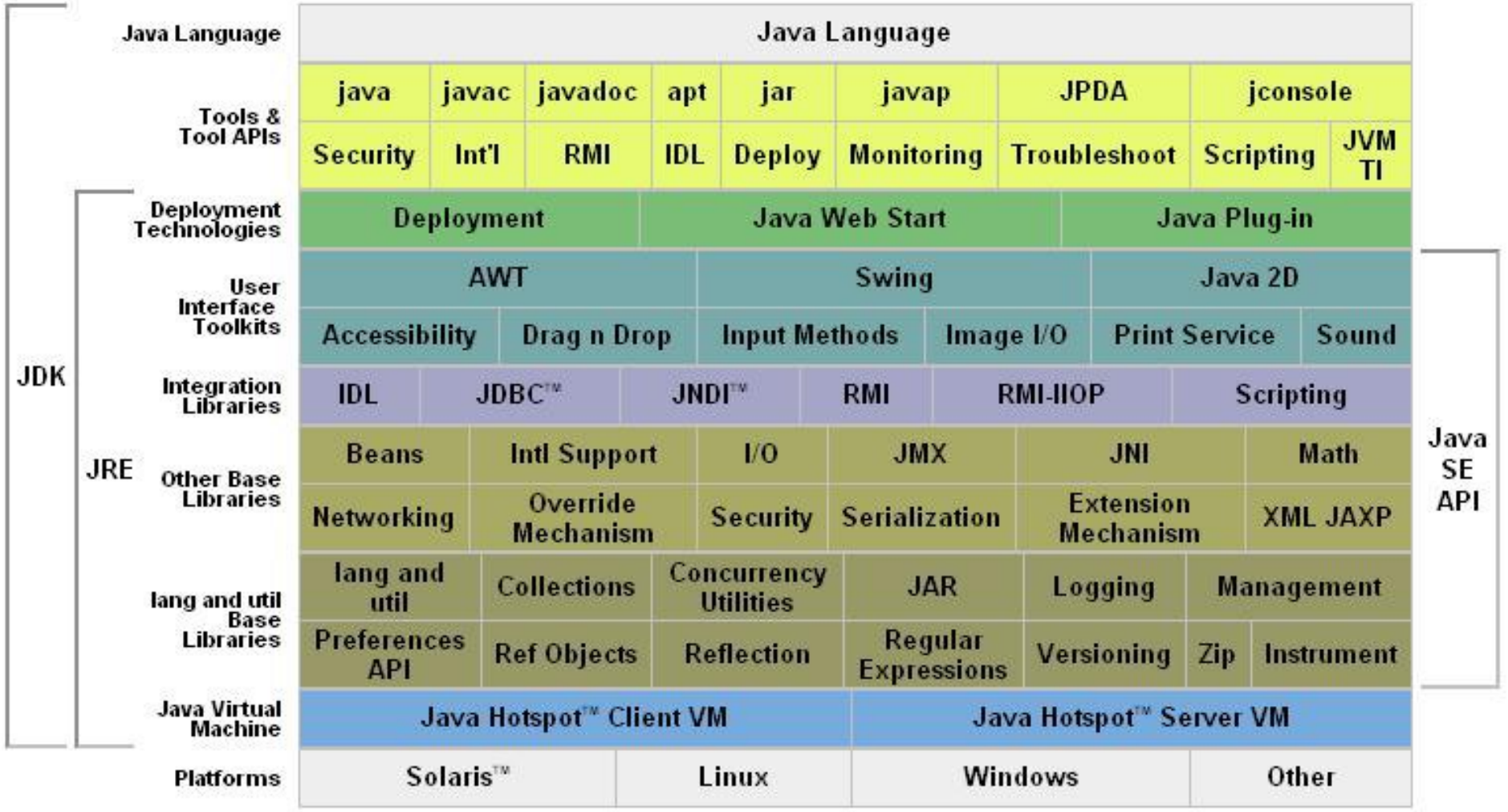
Curso de Java

“Java para Web”

Rogelio Ferreira Escutia

Tecnología Java



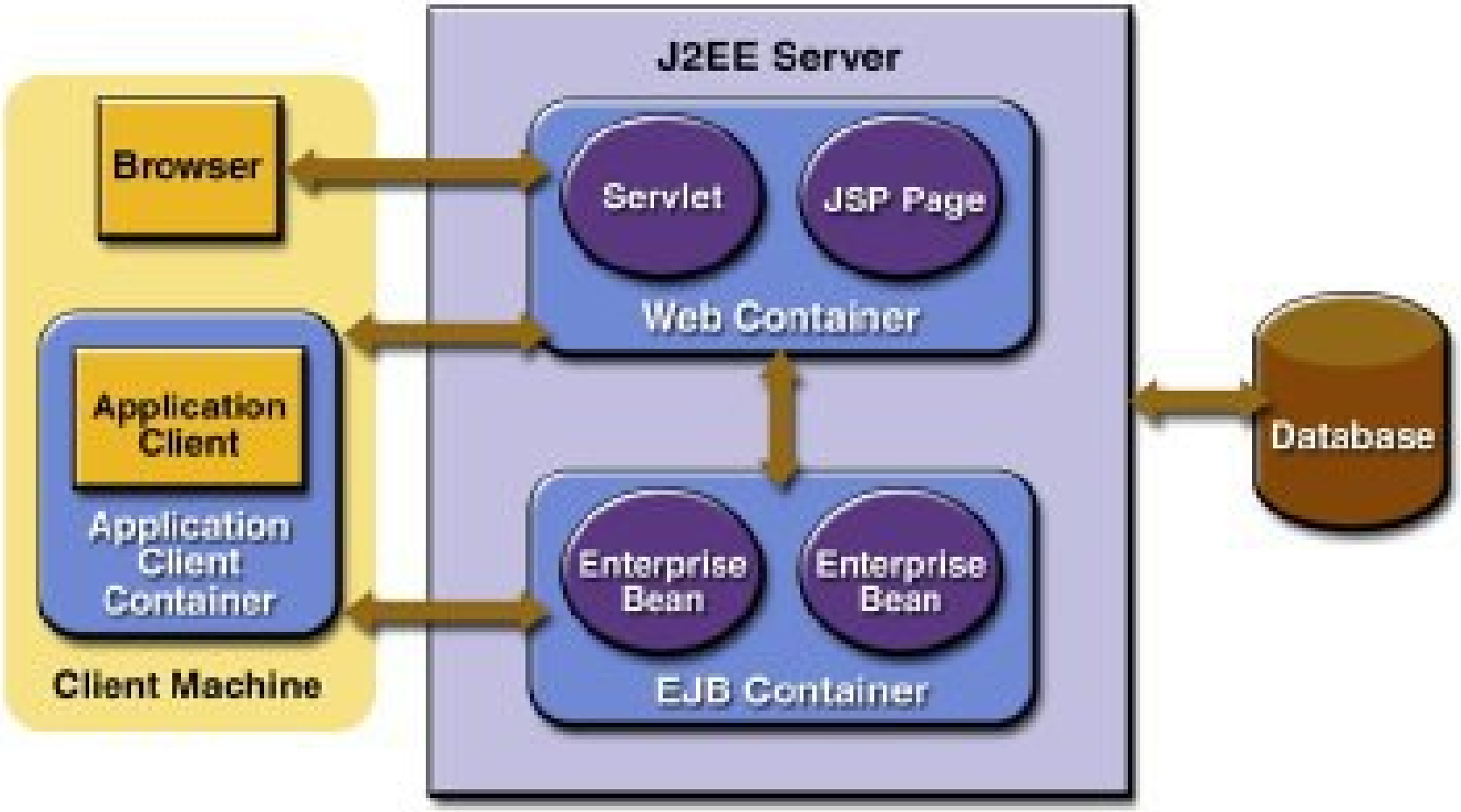
J2EE

- Para el desarrollo de apliaciones web, Sun desarrolló J2EE (Java 2 Entrerprise Edition).

- Es un conjunto de especificaciones para desarrollar aplicaciones basado en contenedores, entre las mas importantes:

  - J2EE Servlets
  - J2EE Java Server Pages
  - Enterprise Java Beans.

Contenedores



JSP

- JavaServer Pages (JSP) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

- Esta tecnología es un desarrollo de la compañía Sun Microsystems. La Especificación JSP 1.2 fue la primera que se liberó y en la actualidad está disponible la Especificación JSP 2.1.

• JavaServer Pages – Wikipedia, http://es.wikipedia.org/wiki/JavaServer_Pages, noviembre 2011

JSP y Servlets

- **Generalmente las páginas basadas en JSP se utilizan cuando la mayoría del contenido que se envía al cliente es texto estático y marcas, y sólo una pequeña porción del contenido se genera en forma dinámica mediante códigos de Java.**

- **Los Servlets se utilizan comúnmente cuando una pequeña porción del contenido que se envía al cliente es texto estático o marcas, de hecho algunos servlets no producen contenido, en vez de ello realizan una tarea a beneficio del cliente y después invocan a otros servlets o JSP para que proporcionen una respuesta.**

- **En la mayoría de los casos, las tecnologías de los servlets y JSP son intercambiables.**

• 49) Java Cómo Programar, Harvey M.Deitel y Paul J. Deitel, Pearson Educación Quinta Edición, México 2004, ISBN 970-26-0518-0, páginas 1325

Servlets

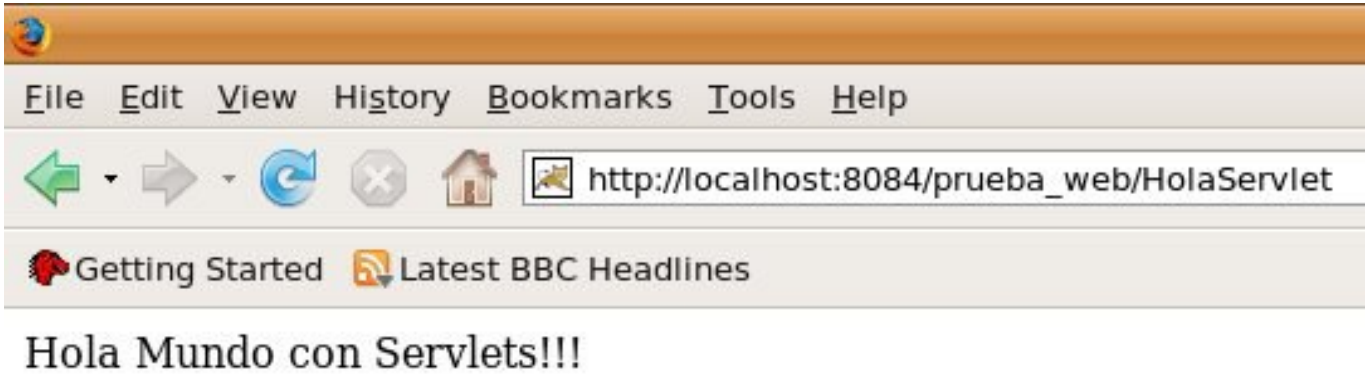
Hola Mundo (servlet)

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class HolaServlet extends HttpServlet {

    protected void doGet( HttpServletRequest peticion,
HttpServletResponse respuesta)
    throws ServletException, IOException {
        respuesta.setContentType("text/html;charset=UTF-8");
        PrintWriter salida = respuesta.getWriter();
        salida.println("<html>");
        salida.println("Hola Mundo con Servlets!!!");
        salida.println("</html>");
        salida.close();
    }
}
```

• 48) Internet and World Wide Web How to Program, H.M. Deitel, P.J. Deitel y T.R. Nieto, Prentice Hall Second Edition, Upper Saddle New Jersey USA 2002

Ejecutar Servlet

- **Ejecución del servlet “HolaServlet” utilizando JDK 1.6 update 2 con NetBeans 5.5.1 que incluye Tomcat corriendo en el puerto “8084” del proyecto “prueba_web” en un navegador Firefox en Linux Ubuntu 7.04**



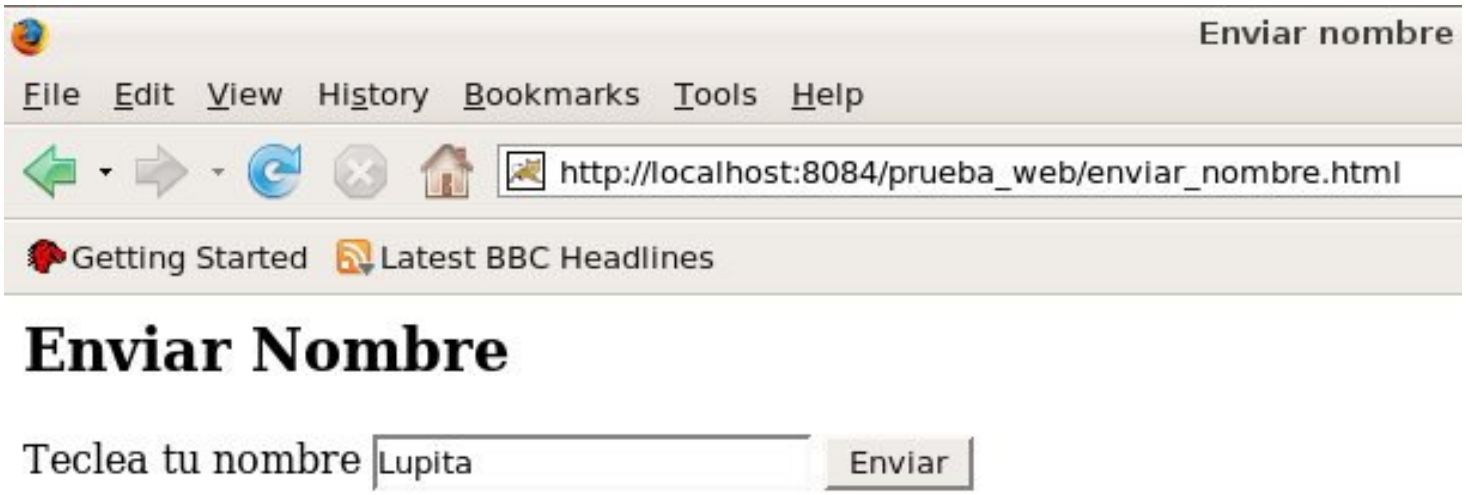
Enviar nombre

- **Página web “enviar_nombre.html” que se encarga de enviar por medio de un formulario una variable (nombre) a un servlet (recibir_nombre).**

```
<html>
  <head>
    <title>Enviar nombre</title>
  </head>
  <body>
     <h2>Enviar Nombre</h2>
     <form action="recibir_nombre" method="get">
        Teclea tu nombre
        <input type="text" name="nombre" />
        <input type="submit" value="Enviar" />
     </form>
  </body>
</html>
```

Enviar nombre

- **Página web “enviar_nombre.html” que se encarga de enviar por medio de un formulario una variable (nombre) a un servlet (recibir_nombre).**



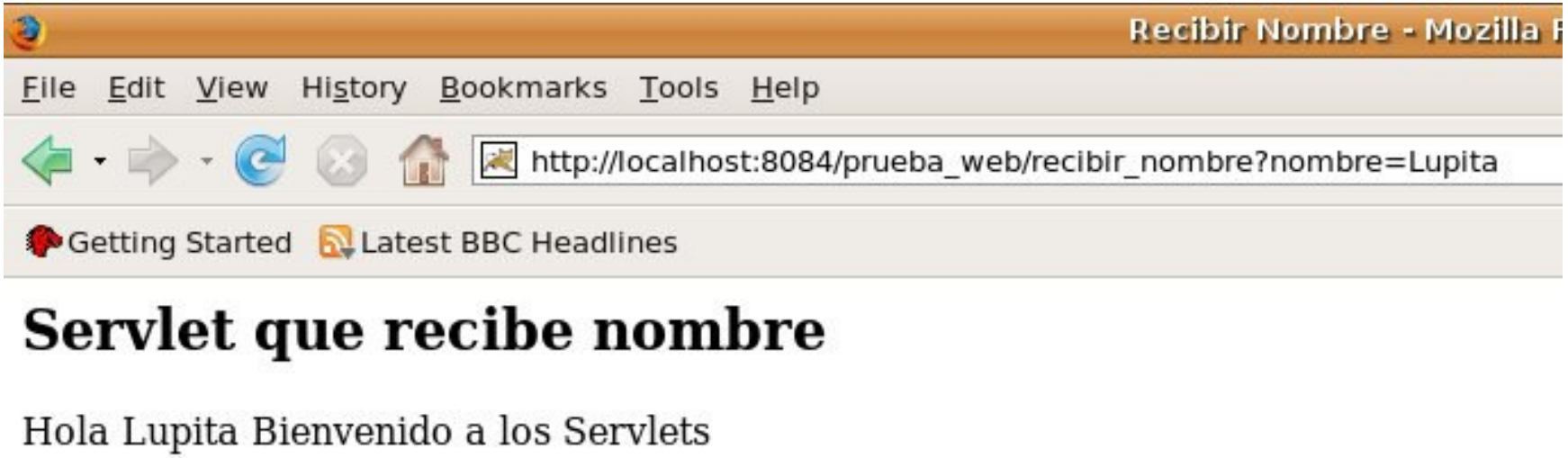
Recibir nombre

- **Servlet “recibir_nombre” que recibe la variable “nombre” que le envió el formulario y lo imprime en pantalla**

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class recibir_nombre extends HttpServlet {

    protected void doGet( HttpServletRequest peticion, HttpServletResponse respuesta)
    throws ServletException, IOException {
        String nombre2 = peticion.getParameter("nombre");
        respuesta.setContentType("text/html;charset=UTF-8");
        PrintWriter salida = respuesta.getWriter();
        salida.println("<html>");
        salida.println("<head>");
        salida.println("<title>Recibir Nombre</title>");
        salida.println("</head>");
        salida.println("<body>");
        salida.println("<h2>Servlet que recibe nombre</h2>");
        salida.println("Hola " + nombre2 + " Bienvenido a los Servlets");
        salida.println("</body>");
        salida.println("</html>");
        salida.close();
    }
}
```

Recibir nombre

- **Servlet “recibir_nombre” que recibe la variable “nombre” que le envió el formulario y lo imprime en pantalla**



JSP

Hola Mundo (JSP)

```
<html>
    <head>
        <title>Hola Mundo con JSP!!!</title>
    </head>
    <body>
        <h2>Hola Mundo con JSP!!!</h2>
        <%=
        new java.util.Date()
        %>
    </body>
</html>
```

Ejecutar JSP

- **Ejecución del código JSP “HolaJSP.jsp” utilizando JDK 1.6 update 2 con NetBeans 5.5.1 que incluye Tomcat corriendo en el puerto “8084” del proyecto “prueba_web” en un navegador Firefox en Linux Ubuntu 7.04**

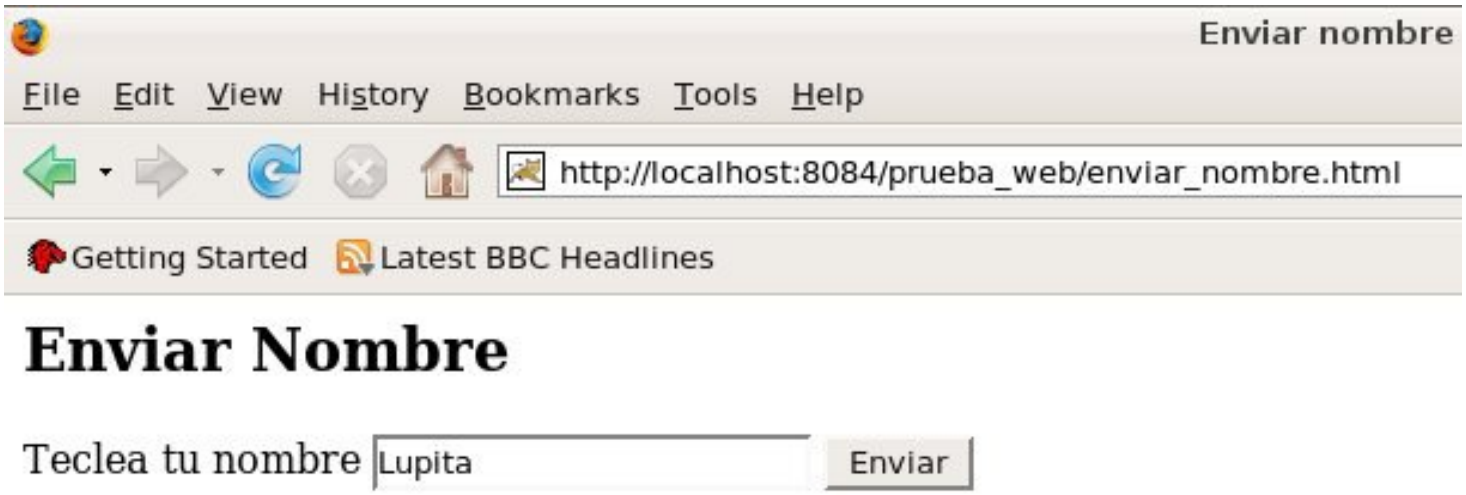
Enviar nombre

- **Página web “enviar_nombre.html” que se encarga de enviar por medio de un formulario una variable (nombre) a un JSP (recibir_nombre.jsp).**

```
<html>
  <head>
    <title>Enviar nombre</title>
  </head>
  <body>
     <h2>Enviar Nombre</h2>
     <form action="recibir_nombre.jsp" method="get">
        Teclea tu nombre
        <input type="text" name="nombre" />
        <input type="submit" value="Enviar" />
     </form>
  </body>
</html>
```

Enviar nombre

- **Página web “enviar_nombre.html” que se encarga de enviar por medio de un formulario una variable (nombre) a un JSP (recibir_nombre.jsp).**



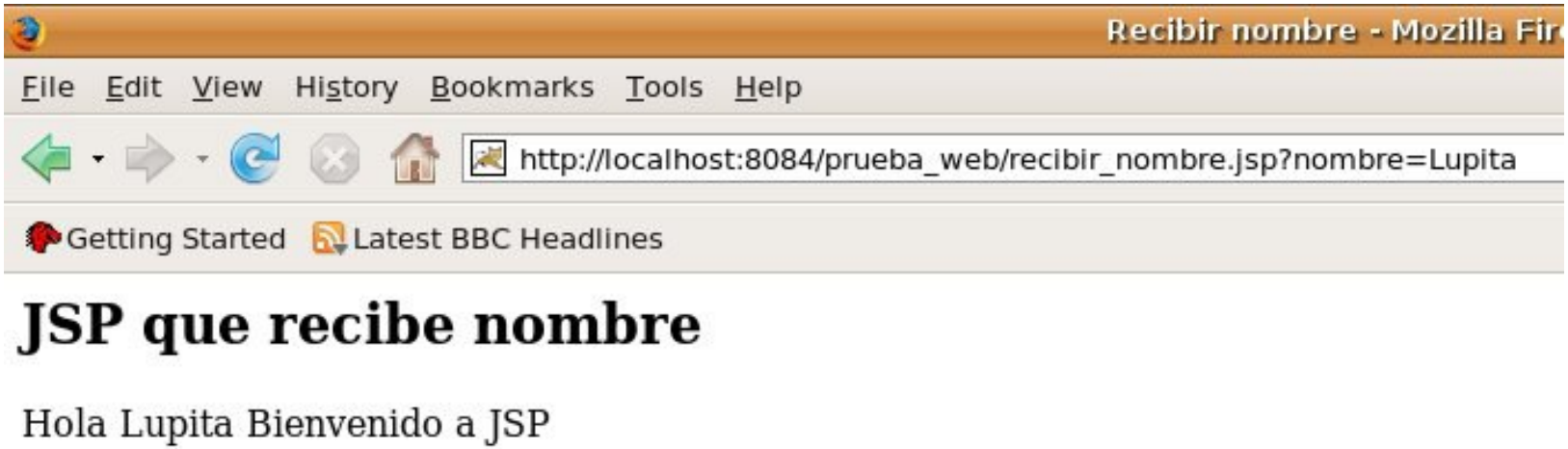
Recibir nombre

- **JSP “recibir_nombre.jsp” que recibe la variable “nombre” que le envió el formulario y lo imprime en pantalla**

```
<html>
    <head>
        <title>Recibir nombre</title>
    </head>
    <body>
        <h2>JSP que recibe nombre</h2>
        <%
        String nombre2 = request.getParameter("nombre");
        %>
        Hola <%= nombre2 %> Bienvenido a JSP
    </body>
</html>
```

Recibir nombre

- **Código JSP “recibir_nombre.jsp” que recibe la variable “nombre” que le envió el formulario y lo imprime en pantalla**



Manejo de Sesiones

- Para poder hacer uso de las sesiones se debe poner el atributo session de la directiva page a true, de esta forma se notifica al contenedor que la página interviene en un proceso que utiliza las sesiones del protocolo HTTP

- <%@page import="java.util.*" session='true'%> //no necesario en Servlets

Crear una sesión

- Para obtener la sesión de un usuario se utiliza el método getSession() que devuelve una interfaz de tipo HttpSession

- <%@page import="java.util.*" session='true'%> //no necesario en Servlets
- <% //no necesario en Servlets
- HttpSession sesion=request.getSession();
- %> //no necesario en Servlets

Identificar una sesión

- Para obtener la sesión de un usuario se utiliza el método getSession() que devuelve una interfaz de
- tipo HttpSession

- <%@page import="java.util.*" session='true'%>
- <%
- HttpSession session = request.getSession();
- out.print("Id. de la Sesion: "+sesion.getId());
- %>

Fecha de creación

- Es posible conocer el momento en el que se creó la sesión

```
<%@page import="java.util.*" session="true"%>
<%
HttpSession sesion=request.getSession();
out.println("Creación: "+sesion.getCreationTime());
Date momento=new Date(sesion.getCreationTime());
out.println("<BR>Creación: "+momento);
%>
```

Fecha de último acceso

- También se puede conocer la fecha y hora de la última vez que el cliente accedió al servidor con el que
- se creó la sesión, utilizando el método getLastAccesedTime()

```
<%@page import="java.util.*" session='true'%>
<%
HttpSession sesion=request.getSession();
Date acceso=new Date(sesion.getLastAccessedTime());
out.println("Último acceso: "+acceso+"<br>");
%>
```

Tiempo de conexión

- Teniendo en cuenta el momento en el que ser creó la sesión y la última vez que se accedió al servidor, se puede conocer el tiempo que lleva el cliente conectado al servidor, o lo que es lo mismo el tiempo que lleva el usuario navegando por las páginas JSP

- <%@page import="java.util.*" session='true'%>
- <%
- HttpSession sesion=request.getSession();
- long longDuracion=sesion.getLastAccessedTime();
- sesion.getCreationTime();
- Date duracion=new Date(longDuracion);
- out.println("Duracion:"+duracion.getMinutes() +"min."+duracion.getSeconds()+"seg");

Detección de sesión

- La interfaz HttpSession ofrece el método isNew() mediante el cual es posible saber si la sesión creada
- es nueva o se está tomando de una previamente creada

```
<%@page import="java.util.*" session='true'%>
<%
HttpSession sesion=request.getSession();
out.println("nueva: "+sesion.isNew());
%>
```

Guardar datos en sesión

- Para guardar un objeto en una sesión se utiliza el método setAttribute(), que ha sustituido al método
- putValue(). Este método utiliza dos argumentos

- El primero es el nombre que identificará a esa variable.
- El segundo es el dato que se va a guardar.

- <%@page import=”java.util.*” session=”true” %>
- <%
- HttpSession sesion=request.getSession();
- sesion.setAttribute(“trabajo”,”Sesiones en JSP”);
- %>

Guardar datos en sesión

- Si se quiere pasar un parámetro que no sea un objeto es necesario realizar una conversión

- <%@page import="java.util.*" session="true" %>
- <%
- HttpSession sesion=request.getSession();
- Integer edad=new Integer(26);
- sesion.setAttribute("edad",edad);
- %>

Recuperar datos de sesión

- Los datos que se guardan en la sesión permanecen ahí a la espera de ser utilizados. Para ello es necesario realizar el proceso contrario a cuando se graban, comenzando por la recuperación del objeto de la sesión para empezar a ser tratado.

- <%@page import="java.util.*" session="true" %>
- <%
- HttpSession sesion=request.getSession();
- String nombre=(String)sesion.getAttribute("nombre");
- out.println("Contenido de nombre: "+nombre);
- %>

Error al recuperar datos

- Si no existe ningún objeto almacenado en la sesión bajo el identificador que se utiliza en el método getAttribute(), el valor devuelto será null. Por ello habrá que prestar especial atención ya que si se realiza el cast de un valor null el contenedor JSP devolverá un error. Lo mejor en estos casos es adelantarse a los posibles errores que pueda haber

```
<%@page import="java.util.*" session="true" %>
<%
if(sesion.getAttribute("nombre")!=null){
        String nombre=(String)sesion.getAttribute("nombre");
        out.println("Contenido de nombre: "+nombre);
}
%>
```

Destruir sesion

- Los datos almacenados por las sesiones pueden destruirse en tres casos:
- El usuario abandona aplicación web (cambia de web o cierra el navegador)
- Se alcanza el tiempo máximo permitido de inactividad de un usuario (timeout).
- El servidor se para o se reincia.

- Pero la situación más probable es querer iniciar las sesiones o dar por finalizada una si se han cumplido
- una o varias condiciones. En este caso no es necesario esperar a que ocurra alguno de los tres casos
- citados anteriormente, ya que mediante el método invalidate() es posible destruir una sesión concreta.

- <%
- [...]
- sesion.invalidate();
- %>

Java para Web