Groovy & Grails: Desarrollo rápido de aplicaciones

Sesión 6: Construir la interfaz de usuario (I)

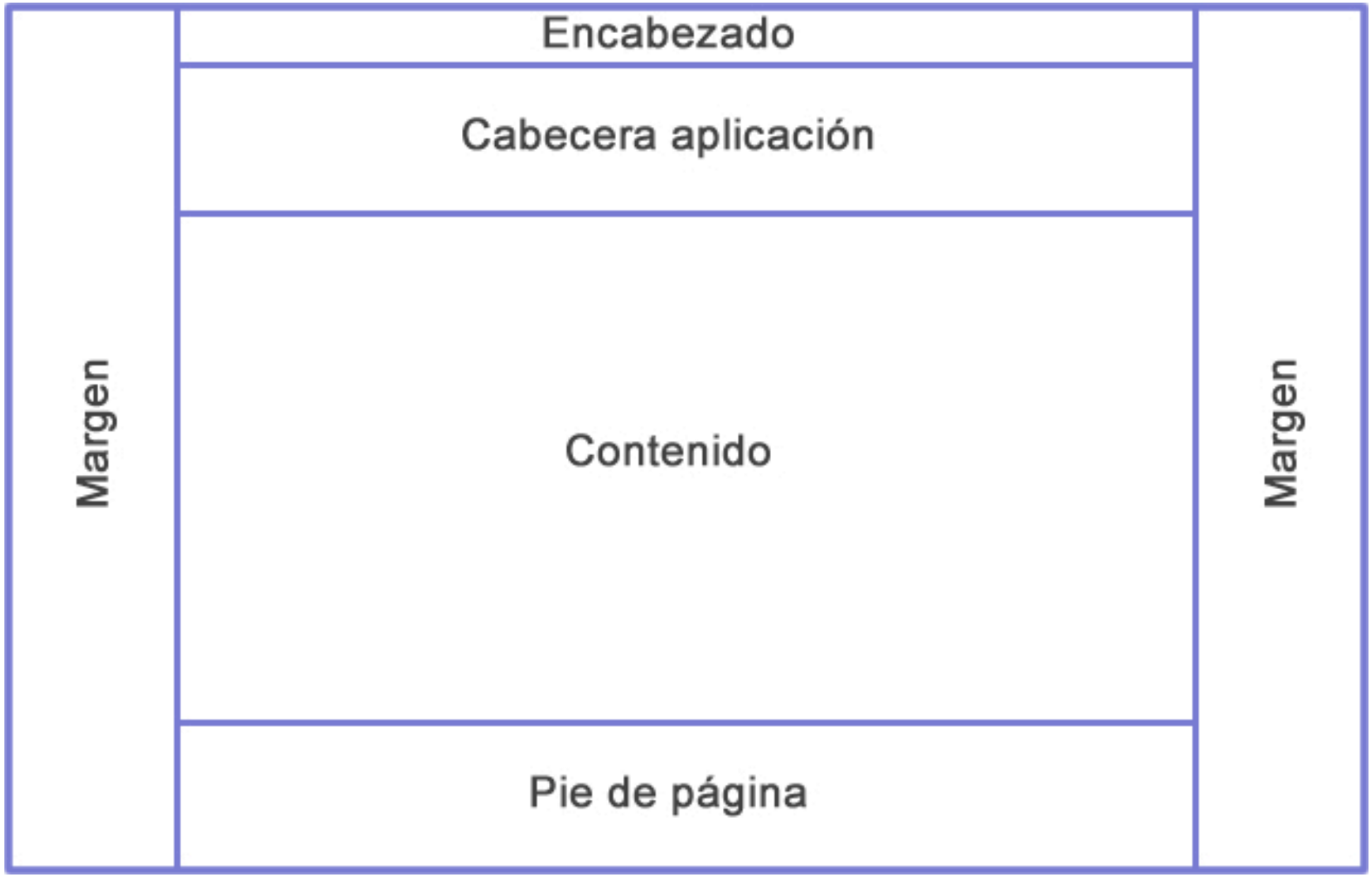
Construir la interfaz de usuario (I)

- Plantillas
- Etiquetas
- Primer contacto con los controladores

Plantillas

- Nuestra primera plantilla: el pie de página
- Otra plantilla algo más complicada: el encabezado

Plantillas


Encabezado
Cabecera aplicación
Contenido
Pie de página
Margen
Margen

Plantillas

- Se ubican en el directorio *grails-app/views*
- Por convenio, su nombre comienza por un subrayado bajo _
- Todas las plantillas relacionadas con una clase de dominio deben ir en el mismo directorio
- Las plantillas genéricas, deben estar en un directorio común
- Las plantillas tienen extensión GSP (Groovy Server Pages)

Nuestra primera plantilla: el pie de página

- Crearemos el archivo *_footer.gsp* en el directorio *grails-app/views/common*
- En el directorio *common* irán todas las plantillas comunes de la aplicación

Nuestra primera plantilla: el pie de página

```
<span class="copyright">
        © 2009 Groovy&Grails: desarrollo rápido de aplicaciones<br/>
        Aplicación Biblioteca creada por Fran García
</span>
```

Nuestra primera plantilla: el pie de página

- El archivo *grails-app/views/layout/main.gsp* es el archivo que se encarga de renderizar los contenidos de las páginas
- La etiqueta <*g:render*> nos servirá para insertar la plantilla creada en las páginas de nuestra aplicación

Nuestra primera plantilla: el pie de página

```
<html>
        <head>
                ....
        </head>
        <body>
                ....
                <g:layoutBody />
                <div id="piedepagina">
                        <g:render template="/common/footer"/>
                </div>
        </body>
</html>
```

Nuestra primera plantilla: el pie de página

- Podemos modificar la hoja de estilos desde el archivo *web-app/css/main.css*

```
#piedepagina {
    clear:both;
    text-align:center;
    padding:3px;
    border-top:1px solid #333;
}
```

Otra plantilla algo más complicada: el encabezado

- Esta plantilla mostrará los enlaces necesarios para *identificarse* en la aplicación y para *registrarse* como nuevo usuario

- Una vez el usuario se haya identificado, se debe mostrar su nombre completo acompañado de un enlace para abandonar la aplicación de forma segura

Otra plantilla algo más complicada: el encabezado

- Modificamos el archivo *grails-app/views/layout/main.gsp* para insertar el encabezado

```
.....
<div id="spinner" class="spinner" style="display:none;">
    <img src="${createLinkTo(dir:'images',file:'spinner.gif')}" alt="Spinner" />
</div>
<div id="encabezado">
    <g:render template="/common/header"/>
</div>
<div class="logo"><img src="$
{createLinkTo(dir:'images',file:'grails_logo.jpg')}" alt="Grails" /></div>
.....
```

Otra plantilla algo más complicada: el encabezado

- Creamos la plantilla en *grails-app/views/common/* y la llamaremos *_header.gsp*

```
<div id="menu">
   <nobr>
      <g:if test="${session.usuario}">
         <b>${session.usuario?.nombre} ${session.usuario?.apellidos}</b> |
         <g:link controller="usuario" action="logout">Logout</g:link>
      </g:if>
      <g:else>
         <g:link controller="usuario" action="login">Login</g:link>
      </g:else>
   </nobr>
</div>
```

Otra plantilla algo más complicada: el encabezado

- Modificamos también la hoja de estilos

```
#encabezado {
    text-align:left;
    width:778px;
    margin: 0px auto;
    padding: 5px 0;
}
#encabezado #menu{
    float: right;
    width: 240px;
    text-align:right;
    font-size:10px;
}
```

Otra plantilla algo más complicada: el encabezado

- Se comprueba si el usuario está o no identificado en el sistema con la etiqueta *<g:if>*
- En caso afirmativo, se muestra su nombre completo
- En caso contrario, se muestra un enlace para que el usuario se identifique

Otra plantilla algo más complicada: el encabezado

```
<html>
   <head>
        ….
   </head>
   <body>
     <div id="pagina">
          ….
        <div id="encabezado”><g:render template="/common/encabezado"/></div>
        <div id="cabecera”><h1>Biblioteca</h1></div>
        <div id="contenido”><g:layoutBody /></div>
        <div id="piedepagina”><g:render template="/common/piedepagina"/></div>
     </div>
   </body>
</html>
```

Otra plantilla algo más complicada: el encabezado

```
#cabecera {
    width: 778px;
    background: #FFFFFF url(../images/background-cabecera.jpg) repeat-y;
    height: 70px;
    margin: 0px auto;
}

#cabecera h1 {
    font-family: Arial, sans-serif;
    color: White;
    padding: 20px 0 0 6px;
    font-size: 1.6em;
}
```

Otra plantilla algo más complicada: el encabezado

```
body {
    margin: 0px;
    padding: 0px;
    text-align: center;
    font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
    font-style: normal;
    font-variant: normal;
    font-weight: normal;
    font-size: 13px;
    line-height: normal;
    font-size-adjust: none;
    font-stretch: normal;
    color: #333333;
}
```

Otra plantilla algo más complicada: el encabezado

```
#pagina {
    width: 778px;
    margin: 0px auto;
    padding: 4px 0;
    text-align: left;
}

#contenido {
    float: left;
    color: #000;
}
```

Otra plantilla algo más complicada: el encabezado

- Modificaremos también el archivo *grails-app/views/index.gsp* para cambiar el texto que aparece por uno que indique las características principales de la aplicación

Otra plantilla algo más complicada: el encabezado

Etiquetas

- Etiquetas lógicas
- Etiquetas de iteración
- Etiquetas de asignación
- Etiquetas de enlaces
- Etiquetas AJAX

Etiquetas

- Etiquetas de formularios
- Etiquetas de interfaz de usuario
- Etiquetas de renderizado
- Etiquetas de validación

Etiquetas lógicas

Etiqueta	Descripción
<g:if>	Evalúa una expresión y actúa en consecuencia
<g:else>	La parte *else* del bloque *if*
<g:elseif>	La parte *elseif* del bloque *if*

Etiquetas de iteración

Etiqueta	Descripción
<g:while>	Ejecuta un bloque de código mientras una condición se evalúe a cierto
<g:each>	Itera sobre una colección de objetos
<g:collect>	Itera sobre una colección y transforma los resultados tal y como se defina en el parámetro *expr*
<g:findAll>	Itera sobre una colección donde los elementos se corresponden con la expresión *Gpath* definida en el parámetro *expr*
<g:grep>	Itera sobre una colección donde los elementos se corresponden con el filtro definido en el parámetro *expr*

Etiquetas de asignación

Etiqueta	Descripción
<def>	Define una variable para que pueda ser utilizada en el ámbito de una página GSP. Esta etiqueta ya no se utiliza y se debe sustituir por la etiqueta <set>
<set>	Establece el valor de una variable utilizada en una página GSP

Etiquetas de enlaces

Etiqueta	Descripción
<g:link>	Crea un enlace HTML utilizando los parámetros pasados
<g:createLink>	Crea un enlace HTML que puede ser utilizado dentro de otras etiquetas
<g:createLinkTo>	Crea un enlace a un directorio o un fichero

Etiquetas AJAX

Etiqueta	Descripción
<g:remoteField>	Crea un campo de texto que invoca un enlace cuando se modifica el contenido de este campo
<g:remoteFunction>	Crea una función remota que se invoca con un evento del DOM
<g:remoteLink>	Crea un enlace que llama a una función remota
<g:formRemote>	Crea una etiqueta de formulario que ejecuta una llamada a un procedimiento AJAX para serializar los elementos del formulario
<g:javascript>	Incluye librerías de Javascript y scripts
<g:submitToRemote>	Crea un botón que ejecuta una llamada a un procedimiento AJAX para serializar los elementos del formulario

Etiquetas de formularios

Etiqueta	Descripción
<g:actionSubmit>	Crea un botón de tipo *submit*
<g:actionSubmitImage>	Crea un botón de tipo *submit* con una imagen
<g:checkBox>	Crea un elemento de formulario de tipo checkbox
<g:currencySelect>	Crea un campo de tipo *select* con un listado de monedas
<g:datePicker>	Crea un elemento de formulario para seleccionar una fecha con día, mes, año, hora, minutos y segundos
<g:form>	Crea un formulario
<g:hiddenField>	Crea un campo oculto
<g:localeSelect>	Crea el elemento de formulario de tipo *select* con un listado de posibles localizaciones

Etiquetas de formularios

Etiqueta	Descripción
<g:radio>	Crea un elemento de formulario de tipo *radio*
<g:radioGroup>	Crea un grupo de elementos de formulario de tipo *radio*
<g:select>	Crea un elemento de formulario de tipo *select combo box*
<g:textField>	Crea un elemento de formulario de tipo *text*
<g:textArea>	Crea un elemento de formulario de tipo *textarea*
<g:timeZoneSelect>	Crea un elemento de formulario de tipo *select* con un listado de zonas horarias

Etiquetas de interfaz de usuario

Etiqueta	Descripción
<g:richTextEditor>	Crea un elemento de formulario de tipo *textarea* para la introducción de texto con formato enriquecido. Por defecto, el editor utilizado es *fckeditor*

Etiquetas de renderizado

Etiqueta	Descripción
<g:applyLayout>	Aplica un determinado diseño a una página o una plantilla
<g:encodeAs>	Aplica una codificación dinámica a un bloque de código HTML
<g:formatDate>	Aplica el formato *SimpleDateForma* a una fecha
<g:formatNumber>	Aplica el formato *DecimalFormat* a un número
<g:layoutHead>	Muestra una determinada cabecera para una página
<g:layoutBody>	Muestra un determinado contenido para una página
<g:layoutTitle>	Muestra un determinado título para una página
<g:meta>	Muestra las propiedades *meta* de una página web

Etiquetas de renderizado

Etiqueta	Descripción
<g:render>	Muestra un modelo utilizando una plantilla
<g:renderErrors>	Muestra los errores producidos en una página
<g:pageProperty>	Muestra una propiedad de una página
<g:paginate>	Muestra los típicos botones *Anterior* y *Siguiente* y las *migas de pan* cuando se devuelven muchos resultados
<g:sortableColumn>	Muestra una columna de una tabla con la posibilidad de ordenarla

Etiquetas de validación

Etiqueta	Descripción
<g:eachError>	Itera a través de los errores producidos en una página
<g:hasErrors>	Comprueba si se ha producido algún error
<g:message>	Muestra un mensaje
<g:fieldValue>	Muestra el valor de un elemento de un formulario

Primer contacto con los controladores

- Necesitamos un sistema de identificación en el sistema
- Y un método para abandonar la aplicación de forma segura
- El usuario debe seleccionar uno cualquier de un listado de posibles usuarios sin comprobar la contraseña

Primer contacto con los controladores

- Necesitamos crear un nuevo método en el controlador de la clase *Usuario*
- El método se llamará *login()* y lo dejaremos vacío puesto que será la vista correspondiente de la clase *Usuario*

```
def login = { }
```

Primer contacto con los controladores

- Creamos la vista para la página *login.gsp*

Primer contacto con los controladores

```
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
  <meta name="layout" content="main"/>
  <title>Login</title>
 </head>
 <body>
  <div class="body">
     <g:if test="${flash.message}">
        <div class="message">
           ${flash.message}
        </div>
     </g:if>
     ....
</html>
```

Primer contacto con los controladores

```
<html>
    ....
      <p>
         Bienvenido a la aplicación Biblioteca. Por favor, identifícate.
      </p>
      <form>
         <span class="nameClear">
            <label for="login">
               Selecciona el usuario:
            </label>
         </span>
         <g:select name="login" from="${Usuario.list()}" optionKey="login"
optionValue="login"></g:select>
     ....
</html>
```

Primer contacto con los controladores

```
<html>
    ....
          <br/>
          <div class="buttons">
             <span class="button"><g:actionSubmit value="Login"
action="handleLogin"/></span>
          </div>
       </form>
   </div>
  </body>
</html>
```

Primer contacto con los controladores

- Con *<meta name="layout" content="main"* le indicamos que *main.gsp* actúa como contenedor para el archivo *login.gsp*
- El archivo *main.gsp* tiene una etiqueta *<g:layoutBody>* que será donde se inserte el código generado para *login.gsp*
- Se comprueba que no haya mensajes flash a mostrar

Primer contacto con los controladores

- Se define el formulario para seleccionar al usuario
- Podemos modificar el valor del parámetro *optionValue* para que muestre el *nombre* y *apellidos*
- Definición del botón de tipo *submit* para gestionar la petición correspondiente con el método *handleLogin()*

Primer contacto con los controladores

```
def handleLogin = {
    def usuario = Usuario.findByLogin(params.login)
    if (!usuario) {
        flash.message = "El usuario ${params.login} no existe"
        redirect(controller: 'usuario', action:'login')
        return
    }
    else {
        session.usuario = usuario
        redirect(controller:'operacion')
    }
}
```

Primer contacto con los controladores

- Se comprueba que el usuario seleccionado coincida con alguno en la base de datos con el método GORM *findByLogin()*
- En caso afirmativo, se almacenan los datos correspondientes en la variable *session* y se redirecciona al usuario para que empiece a operar
- Si el usuario no existe, se carga un mensaje flash indicando el problema

Primer contacto con los controladores

- Se comprueba que el usuario seleccionado coincida con alguno en la base de datos con el método GORM *findByLogin()*
- En caso afirmativo, se almacenan los datos correspondientes en la variable *session* y se redirecciona al usuario para que empiece a operar
- Si el usuario no existe, se carga un mensaje flash indicando el problema y se vuelve a cargar la página login.gsp

Primer contacto con los controladores

- Definimos también el método *logout()* para abandonar la sesión de forma segura
- Se destruyen los datos de la variable *session* y se redirecciona al usuario a la página *login.gsp*

```
def logout = {
    if (session.usuario){
        session.usuario = null
        redirect(controller: 'usuario', action:'login')
    }
}
```