|  |  |  |
| --- | --- | --- |
| S:\EMF\CPROF\Identite_EMF\2015-Identites\Logos des sections COULEURS\Logo_EMF-Informatique_FR_RVB_25.jpg |  | Module du 23.03.2023 au 23.06.2023 |

Réaliser des applications web en session-handling

Rapport personnel

Date de création : 23.03.2023  
Version finale du 23.06.2023

Table des matières

[1 Introduction 1](#_Toc138426166)

[2 Tests technos 1](#_Toc138426167)

[2.1 TT01 - Installation infrastructure : Tomcat 1](#_Toc138426168)

[2.1.1 Tomcat 1](#_Toc138426169)

[2.1.1.1 Installation service 1](#_Toc138426170)

[2.1.1.2 Exécuter le service 2](#_Toc138426171)

[2.1.1.3 Résultat 2](#_Toc138426172)

[2.1.1.4 Config Netbeans 3](#_Toc138426173)

[2.1.2 Tizoo Configuration Tomcat 4](#_Toc138426174)

[2.1.2.1 Modification du fichier de configuration server.xml 4](#_Toc138426175)

[2.1.2.2 Redémarrer Tomcat 5](#_Toc138426176)

[2.1.2.3 Mise à jour de l’adresse email 6](#_Toc138426177)

[2.1.3 Tizoo Déploiement d'un .war 6](#_Toc138426178)

[2.1.3.1 Préparation d’hébergement 6](#_Toc138426179)

[2.1.3.2 Déploiement d'un projet sur l'hébergement 6](#_Toc138426180)

[2.1.3.3 Contrôle du déploiement des fichiers de projet 7](#_Toc138426181)

[2.2 TT02 – Netbeans et JSP 8](#_Toc138426182)

[2.2.1 Mettre en place une page Web 8](#_Toc138426183)

[2.2.2 Tomcat manager 10](#_Toc138426184)

[2.2.3 Créer une page jsp 11](#_Toc138426185)

[2.2.4 Hébergement 12](#_Toc138426186)

[2.2.5 Synthèse 13](#_Toc138426187)

[2.3 TT03 – HTML et JSP 13](#_Toc138426188)

[2.3.1 Page login 13](#_Toc138426189)

[2.3.2 Résultat 14](#_Toc138426190)

[2.4 TT04 - Consommer un webservice depuis une page Jsp 14](#_Toc138426191)

[2.4.1 Code important 14](#_Toc138426192)

[2.4.2 Synthèse 15](#_Toc138426193)

[2.5 TT05 - Accéder à une DB par JDBC 15](#_Toc138426194)

[2.5.1 Code important 15](#_Toc138426195)

[2.5.2 Hébergement/ DB sur cpanel 17](#_Toc138426196)

[2.5.3 Synthèse 18](#_Toc138426197)

[2.6 TT06 – Transfert de Bean entre JSP 18](#_Toc138426198)

[2.6.1 Code important 18](#_Toc138426199)

[2.6.1.1 Imports 20](#_Toc138426200)

[2.6.1.2 Instanciation des Beans 20](#_Toc138426201)

[2.6.1.3 Redirection vers des pages 20](#_Toc138426202)

[2.6.2 Synthèse 20](#_Toc138426203)

[2.7 TT07 - Création et utilisation d'une servlet 20](#_Toc138426204)

[2.7.1 Partie 1 20](#_Toc138426205)

[2.7.2 Partie 2 23](#_Toc138426206)

[2.8 TT08 – Servlet, Utilisation de la session 24](#_Toc138426207)

[2.8.1 Code important 24](#_Toc138426208)

[2.8.1.1 GatewayServlet.java 24](#_Toc138426209)

[2.8.1.2 maJspLogge.jsp 24](#_Toc138426210)

[2.8.1.3 erreur.jsp 25](#_Toc138426211)

[2.8.2 Résultat 25](#_Toc138426212)

[2.9 TT10 – Création utilisation de service REST 26](#_Toc138426213)

[2.9.1 Création d’un serveur REST 26](#_Toc138426214)

[2.9.1.1 Création de projet 26](#_Toc138426215)

[2.9.1.2 Configuration du projet 28](#_Toc138426216)

[2.9.1.3 Classe REST 28](#_Toc138426217)

[2.9.1.4 Test avec Postman 30](#_Toc138426218)

[2.9.2 Création d’un client REST 31](#_Toc138426219)

[2.9.2.1 Création du projet 31](#_Toc138426220)

[2.9.2.2 Création de Servlet 32](#_Toc138426221)

[2.9.2.3 Page client 34](#_Toc138426222)

[2.9.2.4 Codage Servlet 34](#_Toc138426223)

[2.9.2.5 Résultat 35](#_Toc138426224)

[3 Synthèse S1 35](#_Toc138426225)

[4 Conclusion 36](#_Toc138426226)

# Introduction

Dans ce module on va faire un projet a deux. On va faire une application de gestion de foot :

Dans notre projet, nous allons faire deux pages différentes. Sur la page de visite, on peut voir les équipes de football, dans quelle ligue elles jouent, un nom d'utilisateur et un mot de passe pour s'inscrire. Une fois connecté, on peut ajouter une équipe, supprimer une équipe ou éditer le nom ou l'abréviation de l'équipe. Sur la page d'administration, on ne peut se connecter qu'avec un seul compte, il n'y a donc qu'un seul champ pour le mot de passe. Une fois ce dernier saisi, on est redirigé vers une page où sont affichés les différents comptes d'utilisateurs, sur laquelle on peut ensuite supprimer les comptes ou en créer un nouveau. Ci-dessous, vous pouvez voir quelques dessins de la manière dont nous avons imaginé le tout :

# Tests technos

## TT01 - Installation infrastructure : Tomcat

### Tomcat

#### Installation service

Installer le programme suivant :

<https://archive.apache.org/dist/tomcat/tomcat-8/v8.5.84/bin/apache-tomcat-8.5.84.exe>

On a installé le programme avec le port 8084. Notre username et un mot de passe.

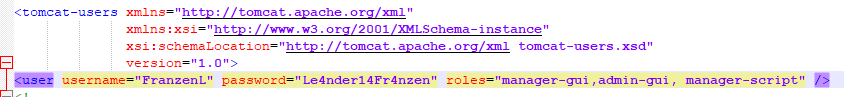
Les API suivantes sont installées :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Dans : C:\Program Files\Apache Software Foundation\Tomcat 8.5\conf\tomcat-users.xml

On a du ajoutez le rôle manager-script



#### Exécuter le service

Maintenant exécuter le startup.bat file depuis cmd :

Ein Bild, das Text, orange enthält.

Automatisch generierte Beschreibung

<https://learn.jamf.com/en-US/bundle/technical-articles/page/Configuring_JAVA_HOME_and_JRE_HOME_Environment_Variables_on_a_Windows_Server.html>

Pour résoudre ce probleme on doit aller dans les options avancées du système :

Ein Bild, das Text, Screenshot, Im Haus enthält.

Automatisch generierte Beschreibung



Restart cmd et relancez la commande :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

#### Résultat

Ein Bild, das Website enthält.

Automatisch generierte Beschreibung

#### Config Netbeans

On va ajouter un serveur dans l’onglet services

Ein Bild, das Text enthält.

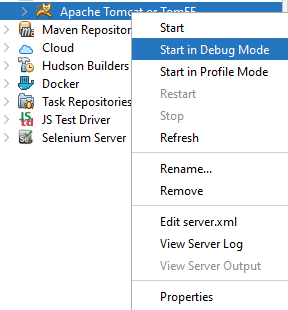
Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

##### Start server

Avec une clique droite on peut lancer le serveur en debug mode.



##### Stop server

Lancez en mode CMD :

netstat -ano | findstr :8084

taskkill /PID 10160 /F

Ein Bild, das Text enthält.

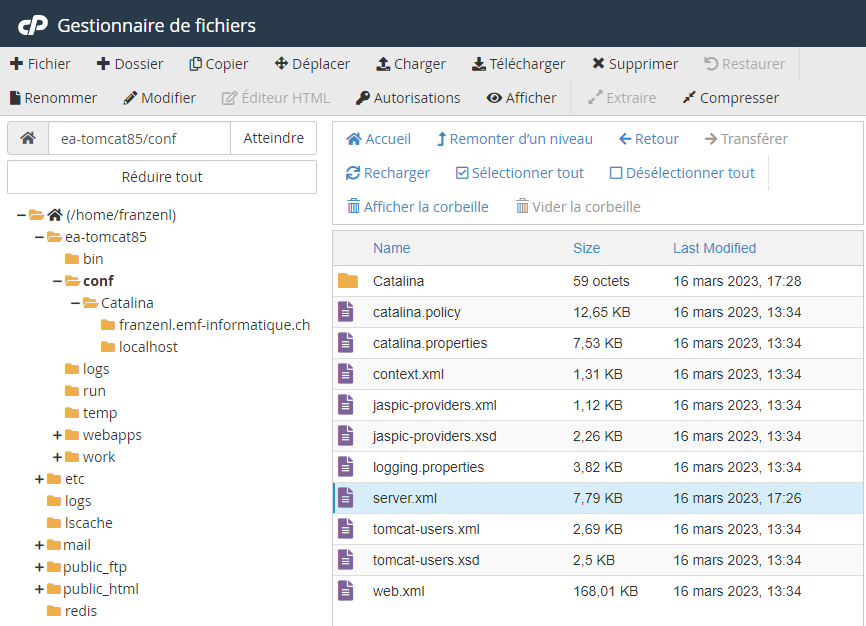
Automatisch generierte Beschreibung



### Tizoo Configuration Tomcat

#### Modification du fichier de configuration server.xml

Dans le gestionnaire de fichiers dans cPanel, on va trouver le dossier conf qui contient le fichier server.xml.



Sur la ligne 145 on va mettre unpackWARs sur true. Et deployXML sur true.





#### Redémarrer Tomcat

Dans l’onglet avancé on peut ouvrir le terminal.

On va taper les commandes suivantes :

export PATH=$(dirname $(readlink /usr/local/cpanel/3rdparty/bin/perl)):$PATH

ubic restart ea-tomcat85

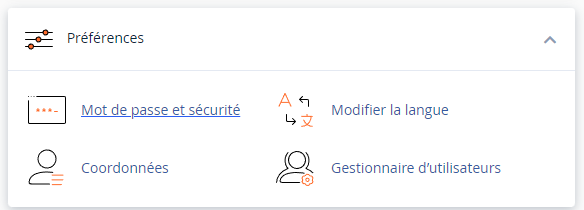
ubic status ea-tomcat85

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

#### Mise à jour de l’adresse email

Dans préférences on voit qu’il y a « Cordonnées » :



Ici on peut mettre l’adresse mail correcte quel correspond à notre.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

### Tizoo Déploiement d'un .war

#### Préparation d’hébergement

On a fait un backup de tout le dossier public\_html :

Et ensuite on a effacé tous qu’on a créé nous-même.



#### Déploiement d'un projet sur l'hébergement

Le build d'un projet java web produit un .war dans le répertoire /dist

Ce monProjet.war doit être renommer **java**\_monProjet.**war** pour être déployé (préfixe = java).

Pour tester, on va déployer un fichier de test dans public\_html :

Ein Bild, das Tisch enthält.

Automatisch generierte BeschreibungEin Bild, das Text enthält.

Automatisch generierte Beschreibung

Pour tester on accède au site :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

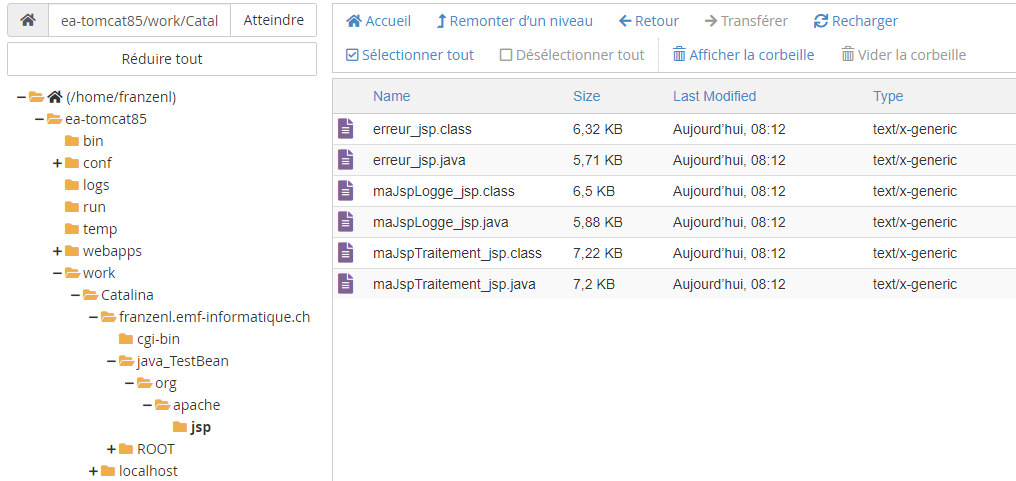
Si vous entrez le nom Charles et mot de passe Emf123, vous obtenez

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

#### Contrôle du déploiement des fichiers de projet

A noter que les fichiers sont visibles à cet emplacement uniquement après le premier appel de votre application.



On voit bien que les fichiers sont visibles, du coup le contrôle s’est passé bien.

## TT02 – Netbeans et JSP

### Mettre en place une page Web

On créer un nouveau projet Web, avec Ant :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Donnez un nom au projet et choisir un emplacement. On utilise le répertoire lib.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Choisir le serveur (proposé par défaut) et Finish.

On ne sélectionne pas de Framework (si Next)

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

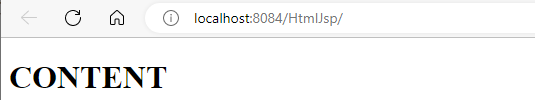
On créer une page index.html

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Après on peut lancer l’application pour qu’on peut l’ouvrir dans le navigateur :

Si on ne peut pas lancer l’application, on doit [kill le serveur](#_Stop_server) avec la commande et le redémarrer avec NetBeans.



### Tomcat manager

Pour accéder au manager Tomcat on doit cliquer ici :

Ein Bild, das Text, Website enthält.

Automatisch generierte Beschreibung

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

### Créer une page jsp

On créer une page jsp dans le même projet :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

On peut maintenant accéder sur le site jsp :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

### Hébergement

Pour héberger sur cpanel, on doit en premier build l’application, pour qu’il créer le dossier dist :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Maintenant on peut uploader ce fichier .war sur cpanel.

On doit renommer le fichier avec le préfixe : java\_



Voilà, le résultat :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

### Synthèse

Comment créer une page web JSP et l'utiliser dans Netbeans. A la fin, si on veut mettre la page sur cpanel, il suffit de faire un « build » du projet et de mettre le fichier .war dans le dossier dist sur cpanel dans le dossier public\_html.

## TT03 – HTML et JSP

### Page login

On créer un nouveau projet Web dans Netbeans. On va compléter le fichier index.html avec le code fournis afin d'obtenir un formulaire avec authentification.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

On créer un nouveau jsp file qui sera appelé du file JSP. checkLogin.jsp

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

### Résultat

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## TT04 - Consommer un webservice depuis une page Jsp

### Code important

Dans ce teste on va utiliser un autre service externe pour afficher les données de météo en fonctionne de quelle ville on tape dans l’input :

Ici on a juste l’input ou on va taper notre ville.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

La page qui va être appelé tant qu’on a appuyé sur le bouton.

Le code important ici c’est de faire l’url avec la ville tapée. Il va chercher l’image sur la site de météo avec la ville demandé.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

### Synthèse

La synthèse pour ce test est qu’on peut utiliser des autres services qu’on peut accéder déçu avec une requête http pour qu’on peut prendre des informations supplémentaires externe. Sinon on devrait développer un programme pour prendre des mesures de météo. Et ça c’est trop lourd.

## TT05 - Accéder à une DB par JDBC

### Code important

Pour afficher les pays tirés depuis la base de données, on a le code suivant :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Connection vers la base de données :

Ein Bild, das Text enthält.

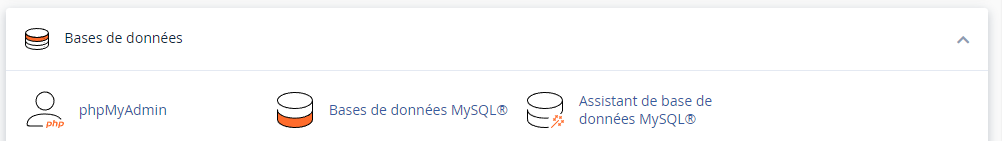
Automatisch generierte Beschreibung

Prise de pays depuis base de données :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

### Hébergement/ DB sur cpanel



Sur bases de données MySQL on doit aller créer une nouvelle DB et un nouveau user. Quand on a créé les deux, on peut mettre l’user sur la base de données qu’on a créé.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Après on peut aller sur phpMyAdmin et on voit qu’il y a la base de données.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

### Synthèse

Dans ce test on a regardé comment faire la connexion sur une base de données, il faut avoir le driver mySql dans la librairie.



## TT06 – Transfert de Bean entre JSP

### Code important

L’index, ou on a la structure du formulaire pour taper le username et le mot de passe :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ici on a le code pour traiter le login. On check si le login est juste et si le login est ok, il va changer la page vers la page loggée. Sinon page erreur.

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Si c’est bon :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Si ce n’est pas bon :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

#### Imports



#### Instanciation des Beans



#### Redirection vers des pages



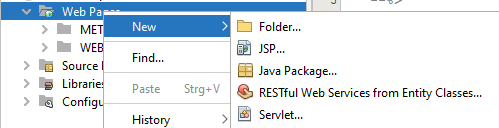
### Synthèse

Pour ce test on a utilisé les beans. Pour les utiliser on doit toujours instancier les beans et les importer. On a bien utilisé les commandes jsp pour différentes choses. Comme la redirection des pages, par exemple quand le login est un succès, il va changer vers le page loggé. Aussi pour utiliser les beans on a besoin la commande jsp:useBean dans le body.

## TT07 - Création et utilisation d'une servlet

### Partie 1

On créer un nouveau servlet :





On la place dans le package workers :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

On coche le case: add information to deployment descriptor

Ein Bild, das Text enthält.

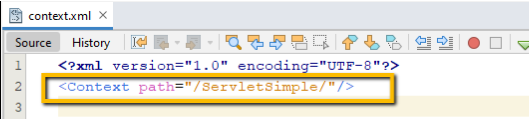
Automatisch generierte Beschreibung

Si on run le file, il l’ouvre, mais il y a une url différente :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

L'url est composé du chemin de context correspondant au nom du projet, stocké dans context.xml, et de l'url-pattern de la servlet défini dans le fichier web.xml



La balise <servlet-mapping> est utilisée pour définir l'url-mapping, c’est-à-dire l’adresse à laquelle il faudra accéder pour voir cette Servlet.

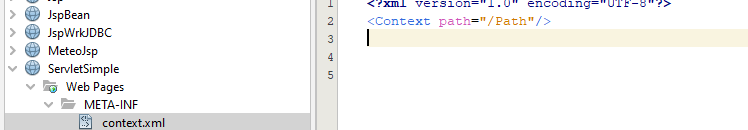
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Pour obtenir l’url suivant pour accéder au fichier servlet on doit appliquer ce code :

localhost:8085/Path/File

Dans le fichier context.xml :



Dans le fichier web.xml :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Résultat :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

### Partie 2

Maintenant j’ai rajouté le index.html qui demande de se loger. Et si on clique sur soumettre, on va vers la page servlet :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Page servlet :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Code dans le fichier index.html :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Code dans le fichier servlet :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## TT08 – Servlet, Utilisation de la session

### Code important

On a de nouveau la page de login. On va vérifier si le login est OK ou pas dans GatewayServlet. Et on va rediriger vers la page correspondante :

#### GatewayServlet.java

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

//SESSION

HttpSession session = request.getSession();

session.setMaxInactiveInterval(20);

//Récupération login

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

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

//check if login est correcte

if (username.equals("user") && pw.equals("pass")) {

//BEANS

BeanInfo info = new BeanInfo();

Adresse adresse = new Adresse("Franzen", "Leander", "Zur Schür 10", 1736, "St. Silvester");

//SET INFO

info.setAdr(adresse);

info.setAccess(true);

info.setNom(username);

//ajouter a la session les infos

session.setAttribute("info", info);

//Forwarder to site maJspLogge.jsp

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

dispatch.forward(request, response);

} else {

//Bean error

BeanError error = new BeanError();

error.setActif(true);

error.setContexte(request.getContextPath());

error.setMessage("Login failed");

error.setRedirection("erreur.jsp");

error.setSource(request.getServletPath());

//ajouter a la session les infos

session.setAttribute("error", error);

//Forwarder to site erreur.jsp

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

dispatch.forward(request, response);

}

}

On a après les deux pages jsp qui sont : maJspLogge.jsp quand le login est OK, et erreur.jsp wuand le login n’est pas OK

#### maJspLogge.jsp

<%--

Document : maJspLogge

Created on : 30.03.2023, 14:03:42

Author : FranzenL

--%>

<%@page import="javax.sound.midi.MidiDevice.Info"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

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

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page Loggé</title>

</head>

<body>

<jsp:useBean id="info" scope="session" class ="beans.BeanInfo"/>

<h1>Vous êtes loggé, voici les informations de l'utilisateur!</h1>

<p>

Adresse : <%=info.getAdr()%> <br>

Nom : <%=info.getNom()%> <br>

<p>

</body>

</html>

#### erreur.jsp

<%--

Document : erreur

Created on : 24.03.2023, 14:12:59

Author : FranzenL

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

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

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

</head>

<body>

<jsp:useBean id="error" scope="session" class="beans.BeanError"/>

<h1>Page d'erreur</h1>

<p> <%=error.getMessage()%><br>

<%=error.getNum()%><br>

<%=error.getSource()%><br>

<%=error.getContexte()%><br>

<%=error.getRedirection()%><br>

</p>

<a href="index.html">Retour à la page de login.</a>

</body>

</html>

### Résultat

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Quand le login était un succès :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Quand il y a une erreur :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## TT10 – Création utilisation de service REST

### Création d’un serveur REST

#### Création de projet

On créer de nouveau comme normal un projet de web avec un nom.

Ein Bild, das Text enthält.

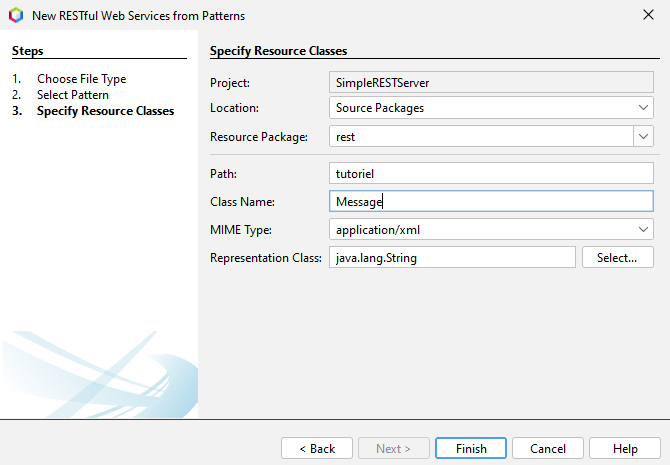
Automatisch generierte Beschreibung

On va créer un nouveau fichier RESTful



Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



Ce que nous donnes deux fichiers :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

#### Configuration du projet

Dans la classe « ApplicationConfig », une seule ligne nous intéresse. Elle permet de définir le répertoire de toutes nos applications WEB du projet.

Ein Bild, das Text, Brief enthält.

Automatisch generierte Beschreibung

#### Classe REST

Dans notre classe Message on peut maintenant mettre des méthodes.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Notre première annotation sera le verbe HTML que l’on va utiliser dans notre méthode. En REST, il y a quatre types d’opérations possibles : le CRUD :

* Create : POST
* Read : GET
* Update : PUT
* Delete : DELETE

Alors dans notre cas pour recevoir le message on va mettre l’annotation @GET en dessus de la méthode :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Notre String sera renvoyé en texte brut. Nous allons rajouter l’annotation « @Produces(javax.ws.rs.core.MediaType.TEXT\_PLAIN) ».

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Il existe également une annotation « @Consumes » qui permet de spécifier le type MIME d’entrée de la méthode.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

#### Test avec Postman

Pour tester nos méthodes de type RESTful, on va utiliser Postman.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Pour faire cette requête on doit regarder sur diffèrent choses :

**Context.xml :**

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

**ApplicationConfig.java :**

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

**Message.java**

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Brief enthält.

Automatisch generierte Beschreibung

* localhost :8085 : Il s’agit de l’adresse de notre site. Cette adresse signifie que nous sommes sur le serveur local de Netbeans, étant donné que Netbeans utilise le port 8084.
* Simple\_REST\_Server : Il s’agit de notre nom de projet (ou le nom de contexte).
* webresources : il s’agit du répertoire qui contient les ressources de nos applications. Ce répertoire peut être modifié sur la classe «ApplicationConfig ».
* tutoriel : il s’agit du répertoire contenant notre application. Ce répertoire peut être défini sur la classe de l’application, dans l’annotation @Path.
* getMessage : Il s’agit de la méthode ou on a donnée l’annotation @Path.

### Création d’un client REST

#### Création du projet

On créer de nouveau comme normal un projet de web avec un nom.

Cette fois on va choisir « RESTful Java Client ».

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Pour REST ressource, on va le prendre depuis le projet serveur qu’on a créé avant.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Finish  
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

#### Création de Servlet

On va créer un servlet :

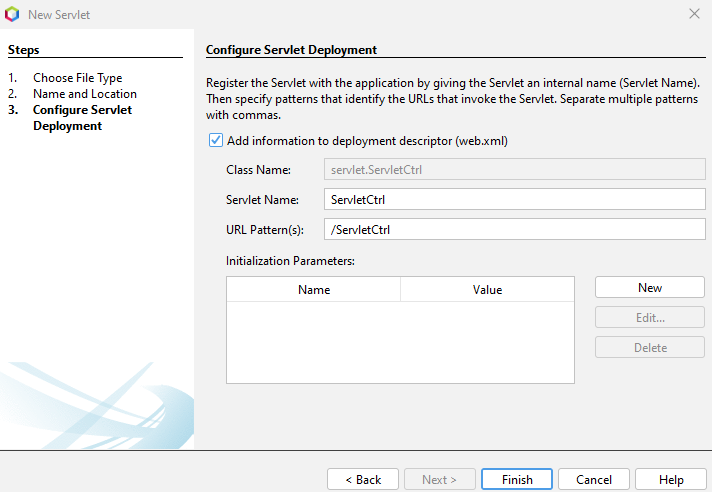
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

On va l’appeler comme suivant :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung





#### Page client

Pour la page client, on va mettre deux boutons pour faire les deux requetes : getMessage et getAuthor :

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

#### Codage Servlet

Premièrement on va instancier notre client :

Ein Bild, das Website enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

#### Résultat

Si on veut héberger notre site, on doit juste modifier l’URL de base dans la classe client :

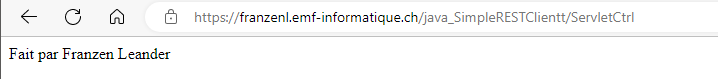


Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



# Synthèse S1

Cette semaine, nous avons vu comment utiliser Tomcat. Nous l'avons installé et avons regardé comment Tomcat fonctionne. Tomcat est un serveur web virtuel. Nous l'avons connecté à Netbeans pour que l'on puisse y programmer les pages et quand on lance l'application, elle ouvre la page sur Tomcat dans le navigateur. Quand on clique sur "build" dans Netbeans, un fichier .war est créé dans le dossier dist. On télécharge ensuite ce fichier sur CPanel et on peut accéder à la page par nom de fichier. Pour que cela fonctionne, le nom doit commencer par java\_.

On peut intégrer du code Java dans le fichier html : <%...code...%>.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ainsi, on peut par exemple récupérer des informations d'input à partir du code. On peut utiliser des autres services qu’on peut accéder déçu avec une requête http pour qu’on peut prendre des informations supplémentaires externe.

Dans le code Java, on peut aussi envoyer des balises html, avec la méthode out.println. Ainsi, il n'est pas nécessaire de fermer et de rouvrir la balise java.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Nous avons également vu comment connecter une base de données à Cpanel et en tirer des informations. Pour cela, il faut ajouter la base de données dans cpanel, créer un utilisateur et le lier à la base de données. Pour cela, il faut encore le MySQL connector driver dans la bibliothèque de l'application.

# Conclusion

Je trouve qu’on a bien travaillé durant ce module. La communication était très bien dans le groupe on savait s’aider entre deux et on pourrait mettre en place le projet très efficace. Dans la remédiation, les choses étaient beaucoup plus claires qu’avant. On pouvait bien terminer le travail.