Applications Web

Cours 3bis: Filtres

Introduction

- Introduits dans la version 2.3 de la spec.
- Intercepter les requêtes et les réponses et de les transformer ou pour utiliser les informations qu'elles contiennent.
- Ne créent pas de réponses, mais fournissent des fonctions "universelles" qui peuvent être associées à n'importe quelle servlet.

Introduction

- Permettent de réaliser des tâches récurantes.
- De renvoyer différents formats au client
- Exemples:
  - Authentification, journal d'accès, conversions, localisation, transformation XSL, ...

Ecrire des filtres

- **L'API des filtres est définie dans 3 classes:** Filter, FilterChain **et** FilterConfig **dans** javax.servlet
  - http://java.sun.com/j2ee/sdk_1.3/techdocs/api/javax/servlet/Filter.html
- **On définit un filtre en implantant la classe** Filter, **une chaîne de filtres (**FilterChain**) décrit comment une série de filtres est exécuté et le** FilterConfig **contient les données d'initialisation.**

Ecrire des filtres

- La méthode principale dans Filter est doFilter:
  - Examiner les entêtes de la requête
  - Modifier l'objet requête (entêtes, données, ...)
  - Modifier l'objet réponse
  - Appeler le filtre suivant
  - Examiner les entêtes de la réponse après un autre filtre
  - Lever des exceptions
- Deux méthodes init et destroy.
- Init est appelée au démarrage pour récupérer les paramètres de FilterConfig

Exemple

```
public final class FiltreCompteur implements Filter {
   private FilterConfig filterConfig = null;

   public void init(FilterConfig filterConfig) throws ServletException {
      this.filterConfig = filterConfig;
   }

   public void destroy() {
      this.filterConfig = null;
   }

   public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)  throws IOException, ServletException {
      if (filterConfig == null)
         return;
      StringWriter sw = new StringWriter();
      PrintWriter writer = new PrintWriter(sw);
      Compteur compteur = (Compteur)filterConfig.getServletContext().getAttribute("Compteur");
      writer.println();
      writer.println("Le nombre de visites est: " + compteur.incrementerCompteur());
      writer.println();

      // On écrit la chaîne
      writer.flush();
      filterConfig.getServletContext().log(sw.getBuffer().toString());
      ...
      chain.doFilter(request, wrapper);
      ...
   }
}
```

Configuration de filtre

- Déclarer le filtre dans le web.xml
  - **Deux directives** <filter> **et** <filter-mapping>

```
<filter>
   <filter-name>Filtre Compteur</filter-name>
   <filter-class>FiltreCompteur</filter-class>
</filter>
```

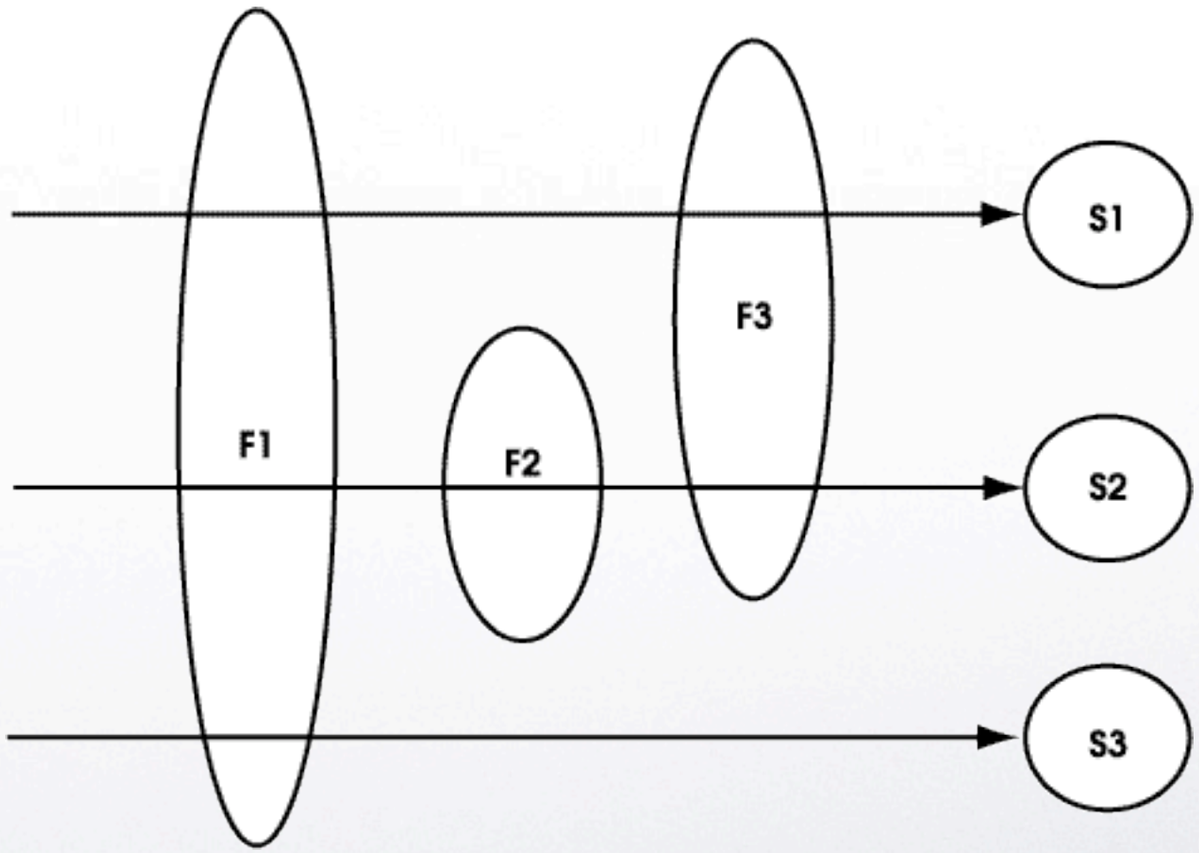
```
<filter-mapping>
   <filter-name>Filtre Compteur</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

Configuration de filtre

```
<filter>
   <filter-name>Compression Filter</filter-name>
   <filter-class>CompressionFilter</filter-class>
   <init-param>
      <param-name>compressionThreshold</param-name>
      <param-value>10</param-value>
   </init-param>
</filter>
...
<filter-mapping>
   <filter-name>Compression Filter</filter-name>
   <servlet-name>CompressionTest</servlet-name>
</filter-mapping>
<servlet>
   <servlet-name>CompressionTest</servlet-name>
   <servlet-class>CompressionTest</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>CompressionTest</servlet-name>
   <url-pattern>/CompressionTest</url-pattern>
</servlet-mapping>
```

Mapping

- Un filtre peut être associé à une ou plusieurs servlet.
- S1 est associée à F1 et F3, quand on appelle S1 doFilter de F1 est exécuté, il appelle ensuite doFilter de F3 qui à la fin de son exécution rend le contrôle à F1




Exercices

- Ecrire la méthode doFilter qui transforme la sortie d'une servlet en utilisant XSLT
- Quel est le bon chaînage pour les filtres CompressionFilter, XSLTFilter et EncodeFilter pour une servlet