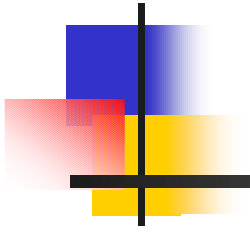
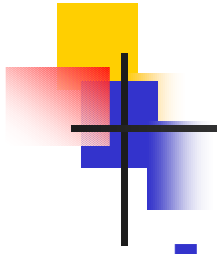


4.2 Tutorial de JSP

Introducción (1)

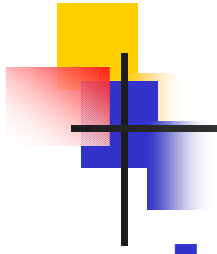
- Situación en el apartado anterior: el texto generado para el aspecto (vista) de la interfaz gráfica (HTML, WML, etc.) está mezclado con código Java
  - No es posible usar herramientas de generación de HTML, WML, etc. directamente
  - La generación del texto necesita ser escrita por una persona con conocimientos de Java => económicamente muy costoso
  - Cambios al aspecto de la interfaz gráfica requieren recompilación, creación de un nuevo fichero **`.war`** y re-arranque del servidor
    - En una aplicación web, especialmente en Internet, los cambios a la interfaz gráfica son muy frecuentes

Introducción (2)

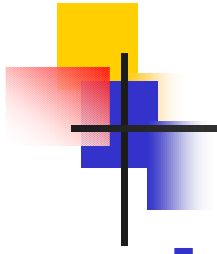


- Situación a la que queremos llegar
  - Separación de roles
    - Personas que realizan el aspecto gráfico => diseñadores gráficos o similares
      - Conocimientos de diseño gráfico y herramientas para generación de HTML y WML
    - Personas que implementan el controlador y el modelo => informáticos
      - Conocimientos de diseño e implementación
  - Se deberían poder usar directamente las herramientas de diseño de páginas web
  - Las actualizaciones al aspecto gráfico no deben provocar un re-arranque del servidor

Introducción (y 3)



- En este apartado
  - Estudiaremos los aspectos principales de JSP (Java Server Pages), como primer paso para alcanzar los anteriores hitos
  - No resolveremos el problema de la separación de roles en su totalidad
    - Una parte la tenemos resuelta, dado que hemos aprendido ha diseñar e implementar un modelo que no depende de la vista
    - Los apartados 4.3 y 4.4 completarán los conocimientos necesarios para resolver totalmente el problema
    - Los apartados 4.5 y 4.6 ilustrarán dos aplicaciones completas que reflejan la situación a la que queremos llegar

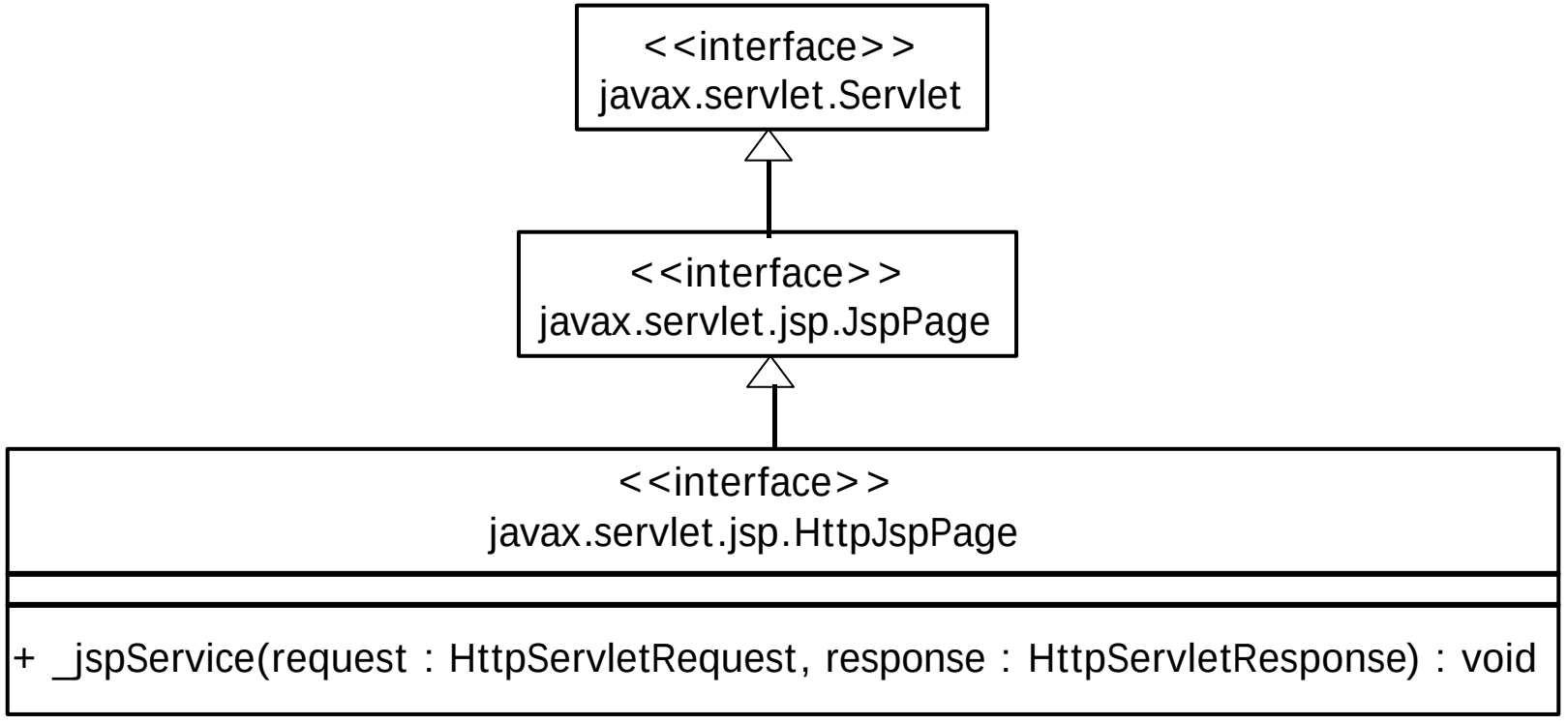


¿ Qué es JSP ? (1)

- ¿ Qué es JSP ?
  - A modo de ejemplo, una página JSP que genera HTML
    - Tiene el aspecto de una página HTML
    - Puede incluir scriptlets (scripts) para generar HTML dinámicamente
    - Típicamente los scriptlets se escriben en Java

¿ Qué es JSP ? (2)

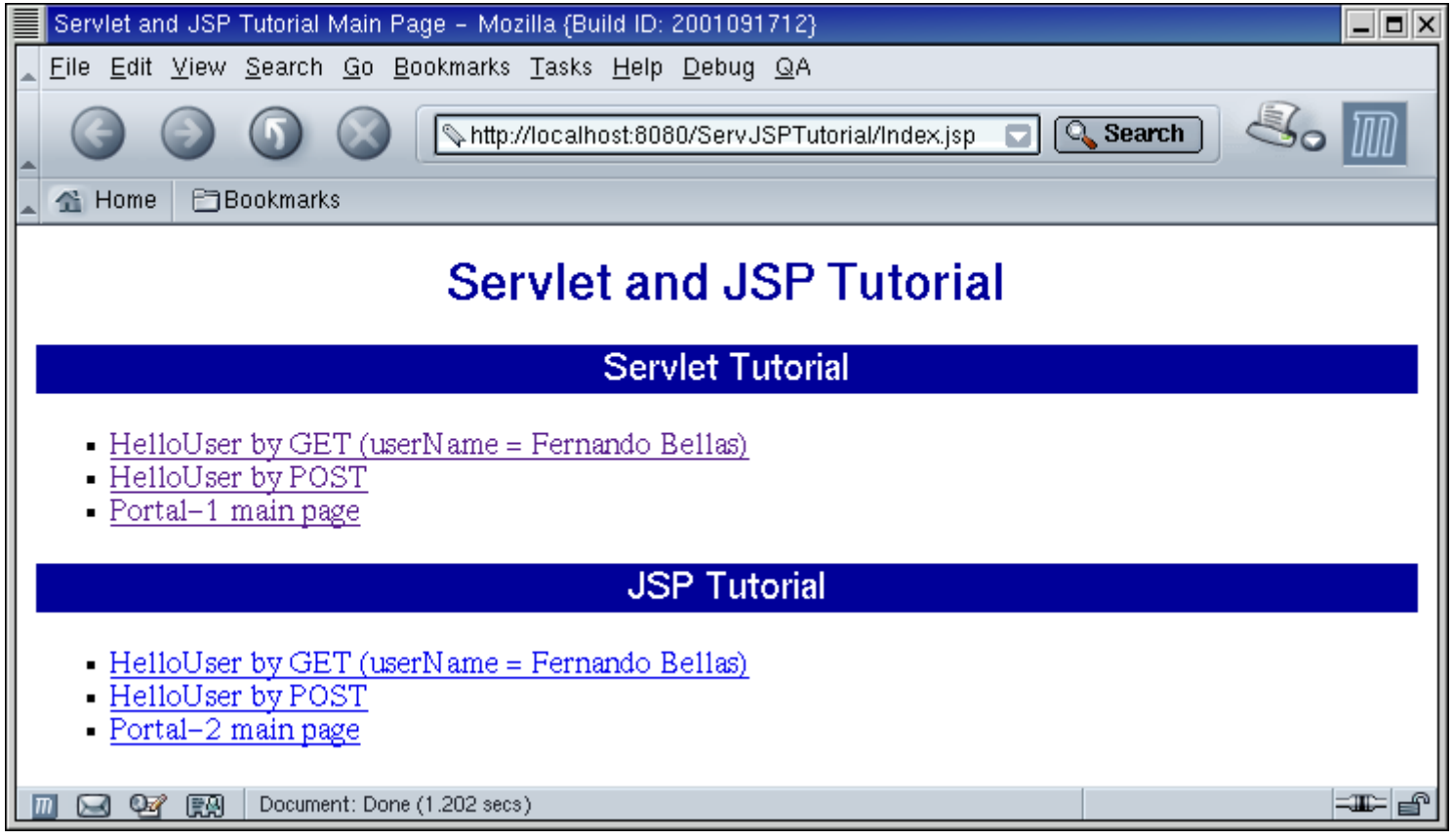
- En realidad, una página JSP es un tipo especial de servlet (**`javax.servlet.jsp`** y **`javax.servlet.jsp.tagext`**) orientado a generar el texto de la interfaz gráfica
  - Invocables por GET y POST



¿ Qué es JSP ? (y 3)

- ¿ Qué ocurre cuando se accede a una página JSP ?
  - Si es la primera vez, el servidor de aplicaciones genera un servlet (que implementa **`javax.servlet.jsp.HttpJspPage`**) a partir de la página JSP, lo compila y lo carga en memoria
  - Si no es la primera vez, le pasa la petición al servlet (ya compilado y creado en memoria)
  - Si la página se ha modificado desde la última compilación, el servidor se da cuenta, genera el nuevo servlet, lo compila y lo carga de nuevo

Página principal del tutorial



Index.jsp

```
<html>

...

<ul>
<li><a href="Hello2/HelloUser.jsp?userName=Fernando+Bellas">
HelloUser by GET (userName = Fernando Bellas)</a></li>
<li><a href="Hello2/HelloUserByPost.html">HelloUser by POST</a></li>
<li><a href="<%= response.encodeURL("Portal2/MainPage.jsp") %>">
Portal-2 main page</a></li>
</ul>

...

</html>
```

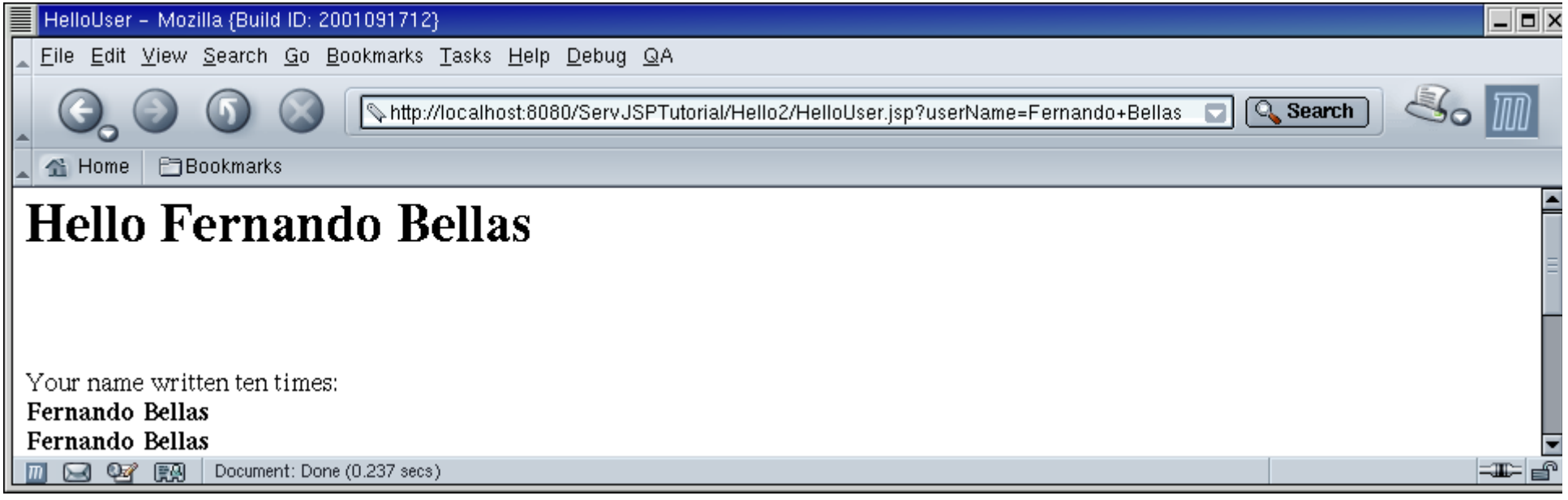
Portal2/HelloUserByPost.html

```
<html>
...
<form method="POST" action="HelloUser.jsp">
<table width="100%" border="0" align="center" cellspacing="12">

    <tr>
        <th align="right" width="50%">User name</th>
        <td align="left">
            <input type="text" name="userName" size="16“
                   maxlength="16">
        </td>
    </tr>

    <tr>
        <td width="50%“></td>
        <td align="left" width="50%">
            <input type="submit" value="Say me hello">
        </td>
    </tr>
</table>
</form>
...
</html>
```

Demo HelloUser (1)

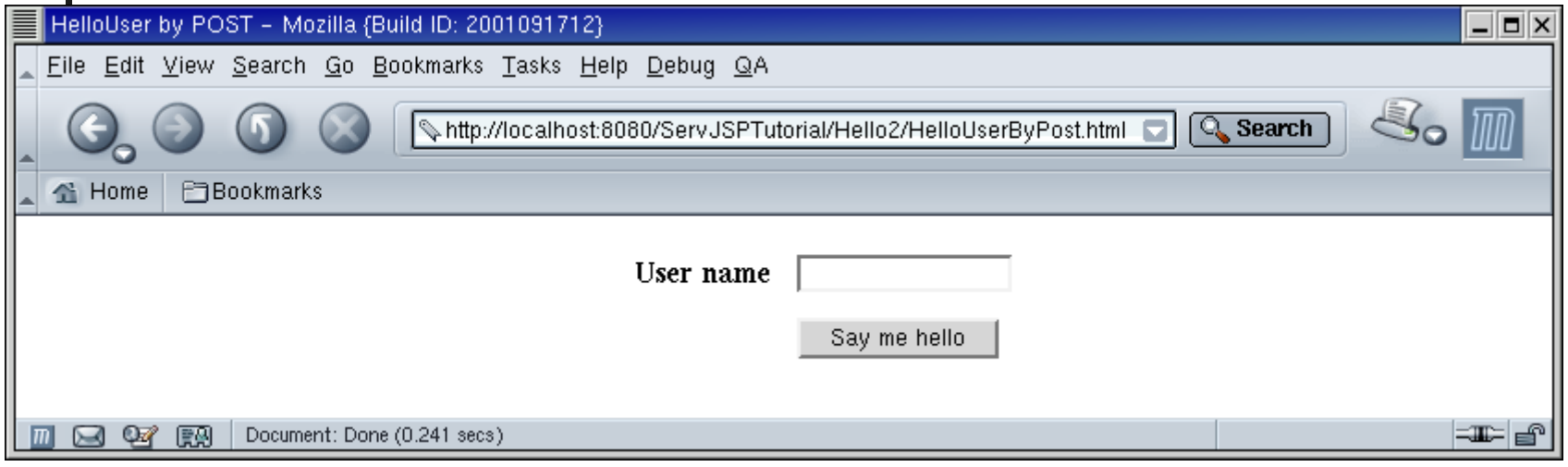
Servlet and JSP Tutorial Main Page

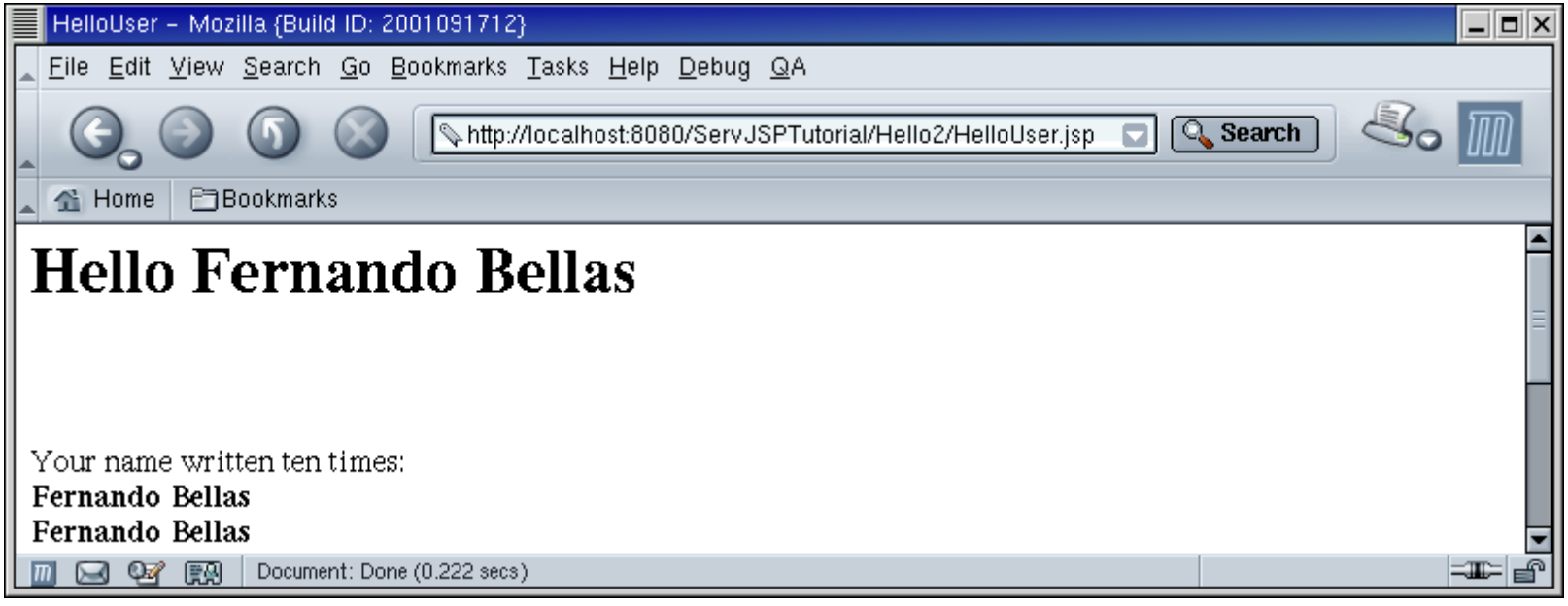
Clic en HelloUser by GET (userName = Fernando Bellas)



Demo HelloUser (y 2)

Servlet and JSP Tutorial Main Page — Clic en HelloUser by POST

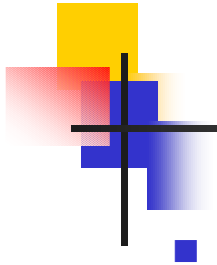




Hello2/HelloUser.jsp

```
<html>
<head>
<title>HelloUser</title>
</head>

<body text="#000000" bgcolor="#ffffff">
<h1>Hello <%= request.getParameter("userName") %> </h1>

<br><br><br>

Your name written ten times:<br>

<%
    String name = request.getParameter("userName");

    for (int i=0; i<10; i++) {
%>
        <b><%= name %></b> <br>
<%
    }
%>

</body>
</html>
```

Comentarios

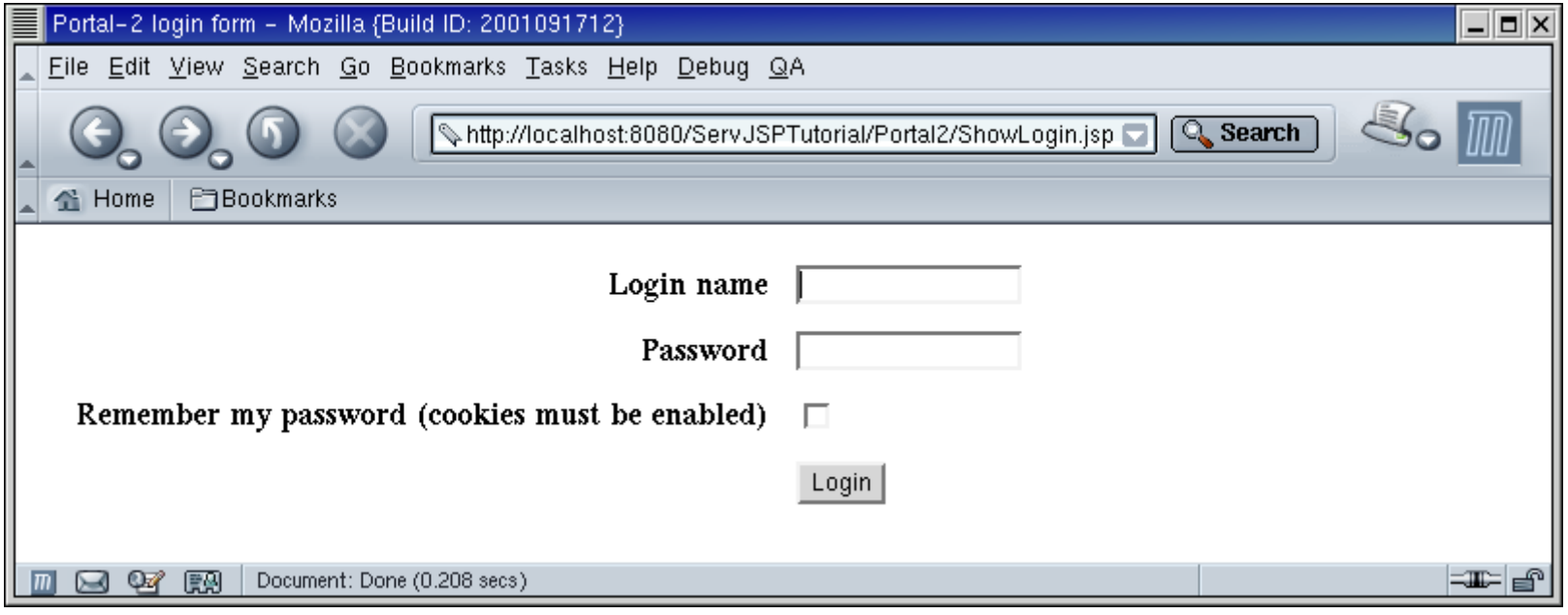


- Usa **<%= expresión %>** para incluir expresiones Java
  - La expresión es evaluada en tiempo de ejecución y convertida a un **String**
- Usa scriptlets para incluir código Java
  - **<% ... %>**
- Objetos implícitos
  - **request**: **javax.servlet.http.HttpServletRequest**
  - **response**: **javax.servlet.http.HttpServletResponse**
  - **session**: **javax.servlet.http.HttpSession**
  - **out**: **javax.servlet.jsp.JspWriter**
  - Algunos más
    - Los veremos a medida nos hagan falta

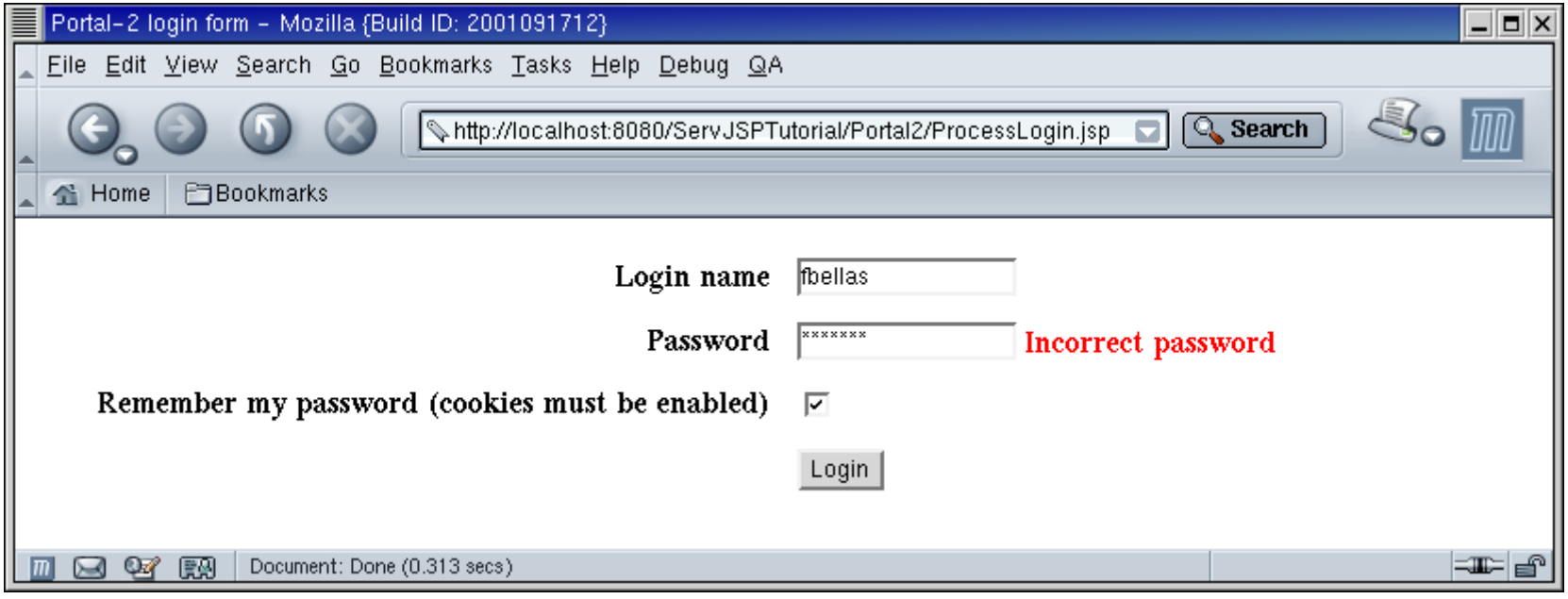
Demo Portal-2 (1)

Servlet and JSP Tutorial Main Page

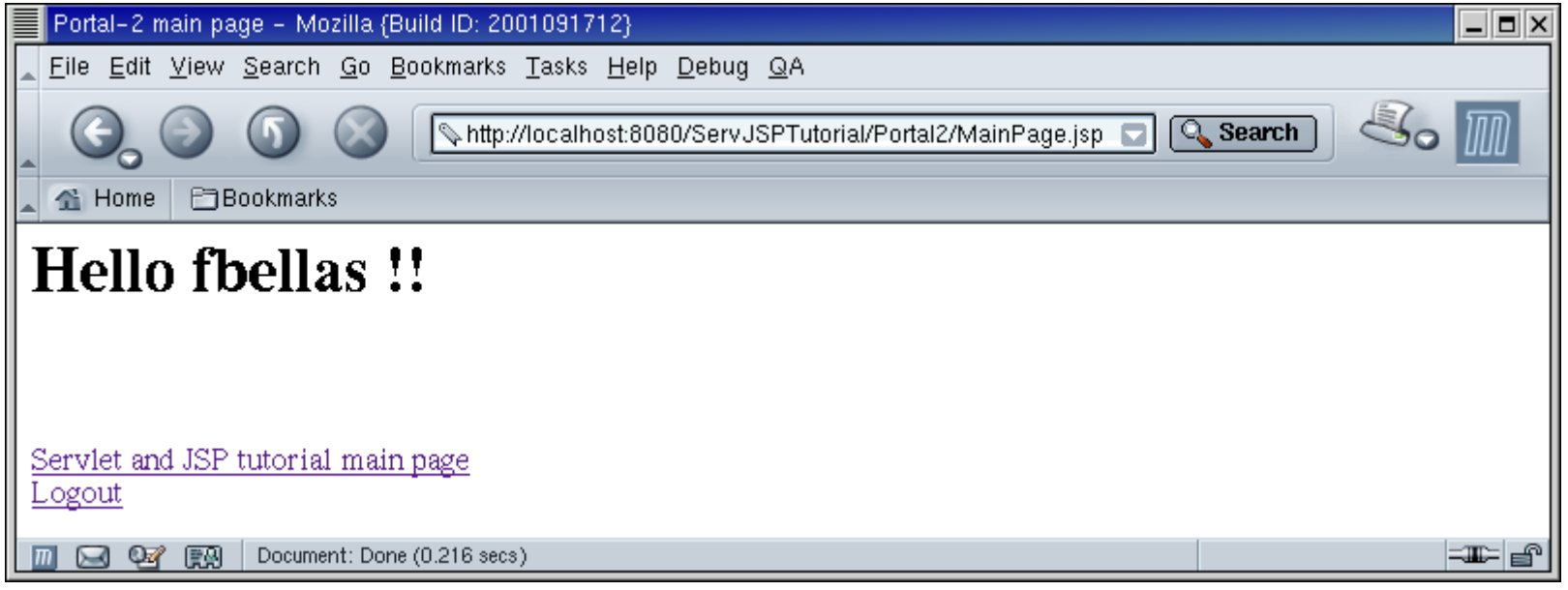
Clic en Portal-2 main page



Demo Portal-2 (2)



Demo Portal-2 (y 3)



Clic en Servlet and JSP tutorial main page

Servlet and JSP Tutorial Main Page

Terminar y lanzar el navegador dos días más tarde

Acceder a Servlet and JSP tutorial main page

Servlet and JSP Tutorial Main Page

Clic en Portal-2 main page

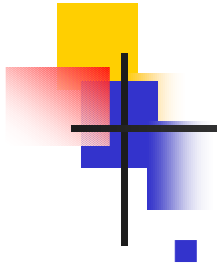
Clic en Logout

Servlet and JSP Tutorial Main Page

Clic en Portal-2 main page

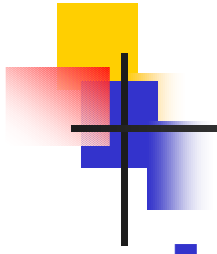
Portal-2 login form

Organización del código



- Páginas JSP en **`WEB-INF/Portal-2`**
  - **`MainPage.jsp`**
  - **`ShowLogin.jsp`**
  - **`ProcessLogin.jsp`**
  - **`ProcessLogout.jsp`**
- En **`es.udc.fbellas.j2ee.servjsptutorial.portal2`**
  - **`LoginManager`**
  - **`LoginForm`**

Concepto de cookie (1)



- El API de servlets permite enviar cookies al navegador del usuario
  - Una cookie tiene un nombre y un valor asociado (cadena de caracteres)
    - `Cookie cookie = new Cookie("loginName", loginName);`
  - Cada navegador debería soportar alrededor de 20 cookies por cada sitio web al que está conectado, 300 en total y puede limitar el tamaño de cada cookie a 4 Kbytes
- Para enviar una o varias cookies al navegador se incluyen en la **`response`**
  - `response.addCookie(cookie);`
- Cada vez que el navegador hace una petición, todas las cookies relativas a esa aplicación web llegan en la **`request`**
  - `Cookie[] cookies = request.getCookies();`

Concepto de cookie (2)

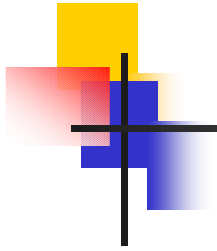
- Tiempo de vida de una cookie
  - **`cookie.setMaxAge(seconds);`**
  - **`seconds > 0`** => la cookie se almacenará persistentemente en el navegador durante ese número de segundos
    - Los navegadores suelen almacenar este tipo de cookies en ficheros locales
  - **`seconds == 0`** => eliminar la cookie
  - **`seconds < 0`** => la cookie no se almacenará persistentemente y dejará de existir cuando el navegador termine su ejecución
    - Los navegadores mantienen este tipo de cookies en memoria

Concepto de cookie (3)

- En el ejemplo
  - Gestión de la sesión: idem ejemplo anterior
  - Se valida nombre de login y password de forma ficticia
    - La password es válida si es igual al nombre de login
  - Cada vez que un usuario hace login correctamente, se le crea una sesión, y si ha seleccionado “recordar mi password”, se le envían a su navegador dos cookies: **`loginName`** y **`password`**, con tiempo de vida = 30 días
  - Cada vez que un usuario accede a la página principal del portal, si su sesión no existe o no tiene el **`loginName`** en la sesión, se comprueba si en la **`request`** llegan las cookies **`loginName`** y **`password`**, y que son correctas
    - En caso afirmativo, se le crea una sesión y se le redirige a la página principal
    - En caso negativo, se le redirige a la página de login
  - Cada vez que el usuario hace un logout, se le destruye su sesión, se eliman las cookies de su navegador y se le redirige a la página principal del tutorial

Concepto de cookie (y 4)

- Seguridad
  - En un ejemplo real, el valor de la cookie **`password`** sería la password cifrada
    - MiniPortal lo hará así
  - Aún así, las cookies no son un mecanismo seguro para el usuario
    - Si un usuario tiene acceso al fichero de cookies de otra persona, puede copiar las cookies **`loginName`** y **`password`** a su fichero de cookies, y por tanto, entrar en la aplicación web con su identidad
    - Los sitios web que tienen la opción de “recordar mi password” suelen advertir del problema

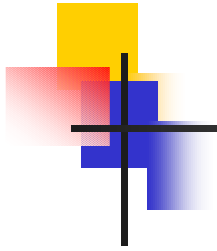


es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginManager (1)

```
public final class LoginManager {

    private final static String LOGIN_NAME_SESSION_ATTRIBUTE =
        "loginName";

    private static final String LOGIN_NAME_COOKIE = "loginName";
    private static final String PASSWORD_COOKIE = "password";

    private static final int
        COOKIES_TIME_TO_LIVE_REMEMBER_MY_PASSWORD =
            30 * 24 * 3600; // 30 days
    private static final int COOKIES_TIME_TO_LIVE_REMOVE = 0;


    private LoginManager() {}
```

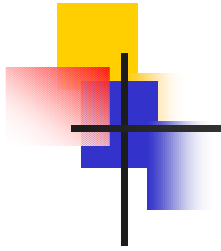
es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginManager (2)

```
public final static void login(HttpServletRequest request,
        HttpServletResponse response, String loginName,
        String password, boolean rememberMyPassword)
        throws IncorrectPasswordException {

        validateLogin(loginName, password);

        HttpSession session = request.getSession(true);
        session.setAttribute(LOGIN_NAME_SESSION_ATTRIBUTE, loginName);

        if (rememberMyPassword) {
            leaveCookies(response, loginName, password,
                COOKIES_TIME_TO_LIVE_REMEMBER_MY_PASSWORD);
        }


    }
```



es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginManager (3)

```
public final static void logout(HttpServletRequest request,
    HttpServletResponse response) {

    HttpSession session = request.getSession(false);

    if (session != null) {
        session.invalidate();
    }

    leaveCookies(response, "", "", COOKIES_TIME_TO_LIVE_REMOVE);

}
```



es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginManager (4)

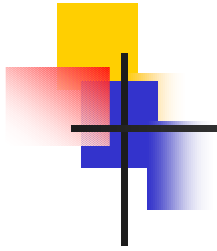
```
public final static String getLoginName(
    HttpServletRequest request) {

    /* Try to get login name from session. */
    String loginName = null;
    HttpSession session = request.getSession(false);

    if (session != null) {

        loginName = (String) session.getAttribute(
            LOGIN_NAME_SESSION_ATTRIBUTE);

        if (loginName != null) {
            return loginName;
        }

    }
```

es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginManager (5)

```
        /*
         * The user had not logged in or his/her session has expired.
         * We need to check if the user has selected "remember my
         * password" in the last login (login name and password will
         * come as cookies). If so, we save login name in the session.
         */
        loginName = getLoginNameFromCookies(request);

        if (loginName != null) {
            session = request.getSession(true);
            session.setAttribute(LOGIN_NAME_SESSION_ATTRIBUTE,
                loginName);
        }

        return loginName;

    }
```

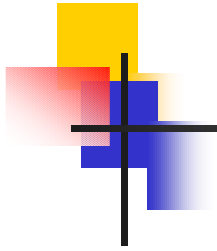
es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginManager (6)

```
private final static void validateLogin(String loginName,
    String password) throws IncorrectPasswordException {
    if (!loginName.equals(password)) {
        throw new IncorrectPasswordException();
    }
}

private final static void leaveCookies(
    HttpServletResponse response, String loginName,
    String password, int timeToLive) {

    /* Create cookies. */
    Cookie loginNameCookie = new Cookie(LOGIN_NAME_COOKIE,
        loginName);
    Cookie passwordCookie = new Cookie(PASSWORD_COOKIE, password);

    /* Set maximum age to cookies. */
    loginNameCookie.setMaxAge(timeToLive);
    passwordCookie.setMaxAge(timeToLive);

    /* Add cookies to response. */
    response.addCookie(loginNameCookie);
    response.addCookie(passwordCookie);

}
```



es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginManager (7)

```
private final static String getLoginNameFromCookies(
    HttpServletRequest request) {

    /* Are there cookies in the request ? */
    Cookie[] cookies = request.getCookies();
    if (cookies == null) {
        return null;
    }

    /* Check if login name and password come as cookies. */
    String loginName = null;
    String password = null;
    int foundCookies = 0;
```



es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginManager (8)

```
for (int i=0; (i<cookies.length) && (foundCookies < 2); i++) {
    if (cookies[i].getName().equals(LOGIN_NAME_COOKIE)) {
        loginName = cookies[i].getValue();
        foundCookies++;
    }
    if (cookies[i].getName().equals(PASSWORD_COOKIE)) {
        password = cookies[i].getValue();
        foundCookies++;
    }
}

if (foundCookies != 2) {
    return null;
}
```

es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginManager (y 9)

```
        /* Validate loginName and password. */
        try {
            validateLogin(loginName, password);
            return loginName;
        } catch (IncorrectPasswordException e) {
            return null;
        }

    }

}
```

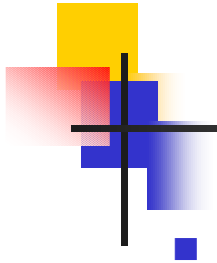
Portal2/MainPage.jsp

```
<%@ page
    import="es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginManager" %>

<html>
<head>
<title>Portal-2 main page</title>
</head>

<body text="#000000" bgcolor="#ffffff" link="#000ee0" vlink="#551a8b"
      alink="#000ee0">

<%
    String loginName = LoginManager.getLoginName(request);

    if (loginName == null) { // User has not logged in yet.
        response.sendRedirect(response.encodeRedirectURL(
            "ShowLogin.jsp"));
    } else {
%>
```

Portal2/MainPage.jsp (y 2)

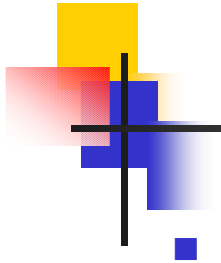
```
<%-- User has logged in. --%>

<h1>Hello <%=loginName%> !!</h1>
<br>
<br>
<br>
<a href="<%= response.encodeURL("../Index.jsp") %>">
Servlet and JSP tutorial main page</a>
<br>
<a href="<%= response.encodeURL("ProcessLogout.jsp") %>">Logout</a>

<%  } %>

</body>

</html>
```

Comentarios



- **<%@ ... %>**
  - Directiva
  - En particular, la directiva **page** tiene varios atributos, entre otros
    - **import** (opcional): permite importar una o más clases (separadas por comas)
    - **session** (opcional): por defecto su valor es **true** => se crea una nueva sesión para el usuario si ésta no existía
- **<%-- ... --%>**
  - Comentarios
  - No aparecen en la respuesta generada, a diferencia de los comentarios HTML

JavaBeans (1)



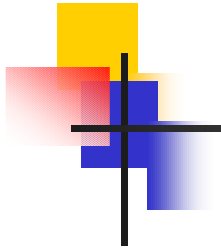
- Un JavaBean es una clase Java que sigue un conjunto de guías de estilo para permitir la creación de componentes Java reusables
  - El campo en el que más éxito han tenido ha sido en la creación de componentes reusables AWT/Swing
  - JSP soporta este concepto, pero sólo es preciso tener en cuenta un subconjunto muy pequeño de las guías de estilo
  - Muchas veces se abusa de esta palabra, y se aplica a cualquier clase Java
  - Otras veces, mucha gente confunde “JavaBean” con “Enterprise JavaBean (EJB)”
- JSP soporta el uso de JavaBeans como mecanismo de acceder a objetos enganchados a la **`request`** y la sesión (entre otros)
  - Pretende ocultar esa parte del API de los servlets desde las páginas JSP, y reducir así el código Java que hay en ellas

JavaBeans (y 2)

- En el contexto de JSP, un JavaBean será una clase serializable que obedece las siguientes guías de estilo
  - Constructor público sin argumentos
  - Dispone de propiedades que se pueden consultar con métodos **`getXXX`** y/o modificar con métodos **`setXXX`**
    - Las propiedades pueden ser univaluadas
      - **`public TipoPropiedad getNombrePropiedad()`**
      - **`public void setNombrePropiedad(TipoPropiedad propiedad)`**
    - O multivaluadas
      - **`public TipoPropiedad[] getNombrePropiedad()`**
      - **`public void setNombrePropiedad(TipoPropiedad[] propiedad)`**
    - **`TipoPropiedad`**: tipo básico (o su equivalente objetual), **`String`** o tipo para el que exista un **`java.beans.PropertyEditor`**
      - Normalmente trabajaremos con **`String`**
  - Ej.: **`es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginForm`**

es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginForm (1)

```
public class LoginForm implements Serializable {

    private String loginName;
    private String password;
    private String rememberMyPassword;

    public LoginForm() {
        loginName = "";
        password = "";
        rememberMyPassword = null;
    }

    public String getLoginName() {
        return loginName;
    }

    public void setLoginName(String loginName) {
        this.loginName = loginName.trim();
    }
```



es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginForm (y 2)

```
    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getRememberMyPassword() {
        return rememberMyPassword;
    }

    public void setRememberMyPassword(String rememberMyPassword) {
        this.rememberMyPassword = rememberMyPassword;
    }

}
```

Uso de JavaBeans en el ejemplo (1)

- Acciones estándar
  - Tags XML **`jsp:XXX`** (espacio de nombres **`jsp`**)
  - JSP define un pequeño número de acciones estándar
- Acción **`jsp:useBean`**

```
<jsp:useBean
    id="loginForm" scope="request"
    class="es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginForm"/>
```

  - En el servlet generado
    - Se declara la variable **`loginForm`**
    - Se examina si existe un atributo en el ámbito especificado con ese nombre, y si no existe se crea
      - **`request.setAttribute(“loginForm”, new LoginForm());`**
    - Se asigna el atributo a la variable
      - **`loginForm = (LoginForm) request.getAttribute("loginForm");`**

Uso de JavaBeans en el ejemplo (2)

- Acción **`jsp:useBean`** (cont)
  - Ámbito
    - **`page`**
      - Bean sólo visible en esta página (se destruye al final de cada ejecución)
      - Bean enganchado como atributo en **`javax.servlet.jsp.PageContext`**
    - **`request`**
      - Bean enganchado como atributo a **`javax.servlet.ServletRequest`**
    - **`session`**
      - Bean enganchado como atributo a **`javax.servlet.http.HttpSession`**
    - **`application`**
      - Visible a toda la aplicación web
      - Bean enganchado como atributo a **`javax.servlet.ServletContext`**

Uso de JavaBeans en el ejemplo (y 3)

- Acción **`jsp:setProperty`**

```
<jsp:setProperty name="loginForm" property="*"/>
```

  - La usaremos en **`ProcessLogin.jsp`** para dar automáticamente valor a las propiedades de **`loginForm`**
  - Los parámetros tienen que llamarse **`loginName`** y **`password`**
    - Como en los métodos **`getXXX`** y **`setXXX`** pero la primera letra en minúscula
  - En el servlet generado se llama a los métodos **`setXXX`**
- Acción **`jsp:getProperty`**

```
<jsp:getProperty name="loginForm" property="password"/>
```

  - Permite recuperar el valor de una propiedad
  - La usaremos en **`ShowLogin.jsp`** para recuperar el valor de una propiedad
  - En el servlet generado se llama al método **`getXXX`**

Portal2/ShowLogin.jsp (1)

```
<%@ page import="java.util.Map, java.util.HashMap,
                 es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginForm" %>

<jsp:useBean
    id="loginForm" scope="request"
    class="es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginForm"/>

<html>
<head>
<title>Portal-2 login form</title>
</head>

<body text="#000000" bgcolor="#ffffff">
```

Portal2/ShowLogin.jsp (2)

```
<%-- Get errors. --%>
<%
    String loginNameErrorMessage = "";
    String passwordErrorMessage = "";
    Map errors = (Map) request.getAttribute("errors");

    if (errors != null) {

        String errorHeader = "<font color=\"red\"><b>";
        String errorFooter = "</b></font>";

        if (errors.containsKey("loginName")) {
            loginNameErrorMessage = errorHeader +
                errors.get("loginName") +  errorFooter;
        }

        if (errors.containsKey("password")) {
            passwordErrorMessage = errorHeader +
                errors.get("password") + errorFooter;
        }

    }
%>
```

Portal2/ShowLogin.jsp (3)

```
<form method="POST" action="<%= response.encodeURL("ProcessLogin.jsp")
    %>">

<table width="100%" border="0" align="center" cellspacing="12">

<%-- Login name --%>

    <tr>
        <th align="right" width="50%">
            Login name
        </th>
        <td align="left">
            <input type="text" name="loginName"
                value="<jsp:getProperty name="loginForm"
                    property="loginName"/>"
                size="16" maxlength="16">
            <%= loginNameErrorMessage %>
        </td>
    </tr>
```

Portal2/ShowLogin.jsp (4)

```
<%-- Password --%>

    <tr>
        <th align="right" width="50%">
            Password
        </th>
        <td align="left">
            <input type="password" name="password"
                value="<jsp:getProperty name="loginForm"
                    property="password"/>"
                size="16" maxlength="16">
            <%= passwordErrorMessage %>
        </td>
    </tr>
```

Portal2/ShowLogin.jsp (5)

```
<%-- Remember my password --%>

    <tr>
        <th align="right" width="50%">
            Remember my password (cookies must be enabled)
        </th>
        <td align="left">
            <input type="checkbox" name="rememberMyPassword"
                value="rememberMyPassword"

                <% if (loginForm.getRememberMyPassword() != null) { %>
                    checked
                <%  } %>

                >
        </td>
    </tr>
```

Portal2/ShowLogin.jsp (y 6)

```
<%-- Login button --%>

    <tr>
        <td width="50%"></td>
        <td align="left" width="50%">
            <input type="submit" value="Login">
        </td>
    </tr>

</table>

</form>

</body>

</html>
```

Portal2/ProcessLogin.jsp (1)

```
<%@ page import="java.util.Map, java.util.HashMap,
   es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginForm,
   es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginManager,
   es.udc.fbellas.j2ee.servjsptutorial.portal2.IncorrectPasswordException"
   %>

<jsp:useBean id="loginForm" scope="request“
    class="es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginForm"/>

<jsp:setProperty name="loginForm" property="*"/>
```

Portal2/ProcessLogin.jsp (2)

```
<%-- Mandatory parameters are present ? --%>

<%
    String loginName = loginForm.getLoginName();
    String password = loginForm.getPassword();
    Map errors = new HashMap();

    if ( (loginName == null) || (loginName.length() == 0) ) {
        errors.put("loginName", "Mandatory field");
    }

    if ( (password == null) || (password.length() == 0) ) {
        errors.put("password", "Mandatory field");
    }

    if (!errors.isEmpty()) {
        request.setAttribute("errors", errors);
%>
        <jsp:forward page="ShowLogin.jsp" />
<%
    }
%>
```

Portal2/ProcessLogin.jsp (y 3)

```
<%-- Try to login. --%>

<%
    boolean rememberMyPassword =
        loginForm.getRememberMyPassword() != null;

    try {
        LoginManager.login(request, response, loginName, password,
            rememberMyPassword);
    } catch (IncorrectPasswordException e) {
        errors.put("password", "Incorrect password");
    }

    if (errors.isEmpty()) {
        response.sendRedirect(response.encodeRedirectURL(
            "MainPage.jsp"));
    } else {
        request.setAttribute("errors", errors);
%>
        <jsp:forward page="ShowLogin.jsp" />
<%
    }
%>
```

Portal2/ProcessLogout.jsp

```
<%@ page
   import="es.udc.fbellas.j2ee.servjsptutorial.portal2.LoginManager" %>

<%
    LoginManager.logout(request, response);
    response.sendRedirect(response.encodeRedirectURL("../Index.jsp"));
%>
```

Comentarios (1)

- **`jsp:forward`**
  - Acción equivalente a los forwards que se han hecho en el ejemplo del anterior
    - **`RequestDispatcher.forward(ServletRequest, ServletResponse)`**
  - El atributo **`page`** acepta el mismo tipo de URLs
- Otros directivas y acciones útiles
  - Directiva **`include`**

    `<%@ include file="DefaultPageFooter.jsp" %>`
    - Se resuelve en tiempo de traducción (antes de compilar)
    - Similar a la directiva **`include`** del preprocesador de C/C++
    - Puede incluir cualquier fichero de texto
    - Buena para evitar copy-n-paste de secciones comunes en páginas JSP
    - El atributo **`file`** acepta el mismo tipo de URLs que **`page`** en **`jsp:forward`**

Comentarios (2)

- Otros directivas y acciones útiles (cont)
  - Acción **`jsp:include`**

```
<jsp:include page="forecast">
    <jsp:param name="city" value="COR"/>
    <jsp:param name="state" value="GAL"/>
</jsp:include>
```

    - Permite incluir la respuesta de la invocación de una URL en tiempo de ejecución
    - El atributo **`page`** acepta mismo el tipo de URLs que **`file`** en la directiva **`include`**
    - Equivalente a **`RequestDispatcher.include(ServletRequest, ServletResponse)`**

Comentarios (3)

- URL rewriting
  - Las URLs que ve el navegador (los forwards no cuentan) se han generado de manera especial
  - En los tags **`<a href ... >`** generados

```
<li><a href="<%=
response.encodeURL("Portal2/MainPage.jsp") %>">
Portal-2 main page</a></li>
```

  - Y en **`sendRedirect`**

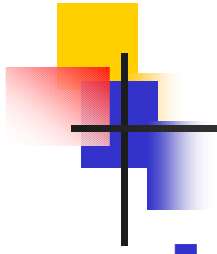
```
response.sendRedirect(
    response.encodeRedirectURL("../Index.jsp"));
```

  - Para entender el motivo es preciso comprender cómo hace el servidor de aplicaciones web para saber a qué sesión corresponde cada petición HTTP que recibe

Comentarios (4)

- URL rewriting (cont)
  - Si el navegador acepta cookies, cuando se crea una sesión, el servidor de aplicaciones web le envía al navegador la cookie **`jsessionid`**
  - Si el navegador no acepta cookies, y se quiere hacer uso de sessiones, es preciso que el programador codifique todas las URLs para que lleven incrustado **`jsessionid`**

    `../Index.jsp;jsessionid=9fab993aca36c8a3af423ba`
    - Cuando el servidor recibe una petición HTTP parsea la URL y utiliza el valor de **`jsessionid`** para asociar la petición con la sesión
  - Conclusión: para que una aplicación que usa sesiones funcione tanto si el usuario acepta cookies como si no, debe aplicar URL rewriting a todas las URLs que ve el navegador
    - Si el navegador acepta cookies, **`response.encodeURL`** y **`response.encodeRedirectURL`** no modifican la URL



Comentarios (y 5)

- Sesiones y múltiples ventanas
  - Normalmente todas las ventanas de un navegador forman parte del mismo proceso
  - Existen navegadores que se pueden configurar para que cada ventana corra en su propio proceso => desde cada ventana se provocará la creación de una nueva sesión
  - Cuando se usa URL rewriting, si el usuario crea un nueva ventana y sobre ella teclea una URL (que casi con seguridad no incluirá la especificación de **`jsessionid`**) de una aplicación web (que utiliza sesiones) que está siendo visualizada en otra ventana => se creará otra sesión en el servidor