



Servlets de Java

I- Contenido

Introducción

Los servlets, son a los servidores lo que los applets a los browsers. Se podría definir un servlet como un programa escrito en Java que se ejecuta en el marco de un servicio de red, (un servidor HTTP, por ejemplo), y que recibe y responde a las peticiones de uno o más clientes.

La tecnología Servlet proporciona las mismas ventajas del lenguaje Java en cuanto a portabilidad ("write once, run anywhere") y seguridad, ya que un servlet es una clase de Java igual que cualquier otra, y por tanto tiene en ese sentido todas las características del lenguaje.

Los servlets, una vez que son llamados por primera vez, quedan activos en la memoria del servidor hasta que el programa que controla el servidor los desactiva. De esta manera se minimiza en gran medida el tiempo de respuesta. Además, los servlets se benefician de la gran capacidad de Java para ejecutar métodos en ordenadores remotos, para conectar con bases de datos, para la seguridad en la información, etc. Se podría decir que las clases estándar de Java ofrecen resueltos muchos problemas que con otros lenguajes tiene que resolver el programador.



Características

Son independientes del servidor utilizado y de su sistema operativo, lo que quiere decir que a pesar de estar escritos en Java, el servidor puede estar escrito en cualquier lenguaje de programación, obteniéndose exactamente el mismo resultado que si lo estuviera en Java.

Los servlets pueden llamar a otros servlets, e incluso a métodos concretos de otros servlets. De esta forma se puede distribuir de forma más eficiente el trabajo a realizar. Por ejemplo, se podría tener un servlet encargado de la interacción con los clientes y que llamara a otro servlet para que a su vez se encargara de la comunicación con una base de datos. De igual forma, los servlets permiten redireccionar peticiones de servicios a otros servlets (en la misma máquina o en una máquina remota).

Los servlets pueden obtener fácilmente información acerca del cliente (la permitida por el protocolo HTTP), tal como su dirección IP, el puerto que se utiliza en la llamada, el método utilizado (GET, POST,...), etc.

Permiten además la utilización de cookies y sesiones, de forma que se puede guardar información específica acerca de un usuario determinado, personalizando de esta forma la interacción cliente-servidor. Una clara aplicación es mantener la sesión con un cliente.

Los servlets pueden actuar como enlace entre el cliente y una o varias bases de datos en arquitecturas cliente-servidor de 3 capas (si la base de datos está en un servidor distinto).

Asimismo, pueden realizar tareas de proxy para un applet. Debido a las restricciones de seguridad, un applet no puede acceder directamente por ejemplo a un servidor de datos localizado en cualquier máquina remota, pero el servlet sí puede hacerlo de su parte.

Los servlets permiten la generación dinámica de código HTML dentro de una propia página HTML. Así, pueden emplearse servlets para la creación de contadores, banners, etc.



Creando un Servlet

Antes de comenzar la elaboración de servlets es preciso importar los paquetes `javax.servlet`, `javax.servlet.http` y `java.io`. Debe de heredarse la clase abstracta `HttpServlet`, de la cual se implementan los métodos `doGet()` y `doPost()`, la clase debe situarse en un directorio específico del servidor o contenedor web.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class MuestraMensaje extends HttpServlet {
//variables y métodos
}
```

La clase `HttpServlet` ofrece una serie de métodos que se corresponden con distintos tipos de peticiones del protocolo HTTP, y además ofrece un método que tiene que ver con el ciclo de vida de un servlet.

Algunos de los métodos que podemos encontrar en la clase `HttpServlet` y que podemos sobrescribir en nuestro servlet son los siguientes:

- **`void doDelete(HttpServletRequest req, HttpServletResponse res):`**

Permite que el servlet trate una petición DELETE del protocolo HTTP. La operación DELETE permite a un usuario eliminar un documento o página Web del servidor.

- **`void doGet(HttpServletRequest req, HttpServletResponse res):`**

Permite que el servlet trate una petición GET del protocolo HTTP. Este método es de los típicos que podemos encontrar en un servlet.

- **`void doOptions(HttpServletRequest req, HttpServletResponse res):`**

Trata una petición OPTIONS del protocolo HTTP. La petición OPTIONS determina que métodos del protocolo HTTP soporta el servidor.



- **void doPost(HttpServletRequest req, HttpServletResponse res):**

Permite que el servlet trate una petición POST. Este método es de los típicos que podemos encontrar en un servlet.

- **void doPut(HttpServletRequest req, HttpServletResponse res):**

La petición PUT permite al usuario enviar un fichero al servidor de forma similar a como se envía un fichero a través del protocolo FTP (File Transfer Protocol).

- **void doTrace(HttpServletRequest req, HttpServletResponse res):**

Una petición TRACE devuelve las cabeceras enviadas junto con esta petición, de vuelta al cliente, por lo que se utiliza para depuración.

- **long getLastModified(HttpServletRequest req):**

Devuelve en milisegundos la hora y la fecha en la que el objeto HttpServletRequest se modificó por última vez.

- **void service(HttpServletRequest req, HttpServletResponse res):**

Recibe las peticiones estándar del protocolo HTTP y las redirecciona a los métodos doXXX adecuados definidos en la clase del servlet, se puede decir que es el punto común de entrada de todos los tipos de peticiones realizadas a un servlet. Desde el método service() se invocarán los distintos métodos doXXX de la clase HttpServlet. No es necesario ni recomendable sobrescribir este método. Este es el método que tiene que ver con el ciclo de vida definido por los servlets.

- **void service(ServletRequest req, ServletResponse res):**

Este método redirige todas las peticiones que realizan los clientes al método service() anterior. No hay ninguna necesidad para sobrescribir este método.



Se debe señalar que todos estos métodos, menos el método `getLastModified()`, lanzan dos excepciones: `javax.servlet.ServletException` y `java.io.IOException`.

Ciclo de vida del Servlet

Cuando el servlet se crea por primera vez se invoca el método `init()`, por lo tanto este método contendrá el código de inicialización del servlet.

Después de esto, cada petición realizada por un usuario sobre el servlet se traduce en un nuevo hilo de ejecución (thread) que realiza una llamada al método `service()`. Múltiples peticiones concurrentes normalmente generan múltiples hilos de ejecución que llaman de forma simultánea al método `service()`, aunque el servlet puede implementar un interfaz especial que permite que únicamente se pueda ejecutar un único hilo de ejecución al mismo tiempo.

El método `service()` invocará los métodos `doGet()` o `doPost()` o cualquier otro método `doXXX`, dependiendo del tipo de petición HTTP recibida.

Finalmente, cuando el servidor decide descargar el servlet, se ejecuta anteriormente el método `destroy()`.

Como se puede ver el ciclo de vida de un servlet sigue el siguiente esquema:

- Ejecución del método `init()` para inicialización del servlet.
- Sucesivas ejecuciones del método `service()` en distintos hilos de ejecución, que resultan en una llamada a un método `doXXX`.
- Ejecución del método `destroy()` para realizar labores de liberación de recursos.

El método `init()` es invocado cuando el servlet es creado y no es llamado de nuevo por cada petición que los usuarios realicen sobre el servlet. De esta forma es utilizado por inicializaciones que tiene lugar una única vez, en este aspecto es muy similar al método `init()` del ciclo de vida de los applets.



El servlet puede ser creado cuando un usuario utiliza por primera vez el servlet a través de la URL correspondiente o bien cuando se inicia la ejecución de servidor que contiene los servlets. La forma en la que se crea el servlet depende si el servlet se encuentra registrado en el servidor Web o no, si no se encuentra registrado en el servidor Web, el servlet se creará cuando el primer usuario lo invoque, pero si se encuentra registrado en el servidor se creará cuando se inicie la ejecución del servidor.

El método `init()` ofrece dos versiones, una de ellas no recibe ningún parámetro y la otra versión recibe como parámetro un objeto `ServletConfig`. La primera versión se utiliza cuando el servlet no necesita leer ninguna configuración que puede variar entre distintos servidores.

La definición de este método tiene el aspecto del siguiente código:

```
Public void init() throws ServletException{  
    //Codigo de inicializacion  
}
```



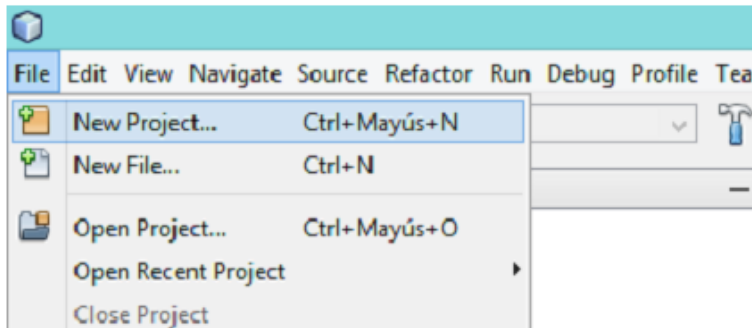
II- Procedimiento

Para desarrollar Servlets en NetBeans es preciso utilizar un contenedor Web, para efectos de esta práctica se utilizara el GlassFish. Aprovechando las virtudes de los entornos de desarrollo integrados, como NetBeans. Los programas de tipo servlet se construirán a partir de proyectos web de la IDE en cuestión.

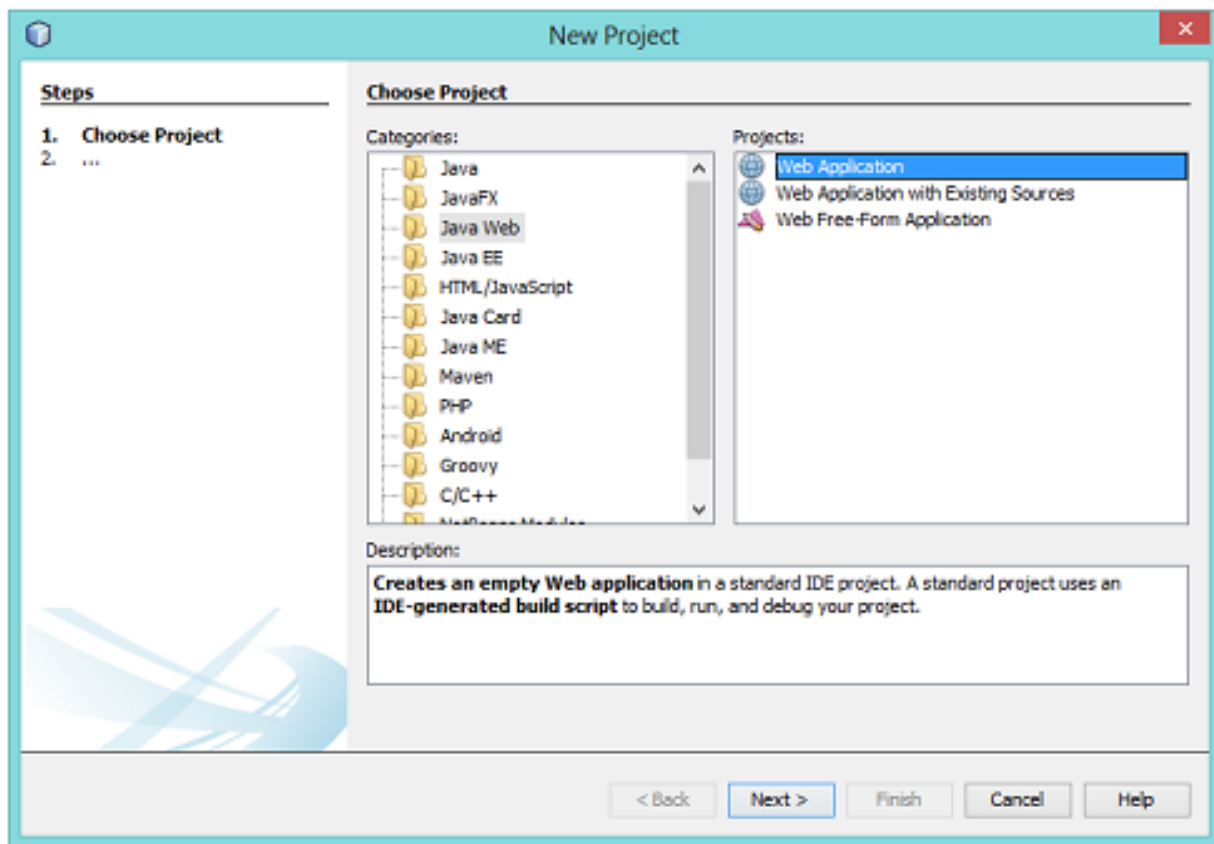
Antes es necesario conocer cómo se crearan todos nuestros ejemplos y ejercicios.

Creando Nuevo Proyecto Web Application

El primer paso es crear un proyecto web, para crearlo hay que ir al menú *File*, opción *New Project...*



Elegimos la categoría *Java Web* y la opción *Web Application* tal como se muestra en la imagen:





Ahora hacemos click en Next, asignamos o buscamos la localización del proyecto o *Project Location* y le asignamos el nombre o *Project Name*, el cual en nuestro caso tendrá **Prueba**. Tal como se muestra en la siguiente imagen:

New Web Application

Steps

1. Choose Project
2. **Name and Location**
3. Server and Settings
4. Frameworks

Name and Location

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

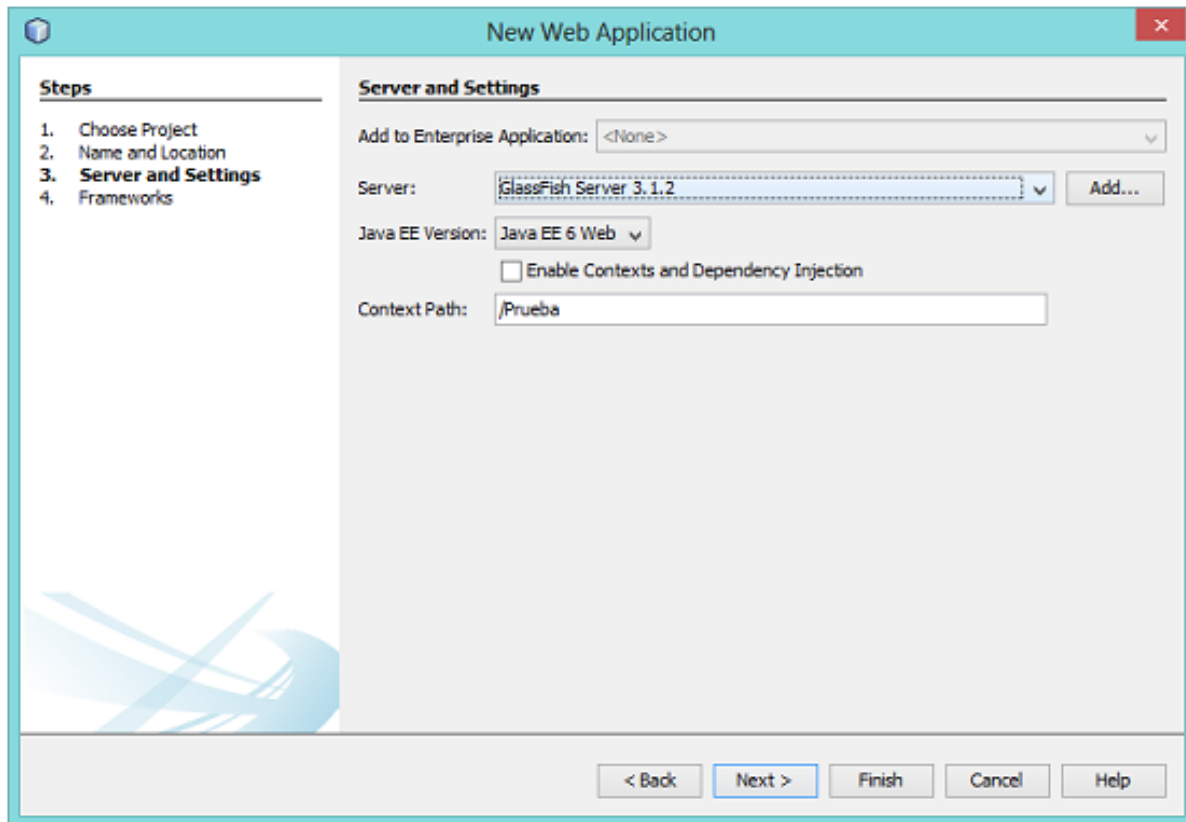
Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

< Back Next > Finish Cancel Help



Luego hacemos clic en Next y seleccionamos el servidor GlassFish por último clic en Finish.



Al hacer clic en Finish o Finalizar se nos creará el nuevo proyecto web.



Página principal Index

El proyecto incluye una carpeta *Web Pages* y una página jsp llamada *index* que será la principal o la primera en mostrarse en el navegador. Esta página principal contiene etiquetas HTML.

Agregaremos la siguiente estructura dentro del cuerpo del documento html (entre las etiquetas **body**):

```
<h1>Ejemplo Servlet</h1>
<form action="prueba" method="post">
  nombre: <input type="text" name="nombre" size=15><br>
  apellidos: <input type="text" name="apellidos" size=30><p>
  opinión de este sitio web<br>
  <input type="radio" checked name="opinion"
  value="buena">buena<br>
  <input type="radio" name="opinion" value="regular">regular<br>
  <input type="radio" name="opinion" value="mala">mala<p>
  comentarios <br>
  <textarea name="comentarios" rows=6 cols=40>
  </textarea><p>
  <input type="submit" name="botonenviar" value="enviar">
  <input type="reset" name="botonlimpiar" value="limpiar">
</form>
```

Diagram annotations: A blue circle with the number '1' points to the opening tag of the form (`<form action="prueba" method="post">`). A blue circle with the number '2' points to the `method="post"` attribute in the same tag.

El código anterior tiene las Etiquetas HTML necesarias para la creación de un formulario sencillo, con elementos básicos, del cual se pretende enviar la información que en él contenga hacia un servlet, que se encargará de procesar los datos, el cual crearemos a continuación llamado *prueba* (1) y el método por el cual se enviará será el **post** (2).

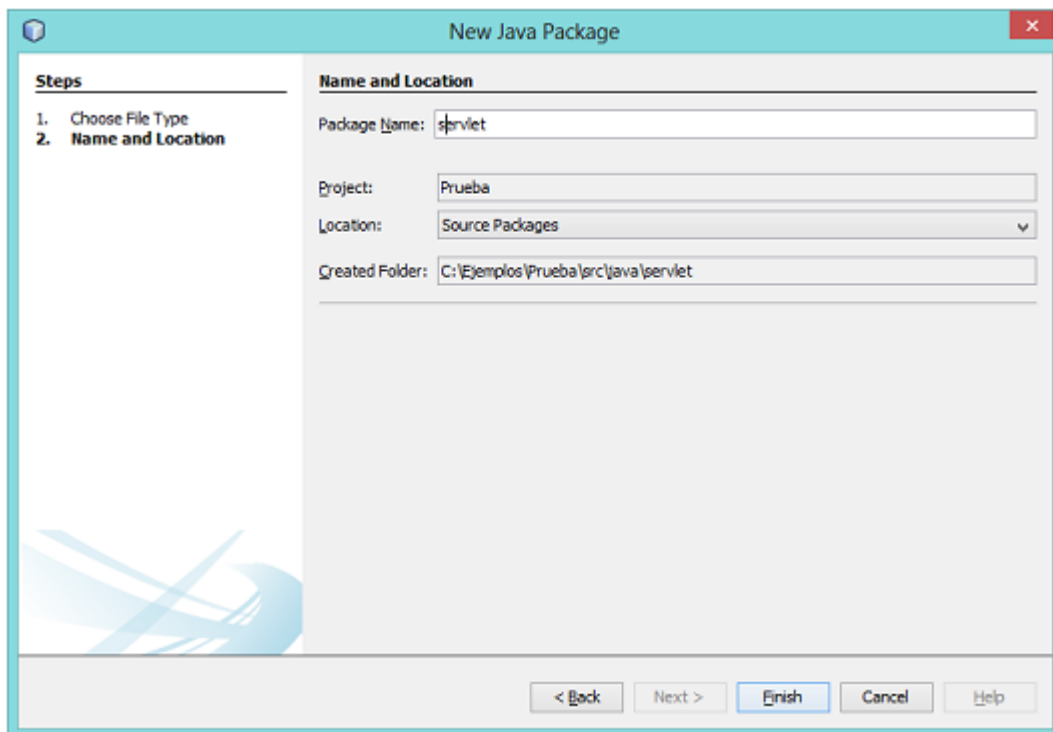


Creando Servlet

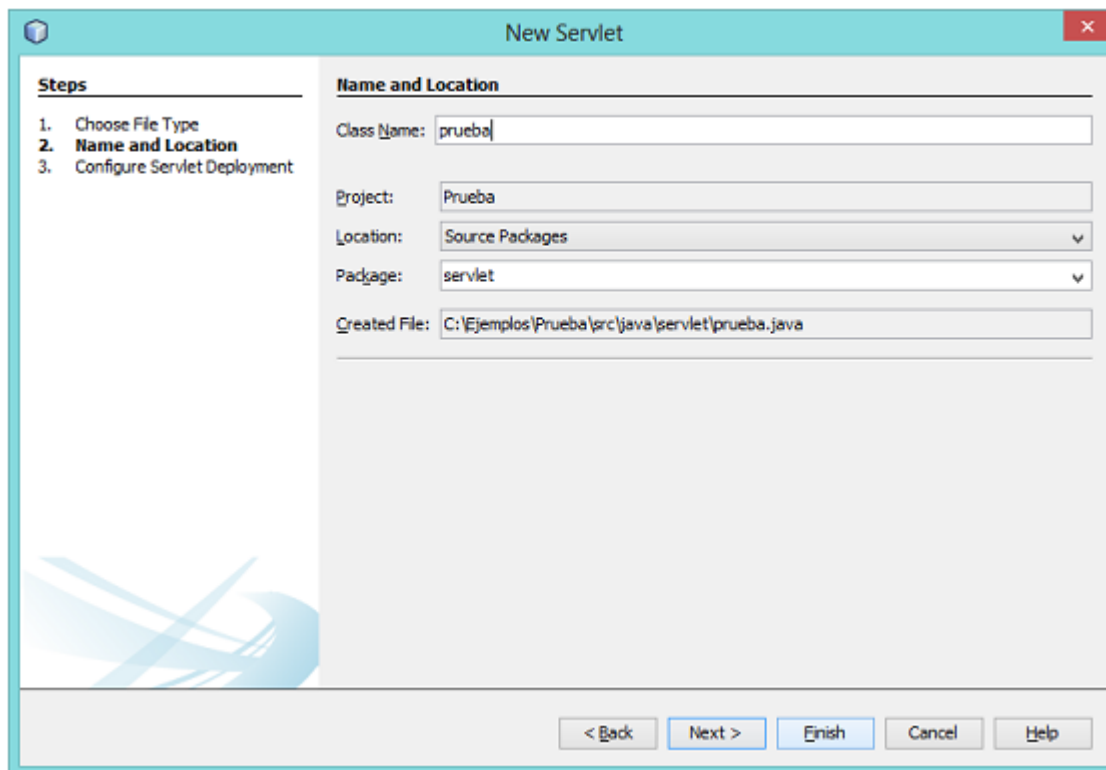
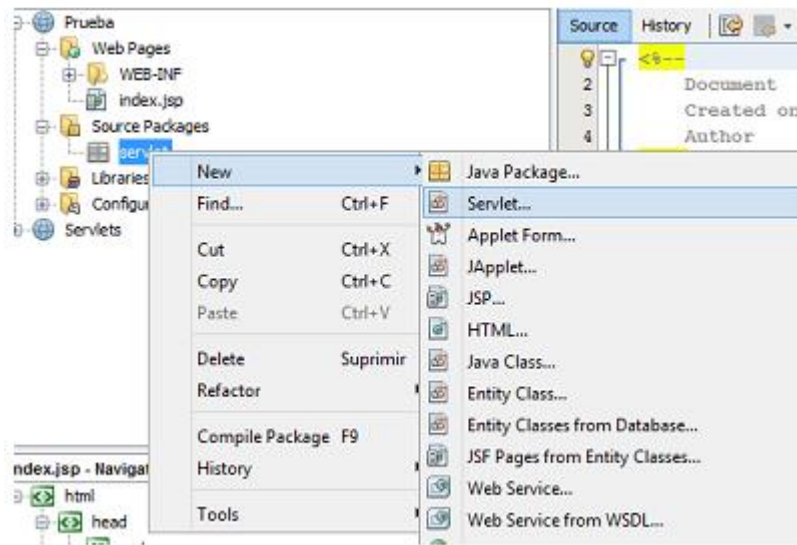
Para desarrollar nuestro servlet debemos hacer click derecho en la opción Source Packages del árbol del proyecto y seleccionar la opción New y luego Java Package.



Creamos un **Java Package** con el nombre *servlet*.



Procedemos a incorporar una clase de tipo servlet al paquete creado, clic derecho en el Java Package **servlet** new y seleccionamos **Servlet**. Y le asignamos el nombre *prueba*.



Automáticamente se creará el código del servlet, incluyendo la sobrecarga de los métodos `doGet()`, `doPost()`, `getServletInfo()`.



```

21  @WebServlet(name = "prueba", urlPatterns = {"/prueba"})
22  public class prueba extends HttpServlet {
23
24      /**
25       * Processes requests for both HTTP GET and POST
26       * methods.
27       *
28       * @param request servlet request
29       * @param response servlet response
30       * @throws ServletException if a servlet-specific error occurs
31       * @throws IOException if an I/O error occurs
32       */
33      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
34          throws ServletException, IOException {
35          response.setContentType("text/html;charset=UTF-8");
36          try (PrintWriter out = response.getWriter()) {
37              /* TODO output your page here. You may use following sample code. */
38              out.println("<!DOCTYPE html>");
39              out.println("<html>");
40              out.println("<head>");
41              out.println("<title>Servlet prueba</title>");
42              out.println("</head>");
43              out.println("<body>");
44              out.println("<h1>Servlet prueba at " + request.getContextPath() + "</h1>");
45              out.println("</body>");
46              out.println("</html>");
47          }
48      }
49
50      HttpServlet methods. Click on the + sign on the left to edit the code.
88

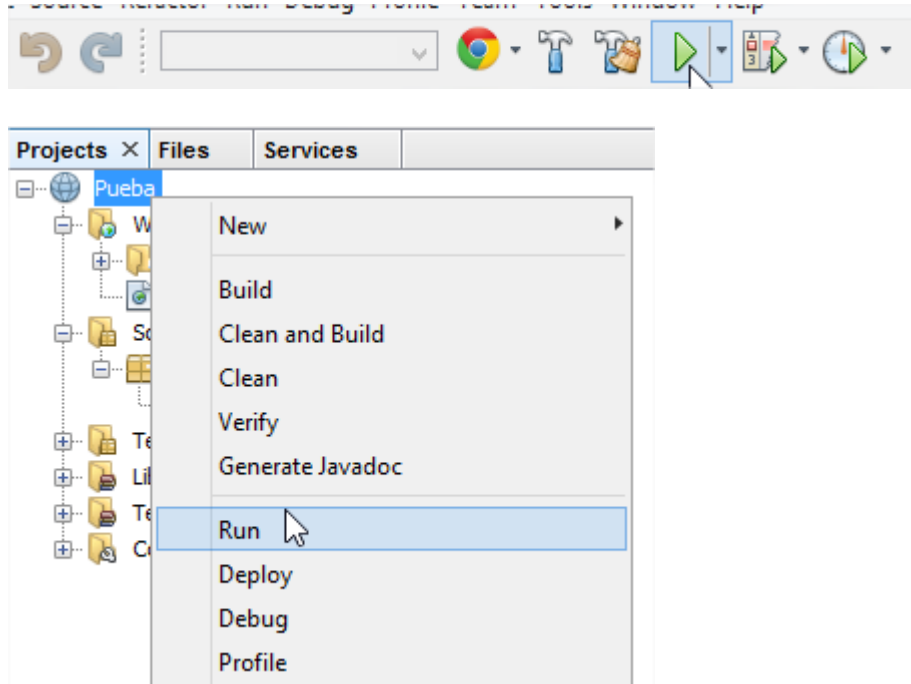
```

- (1) Lo que queremos mostrar en el navegador deberá estar escrito en HTML (el lenguaje de marcado que interpreta el navegador) por lo que si queremos mostrar un contenido desde una clase Servlet debemos imprimir las etiquetas HTML haciendo uso de *out.println("código HTML")*.
- (2) Si se desea agregar o modificar el código de los métodos sobrecargados doGet(), doPost(), getServletInfo(). Estos se encuentran al expandir el código al darle clic en el signo [+]
- (3) El código que se encuentra más abajo será agregado antes del cierre de la etiqueta **body**

Ejecución de la aplicación en el navegador



Para ver en ejecución el servlet es necesario darle clic (si el proyecto que deseamos ejecutar es el activo) a la opción **Run Project**, o darle clic derecho sobre el proyecto y la opción **Run**.





Capturando datos del formulario

Ya hemos creado el servlet, agregaremos el código necesario para capturar los datos que son enviados desde el formulario, para ello se agregara lo siguiente en el punto indicado arriba:

```
String nombre=request.getParameter("nombre");
String apellidos=request.getParameter("apellidos");
String opinion=request.getParameter("opinion");
String comentarios=request.getParameter("comentarios");
out.println("<h2>Valores recogidos del ");
out.println("formulario: </h2>");
out.println("<strong>Nombre: </strong>" + nombre);
out.println("<br><strong>Apellido: </strong>" + apellidos);
out.println("<br><br><strong>Opini&ocirc;n: </strong>" + opinion);
out.println("<br><strong>Comentarios: </v>" + comentarios);
```

- (1) Capturamos los valores de los elementos del formulario que se han nombrado de esa forma y lo almacenamos en una variable de tipo String.
- (2) Mostramos los datos imprimiéndolo en código HTML.

Para hacer la prueba nuevamente solo es necesario escribir en la url del navegador Web lo siguiente.

<http://localhost:8080/Prueba>



III- Ejercicios

Mostrar en una página web haciendo uso de servlet:

1. Crear un formulario de Registro de usuario básico al enviar los datos el servlet manejará esos datos mostrando en pantalla:

Gracias por registrarse en nuestro sitio <Nombre y apellido>

En estos momentos se ha enviado a su correo <correo> la información de su cuenta.

Su usuario es: <usuario>

Su contraseña: <contraseña>

En caso de olvidar estos datos guarde este correo.

2. Tomando en cuenta la información calcular:

El área de un triángulo cuyos lados son, a, b, c se puede calcular por la fórmula

$$A = \sqrt{p(p-a)(p-b)(p-c)}$$

Donde $p = (a + b + c) / 2$. Crear un HTML y escribir un servlet que lea las longitudes de los tres lados de un triángulo y calcule el área del triángulo.