INTRODUCCIÓN A LOS SERVLETS

Eduard Lara

INDICE

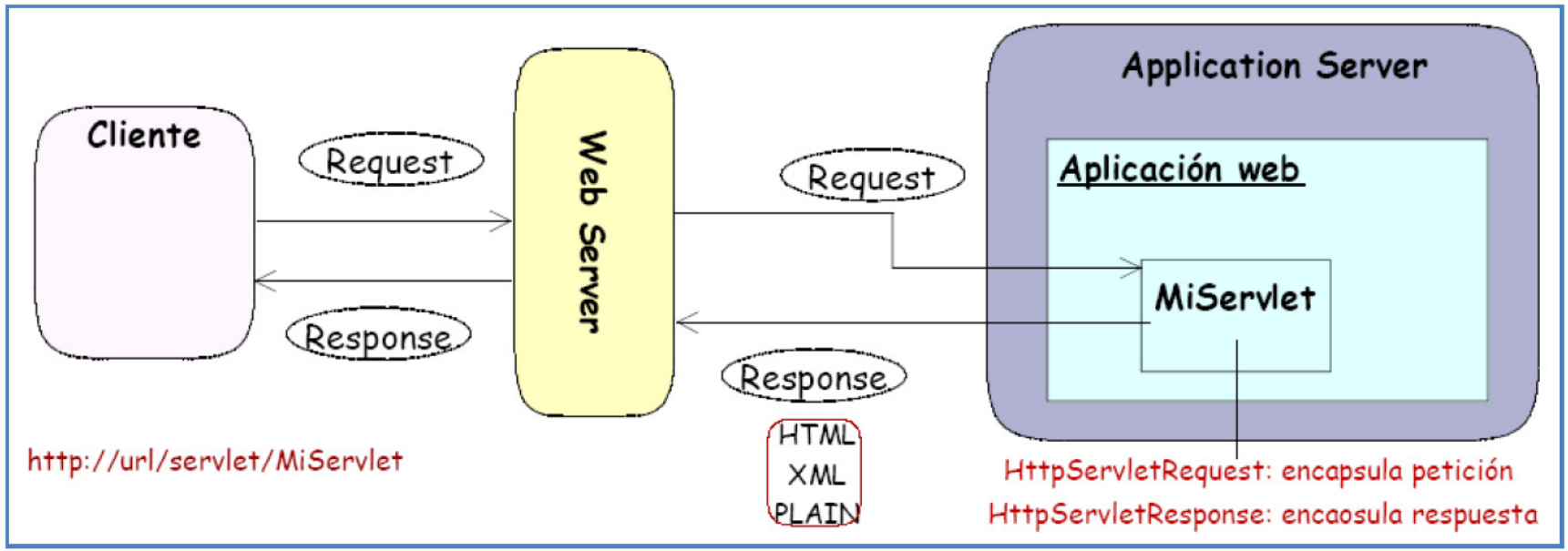
1. Introducción a los servlets
2. Características de los servlets
3. API de desarrollo de los servlets
4. Ciclo de vida de un servlet
5. Mapeo del servlet
6. Atributos

1. INTRODUCCIÓN A LOS SERVLETS

- Los Servlets son el corazón de J2EE. Es la pieza básica de cualquier aplicación, alrededor de la cual se organizan el resto de elementos.
- Los servlets son objetos de negocio, escritos en Java, que deben de heredar de la clase HTTPServlet. Esto es así para respetar el estándar de J2EE, y que cualquier servidor de aplicaciones sea capaz de utilizarla.
- Pueden recibir peticiones de un muchos clientes y generar respuestas (pueden generar documentos HTML, XML, texto plano...).
- Una misma instancia de un servlet puede ejecutarse de manera concurrente para satisfacer peticiones solapadas en el tiempo.

1. INTRODUCCIÓN A LOS SERVLETS

- Como vemos en la siguiente imagen, el servlet recoge la petición (request) y emite la respuesta (response).



2. CARACTERÍSTICAS DE LOS SERVLETS

- Son independientes del servidor utilizado y de su S.O.
- Los servlets pueden llamar a 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.

2. CARACTERÍSTICAS DE LOS SERVLETS

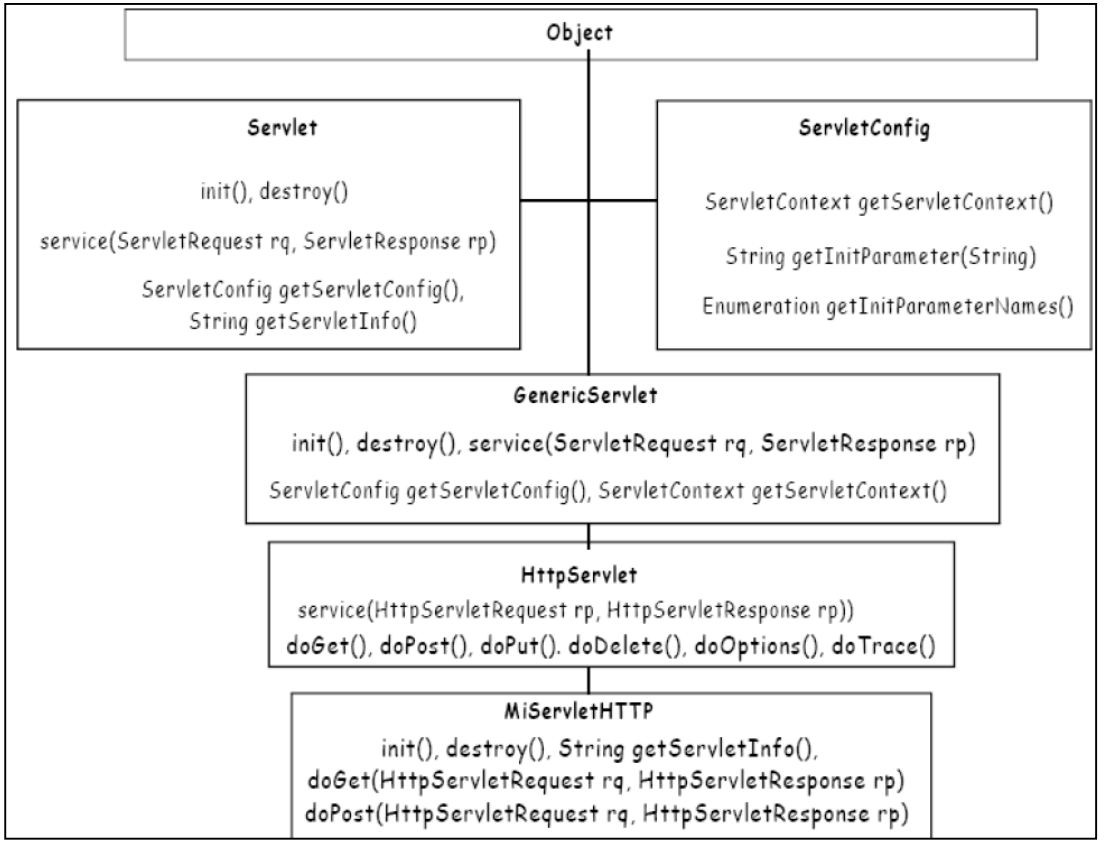
- Permiten además la utilización de cookies y sesiones, de forma que se puede guardar información específica acerca de un usuario determinado.
- Los servlets pueden actuar como enlace entre el cliente y una o varias bases de datos en arquitecturas cliente-servidor de 3 capas.
- Pueden realizar tareas de proxy para un applet. Debido a las restricciones de seguridad, pe. un applet no puede acceder directamente a un servidor de datos localizado en cualquier máquina remota, pero el servlet sí puede hacerlo de su parte.
- Al igual que los programas CGI, 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.

3. API DE DESARROLLO DE LOS SERVLETS

- Oracle proporciona un conjunto de clases Java para el desarrollo de servlets. Este API se encuentra diseñado como una extensión del JDK.
- Consta de dos paquetes javax.servlet y javax.servlet.http.
- Este último es una particularización del primero para el caso del protocolo HTTP, que es el más utilizado. Mediante este diseño lo que se consigue es que se mantenga abierta la posibilidad de implementar servlets para otros protocolos existentes (FTP, POP, SMTP).
- Una clase Java se dice que es un HTTPServlet, si hereda de la clase javax.servlet.http.HttpServlet y su entorno de ejecución es un servidor HTTP.

3. API DE DESARROLLO DE LOS SERVLETS

- Una clase que herede de javax.servlet.GenericServlet ya se considera un servlet, pero lo más común es que trabajemos con HttpServlet.



3. API DE DESARROLLO DE LOS SERVLETS

El paquete javax.servlet define los siguientes elementos:

- Interfaz Servlet
- Interfaz ServletRequest
- Interfaz ServletResponse
- Interfaz ServletConfig
- Interfaz ServletContext
- Clase GenericServlet

3.1. INTERFAZ SERVLET

- La deben implementar todos los servlets.
- El servidor invoca a los métodos init() y destroy() para iniciar y detener un servlet.
- Los métodos getServletConfig() y getServletInfo() retornan información acerca del servlet.
- El método service() es invocado por el servidor para que el servlet realiza su servicio (una petición). Este método dispone de dos parámetros. uno del tipo de la interfaz ServletRequest y otro del tipo de la interfaz ServletResponse.

3.2. INTERFAZ SERVLETREQUEST

Este interfaz encapsula la petición de servicio de un cliente. Define una serie de métodos tendentes a obtener información del servidor, solicitante y solicitud. Los métodos más importantes son:

- int getContentLength(): Devuelve el tamaño de la petición del cliente o -1 si es desconocido.
- String getContentType(): Devuelve el tipo de contenido MIME de la petición o null si éste es desconocido.
- String getProtocol(): Devuelve el protocolo y la versión de la petición como un String en la forma <protocolo>/<versión mayor>.<versión menor>
- String getScheme(): Devuelve el tipo de esquema de la URL de la petición: http, https, ftp...
- String getServerName(): Devuelve el nombre del host del servidor que recibió la petición..
- int getServerPort() Devuelve el número del puerto en el que fue recibida la petición.

3.2. INTERFAZ SERVLETREQUEST

- String getRemoteAddr() Devuelve la dirección IP del ordenador que realizó la petición.
- String getRemoteHost() Devuelve el nombre completo del ordenador que realizó la petición.
- String getParameter(String) Devuelve un String que contiene el valor del parámetro especificado, o null si dicho parámetro no existe. Sólo debe emplearse cuando se está seguro de que el parámetro tiene un único valor.
- String[] getParameterValues(String) Devuelve los valores del parámetro especificado en forma de un array de Strings, o null si el parámetro no existe. Útil cuando un parámetro puede tener más de un valor.
- Enumeration getParameterNames() Devuelve una enumeración en forma de String de los parámetros encapsulados en la petición. No devuelve nada si el InputStream está vacío.

3.3. INTERFAZ SERVLETRESPONSE

Esta interfaz es utilizada por un servlet para enviar información al solicitante de una petición.

Los métodos más importantes son:

- ServletOutputStream getOutputStream() Permite obtener un ServletOutputStream para enviar datos binarios.
- PrintWriter getWriter() Permite obtener un PrintWriter para enviar caracteres.
- setContentType(String) Establece el tipo MIME de la salida.("text/html")
- setContentLength(int) Establece el tamaño de la respuesta

3.4. INTERFAZ SERVLETCONFIG

La utiliza un servidor para pasar información sobre la configuración a un servlet. Sus métodos los utiliza el Servlet para recuperar esta información, por ejemplo los parámetros iniciales del servlet. Un servlet puede recuperar el objeto de esta interfaz a través del método getServletConfig().

Los métodos más importantes son:

- String getInitParameter(String) permite obtener el valor de un parámetro inicial, el nombre de este parámetro se especifica como argumento del método.
- Enumeration getInitParameterNames() permite obtener todos los nombres de los parámetros iniciales.

3.5. INTERFAZ SERVLETCONTEXT

Define el entorno en que se ejecuta el servlet. Proporciona métodos que utilizan los servlets para acceder a información sobre el entorno. La información acerca del servidor está disponible en todo momento a través de un objeto de la interface ServletContext. Un servlet puede obtener dicho objeto mediante el método getServletContext() aplicable a un objeto ServletConfig.

Los métodos más importantes son:

- Object getAttribute(String) Devuelve información acerca de determinados atributos del tipo clave/valor del servidor. Es propio de cada servidor.
- String getMimeType(String) Devuelve el tipo MIME de un determinado fichero.
- public abstract String getRealPath(String) Traduce una ruta de acceso virtual a la ruta relativa al lugar donde se encuentra el directorio raíz de páginas HTML
- String getServerInfo() Devuelve el nombre y la versión del servicio de red en el que está siendo ejecutado el servlet.

3.6. INTERFAZ GENERICSERVLET

- Implementa el interfaz Servlet. Se puede heredar de esta clase para construir clases Servlet propias.
- La clase GenericServlet es una clase abstract puesto que su método service() es abstract. Esta clase implementa dos interfaces, de las cuales la más importante es la interface Servlet.
- Cualquier clase que derive de GenericServlet deberá definir el método service(). Es muy interesante observar los dos argumentos que recibe este método, correspondientes a las interfaces: ServletRequest y ServletResponse.

3.6. INTERFAZ GENERICSERVLET

- El primer argumento referencia a un objeto que describe por completo la solicitud de servicio que se le envía al servlet. Si la solicitud de servicio viene de un formulario HTML, por medio de ese objeto se puede acceder a los nombres de los campos y a los valores introducidos por el usuario; puede también obtenerse cierta información sobre el cliente.
- El segundo argumento es un objeto con una referencia de la interface ServletResponse, que constituye el camino mediante el cual el método service() se conecta de nuevo con el cliente y le comunica el resultado de su solicitud. Además, dicho método deberá realizar cuantas operaciones sean necesarias para desempeñar su cometido: escribir y/o leer datos de un fichero, comunicarse con una base de datos, etc.
- El método service() es realmente el corazón del servlet.

3.7. EL PAQUETE JAVAX.SERVLET.HTTP

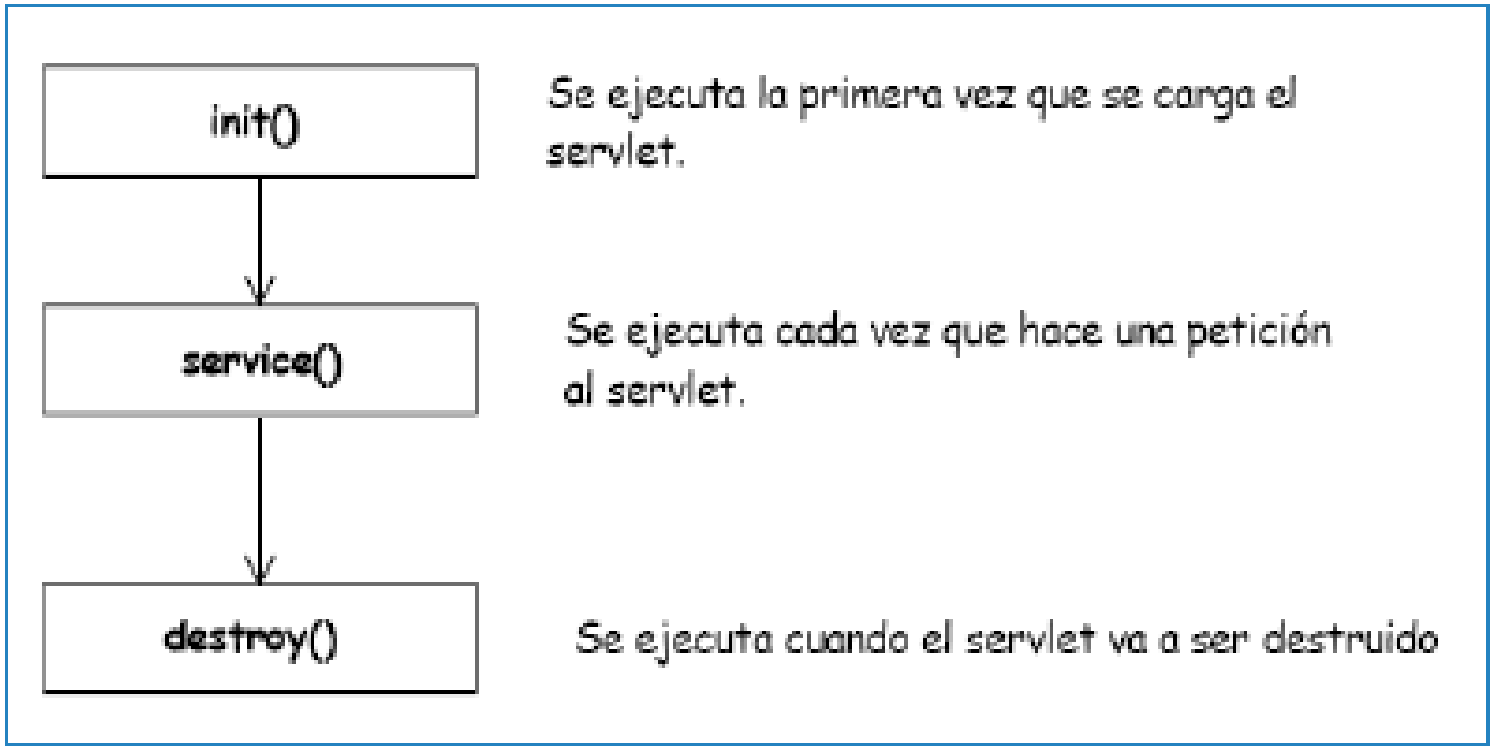
- En la práctica, salvo para desarrollos muy especializados, todos los servlets deberán construirse a partir de la clase HttpServlet, sub-clase de GenericServlet.
- La clase HttpServlet ya no es abstract y dispone de una implementación o definición del método service(). Dicha implementación detecta el tipo de servicio o método HTTP que le ha sido solicitado desde el browser y llama al método adecuado de esa misma clase (doPost(), doGet(),etc..).
- Cuando el programador crea una sub-clase de HttpServlet, por lo general no tiene que redefinir el método service(), sino uno de los métodos más especializados (normalmente doGet(), doPost()), que tienen los mismos argumentos que service(): dos objetos de las clases HttpServletRequest y HttpServletResponse que implementa las interfaces ServletRequest y ServletResponse.

3.8. CLASES DE JAVAX.SERVLET.HTTP

- HttpServletRequest; Amplía la interfaz ServletRequest y agrega métodos para acceder a los detalles de una solicitud HTTP.
- HttpServletResponse; Amplía la interfaz ServletResponse y agrega constantes y métodos para devolver respuestas específicas del HTTP.
- HttpSession; Se implementa por servlets para permitir sesiones navegador-servidor que abarcan múltiples pares de solicitudes-respuestas.
- Cookie; Representa una Cookie HTTP. Una Cookie es un conjunto de información que se genera en el servidor y se almacena en los clientes individuales.
- HttpServlet; Amplía de GenericServlet con el fin de utilizar interfaces HttpServletRequest y HttpServletResponse.

4. CICLO DE VIDA DE UN SERVLET

En un GenericServlet tenemos los siguientes métodos para gestionar su ciclo de vida. Al ser los servlets componentes manejados por el contenedor web estos métodos serán invocados por él.

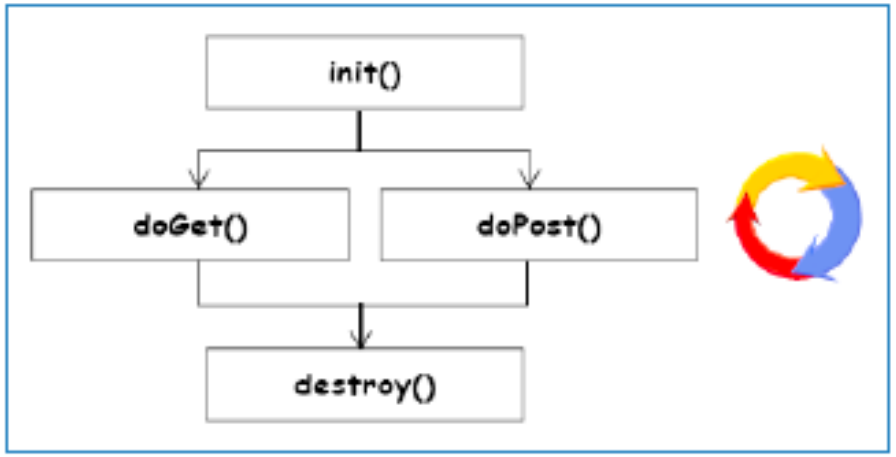


4. CICLO DE VIDA DE UN SERVLET

- El método **init** lo invoca el contenedor de servlets, una sola vez; resulta útil para inicializar recursos que serán necesarios en la ejecución del servlet (abrir ficheros, conectar bases de datos, establecer comunicaciones, etc.).
- El método **destroy** se invoca cuando el servlet va a ser descargado del servidor; en su interior se debería programar la liberación de recursos utilizados en el método init.
- El método **service** realiza el trabajo "cotidiano" del servlet: dar respuesta a las distintas peticiones de los clientes. Cada vez que un cliente realiza una petición (GET, POST, etc.) el contenedor invoca al método service.

4. CICLO DE VIDA DE UN SERVLET

En un HttpServlet los métodos de ciclo de vida son los siguientes:



Como podemos comprobar el método service se desglosa en dos:

- **goGet();** capturará las peticiones enviadas a través del método GET.
- **doPost();** capturará las peticiones enviadas a través del método POST.

4. CICLO DE VIDA DE UN SERVLET

```
public class ServletTienda extends HttpServlet {

    // Este metodo lo invoca el contenedor web en el momento que destruye la instancia del servlet
    @Override
    public void destroy() {...}

    // Este metodo lo invoca el contenedor web en el momento que crea la instancia del servlet
    @Override
    public void init(ServletConfig config) throws ServletException {...}

    // Este metodo lo invoca el contenedor web cuando entra una petición por el método GET
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {...}

    // Este metodo lo invoca el contenedor web cuando entra una petición por el método POST
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {...}
```

5. MAPEO DEL SERVLET

Una vez que tenemos el servlet creado el siguiente paso es mapearlo para poder invocarlo a través de la url.

Actualmente tenemos dos formas de hacerlo:

- A través del descriptor de despliegue web.xml; esta es la forma tradicional como se ha venido haciendo siempre.
- A través de anotaciones; a partir de JEE 6 se contempla esta opción.

5.1. MAPEO DEL SERVLET EN EL DESCRIPTOR WEB.XML

Para mapear el servlet necesitamos de dos secciones:

- La declaración del servlet; que consiste en asociar un alias (<servlet-name>) al nombre de la clase del servlet (servlet-class).
- El mapeo del servlet; consiste en asociar el alias (<servlet-name>) con el patron url (<url-pattern>) que se utilizara para enviar la petición en la url.

```
<!-- Declaración del servlet   -->
<servlet>
    <servlet-name>Servlet</servlet-name>
    <servlet-class>app.web.ServletTienda</servlet-class>
</servlet>

<!-- Mapeo del servlet   -->
<servlet-mapping>
    <servlet-name>Servlet</servlet-name>
    <url-pattern>/controlador</url-pattern>
</servlet-mapping>
```

5.1. MAPEO DEL SERVLET EN EL DESCRIPTOR WEB.XML

Una vez que ya tenemos el servlet mapeado podremos invocarlo a través de una petición.

1) Veamos un ejemplo de petición mediante un link:

```
<a href="controlador?opcion=1">Consultar todos los productos</a><br>
```

Como podemos comprobar no utilizamos su alias y tampoco el nombre completo de su clase. El servlet se invoca a través de su patrón url.

2) Otro ejemplo es como invocarlo a través del action de un formulario:

```
<form action="controlador" method="post">
    Introduce Id del producto:
    <input type="text" name="codigo" />
    <input type="hidden" name="opcion" value="2" />
    <input type="submit" value="ENVIAR" />
</form>
```

5.2. MAPEO DEL SERVLET A TRAVÉS DE ANOTACIONES

En la siguiente imagen vemos otra forma de mapear servlets. La opción de mapear el servlet con una notación al inicio de la clase del propio Servlet, se contempla a partir de JEE 6.

```
@WebServlet(name="Servlet", urlPatterns={"/controlador"})
public class ServletTienda extends HttpServlet {
```

6. ATRIBUTOS

A veces será necesario pasar cierto objeto de un servlet a una página JSP. Existe una forma de hacerlo, que es encapsular el objeto que deseamos pasar en el objeto request.

Veamos el siguiente ejemplo donde guardamos la lista de los productos como atributo de la petición.

```
// Guardamos el producto encontrado como atributo de la peticion
request.setAttribute("encontrado", encontrado);

// Elegir la vista para mostrar
RequestDispatcher rd = request.getRequestDispatcher("/mostrarProducto.jsp");
// Redirigir hacia esa vista
rd.forward(request, response);
```

6. ATRIBUTOS

El recurso destino, en este caso una jsp, utiliza los objetos request y response como si la petición hubiese sido realizada directamente sobre ella.
Nosotros podemos incluir dentro del request tantos objetos como deseemos que la aplicación destino sea capaz de recoger.
La instrucciones clave para realizar este proceso están recogidas como métodos propios del request, estos métodos son:

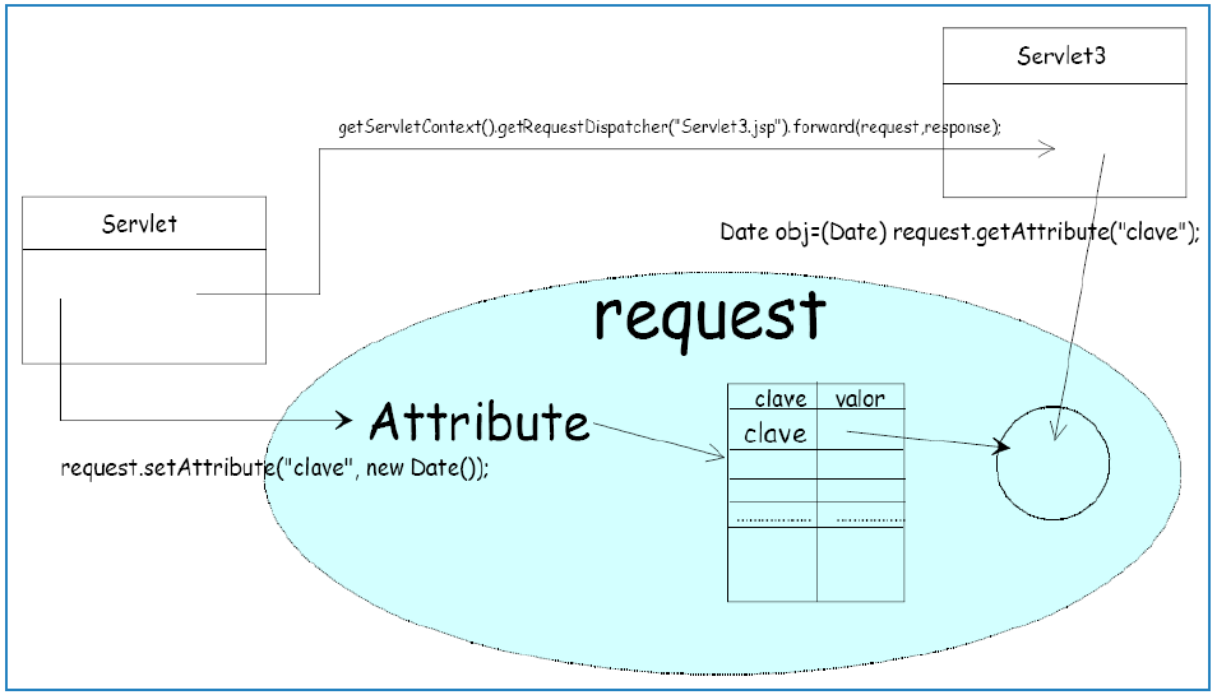
- request.setAttribute(String, Object), para guardar un objeto (Object) en el request, con un identificador, clave (String).
- request.getAttribute(String), para recuperar el objeto guardado con un identificador (String).

6. ATRIBUTOS

Recuperación de un atributo en una página jsp.

```
<% List<Producto> lista = (List)request.getAttribute("listaProductos"); %>
```

Esquema de paso de atributos entre servlets:



RECUERDA QUE...

- Un servlet es un componente que administra el contenedor web.
- Podemos declarar y mapear el servlet a través de código xml o usando anotaciones.
- Los parámetros de la petición se reciben en el servidor.
- Los atributos se crean en el servidor para poder enviar datos a otros componentes.
- Desde el servlet podemos lanzar solicitudes a otros servlets o jsp.