

## Práctica No. 4. Desplegando y Leyendo de Elementos de Control (Vistas Estáticas)

### Preparación del Entorno

- Descargar de la página web [http://academicos.azc.uam.mx/jfg/pags/tarea\\_taller\\_web.html](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 **LecturaElementosControlBase.zip**

Para poder leer un objeto de una vista, se deben realizar las siguientes acciones:

- “Atrapar” la invocación a la vista (**GET**)
- Mapear el objeto en un Controlador
- Desplegar la vista con la forma (*form*) a capturar
- Recibir los datos (**POST**) en el Controlador
- Desplegar una nueva vista

Crear la clase **Alumno** en el paquete **uam.tdaw.control.clases**

### Alumno.java

```
package uam.tdaw.control.clases;

public class Alumno {

    private String matricula;
    private String nombre;
    private String primerApellido;
    private String segundoApellido;

    public String getMatricula() {
        return matricula;
    }
    public void setMatricula(String matricula) {
        this.matricula = matricula;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getPrimerApellido() {
        return primerApellido;
    }
    public void setPrimerApellido(String primerApellido) {
        this.primerApellido = primerApellido;
    }
}
```

```

    }
    public String getSegundoApellido() {
        return segundoApellido;
    }
    public void setSegundoApellido(String segundoApellido) {
        this.segundoApellido = segundoApellido;
    }

    public String toString(){
        String mensaje = matricula + ", " + nombre + ", " + primerApellido
+ ", " + segundoApellido;
        return mensaje;
    }
}

```

A continuación se creará la vista, para esto se usarán las etiquetas (tags) de Spring, para esto se debe agregar la siguiente línea en el archivo JSP

```
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="tag"%>
```

### campos\_texto\_vista.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>Mostrando y Leyendo Campos de Texto</h1>

    <tag:form action="registrar" method="POST"
modelAttribute="alumnoFrm">

        <h2>Matrícula del Alumno</h2>
        <tag:input path="matricula"/>

        <h2>Nombre</h2>
        <tag:input path="nombre"/>

        <h2>Primer Apellido</h2>
        <tag:input path="primerApellido"/>

        <h2>Segundo Apellido</h2>
        <tag:input path="segundoApellido"/>

```

```

        <tag:button>Enviar Otro</tag:button>

        </tag:form>
        <br><br>
<a href="principal">Menú Principal</a>

</body>
</html>

```

Observar la línea `<tag:form action="registrar" method="POST" modelAttribute="alumnoFrm">`

Aquí se tienen las siguientes propiedades:

Atributo	Representa
<code>action</code>	El valor a ser "capturado" por alguno de los métodos en una clase controladora
<code>method</code>	El método de envío
<code>modelAttribute</code>	Como se conoce al objeto que está mapeado en la forma

Dentro de cada elemento `input` se encuentra la propiedad `path` que representa el nombre de cada uno de los atributos que se leerán en la forma.

Crear la clase *TextoController* en el paquete *uam.tdaw.control.controladores*

### TextoController.java

```

package uam.tdaw.control.controladores;

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

import uam.tdaw.control.clases.Alumno;

@Controller
public class TextoController {
    @RequestMapping(value="campos_texto")
    public ModelAndView displayLogin(){
        System.out.println("Mapeando al alumno en la forma");
        ModelAndView model = new
ModelAndView("/elementos_control/campos_texto_vista");
        Alumno alumno = new Alumno();
        model.addObject("alumnoFrm", alumno);
        return model;
    }
}

```

Este método es el encargado de capturar la invocación a la liga (`campos_texto`) y realizar el mapeo de un objeto **Alumno** a la forma a través del atributo `alumnoFrm`.

Para obtener los datos de la forma, se creará un nuevo método que atraparé la invocación al nombre especificado en el atributo `action`, en este caso `registrar`.

También se indica el tipo de objeto mapeado que se estará recibiendo.

### TextoController.java

```
package uam.tdaw.control.controladores;

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

import uam.tdaw.control.clases.Alumno;

@Controller
public class TextoController {
    @RequestMapping("campos_texto")
    public ModelAndView cargarFormaAlumno(){
        System.out.println("Mapeando al alumno en la forma");
        ModelAndView model = new
ModelAndView("/elementos_control/campos_texto_vista");
        Alumno alumno = new Alumno();
        model.addObject("alumnoFrm", alumno);
        return model;
    }

    @RequestMapping(value = "registrar", method=RequestMethod.POST)
    public ModelAndView
registrarAlumno(@ModelAttribute("alumnoFrm")Alumno alumno){

        System.out.println(alumno.getMatricula());
        System.out.println("Los datos leídos son: " +
alumno.toString());
        return new ModelAndView("/elementos_control/final");
    }
}
```

En este caso, se agrega la línea `@ModelAttribute("alumnoFrm")Alumno alumno` a los parámetros del método, indicando que se estará recuperando un cierto atributo `alumnoFrm` enviado de la forma y se mapeará como un objeto `Alumno`.

Se estará enviando a llamar una página genérica, en este caso `/elementos_control/final`

Se agregará la lectura de un campo de contraseña y uno de área de texto, para esto, se agregarán los siguientes atributos a la clase `alumno`, junto con sus respectivos métodos `get/set` y se agregan al `toString()`. No es necesario realizar cambios en la clase controladora.

### Alumno.java

```
private String password;
private String descripcion;

public String toString(){
    String mensaje = matricula + ", " + nombre + ", " + primerApellido
+ ", " + segundoApellido + ", " + password + ", " + descripcion;
    return mensaje;
}
```

### Lectura de datos con Radio Buttons

Se leerán datos a partir de una forma que contenga elementos de *Radio Buttons*.

### radio\_buttons\_vista.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>Mostrando y Leyendo Radio Buttons</h1>

    <tag:form action="datosRadioButtons" method="POST"
modelAttribute="opcionesRB">

        <h3>Selecciona el Género</h3>
        <tag:radiobutton path="genero" value="M" label="Masculino" checked =
"checked"/>
        <tag:radiobutton path="genero" value="F" label="Femenino"/>
```

```

    <h3>Selecciona la Edad</h3>
    <tag:radio button path="edad" value="MAY" label="Mayor de Edad"
checked = "checked" />
    <tag:radio button path="edad" value="MEN" label="Menor de Edad" />

    <h3>Selecciona la Zona Geográfica</h3>
    <tag:radio button path="zona" value="NTE" label="Zona Norte" checked
= "checked" />
    <tag:radio button path="zona" value="SUR" label="Zona Sur" />
    <tag:radio button path="zona" value="CNT" label="Zona Centro" />

    <tag:button>Enviar Datos</tag:button>

</tag:form>

<br><br>
<a href="principal">Menú Principal</a>

</body>
</html>

```

Al igual que con los datos de texto, se debe mapear la forma con una clase, por lo que se creará la clase **DatosGenerales** en el paquete **uam.tdaw.control.clases**

#### DatosGenerales.java

```

package uam.tdaw.control.clases;

public class DatosGenerales {

    private String genero;
    private String edad;
    private String zona;

    public String getGenero() {
        return genero;
    }
    public void setGenero(String genero) {
        this.genero = genero;
    }
    public String getEdad() {
        return edad;
    }
    public void setEdad(String edad) {
        this.edad = edad;
    }
    public String getZona() {
        return zona;
    }
}

```

```

    public void setZona(String zona) {
        this.zona = zona;
    }

    public String toString(){
        String mensaje="";
        mensaje = genero+", "+edad+", "+zona;
        return mensaje;
    }
}

```

De la misma forma, se creará un método que mapeé el objeto y presente la forma y otro que capture los datos, esto será en la clase **RadioButtonController** que se creará en el paquete **uam.tdaw.control.controladores**

### RadioButtonController.java

```

package uam.tdaw.control.controladores;

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

import uam.tdaw.control.clases.DatosGenerales;

@Controller
public class RadioButtonController {

    @RequestMapping("radio_buttons")
    public ModelAndView cargarFormaAlumno(){
        System.out.println("Mapeando los Radio Buttons en la forma");
        ModelAndView modelo = new
ModelAndView("/elementos_control/radio_buttons_vista");
        DatosGenerales datosGenerales = new DatosGenerales();
        modelo.addObject("opcionesRB", datosGenerales);
        return modelo;
    }

    @RequestMapping(value="datosRadioButtons",
method=RequestMethod.POST)
    public ModelAndView
registrarDatos(@ModelAttribute("opcionesRB")DatosGenerales datos){
        System.out.println("Seleccionaste los datos: " +
datos.toString());
        return new ModelAndView("/elementos_control/final");
    }
}

```

## Lectura de datos con CheckBoxes

Se leerá de dos formas, una usándolo como elemento de validación y otro para seleccionar varias opciones. La primera se asignará como un booleano y la segunda como un arreglo de cadenas (String). Para esto se creará la clase *Preferencias* en el paquete *uam.tdaw.control.clases*

### Preferencias.java

```
package uam.tdaw.control.clases;

public class Preferencias {

    private boolean acepto;
    private String[] preferencias;

    public boolean isAcepto() {
        return acepto;
    }
    public void setAcepto(boolean acepto) {
        this.acepto = acepto;
    }
    public String[] getPreferencias() {
        return preferencias;
    }
    public void setPreferencias(String[] preferencias) {
        this.preferencias = preferencias;
    }
}
```

Se desarrollará la vista para capturar los datos.

### checkboxesvista\_vista.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>Mostrando y Leyendo CheckBoxes</h1>

    <br><br>
    <tag:form action="registrarPreferencias" method="POST"
modelAttribute="opcionesCB">
```

```

<a href="principal">Menú Principal</a>
<br>
<br>

<tag:checkbox path="acepto" label="Soy un buen programador"/>

<h3>¿Que lenguajes de programación prefieres?</h3>

<tag:checkbox path="preferencias" value="JAVA" label="Java"/>
<tag:checkbox path="preferencias" value="CPP" label="C++"/>
<tag:checkbox path="preferencias" value="NET" label=".Net"/>
<tag:checkbox path="preferencias" value="C" label="C"/>

<br><br>
<tag:button>Enviar Preferencias</tag:button>

</tag:form>

</body>
</html>

```

El control para presentar la vista y recibir los datos se programará en la clase **CheckBoxController** y se creará en el paquete **uam.tdaw.control.controladores**

#### CheckBoxController.java

```

package uam.tdaw.control.controladores;

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

import uam.tdaw.control.clases.Preferencias;

@Controller
public class CheckBoxController {

    @RequestMapping("checkboxes")
    public ModelAndView mostrarCheckBoxes(){
        System.out.println("Mapeando los Checkboxes en la forma");
        ModelAndView modelo = new
ModelAndView("/elementos_control/checkboxes_vista");

        Preferencias preferencias = new Preferencias();
        modelo.addObject("opcionesCB", preferencias);
        return modelo;
    }
}

```

```

    @RequestMapping(value="registrarPreferencias",
method=RequestMethod.POST)
    public ModelAndView
recuperarOpcionesCB(@ModelAttribute("opcionesCB")Preferencias
preferencias){
        ModelAndView modelo = new
ModelAndView("/elementos_control/final");

        System.out.println("Eres buen programador: " +
preferencias.isAcepto());
        System.out.println("Seleccionaste " +
preferencias.getPreferencias().length);

        String [] listaPreferencias = preferencias.getPreferencias();

        for(int i=0;i<listaPreferencias.length;i++){
            System.out.println(listaPreferencias[i]);
        }
        return modelo;
    }
}

```

### Lectura de datos con Listas de Selección

Se leerán opciones de dos listas independientes, para esto se creará la clase **OpcionesLista** en el paquete **uam.tdaw.control.clases**

#### OpcionesLista.java

```

package uam.tdaw.control.clases;

public class OpcionesLista {

    private String licenciatura;
    private String unidad;

    public String getLicenciatura() {
        return licenciatura;
    }
    public void setLicenciatura(String licenciatura) {
        this.licenciatura = licenciatura;
    }
    public String getUnidad() {
        return unidad;
    }
    public void setUnidad(String unidad) {
        this.unidad = unidad;
    }
}

```

```

    public String toString(){
        String mensaje="";
        mensaje = licenciatura+","+unidad;
        return mensaje;
    }
}

```

De la misma forma, se completará la vista para leer las opciones.

### listas\_seleccion\_vista.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>Mostrando y Leyendo Listas de Selección</h1>

<br><br>
<a href="principal">Menú Principal</a>

    <tag:form action="leerSelecciones" method="POST"
    modelAttribute="opcionesSelect">

        <h3>Escoge la mejor carrera!!</h3>
        <tag:select path="licenciatura">

            <tag:option value="CVELIC01">Licenciatura en Ingeniería en
            Computación</tag:option>
            <tag:option value="CVELIC02">Licenciatura en Ingeniería en
            Electrónica</tag:option>
            <tag:option value="CVELIC03">Licenciatura en Ingeniería
            Civil</tag:option>
            <tag:option value="CVELIC04">Licenciatura en Ingeniería
            Ambiental</tag:option>

        </tag:select>

        <h3>Escoge la mejor Unidad de la UAM!!</h3>
        <tag:select path="unidad">

```

```

        <tag:option value="UAMA">Unidad Azcapotzalco</tag:option>
        <tag:option value="UAMI">Unidad Iztapalapa</tag:option>
        <tag:option value="UAMX">Unidad Xochimilco</tag:option>
        <tag:option value="UAMC">Unidad Cuajimalpa</tag:option>
        <tag:option value="UAML">Unidad Lerma</tag:option>
    </tag:select>

    <tag:button>Enviar Selecciones</tag:button>

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

```

Finalmente el control con los métodos para mapear y desplegar y recibir las opciones seleccionadas se muestra en la clase **ListaSelectController** en el paquete **uam.tdaw.control.controladores**

### ListaSelectController.java

```

package uam.tdaw.control.controladores;

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

import uam.tdaw.control.clases.OpcionesLista;

@Controller
public class ListaSelectController {

    @RequestMapping("listas_seleccion")
    public ModelAndView mostrarListaSelect(){
        System.out.println("Mapeando las Listas en la forma");
        ModelAndView modelo = new
ModelAndView("/elementos_control/listas_seleccion_vista");
        OpcionesLista opcionesLista = new OpcionesLista();
        modelo.addObject("opcionesSelect", opcionesLista);
        return modelo;
    }
    @RequestMapping("leerSelecciones")
    public ModelAndView
procesarOpciones(@ModelAttribute("opcionesSelect")OpcionesLista opciones){
        System.out.println("Tus opciones son:" + opciones.toString());
        ModelAndView modelo = new
ModelAndView("/elementos_control/final");
        return modelo;
    }
}

```

## Navegación entre Formas

Es posible realizar una navegación entre las formas utilizando ligas (*url*), los métodos correspondientes de cada controlador atraparán la invocación, por ejemplo en la página de elementos de texto se podrá ir a través de una liga a la página de radio buttons.

### campos\_texto\_vista.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>Mostrando y Leyendo Campos de Texto</h1>
    <br><br>

    <tag:form action="registrar" method="POST"
modelAttribute="alumnoFrm">
        <h2>Matrícula del Alumno</h2>
        <tag:input path="matricula"/>
        <h2>Nombre</h2>
        <tag:input path="nombre"/>
        <h2>Primer Apellido</h2>
        <tag:input path="primerApellido"/>
        <h2>Segundo Apellido</h2>
        <tag:input path="segundoApellido"/>
        <h2>Contraseña</h2>
        <tag:password path="password"/>
        <h2>Descripción</h2>
        <tag:textarea path="descripcion"/>

        <tag:button>Enviar</tag:button>

    </tag:form>
    <br><br>
    <a href="radio_buttons">Ir a la Página de Radio Buttons</a>
    <br><br>
<a href="principal">Menú Principal</a>

</body>
</html>
```

Sin embargo, al momento de pensar en una cierta secuencia, se produce un error debido a que nunca se invoca el método que mapea un objeto en una forma y la presenta.

Se creará la siguiente secuencia:

Elementos de Texto → Checkboxes → Listas Selección → Radio Buttons → Página Final

En el método que recibe los datos de la forma, se debe además de invocar a la página (no a la *url* a ser “capturada”) mapear el objeto correspondiente.

### TextoController.java

```
@RequestMapping(value = "registrar", method=RequestMethod.POST)
    public ModelAndView
    registrarAlumno(@ModelAttribute("alumnoFrm")Alumno alumno){

        System.out.println(alumno.getMatricula());
        System.out.println("Los datos leídos son: " +
alumno.toString());

        ModelAndView modelo = new
ModelAndView("/elementos_control/checkboxes_vista");
        Preferencias preferencias = new Preferencias();
        modelo.addObject("opcionesCB", preferencias);

        return modelo;

    }
```

### CheckBoxController.java

```
@RequestMapping(value="registrarPreferencias", method=RequestMethod.POST)
    public ModelAndView
    recuperarOpcionesCB(@ModelAttribute("opcionesCB")Preferencias
preferencias){

        ModelAndView modelo = new
ModelAndView("/elementos_control/listas_seleccion_vista");

        System.out.println("Eres buen programador: " +
preferencias.isAcepto());
        System.out.println("Seleccionaste " +
preferencias.getPreferencias().length);

        String [] listaPreferencias = preferencias.getPreferencias();

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

```

    }

    OpcionesLista opcionesLista = new OpcionesLista();
    modelo.addObject("opcionesSelect", opcionesLista);

    return modelo;
}

```

### ListaSelectController.java

```

@RequestMapping("leerSelecciones")
public ModelAndView
procesarOpciones(@ModelAttribute("opcionesSelect")OpcionesLista opciones){
    System.out.println("Tus opciones son:" + opciones.toString());
    ModelAndView modelo = new
ModelAndView("/elementos_control/radio_buttons_vista");
    DatosGenerales datosGenerales = new DatosGenerales();
    modelo.addObject("opcionesRB", datosGenerales);
    return modelo;
}

```

### Iniciando la Aplicación con una Form (Controlador)

Hasta el momento se ha cargado una página sencilla y de ahí se ha navegado hacia los controladores y otras vistas, pero es común que la primera página ya deba contener una forma, por ejemplo una página de login. Para realizar esto, se creará una clase llamada *Usuario* en el paquete *uam.tdaw.control.clases*

### Usuario.java

```

package uam.tdaw.control.clases;
public class Usuario {
    private String usuario;
    private String password;

    public String getUsuario() {
        return usuario;
    }
    public void setUsuario(String usuario) {
        this.usuario = usuario;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}

```

Se creará la vista **login.jsp** en el directorio **jsp/elementos\_control**

### login.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>

    <tag:form action="validarUsuario" method="POST"
modelAttribute="datosUsuario">
    <h3>Usuario</h3>
    <tag:input path="usuario"/>

    <h3>Password</h3>
    <tag:password path="password"/>

    <br><br>

    <tag:button>Validar Usuario</tag:button>

    </tag:form>

</body>
</html>
```

También se creará el controlador necesario para presentar la vista y posteriormente capturar los datos, posteriormente se “simulará” una validación y si es correcta, se presentará la vista de campos de texto, si no, se mostrará nuevamente la pantalla de login. La clase será **ValidarUsuarioController** en el paquete **uam.tdaw.control.controladores**

### ValidarUsuarioController.java

```
package uam.tdaw.control.controladores;

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

```

import uam.tdaw.control.clases.Alumno;
import uam.tdaw.control.clases.Usuario;

@Controller
public class ValidarUsuarioController {

    @RequestMapping("/")
    public ModelAndView mostrarLogin(){
        ModelAndView modelo = new
ModelAndView("/elementos_control/login");
        Usuario usuario = new Usuario();
        modelo.addObject("datosUsuario", usuario);
        return modelo;
    }

    @RequestMapping(value="validarUsuario",method=RequestMethod.POST)
    public ModelAndView validarUsuario(@ModelAttribute("datosUsuario")
Usuario usuario){
        ModelAndView modelo;

        if(usuario.getUsuario().compareTo("31749")==0 &&
usuario.getPassword().compareTo("31749")==0){
            System.out.println("Validado!!");
            modelo = new
ModelAndView("/elementos_control/campos_texto_vista");
            Alumno alumno = new Alumno();
            modelo.addObject("alumnoFrm", alumno);
        }else{
            System.out.println("Incorrecto!!");
            modelo = new ModelAndView("/elementos_control/login");
        }

        return modelo;
    }
}
}

```

En este caso, el método que atraparía la llamada a una url, atrapará un llamado a lo que sería una raíz @RequestMapping("/")

Esto hace que en cuanto se cargue la aplicación, se “invoque” este método, mapeé el objeto y presente la vista. Para esto es necesario eliminar la lista de archivos de bienvenida en el archivo **web.xml**

#### web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID"
version="3.0">

```

```

<display-name>NavegandoControladores</display-name>
<servlet>
  <servlet-name>controlador</servlet-name>
  <servlet-class>
    org.springframework.web.servlet.DispatcherServlet
  </servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>controlador</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
</web-app>

```

Finalmente se crea la clase para manejar la invocación al menú principal, para esto se utilizará la clase **MenuPrincipalController** en el paquete **uam.tdaw.control.controladores**

#### MenuPrincipalController.java

```

package uam.tdaw.control.controladores;

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

@Controller
public class MenuPrincipalController {

    @RequestMapping("principal")
    public ModelAndView presentarPaginaPrincipal(){
        return new ModelAndView("index");
    }
}

```