Libery
LAB

ESIEE - 5I-IN9 Développement web

Cours 2 - Filtre, servlet et JSP

(Version Java 21)

Libery
LAB

Les servlets

Cours 2 - Filtre, servlet et JSP

(Version Java 21)

Une servlet c’est quoi ?

- Une classe java offrant un service web
- Utilise le protocole HTTP
- Format d’entré et de sortie flexible

A quoi ça sert ?

- Manipuler des ressources
  - Fichiers, Objets,…
- Mettre à disposition des méthodes
  - Règles des gestions, Méthodes de calcul, Interfaces externe, …

Comment ça marche ?

- Nécessite le package « jakarta.servlet »
- Doit implémenter l'interface « Servlet »
- Ou étendre la classe « HttpServlet »
- Nécessite d'implémenter les méthodes
- Une seule instance de la classe (singleton)

Comment ça marche ?

- Méthodes de servlet

  - Méthode GET
    - Généralement utilisée pour la consultation d'une ou plusieurs ressources de même type
    - Permet uniquement le paramétrage de l'url
    - Permet de récupérer des entêtes et un contenu
  - Méthode POST
    - Généralement utilisée pour la création d'une ressource
    - Permet le paramétrage par url
    - Permet le paramétrage par du corps des messages
    - Pas de limitation sur la taille du corps des messages
    - Permet de récupérer des entêtes et un contenu
  - Méthode PUT
    - Généralement utilisée pour la mise à jour d'une ressource
    - Comportement identique au POST
  - Méthode DELETE
    - Généralement utilisée pour la suppression d'une ressource
    - Comportement identique au GET
  - Autres méthodes
    - OPTION, HEAD, TRACE

Comment ça marche ?

- Première implémentation

```java
public class EmptyServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        super.doGet(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        super.doPost(req, resp);
    }

}
```

Comment ça marche ?

- Une autre implémentation

```
public class EmptyServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    @Override
    protected void service(HttpServletRequest arg0, HttpServletResponse arg1)
            throws ServletException, IOException {
        super.service(arg0, arg1);
    }

}
```

Comment ça marche ?

- L'implémentation ne suffit pas, il faut l'activer
  - Utilisation de l'annotation « *@WebServlet* »
  - Configuration de l'annotation avec son url

```
@WebServlet("/esiee/api/hello")
```

Comment ça marche ?

- Première implémentation complète

```java
@WebServlet("/esiee/api/hello")
public class EmptyServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    @Override
    protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String message = "hello";
        ServletOutputStream servletOutputStream = response.getOutputStream();
        response.setContentType("text/html");
        servletOutputStream.print(message);
    }

}
```

Résultat : Ecrit « hello » dans le contexte de sortie

Comment ça marche ?

- Paramètres d'une servlet
  - HttpServletRequest
    - Permet d'accéder à la session, aux paramètres url, au corps de la méthode
  - HttpServletResponse
    - Permet de construire la réponse de la requête

Comment ça marche ?

- Récupération des paramètres de requêtes (url)
  - request.getParameter(<Nom du paramètre>)
- Récupération du corps de la requête
  - request.getReader()
- Récupération du contexte de session
  - request.getSession()

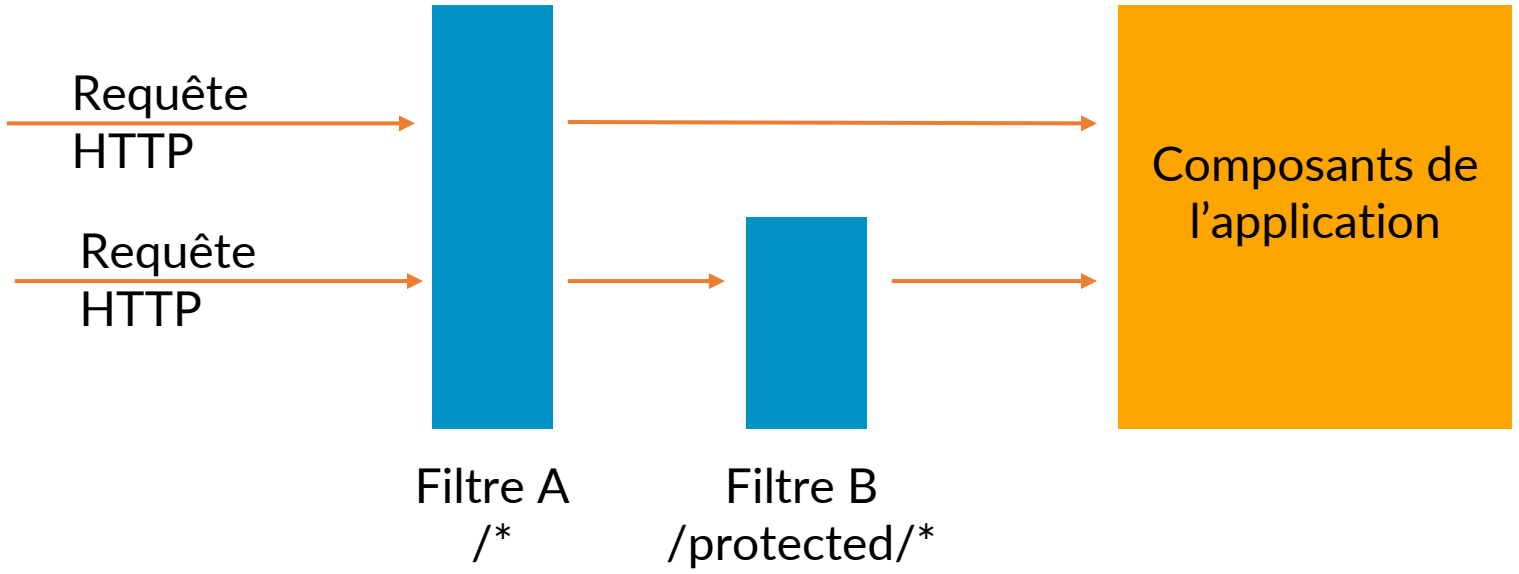
Libery
LAB

Les filtres

Cours 2 - Filtre, servlet et JSP

Un filtre c’est quoi ?

- Une classe java interceptant les requêtes faites sur le serveur



A quoi ça sert ?

- A mettre en place des traitements génériques commun à plusieurs types de requêtes
    - Forcer l'encodage en UTF8
    - Vérification des droits utilisateurs
    - Compteur de nombre de connexion
    - …

Comment ça marche ?

- Nécessite le package « jakarta.servlet »
- Doit implémenter l'interface « Filter »
- Doit implémenter les méthodes définit par l'interface
- Une seule instance de la classe (singleton)

Comment ça marche ?

- Première implémentation

```
public class EmptyFilter implements Filter{

    @Override
    public void destroy() {
        // TODO Auto-generated method stub
    }

    @Override
    public void doFilter(ServletRequest arg0, ServletResponse arg1, FilterChain arg2)
            throws IOException, ServletException {
        // TODO Auto-generated method stub
    }

    @Override
    public void init(FilterConfig arg0) throws ServletException {
        // TODO Auto-generated method stub
    }

}
```

Comment ça marche ?

- L'implémentation ne suffit pas, il faut l'activer

  - Utilisation de l'annotation « **@WebFilter** »
  - Configuration du périmètre url avec « **urlPatterns** »
  - Configuration d'une description « **description** »

```
@WebFilter(urlPatterns={"/*"}, description="Filtre UTF8")
```

Comment ça marche ?

- Cela ne suffit toujours pas, il faut libérer les flux
  - On utilise le paramètre « **FilterChain** » de la méthode « **doFilter** »

```
filterChain.doFilter(servletRequest, servletResponse);
```

Comment ça marche ?

- Première implémentation complète

```java
@WebFilter(urlPatterns={"/*"}, description="Filtre UTF8")
public class EmptyFilter implements Filter{

    @Override
    public void destroy() {
    }

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse,
            FilterChain filterChain) throws IOException, ServletException {
        System.out.println("Utilisation servlet !");
        filterChain.doFilter(servletRequest, servletResponse);
    }

    @Override
    public void init(FilterConfig arg0) throws ServletException {
    }

}
```

Ordonner les filtres

- Ordonnancement impossible avec les annotations
- Utilisation du fichier « **web.xml** » pour ordonner

```
<filter-mapping>
    <filter-name>filter1</filter-name>
    <url-pattern />
</filter-mapping>
<filter-mapping>
    <filter-name>filter2</filter-name>
    <url-pattern />
</filter-mapping>
```

Lorsque c'est possible, il est préférable d'utiliser un filtre unique et de déporter les traitements

Quelques méthodes utiles

- HttpServletRequest
  - getRequestURL()
  - getRequestURI()
  - getQueryString()
  - getServletContext().getContextPath()

```
url = http://localhost:8080/tp2/api/monService
uri = /tp2/api/monService
queryString = parameter1=value1
contextPath = /tp2
```

Url de test : « http://localhost:8080/tp2/api/monService?parameter1=value1 »

Libery LAB

Libery
LAB

Les pages JSP

Cours 2 - Filtre, servlet et JSP

Un tag JSP c’est quoi ?

- Fichier avec l’extension « .jsp »
- Une page HTML enrichie avec du Java
- Une page HTML contenant des tags JSP
- Une page HTML construite dynamiquement côté serveur

A quoi ça sert ?

- Obtention de pages HTML adaptées aux périmètres de chaque client
- Meilleur contenu du serveur sur les données de sorties
- Permet l'utilisation des outils java
- Permet d'alléger l'écriture des pages webs

Libery LAB

Comment ça marche ?

- Lecture et transcription de la page par le conteneur web
- Envoi de la page HTML au client

DTD : Document Type Definition
XSD : XML Schema Description

Libery LAB

Comment ça marche ?

- Première page jsp

```
<html>
  <body>
    <h2>Hello World!</h2>
  </body>
</html>
```

Comment ça marche ?

- Page JSP avec intégration de code java

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<html>
  <body>
    <% request.setAttribute("userName", "François"); %>
    <h2>Hello <%= request.getAttribute("userName") %></h2>
  </body>
</html>
```

```html
<html>
  <body>

    <h2>Hello François</h2>
  </body>
</html>
```

Allez plus loin avec les tags

- Un tag JSP c'est quoi ?
  - Une balise XML similaire à une balise HTML mais lié avec une classe java
  - Converti en code HTML par le conteneur pour être compréhensible par les navigateurs
  - Référencé dans une librairie de tags

```
<html>
   <head>
      <np:setMode inSession="consultation" />
      <np:Redirecting urlRedirect="/rid/personSearch.jsp" addContextPath="true" />
   </head>
</html>
```

Allez plus loin avec les tags

- A quoi ça sert un tag ?
  - Permet l'enrichissement d'application web
  - Permet de généraliser des comportements redondants
  - Permet de simplifier l'écriture des pages JSP

Allez plus loin avec les tags

- Comment ça fonctionne
  - Un tag JSP est une classe Java
  - Nécessite d'implémenter plusieurs méthodes
  - Nécessite d'être déclaré dans une librairie
  - Nécessite d'intégrer la librairie dans les pages

```
<tag>
    <name>title</name>
    <tagclass>com.esiee.tp2.tag.Title</tagclass>
    <bodycontent>JSP</bodycontent>
    <info>Affiche la valeur contenue dans la balise sous forme de titre</info>
</tag>
```

```
public class Title extends TagSupport {

    private static final long serialVersionUID = 1L;

    @Override
    public int doStartTag() throws JspException {

        try {
            pageContext.getOut().println ("<h1>");
        } catch (IOException e) {
            e.printStackTrace();
        }
        return Tag.EVAL_BODY_INCLUDE;
    }

    @Override
    public int doEndTag() throws JspException {

        try {
            pageContext.getOut().println ("</h1>");
        } catch (IOException e) {
            e.printStackTrace();
        }
        return Tag.EVAL_PAGE;
    }

}
```

Allez plus loin avec les tags

- Utilisation du tag « title » dans une page jsp
  - Déclaration de l'utilisation de la librairie

```
<%@ taglib uri="/WEB-INF/esiee.tld" prefix="esiee" %>
```

  - Utilisation du tag

```
<esiee:title>Page d'authentification</esiee:title>
```

Page d'authentification

Il existe des tags permettant de faire des itérations

Allez plus loin avec les tags

- La librairie JSTL

  - JSTL : Java server page Standard Tag Library
  - C'est une librairie simplifiant l'intégration de comportement java
  - Ce sont généralement des tags JSP proposé par JEE
  - Permet de clarifier les traitements dans les pages JSP

Allez plus loin avec les tags

- Intégration de la librairie dans la page JSP

```
<%@ taglib uri="jakarta.tags.core" prefix="c" %>
```

- Ajout de la dépendance dans le pom,xml

```
<dependency>
  <groupId>jakarta.servlet.jsp.jstl</groupId>
  <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
  <version>3.0.2</version>
</dependency>

<dependency>
  <groupId>org.glassfish.web</groupId>
  <artifactId>jakarta.servlet.jsp.jstl</artifactId>
  <version>3.0.1</version>
</dependency>
```

- Utilisation des tags en utilisant le prefix « c »

```
<c:if test="${nfFieldsList == null}">
   <c:set var="nfFieldsList" value="${resultColumn.getValueString('field')}" />
</c:if>
```

Libery LAB

Allez plus loin avec les tags

- Quelques tags pratiques de tag JSTL

  - **if** : Permet de réaliser des conditions
  - **forEach** : Permet de réaliser des itérations
  - **set** : Permet d'ajouter une variable dans le contexte
  - **out** : permet d'écrire dans le flux de la page

Libery Lab

Apprendre et partager - La passion du développement