133 - Réaliser des applications Web en Session-Handling

Rapport personnel

Date de création : 23.03.2023
Version 1 du 23.03.2023

Nathan Clapasson

Ecole des Métiers
Berufsfachschule
Fribourg
Informatique

Module
du 23.03.2023 au
05.05.2023

Table des matières

1 Introduction

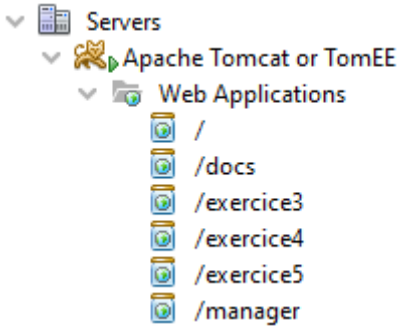
1.1 Objectifs du module

- Connaître des solutions possibles afin de délimiter les fonctionnalités côté client comme serveur en se basant sur les directives.
- Connaître les éléments d'un concept de réalisation pour des applications Web.
- Connaître les avantages de la séparation entre la présentation et la logique du programme ainsi que leur mise en œuvre dans une application Web.
- Connaître des éléments de formulaires ainsi que des fonctions pour vérifier les saisies de l´utilisateur, et comment ceux-ci mis en œuvre conformément aux directives.
- Connaître des techniques courantes pour la réalisation du suivi de session.
- Connaître les risques sur la sécurité et des solutions possibles pour protéger une application Web.
- Connaître des possibilités d'administration de lots de données.
- Connaître des possibilités pour garantir l'authentification et les autorisations dans une application Web.
- Connaître des langages de programmation possibles pour des applications Web.
- Connaître des solutions possibles pour la mise en œuvre d'architectures appropriées pour des applications Web.
- Connaître des possibilités de structuration de code source ainsi que leur mise en œuvre en tenant compte des directives de codification.
- Connaît des procédures de tests, et leur contribution, pour garantir la qualité d'applications Web.
- Connaître des procédures de tests permettant de vérifier les solutions possibles contre les Cross-Site-Scripting, Script-Injection et Session-Hijacking.

2 Tests technologiques

2.1 Exercice 1

Durant cet exercice, nous avons appris à mettre en place Tomcat en local et à le connecter à NetBeans. Pour ajouter un serveur Tomcat dans NetBeans, il faut aller dans les services et ajouter notre serveur en temps que serveur Apache Tomcat. Après cela, le serveur devrait être fonctionnel avec nos applications Java.



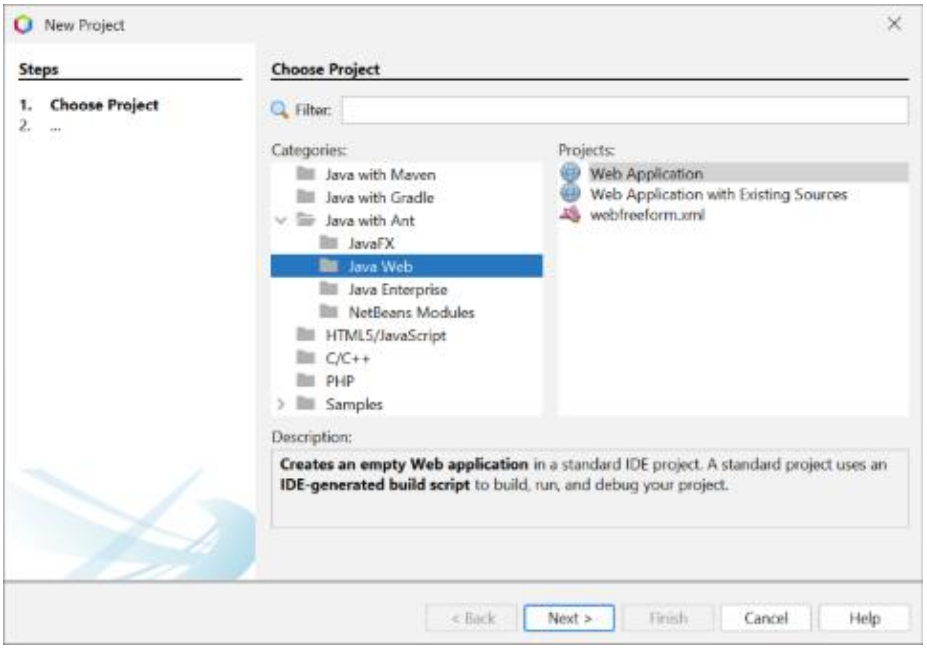
En cas de problème pour arrêter Tomcat, il faut chercher le processus qui pose un problème et le terminer dans l'invite de commande.

```
netstat -ano | findstr :8080

taskkill /PID <ID> /F
```

2.2 Exercice 2

Dans le cadre de ces prochains exercices, nous allons créer des projets sous forme de Web Application avec un dossier dédié pour les librairies et aucun Framework.



Lorsque nous allons sur http://localhost:8080/manager/html, nous pouvons voir la liste des applications tournant actuellement sur le serveur.

Les pages Jakarta Serveur (JSP) sont des pages crées avec Java permettant de générer des pages HTML de façon dynamique. Nous allons régulièrement en utiliser au cours de ce module.

2.3 Exercice 3

Dans cet exercice, nous allons créer une page de login avec des variables. Dans notre index.html, nous allons créer un form avec un user et un mot de passe avec un bouton pour lancer le fichier checkLogin.jsp.

```
<label for="user">Votre nom d'utilisateur :</label>

<input type="text" name="username" id="user" size="50"
maxlength="50"/><br>

<label for="pass">Votre mot de passe :</label>

<input type="password" name="password" id="pass" size="50"
maxlength="50"/><br>

<input type="submit" value="Soumettre"/>
```

Dans le checkLogin.jsp, nous allons récupérer les paramètres et utiliser les variables pour les afficher.

```
<%
    String username = request.getParameter("username");
    String password = request.getParameter("password");
%>
<p>Votre nom d'utilisateur est : <%=username%><br>
   Votre mot de passe est      : <%=password%><br>
</p>
```

2.4 Exercice 4

Dans cet exercice, nous allons reprendre la forme de l'exercice précédant en modifiant afin de récupérer une image dans une API en fonction du paramètre entré.

```
<%
     String selection = request.getParameter("selection");
%>
<h1>Voici la météo de <%=selection%></h1>
<p>
    <a href="https://www.prevision-
meteo.ch/meteo/localite/<%=selection%>"><img src="https://www.prevision-
meteo.ch/uploads/widget/<%=selection%>_0.png" width="650" height="250"
/></a>
</p>
```

2.5 Exercice 5

Dans cet exercice, nous allons récupérer des données dans une base de données à l'aide de JDBC puis les afficher de deux manière : avec out.println et avec des variables.

```
<%
    out.println("Solution avec instruction java out.println()");
    for (String pays : lstPays) {
        out.println(pays + "<br>");
    }
    out.println("<br>");
%>
<div>Solution avec intégration de variables java dans HTML</div>
<%
    for (String pays : lstPays) {
%>
<%=pays%><br>
<%
        }
    }
%>
```

2.6 Exercice 6

Dans cet exercice, l'objectif est de se servir de beans afin de vérifier si un utilisateur peut se connecter. Pour pouvoir importer des classes dans un fichier jsp, il faut utiliser la balise @page.

```
<%@page import="beans.BeanInfo"%>
<%@page import="beans.BeanError"%>
```

Pour utiliser des instances d'une classe, il faut utiliser la balise jsp:useBean.

```
<jsp:useBean id="beanInfo" scope="session" class="beans.BeanInfo"/>
<jsp:useBean id="beanError" scope="session" class="beans.BeanError"/>
```

Le champ scope désigne la portée à laquelle une classe peut être appelée :

- Page : accessible sur l'ensemble de la page JSP (valeur de base)
- Request : accessible pour la durée d'une requête sur l'ensemble de la requête
- Session : accessible pour la durée de la session sur l'ensemble des JSP de la session
- Application : accessible sur l'ensemble des JSP de l'application

Pour appeler une autre page JSP, il faut utiliser la balise jsp :forward.

```
<jsp:forward page="maJspLogge.jsp"></jsp:forward>
```

2.7 Exercice 7

Dans cet exercice, l'objectif est d'apprendre à utiliser des servlets à savoir des classes java permettant de générer des données au format HTML ou XML sur un serveur http.

Lors de la création de notre servlet, deux fichiers sont créés en plus de notre servlet à savoir context.xml qui indique le chemin de notre servlet et web.xml qui permet de modifier les paramètres web de notre servlet (nom, pattern de l'url…).

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    <servlet>
        <servlet-name>MaServlet</servlet-name>
        <servlet-class>workers.MaServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>MaServlet</servlet-name>
        <url-pattern>/MaServlet</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
</web-app>
```

2.8 Exercice 8

Dans cet exercice, l'objectif était d'utiliser un servlet de connexion qui va rediriger l'utilisateur sur deux pages JSP en fonction du résultat.

```
protected void processRequest(HttpServletRequest request,
HttpServletResponse response)

        throws ServletException, IOException {

    response.setContentType("text/html;charset=UTF-8");

    try ( PrintWriter out = response.getWriter()) {

        HttpSession session = request.getSession();

        session.setMaxInactiveInterval(20);

        BeanInfo info = new BeanInfo();

        String username = request.getParameter("username");

        String password = request.getParameter("password");

        if(username.equals(info.getNom()) && password.equals("Emf12345")
&& info.isAccess() == true){

            session.setAttribute("info", info);

            RequestDispatcher dispatch =
request.getRequestDispatcher("pageAutorise.jsp");

            dispatch.forward(request, response);

        }else{

            RequestDispatcher dispatch =
request.getRequestDispatcher("erreur.jsp");

            dispatch.forward(request, response);

        }

    }

}
```

Dans notre servlet, nous allons récupérer la session http et vérifier si les données récupérées en paramètre sont identiques aux données entrées en dur dans le bean BeanInfo. Si c'est le cas, on lance la page autorisée.

2.9 Exercice 10

Dans cet exercice, nous devons créer un serveur et une application REST. Pour le serveur, nous devons créer une application web avec une classe RESTful avec pattern ce qui va nous donner deux classes java : une classe ApplicationConfig et notre application. L'Application-Config nous sert principalement à indiquer le chemin.

```
@GET
@Path("getMessage")
@Produces(javax.ws.rs.core.MediaType.TEXT_PLAIN)
public String getMessage(){
    return "Bonjour tout le monde !";
}

@GET
@Path("getAuthor")
@Produces(javax.ws.rs.core.MediaType.TEXT_PLAIN)
public String getAuthor(){
    return "Nathan Clapasson";
}
```

Dans notre application, nous pouvons à présent créer des requêtes HTML en indiquant le type de requête, le chemin de notre requête et le type de données obtenue en retour.

Constante	String	Description
APPLICATION_ATOM_XML	"application/atom+xml"	XML+ATOM
APPLICATION_FORM_URLENCODED	"application/x-www-form-urlencoded"	Formulaire via URL encodé.
APPLICATION_JSON	"application/json"	Fichier JSON
APPLICATION_OCTET_STREAM	"application/octet-stream"	Fichier de BYTE
APPLICATION_SVG_XML	"application/svg+xml"	Fichier vectoriel
APPLICATION_XHTML_XML	"application/xhtml+xml"	XML dans XHTML
APPLICATION_XML	"application/xml"	Fichier XML
CHARSET_PARAMETER	-	Encodage
MEDIA_TYPE_WILDCARD	"*"	Champ libre, non spécifié
MULTIPART_FORM_DATA	"multipart/form-data"	Formulaire contenant un fichier externe par exemple.
TEXT_HTML	"text/html"	Texte au format HTML
TEXT_PLAIN	"text/plain"	Texte au format brut
WILDCARD	"*/*"	Non spécifié

Pour l'application REST, nous allons créer une application web avec une classe de type REST-ful java client. Lors de sa création, nous avons la possibilité de le générer automatiquement en indiquant le serveur.

Pour finir, nous allons créer un servlet qui va nous permettre de faire des requêtes sur notre serveur.

```
protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try ( PrintWriter out = response.getWriter()) {


        if(request.getParameter("getMessage") != null){
            String reponse = client.getMessage();
            out.print(reponse);
        }


        else if(request.getParameter("getAuthor") != null){
            String reponse = client.getAuthor();
            out.print(reponse);
        }
        else{
            out.print("Quelque chose ne s'est pas bien passé !");
        }
    }
}
```