FILTROS

Eduard Lara

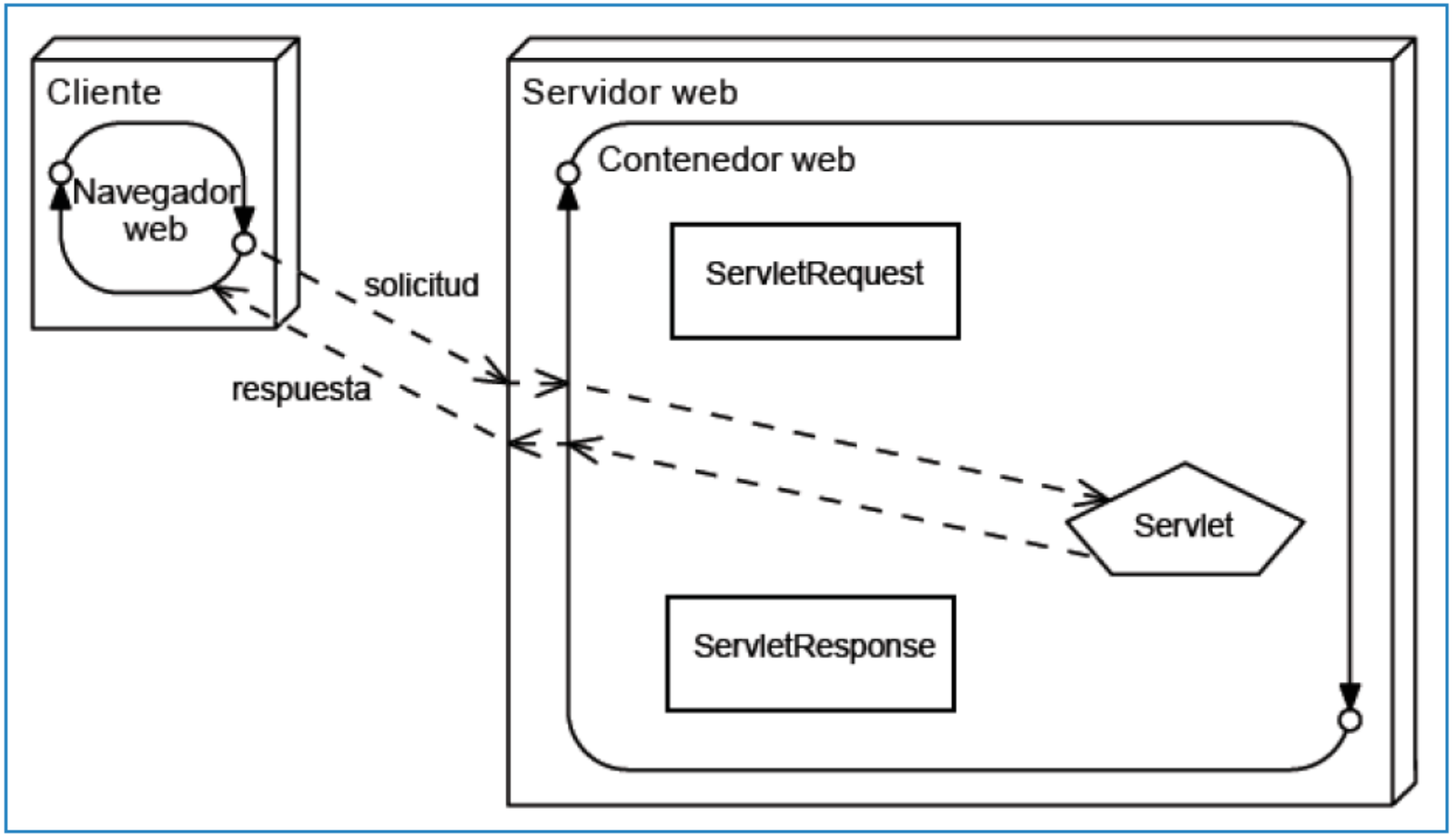
INDICE

1. FILTROS

- Una de las ventajas de la estructura de servlet es que el contenedor web intercepta las solicitudes entrantes antes de que lleguen al código, preprocesando la solicitud y la funcionalidad adicional (como la seguridad).
- El contenedor también puede posprocesar la respuesta que genera el servlet.
- En este capítulo aprenderemos a agregar unos componentes denominados filtros para conseguir esta funcionalidad del contenedor.
- Estos filtros amplían el preprocesamiento de la solicitud y el posprocesamiento de la respuesta.

1. FILTROS

Petición y respuesta al servlet



2. APLICACIÓN DE FILTROS A SOLICITUDES ENTRANTES

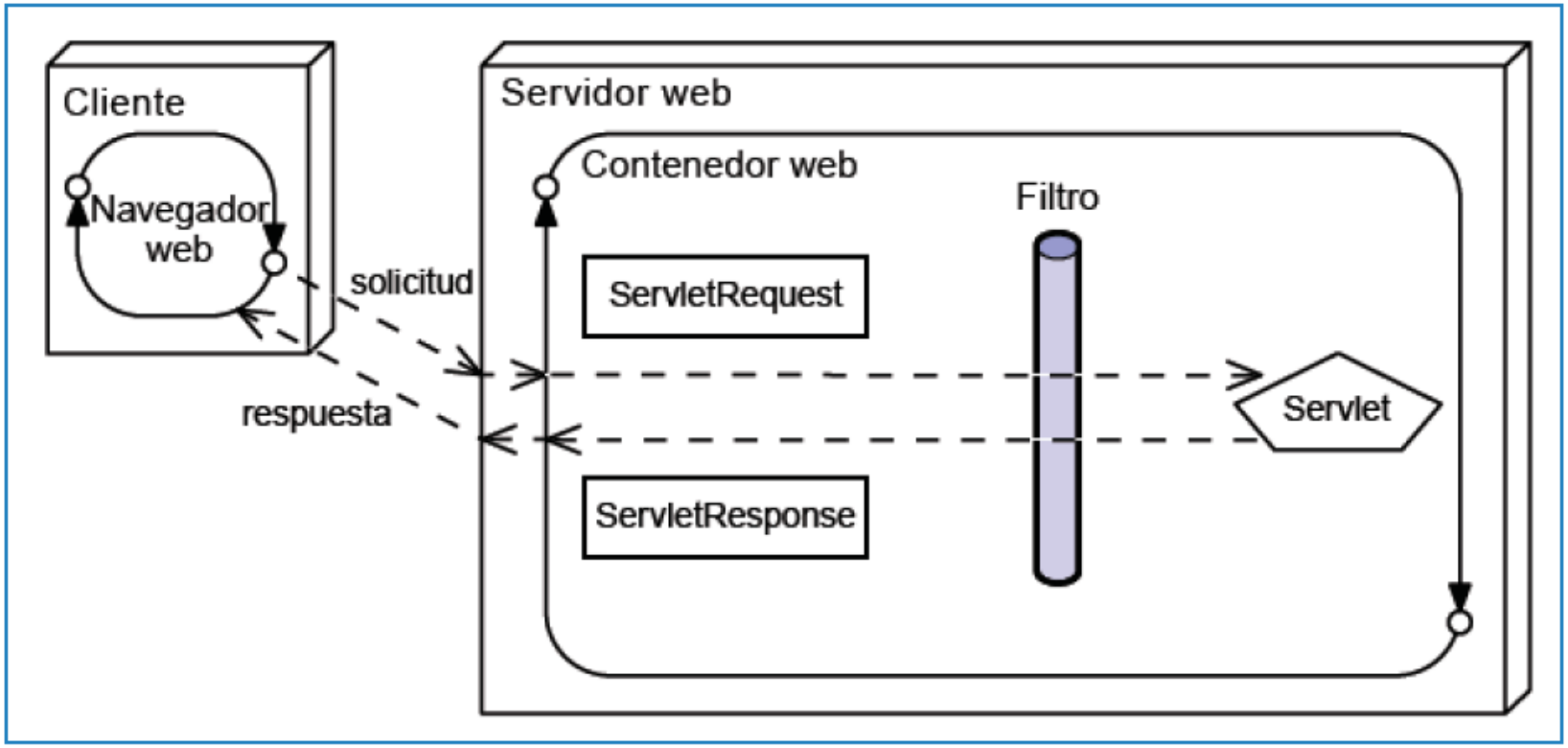
Los filtros son componentes que se pueden escribir y configurar para realizar más tareas de preprocesamiento y posprocesamiento. Cuando el contenedor web recibe una solicitud, se producen varias operaciones:

1. El contenedor web efectúa su preprocesamiento de la solicitud. Lo que sucede en este paso es responsabilidad del proveedor del contenedor.
2. El contenedor web comprueba si algún filtro tiene un patrón URL que coincida con la URL solicitada.
3. El contenedor web busca el primer filtro que tenga un patrón URL coincidente. Se ejecuta el código del filtro.

2. APLICACIÓN DE FILTROS A SOLICITUDES ENTRANTES

4. Si otro filtro tiene un patrón URL coincidente, a continuación se ejecuta su código. Este proceso continúa hasta que no quedan filtros que tengan patrones URL coincidentes.
5. Si no se produce ningún error, la solicitud pasa al servlet de destino.
6. El servlet devuelve la respuesta a su llamador. El último filtro aplicado a la solicitud es el primero que se aplica a la respuesta.
7. El último filtro que se aplicó a la solicitud es el primero que se aplica a la respuesta. El contenedor web puede realizar tareas de posprocesamiento en la respuesta.

2. APLICACIÓN DE FILTROS A SOLICITUDES ENTRANTES



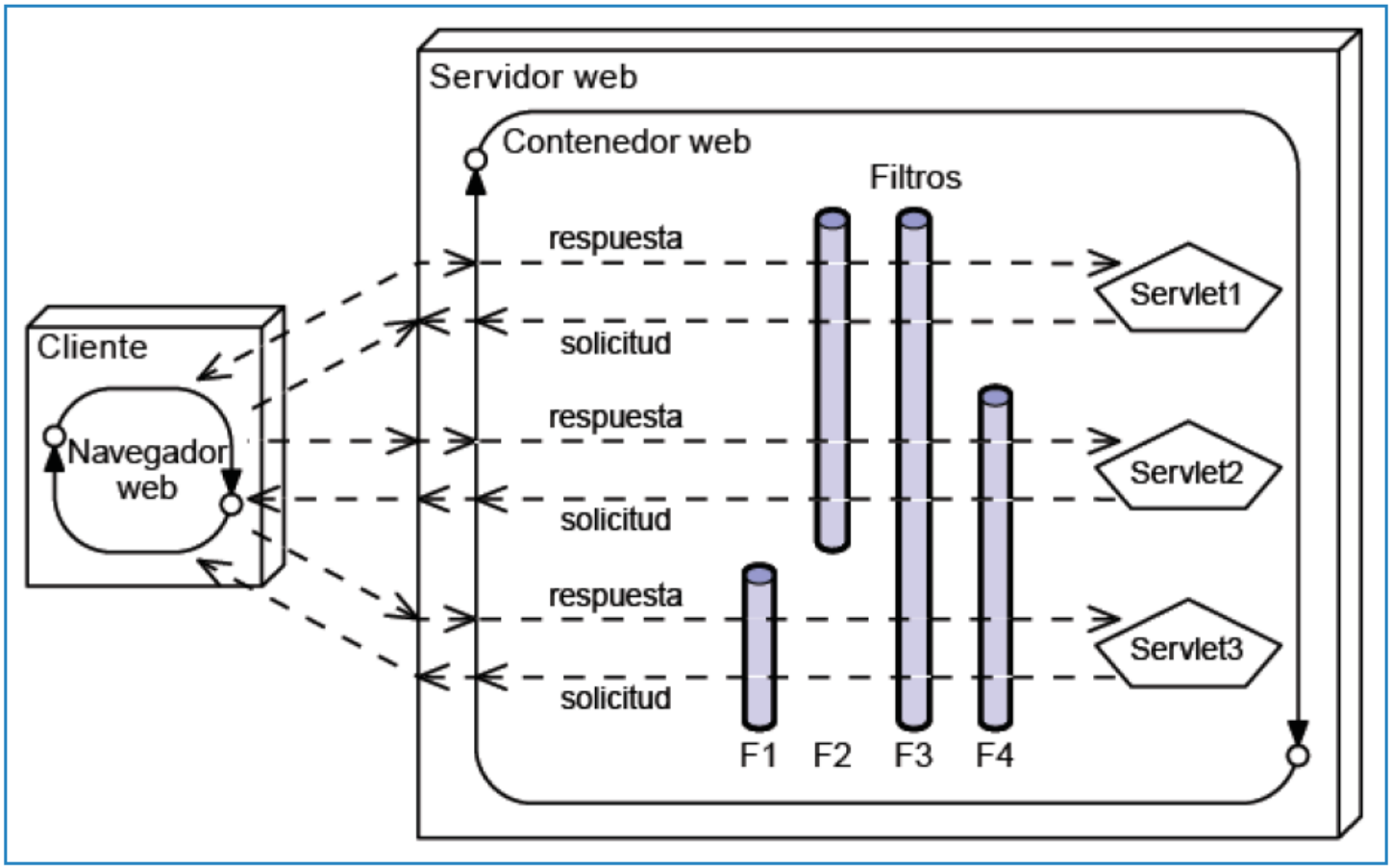
3. USO DE FILTROS

Se pueden usar filtros para operaciones como:

- Bloquear el acceso a un recurso basándose en la identidad del usuario o la pertenencia a una función.
- Auditar las solicitudes entrantes.
- Comprimir el flujo de datos de la respuesta.
- Transformar la respuesta.
- Medir y registrar el rendimiento del servlet.

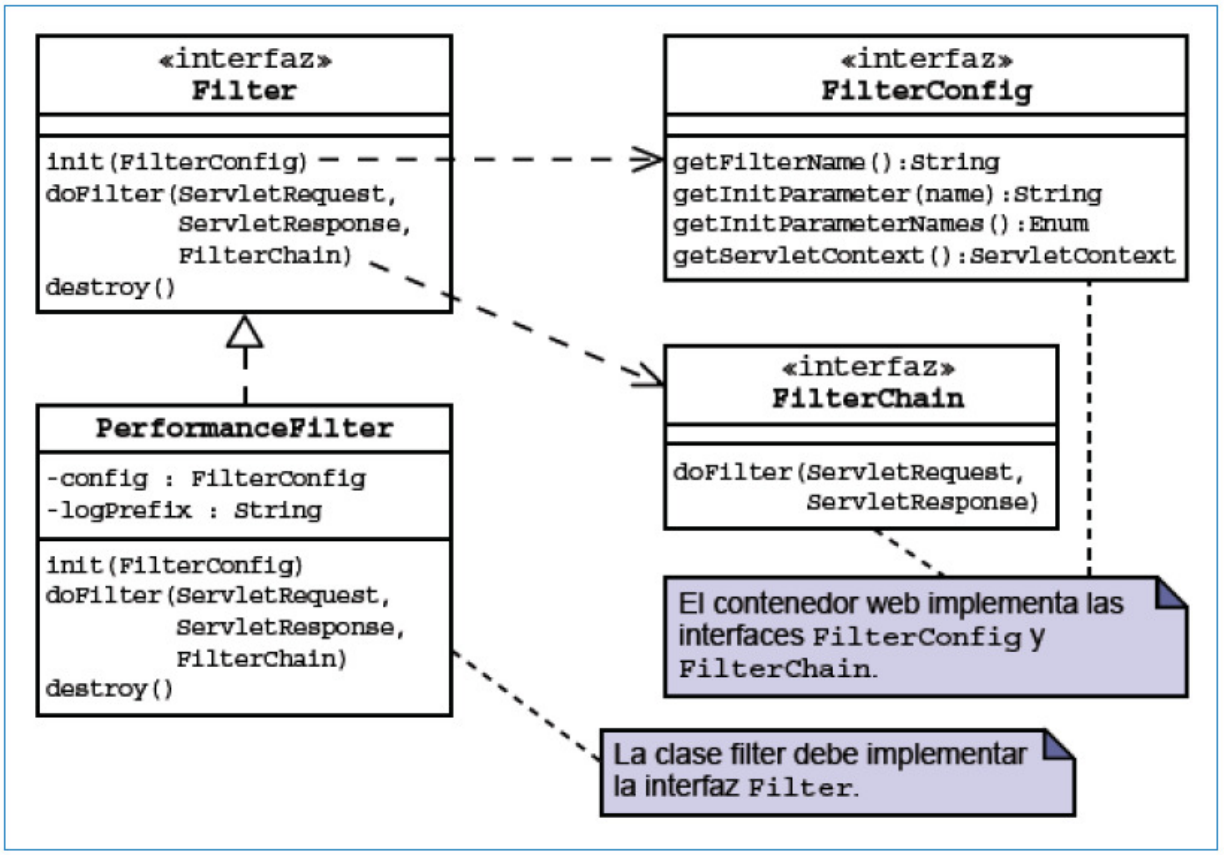
4. APLICACIÓN DE VARIOS FILTROS

Podemos aplicar varios filtros a distintos componentes web. Estos se ejecutarán en cascada.



5. API DE FILTRO

La API de filtro forma parte de la API de servlet básica. Las interfaces se hallan en el paquete javax.servlet.



5. API DE FILTRO

- Cuando se escribe un filtro, se implementa la interfaz Filter.
- El contenedor pasa al método init una referencia FilterConfig. Debe almacenar la referencia FilterConfig como una variable de instancia en el filtro.
- La referencia FilterConfig sirve para obtener parámetros de inicialización del filtro, el nombre del filtro y ServletContext.
- Cuando se llama al filtro, el contenedor pasa al método doFilter las referencias ServletRequest, ServletResponse y FilterConfig.
- La referencia FilterChain sirve para pasar los objetos de solicitud y respuesta al siguiente componente de la cadena (filtro o servlet de destino).

6. CICLO DE VIDA DE UN FILTRO

MÉTODO INIT

- El método init de un filtro se llama una vez cuando el contenedor instancia un filtro.
- A este método se le pasa FilterConfig, que suele almacenarse como una variable miembro para uso posterior.
- En el método init puede realizar cualquier tarea de una sola inicialización.
- Por ejemplo, si el filtro tiene parámetros de inicialización, se pueden leer en el método init aplicando el método getInitParameter del objeto FilterConfig.

6. CICLO DE VIDA DE UN FILTRO

MÉTODO DOFILTER

- El método doFilter se llama para cada solicitud interceptada por el filtro. 'Es el equivalente en filtro al método service de un servlet. Este método recibe tres argumentos: ServletRequest, ServletResponse y FilterChain. El objeto de solicitud se utiliza para obtener información del cliente, por ejemplo, sobre parámetros o encabezado. El objeto de respuesta se utiliza para devolver información al cliente, por ejemplo, valores de encabezado.
- Dentro del método doFilter, debe decidir si se invoca el siguiente componente de la cadena de filtros o si se bloquea la solicitud. Si prefiere invocar el siguiente componente, llame al método doFilter en la referencia FilterChain. El siguiente componente de la cadena puede ser otro filtro o un recurso web, como un servlet.

6. CICLO DE VIDA DE UN FILTRO

MÉTODO DESTROY

- Antes de que el contenedor web elimine una instancia de filtro del servicio, se llama al método destroy.
- Este método sirve para realizar las operaciones que deban producirse al final de la vida del filtro.

7. CONFIGURACIÓN DEL FILTRO

- Como el contenedor web es responsable de los ciclos de vida de los filtros, hay que configurar los filtros en el descriptor de despliegue de la aplicación web.
- Como mínimo, la declaración de un filtro en el archivo web.xml, debe contener los elementos filter-name y filter-class.
- Los parámetros de inicialización opcionales se incluyen dentro de la declaración del filtro en elementos init-param.

```
<filter>
    <filter-name>FiltroMedirTiempo</filter-name>
    <filter-class>app.web.FiltroMedirTiempo</filter-class>

</filter>
```

8. DECLARACIÓN DE UNA ASIGNACIÓN DE FILTRO EN EL ARCHIVO WEB.XML

- Los filtros se ejecutan cuando se solicita el recurso al que están asignados.
- El elemento url-pattern puede especificar una URL determinada o utilizar un carácter comodín para indicar un conjunto de URL.
- El elemento url-pattern puede sustituirse por el elemento servlet-name para crear un filtro que se aplique a un servlet determinado.

```
<filter-mapping>
    <filter-name>FiltroMedirTiempo</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

8. DECLARACIÓN DE UNA ASIGNACIÓN DE FILTRO EN EL ARCHIVO WEB.XML

- Los mismos valores de url-pattern y servlet-name pueden emplearse en múltiples asignaciones de filtro para crear una cadena de filtros que se aplican a la solicitud.
- Los filtros se aplican por el orden en que aparecen en el descriptor de despliegue.
- Todos los filtros con asignación de URL se aplican antes que los filtros con asignación de servlet.
- Los filtros sólo suelen aplicarse a las solicitudes que proceden directamente de un cliente.
- Si suministra un elemento dispatcher para la asignación de filtro, es posible aplicar filtros tanto a las distribuciones internas como a las solicitudes directas de los clientes.

8. DECLARACIÓN DE UNA ASIGNACIÓN DE FILTRO EN EL ARCHIVO WEB.XML

- Un elemento dispatcher con el valor REQUEST indica que el filtro se aplicará a las solicitudes de un cliente.
- Un elemento dispatcher con el valor INCLUDE invocará el filtro se para las llamadas RequestDispatcher.include correspondientes.
- Un elemento dispatcher con el valor FORWARD invocará el filtro se para las llamadas RequestDispatcher.forward correspondientes.
- Un elemento dispatcher con el valor ERROR indica que el filtro se aplicará cuando se produzca un error (la distribución se debe a una condición de error).
- Si quiere aplicar el filtro en múltiples escenarios, puede suministrar múltiples elementos dispatcher.

RECUERDA QUE...

- Un filtro es un componente que permite interceptar peticiones y respuestas contra una petición determinada.
- Los filtros se declaran y mapean en el web.xml