Introducción a los servlets

¡Comencemos!

¿Qué aprenderás?

- Conocer la teoría detrás de los servlets
- Conocer el ciclo de vida de los servlets
- Aplicar ejemplo de uso del ciclo de vida

Introducción

Los servlets son componentes java que tienen la capacidad de generar contenido web dinámico mediante request y response http. Esta tecnología extiende las capacidades de un sistema web que reside dentro de un contenedor web, el cual le otorga una serie de servicios de plataforma.

Utilizando servlets puedes perfectamente generar una página web completa solamente con código java y además utilizar todo el poderío de la orientación a objetos y la vasta biblioteca de clases que ofrece el jdk. Esto se traduce en que puedes armar un sistema web dinámico, con manejo de variables y funciones java como si fuera una aplicación de consola, pero con una salida web mediante http y página html.

Los servlets

Como mencionamos hace poco, un servlet no es más que una clase java que trabaja dentro de un contenedor de servlets como Apache Tomcat, el cual es el encargado de extender la funcionalidad de la clase y darle más "poder" por decirlo de una forma. El servlet fuera del contenedor Tomcat pierde todo su poder y se convierte en un archivo de texto más, con extensión java imposible de ejecutar, ya que esta clase no cuenta con un método "Main" que ejecute la rutina.

Para graficar el flujo de trabajo de una petición web, pensemos en una página web que solicita algún tipo de recurso (puede ser desde un formulario web hasta una galería de fotos, como también el cálculo del iva de algún valor) a un servlet. El servlet recibe la petición mediante el protocolo HTTP y hace su trabajo (puede ser crear los campos del formulario web, ir a buscar una imagen de la galería de fotos o hacer el cálculo del iva) para luego mediante el response (la respuesta) devuelve el recurso al navegador.

La imagen 1 muestra el flujo normal de petición entre clientes (navegador web) y servidor (contenedor de servlets, aquí están los servlets). Esta es la arquitectura típica de un sistema web, la cual es conocida como cliente-servidor.

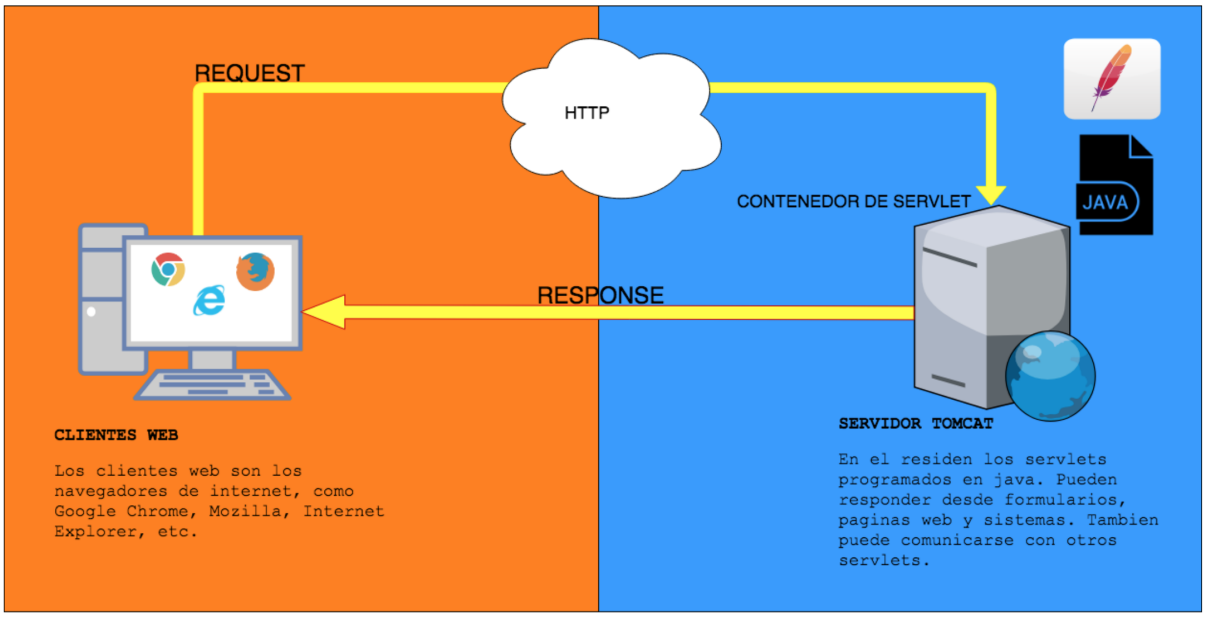

Imagen 1. Flujo cliente web a servlet.
Fuente: Desafío Latam

Ciclo de vida de un Servlet

El ciclo de vida completo de un servlet es administrado por el contenedor de servlet y, antes de empezar a programarlos, hay que entender cómo funciona la creación y el funcionamiento de estos elementos de software.

El ciclo de vida se divide en 4 pasos bien definidos:

1. Carga del servlet.
2. Inicialización del servlet.
3. Captura del request.
4. Destrucción del request.

Carga del servlet

La primera etapa del ciclo de vida de un servlet parte con su inicialización gracias al servidor Tomcat. El servidor ejecuta dos pasos en esta etapa:

- **Loading**: Carga de la clase servlet.

- **Instantiation**: Creación de una instancia del servlet. Para crear una nueva instancia del servlet, el contenedor utiliza un constructor sin parámetros.

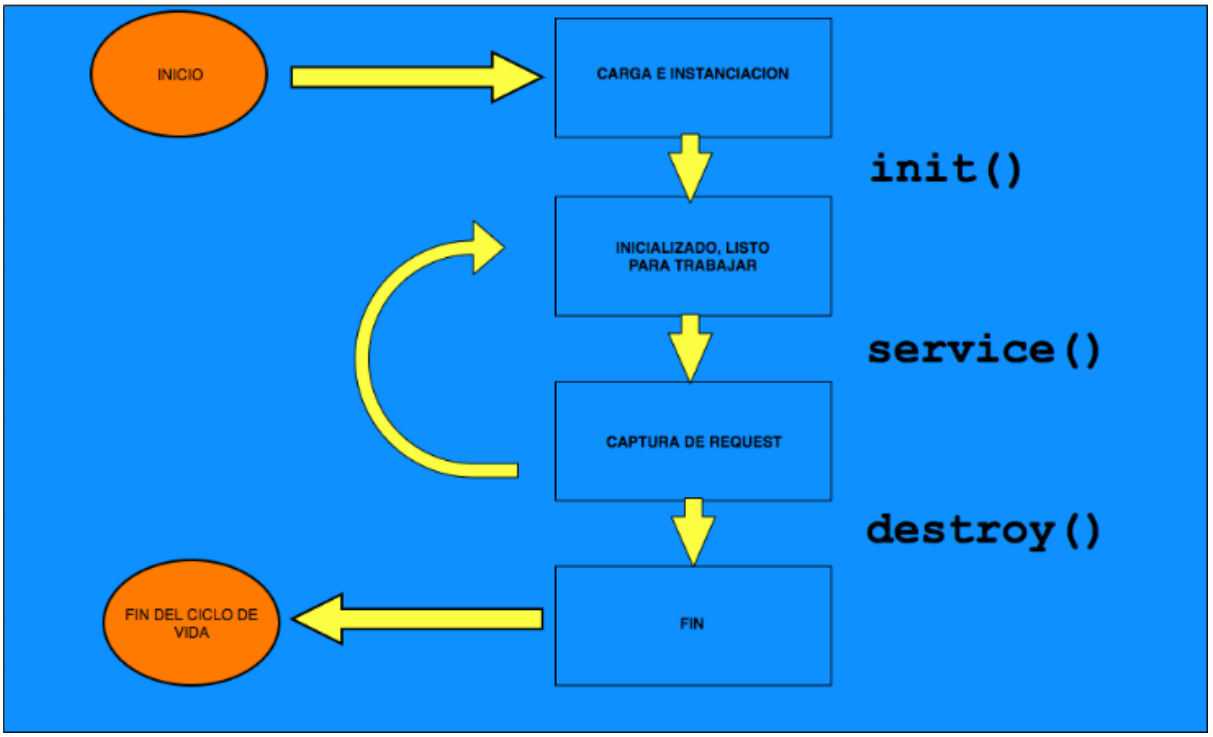

Imagen 2. Ciclo de vida de un servlet.
Fuente: Desafío Latam

Inicialización de Servlets

Después de que el servlet fue instanciado correctamente, el contenedor se encarga de inicializarlo. Ésto lo realiza invocando al método Servlet.init(), que acepta la referencia del objeto *ServletConfig* como parámetro.

El método *Servlet.init()* es invocado por el servidor solamente una vez, inmediatamente después de que el objeto *Servlet.init(ServletConfig)* sea instanciado correctamente. En caso de que el servlet falle en su inicialización, el contenedor lanza las excepciones: *ServletException* y *UnavailableException*.

Captura del Request

Después de la inicialización, la instancia del servlet está lista para atender los request del cliente. El contenedor de servlet realiza las siguientes operaciones cuando la instancia del servlet está lista:

- Crea los objetos *ServletRequest* y *ServletResponse*.
- Después de crear los objetos *ServletRequest* y *ServletResponse*, se invoca al método Servlet.service (*ServletRequest*, *ServletResponse*) al pasar los objetos de solicitud y respuesta.

Mientras el método *service()* procesa el request puede lanzar las excepciones *ServletException*, *UnavailableException*, e *IOException*.

Destruir el Servlet

Cuando el contenedor decide destruir el servlet, ejecuta las siguientes acciones:

- Permite que todos los subprocesos que se ejecutan actualmente en el método de servicio de la instancia de *Servlet* completen sus trabajos y sean liberados.
- Después de que los procesos en ejecución hayan completado sus trabajos, el contenedor *Servlet* llama al método *destroy()* en la instancia de Servlet.
- Una vez ejecutado el método *destroy()*, Del Servlet *container* libera todas las referencias de esta instancia de Servlet para que sea elegible para el *garbage collector*.

Para graficar el ciclo de vida, lo haremos mediante un pequeño servlet, el cual solamente genera una respuesta en html. El proyecto completo lo puede descargar desde *Material Apoyo Lectura*, Ciclo de Vida. El código a examinar es el siguiente:

```
    import java.io.IOException;
```

```
    import java.io.PrintWriter;
    import javax.servlet.ServletException;
    import javax.servlet.annotation.WebServlet;
    import javax.servlet.http.HttpServlet;
    import javax.servlet.http.HttpServletRequest;
    import javax.servlet.http.HttpServletResponse;
    @WebServlet("/cicloVida")
    public class CicloDeVida extends HttpServlet {
       private static final long serialVersionUID = 1L;
       private String saludo;
       public void init() throws ServletException {
              saludo = ("Hola servlet inicializando...");
       }
       public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
              // Seteamos el tipo de response, un html
              response.setContentType("text/html");
              // Declaramos la logica
              PrintWriter out = response.getWriter();
              out.println("<h1>" + saludo + "</h1>");
       }
       public void destroy() {
              // Se destruye el servlet, nada que hacer
       }
    }
```