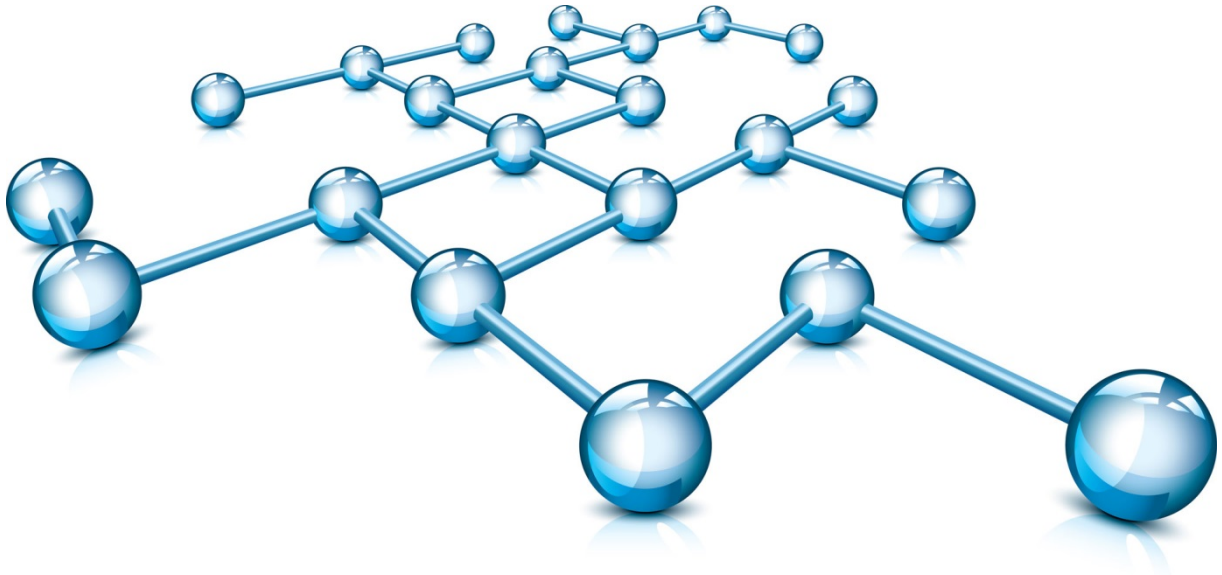
Componentes Web de Java EE

Profesor:
Dr. J. Octavio Gutiérrez García

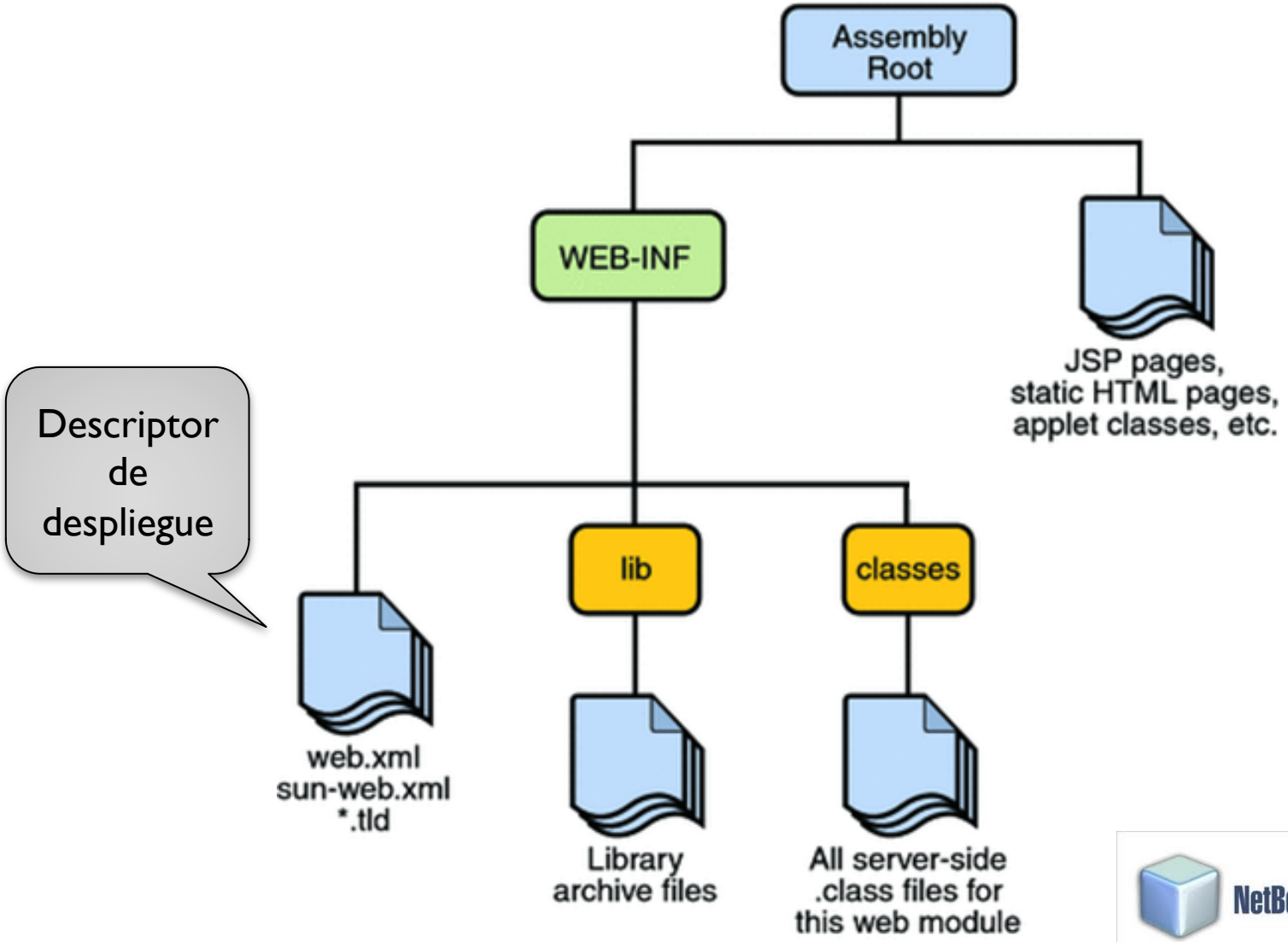
octavio.gutierrez@itam.mx



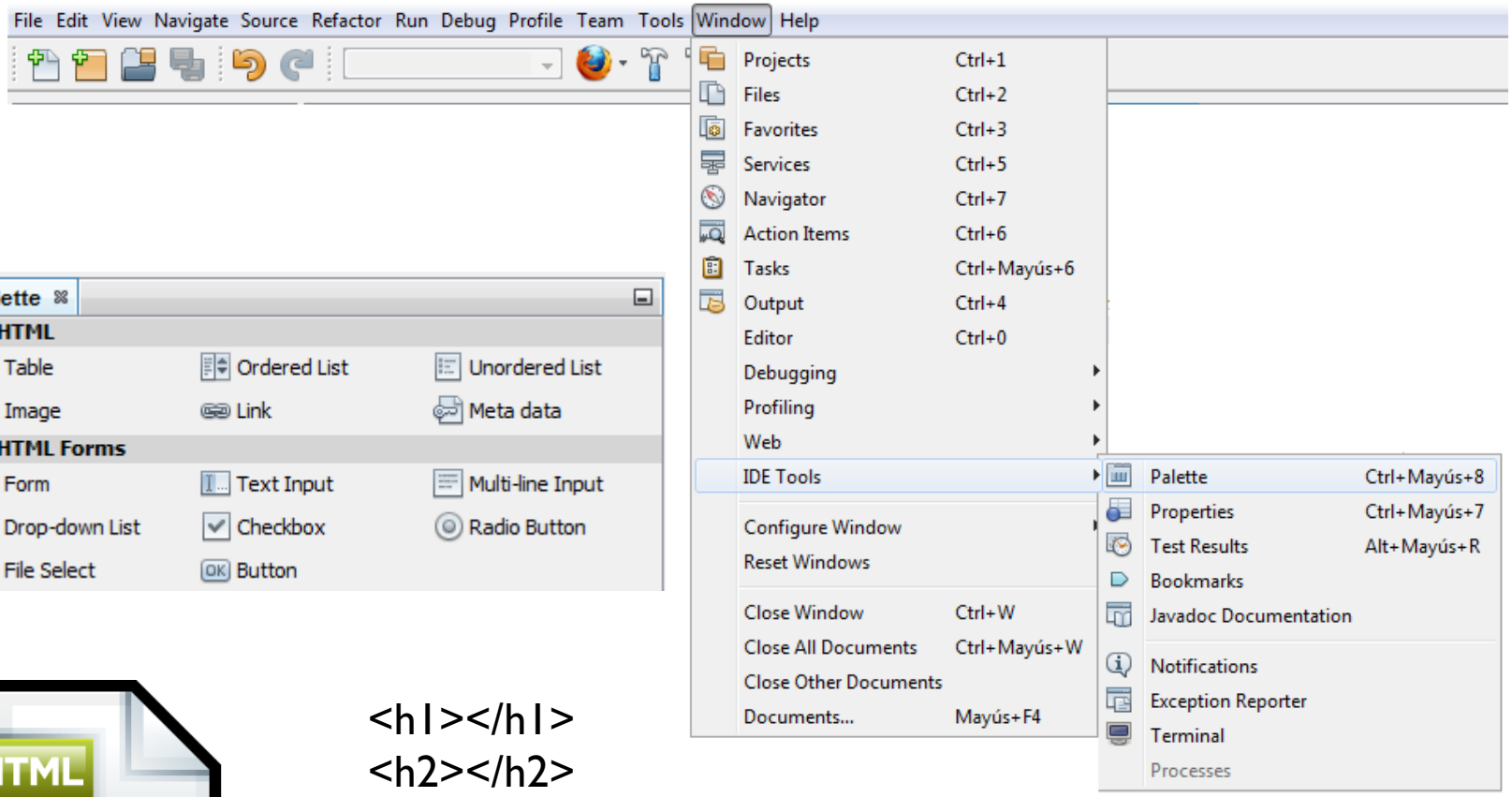
Antes de empezar…

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

Estructura de un módulo web



Jugando con HTML en Netbeans




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

Práctica de HTML

WebApplicationHTML

¡Mi primera forma!

Descripción	Valor
Apellido paterno:	
Apellido materno:	
Género	Hombre Mujer
Región del país	Norte
¿Tiene todas las vacunas?	
Comentarios adicionales	
Foto:	
Tres países donde me gustaría vivir	A. Uruguay B. Turquía C. Marruecos
Mi sitio web favorito es:	The Chronicle of Higher Education
Limpiar	Enviar

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

Página de bienvenida

web.xml

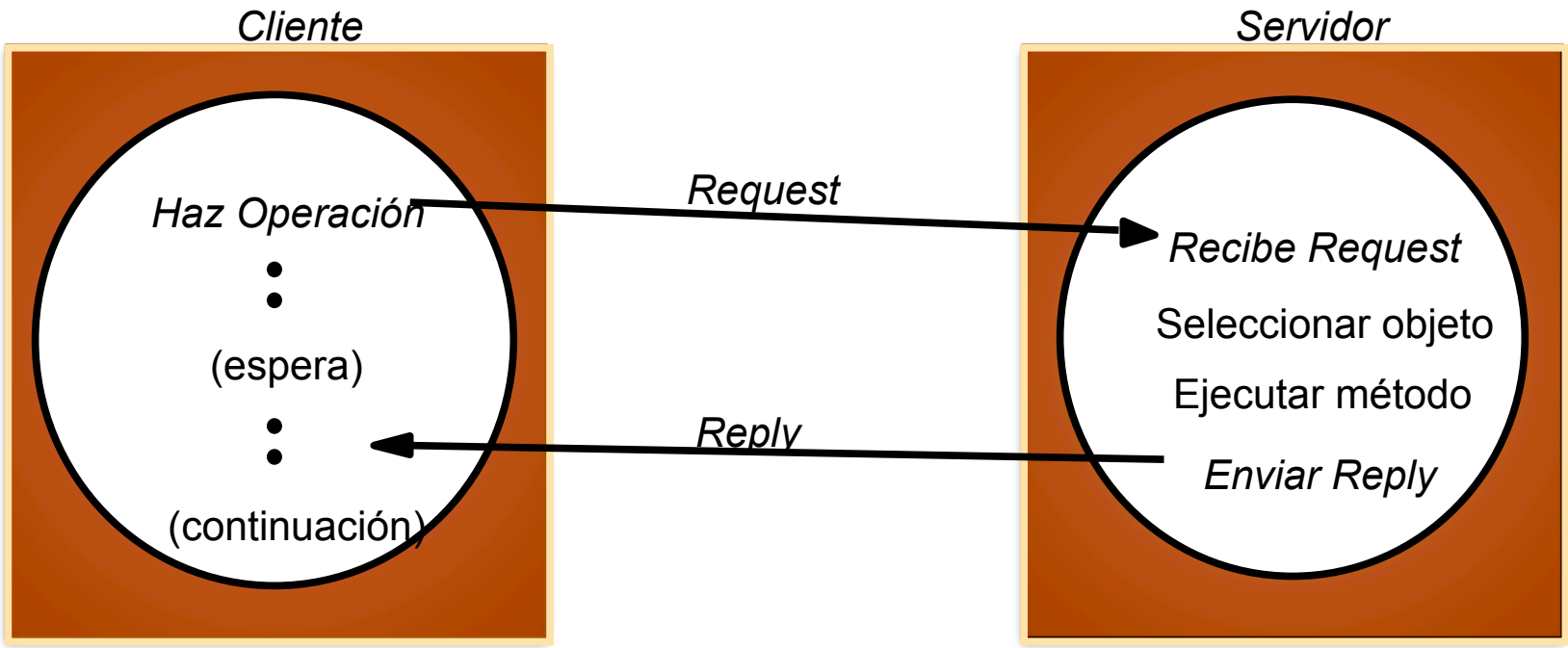
```
<web-app>
…
        <welcome-file-list>
                <welcome-file>mypage1.jsp</welcome-file>
                <welcome-file>mypage2.jsp</welcome-file>
….
        </welcome-file-list>
…
</web-app>
```

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

HyperText
Transfer
Protocol

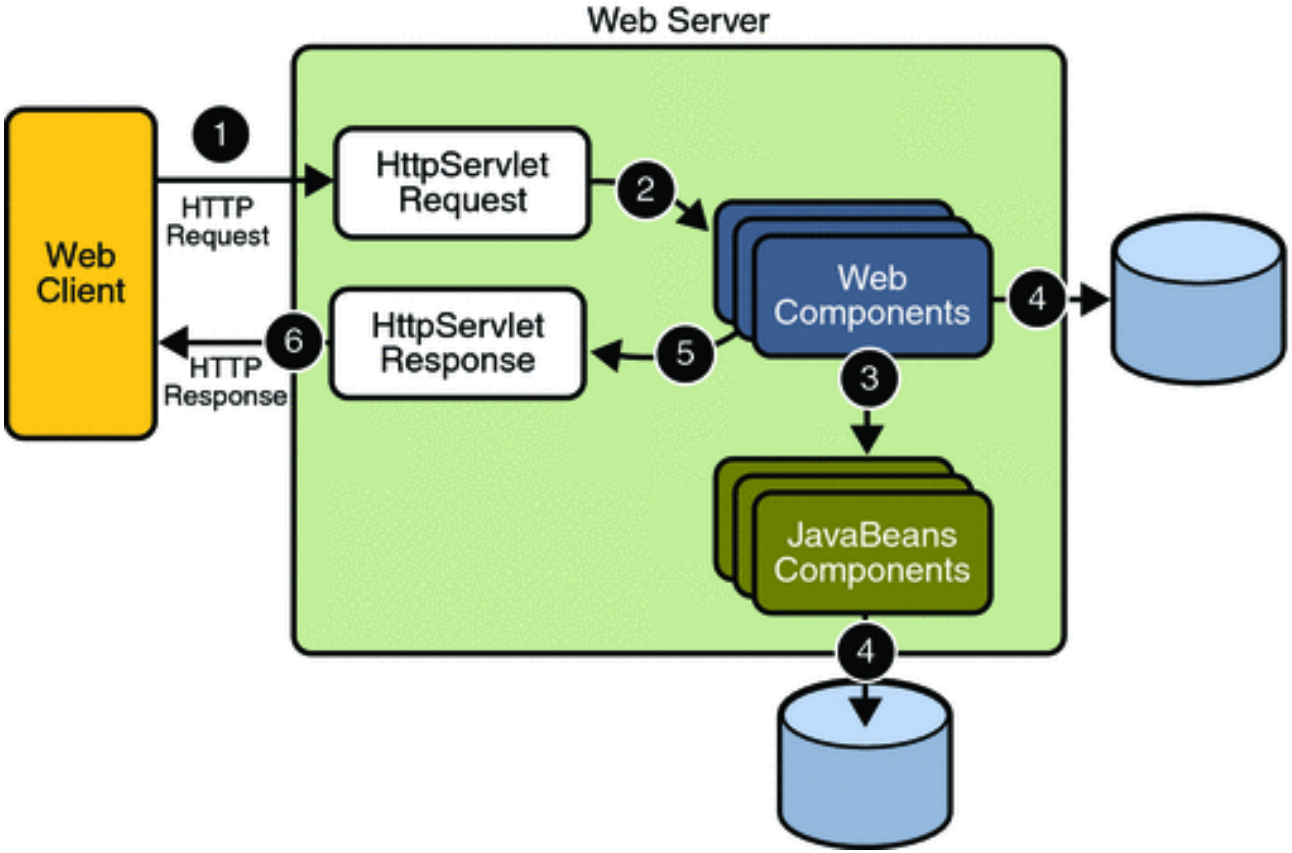
Protocolos request-reply

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><URIsolicitado><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

Scriplet
Expresiones de java embebidas en páginas JSP

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

        }
%>
```

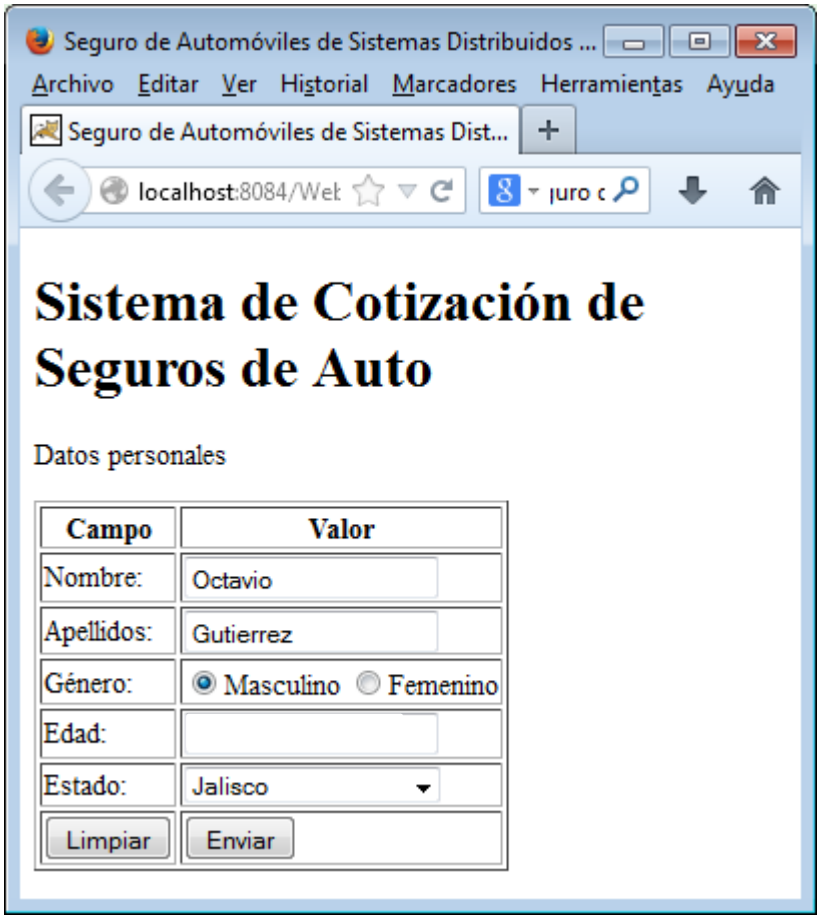
Post

VS

Get

Práctica de laboratorio: request-reply

index.jsp



NetBeans

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/Web

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/Web

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/Web

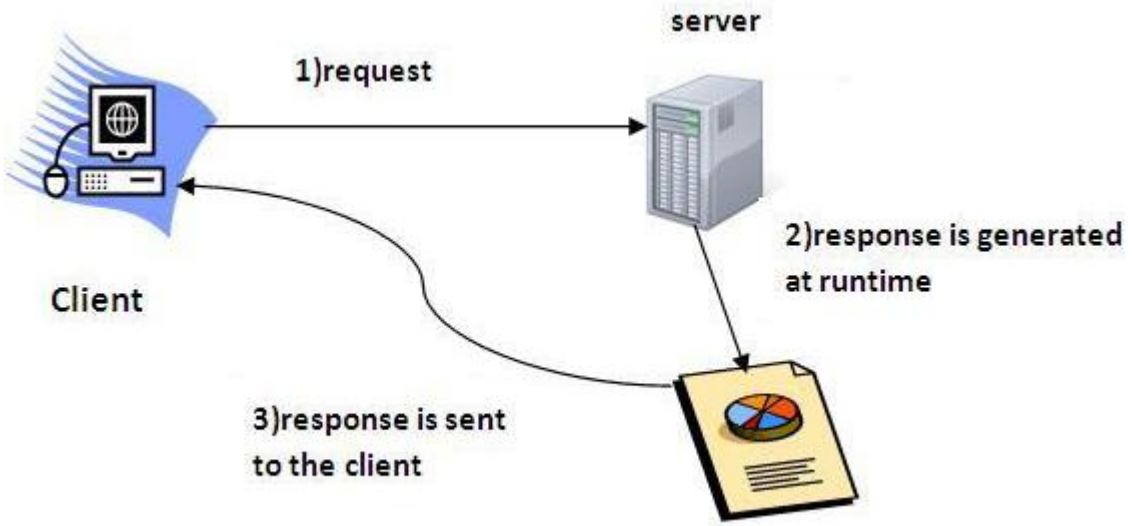
Cotización de seguro para Automóvil

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:

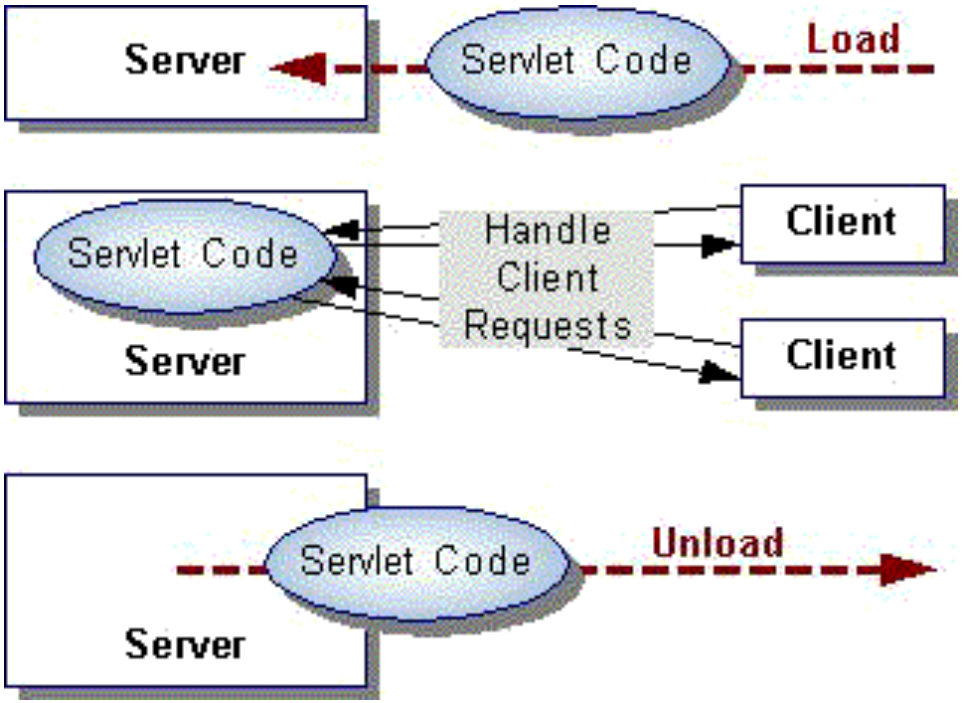
$4,200.00 pesos

Servlets

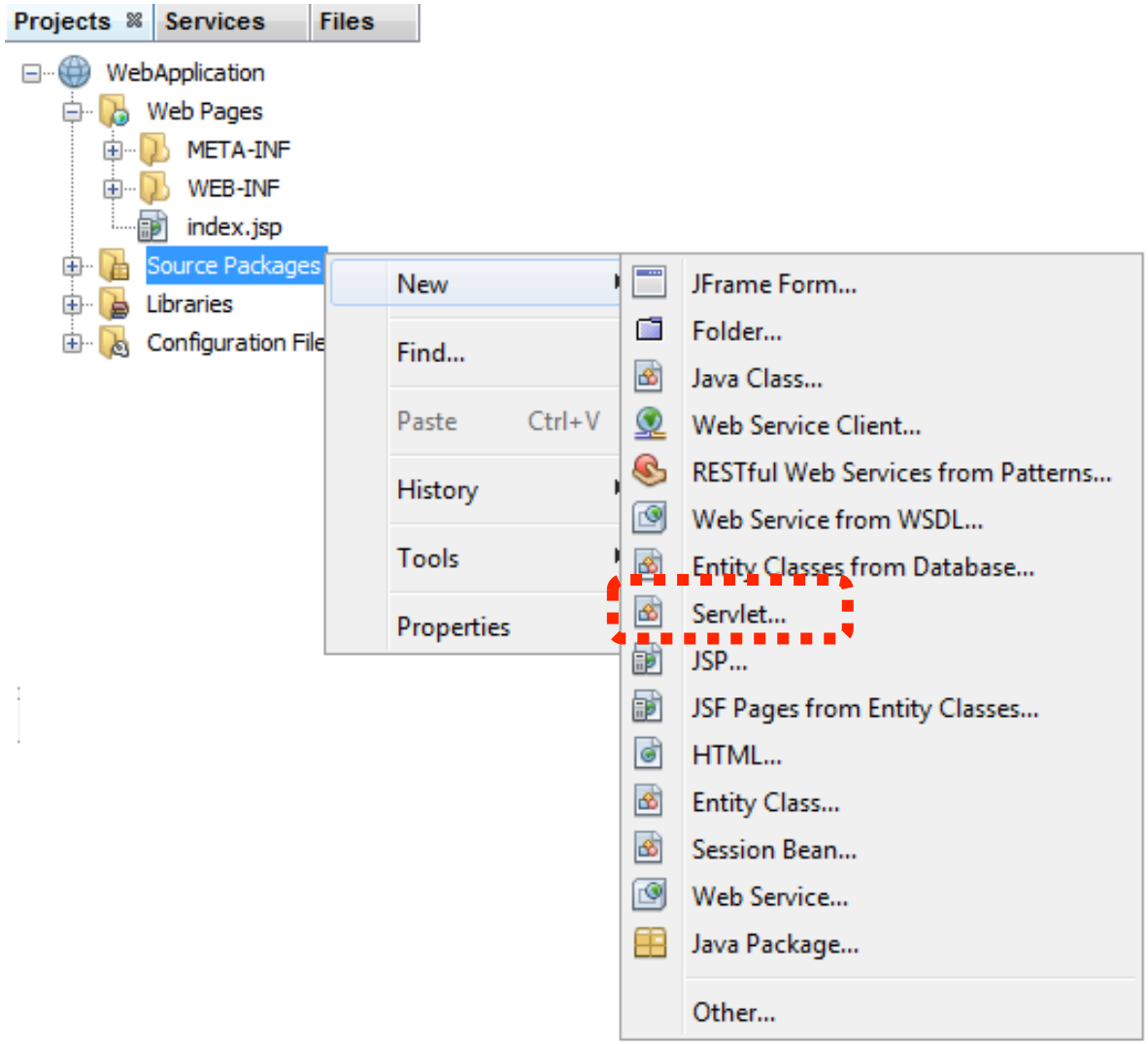
- Un servlet es una clase Java utilizada para ampliar las capacidades de los servidores



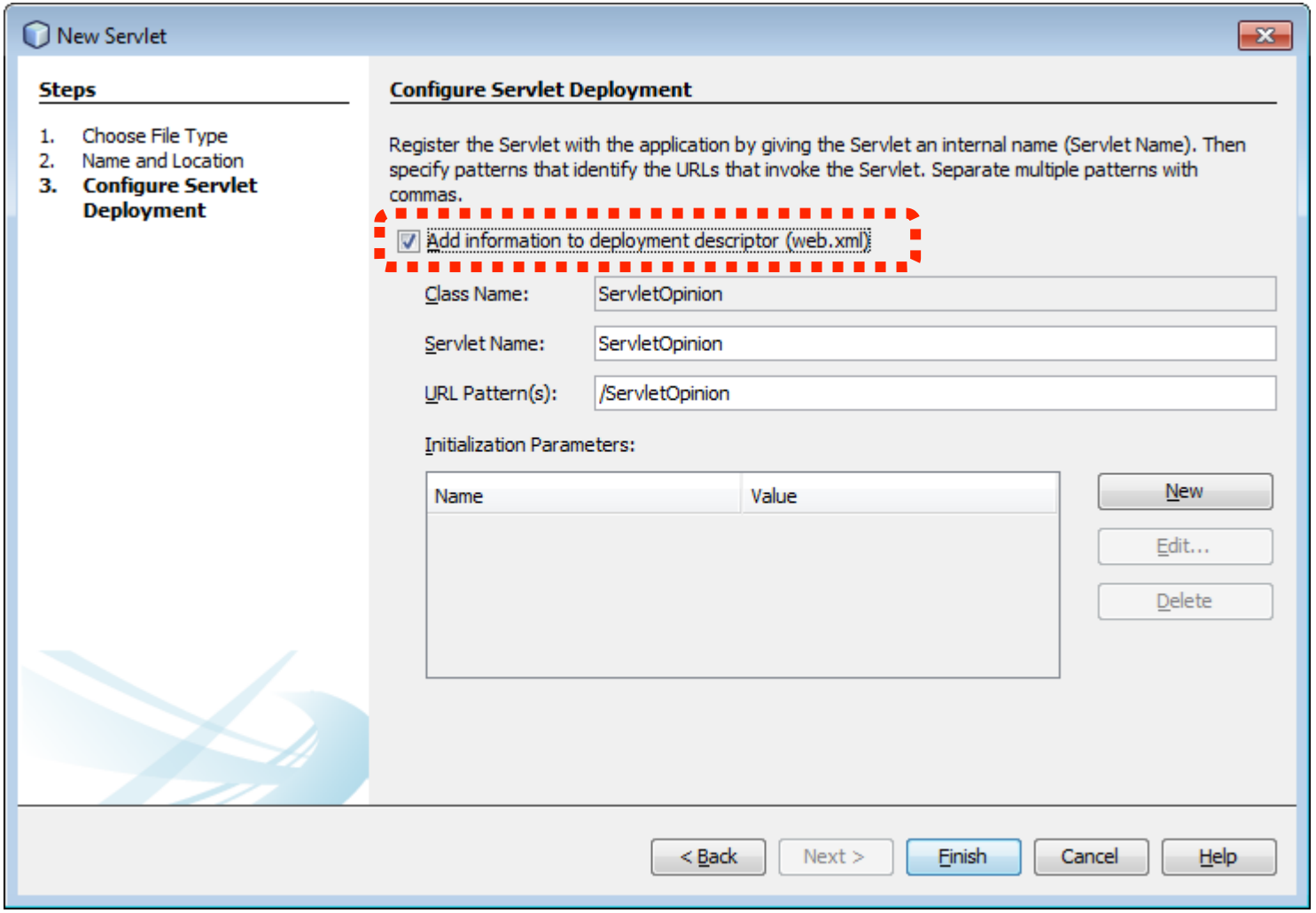
Ciclo de vida de los Servlets



Creación de Servlets en Netbeans



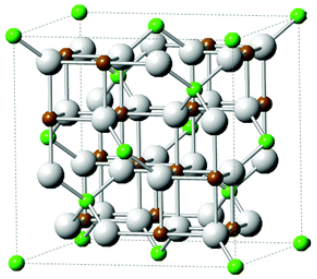
Creación de Servlets en Netbeans



Servlets

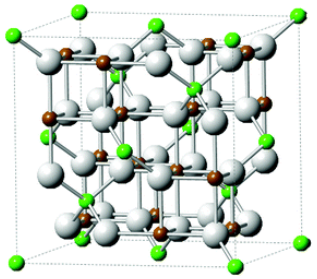
- Clases involucradas
  - java.io.IOException
  - java.io.PrintWriter
  - javax.servlet.ServletException
  - javax.servlet.http.HttpServlet
  - javax.servlet.http.HttpServletRequest
  - javax.servlet.http.HttpServletResponse

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

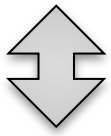
Estructura de un Servlet

```
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>
}
```

¿Cómo utilizo un 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



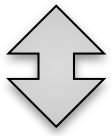
Servlets

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



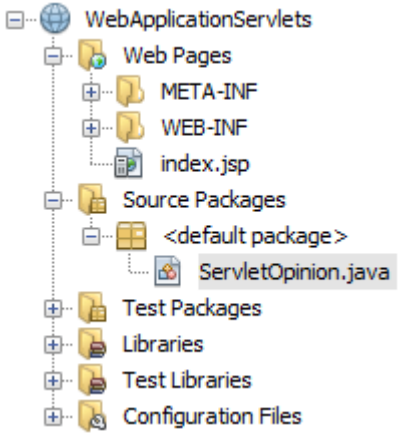
Servlets

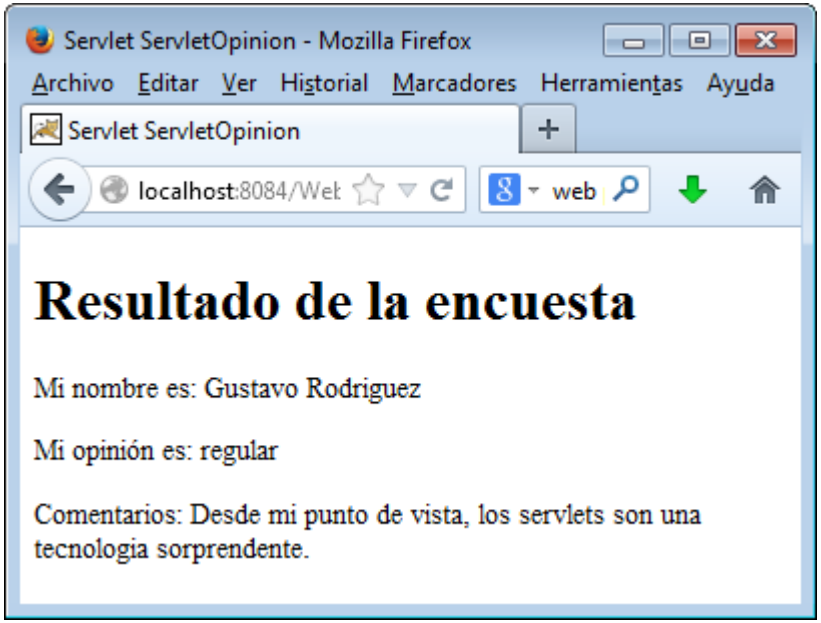
Más adecuado para aplicaciones orientadas a servicios (los endpoints de los servicios web se implementan como servlets).

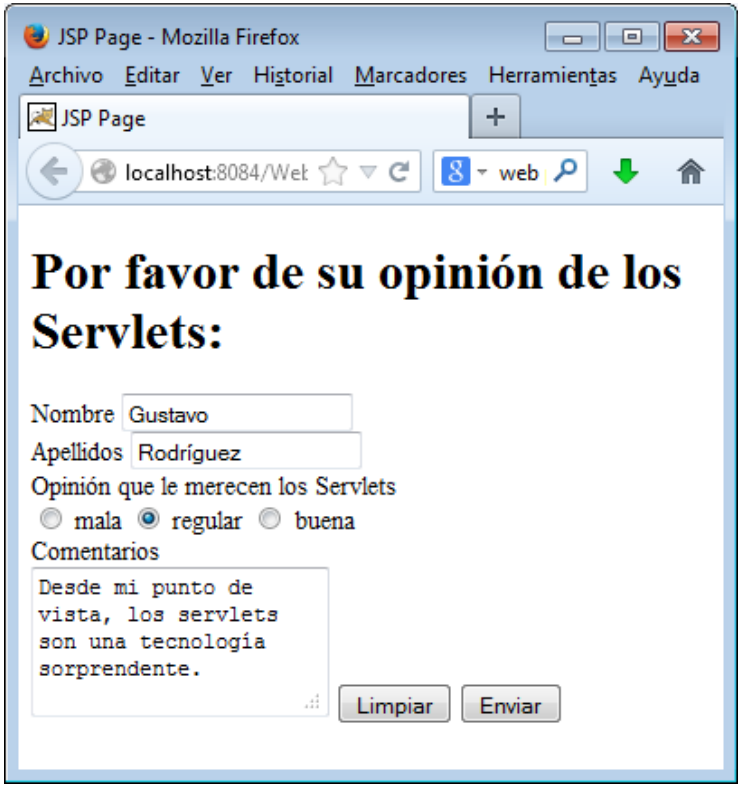
JavaServer Pages

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

Mi primer Servlet







WebApplicationServlets

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

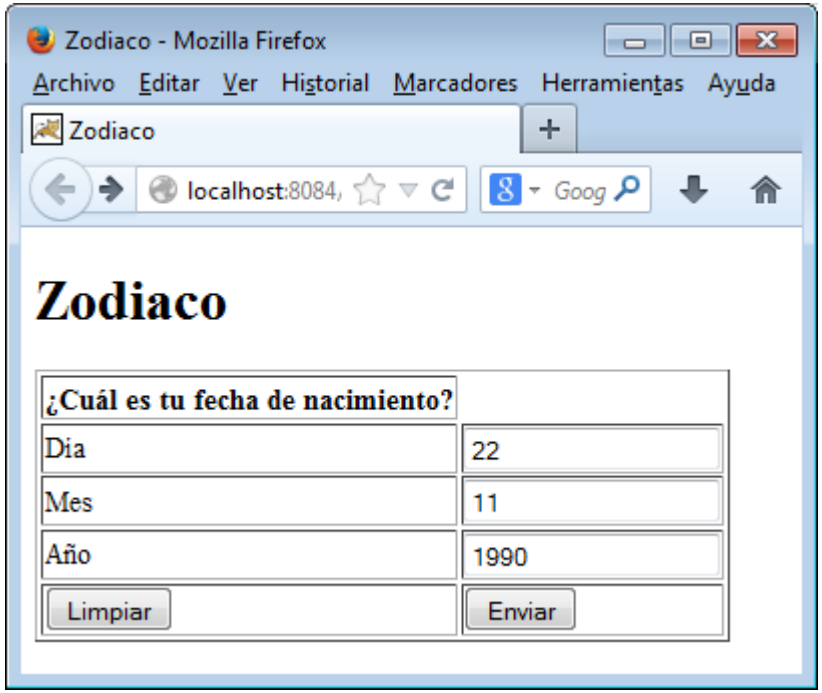
WebApplicationGoogleSearch

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*/);

!Cuidado!
2 equivale a
Marzo

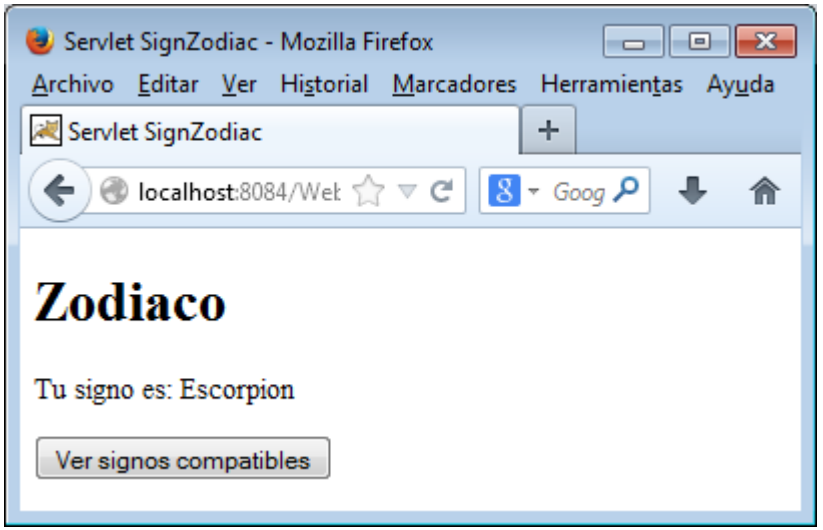
NetBeans

WebApplicationZodiac

Práctica de Servlets: El Zodiaco

SignZodiac Servlet

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

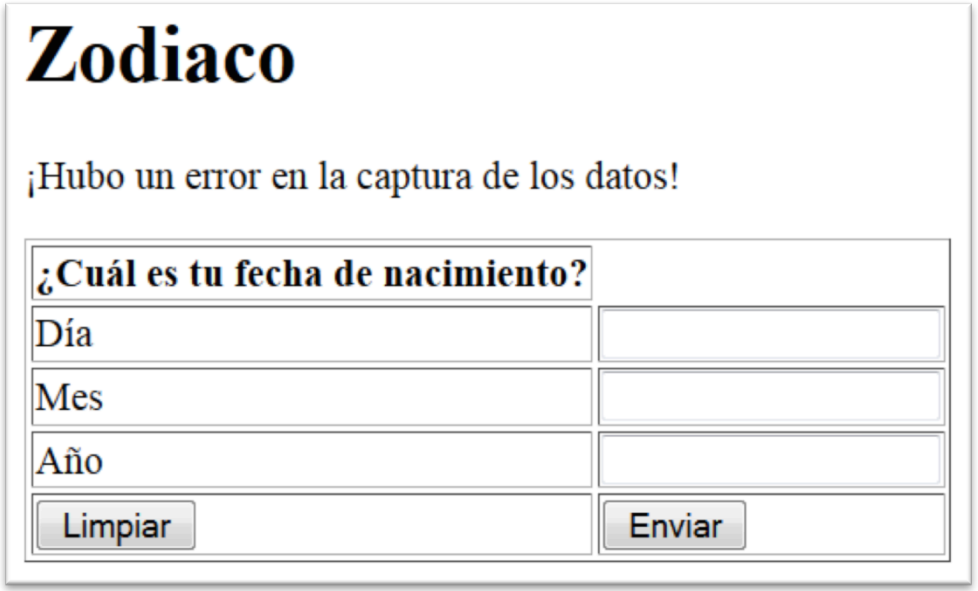


CompatibleSigns Servlet

Práctica de Servlets: El Zodiaco - Versión 1.1

index.jsp

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

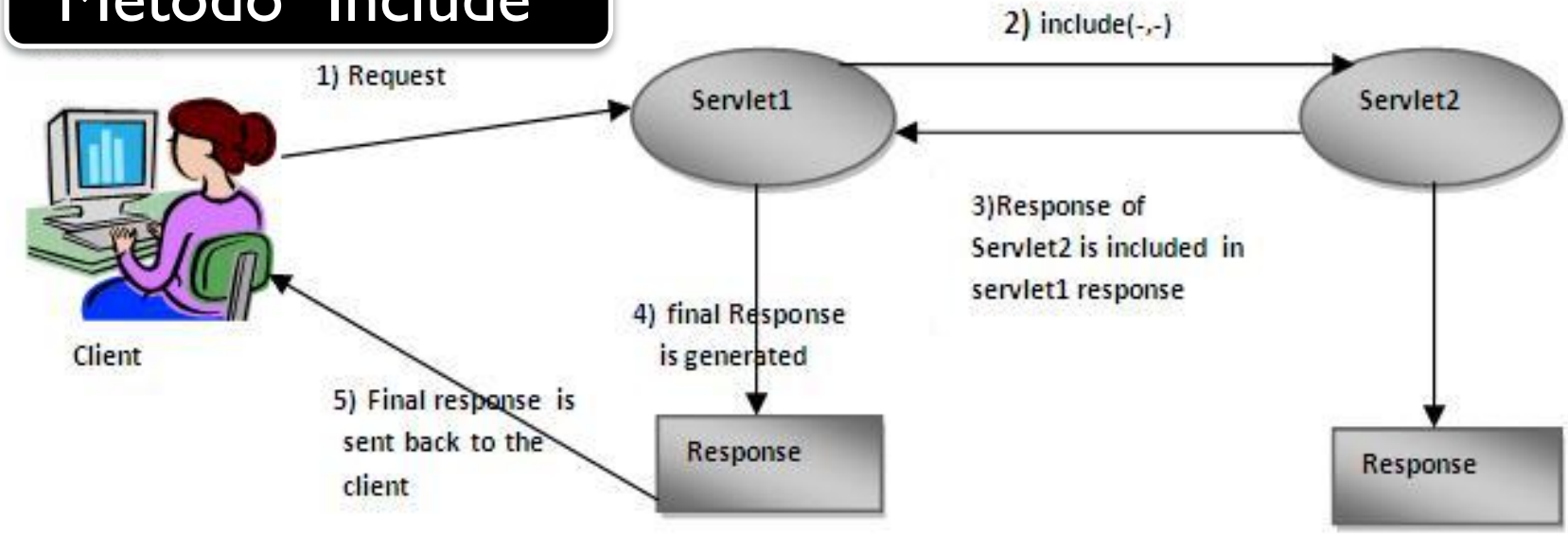


NetBeans

WebApplicationExtendedZodiac

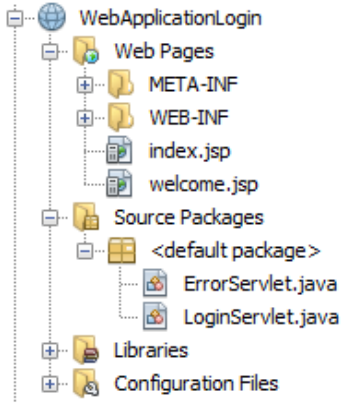
Colaboración entre Servlets

Método "Include"




```
import javax.servlet.RequestDispatcher;

String nextJSP = "/servlet2";
RequestDispatcher dispatcher = getServletContext().getRequestDispatcher(nextJSP);
dispatcher.include(request, response);
```

Colaboración entre Servlets



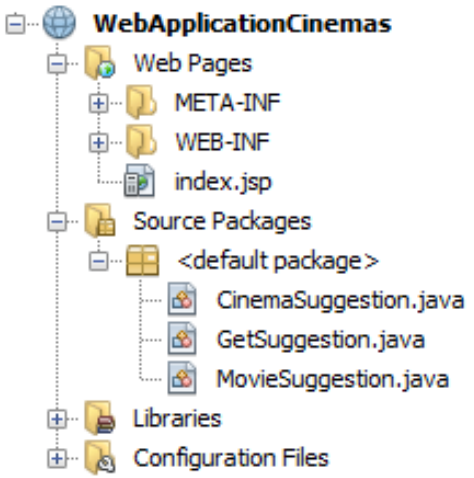
NetBeans

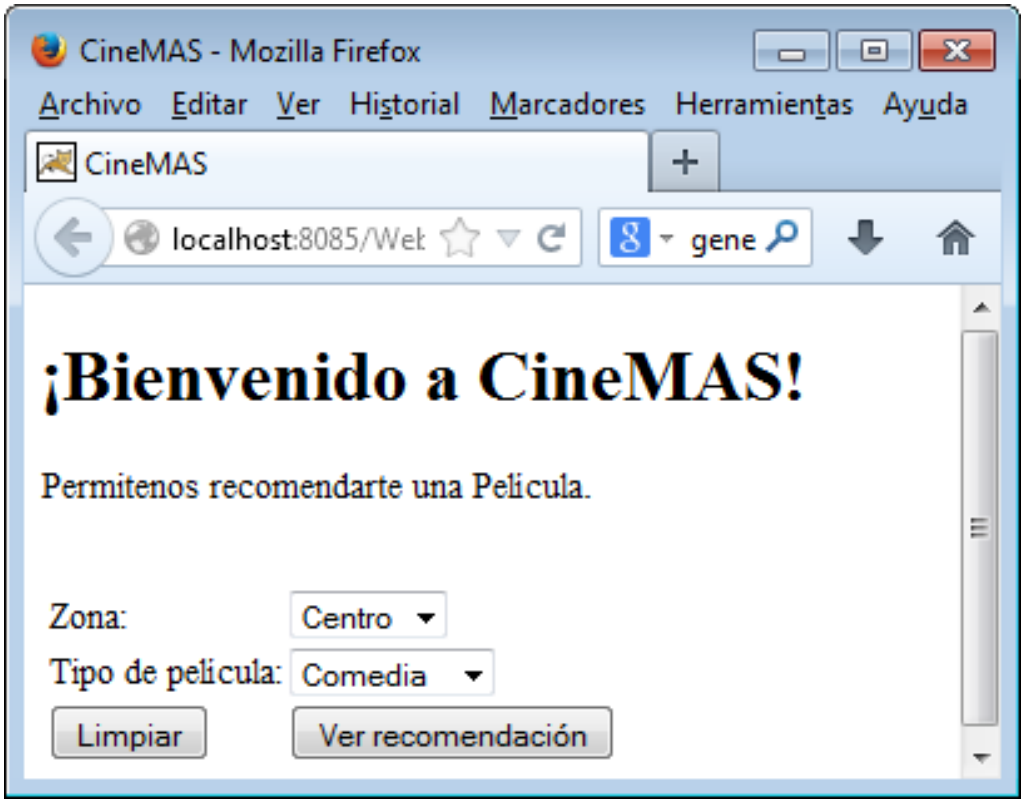
WebApplicationLogin

Práctica
Colaboración entre Servlets

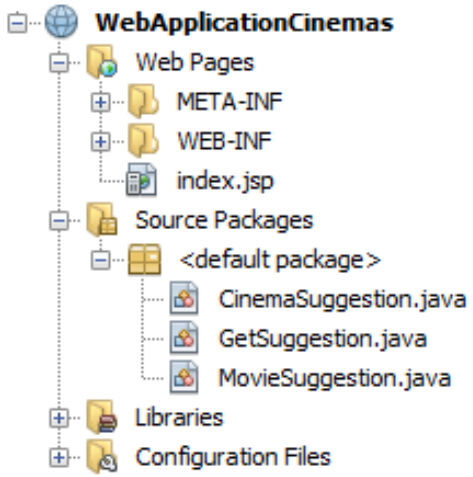
Método "Include"

Index.jsp





Práctica
Colaboración entre Servlets

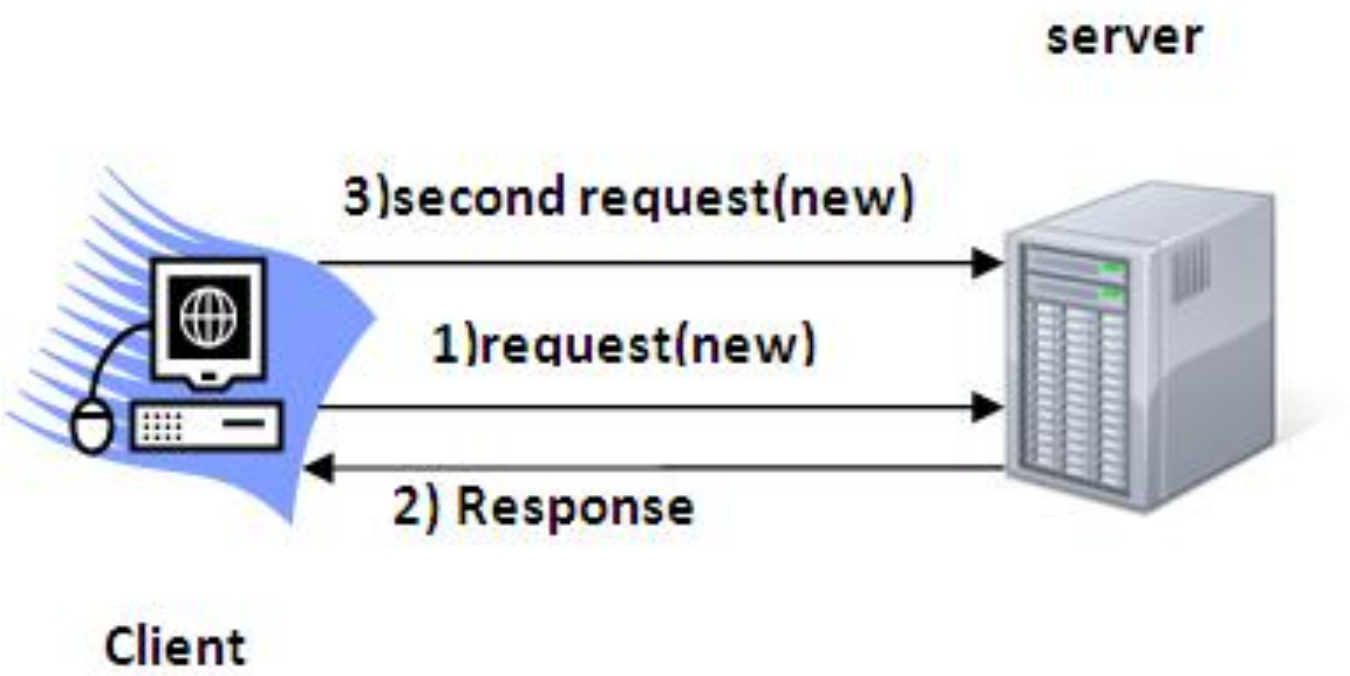


GetSuggestion Servlet

NetBeans

WebApplicationCinemas2

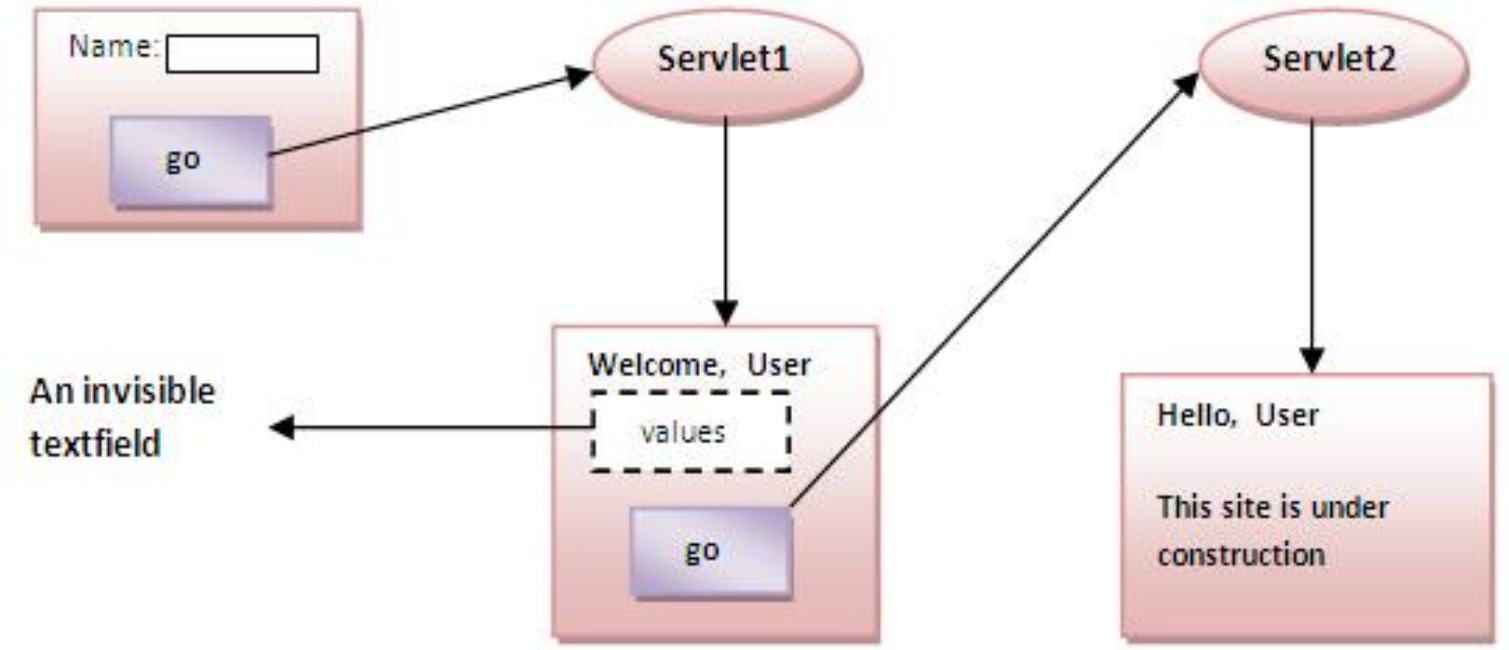
Manejo de Sesiones en Servlets



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='**usrname**' value=""+**n**+"">")

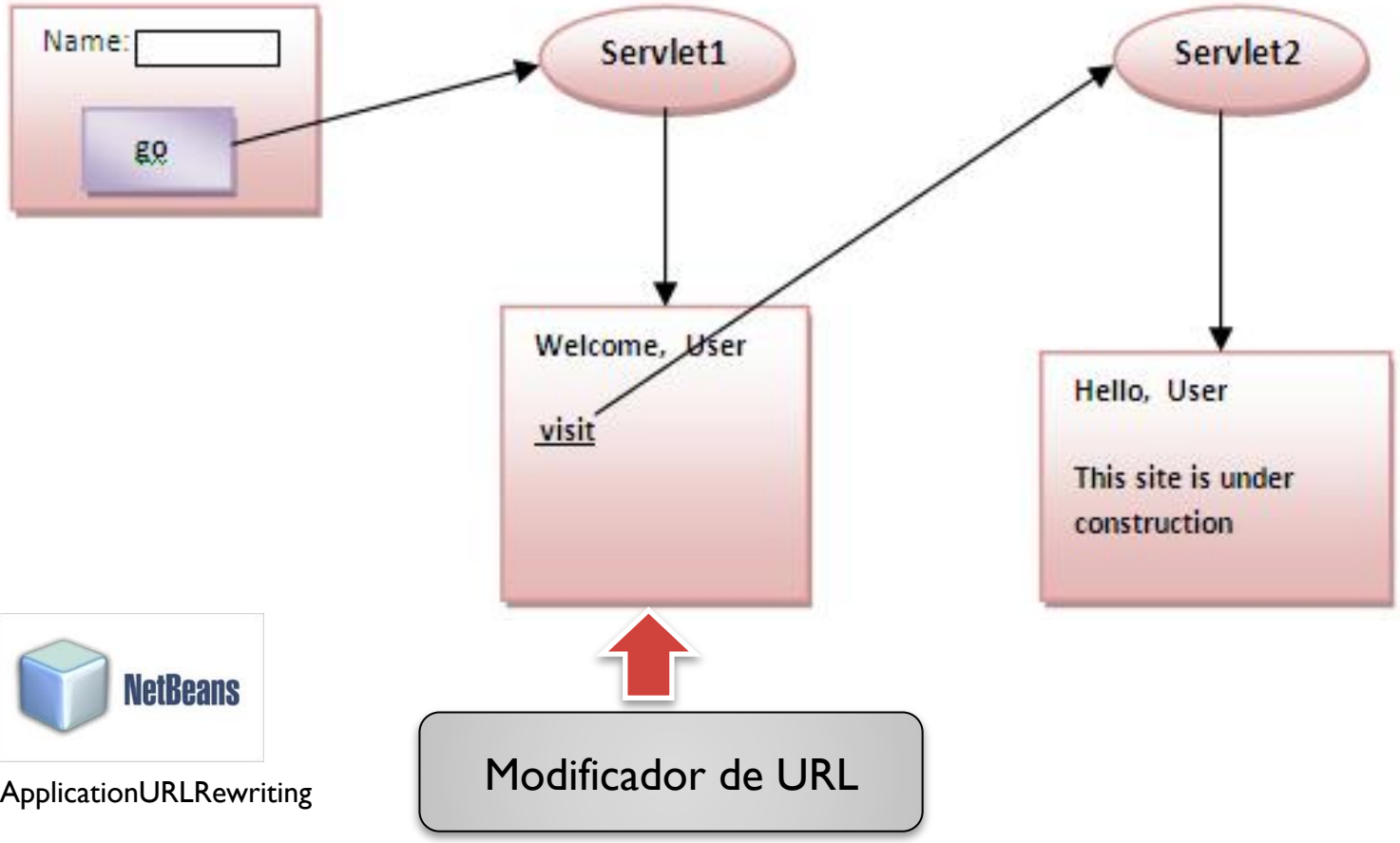
Sesiones con campos ocultos

- 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



NetBeans

WebApplicationURLRewriting

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.

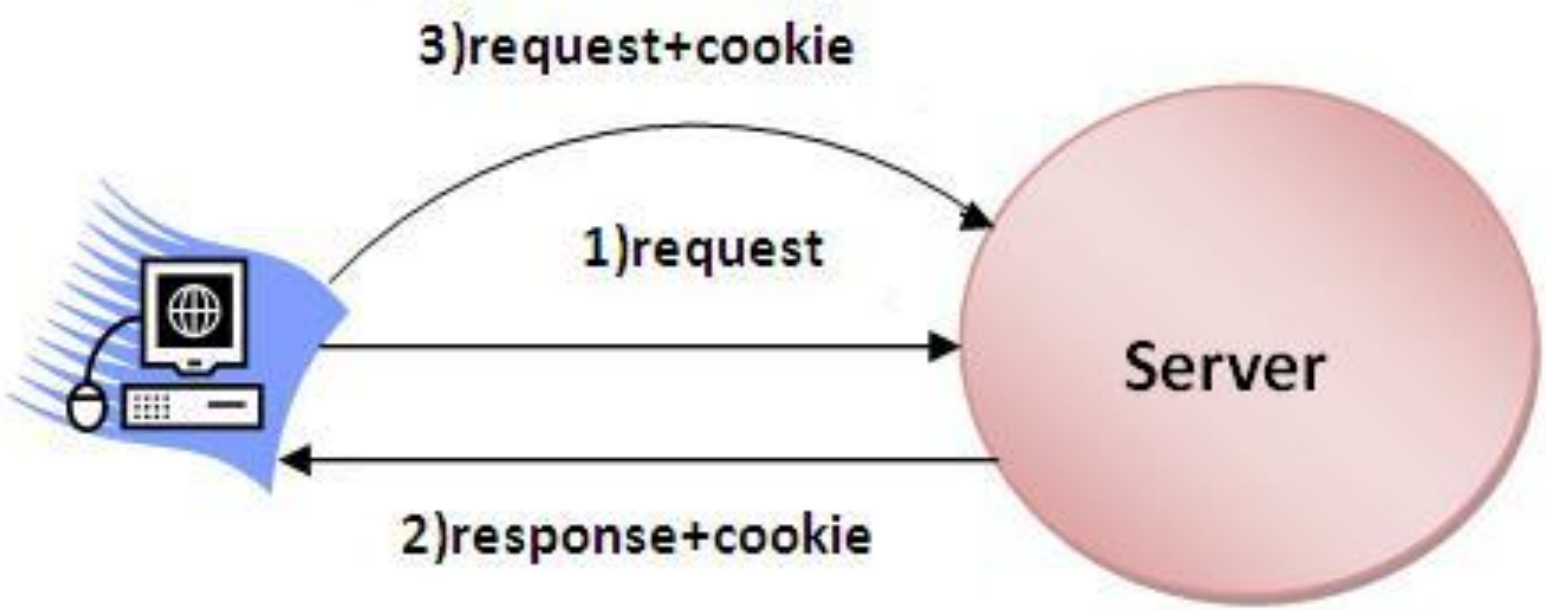
- Desventajas
  - Sólo funciona con **links**.
  - Sólo se puede utilizar texto plano.

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>
```

http://www.w3schools.com/tags/ref_eventattributes.asp

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

¿Cómo crear una Cookie?

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

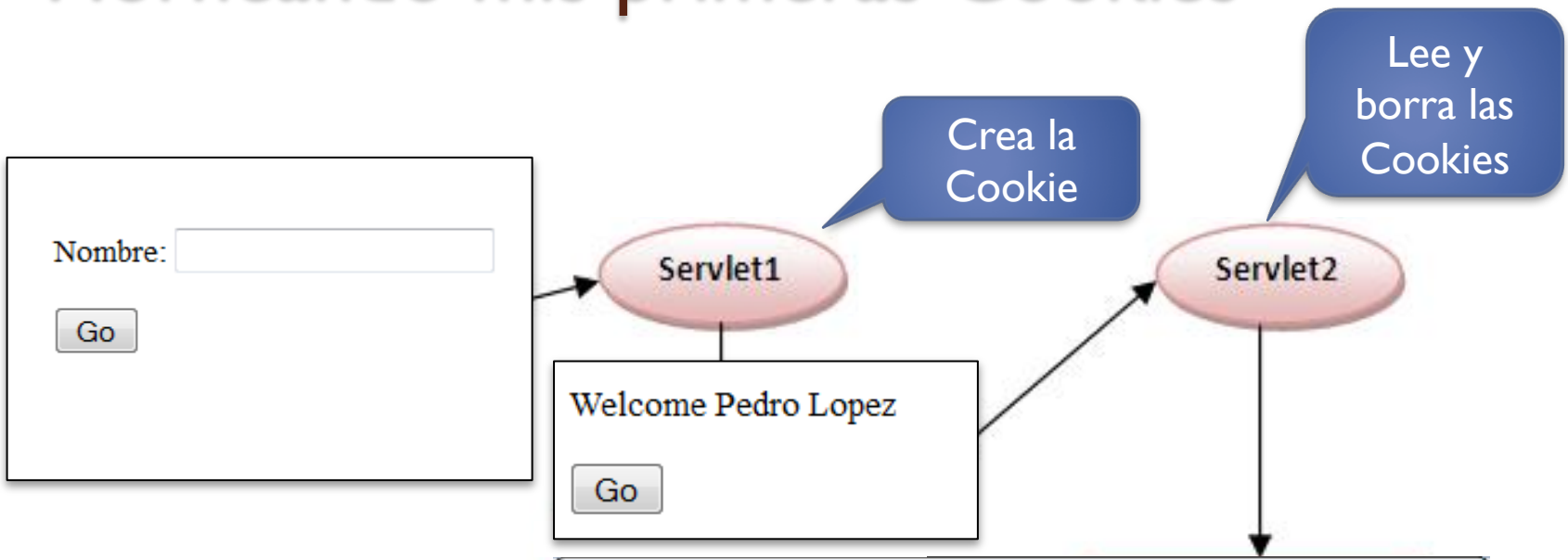
¿Cómo borrar una Cookie?

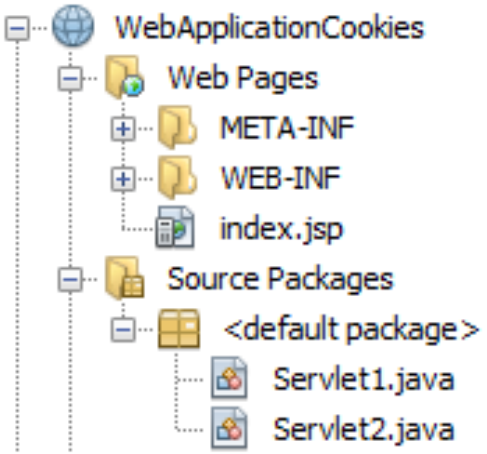
```
Cookie ck=new Cookie("nombre","a");
//borrando el valor de una cookie

ck.setMaxAge(0);
//cambiando la edad máxima a 0 segundos

response.addCookie(ck);
//añadiendo la cookie a la respuesta
```

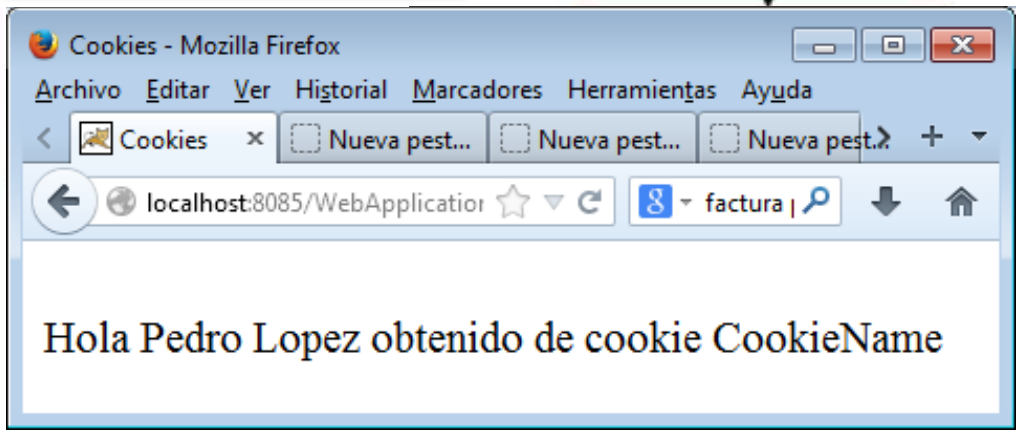
¿Cómo obtener las Cookies?

```
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

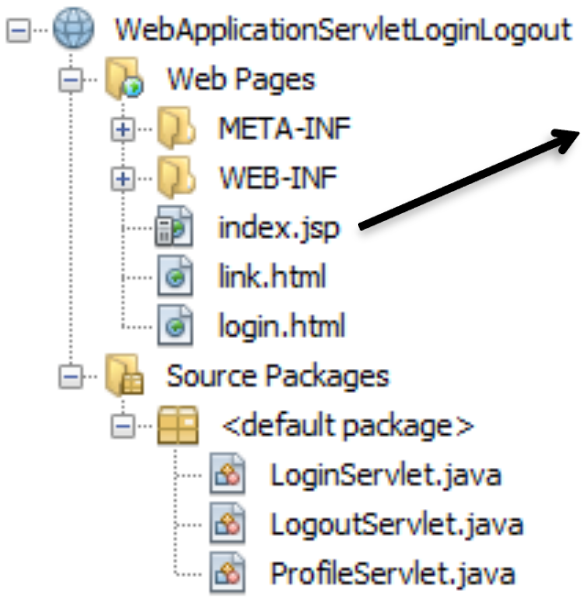
Login

Logout

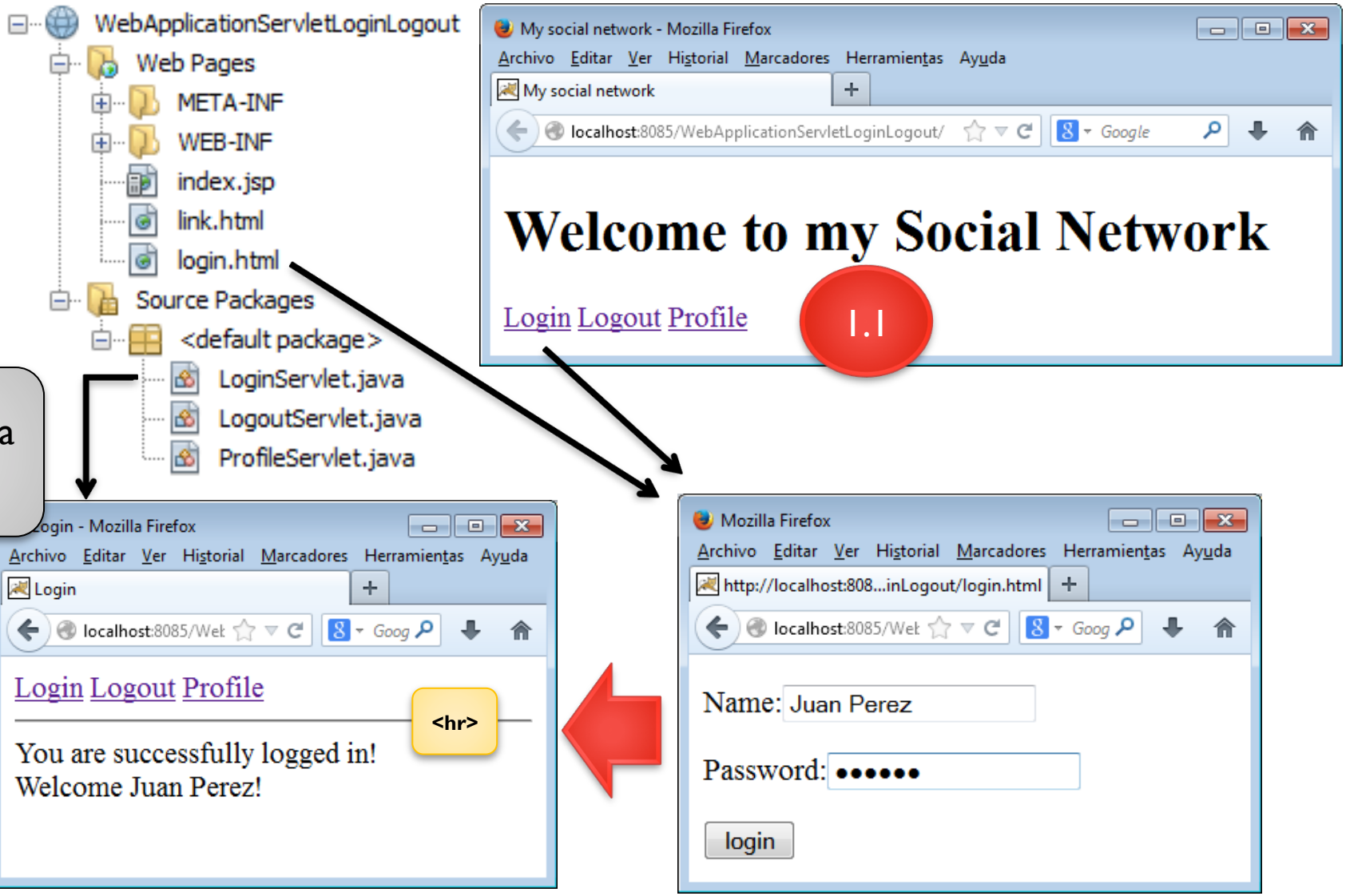
Profile

WebApplicationServletLoginLogout

Práctica de Sesión con Cookies

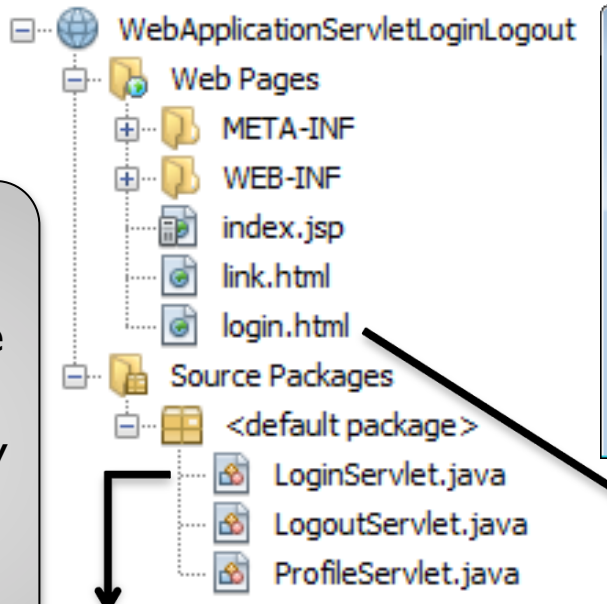


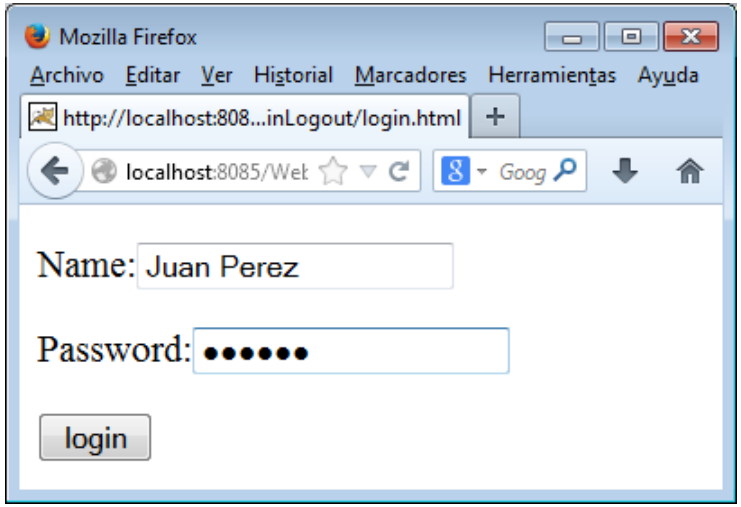
Práctica de Sesión con Cookies



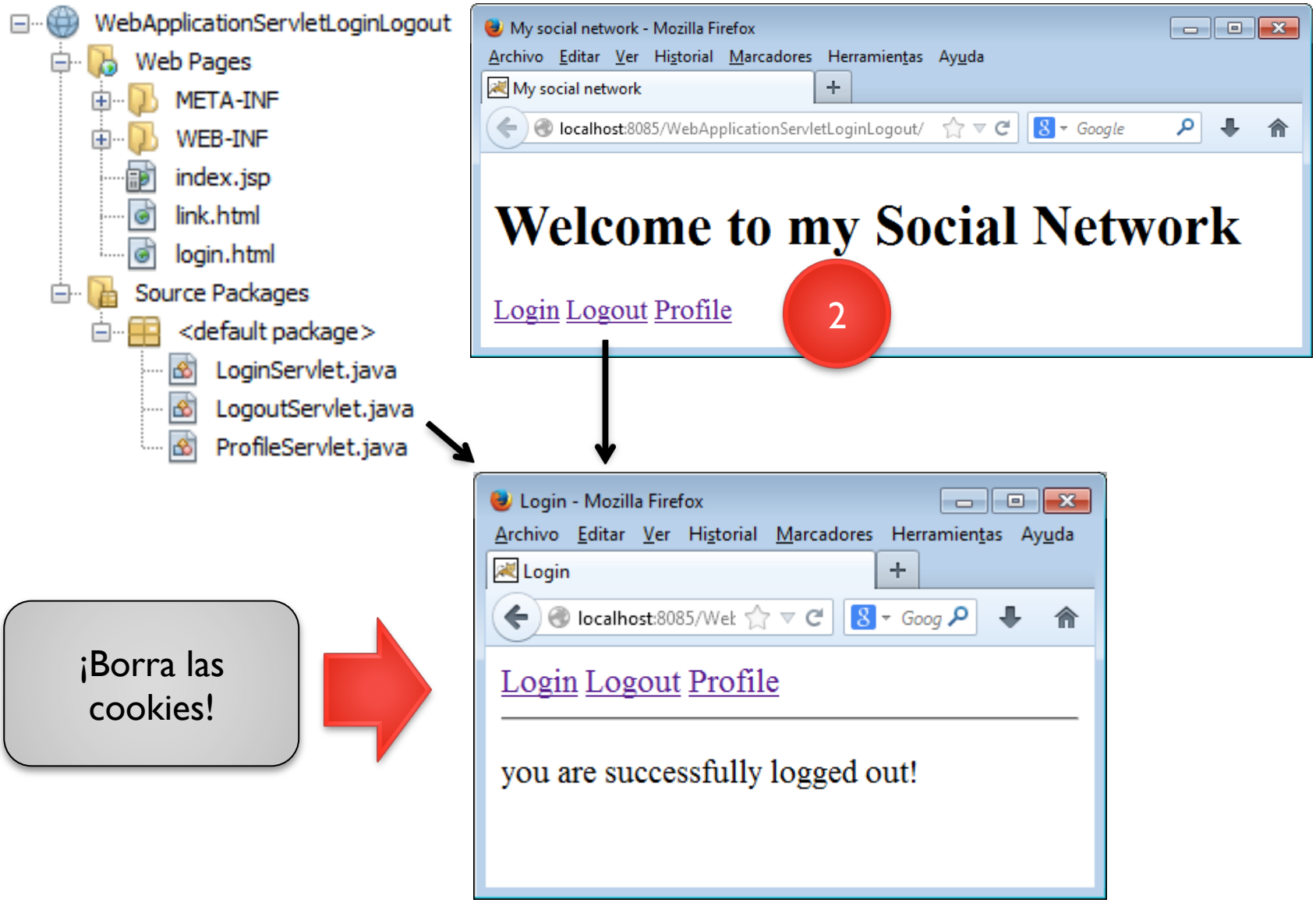
Práctica de Sesión con Cookies

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

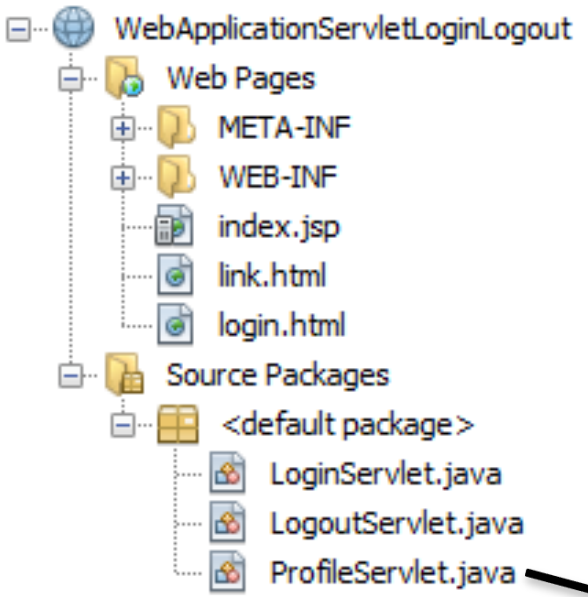


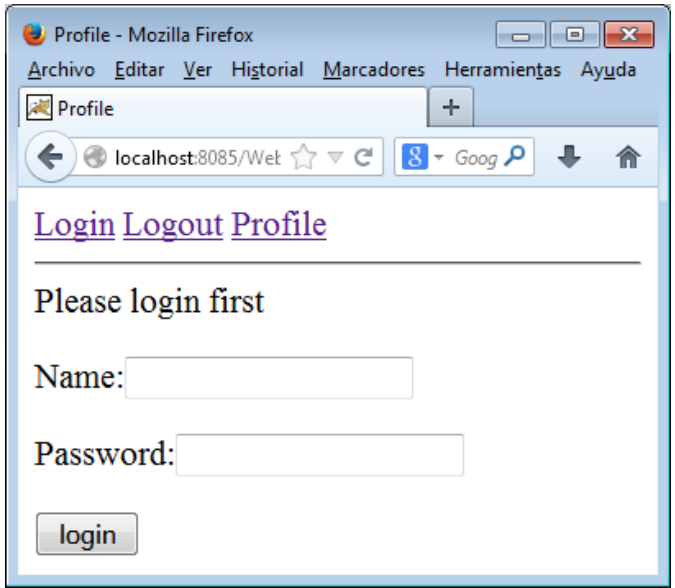


Práctica de Sesión con Cookies



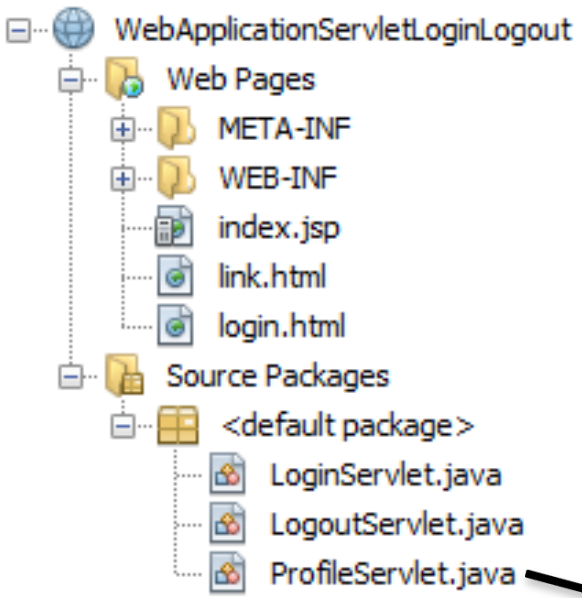
Práctica de Sesión con Cookies





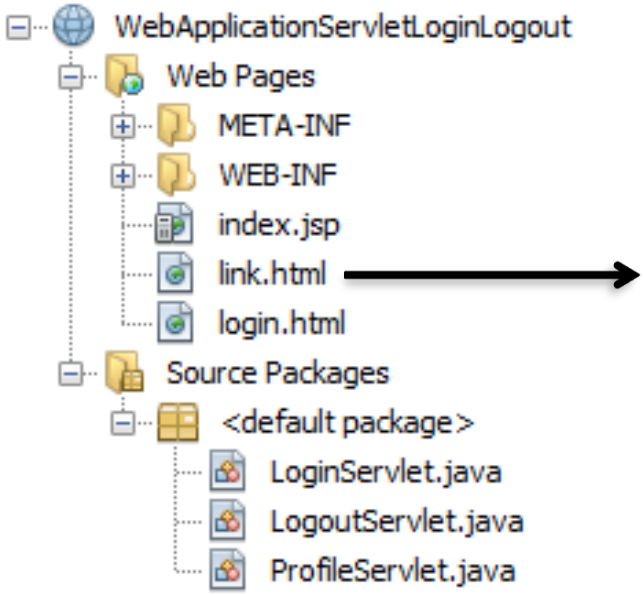
ProfileServlet
Verifica si está una cookie,
Si no está imprime mensaje
e **incluye** "login.html"

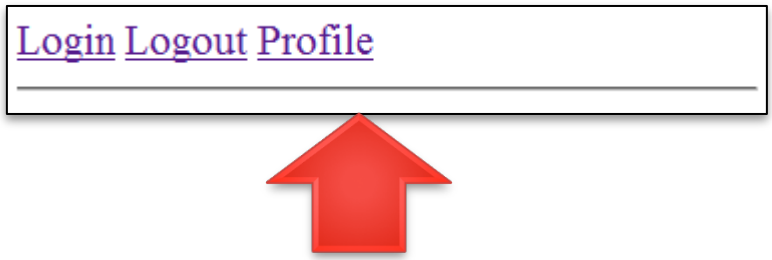
Práctica de Sesión con Cookies



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

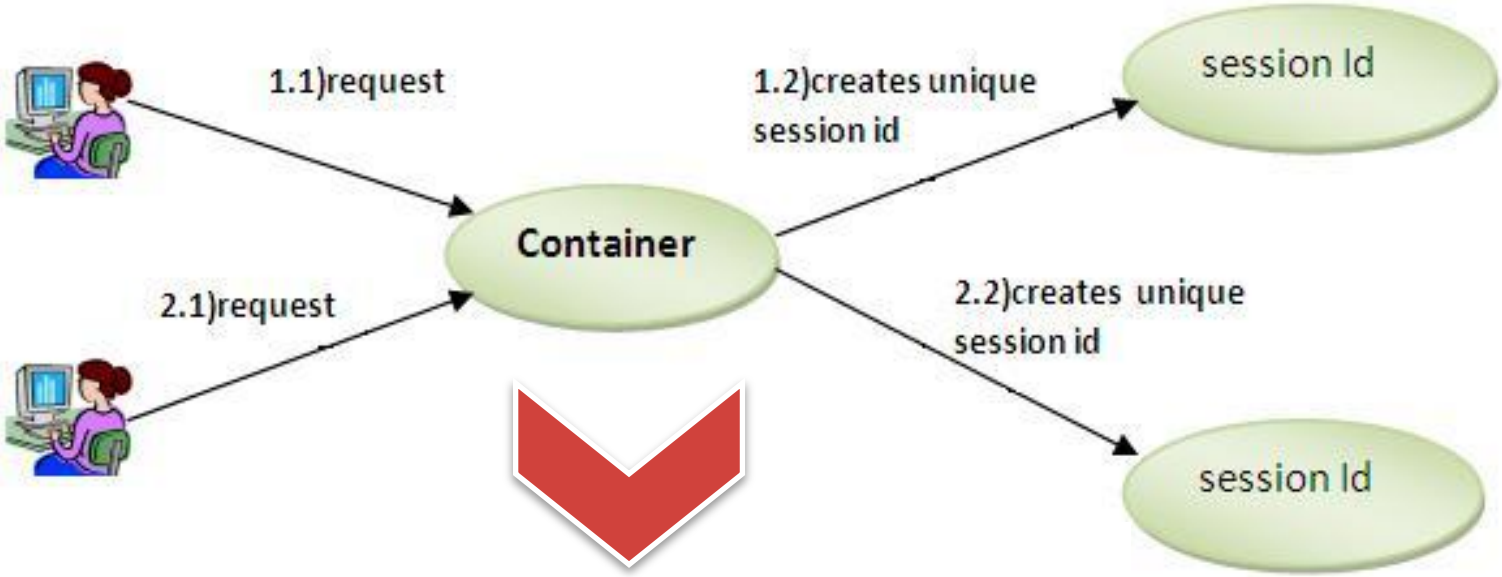
Práctica de Sesión con Cookies





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 HttpServlet**Request**

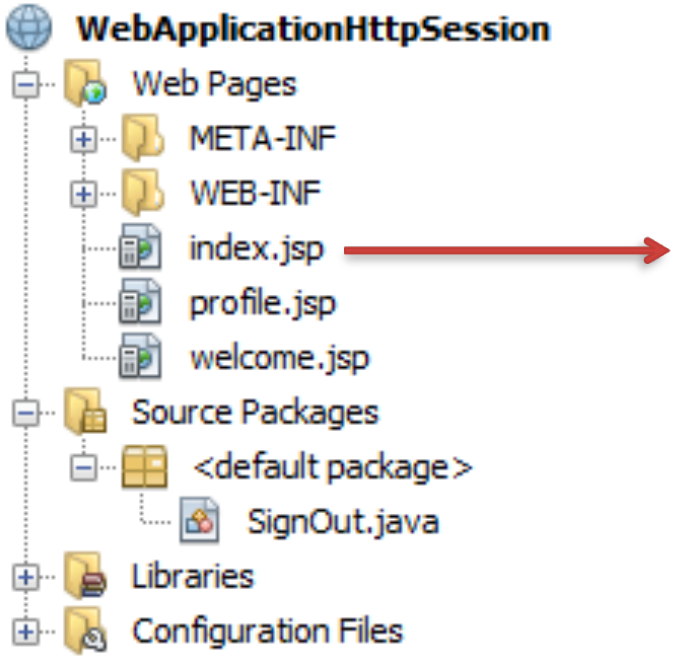
- **import javax.servlet.http.HttpSession;**
- **public HttpSession getSession()**

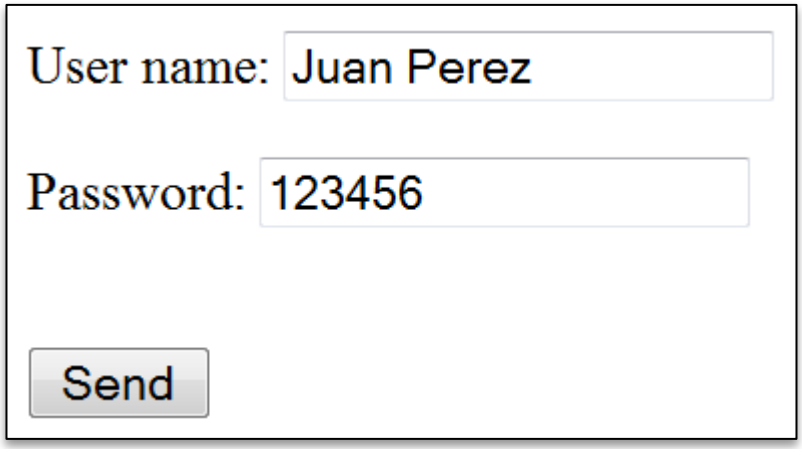
  Devuelve la sesión asociada con la solicitud, o si la solicitud no tiene una sesión, crea una.

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



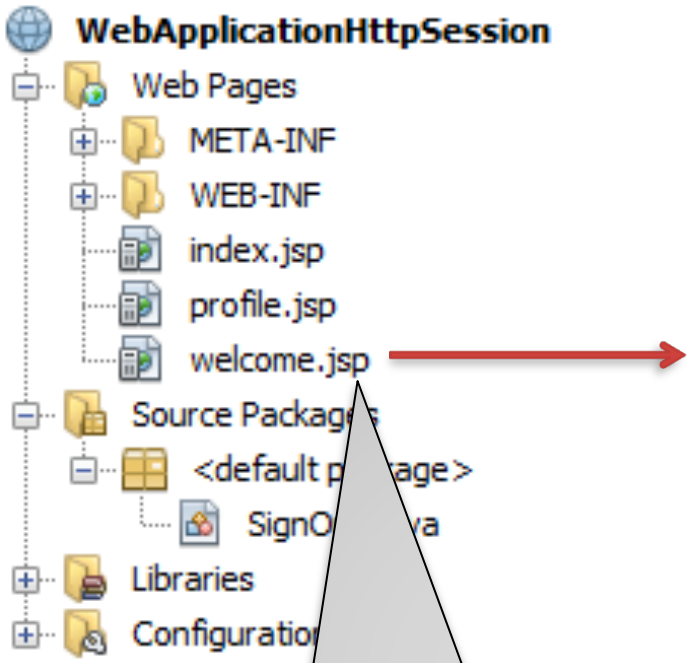


Tips

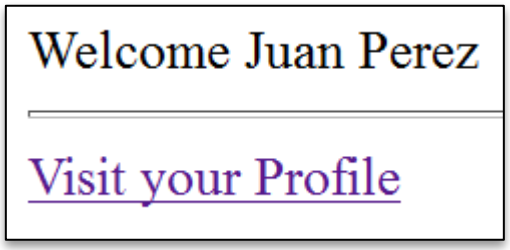
```
HttpSession mySession = request.getSession();
mySession.setAttribute("username", userName);
//////////////////////////////////////////////////
HttpSession mySession = request.getSession();
if (mySession.getAttribute("username")!=null)
```

WebApplicationHttpSession

Manejo de una Sesión HTTP



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

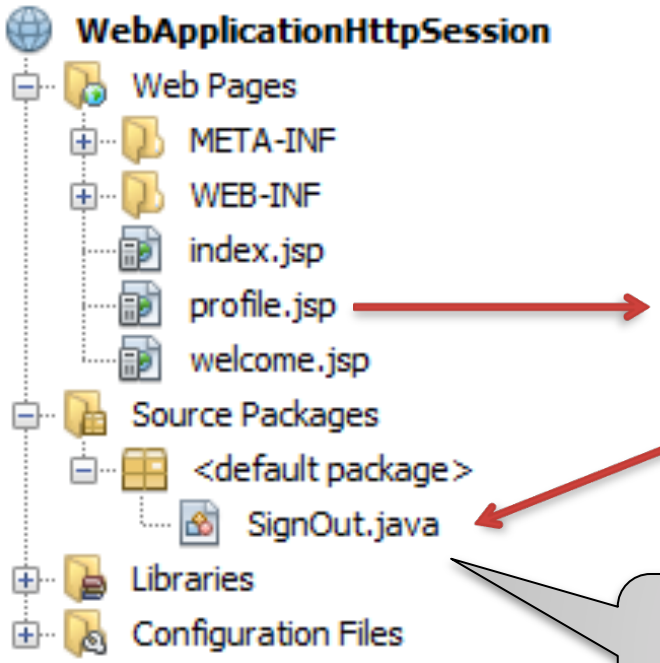


Crea una sesión con atributo "username"

NetBeans

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

NetBeans

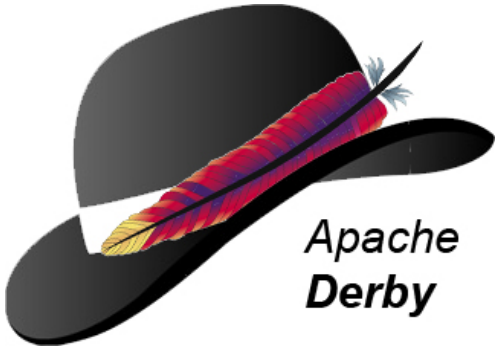
WebApplicationHttpSession

Acceso a Bases de datos

Seguridad de Java **java.policy**

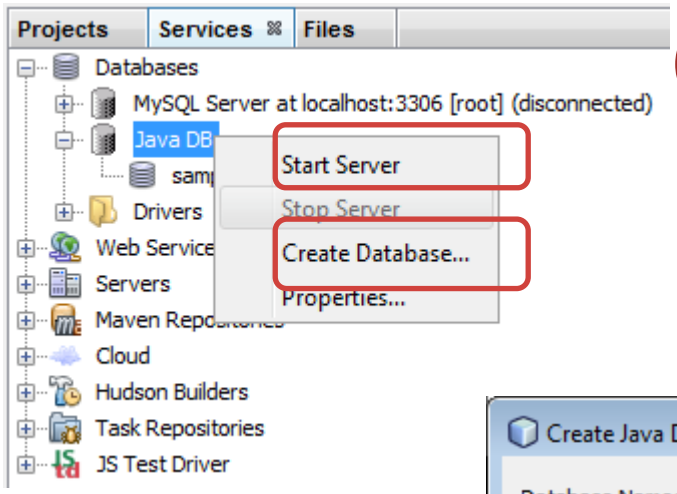
C:\Program Files\Java\jdk1.7.0_51\jre\lib\security\

```
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

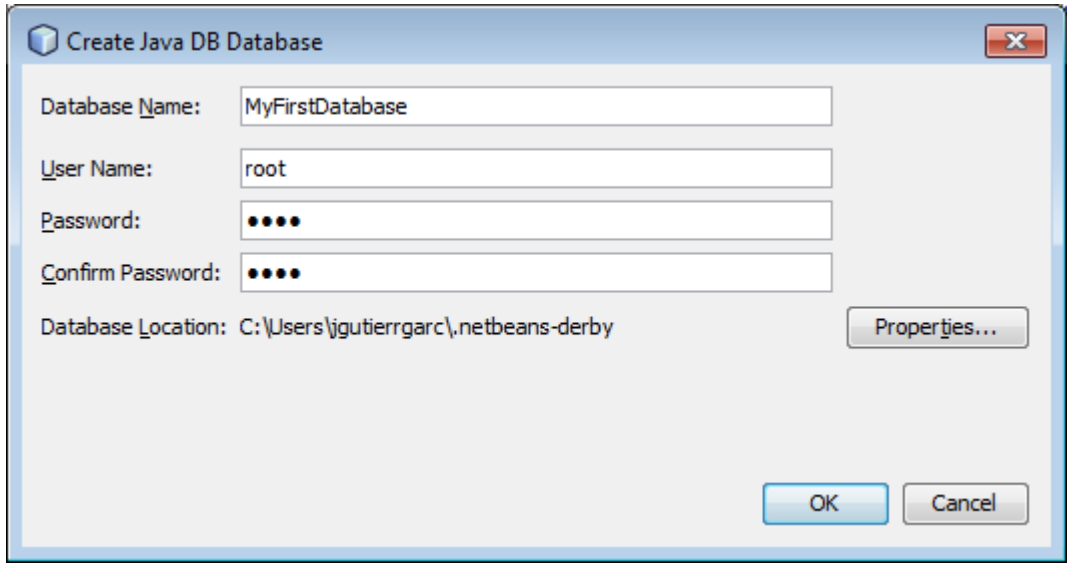


1

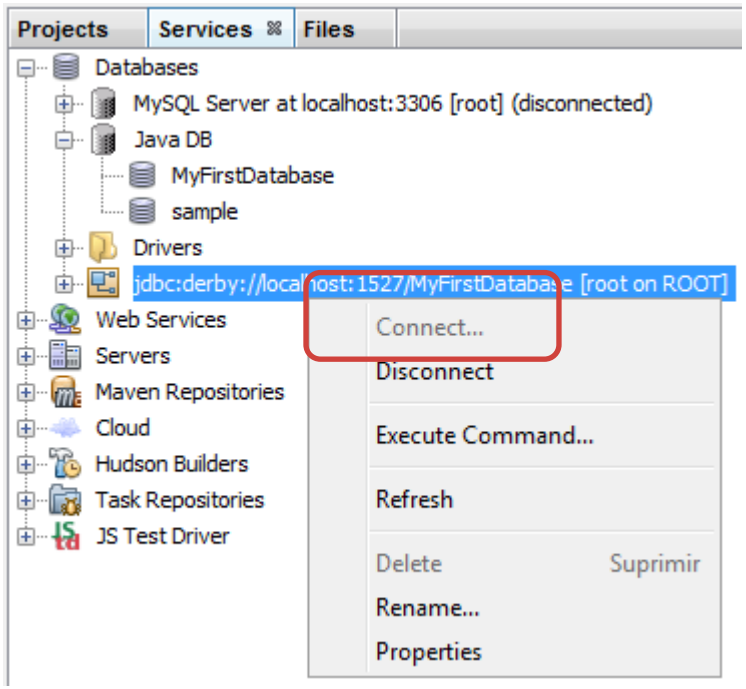
Arrancar el Servidor de BDs

Crear BD

2

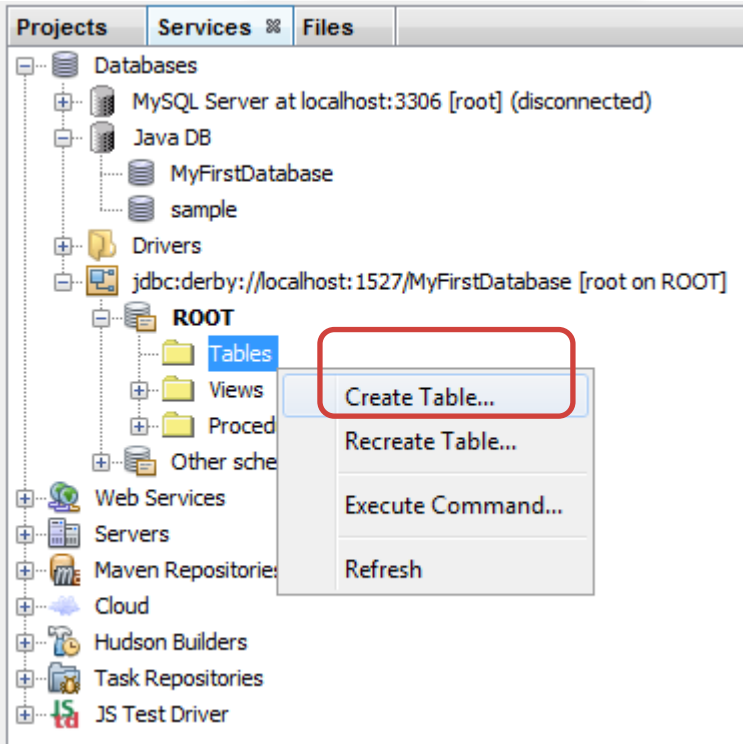


Java DB (Apache Derby)



3 Conectar la BD

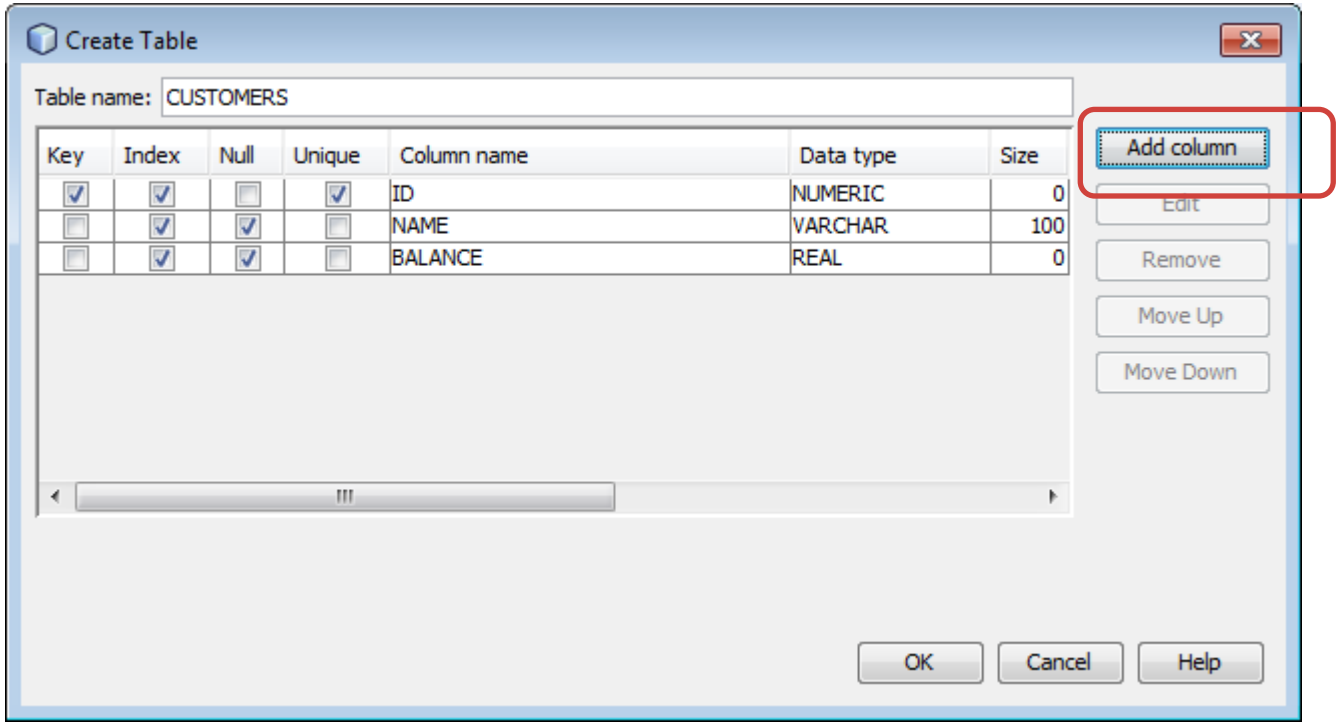
Crear Tabla 4



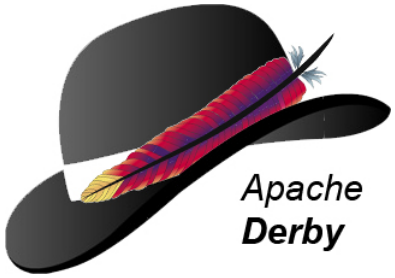
Java DB (Apache Derby)

5

Definir nombre y campos de la Tabla

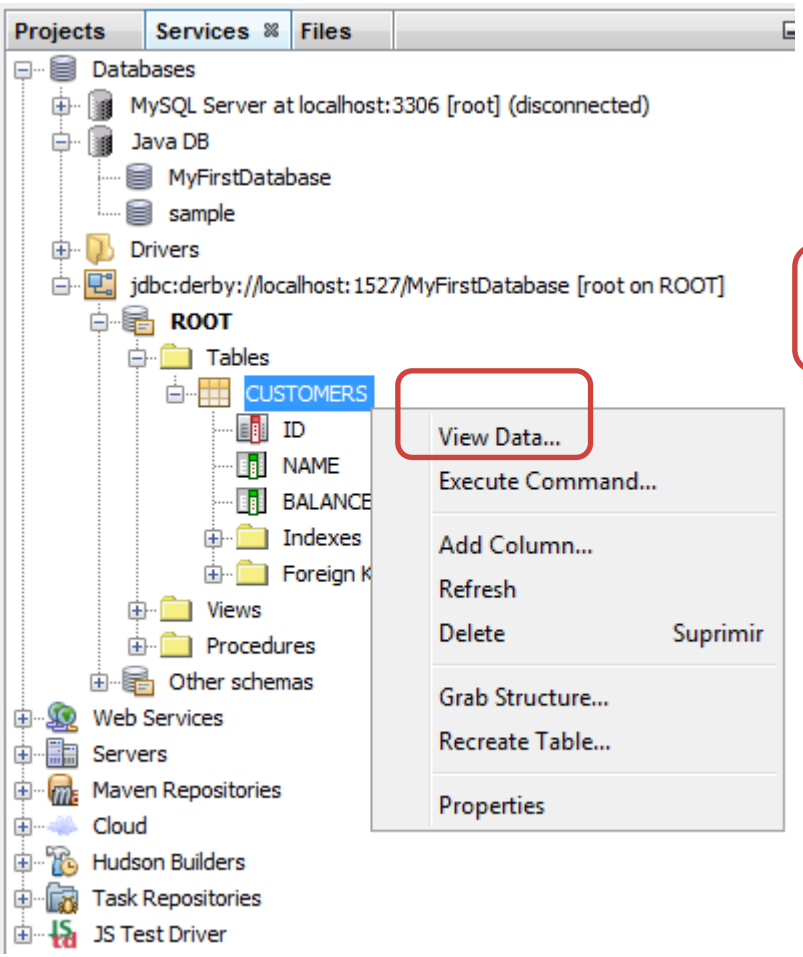


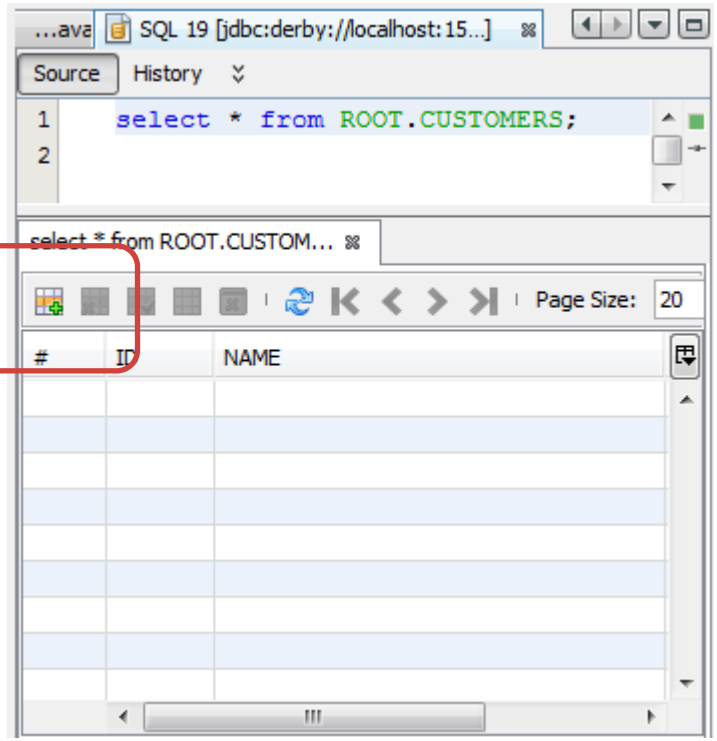
Java DB (Apache Derby)



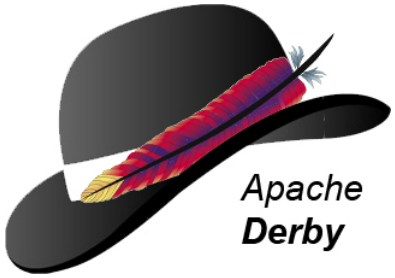
6

Agregar datos



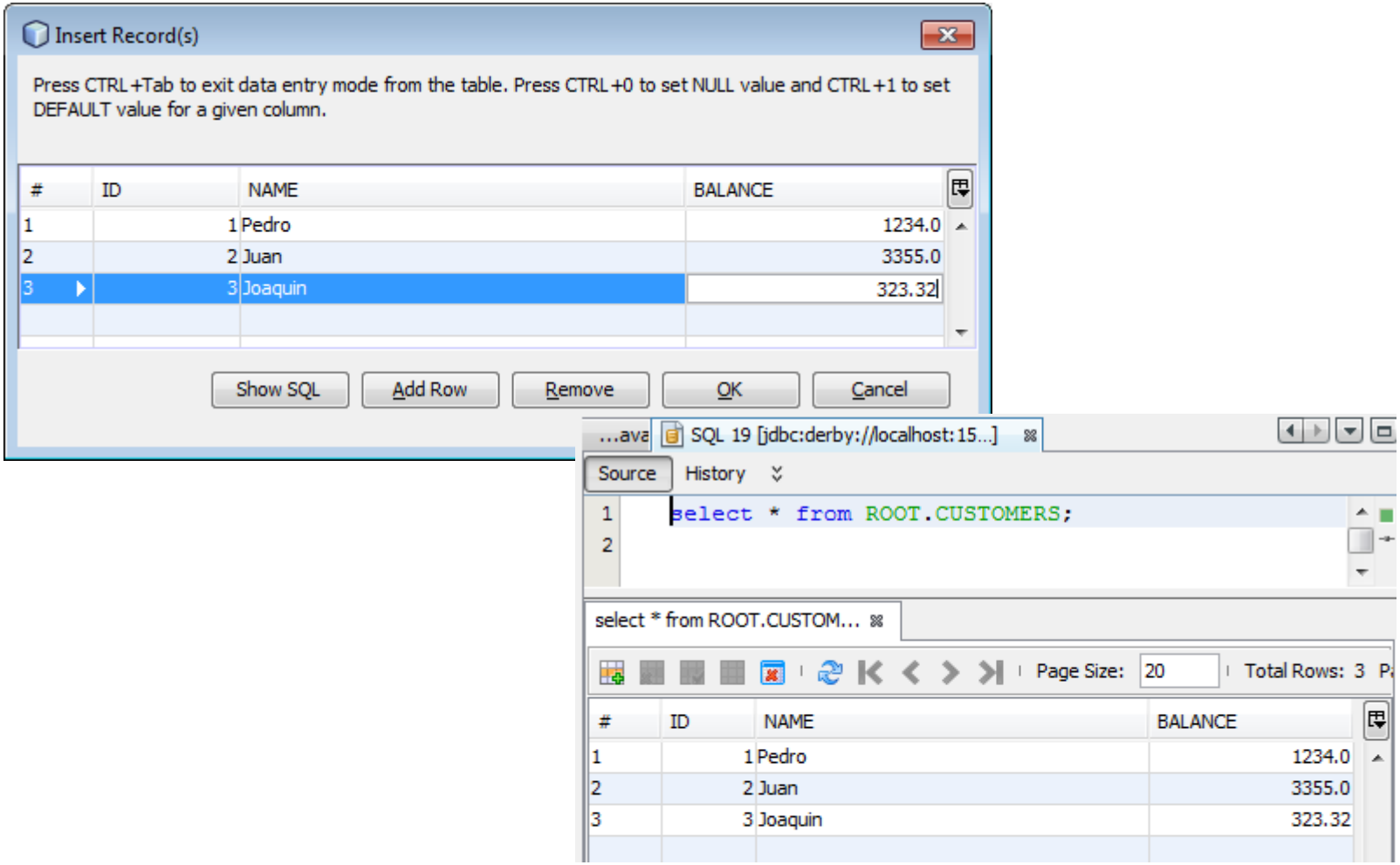


Java DB (Apache Derby)

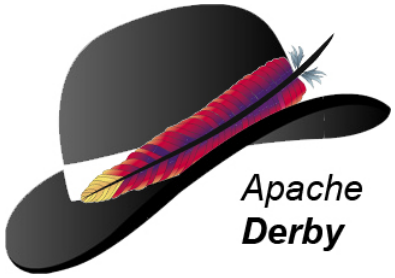


6

Agregar datos



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",
                "root",
                "root");
```

Cargando el driver

Creando la conexión

BD

Ubicación BD

usuario

pw

Java DB (Apache Derby)

Para hacer una consulta

Crear Statement usando la conexión

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

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

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

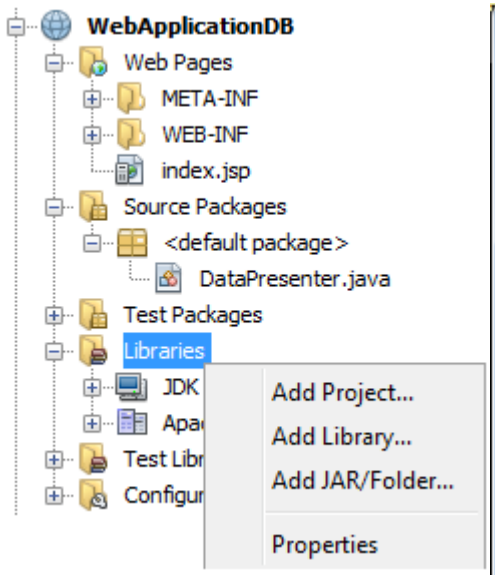
Crear un ResultSet

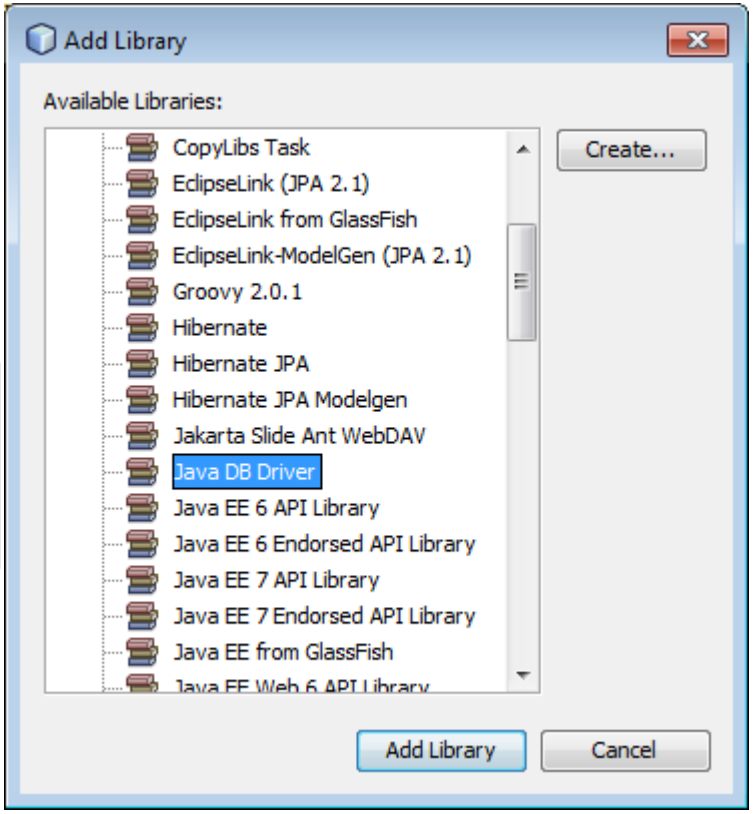
Si se modificó la BD

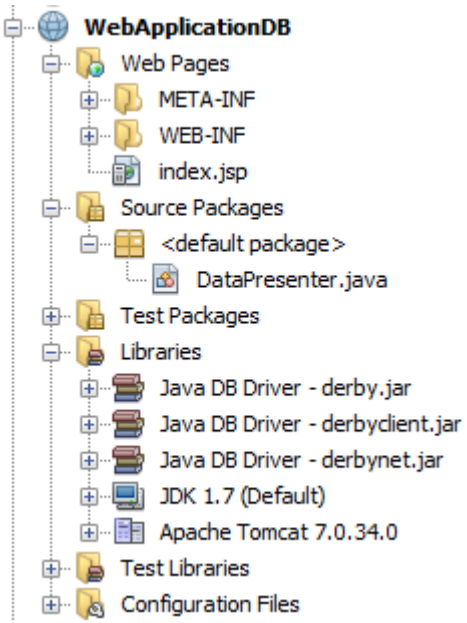
Cerrar la Conexión

Java DB (Apache Derby)

Para configurar el proyecto en Netbeans

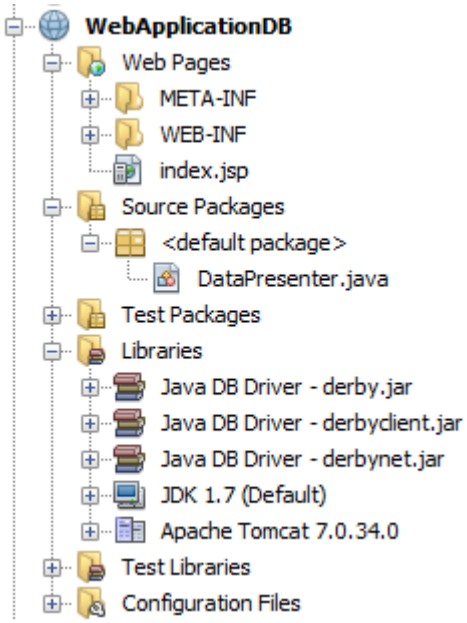


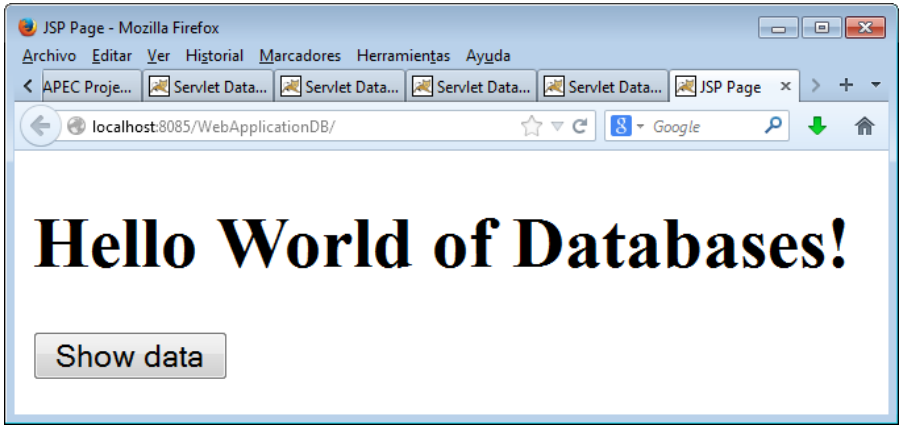


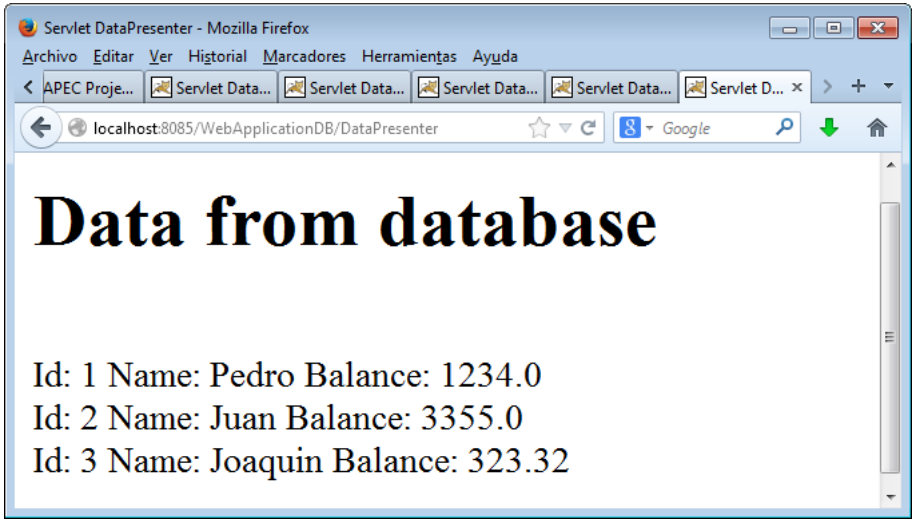


Java DB (Apache Derby)

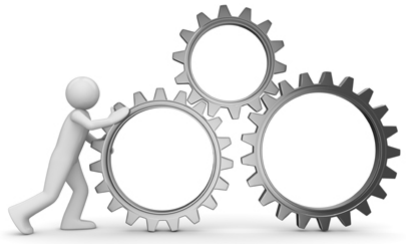
Hola Mundo







WebApplicationDB
Java DB



Instrucciones SQL de utilidad

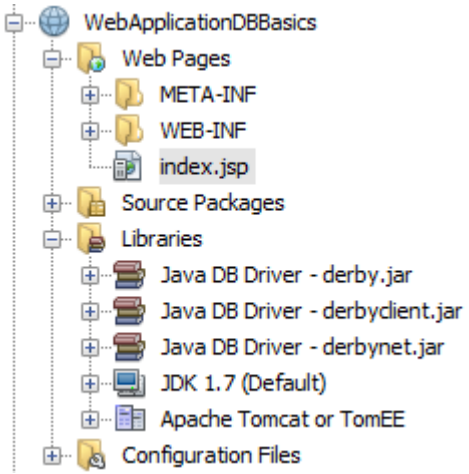
```
Statement query = con.createStatement();

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");
```

w3schools.com SQL Tutorial

http://www.w3schools.com/sql/default.asp

Práctica de Laboratorio

WebApplicationDBBasics
Java DB



Staff

Id: 1 Name: Francisco Balance: 123.0

Id: 2 Name: Jose Luis Balance: 456.0

Add a record

Id: Name: Balance:

Ok

Delete a record

Id:

Ok

Update a record

Id: Name: Balance:

Ok

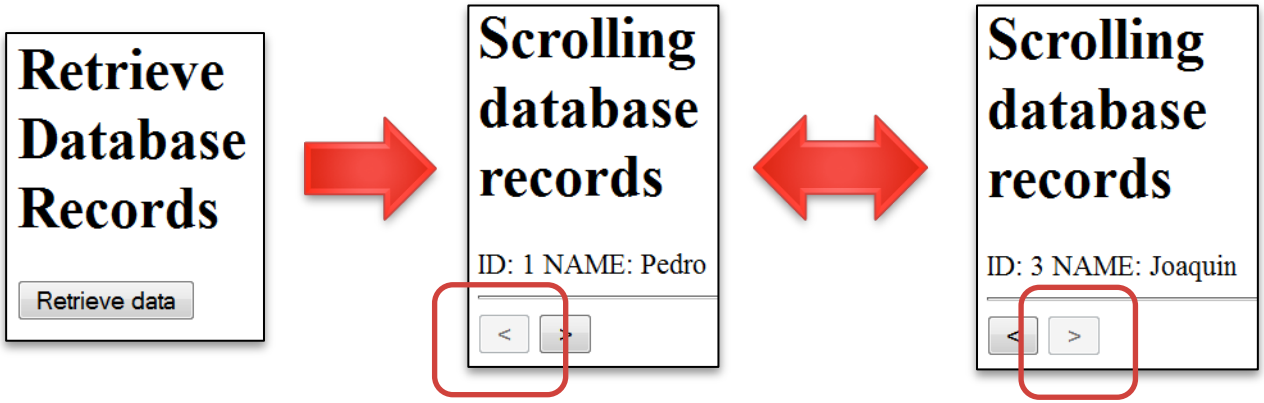
Tip 1: Poner nombres a los botones
if (request.getParameter("add")!=null)

Tip 2: Usar 3 formas

Sugerencia: Crear una BD nueva

Práctica de Laboratorio

WebApplicationDBScroll
Java DB



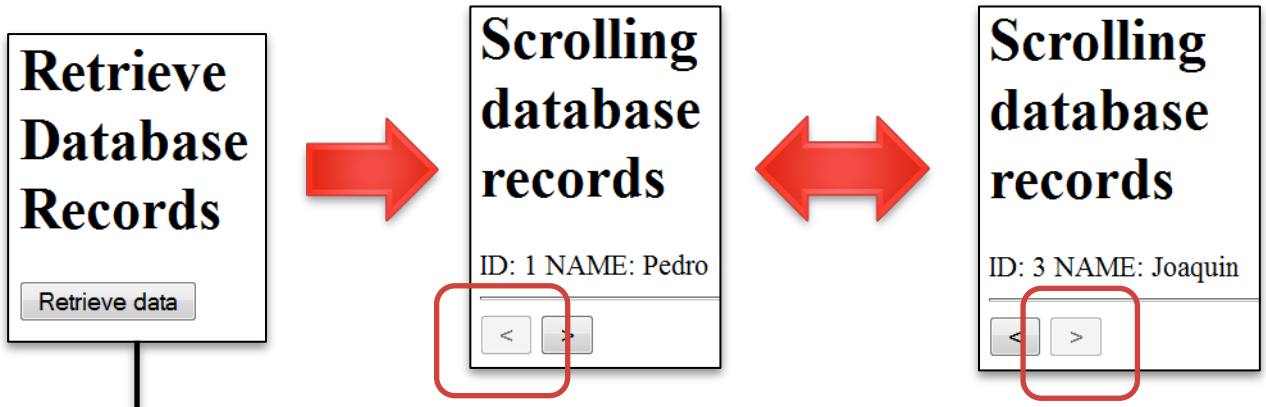
Tip: Usar sesiones HTTP para almacenar el ResultSet.

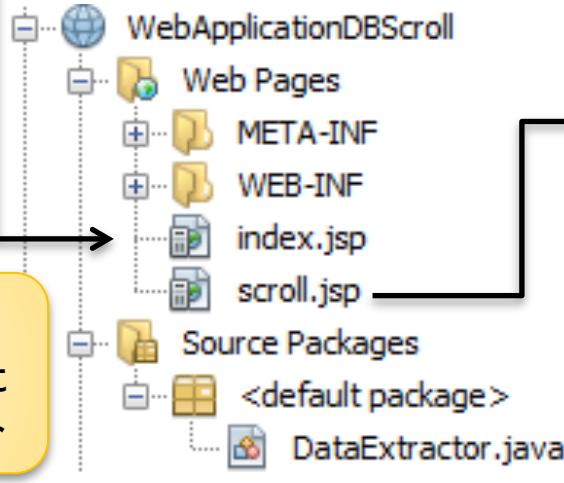
Información adicional

```
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

WebApplicationDBScroll
Java DB





1

index.jsp
llama a Servlet
DataExtractor

2

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

3

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

Práctica de Laboratorio

WebApplicationDBScroll
Java DB

```
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]);
```