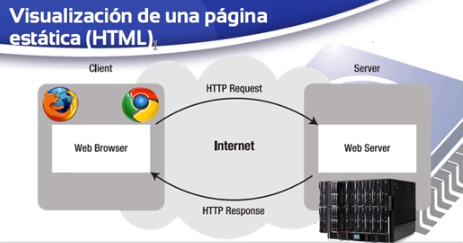
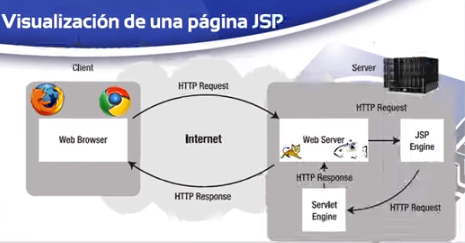
3.2 Desarrollo Web con JSP y Servlet - Diferencia en la ejecución de una página HTML y JSP

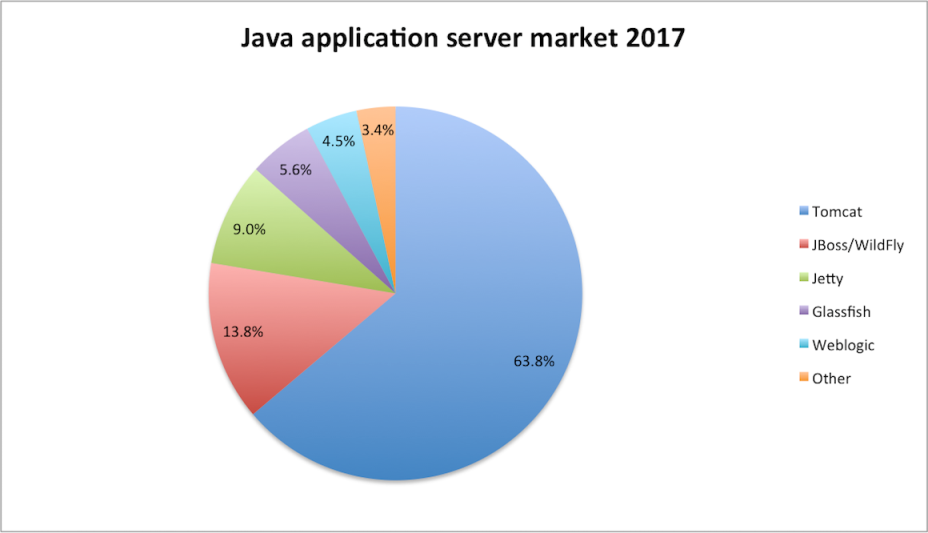




La única diferencia de los servidores Web es que el de la visualización de una página **jsp** debe ser compatible con jsp y servlet,

El servidor Web recoge la petición y la redirige al un **motor** JSP, ese motor JSP, **busca la página** **jsp** y cuando la encuentra, **la transforma** a través de un motor de servlet **en un servlet** y la envía al cliente, realmente lo que se visualiza en el cliente es un servlet.

Servidores Web compatibles con jsp y servlet entre otros tenemos



Diferencia entre JSP y Servlet

La diferencia entre Servlets y JSP es que los Servlets son clases que deben implementar la clase abstracta **HttpServlet**, en especial el método **doGet()** o **doPost()** y deben ser previamente compilados, mientras que los archivos JSP contienen código Java entre código HTML utilizando los símbolos **<%** y **%>**. Por esto un archivo JSP debe ser interpretado por el servidor al momento de la petición por parte del usuario.

Ejemplo **hora.jsp**

<%@ page import="java.util.Date" %>

<html>

<body>

<h1>La hora es: <%= new Date() %></h1>

</body>

</html>

En contenedor de servlets como [Tomcat](http://tomcat.apache.org/) traduce este archivo JSP y lo convierte en un Servlet.

Un servlet es una clase Java que implementa la interfaz Servlet ( heredando de HttpServlet por ejemplo ) y que generalmente escriben en un flujo de datos de una petición HTTP ( como la que un browser hace a un servidor web )

Ejemplo **Hora.java**

package hora;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.util.Date;

public class Hora extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

PrintWriter out = response.getWriter();

out.println("<html>");

out.println("<body>");

out.printf("<h1>La hora es %s </h1>", new Date());

out.println("</body></html>");

}

}

Estos dos ejemplos escriben aproximadamente lo mismo, se puede ver la diferencia entre uno y otro. Mientras que el servlet es una clase Java "normal" el JSP es una plantilla de propósito específico ( hecho para escribir HTML )

Otro ejemplo de **jsp** Básico

1. Crear una aplicación web java en **Netbeans** sin complementos.
2. Eliminar el archivo index.html desde el explorador de proyectos en Netbeans.
3. Clic derecho en la carpeta Web Pages de su proyecto en el explorador de proyectos de Netbeans, seleccionar la opción "New JSP", colocar en nombre index y clic en finalizar.
4. En el nuevo archivo **index.jsp** eliminar todo su contenido, y colocar el siguiente código de ejemplo modificando los datos de conexión (base, usuario y clave solamente):

<%@ page language="java" import="java.sql.\*" %>

<%

/\* == Datos de la base de datos creada en blanco en cPanel == \*/

String base = "personas";

String usuario = "root";

String clave = "juan";

/\* ========================================================= \*/

String url = "jdbc:mysql://localhost:3306/" + base;

String driver = "com.mysql.jdbc.Driver";

Connection conexion = null;

int conectado = 0;

String mensaje = "";

try{

Class.forName(driver);

conexion = DriverManager.getConnection(url,usuario,clave);

conectado = 1;

}catch (Exception ex){

mensaje = ex.toString();

}

%>

<html>

<head>

<title>CONEXION BD PERSONAS</title>

</head>

<body>

<%

if(conectado == 1){

String nombre, apellidos,direccion, sql;

Statement sentencia = null;

ResultSet registro = null;

sentencia = conexion.createStatement();

sql = "select \* from PERSONAS";

registro = sentencia.executeQuery(sql);

%>

<table border="1" width="100%">

<thead>

<tr>

<th>nombre</th>

<th>apellido </th>

<th>direccion </th>

</tr>

</thead>

<tbody>

<%

while(registro.next()){

     nombre = registro.getObject("nombre").toString();

     apellidos = registro.getObject("apellidos").toString();

     direccion = registro.getObject("direccion").toString();

     out.print("<tr>");

     out.print("<td>" + nombre + "</td>");

     out.print("<td>" + apellidos + "</td>");

     out.print("<td>" + direccion + "</td>");

     out.print("</tr>");

}

%>

</tbody>

</table>

<%

}else{

out.print("<h2>Error: " + mensaje + "</h2>");

}

%>

</body>

</html>

1. En el explorador de proyectos de Netbeans en la carpeta "Libraries", clic derecho y seleccionar la opción "Add Library", luego seleccionar "Mysql JDBC Driver" y clic en "Add Library".
2. Con el proyecto listo, en Netbeans clic en el menú superior "Run", opción "Build Project". Esto generará el archivo de distribución .war del proyecto en su computador una carpeta similar a:  
   C:\Users\usuario\Documents\NetBeansProjects\proyecto\dist

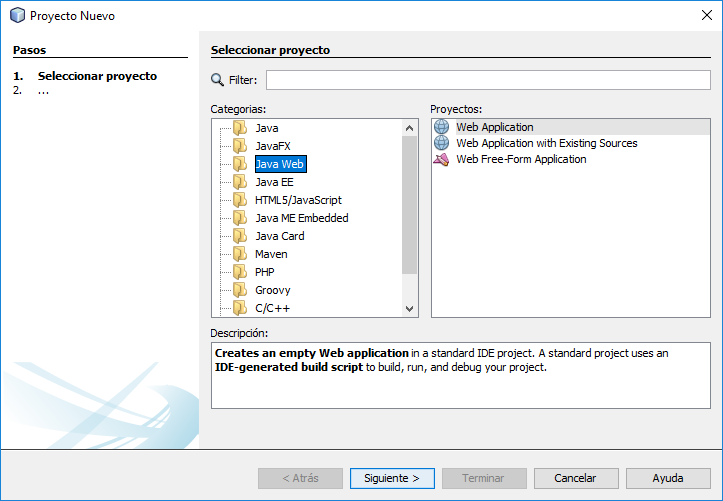
PROGRAMACIÓN WEB CON SERVLETS

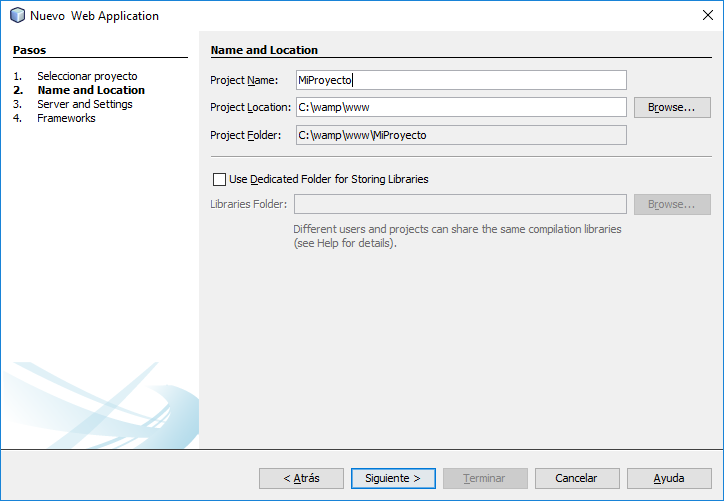
1. Creación de una aplicación Web

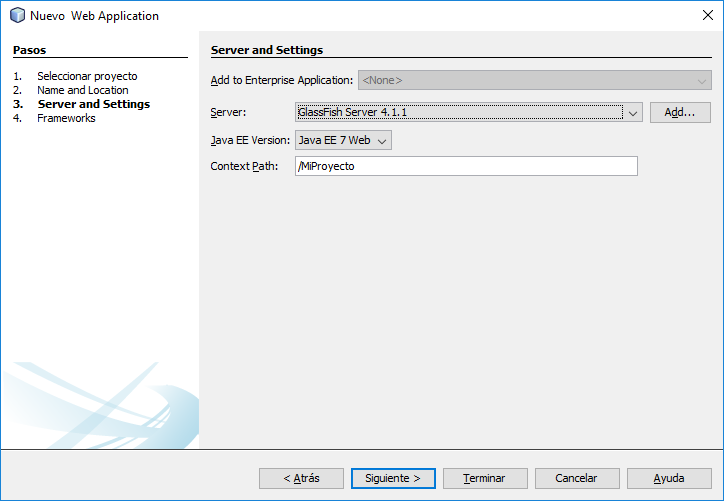
El ejemplo que vamos a realizar consiste en un formulario html y un servlet que captura los datos y crea una página de respuesta. Realizaremos la aplicación mediante NetBeans y haremos el despliegue en el servidor GlassFish.

1.1 Creación del proyecto

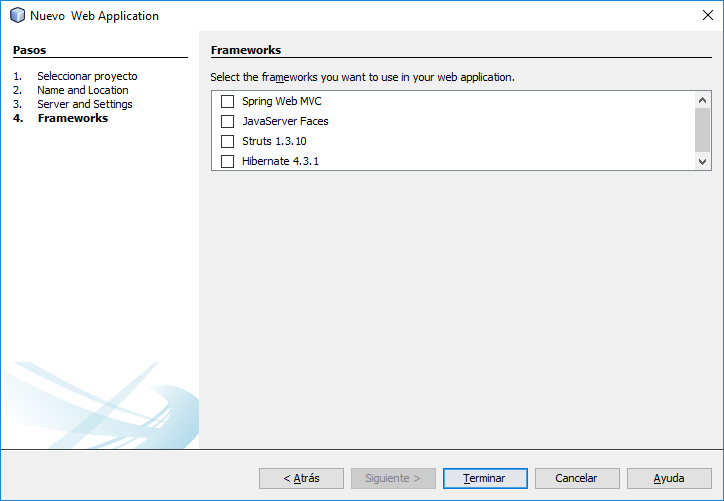
En NetBeans nos vamos a New Project y seleccionamos proyecto Web:





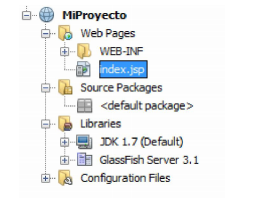


En nuestro caso no utilizaremos ningún framework:



Al principio te creara un index.hmtl, lo **borraremos** y crearemos un nuevo jsp al que llamaremos index.jsp

El resultado de la creación del proyecto es un directorio con la estructura estándar de los proyectos web:



1.2 Desarrollo del formulario

Como podemos apreciar al crear el proyecto se genera la página “index.jsp” que reutilizaremos para codificar nuestro formulario.

* Código final:

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

head>

<title>Ejemplo "Mi Primer Servlet"</title>

</head>

<body>

<form action="MiPrimerServlet" method="POST">

<BR><BR>Introduzca un texto en el cuadro y pulse "Enviar Consulta"

<BR><BR><input type="text" name="TEXTO">

<BR><BR><input type="submit"><input type="reset">

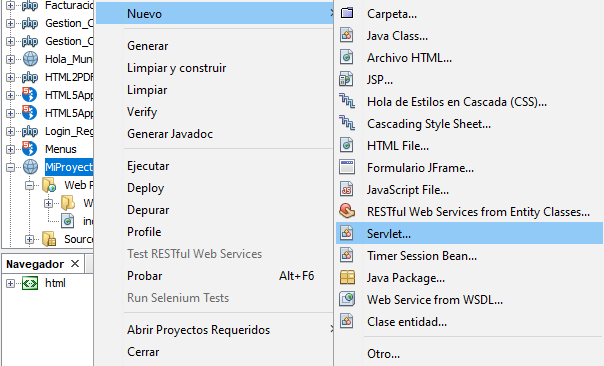
</form>

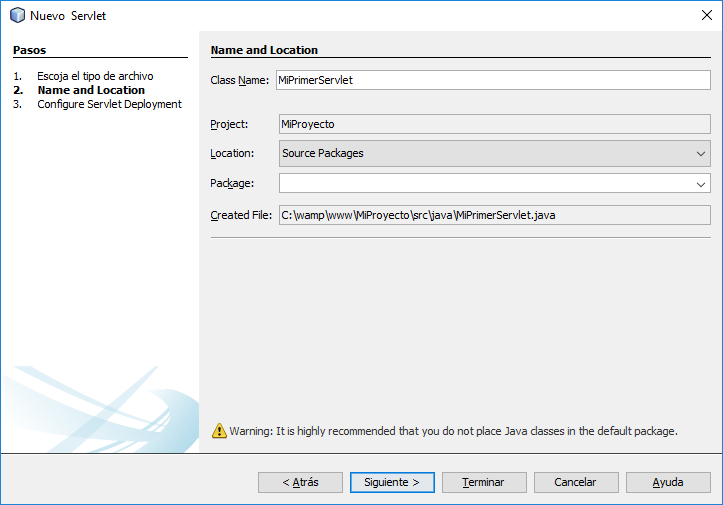
</body>

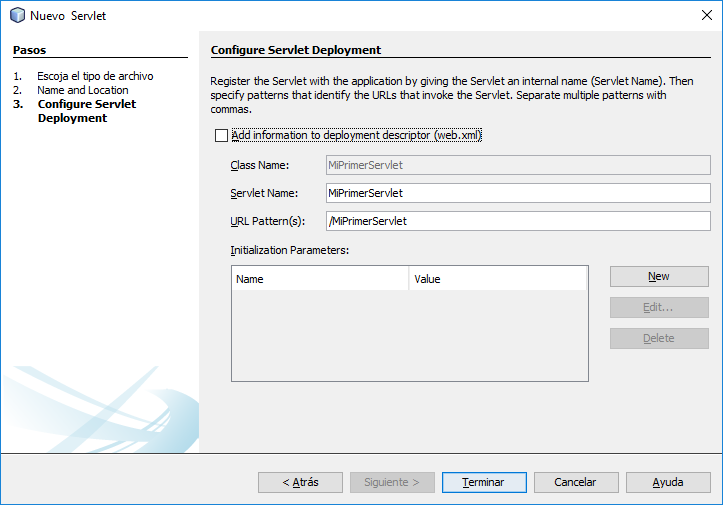
</html>

1.3 Desarrollo del servlet

Una vez realizado el formulario, crearemos un servlet para tratar la información.







Como podemos apreciar en la última pantalla ya no es necesario incluir la información del servlet en el fichero web.xml como se hacía antiguamente, esto se ha sustituido por una anotación en el propio código como veremos a continuación.

* Código final:

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "MiPrimerServlet", urlPatterns = {"/MiPrimerServlet"})

public class MiPrimerServlet extends HttpServlet {

/\*\*

\* Processes requests for both HTTP <code>GET</code>

\* and <code>POST</code> methods.

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse

response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

String txt = request.getParameter("TEXTO");

try {

out.println("<html>");

out.println("<head>");

out.println("<title>Servlet MiPrimerServlet</title>");

out.println("</head>");

out.println("<body>");

out.println("<p>Usted ha escrito : " + txt + "</p>");

out.println("</body>");

out.println("</html>");

} finally {

out.close();

}

}

/\*\*

\* Handles the HTTP <code>GET</code> method.

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP <code>POST</code> method.

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

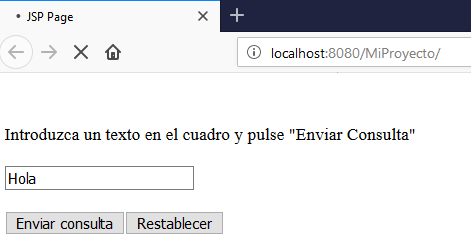
return "Short description";

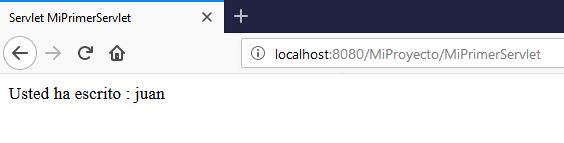
}

}

1.4 Desplegando la aplicación y ejecutándola

Para desplegar y ejecutar la aplicación en el servidor tan solo tenemos que pulsar sobre el botón derecho en **MiProyecto** y seleccionar “**Deploy**” (**desplegar**) y después ejecutar la aplicación o directamente ejecutarla ya que previamente se realizará el despliegue.



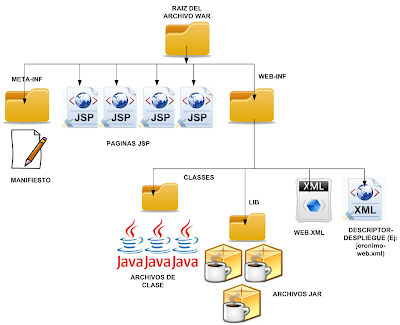


### [Empaquetado de Aplicaciones JEE - Archivos WAR](http://jmaw.blogspot.com/2012/09/empaquetado-de-aplicaciones-jee_28.html)

Los archivos **WAR**, son un tipo especial de JAR utilizado para distribuir los artefactos o contenido de las aplicaciones Web en tecnología JEE: páginas Web HTML o JSP, clases Java, servlets Java, archivos XML, librerías de etiquetas (tag libraries) y otros recursos.

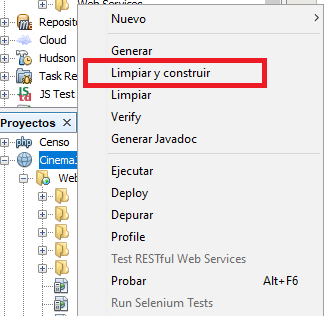
Al igual que los JAR pueden estar firmados digitalmente por temas de seguridad.

**Estructura de un archivo WAR**



Los archivos WAR incluyen carpetas y archivos especiales. Se incluye la carpeta META-INF con el manifiesto como en los JAR, pero además está la carpeta WEB-INF específica de las aplicaciones Web.

Para generar un **.war** utilizando el IDE de Netbeans , una vez tengamos la aplicación web terminada, pulsamos con el botón derecho del ratón sobre la aplicación, seleccionaremos la opción Limpiar y construir



Dentro de la carpeta de tu proyecto verás una nueva carpeta llamada "dist".Dentro la carpeta verás tu archivo WAR creado.