



Universidad de
Oviedo

EPUU
Universidad de Oviedo

SERVLETS

Enol García González
Universidad de Oviedo
10 de noviembre de 2025

CONTENIDOS

- 1 Ciclo de vida
- 2 Registro de servlets
- 3 Paso de parámetros
- 4 Ámbitos de las variables
 - Sesión
 - Contexto

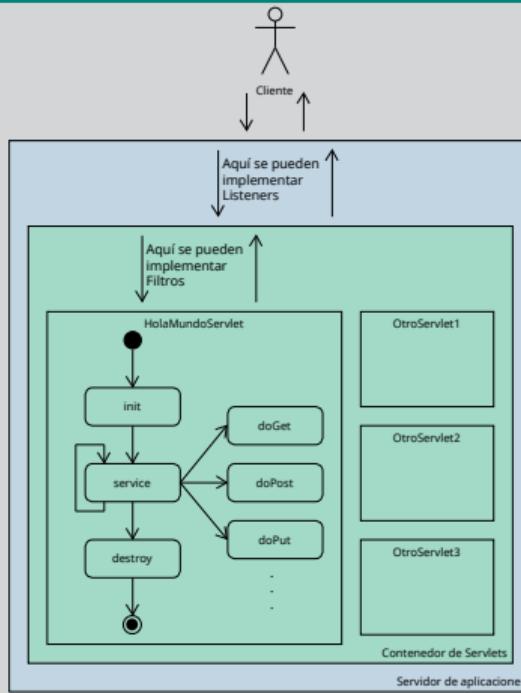
SERVLETS

- Un servlet es un programa Java que acepta peticiones HTTP y genera respuestas utilizando también el protocolo HTTP.
 - En JEE la interface Servlet define el comportamiento mínimo que debe incorporar un Servlet compuesto por los métodos: init, getServletInfo, getServletConfig, destroy y service.
 - Para facilitar la implementación hay una clase abstracta llamada GenericServlet que define un comportamiento básico para init, getServletInfo, getServletConfig y destroy. Esta clase permite que no haya que perder el tiempo en configuraciones del servlet y centrarnos sólo en su funcionalidad.
 - Además, hay otra clase HttpServlet (que es la que usaremos en prácticas) redefine el service y lo separa en diferentes métodos: doGet, doPost, doPut, doDelete, etc. En función del tipo de petición recibida.
-

SERVLETS – CICLO DE VIDA

- ① El servidor de aplicaciones procesa la petición y se la envía al contenedor de Servlets
 - ② El contenedor de Servlets busca el Servlet encargado de procesarla y llama al método service del método concreto.
 - ③ En nuestro caso se llama al método service de la clase HttpServlet. En este punto puede haber dos situaciones:
 - Si tenemos un método doXXX definido para ese método HTTP se invocará.
 - Si no está creado se devolverá un mensaje de error.
 - ④ Una vez procesado el método service, el objeto ServletResponse que se le pasa tendrá la información necesaria para la respuesta. Ese objeto sigue la misma ruta al revés hasta el cliente.
-

SERVLETS – CICLO DE VIDA



SERVLETS – EJEMPLO DE UN HOLA MUNDO

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

public class HolaMundoServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><head>");
        out.println("<title>Servlet Hola Mundo </title>");
        out.println("</head><body>");
        out.println("<h1>Hola Mundo!</h1>");
        out.println("</body></html>");
    }
}
```

SERVLETS – REGISTRO

Con el fichero web.xml

```
<servlet>
  <servlet-name>HolaMundo</servlet-name>
  <servlet-class>uo.sdi.servlet.
    HolaMundoServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>HolaMundo</servlet-name>
  <url-pattern>/HolaMundoCordial</url-
    pattern>
</servlet-mapping>
```

Con una anotación

```
@WebServlet(name = "HolaMundo",
  urlPatterns = {"/HolaMundoCordial"})
public class HolaMundoServlet extends
  HttpServlet {
  /**
   * ...
  }
```

SERVLETS – PARÁMETROS

```
<form action="http://server/app/HelloWorld" method="POST">
    Nombre: <input type="text" name="NombreUsuario" />
    <input type="submit" value="Aceptar" />
</form>
```

```
public class HelloWorld extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String nombre = (String) request.getParameter("NombreUsuario")

    }
}
```

SERVLETS – ÁMBITOS DE LAS VARIABLES

Request

- Sólo vigente mientras se realiza la petición
- Sólo el Servlet que procesa la petición ve los datos

Sesión

- Vigente mientras no se destruya la sesión
- Sólo el Servlet que procesa la petición ve los datos

Contexto

- Vigente durante toda la ejecución del proyecto
- Todos los servlets comparten su información

SERVLETS – SESIÓN

Uno de los mayores problemas del protocolo HTTP es que no mantiene el estado.
Complica guardar las acciones del usuario.

Posibles soluciones:

- Información en la URL
- Campos ocultos
- Cookies

SERVLETS – HTTPSESSION

Al trabajar con Servlets tenemos una solución técnica que nos facilita la gestión de la sesión: El objeto **HttpSession**

- HttpSession almacena objetos en el lado del servidor
- Cada usuario tendrá asociada un objeto HttpSession con un identificador único
- En el cliente se crea una cookie con el identificador de la sesión
- En cada petición, el cliente incluye su identificador de sesión

SERVLETS – USO DE HTTPSESSION

Al desarrollar los servlets podemos utilizar el objeto HttpSession con el método `HttpServletRequest.getSession(true)`. El valor true hará que la sesión se cree en caso de no existir.

La sesión se cerrara automáticamente después de un tiempo sin peticiones o al llamar al método `HttpSession.invalidate()`.

SERVLETS – USO DE HTTPSESSION

Los dos método más importantes, que nos permiten gestionar la información guardada en la sesión son:

- `setAttribute` para registrar un atributo o modificar su valor
- `getAttribute` para recuperar su valor

SERVLETS – CONTEXTO

Es común a todos los Servlets, nos permite compartir información y objetos entre los distintos usuarios.

Se accede con el objeto ServletContext. Sus dos métodos más importantes son:

- `setAttribute` para registrar un atributo o modificar su valor
- `getAttribute` para recuperar su valor

SERVLETS – EJEMPLO CONTADOR

```
public class ContadorServlet extends HttpServlet {  
    protected void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        ServletContext contexto=request.getServletContext();  
  
        Integer contador = (Integer) contexto.getAttribute("contador");  
        if (contador == null)  contador = 0;  
        contador++;  
        contexto.setAttribute("contador", contador);  
    }  
}
```

SERVLETS – VIDA DE LAS VARIABLES

