UCH UNIVERSIDAD DE CIENCIAS Y HUMANIDADES

INGENIERÍA DE SOFTWARE III

Eric Gustavo Coronel Castillo
ecoronel@gmail.com

2016

TEMA: Servlets

Índice

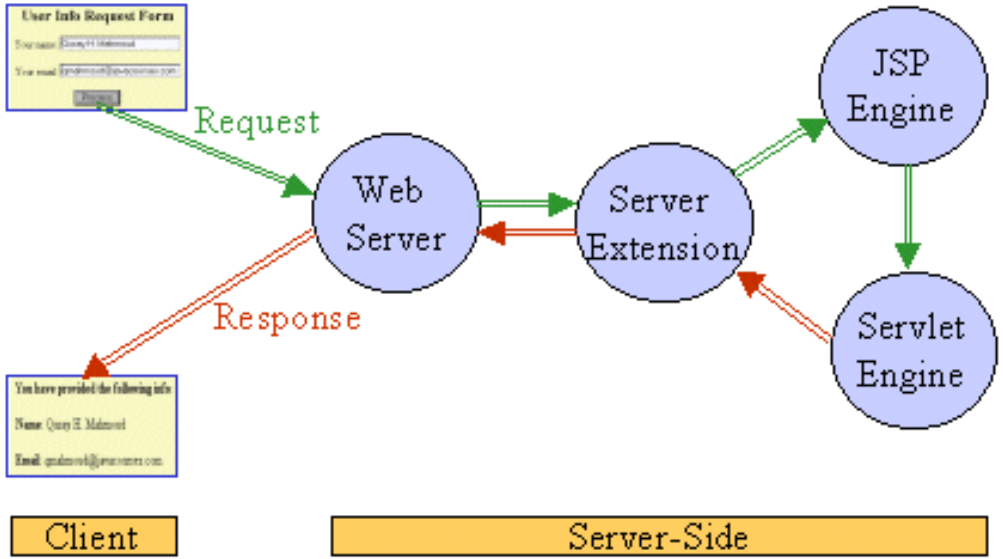
- Objetivo
- ¿Qué es Java Server Pages?
- Elementos Básicos
- Directivas
- Acciones
- Objetos Implícitos

Índice

- Objetivo
- ¿Qué es un Servlet?
- Arquitectura del Paquete Servlet
- Proyecto 01
- Interacción con los Clientes
- Programación de Servlets
- Proyecto 02
- Interacción con un Servlet
- Servlets con Múltiples Mapeos
- Proyecto 03

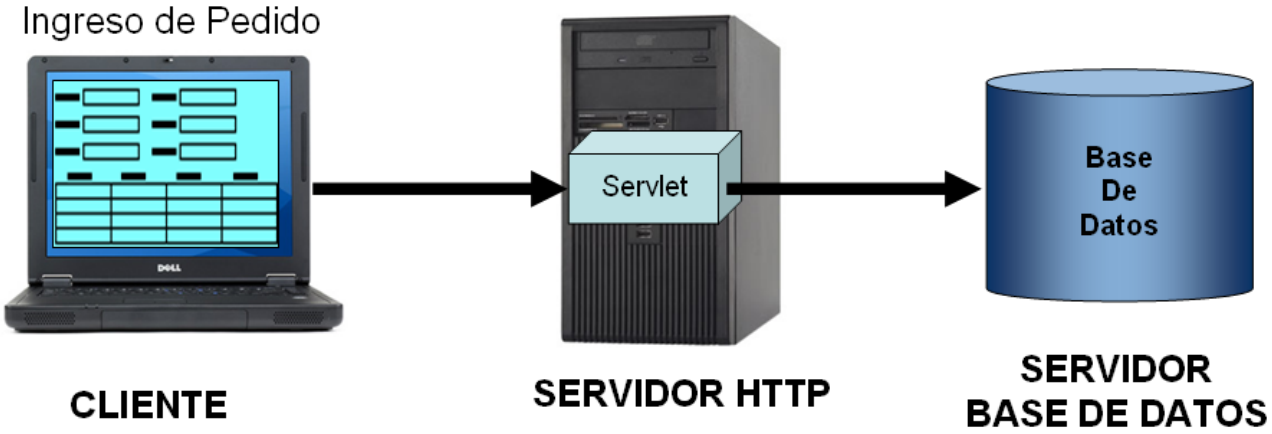
Objetivo

- Entender el funcionamiento de los servlets.
- Aplicar servlets en el desarrollo de aplicaciones web.

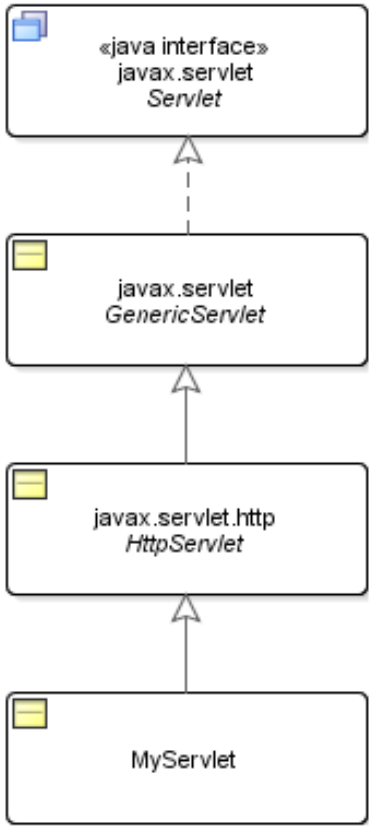


¿Qué es un Servlet?

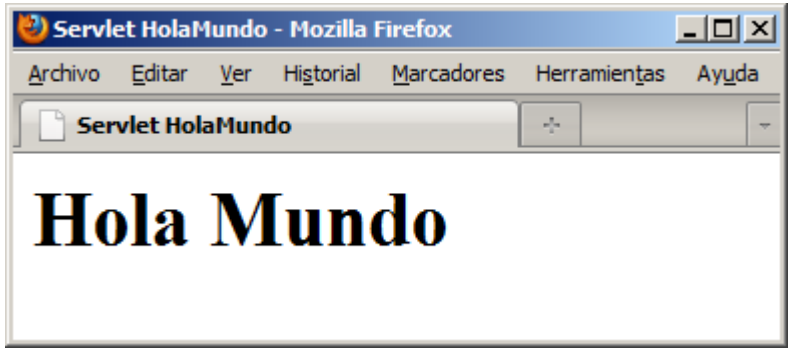
- Los Servlets son módulos que extienden los servidores orientados a requerimiento/respuesta, como los servidores web compatibles con Java.
- Por ejemplo, un servlet podría ser responsable de tomar los datos de un formulario de entrada de pedidos en HTML y aplicarle la lógica de negocios utilizada para actualizar la base de datos de pedidos de una compañía.



Arquitectura del Paquete Servlet



Proyecto 01



Servlet 3.x

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

}
```

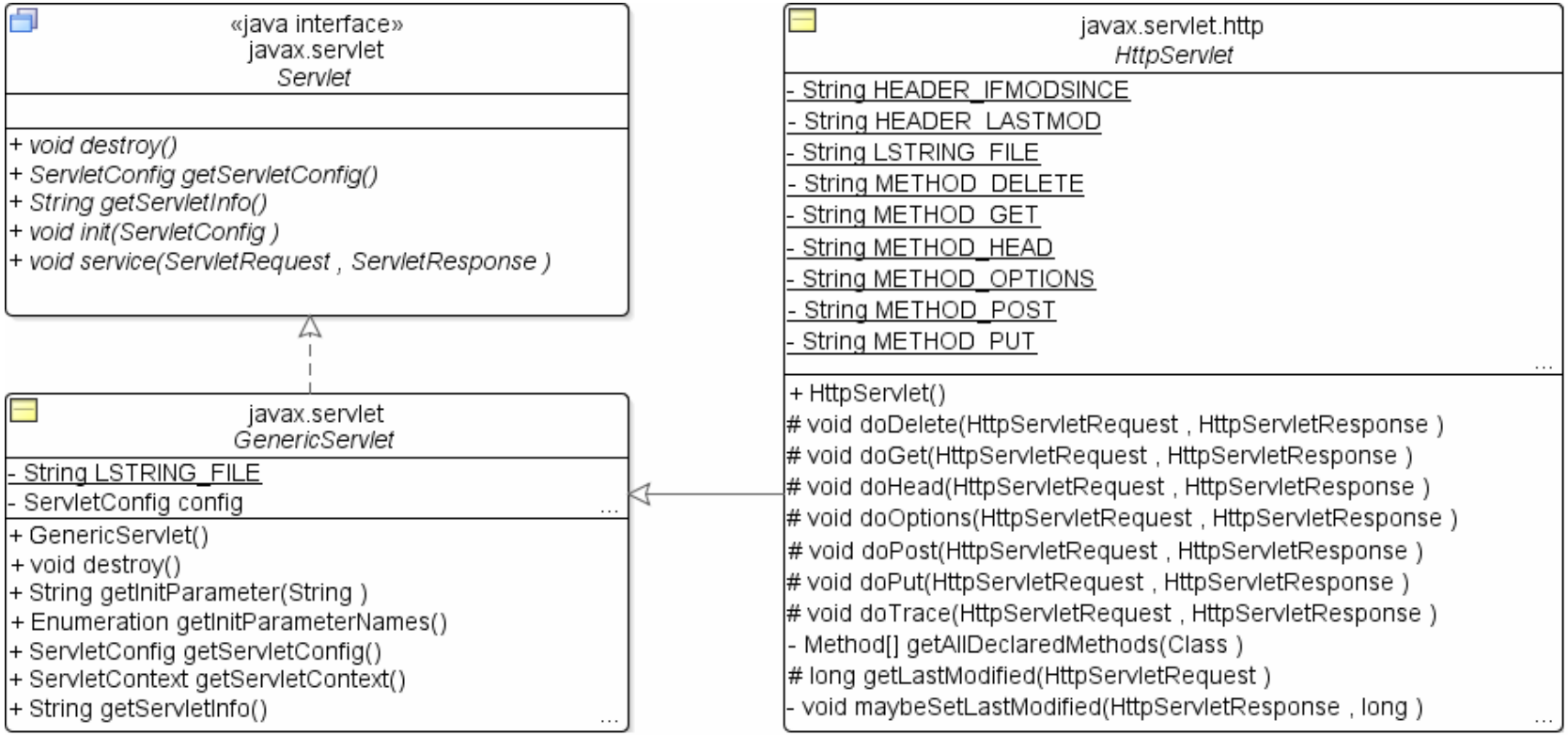
Servlet 2.x

```
<servlet>
        <servlet-name>HolaMundo</servlet-name>
        <servlet-class>project1.HolaMundo</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>HolaMundo</servlet-name>
        <url-pattern>/holamundo</url-pattern>
</servlet-mapping>
```

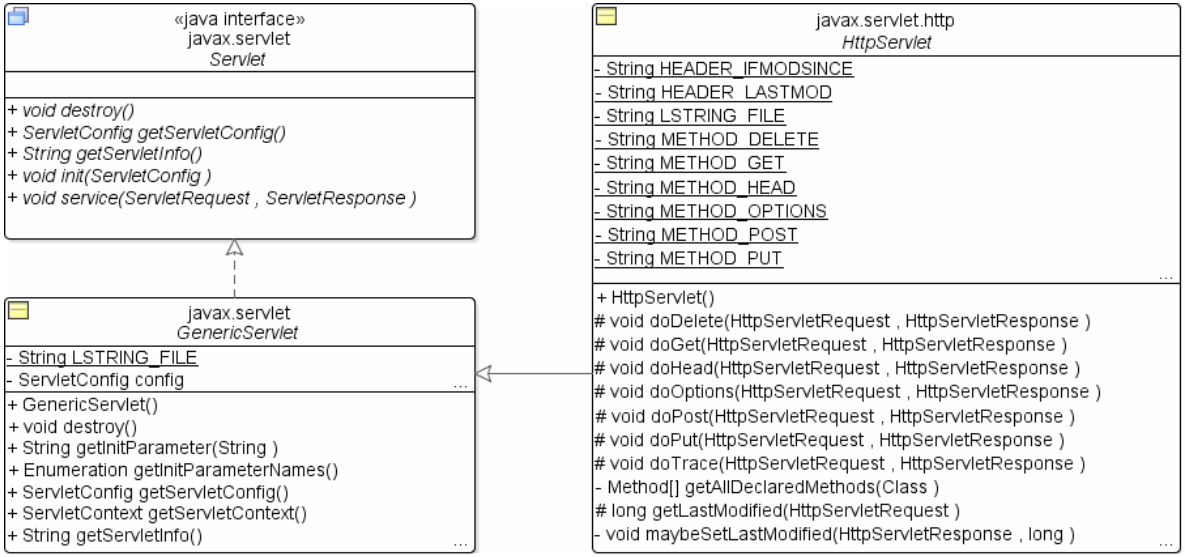
Interacción con los Clientes

- Objetos **HttpServletRequest** y **HttpServletResponse**.
- Requerimientos GET y POST.
- Método service(…).
- Métodos doGet(…) y doPost().

Programación de Servlets

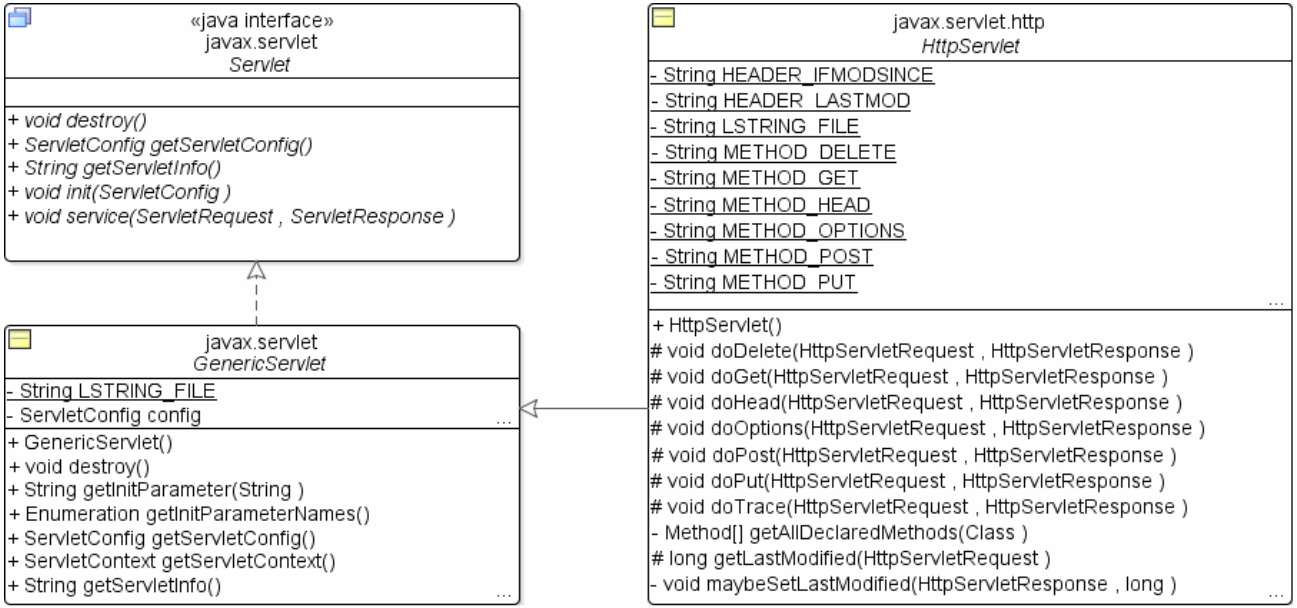


Programación de Servlets



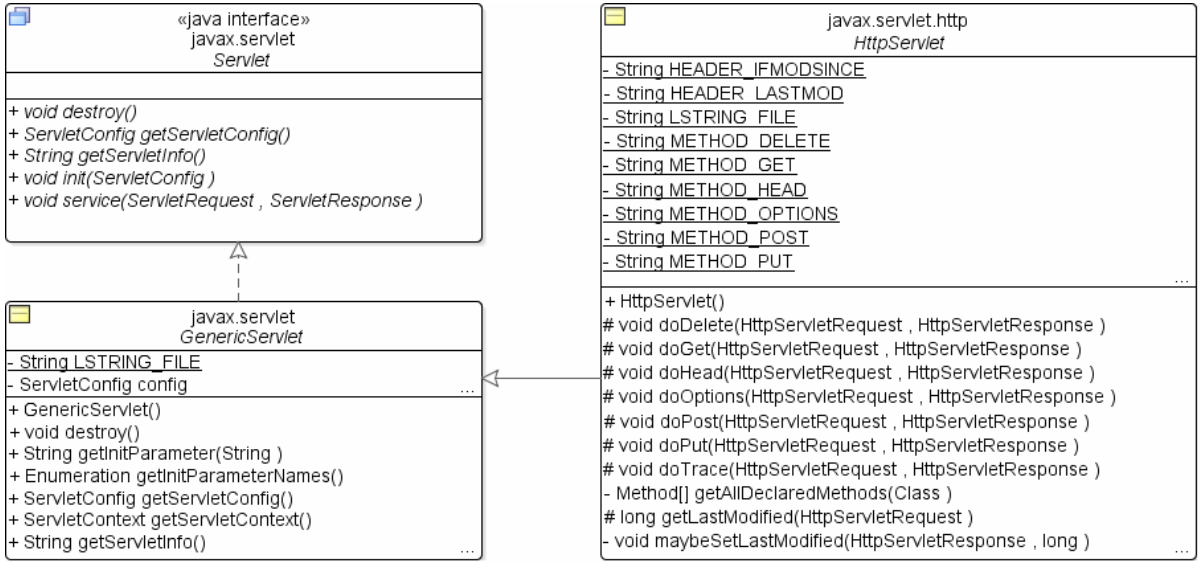
- void init(ServletConfig config): es invocado una sola vez, por el contenedor del servidor JEE compatible donde se hospeda el servlet y se emplea para inicializarlo. Se ejecuta cuando se realiza el primer requerimiento del servlet.

Programación de Servlets



- void destroy(): es invocado por el contenedor antes de que el servlet se descargue de memoria y deje de prestar servicio.

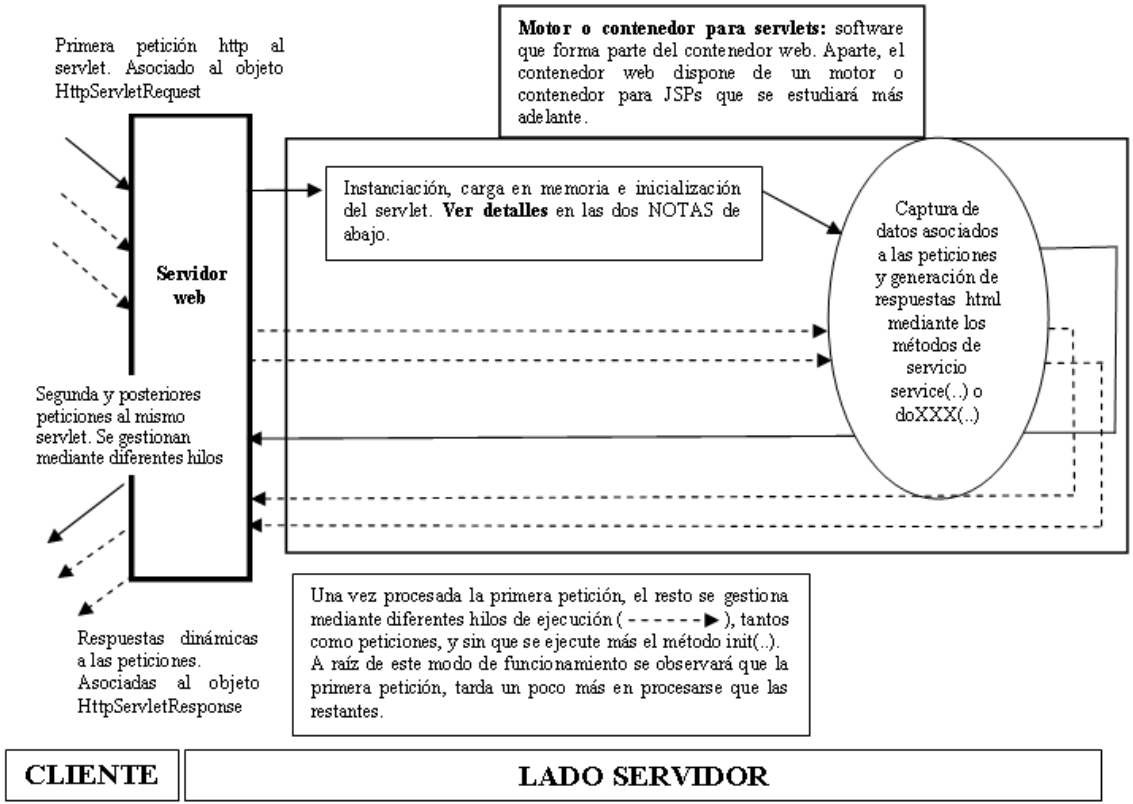
Programación de Servlets



- void service(ServletRequest request, ServletResponse reponse): es invocado por el contenedor para procesar el requerimiento, una vez que el servlet se ha inicializado. Es el llamado método de servicio. Sus argumentos son instancias de las interfaces javax.servlet.ServletRequest y javax.servlet.ServletResponse que modelan, respectivamente, el requerimiento del cliente y la respuesta del servlet.

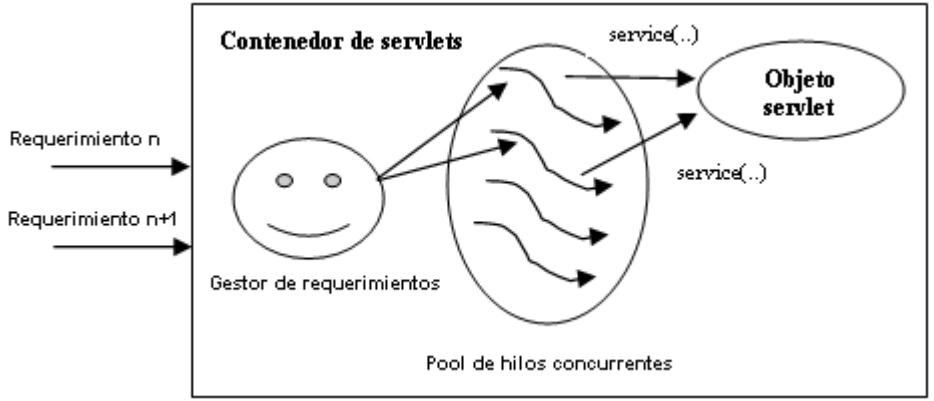
Programación de Servlets

Esquema de Funcionamiento



Programación de Servlets

- Esquema de Funcionamiento



- Finalizada la inicialización, el servlet ya está disponible para procesar los requerimientos y generar una respuesta a los mismos, con el método **service(ServletRequest request, ServletResponse response)**.
- Una vez procesado el primer requerimiento, el resto de requerimientos se gestiona mediante diferentes hilos de ejecución, tantos como requerimientos existan, tal como se puede apreciar en la figura y sin que se ejecute más el método **init(..)**.

Proyecto 02

- En este ejercicio ilustrare el uso de los métodos init(…) y destroy() de un servlet con acceso a la base de datos EurekaBank, utilizando JDBC.
- Se trata de hacer un programa que permita consultar las cuentas de una sucursal. Los recursos que construiremos son los siguientes:

Recurso	Nombre	Descripción
Página HTML	sucursal.html	Página que contiene un formulario para ingresar el código de la sucursal a consultar.
Servlet	Cuentas.java	Servlet que recibe el código de la sucursal y muestra un listado de las cuentas que le pertenecen.

Interacción con un Servlet

- Consideraciones Previas
  - Para hacer referencia a un servlet debemos tener en cuenta como es mapeado en el descriptor de despliegue (archivo web.xml) o en el mismo servlet utilizando la anotación @WebServlet.

```
<servlet>
    <servlet-name>Empleado</servlet-name>
    <servlet-class>servlets.Empleado</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Empleado</servlet-name>
    <url-pattern>/Empleado</url-pattern>
</servlet-mapping>
```

  - La etiqueta url-pattern representa el alias con que debemos hacer referencia al servlet, normalmente se utiliza el mismo nombre de la clase pero no tiene que ser así.

Interacción con un Servlet

- Escribiendo la URL del Servlet en un Navegador Web

  – Los servlets pueden ser llamados directamente escribiendo su URL en el campo dirección del navegador Web.

http://localhost:8080/Proyecto02/Cuentas?sucursal=001

Interacción con un Servlet

- Llamar a un Servlet desde dentro de una página HTML
  - Si el servlet está en otro servidor, debemos utilizar la URL completa.

```
<form method="post" action="http://localhost:8080/Proyecto02/Cuentas">
. . .
. . .
</form>


<a href="http://localhost:8080/Proyecto04/Cuentas?sucursal=001">
        Consultar
</a>
```

Interacción con un Servlet

- Llamar a un Servlet desde dentro de una página HTML
  - Si el servlet está en la misma aplicación sólo debemos hacer referencia al alias del servlet.

```
<form method="post" action=«Cuentas">
. . .
. . .
</form>


<a href=“Cuentas?sucursal=001">
        Consultar
</a>
```

Interacción con un Servlet

- Llamada a un Servlet desde otro Servlet
  - Tenemos dos posibilidades, ejecutar un sendRedirect() o un forward(), que tienen el mismo objetivo, pero que funcionan diferente.
  - A continuación tenemos sus diferencias:
    - forward() se ejecuta completamente en el servidor. Mientras que sendRedirect() conlleva a responder con un mensaje HTTP y esperar a que el navegador cliente acuda a la URL especificada. Es por ello que forward() es más rápido. Y es por ello que sendRedirect() modifica la URL del navegador.
    - forward() permite llamar a un serlvet o página JSP. Por el contrario en sendRedirect() se indica una URL que puede ser incluso una URL externa como "http://gcoronelc.blogspot.com" o cualquier otra.
    - En un forward() se pasan dos argumentos: request y response. Esto permite pasar objetos en el scope request. Mientras que en sendRedirect() los únicos parámetros que se pueden pasar son los de una URL "...?parametro1=valor1....". Obviamente también se podría usar otro scope, pero no el scope request.

Interacción con un Servlet

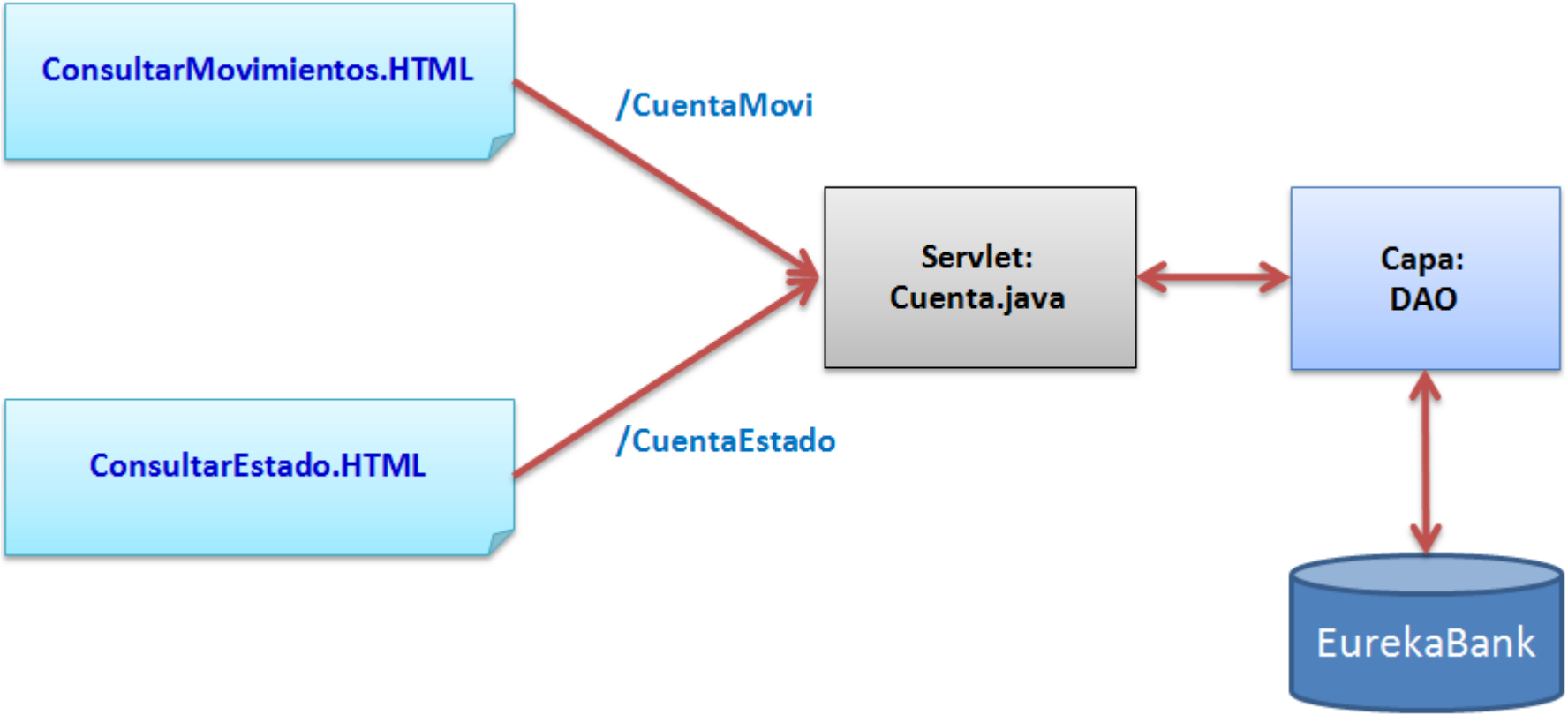
- Llamada a un Servlet desde otro Servlet
  - Supongamos que tenemos dos servlets de nombre Datos y Respuesta. A continuación tenemos dos ejemplos, uno utilizando sendRedirect() y otro utilizando forward().
    - Desde el servlet Datos se realiza un sendRedirect() al servlet Respuesta:

```
response.sendRedirect("Respuesta");
```

    - Desde el servlet Datos se realiza un forward() al servlet Respuesta:

```
RequestDispatcher rd = request.getRequestDispatcher("Respuesta");
rd.forward(request, response);
```

Servlets con Múltiples Mapeos



Servlets con Múltiples Mapeos

- Servlet 2.x

```
<servlet>
        <servlet-name>Cuenta</servlet-name>
        <servlet-class>servlets.Cuenta</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>Cuenta</servlet-name>
        <url-pattern>/CuentaEstado</url-pattern>
</servlet-mapping>
<servlet-mapping>
        <servlet-name>Cuenta</servlet-name>
        <url-pattern>/CuentaMovi</url-pattern>
</servlet-mapping>
```

- Servlet 3.x

```
@WebServlet(name = "Cuenta", urlPatterns = {"/CuentaEstado", "/CuentaMovi"})
public class Cuenta extends HttpServlet {

}
```

Servlets con Múltiples Mapeos

- Programación
  - Desde ConsultarMovimientos.HTML

```
<form method="post" action="CuentaMovi">

        . . .
        . . .

</form>
```

Servlets con Múltiples Mapeos

- Programación
  - Desde ConsultarEstado.HTML

```
<form method="post" action="CuentaEstado">

        . . .
        . . .

</form>
```

Servlets con Múltiples Mapeos

- Programación

```
@Override
protected void service(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

    String urlServlet = request.getServletPath();

    if (urlServlet.equals("/CuentaMovi")) {
       . . .
    } else if (urlServlet.equals("/CuentaEstado")) {
      . . .
    }

}
```

Proyecto 03

- Desarrollar una aplicación web que permita consultar:
  - El estado de una cuenta.
  - Los movimientos de una cuenta.
- Debe utilizar un servlet que responda a ambos requerimientos.