Materia: Tecnologías para la Web.
Tema: Introducción a los Servlets de Java.

Objetivo.

Con base en este ejercicio, diseñar una aplicación web que permita evaluar una expresión regular desde el navegador.

Introducción.

Los servlets de Java son programas que se ejecutan en un servidor Web o de aplicaciones y actúan como una capa intermedia entre las solicitudes procedentes de un navegador Web u otro cliente HTTP y bases de datos o aplicaciones en el servidor HTTP.

Los servlets pueden recopilar información de los usuarios con formularios de páginas web, presentar registros de una base de datos u otra fuente y crear páginas web dinámicamente. A menudo tienen el mismo propósito que los programas implementados utilizando la interfaz común de pasarela (Common Gate Interface CGI). Los servlets ofrecen las siguientes ventajas:

- El rendimiento es significativamente mejor. Se ejecutan dentro del espacio de direcciones de un servidor Web. No es necesario crear un proceso separado para manejar cada solicitud de cliente.
- Son independientes de la plataforma porque están escritos en Java. Son de confianza, ya que el administrador de seguridad de Java en el servidor impone un conjunto de restricciones para proteger los recursos en una máquina servidor.
- Poseen funcionalidad completa de las bibliotecas de clases Java. Pueden comunicarse con applets, bases de datos u otro software a través de los sockets y los mecanismos de RMI.

Además, los servlets realizan las siguientes tareas:

- Leer los datos explícitos enviados por los clientes (navegadores). Leer los datos implícitos o enviar la respuesta de peticiones HTTP enviados por los clientes (navegadores). Esto incluye cookies, tipos de medios y esquemas de compresión que el navegador entiende, y así sucesivamente.
- Procesar los datos y generar los resultados. Puede interactuar con una base de datos, ejecutar una llamada RMI o CORBA, invocar un servicio Web o calcular la respuesta directamente.
- Enviar los datos explícitos (es decir, el documento) a los clientes (navegadores). Este documento puede ser enviado en una variedad de formatos, HTML o XML, binario (imágenes GIF), y otros.

Los servlets se crean utilizando los paquetes `javax.servlet` y `javax.servlet.http`.
El ciclo de vida del servlet se define como el proceso entero de su creación hasta su destrucción:

1. El servlet se inicializa llamando al método `init`().
2. El servlet llama al método `service`() para procesar la solicitud de un cliente.
3. El servlet se termina llamando al método `destroy`().
4. Finalmente, el servlet es basura recolectada por el recolector de basura de la JVM.

El método `service`() es llamado por el contenedor e invoca los métodos `doGet`, `doPost`, `doPut`, `doDelete`, etc., según corresponda. Por lo tanto, no es necesario invocar al método `service`() pero se puede sobrescribir a `doGet`() o `doPost`(), dependiendo del tipo de solicitud que reciba del cliente. Los métodos `doGet`() y `doPost`() son los más utilizados en cada solicitud de servicio.

Los servlets manejan el análisis de datos de forma automática usando los siguientes métodos dependiendo de la situación:

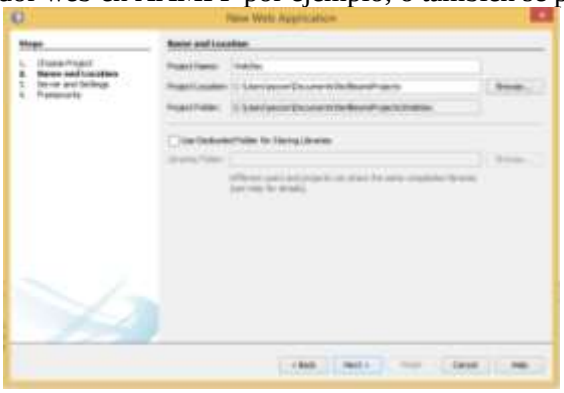
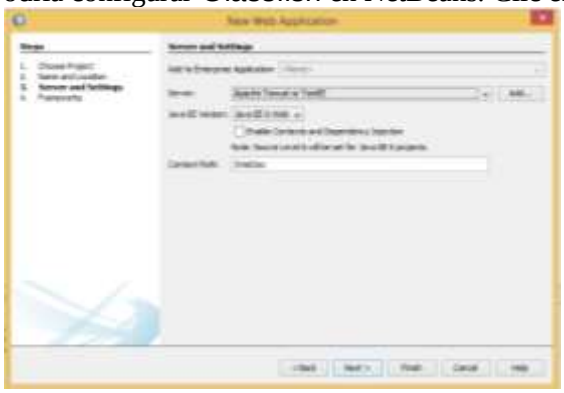
- `getParameter`(): Invoca a `request.getParameter`() para obtener el valor de un parámetro de formulario.
- `getParameterValues`(): Se invoca si el parámetro aparece más de una vez y devuelve varios valores.
- `getParameterNames`(): Se invoca si se desea una lista completa de todos los parámetros de la solicitud actual.

Desarrollo.

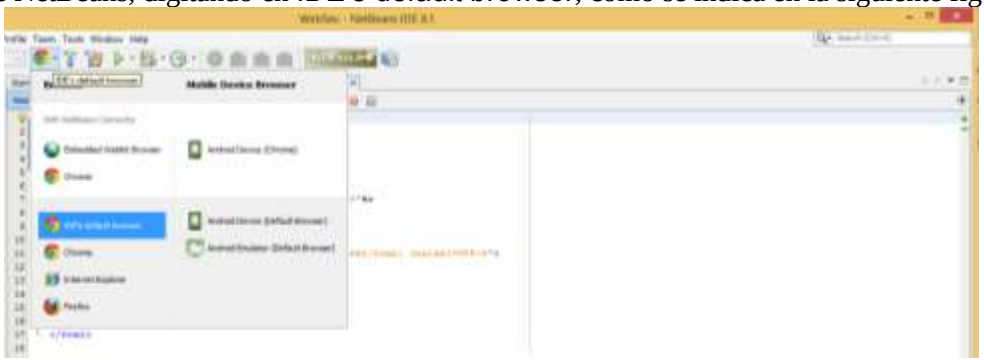
PARTE I.

1. Abrir NetBeans y crear un nuevo proyecto. En el menú principal, seleccionar File > New Project… >, para abrir la ventana New Project. Seleccionar Java Web > Web Application, respectivamente. Clic en Next:


2. En la ventana New Web Application, en el nombre del proyecto ingresar `WebSec`. Clic en Next. Enseguida, seleccionar el servidor web, en este caso Apache Tomcat or TomEE; para ello se debería tener iniciado este servidor web en XAMPP por ejemplo, o también se podría configurar Glassfish en NetBeans. Clic en Finish.

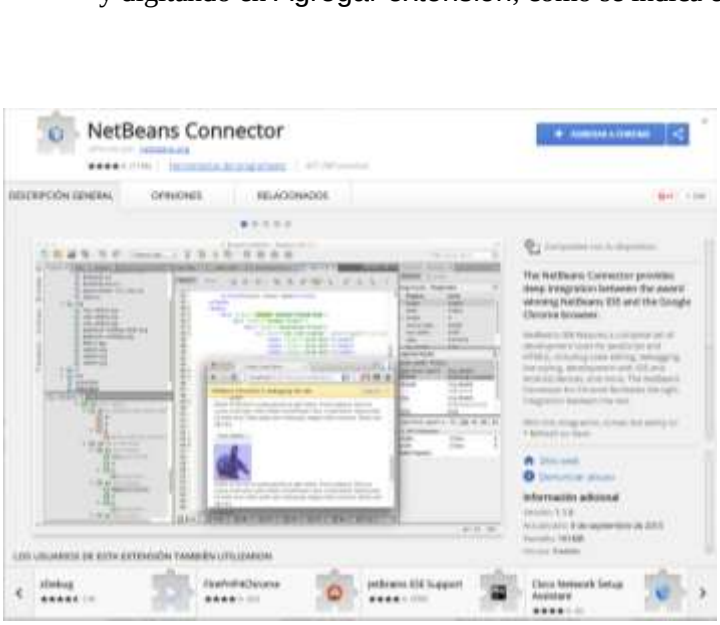



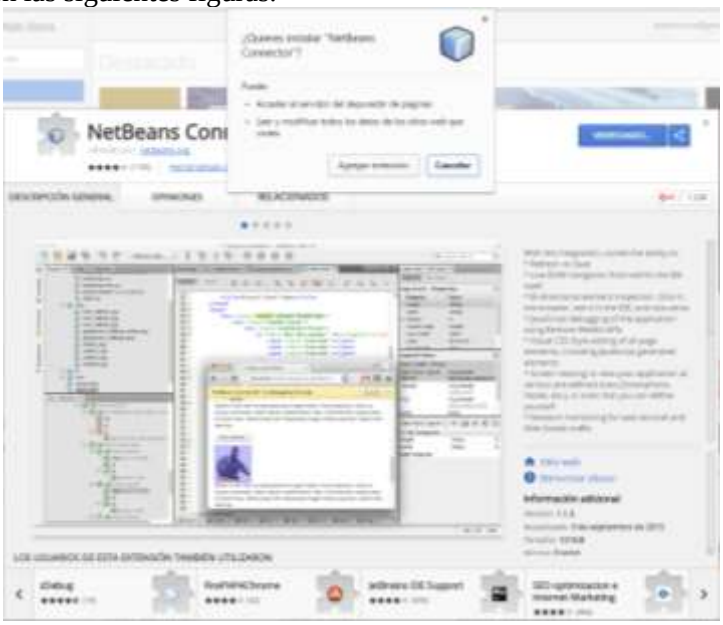
3. Antes de ejecutar la aplicación web, se puede seleccionar un navegador determinado seleccionando uno en el menú principal de NetBeans, digitando en IDE's default browser, como se indica en la siguiente figura:



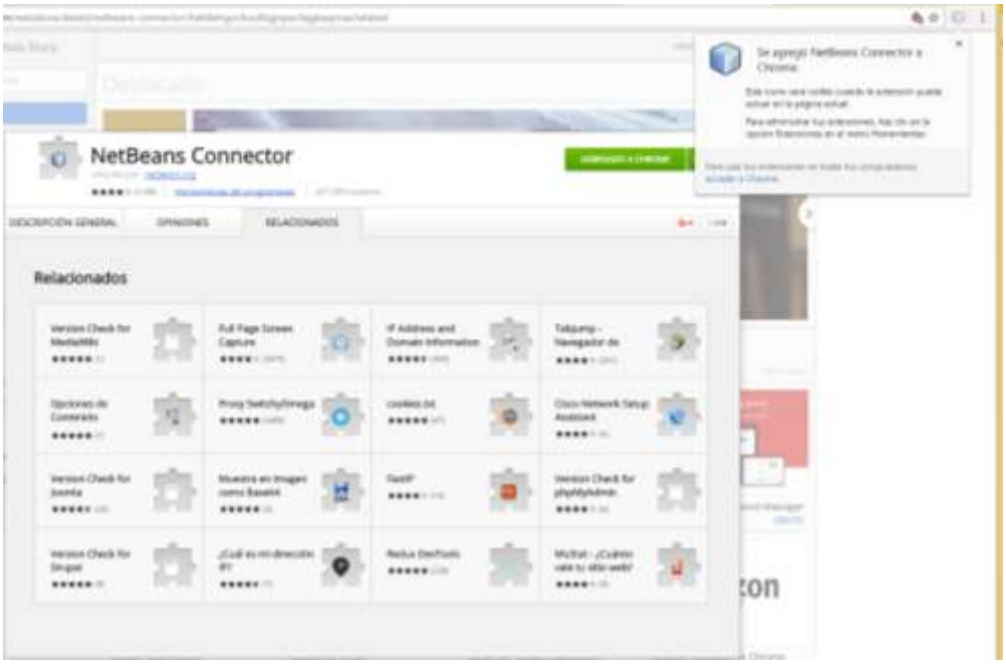
4. Si no se tiene instalada la extensión para el navegador Chrome, se mostrará una ventana que indica la necesidad de instalar el NetBeans Connector Extension. Clic en Go to the Chrome Web Store, como se indica enseguida:

5. Ahora, se muestra la ventana de la instalación de la extensión NetBeans Connector Extension. Clic en la opción + AGREGAR A CHROME en la esquina superior derecha. Enseguida, en la ventana auxiliar aceptar la instalación y digitando en Agregar extensión, como se indica en las siguientes figuras.





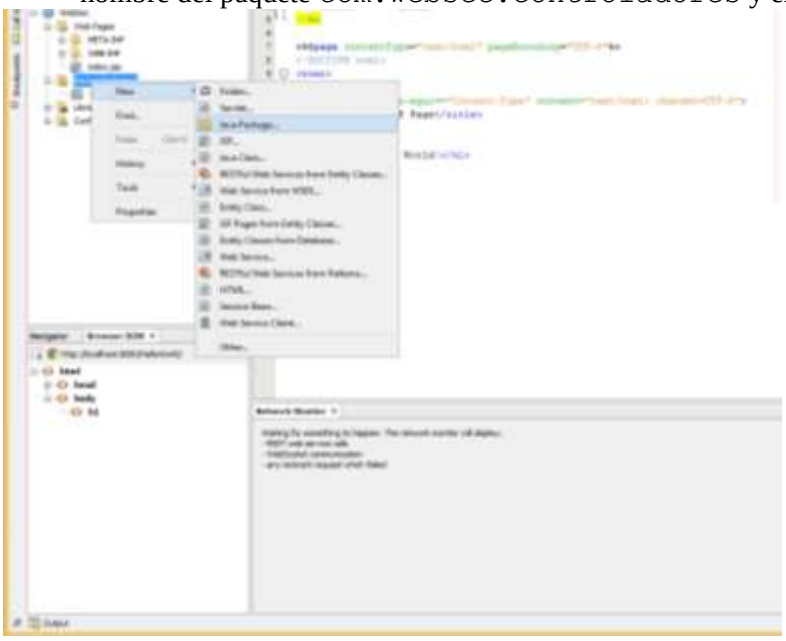
6. La instalación tarda algunos instantes y al terminar se muestra un mensaje de agregado del NetBeans Connector a Chrome.

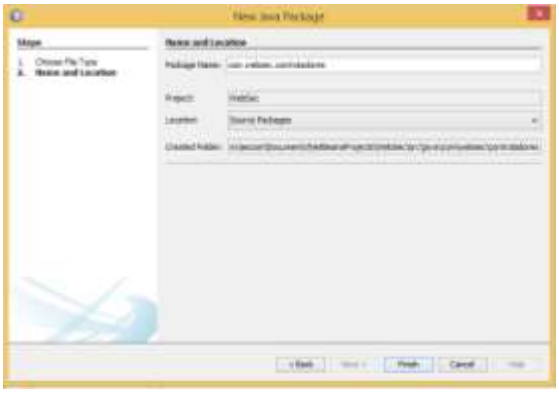


7. Clic derecho del mouse y seleccionar Run en el nombre del proyecto WebSec para ejecutar la aplicación web.

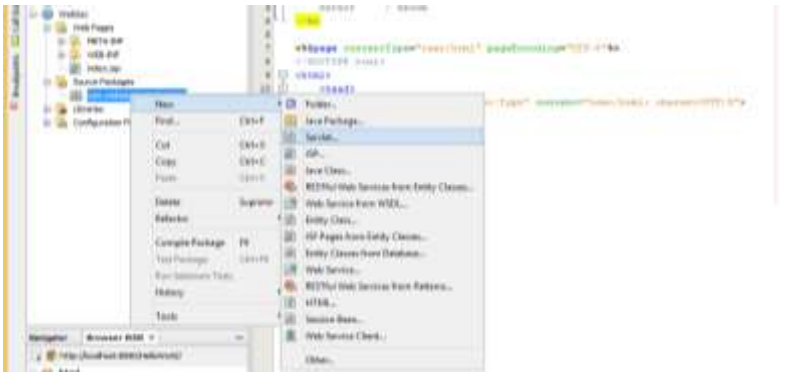
PARTE II.
Construcción del servlet.

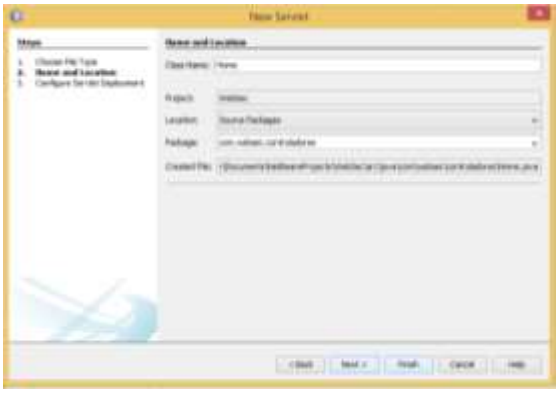
8. Para construir un servlet se crea primero un paquete. Clic derecho en la carpeta Source Package que pertenece al proyecto WebSec. Seleccionar New > New Package. En la ventana New Java Package, ingresar en el nombre del paquete `com.websec.controladores` y enseguida clic en Finish.



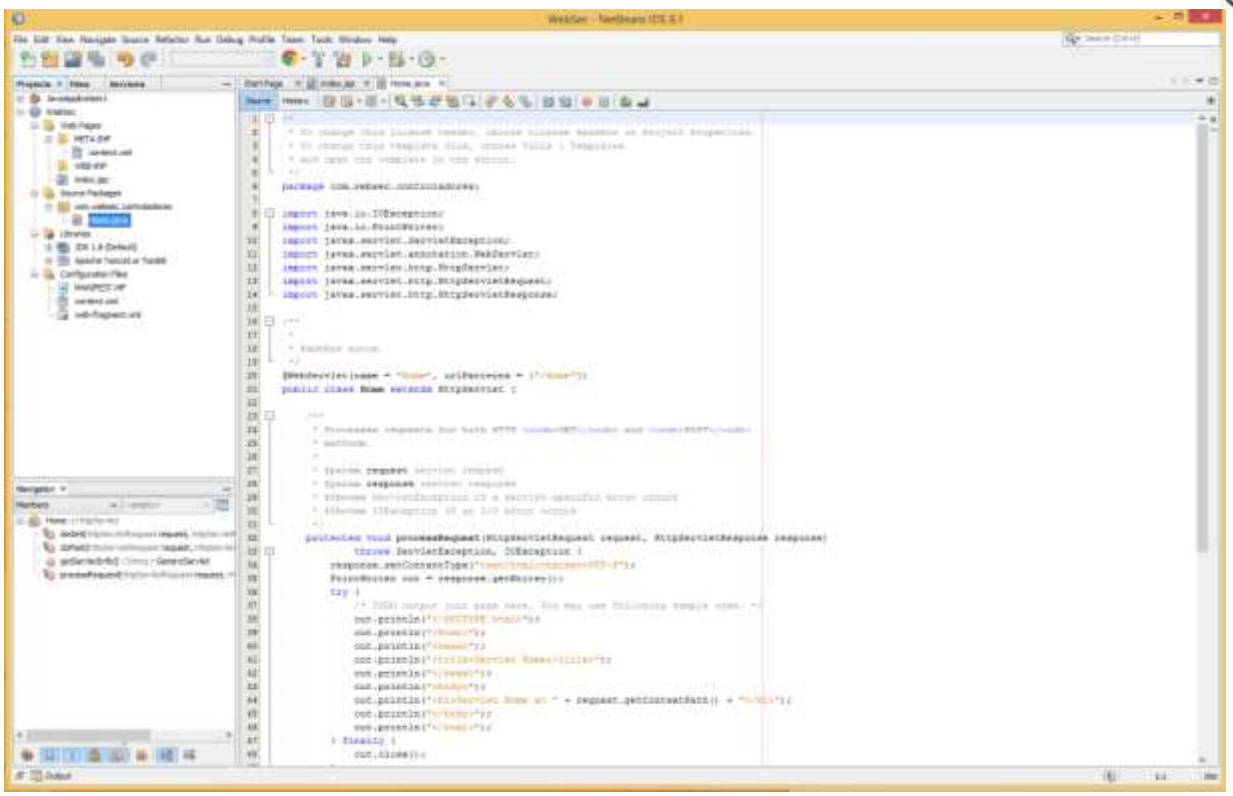


9. Clic derecho en el paquete creado com.websec.controladores para seleccionar New > Servlet. En la ventana mostrada ingresar el nombre de la clase del servlet, en este caso `Home`. Clic en Finish.





10. En el editor de texto de NetBeans se muestra el código fuente del servlet básico.



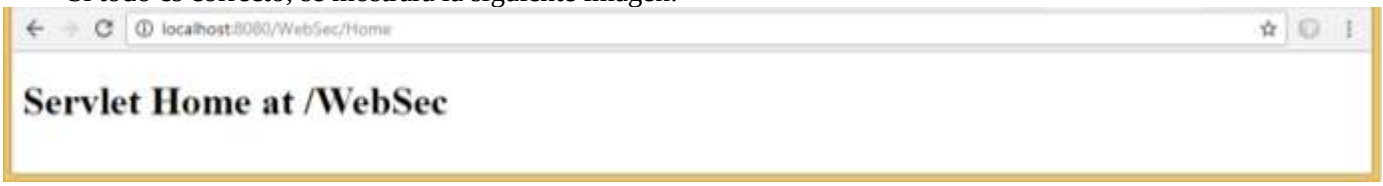
El código principal es similar al siguiente en donde se aprecian los métodos principales del servlet:

```
package com.websec.controladores;
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 = "Home", urlPatterns = {"/Home"})
public class Home extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet Home</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet Home at " + request.getContextPath() + "</h1>");
            out.println("</body>");
            out.println("</html>");
        } finally {
            out.close();
        }
    }
```

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

11. Para ejecutar el servlet, en el campo de texto del navegador ingresar la siguiente URL: `http://localhost:8080/WebSec/Home`

    Si todo es correcto, se mostrará la siguiente imagen:



PARTE III.
Modificación del código del servlet.

12. Una aplicación básica de modificación del código del servlet Home.java es el paso de variables desde el servlet al JSP. Para ello se define la variable `msg` de tipo `String`. El objeto `request` de `HttpServletRequest` y su método `setAttribute`() asigna el atributo `msg` y el objeto `rd` de `RequestDispatcher` carga el archivo JSP.

Cambiar todo el contenido de la clase `Home` por el siguiente código:

```
public class Home extends HttpServlet
{
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
    {
        String msg = "Hola Mundo!";
        request.setAttribute("msg", msg);
        RequestDispatcher rd = request.getRequestDispatcher("/index.jsp");
        rd.forward(request, response);
    }
}
```

Además, agregar la siguiente línea para la importación de la clase `RequestDispatcher`:

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

13. Modificar el código del archivo JSP index.jsp. Agregar la siguiente línea entre las etiquetas `<body>` y `</body>`:

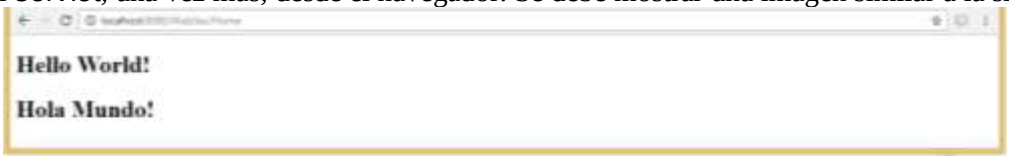
```
<h1>${msg}</h1>
```

Si el archivo index.jsp no existe, se crea digitando clic derecho en la carpeta Web Pages del proyecto WebSec y seleccionando New > JSP. Ingresar `index.jsp` en el nombre del archivo. Clic en Finish.

En el editor se muestra el código básico del archivo JSP:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello World!</h1>
        <h1>${msg}</h1>
    </body>
</html>
```

14. Ejecutar el servlet, una vez más, desde el navegador. Se debe mostrar una imagen similar a la siguiente:



NOTA: Generar un reporte completo documentado con las imágenes pertinentes de lo obtenido en cada parte del ejercicio.