iSIL Instituto San Ignacio

Aprende haciendo

Desarrollo de Software I

Servlets

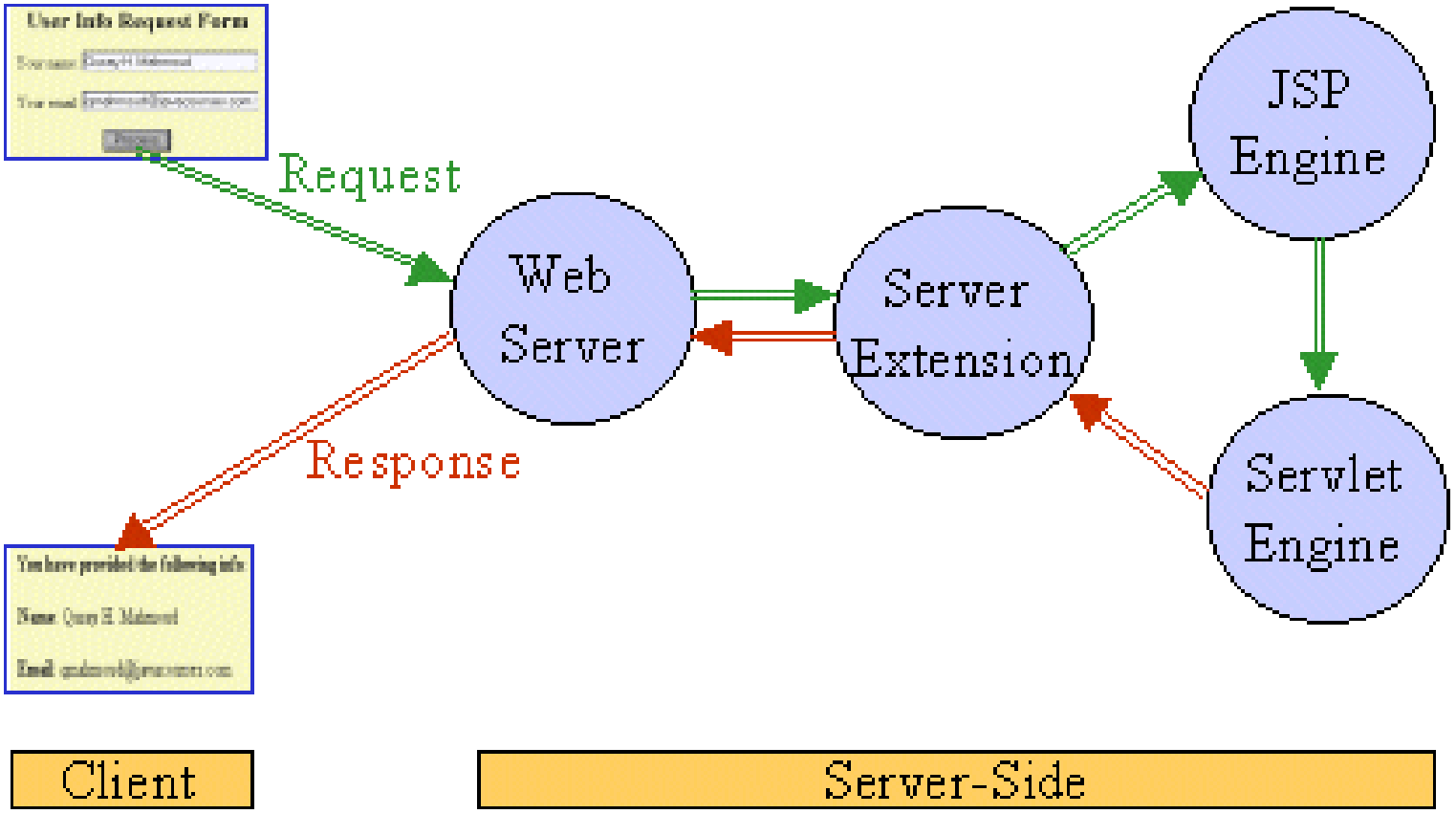
Ing. Eric Gustavo Coronel Castillo
gcoronelc@gmail.com
gcoronelc.blogspot.com

Índice

- Objetivo
- ¿Qué es un servlet?
- Arquitectura del Paquete Servlet
- Interacción con los Clientes
- Programación de Servlets
- Servlets y JavaBeans
- Interacción con un Servlet

Objetivo

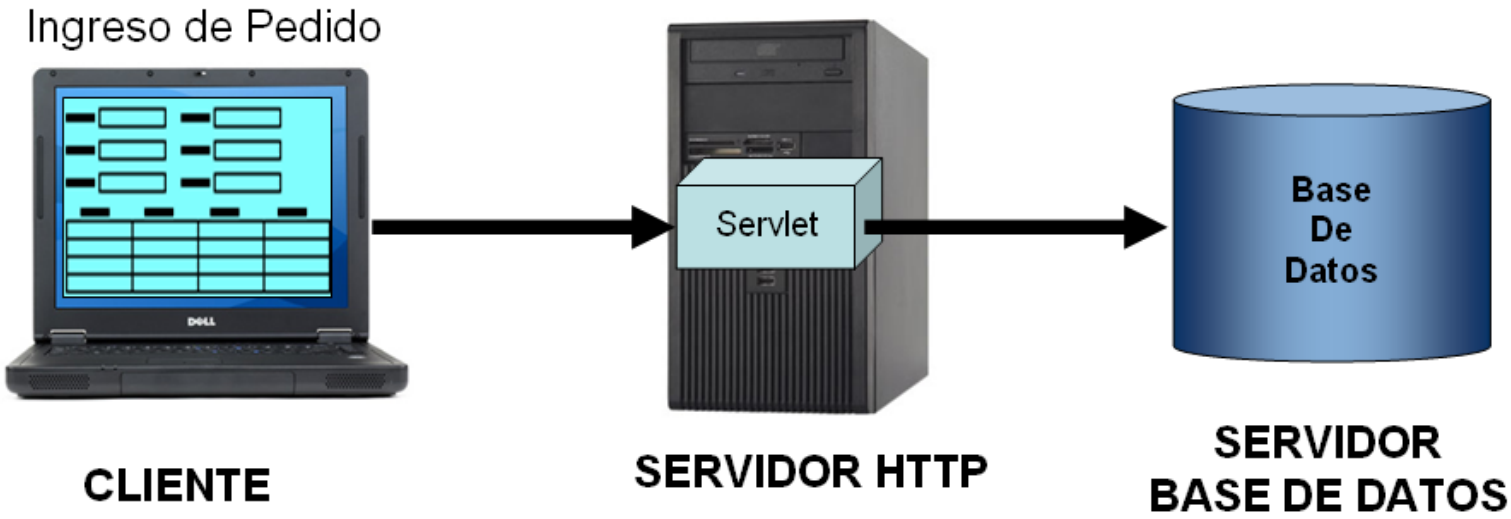
- Entender el funcionamiento de los servlets.
- Aplicar servlets en el desarrollo de aplicaciones web.



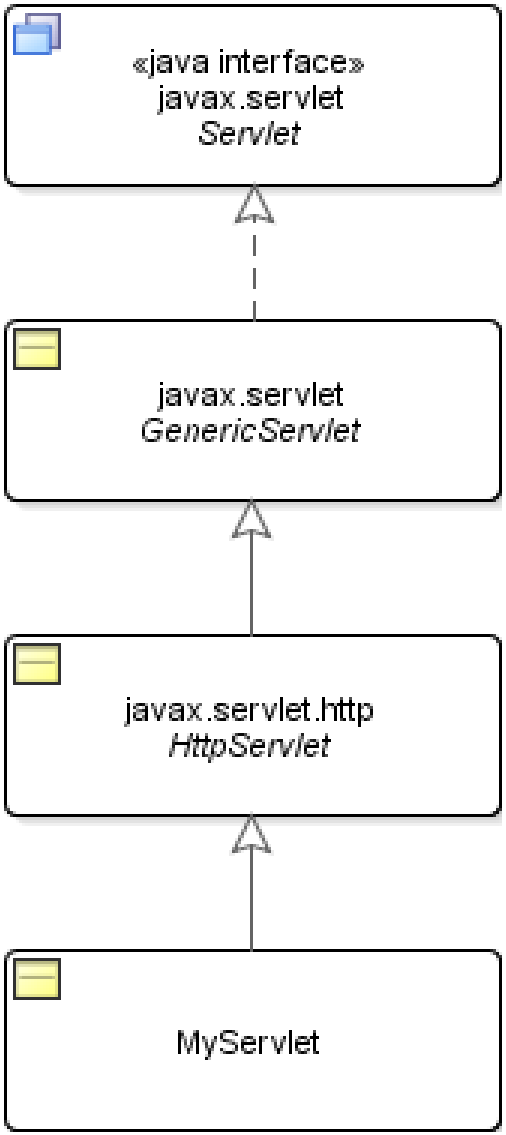
¿Qué es un Servlet?

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.



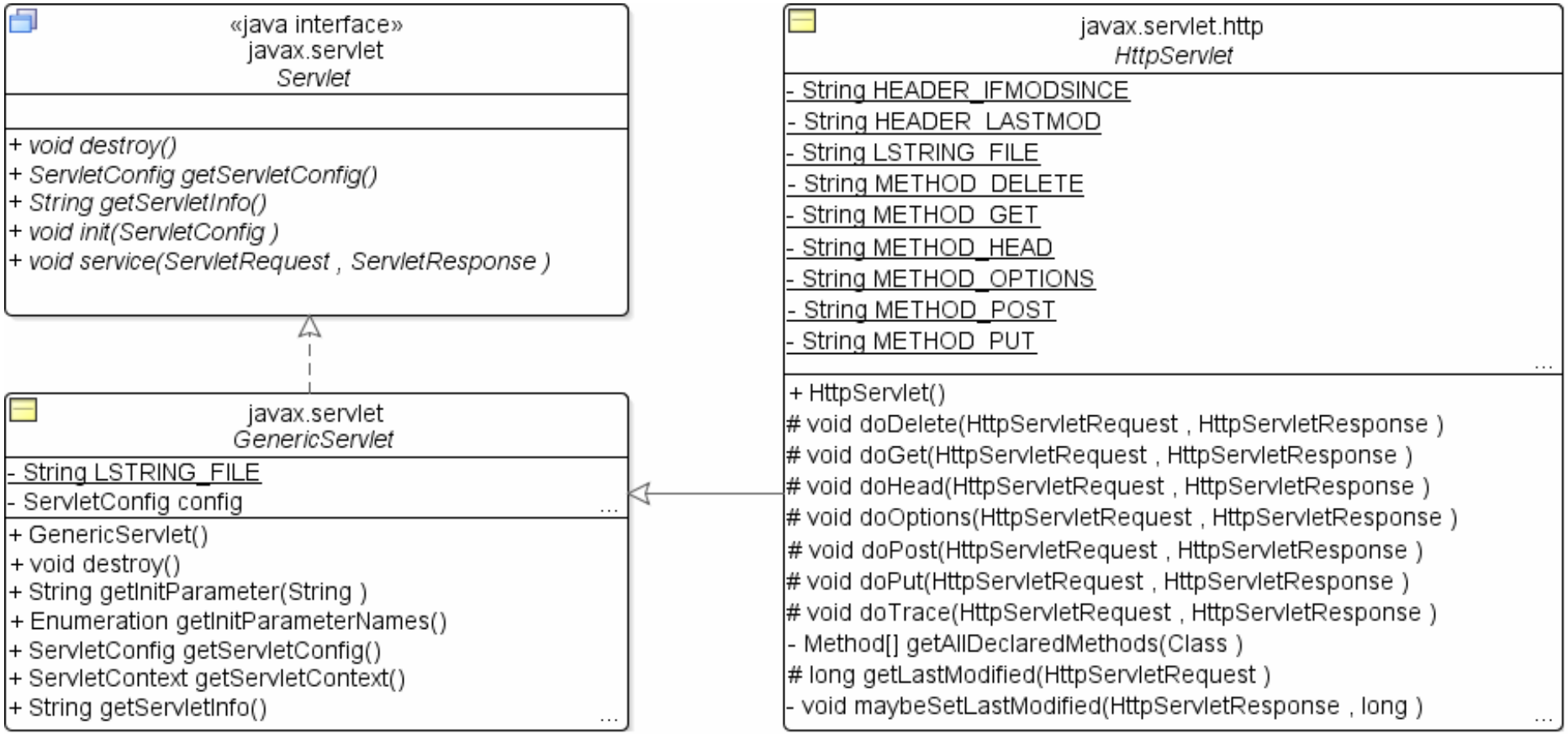
Arquitectura del Paquete Servlet



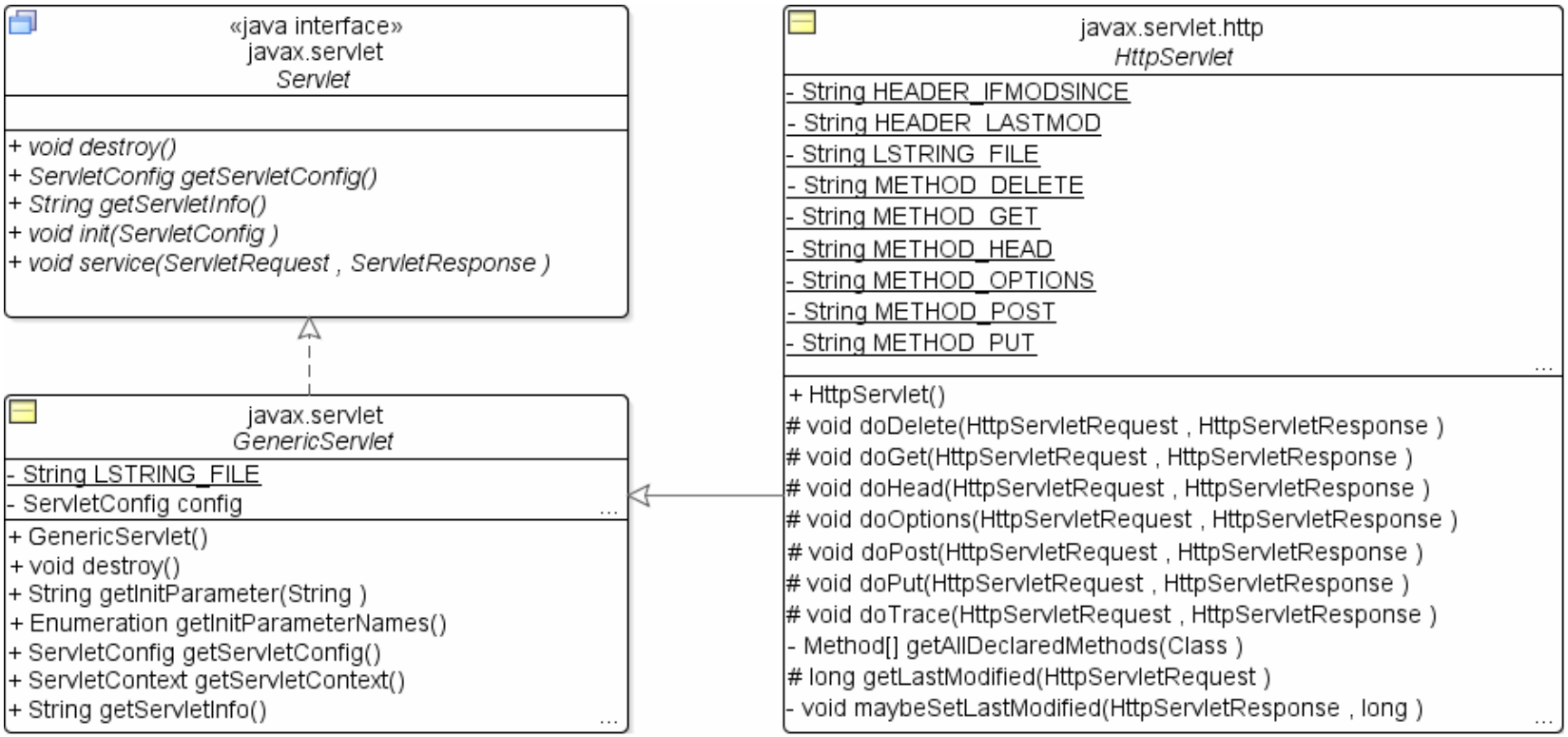
Interacción con los Clientes

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

Programación de Servlets

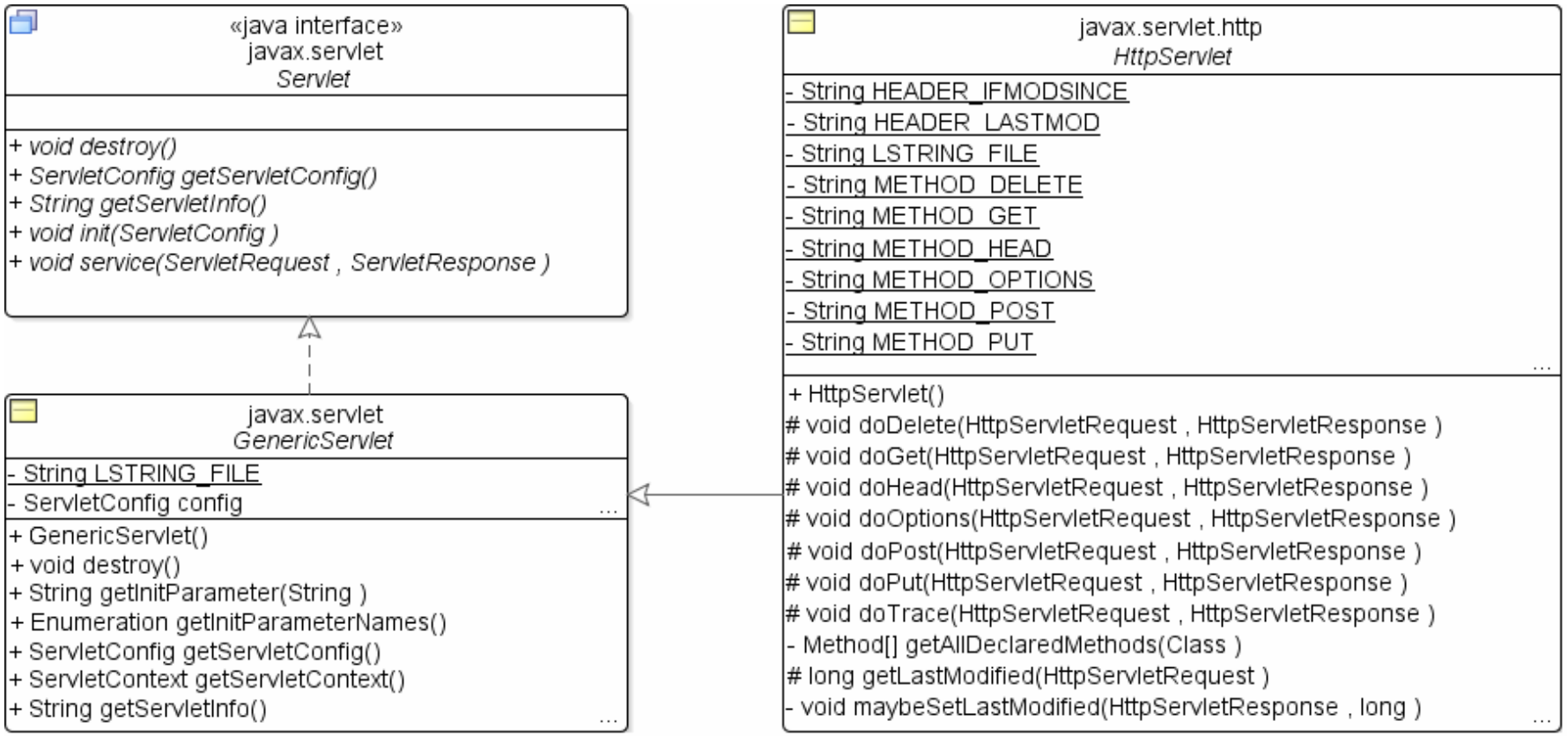


Programación de Servlets



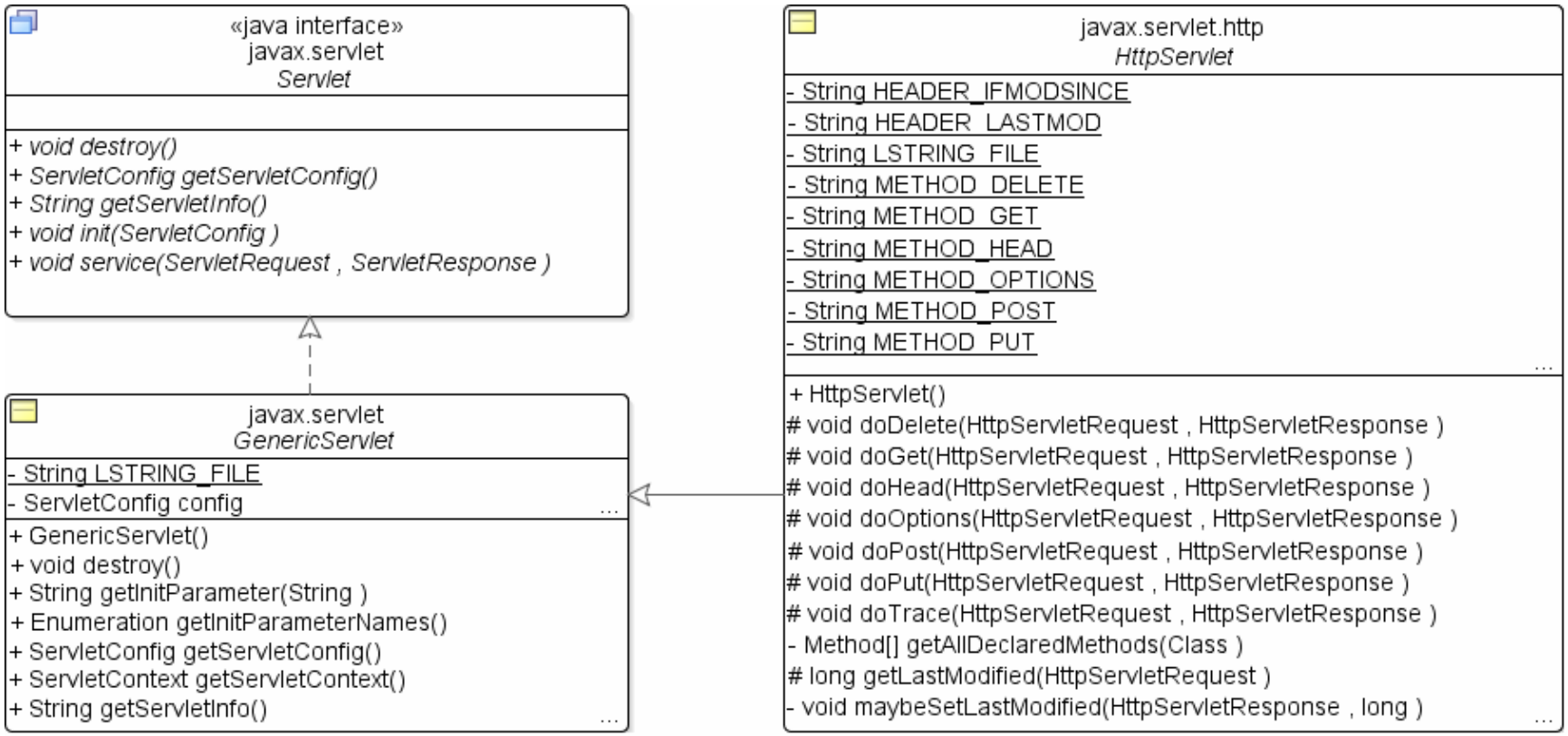
- **void init(ServletConfig config)**: es invocado una sola vez, por el contenedor del servidor JEE compatible donde se hospeda el servlet y se emplea para inicializarlo. Se ejecuta cuando se realiza el primer requerimiento del servlet.

Programación de Servlets



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

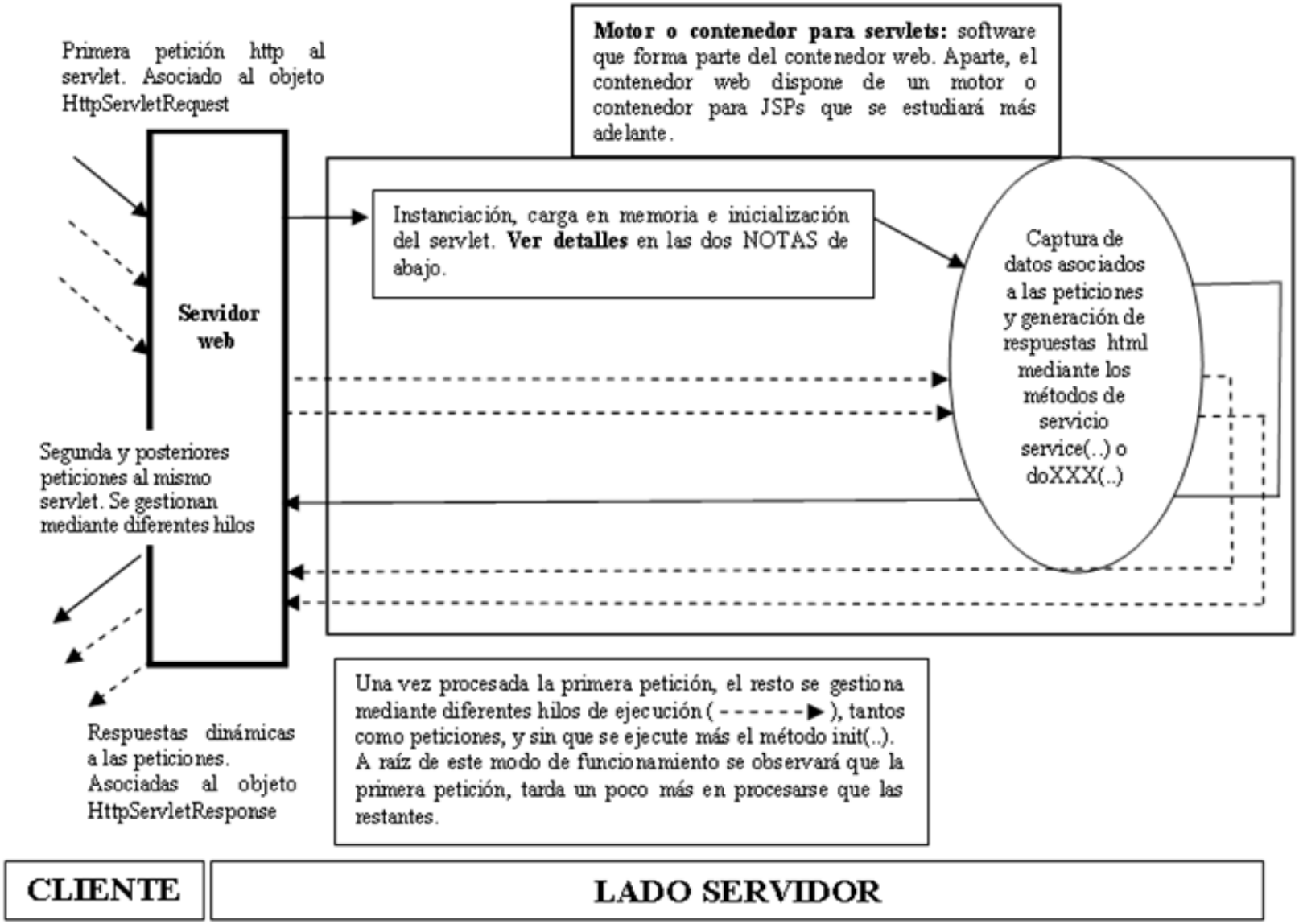
Programación de Servlets



- **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.

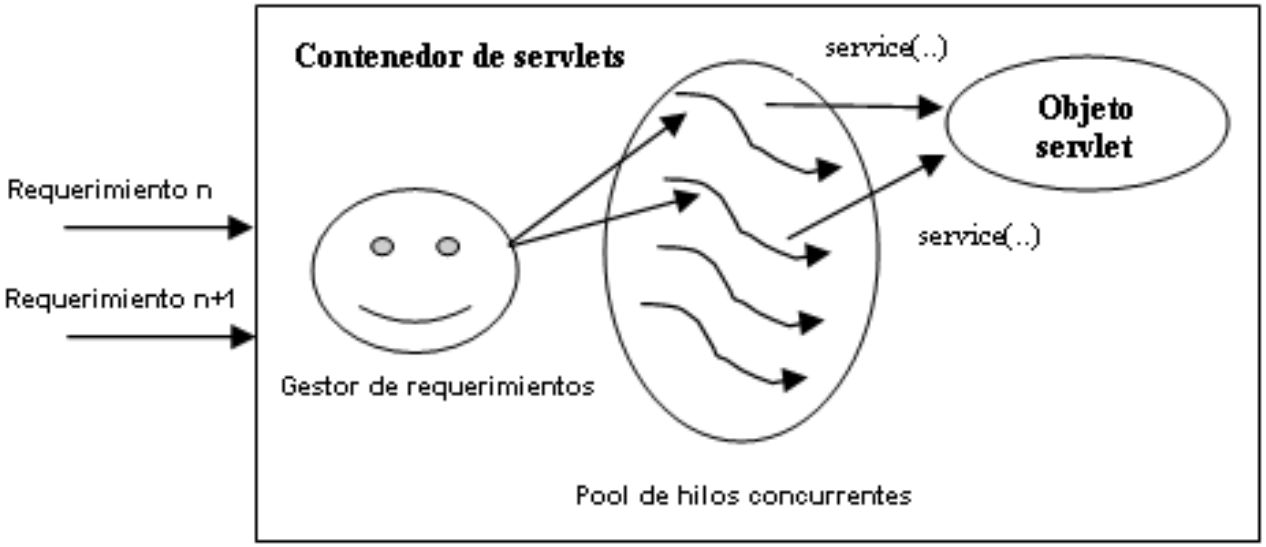
Programación de Servlets

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(..)**.

Servlets y JavaBeans

No es una buena práctica programar el acceso a la base de datos en los servlets, algunas de las razones son las siguientes:

1. Cada servlet estaría consumiendo una conexión a la base de datos.

2. Estaríamos juntando en una sola clase la lógica de control de la aplicación, la lógica del negocio y la capa de acceso a la base de datos, y de lo que se trata es de desacoplar estas capas para un mejor control, desarrollo y reutilización de componentes.

3. Para la capa de acceso a la fuente de datos deberíamos utilizar el patrón de diseño DAO y si tenemos varias fuentes de datos deberíamos utilizar Factory Patterns, ó quizás un framework ORM.

4. También podemos aplicar Factory Patterns a la lógica de negocio.

Interacción con un Servlet

❖ Consideraciones Previas

- Para hacer referencia a un servlet debemos tener en cuenta como se mapeado en el descriptor de despliegue (archivo web.xml).

```
<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/Proyecto04/empleado?sucursal=001

Interacción con un Servlet

❖ Llamar a un Servlet desde dentro de una página HTML

1. Si el servlet está en otro servidor, debemos utilizar la URL completa.

```
<form method="post" action="http://localhost:8080/Proyecto04/empleado">
. . .
. . .
</form>


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

Interacción con un Servlet

❖ Llamar a un Servlet desde dentro de una página HTML

2. Si el servlet está en la misma aplicación sólo debemos hacer referencia al alias del servlet.

```
<form method="post" action="empleado">
. . .
. . .
</form>
```

```
<a href="empleado?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://www.google.com" o cualquier otra.

- En un **forward()** se pasan dos argumentos: request y response. Esto permite pasar objetos en el scope request, por ejemplo. 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);
```

Bibliografía

- Desarrollando Soluciones con Java y MySQL Server
  Eric Gustavo Coronel Castillo
- Piensa en Java
  Bruce Eckel
- Como Programar en Java
  Deitel y Deitel
- Java 2
  Steven Holzner
- La Biblia de Java 2 v5.0
  Herbert Schildt
- Acceso a Bases de Datos con Java-JDBC
  Ángel Esteban

Referencias

- http://www.codestyle.org/java/servlets/faq-API.shtml
- http://www.java2s.com/Code/Java/Servlets/ServletMultipleInclude.htm
- http://www.wdvl.com/Style/Java/Servlets/request.html
- http://www.vc.ehu.es/jiwotvim/ISOFT2007-2008/Teoria/BloqueV/RequestDispatcher.pdf
- http://www.adrformacion.com/cursos/javaser/leccion3/tutorial5.html
- http://raultinoco-cea2.blogspot.com/2009/02/metodos-forward-init-y-sendredirect.html
- http://java.cabezudo.net/trabajos/JEE5/manual/jee5.v0.01.00/index.html