

Servlets

Laboratorio de Aplicaciones Telemáticas

Jesús Arias Fisteus

jaf@it.uc3m.es

Curso 2007/2008

Universidad Carlos III de Madrid

Introducción

- Servlet:
 - Programa Java que se ejecuta en un servidor (normalmente de HTTP) y extiende su funcionalidad.
 - Atiende peticiones recibidas desde los clientes y genera las respuestas.

Introducción

■ Servlets:

- Se ejecutan en una máquina virtual dentro del proceso del servidor.
- Gestionados por un *motor de servlets*.
- Cada petición HTTP recibida se procesa en un hilo, e invoca un método del servlet.
- Además de servidores de HTTP, pueden extender cualquier tipo de servidor (por ejemplo, FTP).

Ventajas fundamentales (I)

- Portabilidad entre plataformas y servidores.
- Potencia:
 - APIs de Java: acceso a red, hilos, manipulación de imágenes, acceso a bases de datos, compresión de datos, internacionalización, RMI, CORBA, serialización de objetos, acceso a directorio, etc.
 - Utilización de código externo, incluido EJBs.

Ventajas fundamentales (II)

- Eficiencia:
 - Instancia permanentemente cargada en memoria por cada servlet.
 - Ejecución de peticiones mediante invocación de un método.
 - Cada petición se ejecuta en un hilo.
 - Mantiene automáticamente su estado y recursos externos: conexiones a bases de datos, conexiones de red, etc.

Ventajas fundamentales (III)

- Seguridad:
 - Lenguaje java: máquina virtual, chequeo de tipos, gestión de memoria, excepciones, etc.
 - Gestor de seguridad de Java.
- Elegancia:
 - Código java: modular, orientado a objetos, limpio y simple.
 - API servlets: potente y fácil de utilizar.

Ventajas fundamentales (IV)

- Integración:
 - Integración fuerte entre servlets y servidor: permite colaboración entre ambos.
- Extensibilidad y flexibilidad:
 - API Servlet extensible.
 - Filtros (cadenas de servlets).
 - Integrable con JSP (Java Server Pages).
- Comunidad grande de desarrolladores.

Concepto de *aplicación Web*

- Conjunto de servlets, JSPs y otros recursos (ficheros HTML, imágenes, ficheros de configuración, etc.) relacionados entre sí por formar parte de la misma aplicación.
- Los recursos de una aplicación Web comparten un prefijo de URL.
- Una aplicación Web se puede empaquetar en un fichero WAR.

Ciclo de vida de un servlet

- Cuando arranca el servidor:
 - Se crea una instancia.
 - Se inicializa el servlet (método `init()`)
- Cuando llega una petición:
 - Se invoca el método `service()` sobre un nuevo hilo.
- Cuando se cierra el servidor:
 - Se invoca el método `destroy()` y después se destruye el servlet.

Consecuencias del ciclo de vida (I)

- Una única máquina virtual:
 - Compartición de datos entre servlets.
- Persistencia de instancias:
 - Consumo reducido de memoria.
 - Eliminación del tiempo de instanciación e inicialización.
 - Persistencia de estado, datos y recursos:
 - Atributos del servlet persistentes.
 - Conexiones permanentes a bases de datos.
 - Persistencia de hilos.

Consecuencias del ciclo de vida (II)

- Necesidad de sincronización:
 - Problemas si se accede a los mismos datos concurrentemente desde distintos hilos (atributos de clase o instancia, base de datos, etc.)
 - *SingleThreadModel*: modo alternativo que evita el acceso concurrente a los atributos de instancia de un *servlet* manteniendo múltiples instancias de cada uno.

API de servlets

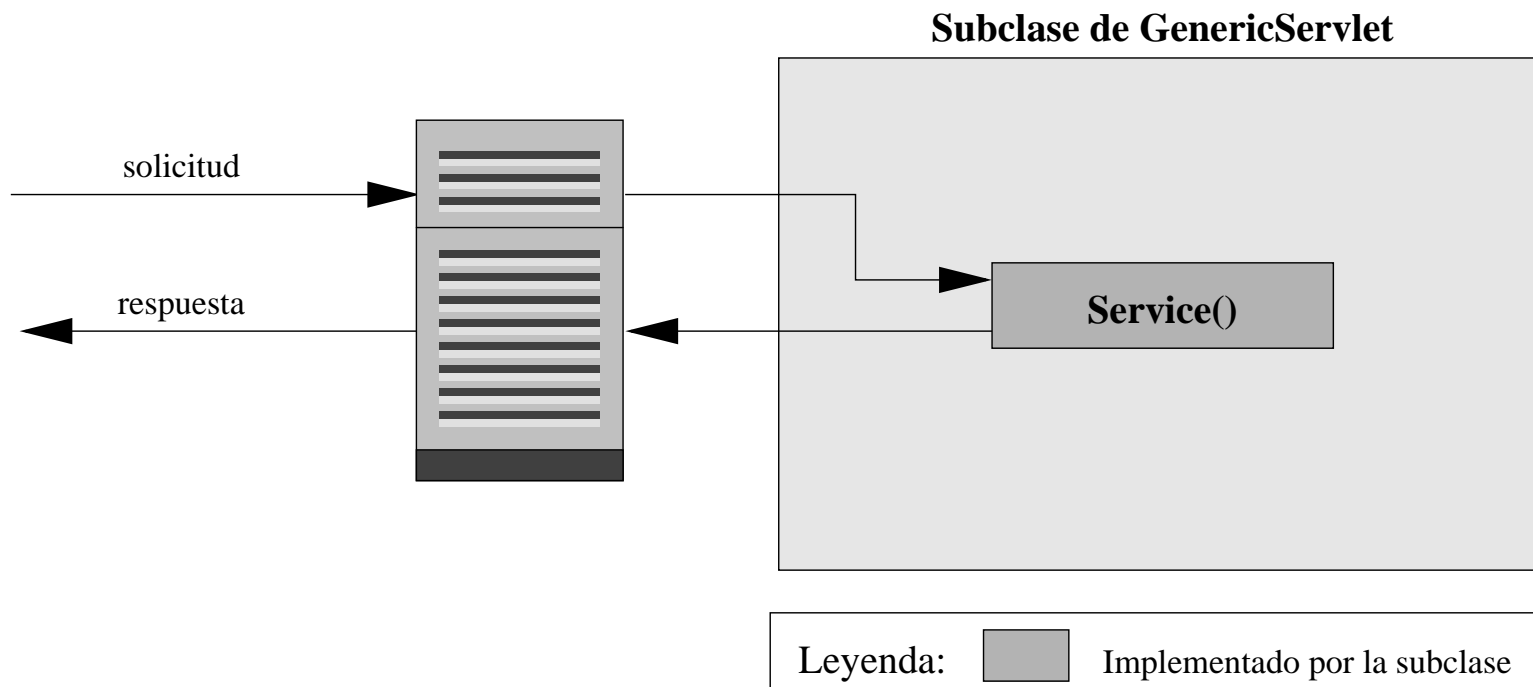
- Conjunto de clases e interfaces que implementan la interfaz entre el servidor y los servlets.
 - Paquete `javax.servlet`: clases genéricas, válidas para cualquier protocolo.
 - Paquete `javax.servlet.http`: extiende la API y la concreta para el protocolo HTTP.
- Estas APIs son comunes para todos los servidores capaces de ejecutar servlets.
- La versión más reciente es API Servlet 2.5 (mayo de 2006).
 - ... pero trabajaremos con API Servlet 2.4.

La interfaz Servlet

- Un servlet implementa la interfaz *javax.servlet.Servlet*.
 - Normalmente hereda la implementación de una de las clases:
 - *javax.servlet.GenericServlet*
 - *javax.servlet.HttpServlet*
- Métodos importantes de la interfaz:
 - *void service(ServletRequest req, ServletResponse res)*
 - *void init(ServletConfig config)*
 - *void destroy()*

El método `service()`

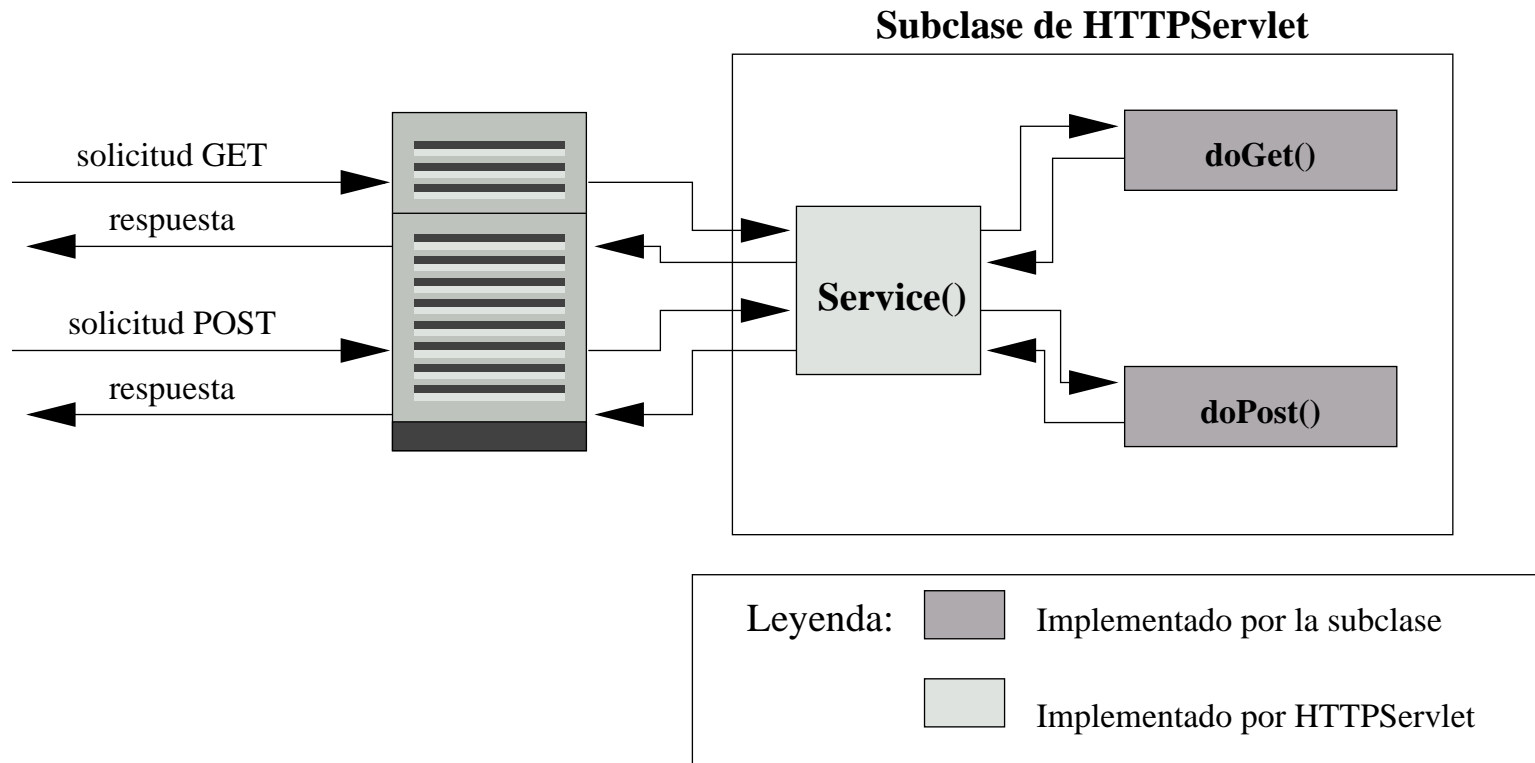
```
void service(ServletRequest req, ServletResponse res)
```



Servlets HTTP

- Heredan de `HttpServlet`, que implementa el método `service()` para que invoque a:
 - *`void doGet(HttpServletRequest req, HttpServletResponse resp)`*
 - *`void doPost(HttpServletRequest req, HttpServletResponse resp)`*
 - *`void do...(HttpServletRequest req, HttpServletResponse resp)`*
 - *`getLastModified(HttpServletRequest req)`*
- Los servlets reescriben sólo los métodos *`doXXX`* que necesiten.

Servlets HTTP



Clases e interfaces útiles

- Interfaz *ServletConfig*
- Interfaz *ServletContext*
- Interfaz *HttpServletRequest*
- Interfaz *HttpServletResponse*
- Interfaz *HttpSession*
- Clase *Cookie*

Acceso a información útil (I)

- Parámetros de inicio:
 - Se configuran en el fichero `WEB-INF/web.xml`
 - *`ServletConfig.getInitParameter()`*
- Información acerca del servidor:
 - *`ServletRequest.getServerName()`*
 - *`ServletRequest.getServerPort()`*
 - *`ServletContext.getServerInfo()`*
 - *`ServletContext.getAttribute(String name)`*

Acceso a información útil (II)

- Información acerca del cliente:
 - *ServletRequest.getRemoteAddr()*
 - *ServletRequest.getRemoteHost()*
 - *HttpServletRequest.getRemoteUser()*
 - *HttpServletRequest.getHeader(...)*

Información de la petición (I)

- Información general:
 - *HttpServletRequest.getMethod()*
 - *HttpServletRequest.getQueryString()*
 - *HttpServletRequest.getHeader(...)*
- Parámetros de la petición:
 - *ServletRequest.getParameter(String name)*
 - *ServletRequest.getParameterValues(String name)*
 - *ServletRequest.getParameterNames()*
 - **Nota:** esta API no funciona con *multipart/form-data*.

Información de la petición (II)

- Cuerpo de la petición:
 - *ServletRequest.setContentType()*
 - *ServletRequest.getContentLength()*
 - *ServletRequest.getInputStream()*
 - *ServletRequest.getReader()*

Respuesta (I)

- El servlet puede escribir, en una respuesta HTTP:
 - Código de estado.
 - Cabeceras (incluidas *cookies*).
 - Cuerpo.
- Código de estado:
 - *HttpServletResponse.sendError(int sc)*
 - *HttpServletResponse.setStatus(int sc)*
 - *HttpServletResponse.sendRedirect(String location)*

Respuesta (II)

- Cabeceras:
 - *HttpServletResponse.setHeader(String name, String value)*
 - *HttpServletResponse.addCookie(Cookie cookie)*
 - *ServletResponse.setContentType(String type)*
 - *ServletResponse.setContentLength(int length)*
- Cuerpo:
 - *ServletResponse.getOutputStream()*
 - *ServletResponse.getWriter()*

Sesiones (I)

- Tomcat mantiene automáticamente las sesiones de usuario:
 - Por defecto, utiliza *cookies* para que el cliente envíe su identificador de sesión en cada petición.
 - Cada sesión se representa con un objeto *HttpSession*.
 - Una sesión caduca tras un tiempo (configurable) sin recibir peticiones correspondientes a la misma.

Sesiones (II)

- Obtención del objeto sesión desde el servlet:
 - *HttpServletRequest.getSession(boolean create)*:
 - Devuelve el objeto de sesión correspondiente a la petición.
 - Con parámetro *true*, crea una nueva sesión si la petición no corresponde a ninguna sesión.
- Se puede almacenar objetos en la sesión:
 - *HttpSession.setAttribute(String name, Object value)*
 - *HttpSession.getAttribute(String name)*

Contexto

- Cada aplicación Web en cada JVM tiene asociado un objeto `ServletContext`:
 - Todos los servlets y JSPs de una misma aplicación Web en una misma JVM comparten el mismo objeto de contexto.
 - Da acceso a información y funciones del servidor.
 - Permite compartir objetos Java entre todos los recursos de la misma aplicación Web:
 - `ServletContext.setAttribute(String name, String value)`
 - `ServletContext.getAttribute(String name)`

Código ejemplo

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

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