Sesión 7: Servlets

Índice

- Concepto de servlet
- Ciclo de vida de un servlet
- Estructura básica de un servlet
- Llamar a un servlet
- Ejemplos básicos

Servlets

- Concepto de servlet
- Ciclo de vida de un servlet
- Estructura básica de un servlet
- Llamar a un servlet
- Ejemplos básicos

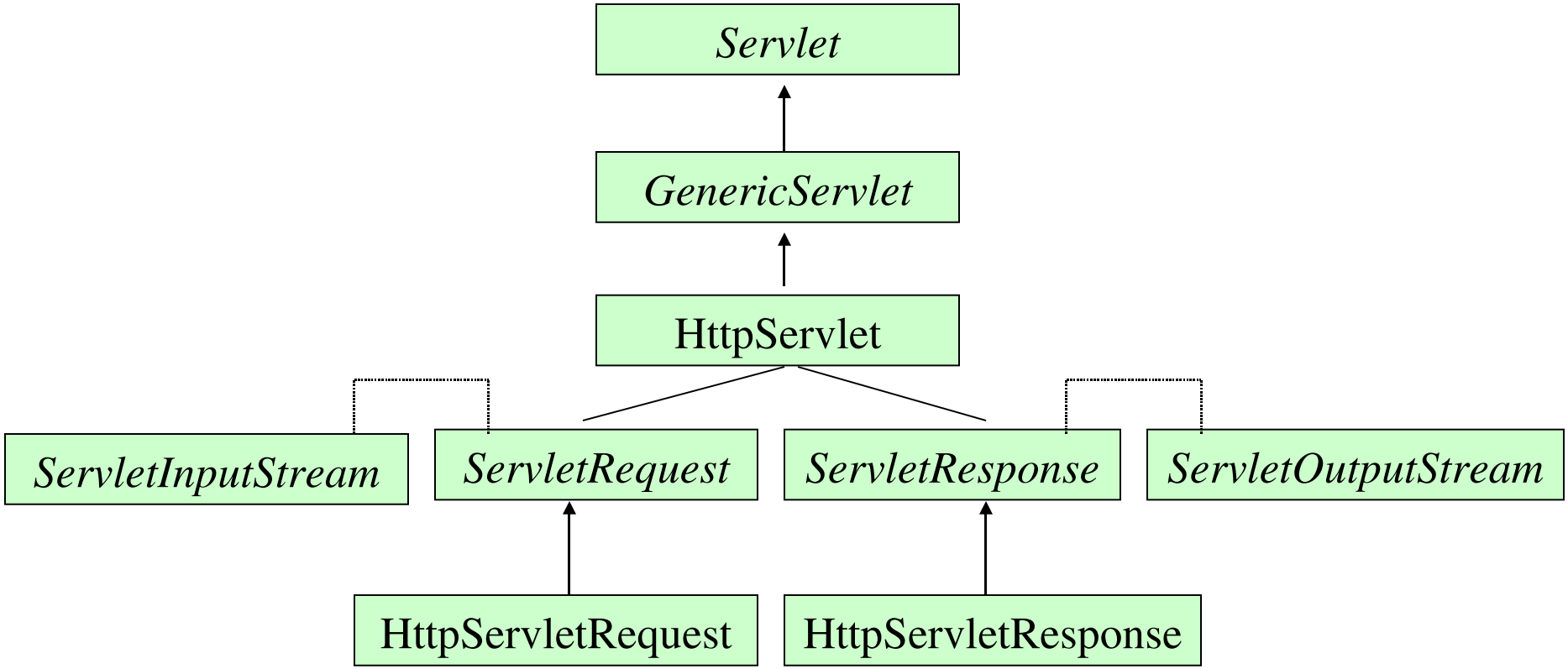
Servlets

- **Un servlet es un programa Java que se ejecuta en un servidor web y construye o sirve páginas web**
- **Más sencillo de usar, eficiente, potente y portable que un CGI**
- **Para trabajar con servlets (y JSP) necesitamos:**
  - **Un servidor web que les dé soporte (p. ej. Tomcat)**
  - **Las librerías necesarias para trabajar con ellos (suelen venir en el servidor. En Tomcat son *servlet-api.jar* y *jsp-api.jar*)**
  - **Opcionalmente, la documentación sobre las clases**

Clases de servlets

- **Toda la arquitectura de servlets está en el paquete *javax.servlet* y derivados, de la librería de servlets**
  - **La interfaz *Servlet* define las características globales**
  - **La clase *GenericServlet* es una clase abstracta que implementa esa interfaz**
  - **La clase *HttpServlet* es un subtipo de la anterior, para servlets que procesen peticiones HTTP**
    - **Trabaja con elementos *ServletRequest (HttpServletRequest)* para recibir las peticiones de los clientes**
    - **Trabaja con elementos *ServletResponse (HttpServletResponse)* para enviar las respuestas a los clientes**

Arquitectura del paquete servlet



Servlets

- Concepto de servlet
- Ciclo de vida de un servlet
- Estructura básica de un servlet
- Llamar a un servlet
- Ejemplos básicos

Ciclo de vida de un servlet

- **Todos los servlets tienen el mismo ciclo de vida:**
  - **El servidor carga e inicializa el servlet**
  - **El servlet procesa N peticiones**
  - **El servidor destruye el servlet**
- ***Inicialización*: para tareas que se hagan una sola vez al iniciar el servlet**

```
public void init() throws ServletException
{ ... }

public void init(ServletConfig conf) throws ServletException
{  super.init(conf);
   ...
}
```

Ciclo de vida de un servlet

- ***Procesamiento de peticiones*: cada petición llama al método *service( )***

```
public void service(HttpServletRequest request,
                    HttpServletResponse response)
throws ServletException, IOException
```

- **Según el tipo de petición, llama a uno de los métodos (todos con los mismos parámetros y excepciones que *service( )*):**

```
public void doGet(...)
public void doPost(...)
public void doPut(...)
public void doDelete(...)
public void doOptions(...)
public void doTrace(...)
```

Ciclo de vida de un servlet

- *Destrucción*: método *destroy( )*

```
public void destroy() throws ServletException
```

- Se debe deshacer todo lo construido en *init( )*
- Se llama a este método cuando todas las peticiones han concluido, o cuando ha pasado un determinado tiempo (en este caso, se debe controlar por código que se destruya cuando debe)

Servlets

- Concepto de servlet
- Ciclo de vida de un servlet
- Estructura básica de un servlet
- Llamar a un servlet
- Ejemplos básicos

Estructura básica de un servlet

```
import javax.servlet.*;
import javax.servlet.http.*;
public class ClaseServlet extends HttpServlet
{
   public void doGet(HttpServletRequest request,
                     HttpServletResponse response)
   throws ServletException, IOException
   {
   // ... codigo para una peticion GET
   }

   public void doPost(HttpServletRequest request,
                      HttpServletResponse response)
   throws ServletException, IOException
   {
   // ... codigo para una peticion POST
   }
}
```

Servlets

- Concepto de servlet
- Ciclo de vida de un servlet
- Estructura básica de un servlet
- Llamar a un servlet
- Ejemplos básicos

Llamada directa de servlets

- Para utilizar un servlet en una aplicación web, se coloca en el directorio *WEB-INF/classes*, con su estructura de paquetes y subpaquetes
- Después, podemos llamar al servlet directamente con:

```
http://host:puerto/<dir-aplicacion>/servlet/<nombre-servlet>
```

- O si lo hemos colocado en el *root* del servidor:

```
http://host:puerto/servlet/<nombre-servlet>
```

- Por ejemplo:

```
http://localhost:8080/miapp/servlet/paquete1.subpaquete1.MiServlet
```

Mapeado de servlets y JSP

- Otra opción es incluir en el fichero descriptor un nombre identificativo del servlet o la página JSP:

```
<servlet>
   <servlet-name>nombre</servlet-name>
   <servlet-class>ClaseServlet</servlet-class>
</servlet>
<servlet>
   <servlet-name>nombre2</servlet-name>
   <jsp-file>/mipagina.jsp</servlet-class>
</servlet>
```

- Con lo que podremos llamar al servlet o página:

```
http://host:puerto/<dir>/servlet/ClaseServlet
http://host:puerto/<dir>/servlet/nombre
http://host:puerto/<dir>/mipagina.jsp
http://host:puerto/<dir>/nombre2
```

Asignar URLs a servlets o páginas JSP

- **Podemos mapear una URL concreta con un nombre de servlet o página JSP (tras <servlet>):**

```
<servlet-mapping>
   <servlet-name>nombre</servlet-name>
   <url-pattern>/ejemploservlet</url-pattern>
</servlet-mapping>
```

- **Con lo que podremos llamar al servlet o página:**

```
http://host:puerto/ejemploservlet
```

- **También podemos usar comodines para mapear un conjunto de direcciones a un servlet o página JSP:**

```
<servlet-mapping>
   <servlet-name>nombre</servlet-name>
   <url-pattern>/ejemploservlet/*.jsp</url-pattern>
</servlet-mapping>
```

Servlets

- Concepto de servlet
- Ciclo de vida de un servlet
- Estructura básica de un servlet
- Llamar a un servlet
- Ejemplos básicos

Ejemplos básicos (I)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ClaseServlet extends HttpServlet
{
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
  throws ServletException, IOException
  {
      PrintWriter out = response.getWriter();
      out.println ("Este es un servlet de prueba");
  }
}
```

Ejemplos básicos (II)

```
...
public class ClaseServletHTML extends HttpServlet
{
   public void doGet(HttpServletRequest request,
                     HttpServletResponse response)
   throws ServletException, IOException
   {
   response.setContentType("text/html");
   PrintWriter out = response.getWriter();
   out.println ("<!DOCTYPE HTML PUBLIC \""+
                "-//W3C//DTD HTML 4.0 " +
                "Transitional//EN\">");
   out.println ("<HTML>");
   out.println ("<BODY>");
   out.println ("<h1>Titulo</h1>");
   out.println ("<br>Servlet que genera HTML");
   out.println ("</BODY>");
   out.println ("</HTML>");
   }
}
```

Ejemplos básicos (III)

```
...
public class ClaseServletHTML2 extends HttpServlet
{
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
  throws ServletException, IOException
  {
      response.setContentType("text/html");
      response.sendRedirect("miPagina.jsp");
  }
}
```