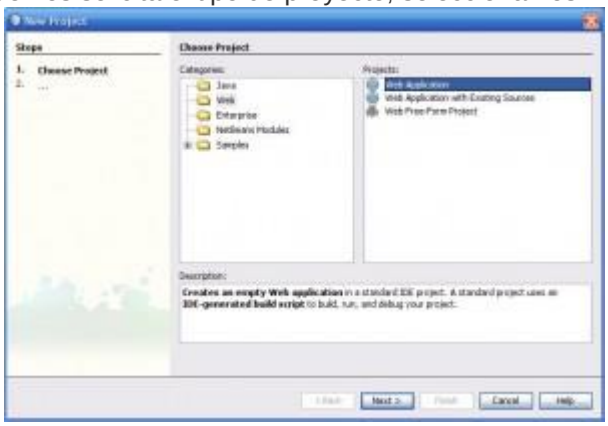
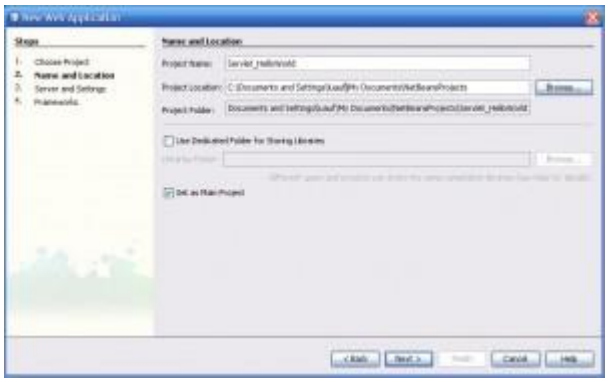
Introducción a Java Servlets con NetBeans

Los Servlets son una de las tecnologías más importantes de Java. Este ejemplo de construcción de un clásico Hola Mundo en una aplicación web J2EE.

- Iniciamos NetBeans y seleccionamos **File \ New Project…**
- Se abre un diálogo que nos solicita el tipo de proyecto, seleccionamos **Web Application**

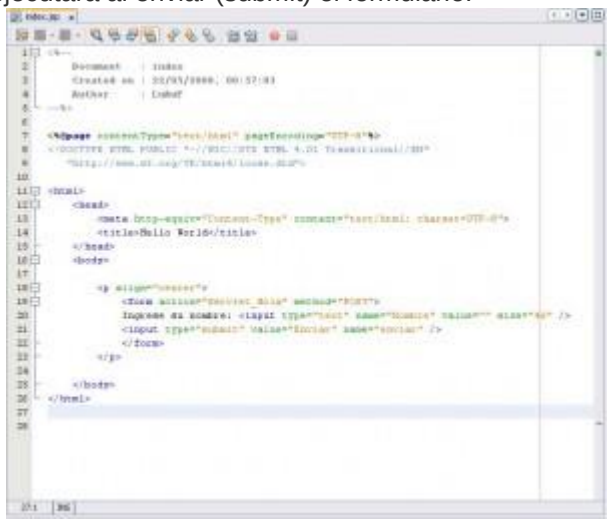


- En el siguiente paso, nos solicita el nombre del proyecto (**Project Name**), en nuestro caso ingresamos Servlet_HelloWorld



- El siguiente paso, nos solicita el servidor de aplicaciones (contenedor de Servlets a utilizar). Así, del combo Server, seleccionamos Tomcat (según la versión que yo tengo instalada, Tomcat 6.0.16)

- Finalmente, nos permite seleccionar el o los frameworks a utilizar (Spring, Struts, JSF, etc). No seleccionamos nada pues, por ahora, para este ejempo, no vale la pena.
- NetBeans crea por su cuenta el proyecto, una estructura de directorios, y dentro de la carpeta **Web Pages** un arhivo **index.jsp**, que será el punto de partida de nuestra aplicación. Si bien es de extensión JSP, por ahora no escribiremos código JSP, sinó simplemente un formulario HTML. En este formulario HTML definiremos en el atributo **action** el nombre del **servlet** que se ejecutará al enviar (submit) el formulario.



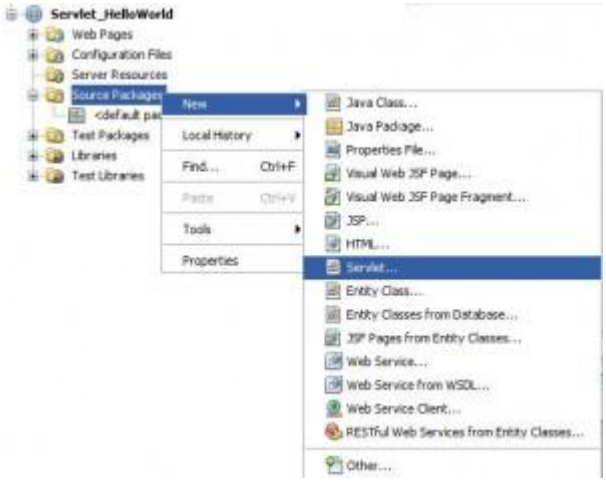
[HTML]

Ingrese su nombre:

Enviar

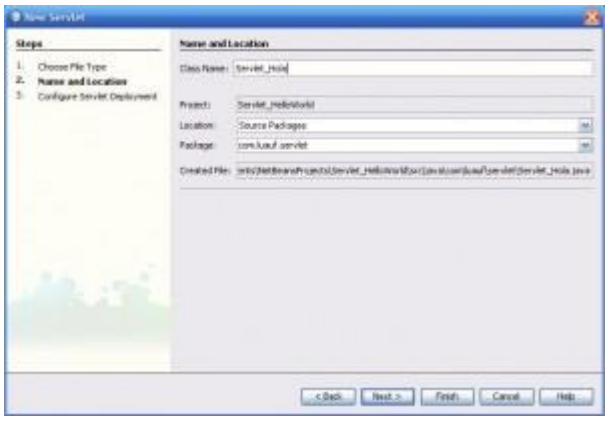
[/HTML]

- Luego, desde el explorador de proyectos, luego de hacer clic derecho en la carpeta Source Packages, seleccionamos **New \ Servlet…**



- Aquí, se abre un diálogo que nos solicita nombre y paquete del servlet.

- En nombre, hay que ingresar el mismo nombre del atributo **action** del formulario creado anteriormente, pues este será el **servlet** que recibirá los datos enviados por el formulario HTML. En nuestro caso, según indicamos en el form: **Servlet_Hola**.
- En paquete se puede ingresar el nombre que se quiera, ahora no es de importancia.



- Dados el nombre del servlet y el paquete, hacemos clic sobre **Finish**.
- Finalizado esto, automáticamente crea una clase con el nombre de servlet dado (Servlet_Hola para nosotros), que hereda de **HttpServlet**. Además redefine (override) algunos métodos (**doGet, doPost, getServletInfo**) y los rellena con un poco de código. Además, crea un método **processRequest**(invocado desde los métodos doGet y doPost) para procesar los formularios que llegan por los métodos GET y POST.
- Nosotros, en este ejemplo, nos limitaremos completar con unas pocas líneas (pues la mayoría la completó automáticamente el NetBeans) el método processRequest para que cree una página HTML que será la respuesta del formulario enviado:

```
[JAVA]
package com.luauf.servlet;
import java.io.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class Servlet_Hola extends HttpServlet {

/**
* Processes requests for both HTTP GET and POST methods.
* @param request servlet request
* @param response servlet response
*/
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
*/
out.println("");
out.println("");
out.println("");
out.println("");
out.println("");
out.println("
");
out.println("Hola " + request.getParameter("nombre").toString() + "");
out.println("
");
out.println("

");
out.println("");



} finally {
out.close();
}
}

//
/**
* Handles the HTTP GET method.
* @param request servlet request
* @param response servlet response
*/
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
/**
* Handles the HTTP POST method.
* @param request servlet request
* @param response servlet response
*/
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
```

```
/**
* Returns a short description of the servlet.
*/
public String getServletInfo() {
return “Short description”;
}
//
}
```
[/JAVA]

- Para finalizar, vamos a ejecutar el proyecto, podemos hacerlo desde el menú **Run** o haciendo abriendo el menú contextual del proyecto (desde el explorador de proyectos) y seleccionando **Run**.
- Al ejecutar una aplicación web con NetBeans, lo primero que hace el mismo es un **Deploy**, algo así como distribuir la aplicación en el servidor. Por más que nuestro servidor sea local y que el NetBeans lo haga transparente para nosotros, debemos entender que el Tomcat se ejecuta cuando ejecutamos la aplicación y que además posee una estructura de directorios (distinta a nuestra estructura de carpetas del proyecto) donde almacena las aplicaciones web que corre, archivos de configuración, paquetes de clases, etc.
- Al ejecutar el proyecto se abrirá el browser predeterminado con la página index.jsp (la que tiene el formulario):

- Si ingresamos nuestro nombre en la caja de texto y apretamos Enviar, el formulario se envía al servlet, quien se ejecuta y nos devuelve una nueva página, con un dato, en particular, cargado dinámicamente: nuestro nombre:

http://luauf.com/2008/05/21/introduccion-a-java-servlets-con-netbeans/

Ejemplo básico de Servlet en Netbeans.

En el siguiente ejemplo crearemos una aplicación que realice conversión de tipo de monedas en una página JSP utilizando Servlet.

Creación de proyecto.

Presionamos CTRL+Mayúsculas+N para crear un nuevo proyecto Web de tipo Web Application asignándole como nombre conversionMoneda.

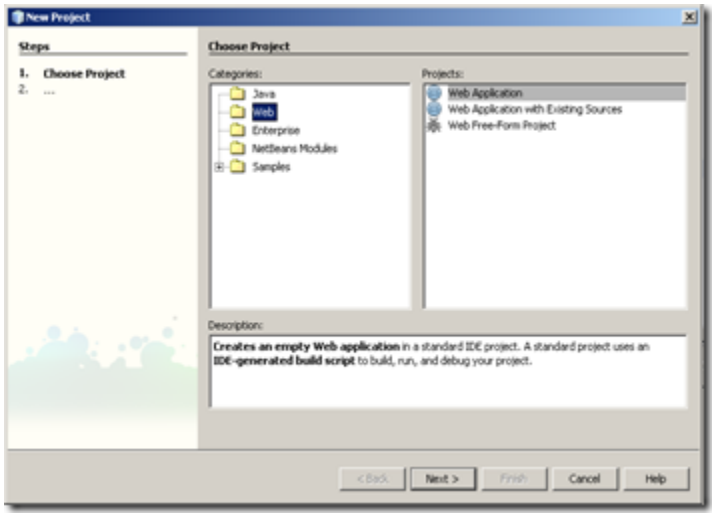


Ilustración 1: Seleccionar tipo de proyecto.

Seleccionamos cono server para nuestro proyecto GlassFish V2 y presionamos en Finish.

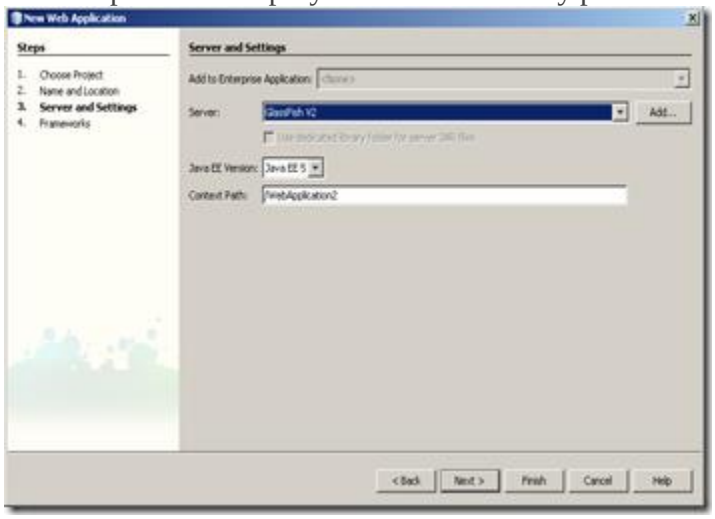


Ilustración 2: Seleccionar tipo de server para el proyecto web.

Modificación de páginas JSP.

Ingresamos en nuestro archivo de inicio haciendo doble clic en el y ingresamos el siguiente código en su interior.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
<title>Página de conversión de monedas</title>
</head>
<body>
<H1>Equivalencia de Monedas</H1>
Conozca la tasa de cambio de las diferentes monedas del mundo.Ingrese la cantidad de dinero a convertir en la primera ventana.Luego elija las monedas de cuales desea conocer la equivalencia.
<form action="convServlet" method="post">
<fieldset>
Convertir esta cantidad:  <input type="text" name="valor" value="${param.valor}"/>
<br><br>
DESDE
<select name="tipoA" size="4">
<option value="USD">Estados Unidos Dólares</option>
<option value="EUR">Europa Euros</option>
<option value="JPY">Japón Yenes</option>
<option value="CLP">Chile Pesos</option>
</select>
A: <select name="tipoB" size="4">
<option value="USD">Estados Unidos Dólares</option>
<option value="EUR">Europa Euros</option>
<option value="JPY">Japón Yenes</option>
<option value="CLP">Chile Pesos</option>
</select>
<br><br>
<input type="submit" value="Conversión"/>
</fieldset>
</form>
</body>
</html>
```

También crearemos una página llamada respuesta.jsp, la cual servirá para recibir los datos de la conversion y mostrarlos en el navegador.

Para crear una nueva página presionamos CTRL+N y seleccionamos el tipo de página JSP.

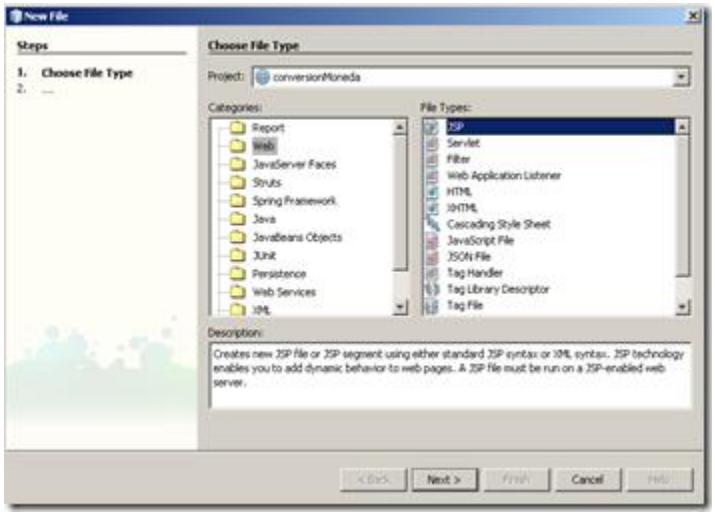


Ilustración 3: Creación página respuesta.jsp.

El código de la pagina será el que se muestra a continuación:

```
<html>
<head>
<meta http-equiv=”Content-Type” content=”text/html; charset=UTF-8“>
<title>JSP Page</title>
</head>
<body>
<h2>Conversión</h2>
<h3>${inicial} ${tipoA} equivalen a: ${final} ${tipoB}</h3>
</body>
</html>
```

Creación Servlet de la aplicación.

Creamos un nuevo archivo de tipo Servlet al que llamaremos convServlet y lo ingresaremos en un package llamado web.

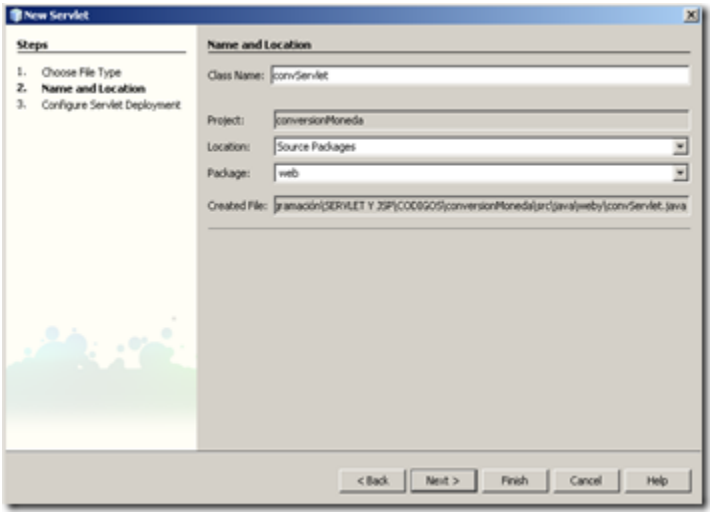


Ilustración 4: Creacion del servlet que regresa la conversion.

En el método llamado processRequest del servlet ingresaremos el siguiente código en el interior de su bloque try.

```
double CLP_USD=600.299, CLP_EUR=761.310, CLP_JPY=6.37478;
double JPY_CLP=0.156998, JPY_EUR=119.358, JPY_USD=94.1549;
double EUR_CLP=0.00131361, EUR_JPY=0.00836902, EUR_USD=0.788451;
double USD_CLP=0.00166592, USD_JPY=0.0106158, USD_EUR=1.26787;
double valor = Integer.parseInt(request.getParameter("valor"));
String tipoA = request.getParameter("tipoA");
String tipoB = request.getParameter("tipoB");
double inicial = valor;
if("USD".equals(tipoA)){
if("EUR".equals(tipoB)){
valor = valor*EUR_USD;
}else if("JPY".equals(tipoB)){
valor = valor*JPY_USD;
}else if("CLP".equals(tipoB)){
valor = valor*CLP_USD;
}
}else if("EUR".equals(tipoA)){
if("USD".equals(tipoB)){
valor = valor*USD_EUR;
}else if("JPY".equals(tipoB)){
valor = valor*JPY_EUR;
}else if("CLP".equals(tipoB)){
valor = valor*CLP_EUR;
}
```

```
}else if("JPY".equals(tipoA)){
if("USD".equals(tipoB)){
valor = valor*USD_JPY;
}else if("EUR".equals(tipoB)){
valor = valor*EUR_JPY;
}else if("CLP".equals(tipoB)){
valor = valor*CLP_JPY;
}
}else if("CLP".equals(tipoA)){
if("USD".equals(tipoB)){
valor = valor*USD_CLP;
}else if("EUR".equals(tipoB)){
valor = valor*EUR_CLP;
}else if("JPY".equals(tipoB)){
valor = valor*JPY_CLP;
}
}
request.setAttribute("final", valor);
request.setAttribute("inicial",inicial);
request.setAttribute("tipoA", tipoA);
request.setAttribute("tipoB", tipoB);
RequestDispatcher rd = request.getRequestDispatcher("respuesta.jsp");
rd.forward(request, response);
```

Ejecutando la aplicación.

Ejecutamos la aplicación presionando la tecla F6, se nos mostrará el navegador con la primera página (index.jsp) que contiene lo que se ve a continuación:

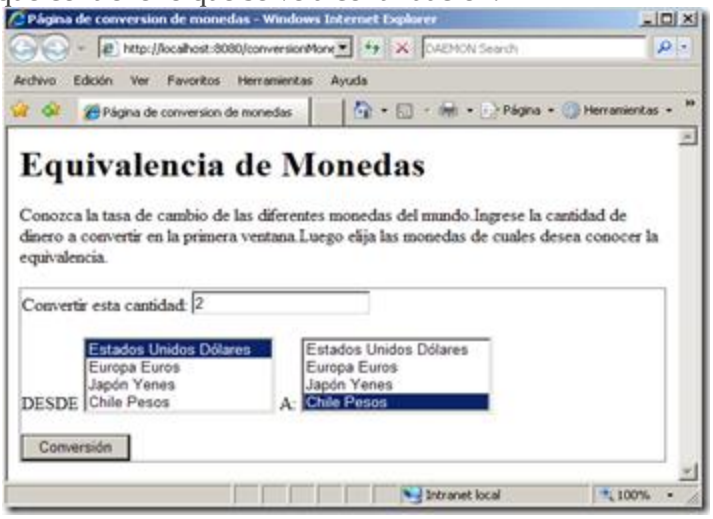


Ilustración 5: Seleccionar tipo cambio.

Después de seleccionar los datos se mostrara la conversion de moneda:

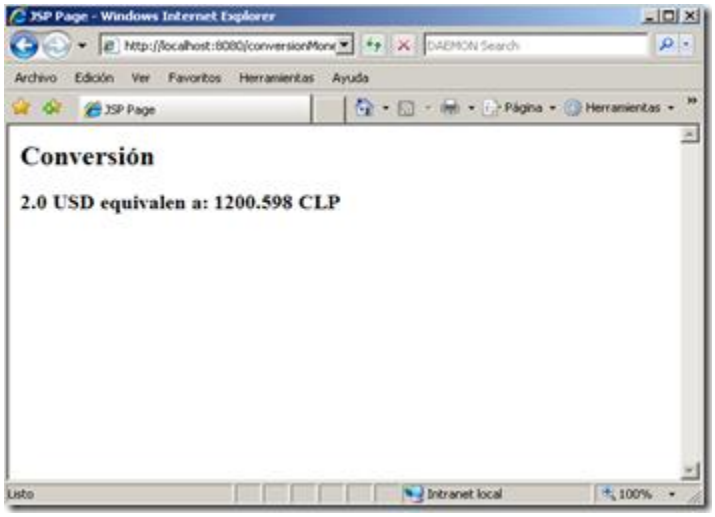


Ilustración 6: Conversión de moneda.

Código del Proyecto.

El código fuente del proyecto utilizado en este tutorial se encuentra aquí https://javalea.wordpress.com/descarga-codigos/