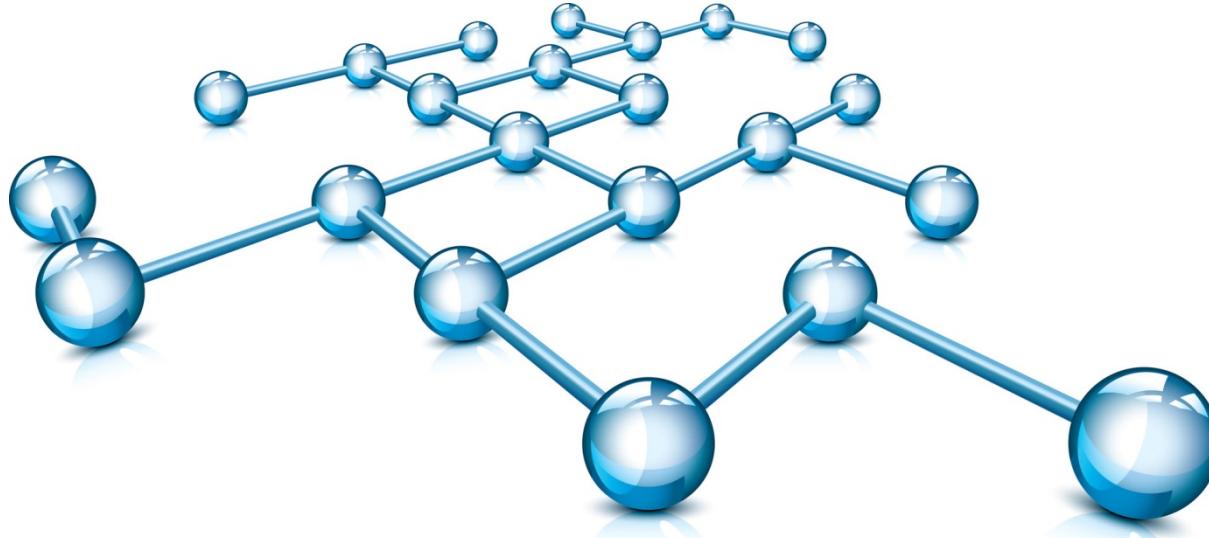


# Componentes Web de Java EE

Profesor:  
Dr. J. Octavio Gutiérrez García

[octavio.gutierrez@itam.mx](mailto:octavio.gutierrez@itam.mx)

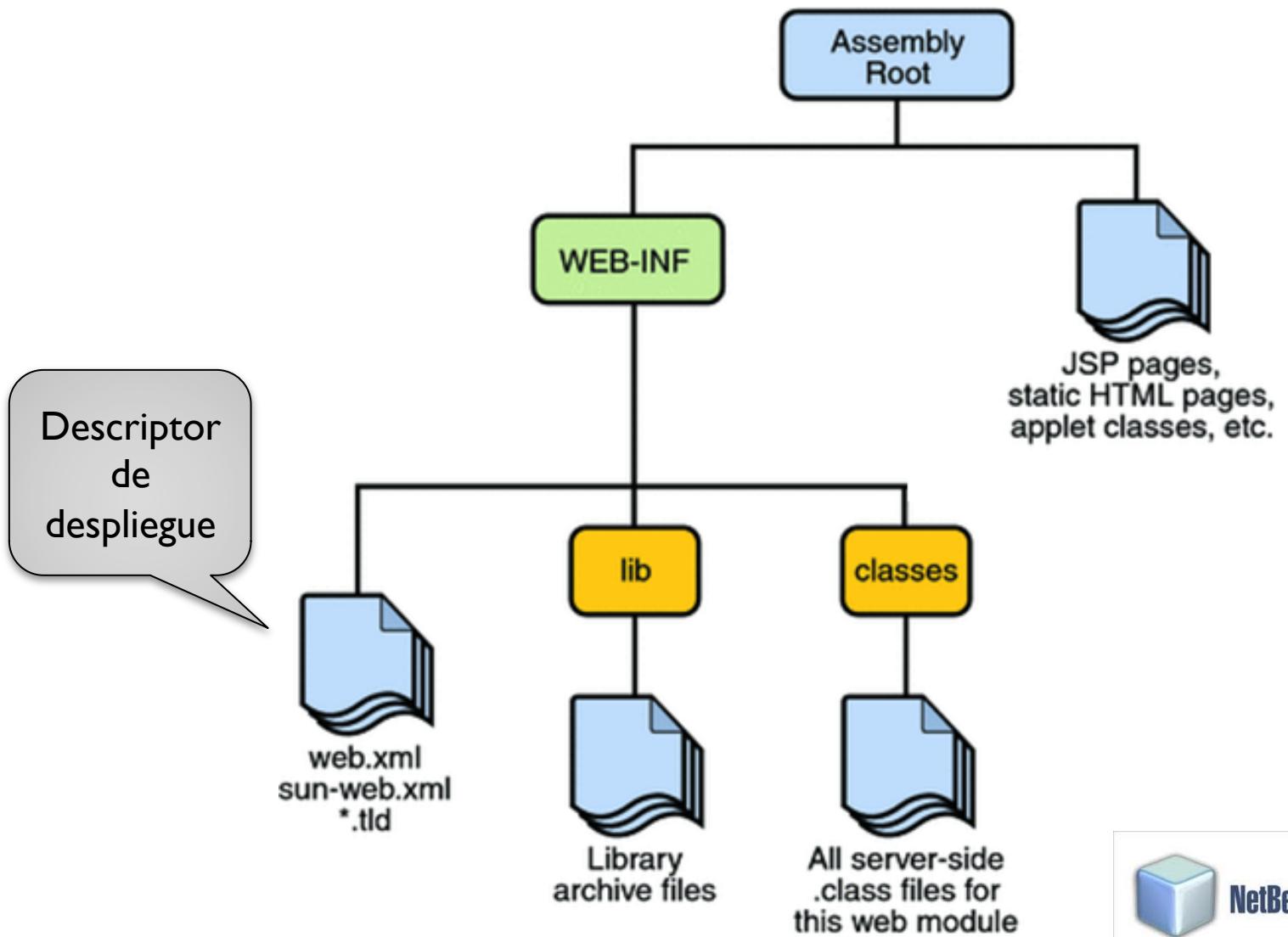


# Antes de empezar...

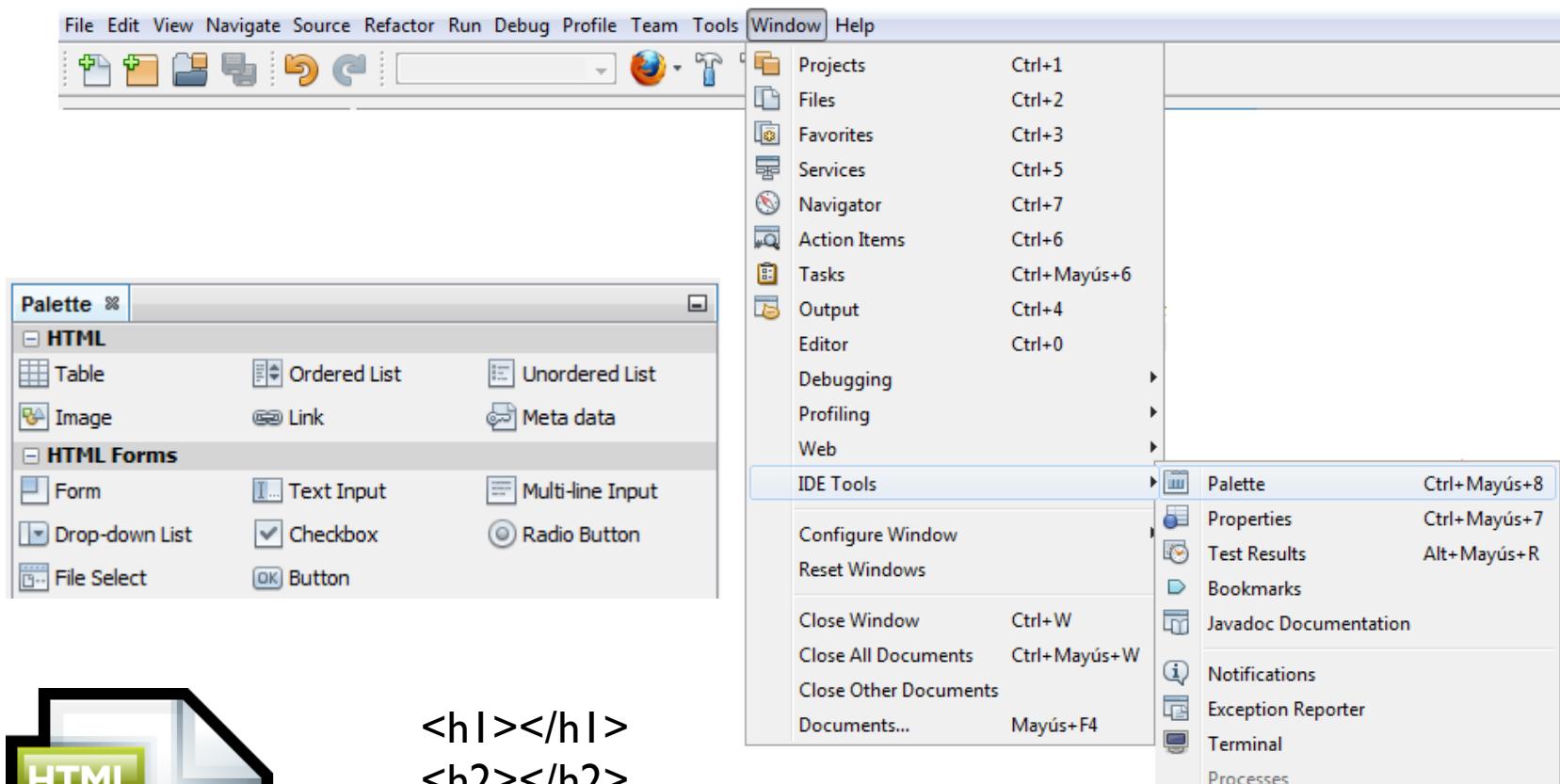
- ¿Cómo crear aplicaciones web en Netbeans?
- ¿Cómo compilar y desplegar una aplicación web?



# Estructura de un módulo web



# Jugando con HTML en Netbeans



The screenshot shows the Netbeans IDE interface. The window title is "Jugando con HTML en Netbeans". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The "Window" menu is open, displaying a list of windows and their keyboard shortcuts. The "Palette" window is currently active, showing HTML and HTML Forms components. Other windows listed include Projects, Files, Favorites, Services, Navigator, Action Items, Tasks, Output, Editor, Debugging, Profiling, Web, IDE Tools, Configure Window, Reset Windows, Close Window, Close All Documents, Close Other Documents, and Documents... The "IDE Tools" section also lists Palette, Properties, Test Results, Bookmarks, Javadoc Documentation, Notifications, Exception Reporter, Terminal, and Processes.

Window	Keyboard Shortcut
Projects	Ctrl+1
Files	Ctrl+2
Favorites	Ctrl+3
Services	Ctrl+5
Navigator	Ctrl+7
Action Items	Ctrl+6
Tasks	Ctrl+Mayús+6
Output	Ctrl+4
Editor	Ctrl+0
Debugging	
Profiling	
Web	
IDE Tools	
Configure Window	
Reset Windows	
Close Window	Ctrl+W
Close All Documents	Ctrl+Mayús+W
Close Other Documents	
Documents...	Mayús+F4
Palette	Ctrl+Mayús+8
Properties	Ctrl+Mayús+7
Test Results	Alt+Mayús+R
Bookmarks	
Javadoc Documentation	
Notifications	
Exception Reporter	
Terminal	
Processes	



```
<h1></h1>
<h2></h2>
<h3></h3>
<h6></h6>
<p></p>
<br>
```

# Práctica de HTML

## ¡Mi primera forma!

Descripción	Valor
Apellido paterno:	<input type="text"/>
Apellido materno:	<input type="text"/>
Género	<input checked="" type="radio"/> Hombre <input type="radio"/> Mujer
Región del país	<input type="button" value="Norte"/>
¿Tiene todas las vacunas?	<input type="checkbox"/>
Comentarios adicionales	<input type="text"/>
Foto:	
Tres países donde me gustaría vivir	<ul style="list-style-type: none"><li>A. Uruguay</li><li>B. Turquía</li><li>C. Marruecos</li></ul>
Mi sitio web favorito es:	<a href="http://The Chronicle of Higher Education">The Chronicle of Higher Education</a>
<input type="button" value="Limpiar"/>	<input type="button" value="Enviar"/>



NetBeans

WebApplicationHTML

# Página de bienvenida

- Por defecto el servidor busca las siguientes páginas de bienvenida en el siguiente orden:

- **welcome-file-list** en web.xml
- index.html
- index.htm
- index.jsp

*Welcome!*

# Página de bienvenida

## **web.xml**

```
<web-app>
```

```
...
```

```
    <welcome-file-list>
```

```
        <welcome-file>mypage1.jsp</welcome-file>
```

```
        <welcome-file>mypage2.jsp</welcome-file>
```

```
....
```

```
    </welcome-file-list>
```

```
...
```

```
</web-app>
```

*Welcome!*

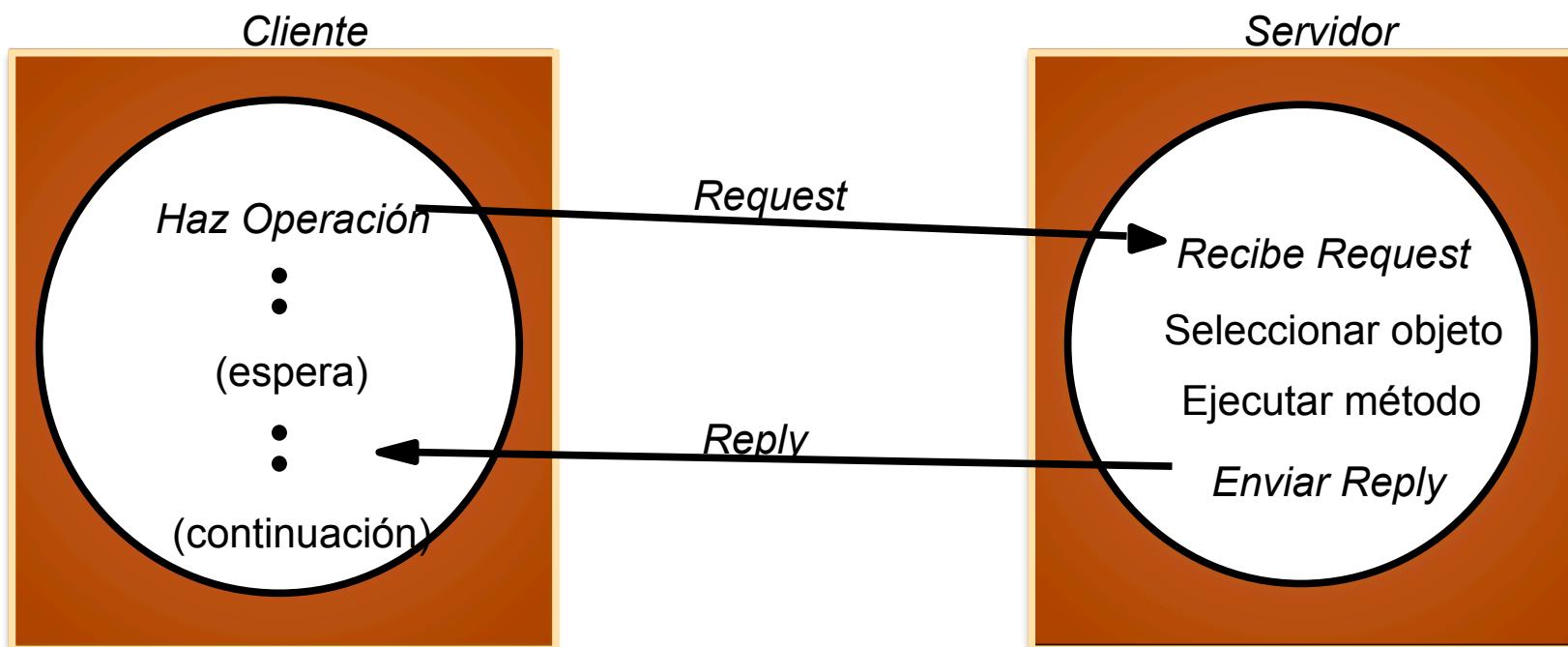
- Protocolos request-reply (*solicitud-respuesta*)



HyperText  
Transfer  
Protocol

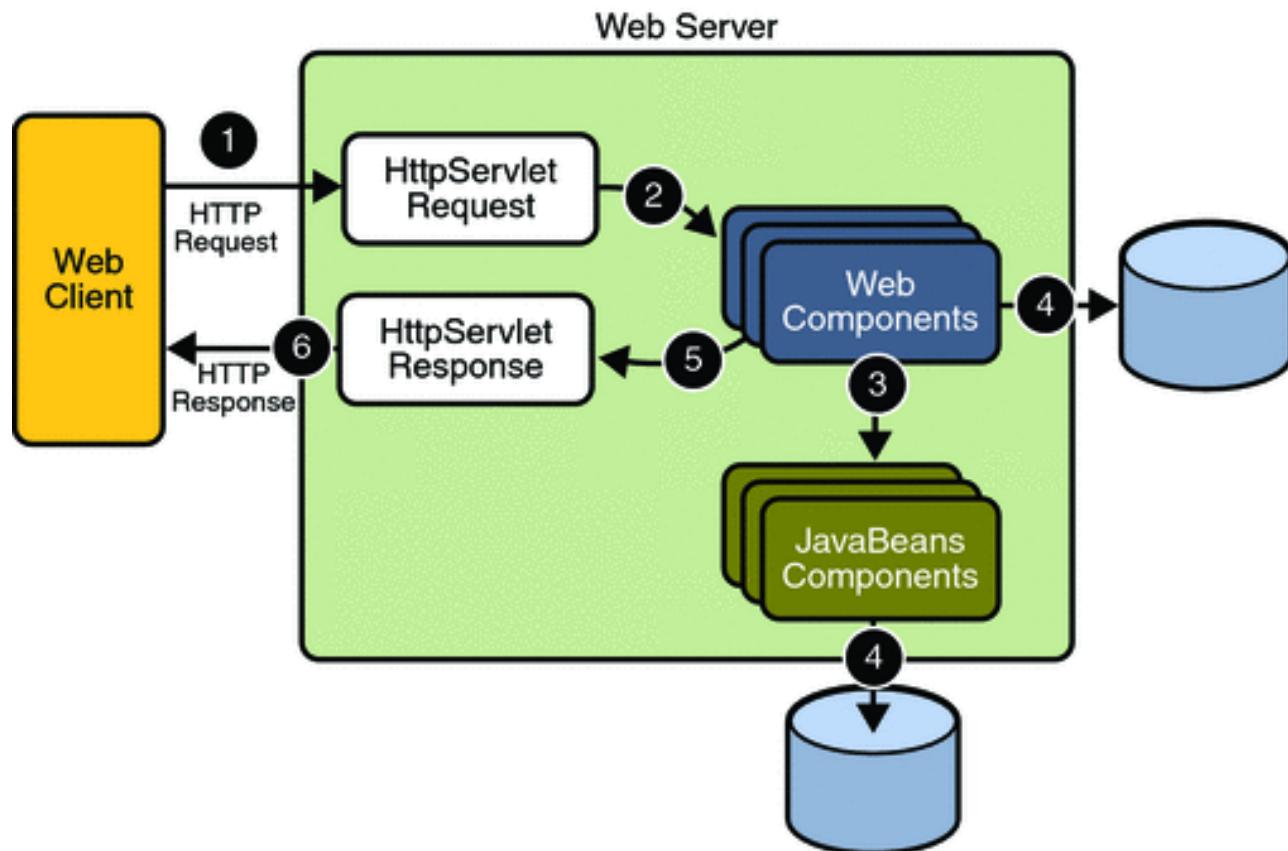
# Protocolos request-reply

Comunicación síncrona (normalmente)



# Aplicaciones Web

- Manejo de solicitudes



# Protocolo request-reply: HTTP

- Implementado sobre TCP
- Orientado a transacciones

Una transacción es un mensaje de petición y respuesta
- Protocolo sin estado

# Protocolo request-reply: HTTP

- Métodos
  - **GET**
    - Solicita una página web
  - **HEAD**
    - Solicita la cabecera de una página web
  - **POST**
    - Envía datos a una aplicación web
  - **PUT**
    - Solicita almacenar una página web
  - **DELETE**
    - Solicita borrar una página web
  - **TRACE**
    - Incluye datos de la petición en la respuesta (debugging)
  - **OPTIONS**
    - Devuelve los métodos soportados para una URL dada

# Protocolo request-reply: HTTP

<Método><URI solicitado><protocolo>

GET www.facebook.com?photo=octavio HTTP/1.1

<versión HTTP>< código estado >< razón >< encabezado >< contenido >

HTTP/1.1

200

OK

nulo



# Recibiendo parámetros con Scriptlets vía POST/GET

## Scriptlet

Expresiones de java embebidas en páginas JSP

<%

```
if(request.getParameter("nombre") != null){  
    out.println("<h1>Tu nombre es: " +  
                request.getParameter("nombre")+"</h1>");
```

}

%>



Post

VS

Get



# Práctica de laboratorio: request-reply

index.jsp



Seguro de Automóviles de Sistemas Distribuidos ...

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

Seguro de Automóviles de Sistemas Dist... +

localhost:8084/Wek

Sistema de Cotización de Seguros de Auto

Datos personales

Campo	Valor
Nombre:	Octavio
Apellidos:	Gutierrez
Género:	<input checked="" type="radio"/> Masculino <input type="radio"/> Femenino
Edad:	
Estado:	Jalisco
Limpiar	Enviar



WebApplicationCarInsurance

# Práctica de laboratorio: request-reply

auto.jsp



Datos de Auto - Mozilla Firefox

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

Datos de Auto

localhost:8084/Wek

Sistema de Cotización de Seguros de Auto

Datos personales:

Estimado Sr. Octavio Gutierrez

Género: masculino

Edad: Desconocida

Estado: Jalisco

Datos del auto

Campo	Valor
Marca:	Chevrolet
Modelo:	2008
Placas:	122-HGZ
Limpiar	Enviar



# Práctica de laboratorio

## cotizacion.jsp

Cotización - Mozilla Firefox

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

Cotización

localhost:8084/Wel juro c

### Cotización de seguro para Automóvil

Estimado Sr(a):Octavio Gutierrez  
En función a los datos proporcionados:  
Género: masculino  
Edad: Desconocida  
Estado: Jalisco  
Marca: Chevrolet  
Modelo: 2008  
Placas: 122-HGZ

La cotización de su seguro es:

\$7,800.00 pesos

Cotización - Mozilla Firefox

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

Cotización

localhost:8084/Wel juro c

### Cotización de seguro para Automóvil

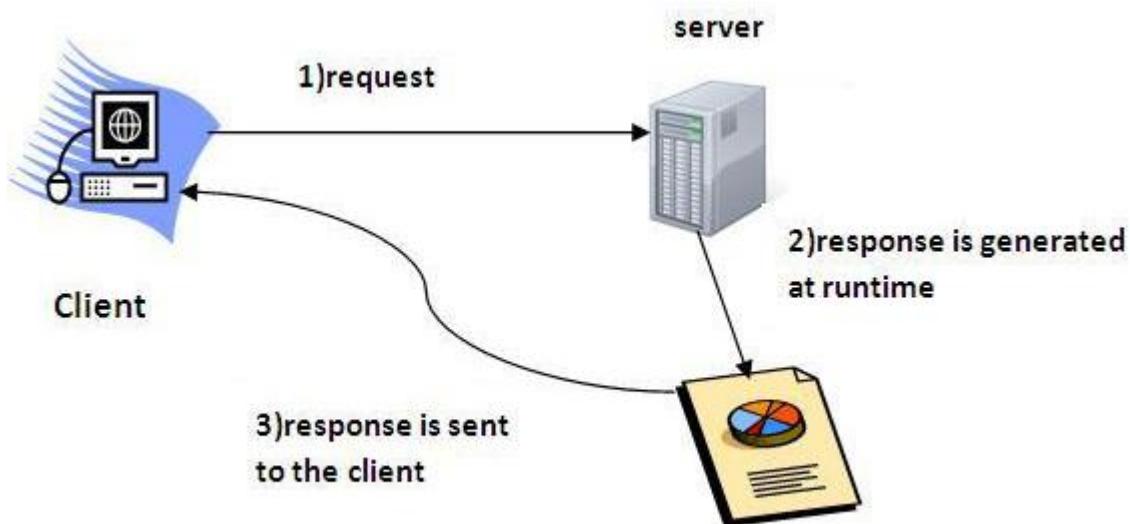
Estimado Sr(a):Octavio Gutierrez  
En función a los datos proporcionados:  
Género: masculino  
Edad: Desconocida  
Estado: Jalisco  
Marca: Chevrolet  
Modelo: 1999  
Placas: 122-HGZ

La cotización de su seguro es:

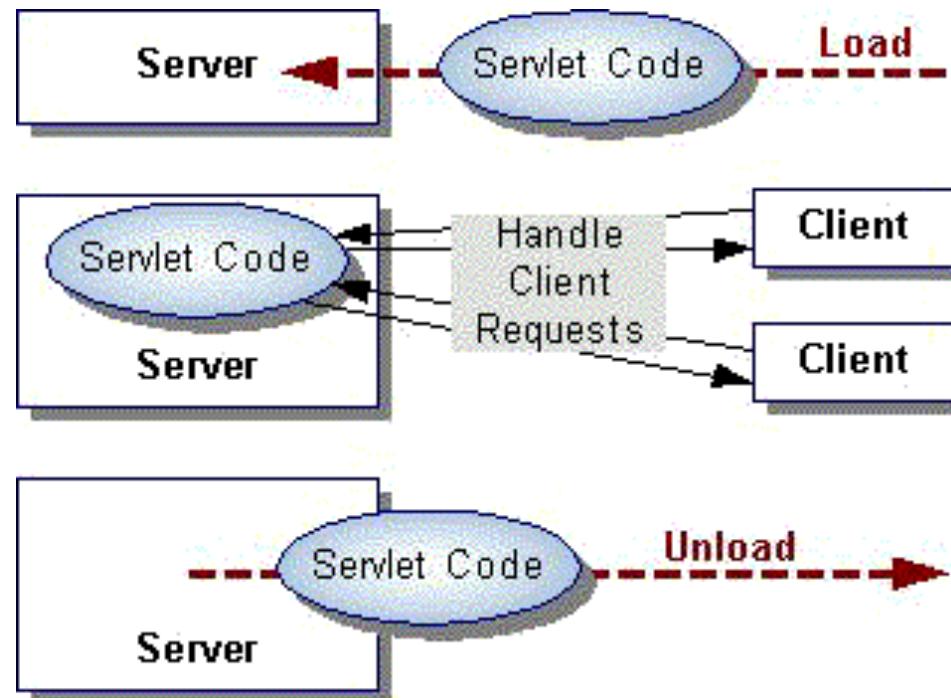
\$4,200.00 pesos

# Servlets

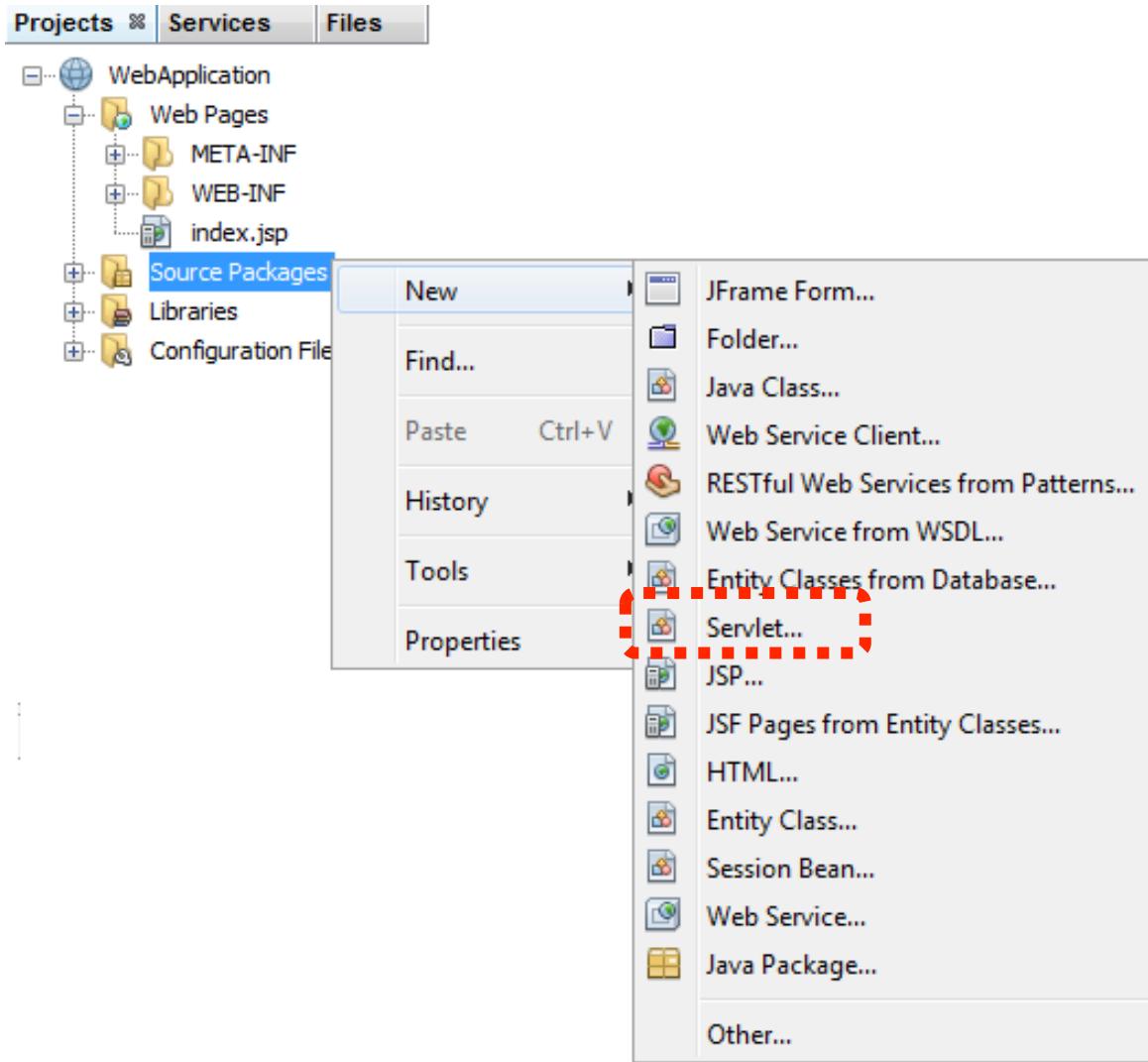
- Un servlet es una **clase Java** utilizada para ampliar las capacidades de **los servidores**



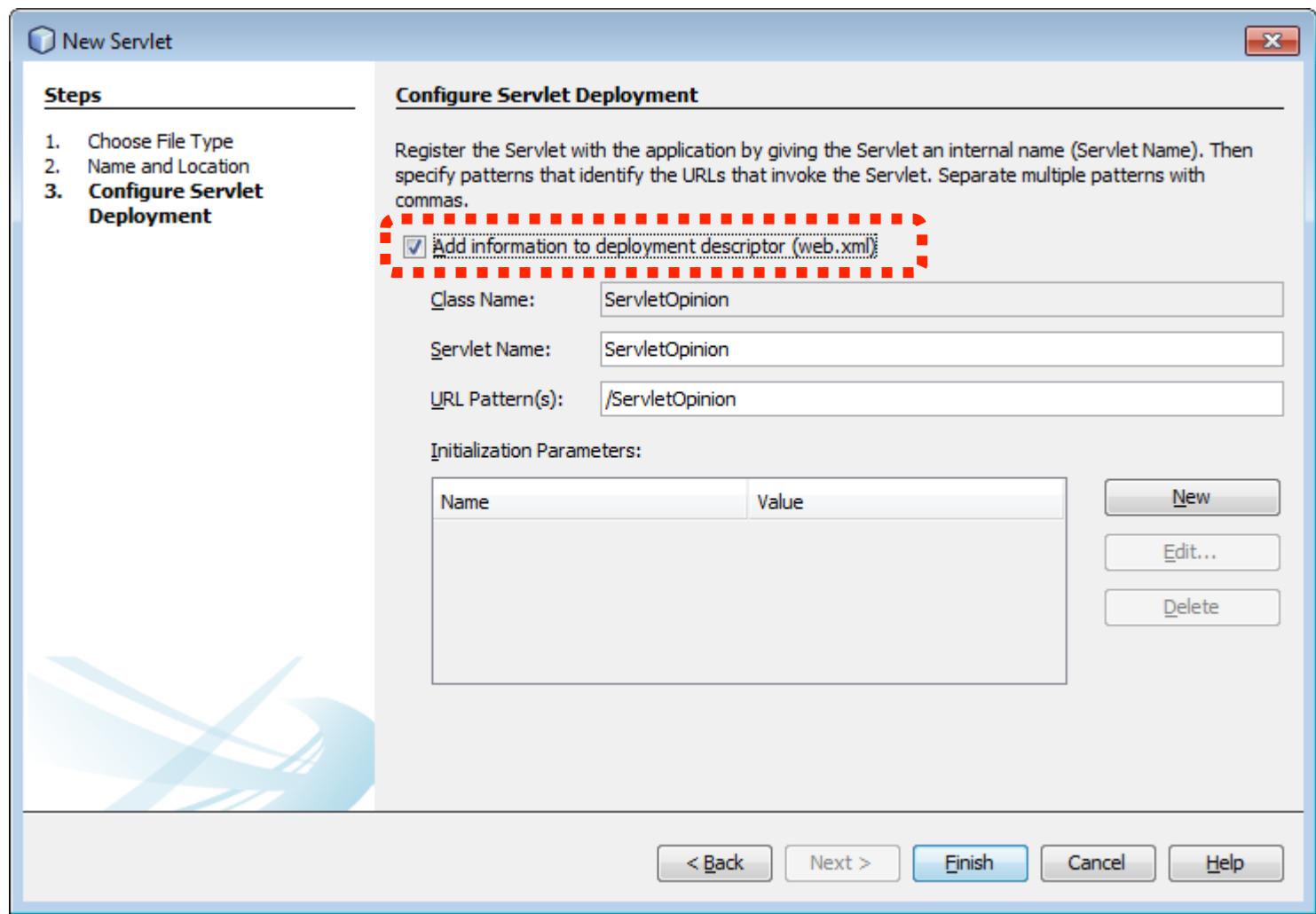
# Ciclo de vida de los Servlets



# Creación de Servlets en Netbeans



# Creación de Servlets en Netbeans

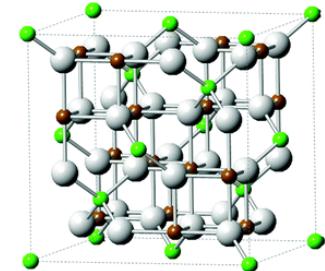


# Servlets

- Clases involucradas
  - `java.io.IOException`
  - `java.io.PrintWriter`
  - `javax.servlet.ServletException`
  - `javax.servlet.http.HttpServlet`
  - `javax.servlet.http.HttpServletRequest`
  - `javax.servlet.http.HttpServletResponse`

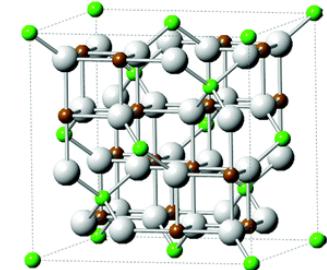


# Estructura de un Servlet



```
public class MyFirstServlet extends HttpServlet {  
    // Processes requests for both HTTP <code>GET</code> and <code>POST</code>  
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException { ...18 lines }  
  
    // Handles the HTTP <code>GET</code> method.  
    @Override  
    protected void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException { ...3 lines }  
  
    // Handles the HTTP <code>POST</code> method.  
    @Override  
    protected void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException { ...3 lines }  
  
    @Override  
    public String getServletInfo() { ...3 lines } // </editor-fold>  
}
```

# Estructura de un Servlet



```
public class MyFirstServlet extends HttpServlet {  
  
    // Processes requests for both HTTP <code>GET</code> and <code>POST</code>  
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException { ...18 lines }  
  
    // Handles the HTTP <code>GET</code> method.  
    @Override  
    protected void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException { ...3 lines }  
  
    // Handles the HTTP <code>POST</code> method.  
    @Override  
    protected void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException { ...3 lines }  
  
    @Override  
    public String getServletInfo() { ...3 lines } // </editor-fold>  
}
```

# Servlet - processRequest

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet NewServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet NewServlet at " + request.getContextPath() + "</h1>");
        out.println("</body>");
        out.println("</html>");
    } finally {
        out.close();
    }
}
```



# Estructura de un Servlet

```
public class MyFirstServlet extends HttpServlet {  
  
    // Processes requests for both HTTP <code>GET</code> and <code>POST</code>  
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException { ...18 lines }  
  
    // Handles the HTTP <code>GET</code> method.  
    @Override  
    protected void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException { ...3 lines }  
  
    // Handles the HTTP <code>POST</code> method.  
    @Override  
    protected void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException { ...3 lines }  
  
    @Override  
    public String getServletInfo() { ...3 lines } // </editor-fold>  
}
```

Get

# Estructura de un Servlet

```
public class MyFirstServlet extends HttpServlet {

    // Processes requests for both HTTP <code>GET</code> and <code>POST</code>
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException { ...18 lines }

    // Handles the HTTP <code>GET</code> method.
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException { ...3 lines }

    // Handles the HTTP <code>POST</code> method.
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException { ...3 lines }

    @Override
    public String getServletInfo() { ...3 lines } // </editor-fold>

}
```

Post

# Estructura de un Servlet

```
public class MyFirstServlet extends HttpServlet {

    // Processes requests for both HTTP <code>GET</code> and <code>POST</code>
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException { ...18 lines }

    // Handles the HTTP <code>GET</code> method.
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException { ...3 lines }

    // Handles the HTTP <code>POST</code> method.
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException { ...3 lines }

    @Override
    public String getServletInfo() { ...3 lines } // </editor-fold>
}


```



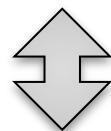
Info

# ¿Cómo utilizo un Servlet?

Nombre del  
Servlet

```
<form action="SearchEngine" method="POST">
    <p> Palabra clave: <input type="text" name="keyword" value="" />
        <input type="submit" value="Buscar" /></p>
</form>
```

# Componentes Web de Java EE



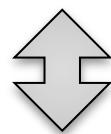
## Servlets

Clases de Java que procesan dinámicamente peticiones y construyen respuestas.

## JavaServer Pages

Documentos basados en texto que se ejecutan como servlets pero permiten un enfoque más natural a la creación de contenido estático

# Componentes Web de Java EE



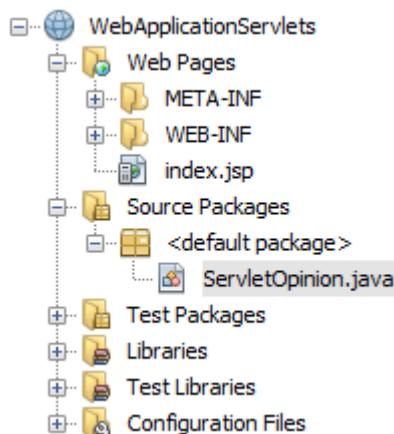
## Servlets

Más adecuado para **aplicaciones orientadas a servicios** (los endpoints de los servicios web se implementan como servlets).

## JavaServer Pages

Más apropiado para **la generación de markup basado en texto**, tales como HTML y XML

# Mi primer Servlet



JSP Page - Mozilla Firefox  
Archivo Editar Ver Historial Marcadores Herramientas Ayuda  
JSP Page +  
localhost:8084/Web web

**Por favor de su opinión de los Servlets:**

Nombre   
Apellidos   
Opinión que le merecen los Servlets  
 mala  regular  buena  
Comentarios  

Desde mi punto de vista, los servlets son una tecnología sorprendente.

Servlet ServletOpinion - Mozilla Firefox  
Archivo Editar Ver Historial Marcadores Herramientas Ayuda  
Servlet ServletOpinion +  
localhost:8084/Web web

## Resultado de la encuesta

Mi nombre es: Gustavo Rodriguez  
Mi opinión es: regular  
Comentarios: Desde mi punto de vista, los servlets son una tecnología sorprendente.



# Forward desde un Servlet

```
import javax.servlet.RequestDispatcher;
```

```
String next = "/pagina.html";
```

```
RequestDispatcher dispatcher =  
getServletContext().getRequestDispatcher(next);
```

```
dispatcher.forward(request,response);
```



# Redirect desde un Servlet

```
response.sendRedirect("pagina.html");
```



# Forward VS Redirect



- **Forward**

- Se realiza internamente por el Servlet.
- El navegador no se entera (su URL original permanece intacta.)
- Los objetos (parámetros) de la solicitud original están disponibles.



- **Redirect**

- La aplicación web instruye al navegador obtener una segunda URL.
- Ligeramente más lento que un Forward.
- Los objetos (parámetros) de la solicitud original NO están disponibles.

# Práctica Rápida: Redirect desde Servlets

```
response.sendRedirect("pagina.html");
```



<http://www.google.com/#q=keyword>



WebApplicationGoogleSearch

# Práctica de Servlets: El Zodiaco



Signo	Fechas
Aries	21 de marzo - 20 de abril
Tauro	21 de abril - 21 de mayo
Géminis	22 de mayo - 21 de junio
Cáncer	22 de junio - 22 de julio
Leo	23 de julio - 22 de agosto
Virgo	23 de agosto - 22 de septiembre
Libra	23 de septiembre - 22 de octubre
Escorpio	23 de octubre - 22 de noviembre
Sagitario	23 de noviembre - 21 de diciembre
Capricornio	22 de diciembre - 20 de enero
Acuario	21 de enero - 19 de febrero
Piscis	20 de febrero - 20 de marzo

# Práctica de Servlets: El Zodiaco

## index.jsp



### TIPS:

```
import java.util.GregorianCalendar;
```

```
GregorianCalendar cal = new GregorianCalendar(2014, 2, 21);  
cal.after( /*GregorianCalendar Object*/);  
cal.before( /*GregorianCalendar Object*/);  
cal.equals(/*GregorianCalendar Object*/);
```

¿Cuál es tu fecha de nacimiento?	
Dia	22
Mes	11
Año	1990
<input type="button" value="Limpiar"/>	<input type="button" value="Enviar"/>

!Cuidado!  
2 equivale a  
Marzo



WebApplicationZodiac

# Práctica de Servlets: El Zodiaco



## SignZodiac Servlet

Si en index.jsp se  
ingresaron letras,  
redireccionar a  
index.jsp nuevamente.  
**Tip: Try-Catch**

Servlet SignZodiac - Mozilla Firefox

Archivo Editar Ver Histórico Marcadores Herramientas Ayuda

Servlet SignZodiac

localhost:8084/Web

Zodiaco

Tu signo es: Escorpión

Ver signos compatibles

## CompatibleSigns Servlet

Servlet CompatibleSigns - Mozilla Firefox

Archivo Editar Ver Histórico Marcadores Herramientas Ayuda

Servlet CompatibleSigns

localhost:8084/Web

Los signos compatibles con Escorpión son:

Aries, Tauro, Geminis, Cáncer, Leo, Virgo, Libra, Escorpio, Sagitario, Capricornio, Acuario y Piscis

# Práctica de Servlets: El Zodiaco - Versión 1.1

## index.jsp

Si hubo un error en la captura, indicarlo al momento de redirigir

## Zodiaco

¡Hubo un error en la captura de los datos!

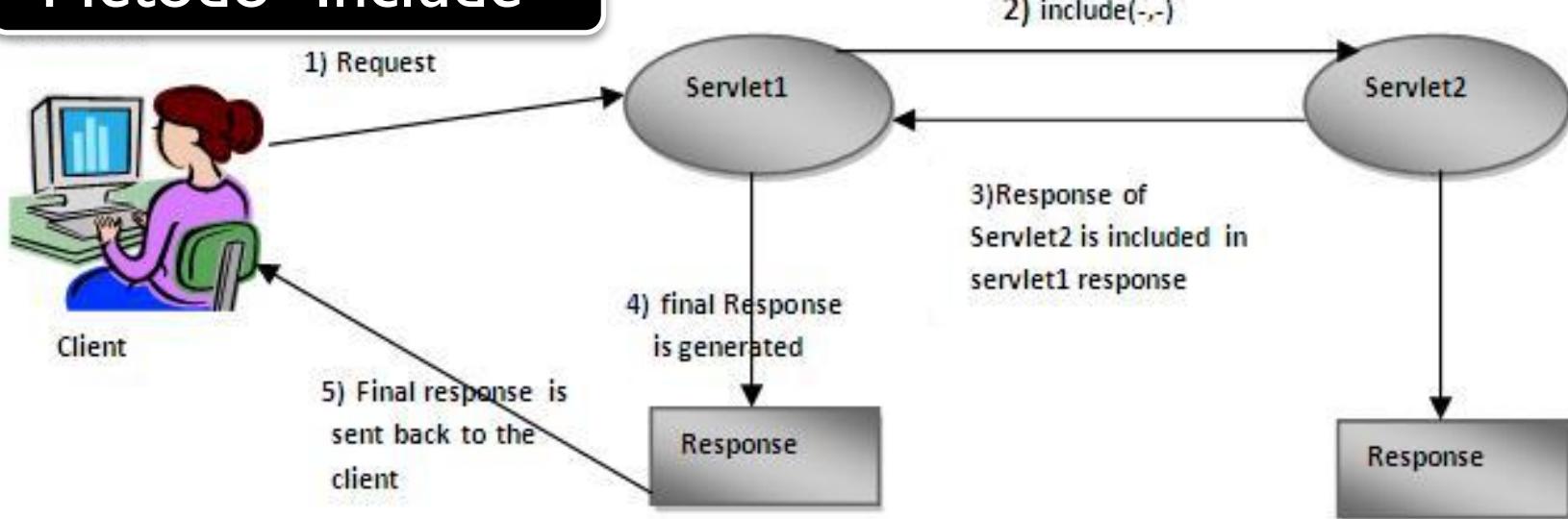
¿Cuál es tu fecha de nacimiento?

Día	
Mes	
Año	
<input type="button" value="Limpiar"/>	<input type="button" value="Enviar"/>



# Colaboración entre Servlets

## Método “Include”

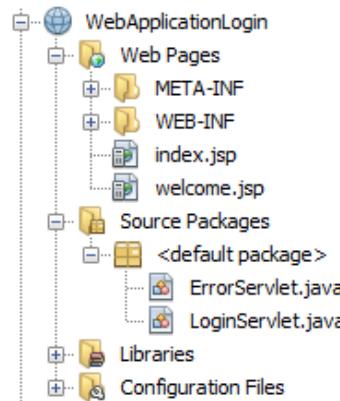


```
import javax.servlet.RequestDispatcher;
```

```
String nextJSP = "/servlet2";
```

```
RequestDispatcher dispatcher = getServletContext().getRequestDispatcher(nextJSP);  
dispatcher.include(request, response);
```

# Colaboración entre Servlets

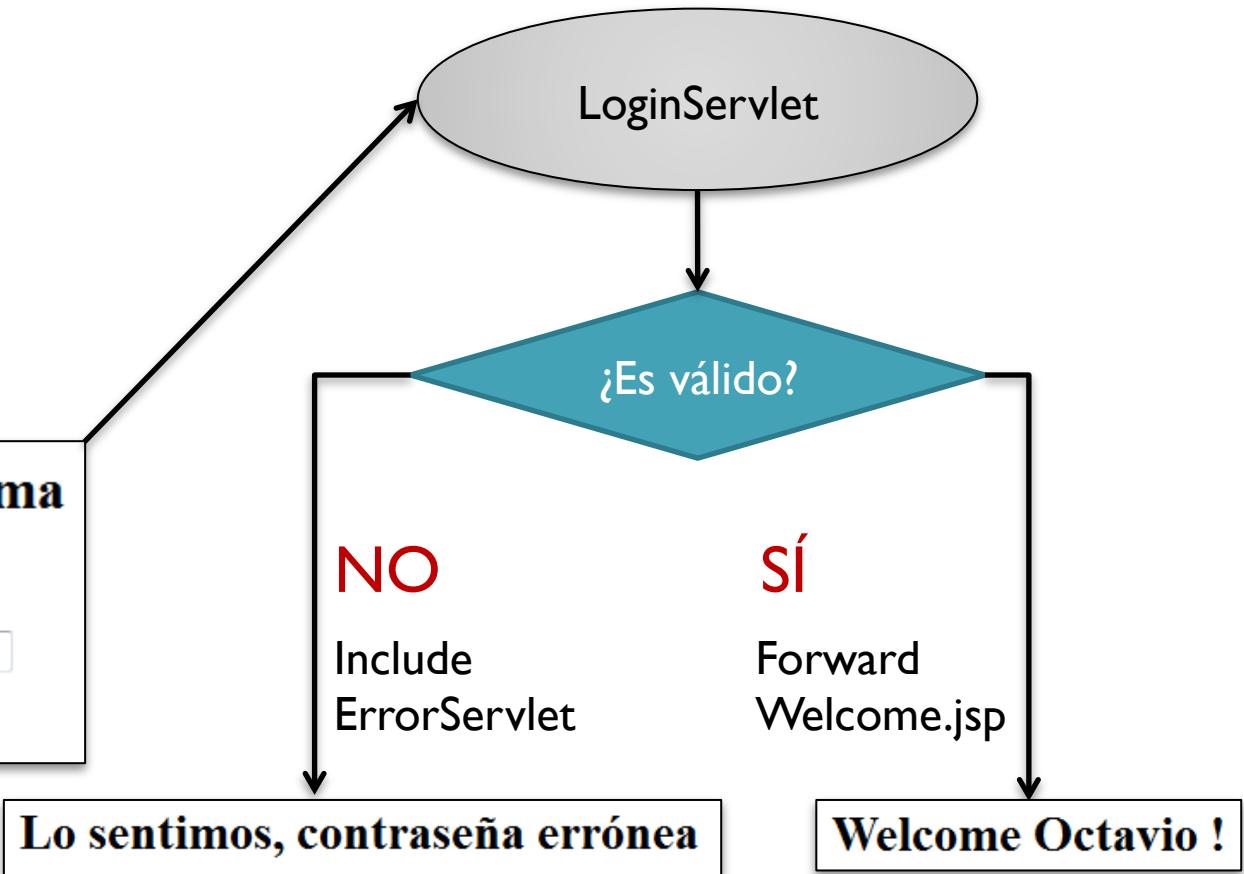


**Ingreso al Sistema**

nombre:

contraseña:

index.jsp



WebApplicationLogin

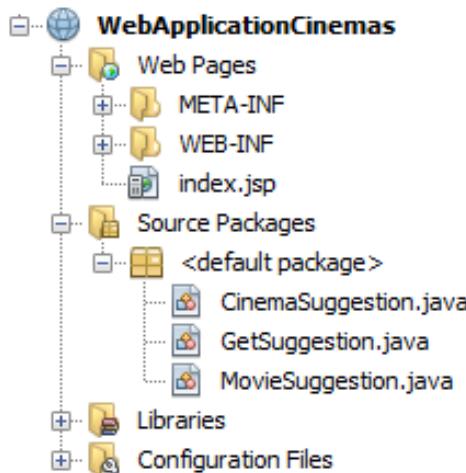
# Práctica

# Colaboración entre Servlets



Método “Include”

Index.jsp



A screenshot of a Mozilla Firefox browser window titled 'CineMAS - Mozilla Firefox'. The address bar shows 'localhost:8085/WebApplicationCinemas'. The main content area displays the following:

**¡Bienvenido a CineMAS!**

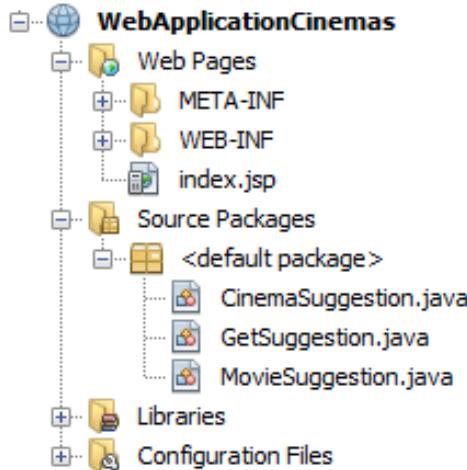
Permitenos recomendarte una Película.

Zona:

Tipo de película:

# Práctica

# Colaboración entre Servlets



WebApplicationCinemas2

## GetSuggestion Servlet

The screenshot shows a Mozilla Firefox window titled 'Servlet GetSuggestion - Mozilla Firefox'. The address bar displays 'localhost:8085/WebApplicationCinemas'. The main content area contains the following text:

**Nuestras recomendaciones son:**

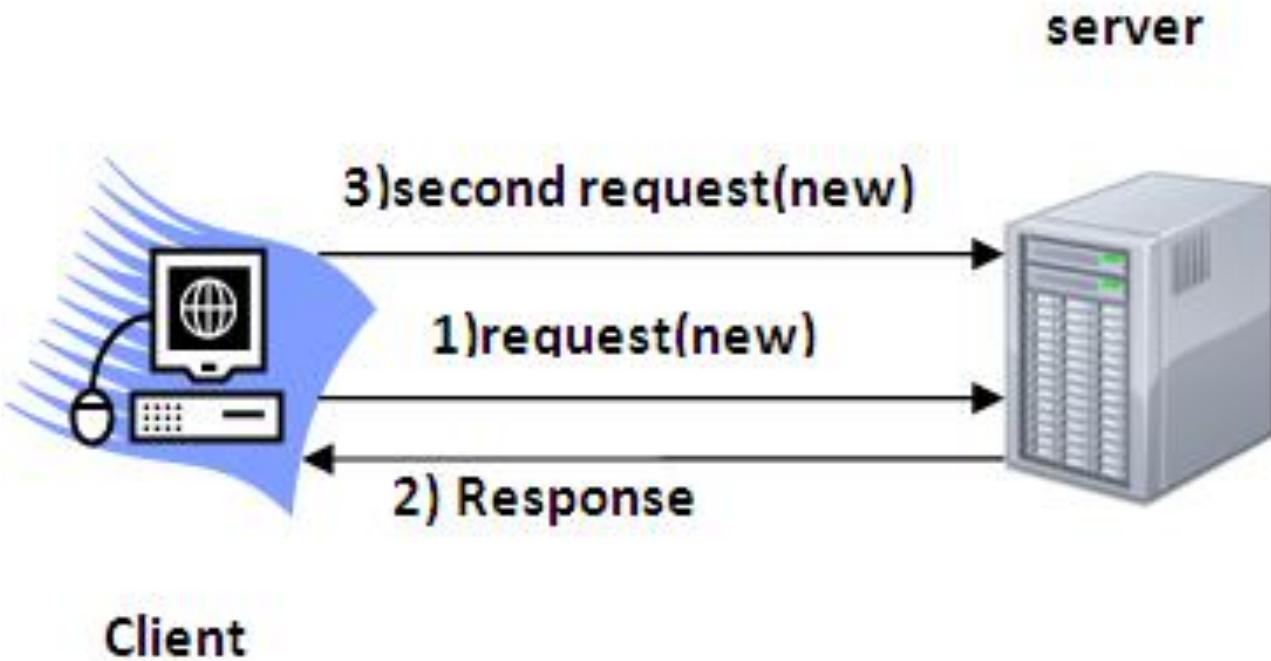
Los cines más cercanos a tu ubicación son:  
CineMas del Centro - CineMas Central

Las películas del género que te gusta son:  
El Chanfle - La Familia Burrón

Two callout boxes point to specific parts of the text:

- A grey callout box on the right points to the text 'CineMas del Centro - CineMas Central' and is labeled 'CinemaSuggestion Servlet'.
- A grey callout box on the right points to the text 'El Chanfle - La Familia Burrón' and is labeled 'MovieSuggestion Servlet'.

# Manejo de Sesiones en Servlets

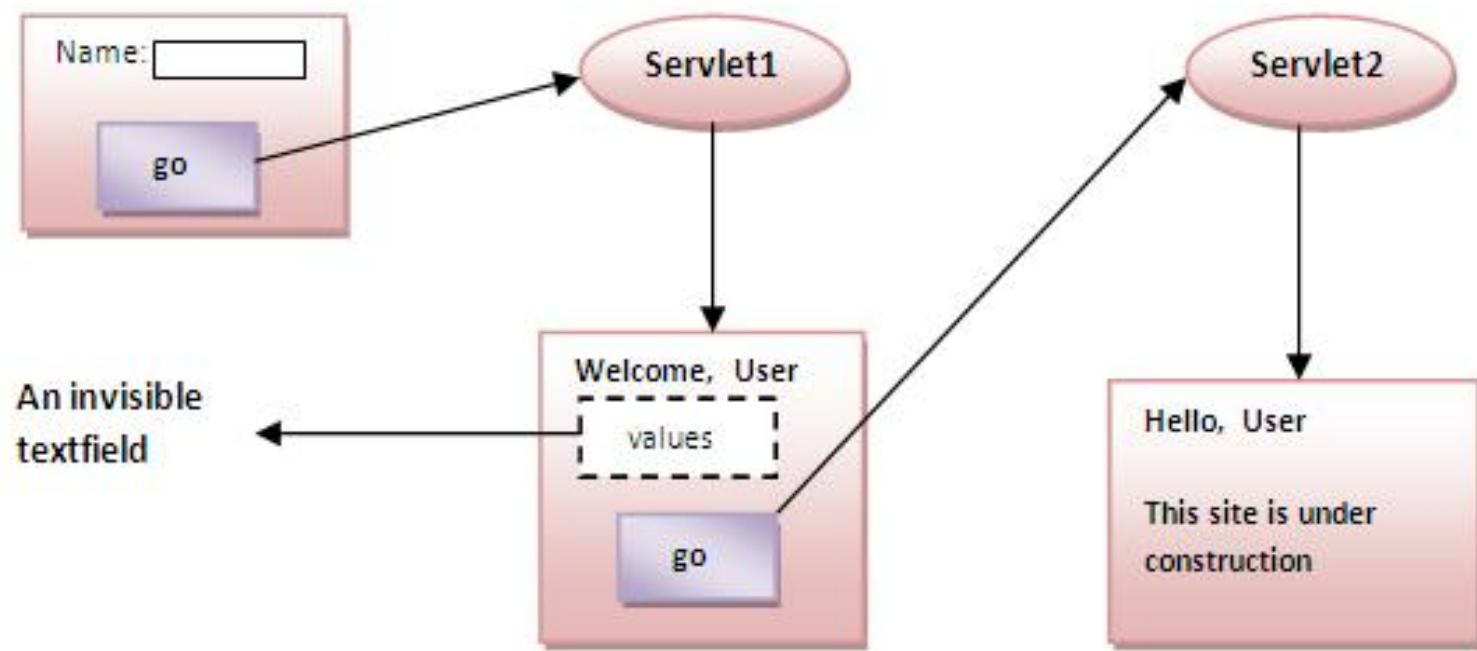


# Manejo de Sesiones en Servlets

- Las sesiones se pueden implementar con:
  - **Hidden Form Fields**
  - **URL Rewriting**
  - **Cookies**
  - **HttpSession**



# Sesiones con campos ocultos



```
out.print("<input type='hidden' name='username' value='"+n+"'">")
```

# Sesiones con campos ocultos

- **Ventaja**

- Siempre funciona independientemente de si las cookies están deshabilitadas o no.

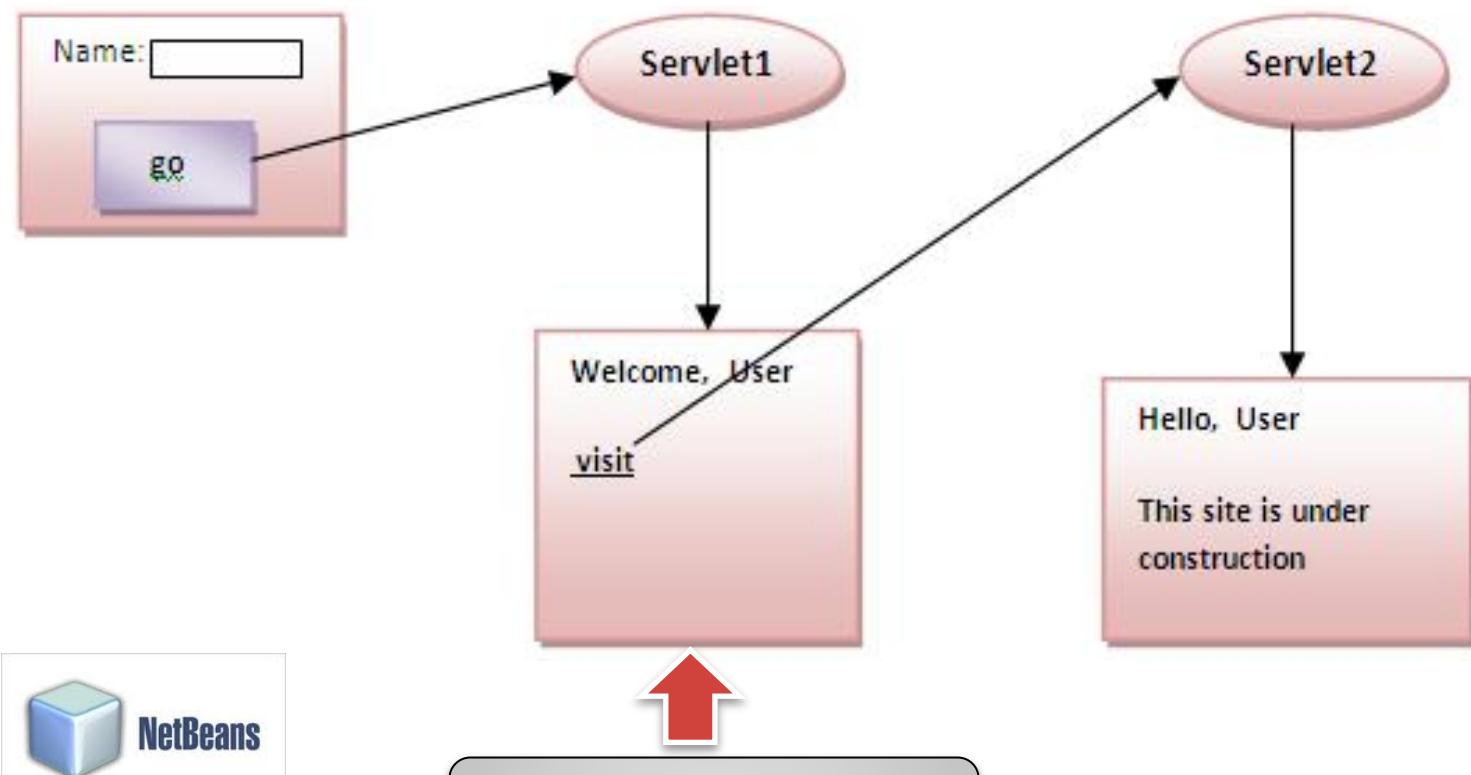
- **Desventajas**

- Se requiere del envío de información extra en cada página.
- Sólo se puede utilizar texto plano.



# Sesiones con Reescritura de URLs

url?name1=value1&name2=value2



# Sesiones con Reescritura de URLs

- **Ventajas**

- Siempre funciona independientemente de si las cookies están deshabilitadas o no.
- No requiere del envío de información extra en cada página.

- **Desventajas**

- Sólo funciona con **links**.
- Sólo se puede utilizar texto plano.

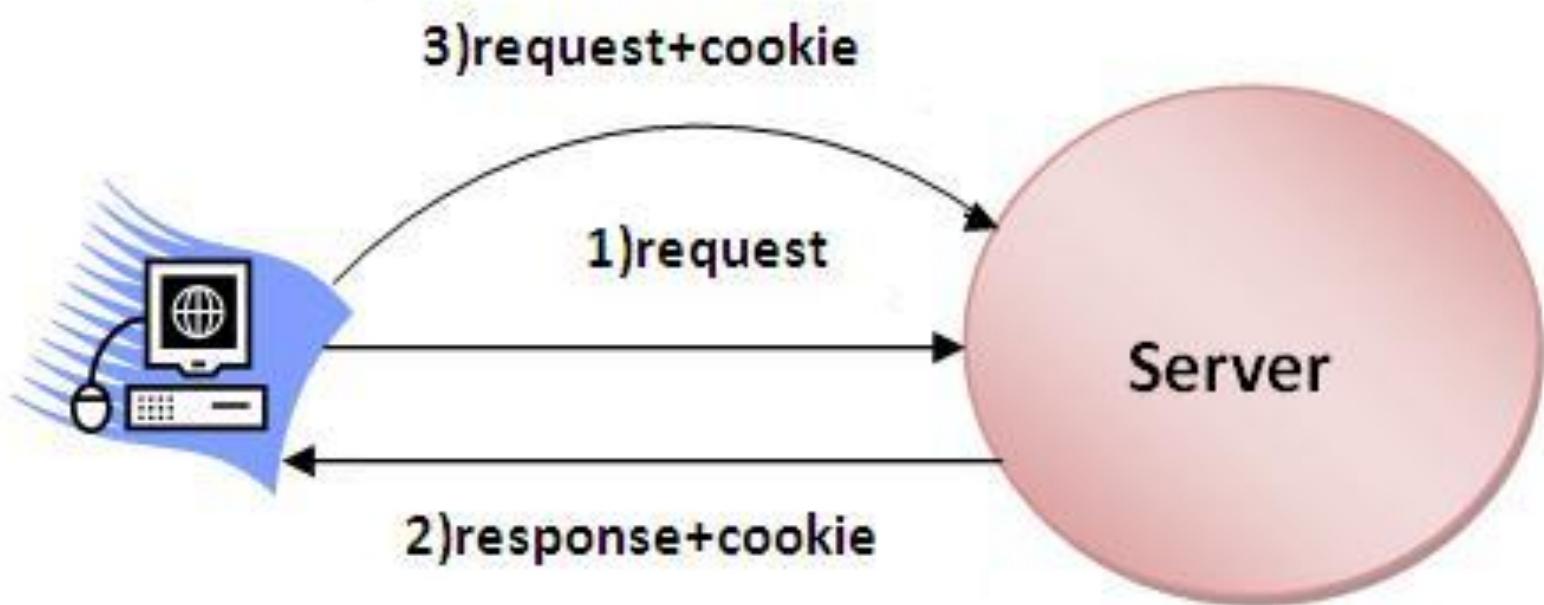


# Sesiones con Cookies en Servlets

- Una cookie es una pequeña pieza de información “**persistente**” entre múltiples solicitudes de clientes.
- Una cookie tiene:
  - Un nombre
  - Un solo valor
  - Atributos opcionales: comentarios, ruta de acceso, edad máxima y versión.



# ¿Cómo funcionan las Cookies?



\* Las cookies se almacenan en la memoria caché del navegador.

# Más sobre cookies

- **Ventajas**

- Técnica más simple de mantener el estado.
- Las cookies se mantienen en el lado del cliente.

- **Desventajas**

- No funcionará si las cookies están deshabilitadas en el navegador.
- Sólo se puede usar texto plano.



# ¿Cómo detectar si las cookies están habilitadas?

```
<html>
  <head>
    <title>Verificador de Cookies</title>
    <script language="javascript" type="text/javascript">
      function detect() {
        if (navigator.cookieEnabled) {
          alert('Your browser has cookies enabled.');
        } else {
          alert('Your browser has cookies disabled.');
        }
      }
    </script>
  </head>
  <body onload="detect();">
    ...
  </body >
</html>
```

# JSESSIONID en JSP Servlet



JSESSIONID es una cookie generada por un contenedor de Servlets, ejemplo: Tomcat & Glassfish.

# Clase Cookies



## **javax.servlet.http.Cookie**

### Constructores

`Cookie()`

`Cookie(String name, String value)`

### Métodos

`public void setMaxAge(int expiry)`

`public String getName()`

`public String getValue()`

`public void setName(String name)`

`public void setValue(String value)`

# Otros métodos para Cookies

**public void addCookie(Cookie ck)**

Método de **HttpServletResponse**.

Añade una cookie a un objeto **response**



**public Cookie[] getCookies()**

Método de **HttpServletRequest**.

Regresa **todas** las cookies de un navegador.

# ¿Cómo crear una Cookie?

```
Cookie ck=new Cookie("nombre","valor");  
response.addCookie(ck);
```



# ¿Cómo borrar una Cookie?

```
Cookie ck=new Cookie("nombre","a");
```

```
//borrando el valor de una cookie
```

```
ck.setMaxAge(0);
```

```
//cambiando la edad máxima a 0 segundos
```

```
response.addCookie(ck);
```

```
//añadiendo la cookie a la respuesta
```



# ¿Cómo obtener las Cookies?

```
Cookie ck[]=request.getCookies();
```

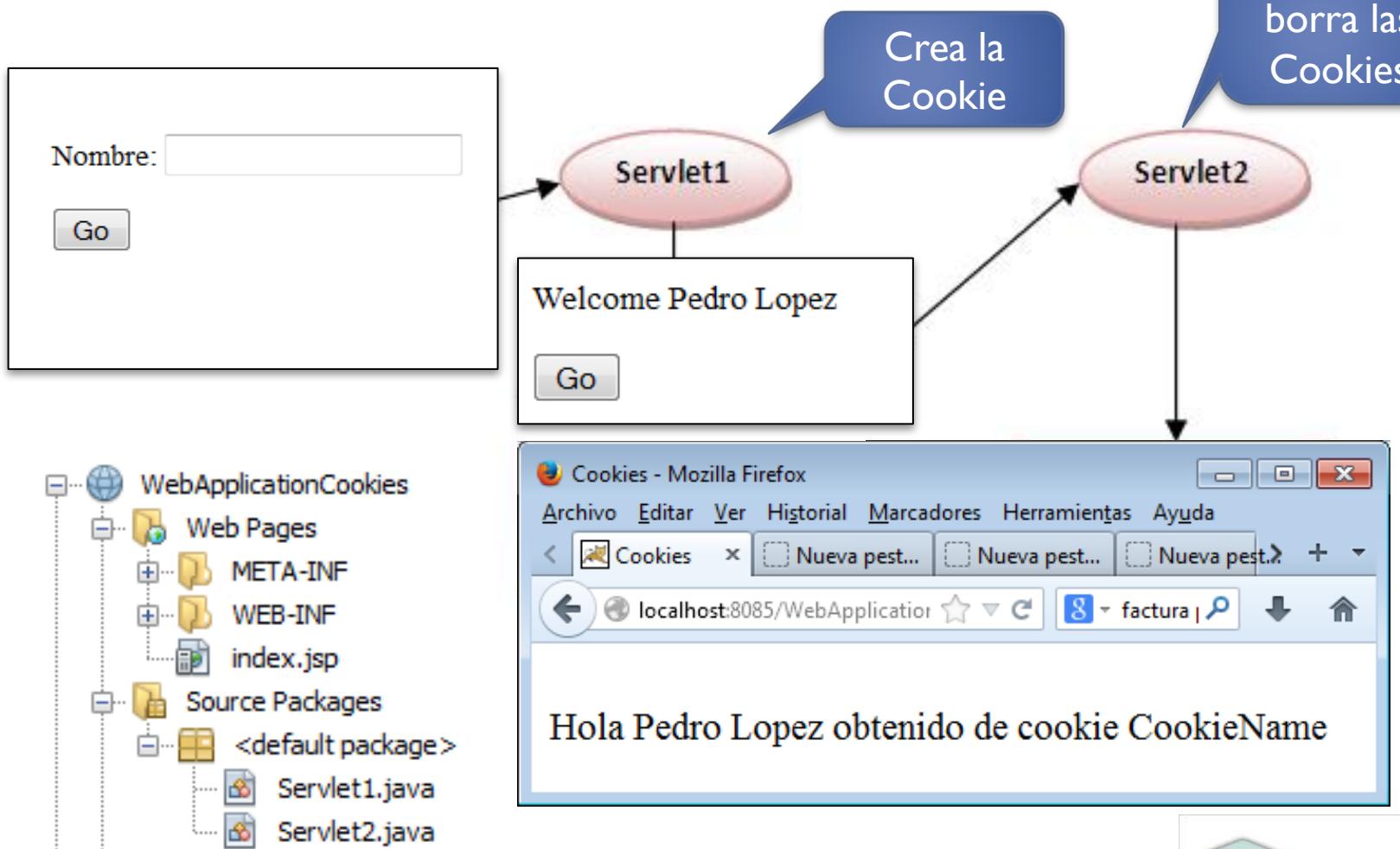
```
for(int i=0; i<ck.length; i++){
```

```
    out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());
```

```
}
```



# Horneando mis primeras Cookies



WebApplicationCookies

# Práctica de Sesión con Cookies

**Login**



**Logout**

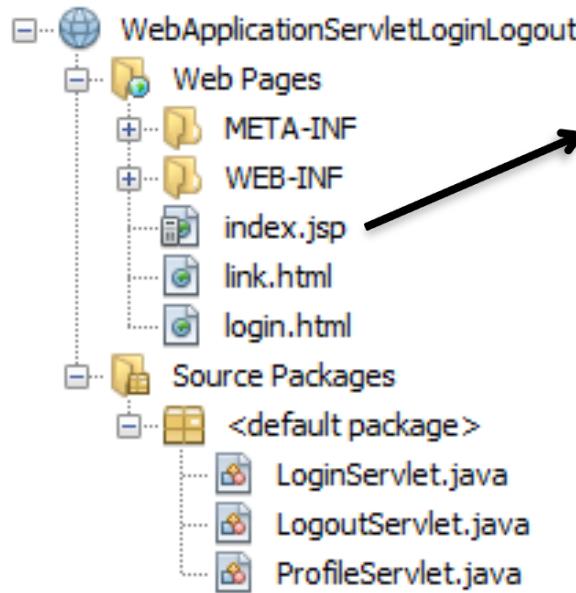


**Profile**

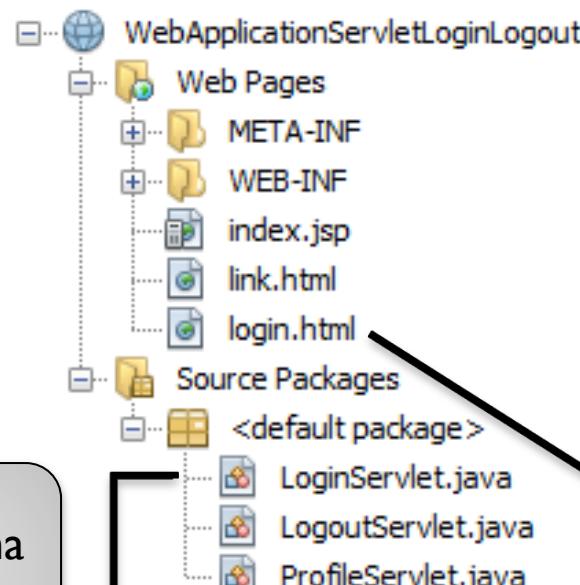


WebApplicationServletLoginLogout

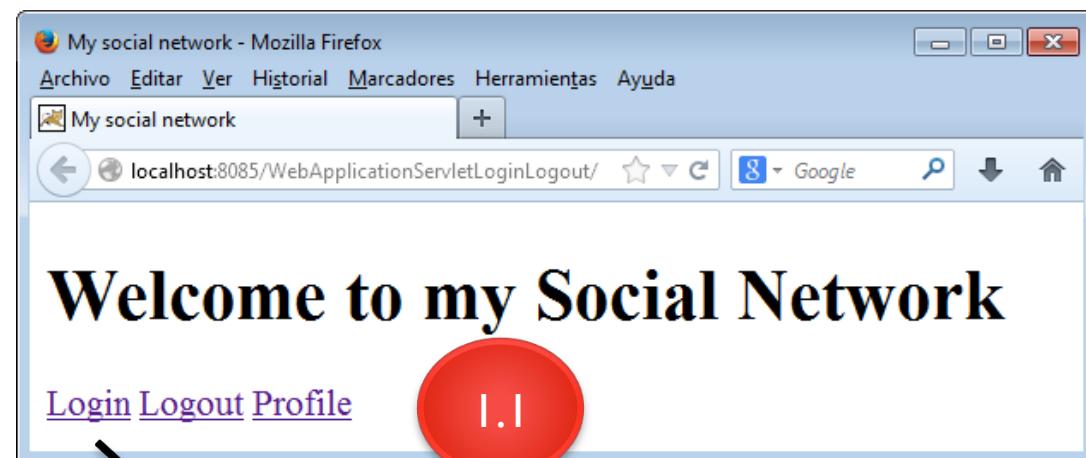
# Práctica de Sesión con Cookies



# Práctica de Sesión con Cookies



¡Crea una cookie!

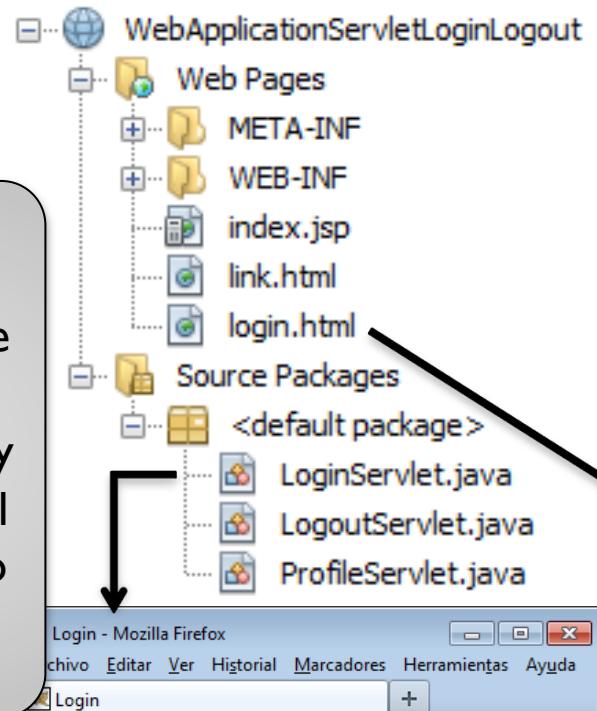


A screenshot of Mozilla Firefox showing a "Login" page. The URL is "localhost:8085/WebApplicationServletLoginLogout/login.html". The page displays the message "You are successfully logged in! Welcome Juan Perez!" followed by a yellow box containing "<hr>". A large red arrow points from this screen back towards the top center of the slide.

A screenshot of Mozilla Firefox showing a login form. The URL is "http://localhost:8085/WebApplicationServletLoginLogout/login.html". The form has fields for "Name" (containing "Juan Perez") and "Password" (containing five dots). A "login" button is at the bottom. A red arrow points from this screen up towards the top center of the slide.

# Práctica de Sesión con Cookies

Imprime mensaje e incluye link.html y login.html utilizando **include**

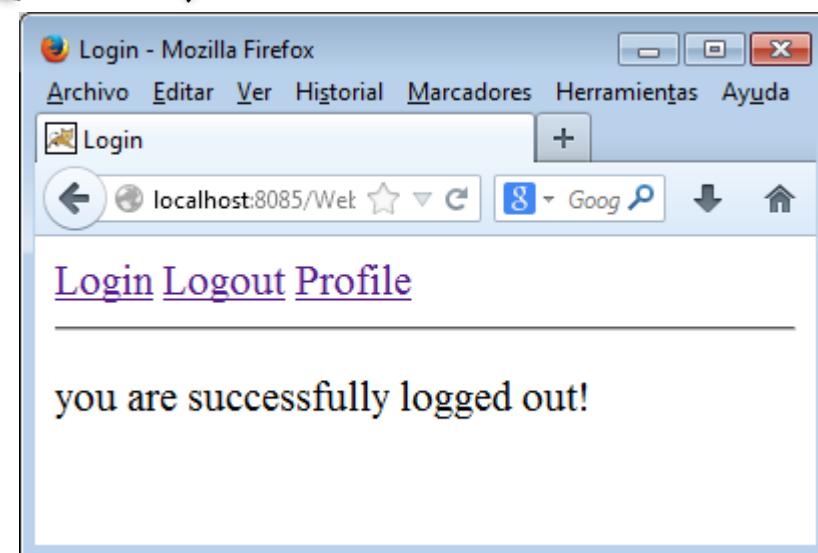
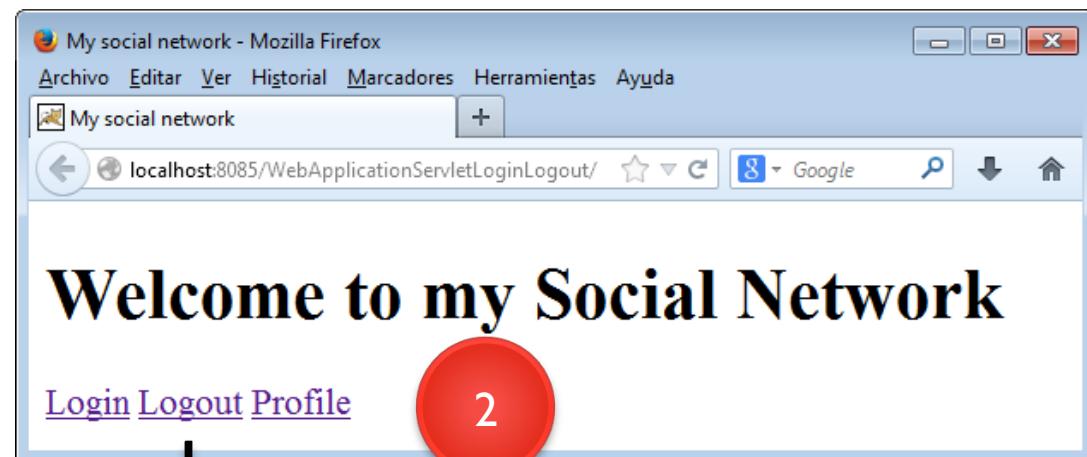
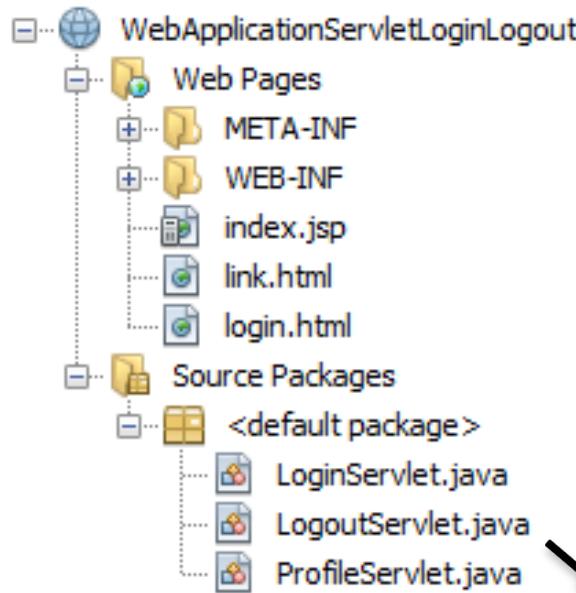


A screenshot of a Mozilla Firefox browser window titled "Login - Mozilla Firefox". The address bar shows "localhost:8085/WebApplicationServletLoginLogout/Login". The page content displays "Login Logout Profile" followed by "Sorry, wrong user name or password!". Below this are input fields for "Name:" and "Password:", and a "login" button at the bottom.



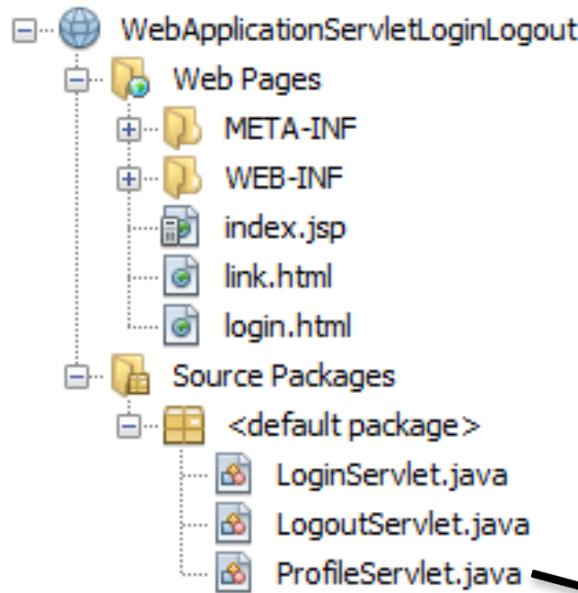
A screenshot of a Mozilla Firefox browser window titled "Mozilla Firefox". The address bar shows "http://localhost:8085/WebApplicationServletLoginLogout/login.html". The page content displays a login form with "Name: Juan Perez" and "Password: \*\*\*\*\*". A "login" button is at the bottom. A large red arrow points from the "login.html" link in the top browser to this login form.

# Práctica de Sesión con Cookies



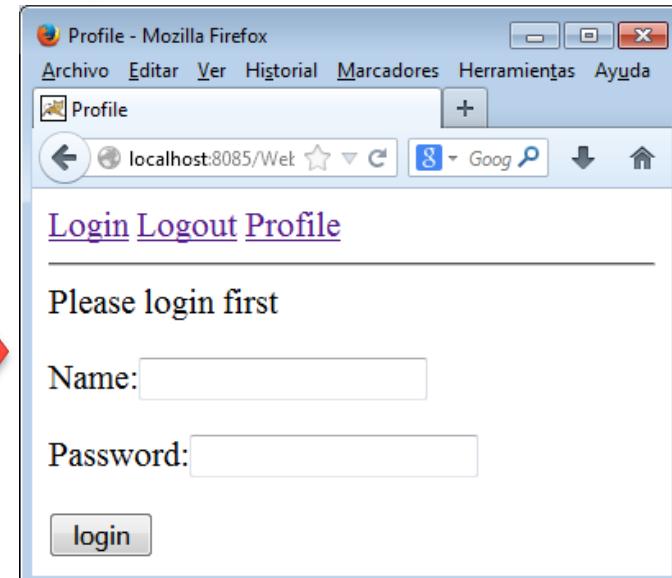
¡Borra las  
cookies!

# Práctica de Sesión con Cookies

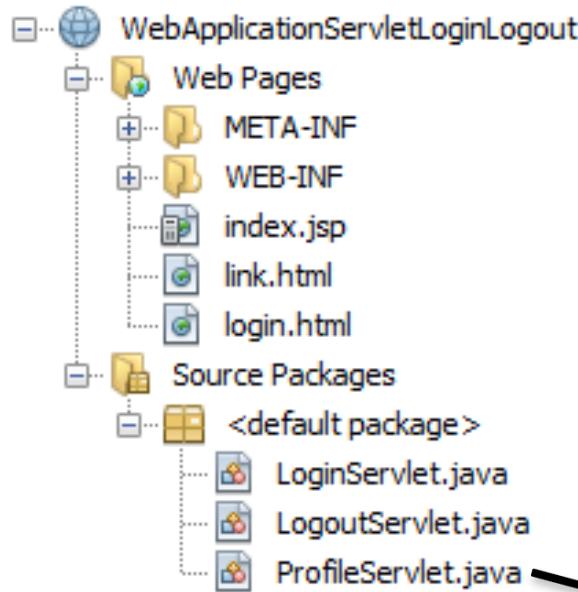


## ProfileServlet

Verifica si está una cookie,  
**Si no está** imprime mensaje  
e **incluye** "login.html"



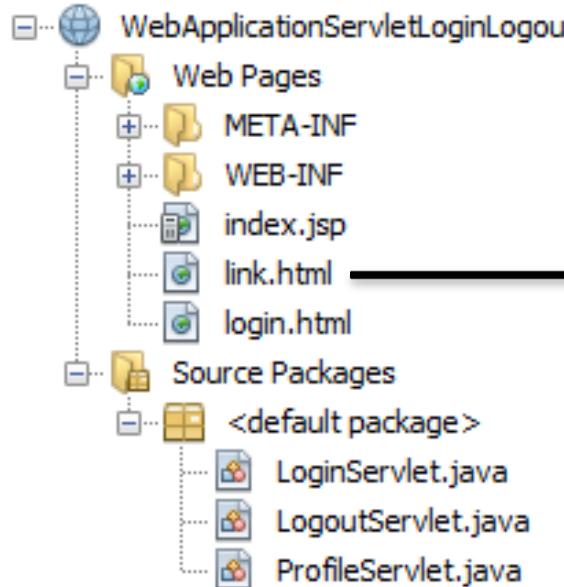
# Práctica de Sesión con Cookies



**ProfileServlet**  
Verifica si está una cookie,  
**Si está** imprime mensaje  
con su nombre



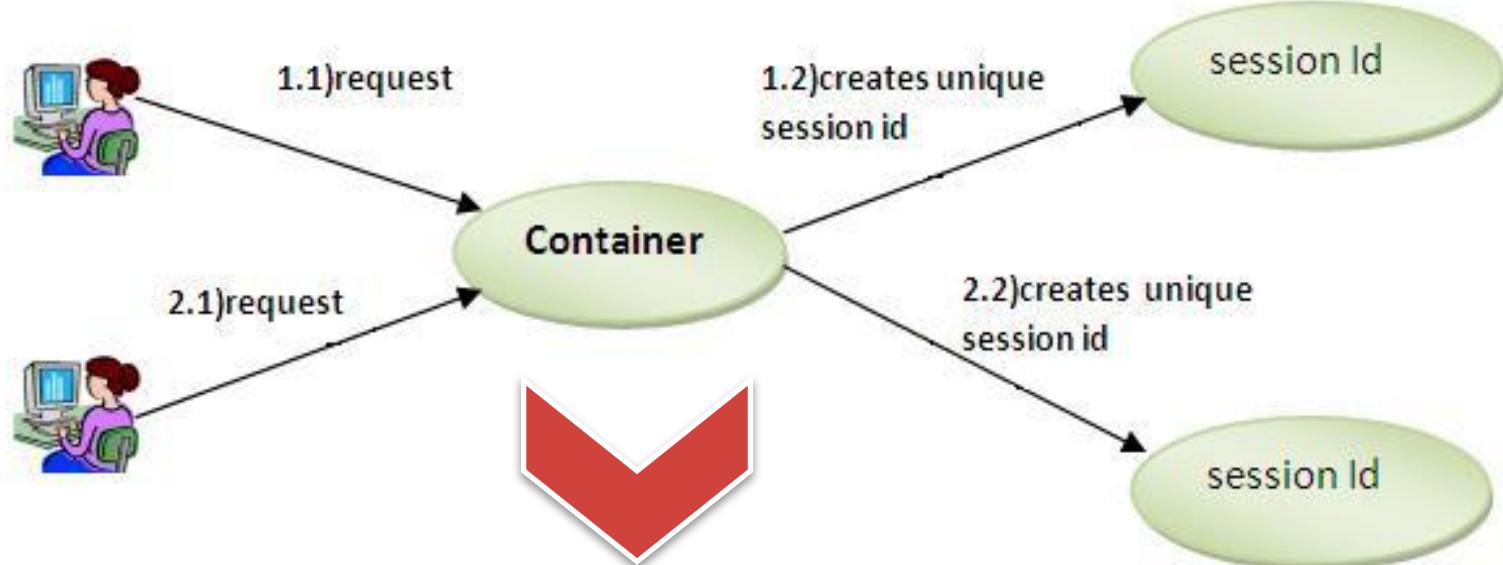
# Práctica de Sesión con Cookies



Login Logout Profile

Se incluye en todos los  
Servlets utilizando **include**

# Sesiones HTTP



GlassFish



# Más sobre Sesiones HTTP

- **Ventajas**

- Se implementa de manera fácil.
- Se puede almacenar cualquier tipo de objeto.
- Son más seguras.

- **Desventajas**

- Más costoso en términos de rendimiento dado que se mantiene en el lado del Servidor.



# Objeto **HttpSession** de **HttpServletRequest**

- **import javax.servlet.http.HttpSession;**
- **public HttpSession getSession()**

Devuelve la sesión asociada con la solicitud, o si la solicitud no tiene una sesión, crea una.

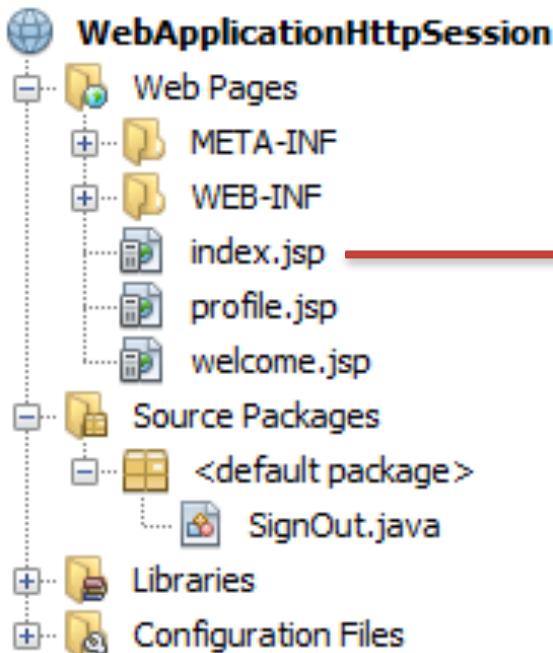


# Métodos de HttpSession

- String `getId()`
- void `setAttribute(String name, Object value)`
- Object `getAttribute(String name)`
- void `removeAttribute(String name)`
- Enumeration `getAttributeNames()`
- void `setMaxInactiveInterval(int seconds)`
- void `invalidate() // Inválida la sesión.`



# Manejo de una Sesión HTTP



Tips

```
HttpSession mySession = request.getSession();
mySession.setAttribute("username", userName);
///////////////////////////////
```

```
HttpSession mySession = request.getSession();
if (mySession.getAttribute("username")!=null)
```

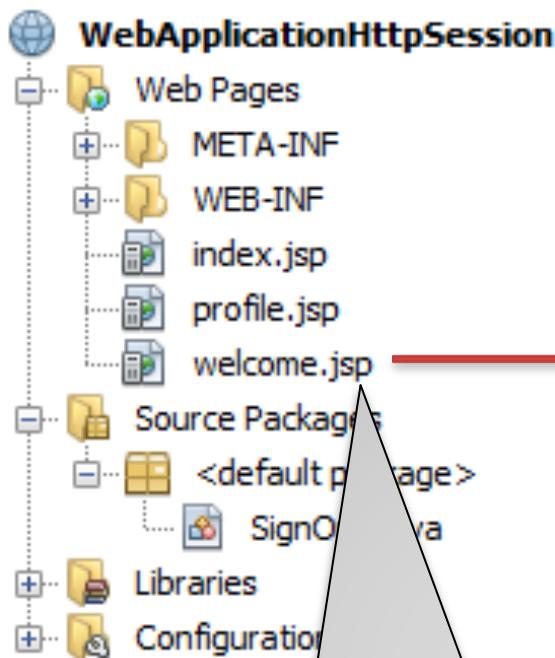
User name:

Password:

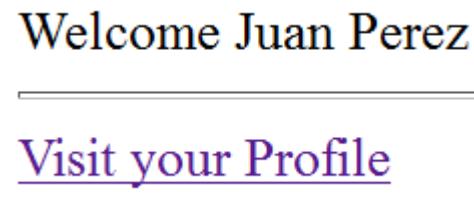


WebApplicationHttpSession

# Manejo de una Sesión HTTP



Si la contraseña fue 123456 ingresa a welcome.jsp, si no redirecciona a index.jsp

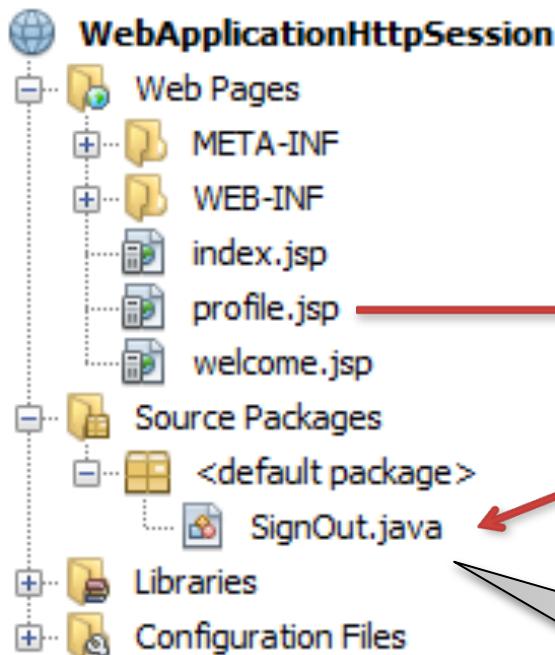


Crea una sesión con atributo  
"username"



WebApplicationHttpSession

# Manejo de una Sesión HTTP



Sólo accede si el valor del atributo  
“username” de la sesión es  
diferente de nulo

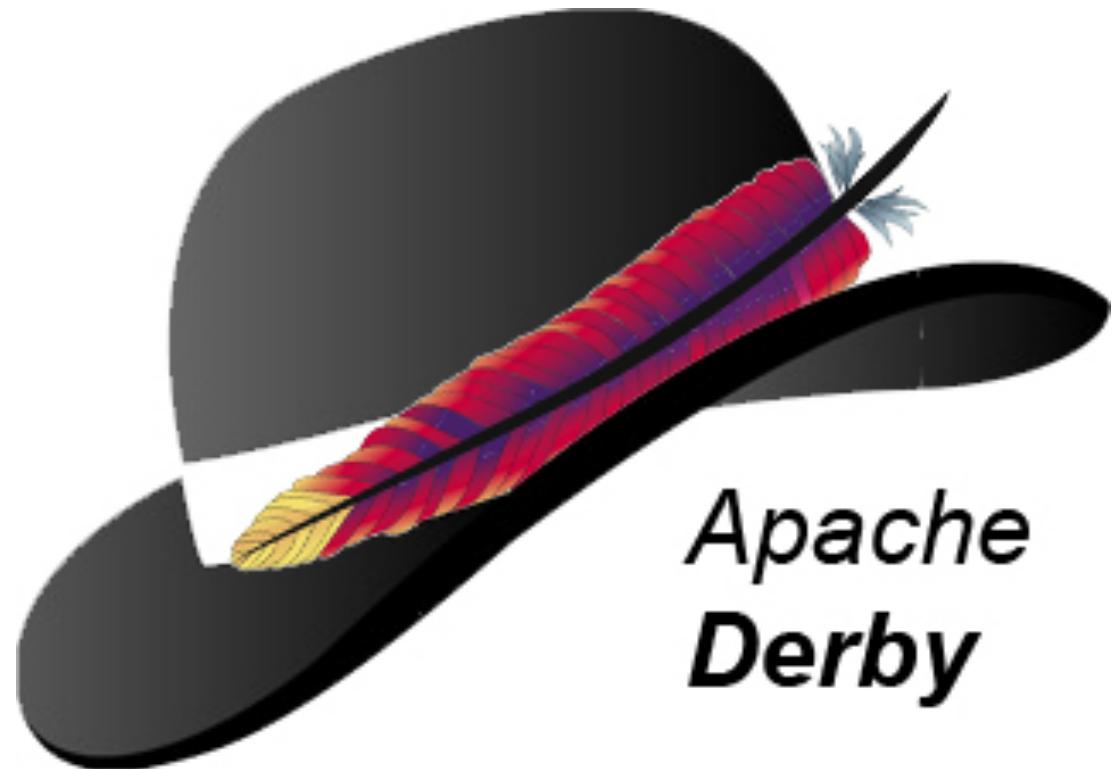
Hi Juan Perez-Session. This is your profile  
[Sign out](#)

Cierra la sesión (invalidate) y  
redirecciona a index.jsp



WebApplicationHttpSession

# Acceso a Bases de datos

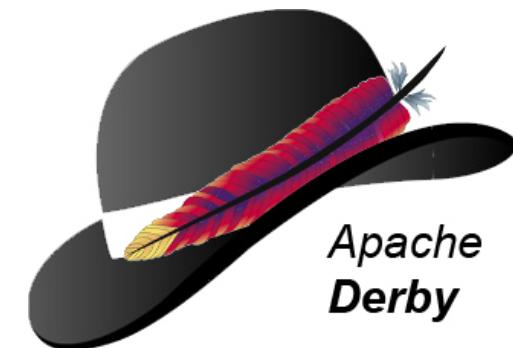


*Apache  
Derby*

# Seguridad de Java **java.policy**

C:\Program Files\Java\jdk1.7.0\_51\jre\lib\security\

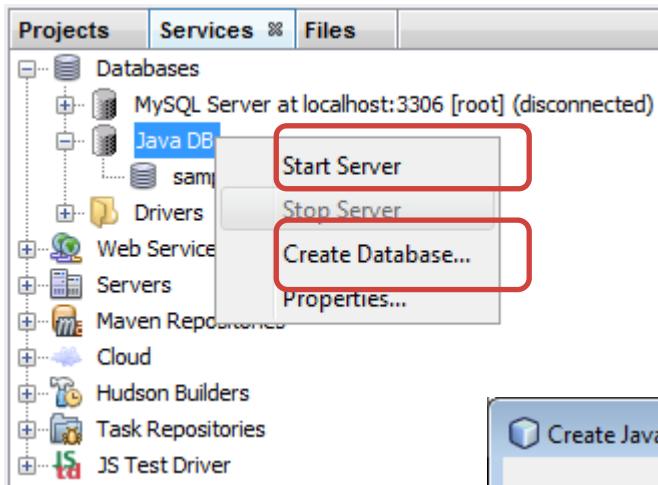
```
grant codeBase "file:${java.home}/../db/lib/*" {  
    permission java.security.AllPermission;  
};  
  
grant codeBase "file:${java.home}/../db/lib/*" {  
    permission java.net.SocketPermission "localhost:1527", "listen";  
    permission java.net.SocketPermission "localhost:1527", "resolve";  
};  
  
grant codeBase "file:/Program Files/Java/jdk1.7.0_51/db/lib/*" {  
    permission java.net.SocketPermission "localhost:1527", "listen";  
    permission java.net.SocketPermission "localhost:1527", "resolve";  
};  
  
// default permissions granted to all domains  
grant {  
    permission java.net.SocketPermission "localhost:1527", "listen";  
    permission java.net.SocketPermission "localhost:1527", "resolve";  
};
```



# Java DB (Apache Derby)

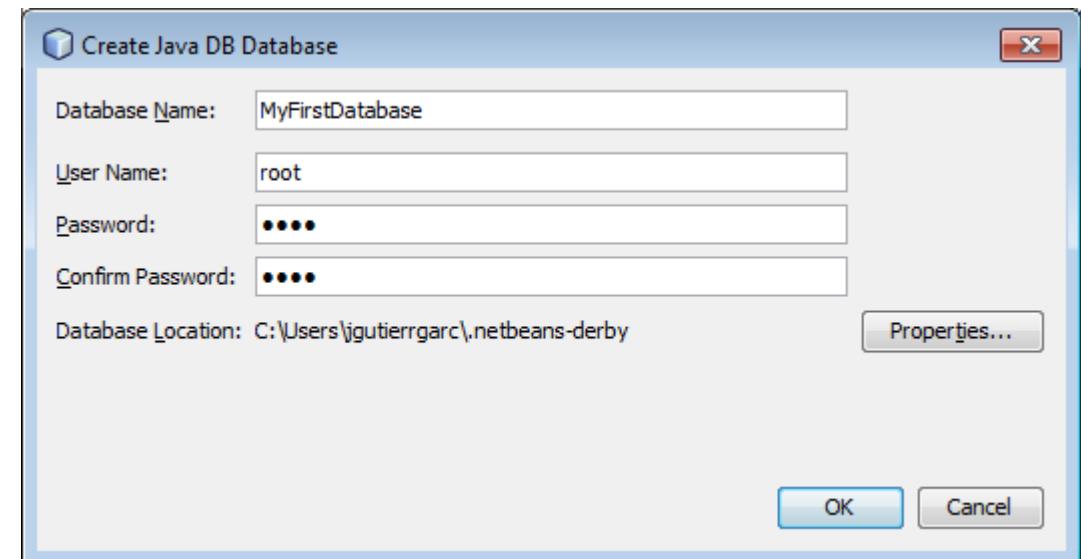


## Configuración Java DB en Netbeans



I

Arrancar el  
Servidor de BDs



2

Crear BD

# Java DB (Apache Derby)



The screenshot shows the IntelliJ IDEA interface with the 'Projects' tab selected. In the 'Databases' section, there is a connection to 'MySQL Server at localhost:3306 [root] (disconnected)' and a connection to 'Java DB' named 'MyFirstDatabase'. A context menu is open over the 'jdbc:derby://localhost:1527/MyFirstDatabase [root on ROOT]' entry, with the 'Connect...' option highlighted by a red box.

3

Conectar la BD

The screenshot shows the IntelliJ IDEA interface with the 'Projects' tab selected. In the 'Databases' section, there is a connection to 'MySQL Server at localhost:3306 [root] (disconnected)' and a connection to 'Java DB' named 'MyFirstDatabase'. Under 'MyFirstDatabase', there is a 'ROOT' schema containing 'Tables', 'Views', 'Proced', and 'Other sche'. A context menu is open over the 'Tables' entry, with the 'Create Table...' option highlighted by a red box.

4

Crear Tabla

# Java DB (Apache Derby)



5

Definir nombre y campos de la Tabla

Create Table

Table name: CUSTOMERS

Key	Index	Null	Unique	Column name	Data type	Size
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ID	NUMERIC	0
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NAME	VARCHAR	100
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	BALANCE	REAL	0

Add column

Edit

Remove

Move Up

Move Down

OK Cancel Help

# Java DB (Apache Derby)



6

## Agregar datos

The screenshot shows the Eclipse IDE interface with the Data Tools Platform plugin. On the left, the Navigator view displays the database structure under the 'Databases' node. A context menu is open over the 'CUSTOMERS' table in the 'ROOT.Tables' section, with the 'View Data...' option highlighted by a red box.

Projects Services × Files

Databases

- MySQL Server at localhost:3306 [root] (disconnected)
- Java DB
  - MyFirstDatabase
  - sample
- Drivers
- jdbc:derby://localhost:1527/MyFirstDatabase [root on ROOT]
  - ROOT
    - Tables
      - CUSTOMERS
        - ID
        - NAME
        - BALANCE
      - Indexes
      - Foreign K
    - Views
    - Procedures
  - Other schemas

The screenshot shows the Derby SQL Editor. At the top, a SQL statement is being run against the 'ROOT.CUSTOMERS' table. Below the statement, the results are displayed in a grid. The first two rows of the result set are highlighted with a red box. The grid has columns labeled '#', 'ID', and 'NAME'. The bottom of the screen shows the Derby interface with its characteristic blue and white color scheme.

...ava SQL 19 [jdbc:derby://localhost:15...]

Source History

```
1 select * from ROOT.CUSTOMERS;
2
```

select \* from ROOT.CUSTOMERS

#	ID	NAME
1	1	John Doe
2	2	Jane Doe
3	3	Bob Smith
4	4	Susan Johnson
5	5	David Wilson
6	6	Emily Davis
7	7	Michael Brown
8	8	Karen Green
9	9	Christopher Grey
10	10	Sarah White
11	11	Matthew Black
12	12	Elizabeth Red
13	13	James Blue
14	14	Olivia Purple
15	15	William Green
16	16	Alexander Yellow
17	17	Mia Pink
18	18	Lucas Orange
19	19	Isabella Purple
20	20	Benjamin Yellow

# Java DB (Apache Derby)



6

Agregar datos

Insert Record(s)

Press CTRL+Tab to exit data entry mode from the table. Press CTRL+0 to set NULL value and CTRL+1 to set DEFAULT value for a given column.

#	ID	NAME	BALANCE
1		1 Pedro	1234.0
2		2 Juan	3355.0
3	▶	3 Joaquin	323.32

Show SQL Add Row Remove OK Cancel

...ava SQL 19 [jdbc:derby://localhost:15...]

Source History

```
1 select * from ROOT.CUSTOMERS;
```

select \* from ROOT.CUSTOMERS

Page Size: 20 Total Rows: 3

#	ID	NAME	BALANCE
1		1 Pedro	1234.0
2		2 Juan	3355.0
3		3 Joaquin	323.32

# Java DB (Apache Derby)



## Librerías para manejo de base de datos

```
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;
```

# Java DB (Apache Derby)



## Para establecer una conexión

```
Class.forName("org.apache.derby.jdbc.ClientDriver");
```

```
Connection con =
```

```
    DriverManager.getConnection(
```

```
        "jdbc:derby://localhost:1527/myFirstDatabase",
```

BD

```
        "root",
```

Ubicación BD

```
        "root");
```

usuario

PW

# Java DB (Apache Derby)



## Para hacer una consulta

```
Statement query = con.createStatement();
ResultSet rs = query.executeQuery("SELECT * FROM CUSTOMERS");
```

Crear Statement  
usando la conexión

```
while(rs.next()) {
    out.println("  Id: "+ rs.getInt("ID"));
    out.println("  Name: "+ rs.getString("NAME"));
    out.println("  Balance: "+ rs.getString("BALANCE"));
}
```

Crear un ResultSet

```
con.commit();
con.close();
```

Si se modificó la BD

Cerrar la Conexión

# Java DB (Apache Derby)



## Para configurar el proyecto en Netbeans

The screenshot illustrates the process of adding the Java DB Driver library to a Netbeans project. On the left, the Netbeans Project Explorer shows a basic structure for a web application named "WebApplicationDB". The "Libraries" node is selected, revealing options like "Add Project...", "Add Library...", and "Add JAR/Folder...". A context menu is open over the "Add Library..." option. In the center, a modal dialog titled "Add Library" lists various available libraries. The "Java DB Driver" item is highlighted with a blue selection bar. On the right, the final state of the project structure is shown after the addition, with the "Java DB Driver - derby.jar" file now listed under the "Libraries" node.

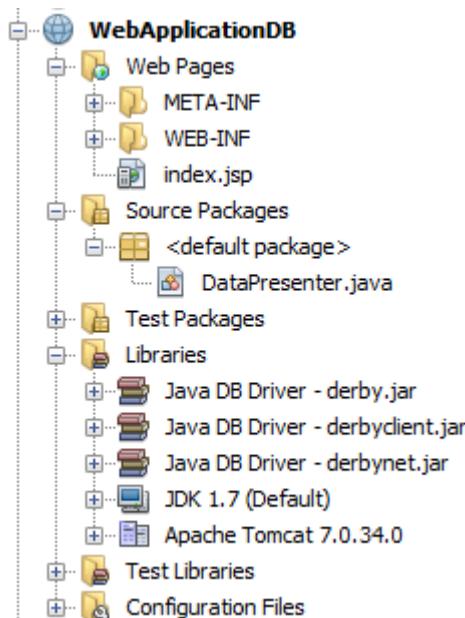
Available Libraries:

- CopyLibs Task
- EclipseLink (JPA 2.1)
- EclipseLink from GlassFish
- EclipseLink-ModelGen (JPA 2.1)
- Groovy 2.0.1
- Hibernate
- Hibernate JPA
- Hibernate JPA Modelgen
- Jakarta Slide Ant WebDAV
- Java DB Driver**
- Java EE 6 API Library
- Java EE 6 Endorsed API Library
- Java EE 7 API Library
- Java EE 7 Endorsed API Library
- Java EE from GlassFish
- Java EE Web 6 API Library

# Java DB (Apache Derby)



## Hola Mundo

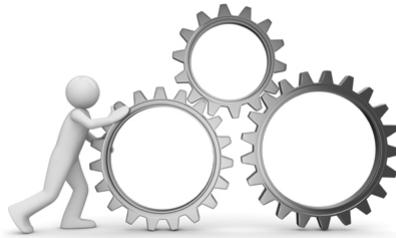


WebApplicationDB  
Java DB

A screenshot of a Mozilla Firefox browser window titled "JSP Page - Mozilla Firefox". The address bar shows "localhost:8085/WebApplicationDB/". The main content area displays the text "Hello World of Databases!" in large, bold, black font. Below it is a "Show data" button.

A screenshot of a Mozilla Firefox browser window titled "Servlet DataPresenter - Mozilla Firefox". The address bar shows "localhost:8085/WebApplicationDB/DataPresenter". The main content area displays the text "Data from database" in large, bold, black font. Below it, a list of database records is shown:

ID	Name	Balance
1	Pedro	1234.0
2	Juan	3355.0
3	Joaquin	323.32



# Instrucciones SQL de utilidad

```
Statement query = con.createStatement();
```

```
query.executeUpdate("INSERT INTO CUSTOMERS  
VALUES (4, 'Jose', 424.0)");
```

```
query.executeUpdate("UPDATE CUSTOMERS  
SET balance=balance*2 where balance<400");
```

```
query.executeUpdate("DELETE FROM CUSTOMERS where ID=4");
```



WebApplicationDBBasics  
Java DB

# Práctica de Laboratorio

## Staff

Id: 1 Name: Francisco Balance: 123.0

Id: 2 Name: Jose Luis Balance: 456.0

### Add a record

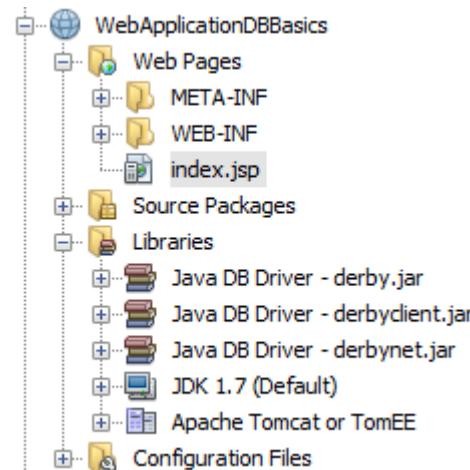
<input type="text" value="Id:"/>	<input type="text" value="Name:"/>	<input type="text" value="Balance:"/>
<input type="button" value="Ok"/>		

### Delete a record

<input type="text" value="Id:"/>
<input type="button" value="Ok"/>

### Update a record

<input type="text" value="Id:"/>	<input type="text" value="Name:"/>	<input type="text" value="Balance:"/>
<input type="button" value="Ok"/>		



**Tip 1:** Poner nombres a los botones  
if (request.getParameter("add")!=null)

**Tip 2:** Usar 3 formas

**Sugerencia:** Crear una BD nueva

# Práctica de Laboratorio



**Tip:** Usar sesiones HTTP para almacenar el ResultSet.

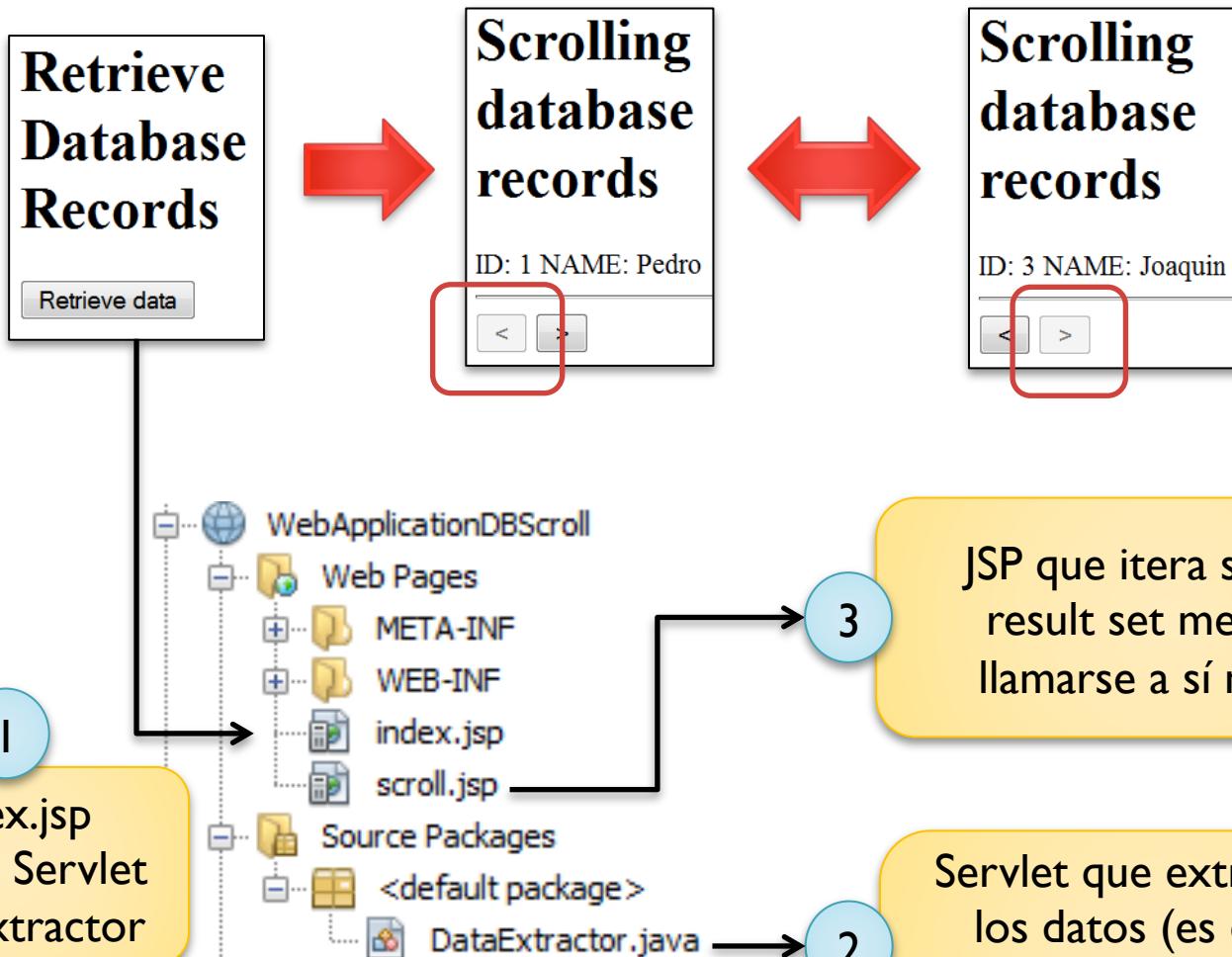
## Información adicional

```
Statement st = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,  
        ResultSet.CONCUR_READ_ONLY);
```

```
ResultSet rs = st.executeQuery("SELECT * FROM customers");
```

```
rs.previous() rs.first() rs.isFirst() rs.last(); rs.isLast() rs.next(); rs.absolute(int row);  
Rs.getRow() rs.getColumnCount()
```

# Práctica de Laboratorio



I  
index.jsp  
llama a Servlet  
DataExtractor

JSP que itera sobre el  
result set mediante  
llamarse a sí mismo.

Servlet que extrae todos  
los datos (es decir, el  
result set), los guarda en  
una sesión HTTP y llama a  
scroll.jsp



WebApplicationDBScroll  
Java DB

# Práctica de Laboratorio

```
private Object[][] ResultSetToArray(ResultSet rs) {  
    Object data[][]=null;  
    try{  
        rs.last();  
        ResultSetMetaData rsmd = rs.getMetaData();  
        int numCols = rsmd.getColumnCount();  
        int numRows =rs.getRow();  
        data=new Object[numRows][numCols];  
        int j = 0;  
        rs.beforeFirst();  
        while (rs.next()){  
            for (int i=0;i<numCols;i++){  
                data[j][i]=rs.getObject(i+1);  
            }  
            j++;  
        }  
    } catch(Exception e){  
        System.out.println(e);  
    }  
    return data;  
}
```

Código que convierte un result set a un arreglo bidimensional de objetos

```
out.println("ID: "+ myResultSet[row][0]);  
out.println("NAME: "+ myResultSet[row][1]);
```