JEE : cookies, sessions et filtres

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

Plan

JEE

Objectif

- Savoir stocker et récupérer des données quel que soit l'emplacement : Servlet, JSP...
- Sécuriser l'accès à notre application Web en installant des filtres sur les Servlets

JEE

Les cookies

- un cookie est un fichier que l'on enregistre sur l'ordinateur du visiteur pour une longue durée.
- il permet généralement de sauvegarder des données sur le visiteur qui nous permet de l'identifier lors de sa prochaine visite

Pour créer un cookie

```
Cookie cookie = new Cookie("nom", "wick");
```

JEE

Pour créer un cookie

```
Cookie cookie = new Cookie("nom", "wick");
```

Pour stocker un cookie

```
response.addCookie(cookie);
```

JEE

Pour créer un cookie

```
Cookie cookie = new Cookie("nom", "wick");
```

Pour stocker un cookie

```
response.addCookie(cookie);
```

Modifier la durée d'un cookie (par défaut la durée d'une session)

```
cookie.setMaxAge(60*60*24);
```

JEE

Pour rendre le cookie inaccessible en JavaScript (réduire les risques de faille XSS)

```
cookie.setHttpOnly(true);
```

JEE

Pour rendre le cookie inaccessible en JavaScript (réduire les risques de faille XSS)

```
cookie.setHttpOnly(true);
```

Pour récupérer des données stockées dans des cookies

```
Cookie cookies [] = request.getCookies();
```

JEE

Pour rendre le cookie inaccessible en JavaScript (réduire les risques de faille XSS)

```
cookie.setHttpOnly(true);
```

Pour récupérer des données stockées dans des cookies

```
Cookie cookies [] = request.getCookies();
```

Pour chercher un cookie

```
for(int i= 0 ; i < cookies.length; i++) {
    if (cookies[i].getName().equals("nom")) {
        ...
    }
}
```

Pour récupérer le nom d'un cookie

```
cookie.getName();
```

JEE

Pour récupérer le nom d'un cookie

```
cookie.getName();
```

Pour récupérer la valeur d'un cookie

```
cookie.getValue();
```

JEE

Les sessions

- un moyen de conserver des données relatives à un visiteur sur toutes les pages de notre site
- ces données seront enregistrées sur le serveur
- pour chaque session, on peut associer une infinité de variables de session de tout type (primitif, objet, tableau d'objets...)
- ces variables seront détruites, si le visiteur se déconnecte, s'il dépasse une certaine durée...

JEE

Considérons l'objet de type `Personne` suivant

```
Personne personne = new Personne (100, "wick", "john");
```

JEE

Considérons l'objet de type `Personne` suivant

```
Personne personne = new Personne (100, "wick", "john");
```

Pour créer une variable session

```
HttpSession session = request.getSession();
```

JEE

Considérons l'objet de type `Personne` suivant

```
Personne personne = new Personne (100, "wick", "john");
```

Pour créer une variable session

```
HttpSession session = request.getSession();
```

Pour ajouter l'objet `personne` dans la session

```
session.setAttribute("perso", personne);
```

JEE

Considérons l'objet de type `Personne` suivant

```
Personne personne = new Personne (100, "wick", "john");
```

Pour créer une variable session

```
HttpSession session = request.getSession();
```

Pour ajouter l'objet `personne` dans la session

```
session.setAttribute("perso", personne);
```

Pour récupérer une donnée de session (dans une Servlet)

```
session.getAttribute("perso");
```

JEE

Pour récupérer une donnée de session (dans une JSP)

```
<%
  Personne p = (Personne) session.getAttribute("perso");
  out.print(p);
%>
```

JEE

Pour récupérer une donnée de session (dans une JSP)

```
<%
  Personne p = (Personne) session.getAttribute("perso");
  out.print(p);
%>
```

Pour récupérer une donnée de session (dans une JSP avec EL)

```
${ sessionScope.perso }
```

JEE

Pour récupérer une donnée de session (dans une JSP)

```
<%
  Personne p = (Personne) session.getAttribute("perso");
  out.print(p);
%>
```

Pour récupérer une donnée de session (dans une JSP avec EL)

```
${ sessionScope.perso }
```

Pour détruire une session

```
session.invalidate();
```

Exercice

1. Créer un nouveau projet **Java EE**.
2. Créer trois Servlets `FirstServlet`, `SecondServlet` et `ThirdServlet` accessibles respectivement via les chemins `first`, `second` et `third`.
3. Créer une vue `index.jsp` contenant un formulaire d'authentification (nom d'utilisateur et mot de passe) et placer la directement dans `webapp`.
4. À la soumission du formulaire, les données seront envoyées au `doPost()` de `FirstServlet`.
5. Pour accéder à n'importe quelle page de l'application (hormis `index.jsp`), l'utilisateur doit s'authentifier.
6. S'il réussit à s'authentifier, il sera enregistré dans la session.
7. Créer trois vues `first.jsp`, `second.jsp` et `third.jsp` dans `WEB-INF` appelées respectivement par `FirstServlet`, `SecondServlet` et `ThirdServlet`.
8. Dans chaque vue (`first.jsp`, `second.jsp` et `third.jsp`), on affiche le nom de la vue, le nom de l'utilisateur connecté, deux liens pour naviguer vers les autres vues et un lien de déconnexion.

JEE

Remarques

Le code qui permet de vérifier si le nom d'une personne est dans la session + redirection est répétitif : dans toutes les Servlets

JEE

Remarques

Le code qui permet de vérifier si le nom d'une personne est dans la session + redirection est répétitif : dans toutes les Servlets

Solution

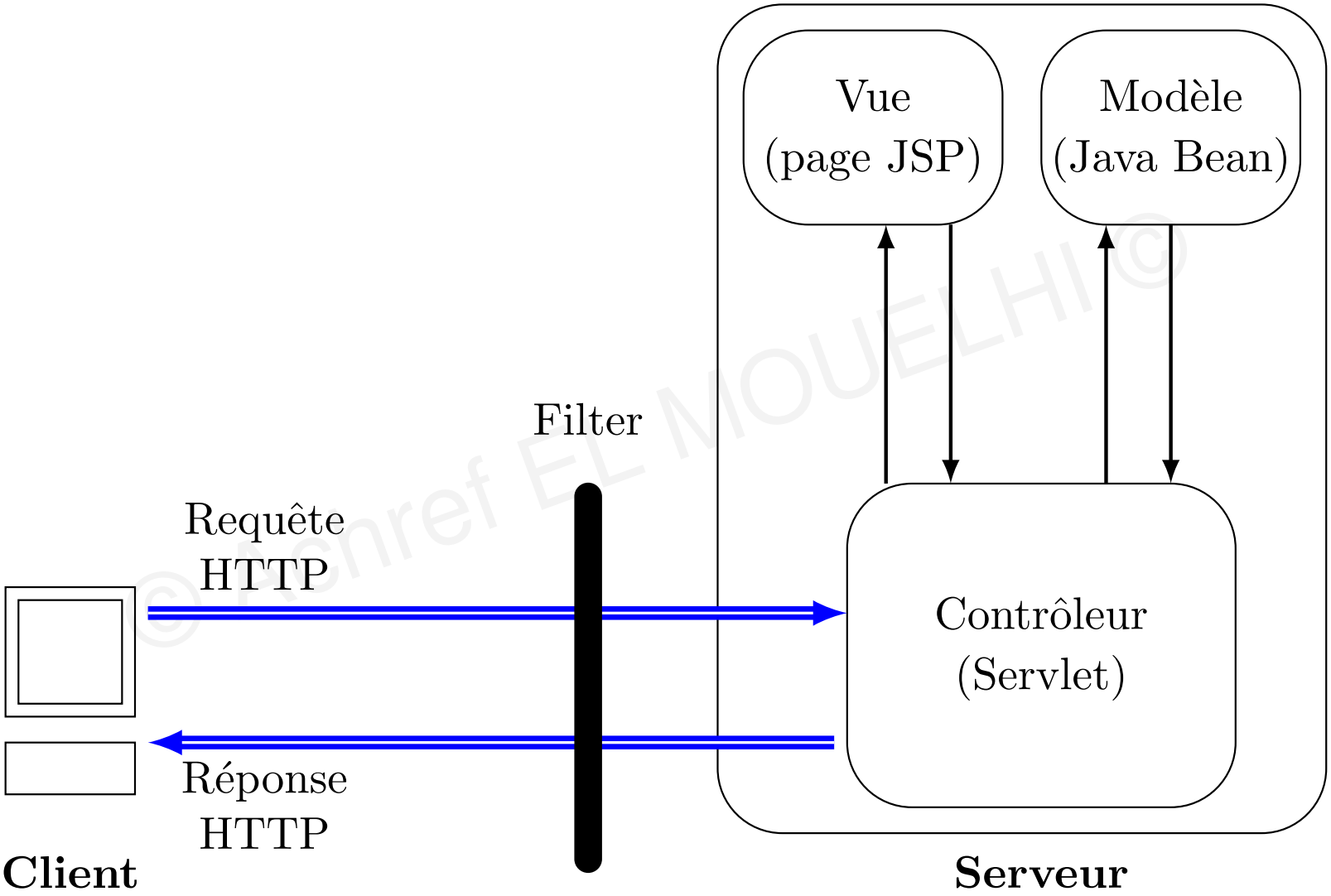
Utiliser les filtres pour tester avant d'arriver dans la Servlet

JEE

Filtre

- Classe `Java` implémentant l'interface `Filter` ayant les méthodes `init()`, `destroy()`, `doFilter()`
- Défini par rapport à une URL (soit dans le fichier `web.xml`, soit avec l'annotation `@WebFilter()`
- s'exécute avant la Servlet ayant la même URL
- Un filtre avec l'URL `/*` se lance avant toutes les Servlets de l'application
- Une fois l'exécution du filtre est terminé, il passe la main à un autre, avec la méthode `chain.doFilter()` (s'il en existe), sinon c'est la Servlet qui sera exécutée.

JEE



JEE

Création d'un filtre sous **Eclipse**

- Faire un clic droit sur `src` situé dans `Java Resources` de notre projet
- Aller dans `New` et choisir `Filter`
- Remplir le champ `Java package:` par `org.eclipse.filter`
- Remplir le champ `Class name:` par un nom suffixé par le mot `Filter` : PrivateFilter (par example)
- Cliquer sur `Next`

Code généré

```
@WebFilter("/PrivateFilter")
public class PrivateFilter implements Filter {

  public PrivateFilter() {
    // TODO Auto-generated constructor stub
  }
  public void destroy() {
    // TODO Auto-generated method stub
  }

   public void doFilter(ServletRequest request, ServletResponse
      response,    FilterChain chain) throws IOException,
      ServletException {
    // TODO Auto-generated method stub
    // place your code here

    // pass the request along the filter chain
    chain.doFilter(request, response);
  }
  public void init(FilterConfig fConfig) throws ServletException {
    // TODO Auto-generated method stub
  }
}
```

JEE

Les filtres

- Supprimer toutes les méthodes de filtre et garder seulement `doFilter()`
- Implémenter cette dernière pour qu'elle vérifie si un nom est dans la session
- Filtre doit être lancé avant l'exécution de toutes les URL sauf celles qui affichent et vérifient le formulaire d'authentification

JEE

Contenu d'`index.jsp`

```
<%@ page language="java" contentType="text/html; charset=ISO
   -8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
  <title>Home page</title>
</head>
<body>
  <h1>Home page</h1>
  <form action=first method=post>
    <input type=text name=nom placeholder="saisir nom"><br>
    <input type=password name=pwd placeholder="saisir mot de
       passe"><br>
    <button type="submit"> Connection </button>
  </form>
</body>
</html>
```

Contenu de `FirstServlet`

```
@WebServlet("/first")
public class FirstServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;
  protected void doGet(HttpServletRequest request, HttpServletResponse
     response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    getServletContext().getRequestDispatcher("/WEB-INF/first.jsp").
       forward(request, response);
  }
  protected void doPost(HttpServletRequest request, HttpServletResponse
     response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    String nom = request.getParameter("nom");
    String pwd = request.getParameter("pwd");
    if (nom.equals("wick") && pwd.equals("john")) {
      request.getSession().setAttribute("nom", nom);
      getServletContext().getRequestDispatcher("/WEB-INF/first.jsp").
         forward(request, response);
    }
    else
      getServletContext().getRequestDispatcher("/index.jsp").forward(
         request, response);
  }
}
```

JEE

Contenu de `first.jsp` (même logique pour `second.jsp` et `third.jsp`)

```
<!DOCTYPE html>
<html>
<head>
  <title>First page</title>
</head>
<body>
  <h2>First page</h2>
  <c:out value="Bonjour ${ sessionScope.nom }"/>
  <div>
    <c:url value="second" var="second" />
    <a href="${ second }"> second </a>
  </div>
  <div>
    <c:url value="third" var="third" />
    <a href="${ third }"> third </a>
  </div>
  <div>
    <c:url value="deconnect" var="deconnect" />
    <a href="${ deconnect }"> déconnexion </a>
  </div>
</body>
</html>
```

JEE

Contenu de `SecondServlet` (même logique pour `ThirdServlet`)

```
@WebServlet("/second")
public class SecondServlet extends HttpServlet {

  private static final long serialVersionUID = 1L;

  protected void doGet(HttpServletRequest request, HttpServletResponse
     response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    getServletContext().getRequestDispatcher("/WEB-INF/second.jsp").
       forward(request, response);
  }

  protected void doPost(HttpServletRequest request, HttpServletResponse
      response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    doGet(request, response);
  }
}
```

JEE

Contenu de `PrivateFilter`

```
@WebFilter("/*")
public class PrivateFilter implements Filter {

  public void doFilter(ServletRequest request, ServletResponse response,   FilterChain chain)
     throws IOException, ServletException {

    HttpServletRequest req = (HttpServletRequest) request;
    HttpServletResponse res = (HttpServletResponse) response;
    HttpSession session = req.getSession();

    // on récupère le nom de la session
    String nom = (String)session.getAttribute("nom");

    // on récupère le chemin demandé par l'utilisateur
    String chemin = req.getServletPath();

    // on récupère la méthode HTTP utilisée (GET ou POST)
    String methode = req.getMethod();

    if (nom != null || chemin.equals("/") || chemin.equals("/index.jsp") || chemin.equals("/
       first") && methode.equals("POST"))
      chain.doFilter(request, response);
    else
      res.sendRedirect(req.getContextPath());
  }
}
```

JEE

On peut aussi supprimer l'annotation `@WebFilter` et utiliser le `web.xml` comme pour les Servlets

```
<filter>
  <filter-name>PrivateFilter</filter-name>
  <filter-class>orh.eclipse.filter.PrivateFilter</
     filter-class>
</filter>

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

JEE

Remarques

- Si on a plusieurs filtres sur une même URL, l'ordre de vérification est l'ordre de déclaration de la section `filter-mapping` dans `web.xml`.
- Si on a plusieurs filtres sur une même URL, et qu'on utilise l'annotation `@WebFilter`, il faut quand-même préciser l'ordre d'exécution dans `web.xml` en ajoutant une section `filter-mapping` pour chaque filtre.