
            <tr>
            <td>${conteo.count}</td>
            <td>${viajeHTML.destino}</td>
            <td>${viajeHTML.preferencia}</td>
            <td>${viajeHTML.tipoViaje}</td>
            <td><a href="editar">Editar Viaje</a></td>
            <td><a href="borrar">Borrar Viaje</a></td>
            </tr>
      </c:forEach>
</table>

</body>
</html>
```

Registro de un Nuevo Viaje

Se agregará una nueva liga para registrar un nuevo viaje.

bienvenido.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
    <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>Bienvenido ${bienvenidaFrmJSP.cliente}</h1>

<a href="nuevoViaje">Seleccionar un Nuevo Viaje</a>
<br><br>

<h2>Edita tus Viajes Pendientes</h2>

<table>
	<tr>
		<th>No.</th>
		<th>Destino</th>
		<th>Preferencias</th>
		<th>Tipo de Viaje</th>
		<th>Editar</th>
		<th>Eliminar</th>
	</tr>

	<c:forEach items="${bienvenidaFrmJSP.viajes}" var="viajeHTML"
varStatus="conteo">
		<tr>
		<td>${conteo.count}</td>
		<td>${viajeHTML.destino}</td>
		<td>${viajeHTML.preferencia}</td>
		<td>${viajeHTML.tipoViaje}</td>
		<td><a href="editar">Editar Viaje</a></td>
		<td><a href="borrar">Borrar Viaje</a></td>
		</tr>
	</c:forEach>
</table>

<br><br>
<a href="nuevoViaje">Seleccionar un Nuevo Viaje</a>
<br><br>

</body>
</html>
```

Se creará el método para atrapar la invocación en la clase controladora ***AdministrarViajeController***

AdministrarViajeController.java

```java
package uam.tdaw.dinamica.controladores;

import java.util.LinkedList;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import uam.tdaw.dinamica.clases.Destino;
import uam.tdaw.dinamica.clases.Preferencia;
import uam.tdaw.dinamica.clases.TipoViaje;
import uam.tdaw.dinamica.clases.Viaje;
import uam.tdaw.dinamica.negocio.AdministrarListas;

@Controller
public class AdministrarViajeController {

	@RequestMapping("nuevoViaje")
	public ModelAndView mostrarViajeFrm(){

		ModelAndView modelo = new ModelAndView("nuevo_viaje");

		AdministrarListas administrarListas = new AdministrarListas();
		LinkedList<Destino>listaDestinos =
administrarListas.listaDestinos();
		LinkedList<TipoViaje>listaTipoViaje =
administrarListas.listaTipoViaje();
		LinkedList<Preferencia>listaPreferencias =
administrarListas.listaPreferencias();

		Viaje viaje = new Viaje();

		modelo.addObject("datosViaje", viaje);
		modelo.addObject("listaDestinosVista",listaDestinos);
		modelo.addObject("listaTipoViajeVista",listaTipoViaje);
		modelo.addObject("listaPreferenciaVista",listaPreferencias);

		return modelo;
	}
}
```

En este caso no se está mapeando una forma completa, sino elementos individuales que se presentarán en la vista para elegir los datos de un ***Viaje***.

nuevo_viaje.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
    <%@ taglib uri="http://www.springframework.org/tags/form"
prefix="tag"%>



<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>Registrar un nuevo Viaje</h1>


     <tag:form action="registrarNuevoViaje" method="POST"
modelAttribute="datosViaje">

     <h3>Selecciona el Destino</h3>
     <tag:select path="destino">

          <tag:options items="${listaDestinosVista}" itemValue="clave"
itemLabel="nombre"/>

     </tag:select>

     <h3>Tipo de Viaje</h3>

     <tag:radiobuttons items="${listaTipoViajeVista}" itemValue="clave"
itemLabel="descripcion" path="tipoViaje"/>

     <h3>Preferencias</h3>
     <tag:checkboxes items="${listaPreferenciaVista}" itemValue="clave"
itemLabel="descripcion" path="seleccionPreferencia"/>

     <br><br>
     <tag:button>Registrar Viaje</tag:button>

     </tag:form>

</body>
</html>
```

Como se leerá un ***Viaje***, se mapea un objeto de tipo Viaje en el atributo *datosViaje*

```
<tag:form action="registrarNuevoViaje" method="POST"
modelAttribute="datosViaje">
```

Se mapean los elementos de la lista de destino de manera directa usando la instrucción:

```
<tag:select path="destino">
          <tag:options items="${listaDestinosVista}" itemValue="clave"
itemLabel="nombre"/>
     </tag:select>
```

En este caso, se tiene un elemento (`listaDestinosVista`) que contiene objetos de tipo ***Destino***, el cuál tiene los atributos ***clave*** y ***nombre***, se indica cuál atributo será la llave del elemento *clave*, , y cuál será el valor a desplegar *nombre*.

Lo mismo sucede con el despliegue de *Radio Buttons*

```
<tag:radiobuttons items="${listaTipoViajeVista}" itemValue="clave"
itemLabel="descripcion" path="tipoViaje"/>
```

Y con el despliegue de *Check Boxes*

```
<tag:checkboxes     items="${listaPreferenciaVista}"     itemValue="clave"
itemLabel="descripcion" path="seleccionPreferencia"/>
```

La forma queda llena, pero no se tiene una selección por defecto, para esto se ralizarán los siguientes cambios:

AdministrarViajeController.java

```
@RequestMapping("nuevoViaje")
     public ModelAndView mostrarUsuarioFrm(){

          ModelAndView modelo = new ModelAndView("nuevo_viaje");

          AdministrarListas administrarListas = new AdministrarListas();
          LinkedList<Destino>listaDestinos =
administrarListas.listaDestinos();
          LinkedList<TipoViaje>listaTipoViaje =
administrarListas.listaTipoViaje();
          LinkedList<Preferencia>listaPreferencias =
administrarListas.listaPreferencias();

          Viaje viaje = new Viaje();
```

```
            TipoViaje seleccionDefecto = listaTipoViaje.getFirst();
            listaTipoViaje.removeFirst();

            modelo.addObject("datosViaje", viaje);
            modelo.addObject("listaDestinosVista",listaDestinos);
            modelo.addObject("listaTipoViajeVista",listaTipoViaje);
            modelo.addObject("listaPreferenciaVista",listaPreferencias);
            modelo.addObject("seleccionDefectoVista", seleccionDefecto);

            return modelo;

      }
```

nuevo_viaje.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
    <%@ taglib uri="http://www.springframework.org/tags/form"
prefix="tag"%>



<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>Registrar un nuevo Viaje</h1>


      <tag:form action="registrarNuevoViaje" method="POST"
modelAttribute="datosViaje">

      <h3>Selecciona el Destino</h3>
      <tag:select path="destino">

            <tag:options items="${listaDestinosVista}" itemValue="clave"
itemLabel="nombre"/>

      </tag:select>

      <h3>Tipo de Viaje</h3>

      <tag:radiobutton path="tipoViaje" value="$
{seleccionDefectoVista.clave}"
      label="${seleccionDefectoVista.descripcion}" checked="checked"/>
```

```
	<tag:radiobuttons items="${listaTipoViajeVista}" itemValue="clave"
itemLabel="descripcion" path="tipoViaje"/>

	<h3>Preferencias</h3>
	<tag:checkboxes items="${listaPreferenciaVista}" itemValue="clave"
itemLabel="descripcion" path="seleccionPreferencia"/>

	<br><br>
	<tag:button>Registrar Viaje</tag:button>

	</tag:form>
</body>
</html>
```

Finalmente se desarrolla el método para leer los datos del Viaje

AdministrarViajeController.java

```
@RequestMapping(value="registrarNuevoViaje", method=RequestMethod.POST)
	public ModelAndView
registrarNuevoViaje(@ModelAttribute("datosViaje")Viaje viaje){

		System.out.println("Se registró el Viaje:");
		System.out.println(viaje.getDestino());
		System.out.println(viaje.getTipoViaje());

		for(int i=0;i<viaje.getSeleccionPreferencia().length;i++){
			System.out.println(viaje.getSeleccionPreferencia()[i]);
		}

		AdministrarListas administrarListas = new AdministrarListas();
		LinkedList<Viaje> listaViajes =
administrarListas.listasViajes();

		BienvenidaFrm bienvenidaFrm = new BienvenidaFrm();
		bienvenidaFrm.setViajes(listaViajes);

		ModelAndView modelo = new
ModelAndView("bienvenido","bienvenidaFrmJSP",bienvenidaFrm);

		return modelo;
	}
```

“Edición” y “Eliminación” de Viajes

Se agregará un atributo que simule la clave de un viaje, para poder manejarla en la edición y la eliminación.

Viaje.java

```
package uam.tdaw.dinamica.clases;

public class Viaje {

	private int idViaje;
	private String destino;
	private String preferencia;
	private String tipoViaje;
	private String [] seleccionPreferencia;

	public int getIdViaje() {
		return idViaje;
	}
	public void setIdViaje(int idViaje) {
		this.idViaje = idViaje;
	}
	public String getDestino() {
		return destino;
	}
	public void setDestino(String destino) {
		this.destino = destino;
	}
	public String getPreferencia() {
		return preferencia;
	}
	public void setPreferencia(String preferencia) {
		this.preferencia = preferencia;
	}
	public String getTipoViaje() {
		return tipoViaje;
	}
	public void setTipoViaje(String tipoViaje) {
		this.tipoViaje = tipoViaje;
	}
	public String[] getSeleccionPreferencia() {
		return seleccionPreferencia;
	}
	public void setSeleccionPreferencia(String[] seleccionPreferencia) {
		this.seleccionPreferencia = seleccionPreferencia;
	}

}
```

También se agregará la funcionalidad para “generar” la llave al momento de crear la lista.

AdministrarListas.java

```
public LinkedList<Viaje> listasViajes(){

		LinkedList<Viaje>lista = new LinkedList<Viaje>();
		for(int i=1;i<=5;i++){
			Viaje viaje = new Viaje();
			viaje.setIdViaje(i);
			viaje.setDestino("Destino Viaje " + i);
			viaje.setPreferencia("Preferencias del Viaje " + i);
			viaje.setTipoViaje("Tipo de Viaje " + i);

			lista.add(viaje);
		}
		return lista;

	}
```

Esta propiedad se agregará a la liga tanto para editar como para eliminar un viaje, para simular que se obtiene su información y se elimina.

bienvenido.jsp

```
<table>
	<tr>
		<th>No.</th>
		<th>Destino</th>
		<th>Preferencias</th>
		<th>Tipo de Viaje</th>
		<th>Editar</th>
		<th>Eliminar</th>
	</tr>

	<c:forEach items="${bienvenidaFrmJSP.viajes}" var="viajeHTML"
varStatus="conteo">
		<tr>
		<td>${conteo.count}</td>
		<td>${viajeHTML.destino}</td>
		<td>${viajeHTML.preferencia}</td>
		<td>${viajeHTML.tipoViaje}</td>
		<td><a href="editarViaje?claveViaje=$
{viajeHTML.idViaje}">Editar Viaje</a></td>
		<td><a href="borrarViaje?claveViaje=$
{viajeHTML.idViaje}">Borrar Viaje</a></td>
		</tr>
	</c:forEach>
</table>
```

Atrapando el elemento a Eliminar

Se simula la eliminación de un viaje y se presenta nuevamente la pantalla de bienvenida.

AdministrarViajeController.java

```
@RequestMapping("borrarViaje")
	public ModelAndView eliminarViaje(@RequestParam("claveViaje") int
idViaje){

		System.out.println("Se eliminará el viaje con id: " + idViaje);

		ValidarUsuario validarUsuario = new ValidarUsuario();
		AdministrarListas administrarListas = new AdministrarListas();

		//Cliente cliente = validarUsuario.validarUsuario(usuario);
		LinkedList<Viaje>listaViajes =
administrarListas.listasViajes();

		//System.out.println("Entró el cliente " +
cliente.getUsuario());

		BienvenidaFrm bienvenidaFrm = new BienvenidaFrm();
		//bienvenidaFrm.setCliente(cliente);
		bienvenidaFrm.setViajes(listaViajes);

		ModelAndView modelo = new
ModelAndView("bienvenido","bienvenidaFrmJSP",bienvenidaFrm);

		return modelo;
	}
```

Atrapando el elemento a Actualizar

Se simula la carga del elemento a actualizar

AdministrarViajeController.java

```
@RequestMapping("editarViaje")
	public ModelAndView
desplegarActualizarViaje(@RequestParam("claveViaje") int idViaje){

		System.out.println("Se actualizará el viaje con id: " +
idViaje);

		ModelAndView modelo = new ModelAndView("nuevo_viaje");

		AdministrarListas administrarListas = new AdministrarListas();
		LinkedList<Destino>listaDestinos =
administrarListas.listaDestinos();
```

```
            LinkedList<TipoViaje>listaTipoViaje =
administrarListas.listaTipoViaje();
            LinkedList<Preferencia>listaPreferencias =
administrarListas.listaPreferencias();

            Viaje viaje = new Viaje();

            TipoViaje seleccionDefecto = listaTipoViaje.getFirst();
            listaTipoViaje.removeFirst();

            modelo.addObject("datosViaje", viaje);
            modelo.addObject("listaDestinosVista",listaDestinos);
            modelo.addObject("listaTipoViajeVista",listaTipoViaje);
            modelo.addObject("listaPreferenciaVista",listaPreferencias);
            modelo.addObject("seleccionDefectoVista", seleccionDefecto);

            return modelo;

      }
```

Ajustes Finales

Al cargar la pantalla de “edición”, se siguen observando las leyendas para Registar un nuevo Viaje, en este caso se actualizarán las etiquetas.

En este caso se creará una variable a modo de bandera para modificar las etiquetas necesarias.

AdministrarViajeController.java

```
@RequestMapping("nuevoViaje")
      public ModelAndView mostrarViajeFrm(){

            ModelAndView modelo = new ModelAndView("nuevo_viaje");

            AdministrarListas administrarListas = new AdministrarListas();
            LinkedList<Destino>listaDestinos =
administrarListas.listaDestinos();
            LinkedList<TipoViaje>listaTipoViaje =
administrarListas.listaTipoViaje();
            LinkedList<Preferencia>listaPreferencias =
administrarListas.listaPreferencias();

            Viaje viaje = new Viaje();

            TipoViaje seleccionDefecto = listaTipoViaje.getFirst();
            listaTipoViaje.removeFirst();

            modelo.addObject("datosViaje", viaje);
            modelo.addObject("listaDestinosVista",listaDestinos);
            modelo.addObject("listaTipoViajeVista",listaTipoViaje);
```

```
            modelo.addObject("listaPreferenciaVista",listaPreferencias);
            modelo.addObject("seleccionDefectoVista", seleccionDefecto);

            modelo.addObject("bandera","REG");

            return modelo;

      }
```

AdministrarViajeController.java

```
@RequestMapping(value="registrarNuevoViaje", method=RequestMethod.POST)
      public ModelAndView
registrarNuevoViaje(@ModelAttribute("datosViaje")Viaje viaje){


            System.out.println("Se registró el Viaje:");
            System.out.println(viaje.getDestino());
            System.out.println(viaje.getTipoViaje());

            for(int i=0;i<viaje.getSeleccionPreferencia().length;i++){
                  System.out.println(viaje.getSeleccionPreferencia()[i]);
            }

            AdministrarListas administrarListas = new AdministrarListas();
            LinkedList<Viaje> listaViajes =
administrarListas.listasViajes();

            BienvenidaFrm bienvenidaFrm = new BienvenidaFrm();
            bienvenidaFrm.setViajes(listaViajes);


            ModelAndView modelo = new
ModelAndView("bienvenido","bienvenidaFrmJSP",bienvenidaFrm);
            modelo.addObject("bandera","REG");

            return modelo;


      }
```

AdministrarViajeController.java

```
@RequestMapping("editarViaje")
      public ModelAndView
desplegarActualizarViaje(@RequestParam("claveViaje") int idViaje){

            System.out.println("Se actualizará el viaje con id: " +
idViaje);

            ModelAndView modelo = new ModelAndView("nuevo_viaje");
```

```
            AdministrarListas administrarListas = new AdministrarListas();
            LinkedList<Destino>listaDestinos =
administrarListas.listaDestinos();
            LinkedList<TipoViaje>listaTipoViaje =
administrarListas.listaTipoViaje();
            LinkedList<Preferencia>listaPreferencias =
administrarListas.listaPreferencias();

            Viaje viaje = new Viaje();

            TipoViaje seleccionDefecto = listaTipoViaje.getFirst();
            listaTipoViaje.removeFirst();

            modelo.addObject("datosViaje", viaje);
            modelo.addObject("listaDestinosVista",listaDestinos);
            modelo.addObject("listaTipoViajeVista",listaTipoViaje);
            modelo.addObject("listaPreferenciaVista",listaPreferencias);
            modelo.addObject("seleccionDefectoVista", seleccionDefecto);
            modelo.addObject("bandera","ACT");

            return modelo;


      }
```

Se maneja esta bandera en la vista

nuevo_viaje.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
    <%@ taglib uri="http://www.springframework.org/tags/form"
prefix="tag"%>
    <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>

      <c:if test="${bandera == 'REG'}">
            <c:set var="leyenda" value="Registrar"/>
      </c:if>
```

```
	<c:if test="${bandera == 'ACT'}">
		<c:set var="leyenda" value="Actualizar"/>
	</c:if>

<h1>${leyenda} un Viaje</h1>



	<tag:form action="registrarNuevoViaje" method="POST"
modelAttribute="datosViaje">

	<h3>Selecciona el Destino</h3>
	<tag:select path="destino">

		<tag:options items="${listaDestinosVista}" itemValue="clave"
itemLabel="nombre"/>

	</tag:select>

	<h3>Tipo de Viaje</h3>

	<tag:radiobutton path="tipoViaje" value="$
{seleccionDefectoVista.clave}"
	label="${seleccionDefectoVista.descripcion}" checked="checked"/>
	<tag:radiobuttons items="${listaTipoViajeVista}" itemValue="clave"
itemLabel="descripcion" path="tipoViaje"/>

	<h3>Preferencias</h3>
	<tag:checkboxes items="${listaPreferenciaVista}" itemValue="clave"
itemLabel="descripcion" path="seleccionPreferencia"/>

	<br><br>
	<tag:button>${leyenda} Viaje</tag:button>

	</tag:form>

</body>
</html>
```

Incluso es posible modificar el valor del *action* a invocar por parte de una forma, en este caso se invocará un método en caso de actualizar y otro en caso de registrar.

nuevo_viaje.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
    <%@ taglib uri="http://www.springframework.org/tags/form"
prefix="tag"%>
        <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>

<c:if test="${bandera == 'REG'}">
            <c:set var="leyenda" value="Registrar"/>
            <c:set var="action" value="registrarNuevoViaje"/>
      </c:if>

      <c:if test="${bandera == 'ACT'}">
            <c:set var="leyenda" value="Actualizar"/>
            <c:set var="action" value="actualizarViaje"/>
      </c:if>

<h1>${leyenda} un Viaje</h1>

      <tag:form action="${action}" method="POST"
modelAttribute="datosViaje">

      <h3>Selecciona el Destino</h3>
      <tag:select path="destino">

            <tag:options items="${listaDestinosVista}" itemValue="clave"
itemLabel="nombre"/>

      </tag:select>

      <h3>Tipo de Viaje</h3>

      <tag:radiobutton path="tipoViaje" value="$
{seleccionDefectoVista.clave}"
      label="${seleccionDefectoVista.descripcion}" checked="checked"/>
      <tag:radiobuttons items="${listaTipoViajeVista}" itemValue="clave"
itemLabel="descripcion" path="tipoViaje"/>

      <h3>Preferencias</h3>
      <tag:checkboxes items="${listaPreferenciaVista}" itemValue="clave"
itemLabel="descripcion" path="seleccionPreferencia"/>

      <br><br>
      <tag:button>${leyenda} Viaje</tag:button>
      </tag:form>
</body>
</html>
```

AdministrarViajeController.java

```
@RequestMapping(value="actualizarViaje", method=RequestMethod.POST)
	public ModelAndView
actualizarViaje(@ModelAttribute("datosViaje")Viaje viaje){


		System.out.println("Se actualizó el Viaje:");
		System.out.println(viaje.getDestino());
		System.out.println(viaje.getTipoViaje());

		for(int i=0;i<viaje.getSeleccionPreferencia().length;i++){
			System.out.println(viaje.getSeleccionPreferencia()[i]);
		}

		AdministrarListas administrarListas = new AdministrarListas();
		LinkedList<Viaje> listaViajes =
administrarListas.listasViajes();

		BienvenidaFrm bienvenidaFrm = new BienvenidaFrm();
		bienvenidaFrm.setViajes(listaViajes);


		ModelAndView modelo = new
ModelAndView("bienvenido","bienvenidaFrmJSP",bienvenidaFrm);
		modelo.addObject("bandera","REG");

		return modelo;


	}
```

Arrastrando un Elemento

Al momento de actualizar un Viaje, es necesario contar con su id, pero este campo no debe ser visible y solo tendrá sentido al actualizar, no al insertar. Se agregará este elemento a la vista.

nuevo_viaje.jsp

```
<tag:form action="${action}" method="POST" modelAttribute="datosViaje">

	<tag:hidden path="idViaje" value="${datosViaje.idViaje}"/>

	<h3>Selecciona el Destino</h3>
	<tag:select path="destino">
		<tag:options items="${listaDestinosVista}" itemValue="clave"
itemLabel="nombre"/>
```

```
      </tag:select>

      <h3>Tipo de Viaje</h3>

      <tag:radiobutton path="tipoViaje" value="$
{seleccionDefectoVista.clave}"
      label="${seleccionDefectoVista.descripcion}" checked="checked"/>
      <tag:radiobuttons items="${listaTipoViajeVista}" itemValue="clave"
itemLabel="descripcion" path="tipoViaje"/>

      <h3>Preferencias</h3>
      <tag:checkboxes items="${listaPreferenciaVista}" itemValue="clave"
itemLabel="descripcion" path="seleccionPreferencia"/>

      <br><br>
      <tag:button>${leyenda} Viaje</tag:button>

      </tag:form>
```

Se registra el id del elemento a actualizar

AdministrarViajeController.java

```
@RequestMapping(value="actualizarViaje", method=RequestMethod.POST)
      public ModelAndView
actualizarViaje(@ModelAttribute("datosViaje")Viaje viaje){

            System.out.println("Se actualizó el Viaje:" +
viaje.getIdViaje());
            System.out.println(viaje.getDestino());
            System.out.println(viaje.getTipoViaje());

            for(int i=0;i<viaje.getSeleccionPreferencia().length;i++){
                  System.out.println(viaje.getSeleccionPreferencia()[i]);
            }

            AdministrarListas administrarListas = new AdministrarListas();
            LinkedList<Viaje> listaViajes =
administrarListas.listasViajes();

            BienvenidaFrm bienvenidaFrm = new BienvenidaFrm();
            bienvenidaFrm.setViajes(listaViajes);

            ModelAndView modelo = new
ModelAndView("bienvenido","bienvenidaFrmJSP",bienvenidaFrm);
            modelo.addObject("bandera","REG");

            return modelo;
      }
```

Recuperando al Cliente

Los datos del cliente se obtienen durante la validación, sin embargo en otras páginas es necesario presentarlo sin pasar previamente por una validación.

Para tener acceso a la información, se colocará el objeto en sesión (*session*) para que esté disponible mientras esté abierta la aplicación. Considerar que no es recomendable colocar muchos objetos en sesión, solo aquellos necesarios, normalmente los datos de un usuario.

El objeto se subirá al validarlo.

ValidacionController.java

```
@RequestMapping(value="validarUsuario", method=RequestMethod.POST)
	public ModelAndView
validarUsuario(@ModelAttribute("usuarioFrm")Usuario usuario, HttpSession
session){

		ValidarUsuario validarUsuario = new ValidarUsuario();
		AdministrarListas administrarListas = new AdministrarListas();

		Cliente cliente = validarUsuario.validarUsuario(usuario);

		session.setAttribute("datosCliente", cliente);

		LinkedList<Viaje>listaViajes =
administrarListas.listasViajes();

		System.out.println("Entró el cliente " + cliente.getUsuario());

		BienvenidaFrm bienvenidaFrm = new BienvenidaFrm();
		bienvenidaFrm.setCliente(cliente);
		bienvenidaFrm.setViajes(listaViajes);

		ModelAndView modelo = new
ModelAndView("bienvenido","bienvenidaFrmJSP",bienvenidaFrm);

		return modelo;
	}
```

Cuando se “borra” un elemento, se despliega la pantalla de bienvenida, anteriormente no se tenía acceso al cliente, ahora se recuperará de la sesión.

AdministrarViajeController.java

```
@RequestMapping("borrarViaje")
	public ModelAndView eliminarViaje(@RequestParam("claveViaje") int
idViaje, HttpSession session){

		System.out.println("Se eliminará el viaje con id: " + idViaje);
```

```
            ValidarUsuario validarUsuario = new ValidarUsuario();
            AdministrarListas administrarListas = new AdministrarListas();

            Cliente cliente = (Cliente)
session.getAttribute("datosCliente");
            LinkedList<Viaje>listaViajes =
administrarListas.listasViajes();

            System.out.println("Está activo el cliente " +
cliente.getUsuario());

            BienvenidaFrm bienvenidaFrm = new BienvenidaFrm();
            bienvenidaFrm.setCliente(cliente);
            bienvenidaFrm.setViajes(listaViajes);

            ModelAndView modelo = new
ModelAndView("bienvenido","bienvenidaFrmJSP",bienvenidaFrm);

            return modelo;
      }
```

De manera similar, se realizará la acción para después de registrar un Viaje y después de actualizarlo.

AdministrarViajeController.java

```
@RequestMapping(value="registrarNuevoViaje", method=RequestMethod.POST)
      public ModelAndView
registrarNuevoViaje(@ModelAttribute("datosViaje")Viaje viaje, HttpSession
session){


            System.out.println("Se registró el Viaje:");
            System.out.println(viaje.getDestino());
            System.out.println(viaje.getTipoViaje());

            for(int i=0;i<viaje.getSeleccionPreferencia().length;i++){
                  System.out.println(viaje.getSeleccionPreferencia()[i]);
            }

            Cliente cliente = (Cliente)
session.getAttribute("datosCliente");


            AdministrarListas administrarListas = new AdministrarListas();
            LinkedList<Viaje> listaViajes =
administrarListas.listasViajes();

            BienvenidaFrm bienvenidaFrm = new BienvenidaFrm();
```

```
            bienvenidaFrm.setViajes(listaViajes);
            bienvenidaFrm.setCliente(cliente);


            ModelAndView modelo = new
ModelAndView("bienvenido","bienvenidaFrmJSP",bienvenidaFrm);
            modelo.addObject("bandera","REG");

            return modelo;


      }
```

AdministrarViajeController.java

```
@RequestMapping(value="actualizarViaje", method=RequestMethod.POST)
      public ModelAndView
actualizarViaje(@ModelAttribute("datosViaje")Viaje viaje, HttpSession
session){


            System.out.println("Se actualizó el Viaje:" +
viaje.getIdViaje());
            System.out.println(viaje.getDestino());
            System.out.println(viaje.getTipoViaje());

            Cliente cliente = (Cliente)
session.getAttribute("datosCliente");


            for(int i=0;i<viaje.getSeleccionPreferencia().length;i++){
                  System.out.println(viaje.getSeleccionPreferencia()[i]);
            }

            AdministrarListas administrarListas = new AdministrarListas();
            LinkedList<Viaje> listaViajes =
administrarListas.listasViajes();

            BienvenidaFrm bienvenidaFrm = new BienvenidaFrm();
            bienvenidaFrm.setViajes(listaViajes);
            bienvenidaFrm.setCliente(cliente);


            ModelAndView modelo = new
ModelAndView("bienvenido","bienvenidaFrmJSP",bienvenidaFrm);


            return modelo;
      }
```

Finalmente se agrega en la pantalla de bienvenida una liga para salir de la aplicación, lo que implica regresar al login y liberar los datos del cliente de la sesión.

bienvenido.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
    <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>Bienvenido ${bienvenidaFrmJSP.cliente}</h1>

<a href="logout">Salir</a>
<h2>Edita tus Viajes Pendientes</h2>
<table>
     <tr>
          <th>No.</th>
          <th>Destino</th>
          <th>Preferencias</th>
          <th>Tipo de Viaje</th>
          <th>Editar</th>
          <th>Eliminar</th>
     </tr>

     <c:forEach items="${bienvenidaFrmJSP.viajes}" var="viajeHTML"
varStatus="conteo">
          <tr>
          <td>${conteo.count}</td>
          <td>${viajeHTML.destino}</td>
          <td>${viajeHTML.preferencia}</td>
          <td>${viajeHTML.tipoViaje}</td>
          <td><a href="editarViaje?claveViaje=$
{viajeHTML.idViaje}">Editar Viaje</a></td>
          <td><a href="borrarViaje?claveViaje=$
{viajeHTML.idViaje}">Borrar Viaje</a></td>
          </tr>
     </c:forEach>
</table>

<br><br>
<a href="nuevoViaje">Seleccionar un Nuevo Viaje</a>
<br><br>
</body>
</html>
```

ValidacionController.java

```
@RequestMapping("logout")
      public ModelAndView salirAplicacion(HttpSession session){

            ModelAndView modelo = new ModelAndView("login");
            Usuario usuario = new Usuario();
            modelo.addObject("usuarioFrm", usuario);
            session.removeAttribute("datosCliente");
            System.out.println("Saliendo y Liberando al cliente de la
sesión");

            return modelo;

      }
```