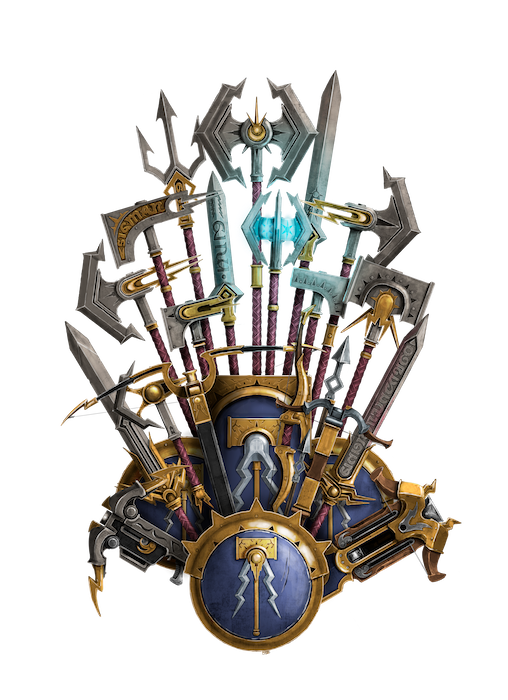
133 – Réaliser des applications Web en Session-Handling

# Rapport personnel

Version 1 du 8 mai 2022

Dreosti Robin

Module du 21.03.2022 au 03.05.2022.

Table de matière

[Rapport personnel 1](#_Toc99973557)

[Table de matière 2](#_Toc99973558)

[*Introduction et contexte du project* 4](#_Toc99973559)

[Use case avec les explications des cas pour les applications 4](#_Toc99973560)

[Activity Diagram d'un cas complet navigant dans les applications avec les explications 4](#_Toc99973561)

[Maquettes du projet avec la navigation intégrée et ses explications 4](#_Toc99973562)

[Sequence System global entre les applications 4](#_Toc99973563)

[Class Diagram complet avec les explications de chaque application 4](#_Toc99973564)

[Navigation Diagram complet avec les explications des applications 4](#_Toc99973565)

[*Bases de données* 4](#_Toc99973566)

[Modèles ER dans EA 4](#_Toc99973567)

[DBUsers 4](#_Toc99973568)

[DBData 4](#_Toc99973569)

[Modèles WorkBench MySQL 5](#_Toc99973570)

[DBUsers 5](#_Toc99973571)

[DBData 5](#_Toc99973572)

[Requêtes SQL utiles 5](#_Toc99973573)

[*Implémentation des applications <<Le client Ap1>> et <<Le client Ap2>>* 6](#_Toc99973574)

[Parties JSP sur serveur Tomcat de chacun 6](#_Toc99973575)

[Parties HTML-CSS-JS de chacun 6](#_Toc99973576)

[*Implémentation de l'aplication <<Le douanier commun>>* 6](#_Toc99973577)

[Partie Servlet sur serveur Tomcat commun 6](#_Toc99973578)

[*Implémentation des aplications <<L'état major Ap1>> et <<L'état major Ap2>>* 6](#_Toc99973579)

[7.1 Parties REST et JDBC MySQL sur compte Tizoo de chacun 6](#_Toc99973580)

[Parties Servlet de chacun 6](#_Toc99973581)

[*Références des JAVADoc* 6](#_Toc99973582)

[*Hébergement* 6](#_Toc99973583)

[*Installation du projet complet avec les 5 applications* 6](#_Toc99973584)

[*Outils, langages (versions, définitions, installations)* 6](#_Toc99973585)

[*Tests de fonctionnement du projet* 6](#_Toc99973586)

[Exercice 1 6](#_Toc99973587)

[Exercice 2 7](#_Toc99973588)

[Exercice 3 9](#_Toc99973589)

[Exercice 4 10](#_Toc99973590)

[Exercice5 12](#_Toc99973591)

[Exercice6 13](#_Toc99973592)

[Exercice 10 17](#_Toc99973593)

[Serveur REST 17](#_Toc99973594)

[Création d’un server DB REST 23](#_Toc99973595)

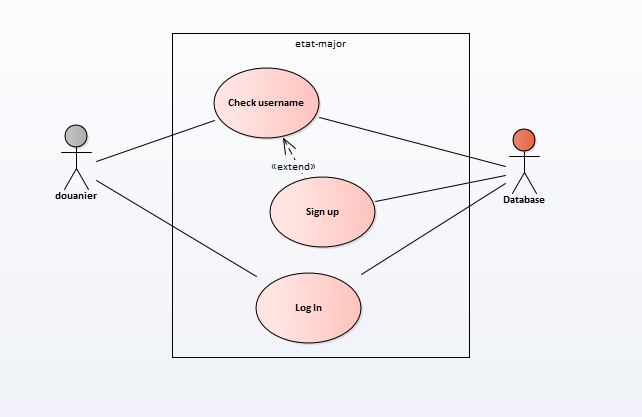
[*Auto-évaluations et conclusions* 24](#_Toc99973596)

[*Tests technologiques selon les exercices* 24](#_Toc99973597)

## *Introduction et contexte du project*

Le concept du projet est basé sur un jeu nommé deep rock galactic, un jeu futuriste sur des nains. Le but est de proposé une page avec des missions tiré d’une base de données avec des objectifs secondaires eux aussi tiré d’une DB. On veut aussi créer un Administrateur qui va pouvoir voir tous les comptes des ressources miné par les nains.

### Use case avec les explications des cas pour les applications



Use case pour le login entre le douanier, l’état-major et la data base

### Activity Diagram d'un cas complet navigant dans les applications avec les explications



### Maquettes du projet avec la navigation intégrée et ses explications



### Sequence System global entre les applications



### Navigation Diagram complet avec les explications des applications



## *Bases de données*

### Modèles ER dans EA

#### DBUsers

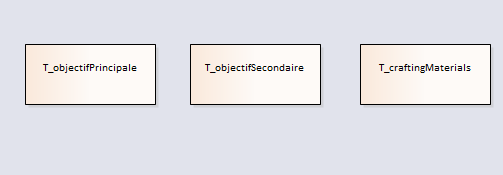
Une image contenant texte

Description générée automatiquement

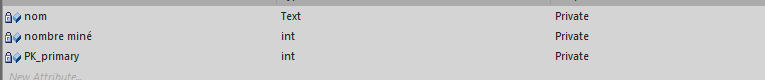
Une image contenant table

Description générée automatiquement

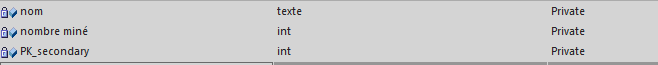
#### DBData



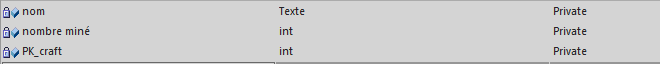
**T\_objectifPrincipale**

****

**T\_objectifSecondaire**

****

**T\_craftingMaterials**

****

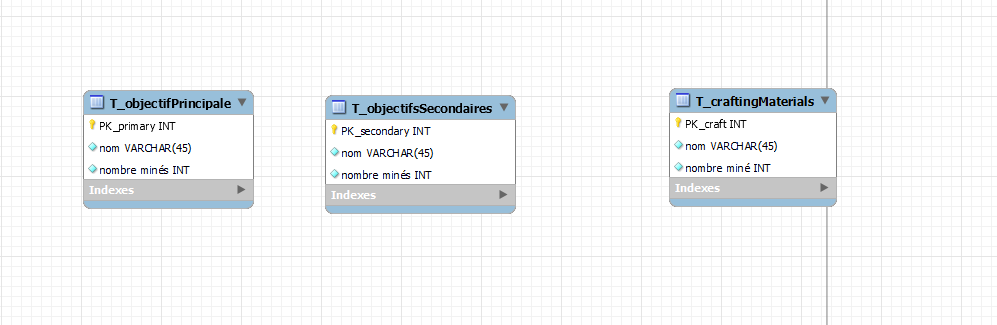
### Modèles WorkBench MySQL

#### DBUsers

Une image contenant texte

Description générée automatiquement

#### DBData



### Requêtes SQL utiles

Select \* from T

## *Implémentation des applications <<Le client Ap1>> et <<Le client Ap2>>*

### Parties JSP sur serveur Tomcat de chacun

Méthode POST du login

function POST\_Login(user, password, loginSuccess, loginError){

$.ajax({

method: "POST",

dataType: "json",

data: "action=login&username="+user+"&password="+password,

url: BASE\_URL,

success: loginSuccess,

error: loginError

});

}

Méthode GET de l’objectif primaire :

function GET\_PrimaryUser(getPrimaryUserSucces, getPrimaryUserFailed){

$.ajax({

method: "GET",

dataType: "json",

data: "action=primaryUser",

url: BASE\_URL,

success: getPrimaryUserSucces,

error: getPrimaryUserFailed

});

}

Méthode GET de l’objectif secondaire :

function GET\_SecondaryUser(getSecondaryUserSucces, getSecondaryUserFailed){

$.ajax({

method: "GET",

dataType: "json",

data: "action=secondaryUser",

url: BASE\_URL,

success: getSecondaryUserSucces,

error: getSecondaryUserFailed

});

}

La méthode GET des deux matériaux disponibles :

function GET\_MaterialsUser(getMaterialsUserSucces, getMaterialsUserFailed){

$.ajax({

method: "GET",

dataType: "json",

data: "action=materialsUser",

url: BASE\_URL,

success: getMaterialsUserSucces,

error: getMaterialsUserFailed

});

}

La méthode GET des datas de l’admin :

function GET\_DataAdmin(getDataAdminSucces, getDataAdminFailed){

$.ajax({

method: "GET",

dataType: "json",

data: "action=dataAdmin",

url: BASE\_URL,

success: getDataAdminSucces,

error: getDataAdminFailed

});

}

### Parties HTML-CSS-JS de chacun

#### HTML

##### Index

<!DOCTYPE html>

<html>

<head>

<title>DeepRockManagement - Login Page</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="css/stylesheet.css">

<link rel="icon" type="image/png" sizes="16x16" href="images/icon.png">

<script src="javascripte/controllers/indexCtrl.js" type="text/javascript"></script>

<script src="javascripte/helpers/serviceHttp.js" type="text/javascript"></script>

<script src="https://code.jquery.com/jquery-3.5.1.min.js" type="text/javascript"></script>

</head>

<body>

<h1>

<span class="title-word title-word-1">D</span>

<span class="title-word title-word-2">R</span>

<span class="title-word title-word-3">G</span>

- DeepRockGalactic Inc

</h1>

<div id="bannerimage"></div>

<br>

<h2 class="center" >↓ Entrez vos détails d'utilisateur ci-dessous ↓</h2>

<br>

<hr class="rounded">

<br>

<br>

<table class="tabelCenter">

<tr>

<td>Nom d'utilisateur :</td>

<td>

<input class="textfiller" type="text" name="username" id="user" maxlength="50" />

</td>

</tr>

<tr>

<td>

<br>

</td>

</tr>

<tr>

<td>Mot de passe :</td>

<td>

<input class="textfiller" type="text" name="password" id="password" maxlength="50" />

</td>

</tr>

</table>

<br>

<br>

<div class="center">

<a href="#" class="indexBtn" onclick="doLogin()">Login</a>

</div>

<br>

<div id="logginInfo" style="display: none;">/!\ Veuillez vérifier vos paramètres de Loggin /!\ </div>

<br>

<hr class="rounded">

<br>

<h2 class="center">Pas de compte ?</h2>

<h2 class="center">Aucun Soucis, enregistrez vous ci-dessous !</h2>

<br>

<div class="center">

<a href="./registration.html" class="indexBtn">Register</a>

</div>

<img src="images/mushroom.png" onclick="mushroom()" >

</body>

</html>

##### Registration

<!DOCTYPE html>

<html>

<head>

<title>DeepRockManagement - Register Page</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<script src="javascripte/helpers/serviceHttp.js" type="text/javascript"></script>

<script src="javascripte/controllers/registrationCtrl.js" type="text/javascript"></script>

<link rel="stylesheet" href="css/stylesheet.css">

<link rel="icon" type="image/png" sizes="16x16" href="images/icon.png">

</head>

<body>

<h1>

<span class="title-word title-word-1">D</span>

<span class="title-word title-word-2">R</span>

<span class="title-word title-word-3">G</span>

- DeepRockGalactic Inc [WORK IN PROGRESS]

</h1>

<div id="bannerimage"></div>

<br>

<h2 class="center" >↓ Entrez les détails de votre compte ci-dessous ↓</h2>

<br>

<hr class="rounded">

<br>

<br>

<table class="tabelCenter">

<tr>

<td>Nom d'utilisateur :</td>

<td>

<input class="textfiller" type="text" name="username" id="user" maxlength="50" />

</td>

</tr>

<tr>

<td>

<br>

</td>

</tr>

<tr>

<td>Nom :</td>

<td>

<input class="textfiller" type="text" name="name" id="name" maxlength="50" />

</td>

</tr>

<tr>

<td>

<br>

</td>

</tr>

<tr>

<td>Prénom :</td>

<td>

<input class="textfiller" type="text" name="surname" id="surname" maxlength="50" />

</td>

</tr>

<tr>

<td>

<br>

</td>

</tr>

<tr>

<td>Mot de passe :</td>

<td>

<input class="textfiller" type="text" name="password" id="password" maxlength="50" />

</td>

</tr>

<tr>

<td>

<br>

</td>

</tr>

</table>

<br>

<hr class="rounded">

<br>

<h2 class="center" >Deep Rock Galactic est fier de vous compter parmis ses membres !</h2>

<br>

<div class="center">

<a href="#" class="indexBtn" onclick="doRegister()">Done</a>

</div>

</body>

</html>

##### Error

<!DOCTYPE html>

<html>

<head>

<title>DeepRockManagement - Oopsy Page</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="css/stylesheet.css">

<link rel="icon" type="image/png" sizes="16x16" href="images/icon.png">

</head>

<body>

<h1>

<span class="title-word title-word-1">D</span>

<span class="title-word title-word-2">R</span>

<span class="title-word title-word-3">G</span>

- DeepRockGalactic Inc

</h1>

<div id="bannerimage"></div>

<br>

<h2 class="center" >Oops !</h2>

<h2 class="center" >Un problème est survenu...</h2>

<br>

<img class="imageMiddleBorder" src="images/errorgif.gif" height="350">

<br>

<br>

<hr class="rounded">

<br>

<br>

<div class="center">

<a href="./index.html" class="indexBtn">Revenir à l'acceuil</a>

</div>

<br>

<br>

<br><!-- <br> -->

</body>

</html>

##### Home

<!DOCTYPE html>

<html>

<head>

<title>DeepRockManagement - Home Page</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<script src="javascripte/helpers/serviceHttp.js" type="text/javascript"></script>

<script src="javascripte/controllers/homeCtrl.js" type="text/javascript"></script>

<script src="https://code.jquery.com/jquery-1.9.1.min.js"></script>

<link rel="stylesheet" href="css/stylesheet.css">

<link rel="icon" type="image/png" sizes="16x16" href="images/icon.png">

</head>

<body onload="criticalLoarder()">

<table class="testing2">

<tr>

<td>

<h1>

<span class="title-word title-word-1">D</span>

<span class="title-word title-word-2">R</span>

<span class="title-word title-word-3">G</span>

- Welcome Miner, Here is your assignement.

</h1>

</td>

<td class="right">

<a onclick="doDelog()" class="indexBtn">Déconnection</a>

</td>

</tr>

</table>

<div id="bannerimage"></div>

<br>

<h2>Objectifs de la mission :</h2>

<table class="testing">

<tr>

<td style="width: 400px">

<div class="bigcard">

<img id="image1" src="" alt="Avatar" class="cardimage">

<div class="container">

<h3 id="name1">placeholder</h3>

<p id="desc1">placeholder</p>

</div>

</div>

</td>

<td style="width: 400px">

<div class="bigcard">

<img id="image2" src="" alt="Avatar" class="cardimage">

<div class="container">

<h3 id="name2">placeholder</h3>

<p id="desc2">placeholder</p>

</div>

</div>

</td>

</tr>

</table>

<br>

<hr class="roundedFC">

<br>

<h2>Matériaux disponibles :</h2>

<table class="tabelCenter">

<tr>

<td>

<div class="card">

<img id="image3" src="" alt="Avatar" style="width:100%">

<div class="container">

<h3 id="name3">placeholder</h3>

</div>

</div>

</td>

<td>

<div class="card">

<img id="image4" src="" alt="Avatar" style="width:100%">

<div class="container">

<h3 id="name4">placeholder</h3>

</div>

</div>

</td>

</tr>

</table>

</body>

</html>

##### AdminHome

<!DOCTYPE html>

<html>

<head>

<title>DeepRockManagement - Home Page</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<script src="javascripte/helpers/serviceHttp.js" type="text/javascript"></script>

<script src="javascripte/controllers/adminHomeCtrl.js" type="text/javascript"></script>

<script src="javascripte/workers/adminHomeWrk.js" type="text/javascript"></script>

<script src="https://code.jquery.com/jquery-1.9.1.min.js"></script>

<link rel="stylesheet" href="css/stylesheet.css">

<link rel="icon" type="image/png" sizes="16x16" href="images/icon.png">

</head>

<body onload="doLoader()">

<table>

<tr>

<td>

<h1>

<span class="title-word title-word-1">D</span>

<span class="title-word title-word-2">R</span>

<span class="title-word title-word-3">G</span>

- Welcome Mission Control.

</h1>

</td>

<td class="right">

<a onclick="doDelog()" class="indexBtn">Déconnection</a>

</td>

</tr>

</table>

<br>

<hr class="roundedFC">

<br>

<h2>Liste des matériaux récupéré par les mineurs :</h2>

<br>

<hr class="rounded">

<br>

<div id="row"></div>

</body>

</html>

#### CCS

/\* DeepRockManagement StyleSheet for Index \*/

/\* Créateur : Julien Dubois \*/

/\* 18.02.2022 \*/

/\* Le document CSS permets d'améliorer le visuel du site web \*/

\*{

font-family:Arial;

color: #6EB98B;

}

html{

background-color: #272C29;

}

h1{

font-size: 36pt;

}

#bannerimage{

width: 100%;

background-image: url("../images/deep-rock-galactic-banner.jpg");

height: 200px;

background-size:cover;

background-position: center;

background-repeat: no-repeat;

display: block;

}

.center{

display: block;

text-align: center;

margin-left: auto;

margin-right: auto;

}

.imageMiddleBorder{

display: block;

text-align: center;

margin-left: auto;

margin-right: auto;

border-left: 5px solid #6EB98B;

border-right: 5px solid #6EB98B;

border-top: 5px solid #6EB98B;

border-bottom: 5px solid #6EB98B;

border-radius: 3px;

}

.tabelCenter{

text-align: right;

margin-left: auto;

margin-right: auto;

}

.textfiller{

size: 50px;

border: 2px solid #0AE45C;

border-radius: 4px;

background-color: #C5F8D8;

color: black;

}

.indexBtn {

box-shadow: 3px 4px 0px 0px #1BA44F;

background:linear-gradient(to bottom, #43D079 5%, #2DB260 100%);

background-color:#31A95F;

border-radius:18px;

border:1px solid #0AE45C;

display:inline-block;

cursor:pointer;

color:#ffffff;

font-family:Arial;

font-size:17px;

padding:7px 25px;

text-decoration:none;

text-shadow:0px 1px 0px #0BB64C;

}

.indexBtn:hover {

background:linear-gradient(to bottom, #29E36F 5%, #1FD664 100%);

background-color:#21C25E;

}

.indexBtn:active {

position:relative;

top:1px;

}

hr.rounded {

border-top: 5px solid #6EB98B;

border-radius: 3px;

width: 16cm;

}

hr.roundedFC {

border-top: 5px solid #6EB98B;

border-radius: 3px;

width: auto;

}

.title-word {

font-family:Neon;

animation: color-animation 4s linear infinite;

}

.title-word-1 {

--color-1: #8debb1;

--color-2: #6EB98B;

--color-3: #6EB98B;

}

.title-word-2 {

--color-1: #6EB98B;

--color-2: #8debb1;

--color-3: #6EB98B;

}

.title-word-3 {

--color-1: #6EB98B;

--color-2: #6EB98B;

--color-3: #8debb1;

}

@keyframes color-animation {

0% {color: var(--color-1)}

32% {color: var(--color-1)}

33% {color: var(--color-2)}

65% {color: var(--color-2)}

66% {color: var(--color-3)}

99% {color: var(--color-3)}

100% {color: var(--color-1)}

}

@font-face {

font-family:Neon;

src: url(../images/neoncircuit.ttf);

}

.right {

text-align: right;

margin-right: 1em;

}

.card {

/\* Add shadows to create the "card" effect \*/

box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2);

transition: 0.3s;

background-color: #343d38;

border-radius: 3px;

}

.bigcard {

/\* Add shadows to create the "card" effect \*/

box-shadow: 0 5px 13px 0 rgba(0,0,0,0.2);

transition: 0.3s;

background-color: #343d38;

border-radius: 3px;

}

.container {

padding: 2px 16px;

}

.cardimage{

width: 200px;

height: 200px;

}

.column {

float: left;

width: 25%;

}

.row:after {

content: "";

display: table;

clear: both;

}

.testing {

table-layout: fixed;

text-align: center;

margin-left: auto;

margin-right: auto;

}

.testing2 {

margin-left: auto;

margin-right: auto;

}

#### JS

##### IndexCtrl

Cette méthode récupère les valeurs nécessaires au login qui sont le username ainsi que le mot de passe puis appelle la méthode permettant de faire un POST du login vers le serveur. Pour cela elle utilise un getElementById.

function doLogin(){

var user = document.getElementById("user").value;

var password = document.getElementById("password").value;

POST\_Login(user ,password , loginSuccess, loginError);

}

Cette méthode est appelée en cas de succes du login, elle va vérifier si l'utilisateur est un administrateur ou un simple utilisateur et le rediriger sur la page en fonction de ce test.

function loginSuccess(data){

console.log("ceci est une réussite");

console.log(JSON.stringify(data));

if (data['isAdmin']) {

window.open("adminHome.html", "\_self");

}else{

window.open("home.html", "\_self");

}

}

Cette méthode est appelée en cas d'erreur de login et va rediriger le visiteur sur une page qui l'informera de l'erreur.

function loginError(data){

console.log("ceci est une erreur");

console.log(data);

window.open("error.html", "\_self");

}

Cette méthode joue un fichier audio si on appuie sur un bouton.

function mushroom(){

var audio = new Audio('./images/Mushroom.mp3');

audio.play();

}

##### HomeCtrl

Cette fonction est appelée lors du chargement de la page et appelle les divers méthodes permettant de faire un GET de chaque informations nécessaire.

function criticalLoarder() {

console.log("Loader Starting...");

loader\_primaryObjectif();

loader\_secondaryObjectif();

loader\_materials();

}

Cette méthode appelle le GET de l'objectif primaire.

function loader\_primaryObjectif() {

GET\_PrimaryUser(getPrimarySuccess, getPrimaryFailed);

}

Cette méthode est appelée en cas de succes du GET de l'objectif primaire puis appelle la méthode permettant de l'afficher.

function getPrimarySuccess(data) {

console.log("Primary User Objective Succeded !");

console.log(JSON.stringify(data));

printPrimaryObjective(data);

}

Cette méthode est appelée en cas d'erreur lors du GET de l'objectif primaire. Et print dans la console le résultat du GET.

function getPrimaryFailed(data) {

console.log("Primary User Objective Failed.");

console.log(JSON.stringify(data));

}

Cette méthode permet d'afficher l'objectif primaire sur la page HTML.

function printPrimaryObjective(primary) {

if ((primary !== null)) {

document.getElementById("image1").setAttribute("src", primary['URL']);

document.getElementById("name1").innerHTML = primary['nom'];

document.getElementById("desc1").innerHTML = primary['descriptif'];

}

}

Cette méthode appelle le GET de l'objectif secondaire.

function loader\_secondaryObjectif() {

GET\_SecondaryUser(getSecondarySuccess, getSecondaryFailed);

}

Cette méthode est appelée en cas de succes du GET de l'objectif secondaire puis appelle la méthode permettant de l'afficher.

function getSecondarySuccess(data) {

console.log("Secondary User Objective Succeded !");

console.log(JSON.stringify(data));

printSecondaryObjective(data);

}

Cette méthode est appelée en cas d'erreur lors du GET de l'objectif secondaire. Et print dans la console le résultat du GET.

function getSecondaryFailed(data) {

console.log("Secondary User Objective Failed.");

console.log(JSON.stringify(data));

}

Cette méthode permet d'afficher l'objectif secondaire sur la page HTML.

function printSecondaryObjective(secondary) {

if ((secondary !== null)) {

document.getElementById("image2").setAttribute("src", secondary['URL']);

document.getElementById("name2").innerHTML = secondary['nom'];

document.getElementById("desc2").innerHTML = secondary['descriptif'];

}

}

Cette méthode appele le GET de les deux matériaux disponibles.

function loader\_materials() {

GET\_MaterialsUser(getMaterialsSuccess, getMaterialsFailed);

}

Cette méthode est appelée en cas de succes du GET des deux matériaux disponibles puis appelle la méthode permettant de l'afficher.

function getMaterialsSuccess(data) {

console.log("Materials Succeded !");

console.log(JSON.stringify(data));

printMaterials(data);

}

Cette méthode est appelée en cas d'erreur lors du GET des matériaux. Et print dans la console le résultat du GET.

function getMaterialsFailed(data) {

console.log("Materials Failed.");

console.log(JSON.stringify(data));

}

Cette méthode permet d'afficher les deux matériaux disponible sur la page HTML.

function printMaterials(materialsArray) {

if ((materialsArray !== null)) {

var materialNo1 = materialsArray[0];

document.getElementById("image3").setAttribute("src", materialNo1['URL']);

document.getElementById("name3").innerHTML = materialNo1['nom'];

var materialNo2 = materialsArray[1];

document.getElementById("image4").setAttribute("src", materialNo2['URL']);

document.getElementById("name4").innerHTML = materialNo2['nom'];

}

}

Cette méthode permet de quitter la page.

function doDelog() {

window.open("index.html", "\_self");

}

##### AdminHomeCtrl

Cette méthode est appelée au chargement de la page et fait appel a un GET permettant de charger les données administrateur.

function doLoader(){

GET\_DataAdmin(loaderSuccess, loaderfailed);

}

Cette méthode est appelée en cas du succes du chargement des données, elle informe de son succes et appelé à méthode printData qui va afficher les données.

function loaderSuccess(data){

console.log("Loader Executed Successfully !")

printData(data);

}

Cette méthode est appelée en cas d'erreur lors du chargement des données, elle va simplement écrire le résultat du GET dans la console.

function loaderfailed(data){

console.log("Loader Failed To Opperate.");

console.log(JSON.stringify(data));

}

Cette méthode permet de quitter la page.

function doDelog() {

window.open("index.html", "\_self");

}

Cette méthode permet d'ajouter du texte HTMl a un id.

function appendToId(id,string){

$(id).append(string);

}

##### AdminHomeWorker

Cette méthode permet de formater les valeurs reçus lors du GET en une carte en code HTML.

function printData(dataArray) {

var added = "";

for (var item in dataArray) {

added = added + '<div class="column">' +

' <div class="card">' +

'<img id="image4" src="'+item['URL']+'" alt="Avatar" class="cardimage">' +

'<div class="container">' +

'<h3 id="name4">'+item['nom']+'</h3>' +

'<p id="desc4">'+item['descriptif']+'</p>' +

'</div>' +

'</div>' +

'</div>';

}

appendToId("#row", added);

}

## *Implémentation de l'aplication <<Le douanier commun>>*

### Partie Servlet sur serveur Tomcat commun

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

PrintWriter out = response.getWriter();

try {

if (request.getParameter("action").equals("login")) {

String result = client.checkLogin(request.getParameter("username"), request.getParameter("password"));

out.println(result);

}

} catch (Exception e) {

e.printStackTrace();

}

}

Méthode pour les POST pour pouvoir faire le login sur le douanier

public class NewJerseyClient {

private WebTarget webTarget;

private Client client;

private static final String BASE\_URI = "https://dreostir.emf-informatique.ch/151TestSite/javaRESTSite/webresources";

public NewJerseyClient() {

client = javax.ws.rs.client.ClientBuilder.newClient();

webTarget = client.target(BASE\_URI).path("Wrk");

}

public String checkLogin(String Login, String password) throws ClientErrorException {

WebTarget resource = webTarget;

resource = resource.path("checkLogin");

Form form = new Form();

form.param("content", Login);

form.param("password", password);

Invocation.Builder request = resource.request(javax.ws.rs.core.MediaType.APPLICATION\_JSON);

Response response = request.post(Entity.form(form), Response.class);

if (response != null) {

return response.readEntity(String.class);

} else {

return "ERROR";

}

}

public void close() {

client.close();

}

}

Client rest pour le server REST qui gère les logins

Méthode pour les GET pour pouvoir faire le login sur le douanier

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("text/html");

response.setHeader("Access-Control-Allow-Origin", "\*");

PrintWriter out = response.getWriter();

URLConnection connection = null;

try {

String result = "error";

if (request.getParameter("action").equals("primaryUser")) {

result = data.getPrimary();

} else if (request.getParameter("action").equals("secondaryUser")) {

result = data.getSecondary();

} else if (request.getParameter("action").equals("materialsUser")) {

result = data.getMineraux();

} else if (request.getParameter("action").equals("dataAdmin")) {

result = data.getAll();

}

out.println(result);

} catch (Exception e) {

e.printStackTrace();

}

}

## *Implémentation des aplications <<L'état major Ap1>> et <<L'état major Ap2>>*

### AP1

Méthode checkLogin sur l’état major AP1

@POST

@Path("checkLogin")

@Consumes(MediaType.APPLICATION\_FORM\_URLENCODED)

@Produces(MediaType.APPLICATION\_JSON)

public String checkLogin(@FormParam("content") String content, @FormParam("password") String password) {

ArrayList<String> arrayUser = new ArrayList<>();

Gson gson;

String result = "not find";

try {

gson = new Gson();

Db db = new Db();

db.connectDB("jdbc:mysql://dreostir.emf-informatique.ch:3306/dreostir\_db133?serverTimezone=UTC", "dreostir\_db133", "aZb@&YA4N9FW");

result = content;

Users user = db.getLogin(content);

if (user != null) {

if ((user.getLogin().equals(content)) && (user.getPassword()).equals(password)) {

result = gson.toJson(user);

}

}else{

result="pas trouvé";

}

db.disconnectDB();

} catch (SQLException e) {

return "{\"error\":\"" + e.getMessage() + "\"}";

}

////On affiche notre résultat.

return result;

}

Méthode getLogin pour récupérer l’user avec son login

public Users getLogin(String Login) throws SQLException {

Users result = null;

PreparedStatement preparedStatement = null;

String statementTexte = "SELECT \* FROM T\_users where login = ?;";

preparedStatement = connection.prepareStatement(statementTexte);

preparedStatement.setString(1, Login);

ResultSet resultSet = preparedStatement.executeQuery();

if (resultSet.next()) {

result = new Users(resultSet.getString("nom"), resultSet.getString("prenom"), resultSet.getString("login"), resultSet.getBoolean("isAdmin"), resultSet.getString("password"));

}else{

result = null;

}

resultSet.close();

preparedStatement.close();

return result;

}

### AP2

Méthode get pour avoir un objectif primaire en fonction d’un PK aléatoire

@GET

@Path("getPrimary")

@Produces(MediaType.APPLICATION\_JSON)

public String getPrimary() {

Gson gson;

String result = "not find";

try {

gson = new Gson();

int nbrPK = (int) (Math.random() \* (8 - 1 + 1)) + 1;

Db db = new Db();

db.connectDB("jdbc:mysql://dreostir.emf-informatique.ch:3306/dreostir\_db133?serverTimezone=UTC", "dreostir\_db133", "aZb@&YA4N9FW");

PrimaryObjectif primary = db.getPrimary(String.valueOf(nbrPK));

if (primary != null) {

result = gson.toJson(primary);

}

db.disconnectDB();

} catch (Exception e) {

return "{\"error\":\"" + e.getMessage() + "\"}";

}

Methode get pour avoir un objectif secondaire a parit d’un PJK aléatoire

@GET

@Path("getSecondary")

@Produces(MediaType.APPLICATION\_JSON)

public String getSecondary() {

Gson gson;

String result = "not find";

try {

gson = new Gson();

int nbrPK = (int) (Math.random() \* (8 - 1 + 1)) + 1;

Db db = new Db();

db.connectDB("jdbc:mysql://dreostir.emf-informatique.ch:3306/dreostir\_db133?serverTimezone=UTC", "dreostir\_db133", "aZb@&YA4N9FW");

result = "not find";

SecondaryObjectif secondary = db.getSecondary(String.valueOf(nbrPK));

if (secondary != null) {

result = gson.toJson(secondary);

}

db.disconnectDB();

} catch (Exception e) {

return "{\"error\":\"" + e.getMessage() + "\"}";

}

//On affiche notre résultat.

return result;

}

Methode get pour avoir deux minéraux différents avec une PK aléatoire

@GET

@Path("getMineraux")

@Produces(MediaType.APPLICATION\_JSON)

public String getMineraux() {

ArrayList<Mineral> arrayMineraux = new ArrayList<>();

Gson gson;

String result = "not find";

try {

gson = new Gson();

Db db = new Db();

db.connectDB("jdbc:mysql://dreostir.emf-informatique.ch:3306/dreostir\_db133?serverTimezone=UTC", "dreostir\_db133", "aZb@&YA4N9FW");

result = "not find";

Mineral mineral;

int nbrPK1 = (int) (Math.random() \* (6 - 1 + 1)) + 1;

mineral = db.getMineral(String.valueOf(nbrPK1));

arrayMineraux.add(mineral);

int nbrPK2 = (int) (Math.random() \* (6 - 1 + 1)) + 1;

do {

nbrPK2 = (int) (Math.random() \* (6 - 1 + 1)) + 1;

} while (nbrPK1 == nbrPK2);

mineral = db.getMineral(String.valueOf(nbrPK2));

arrayMineraux.add(mineral);

if (!arrayMineraux.isEmpty()) {

result = gson.toJson(arrayMineraux);

} else {

result = gson.toJson("error");

}

db.disconnectDB();

} catch (Exception e) {

return "{\"error\":\"" + e.getMessage() + "\"}";

}

//On affiche notre résultat.

return result;

}

Méthode GetPrimary pour avoir les données de la db

public PrimaryObjectif getPrimary(String PK\_Primary) throws SQLException {

PrimaryObjectif result = null;

PreparedStatement preparedStatement = null;

String statementTexte = "SELECT \* FROM T\_primaryObjectifs where PK\_Primary = ?;";

preparedStatement = connection.prepareStatement(statementTexte);

preparedStatement.setString(1, PK\_Primary);

ResultSet resultSet = preparedStatement.executeQuery();

if (resultSet.next()) {

result = new PrimaryObjectif(String.valueOf(resultSet.getString("PK\_Primary")), resultSet.getString("URL"), resultSet.getString("nom"), resultSet.getString("descriptif"), String.valueOf(resultSet.getString("nombreRecolte")), String.valueOf(resultSet.getString("nombreObjectif")));

} else {

result = null;

}

resultSet.close();

preparedStatement.close();

return result;

}

Méthode getSecondary pour les données de la db

public SecondaryObjectif getSecondary(String PK\_Secondary) throws SQLException {

SecondaryObjectif result = null;

PreparedStatement preparedStatement = null;

String statementTexte = "SELECT \* FROM T\_SecondaryObjectifs where PK\_Secondary = ?;";

preparedStatement = connection.prepareStatement(statementTexte);

preparedStatement.setString(1, PK\_Secondary);

ResultSet resultSet = preparedStatement.executeQuery();

if (resultSet.next()) {

result = new SecondaryObjectif(String.valueOf(resultSet.getString("PK\_Secondary")), resultSet.getString("nom"), resultSet.getString("URL"), resultSet.getString("descriptif"), String.valueOf(resultSet.getString("nombreRecolte")), String.valueOf(resultSet.getString("nombreObjectif")));

} else {

result = null;

}

resultSet.close();

preparedStatement.close();

return result;

}

Méthode getMineraux pour avoir les donnéles de la DB

public Mineral getMineral(String PK\_mineral) throws SQLException {

Mineral result = null;

PreparedStatement preparedStatement = null;

String statementTexte = "SELECT \* FROM T\_mineraux where PK\_mineraux = ?;";

preparedStatement = connection.prepareStatement(statementTexte);

preparedStatement.setString(1, PK\_mineral);

ResultSet resultSet = preparedStatement.executeQuery();

if (resultSet.next()) {

result = new Mineral(String.valueOf(resultSet.getString("PK\_mineraux")), resultSet.getString("nom"), String.valueOf(resultSet.getString("nombreRecolte")), resultSet.getString("URL"));

} else {

result = null;

}

resultSet.close();

preparedStatement.close();

return result;

}

### 7.1 Parties REST et JDBC MySQL sur compte Tizoo de chacun

Les deux server REST (état-major) ainsi que la database seront hébergté sur le compte tizoo de l’apprentis dreosti Robin.

## *Références des JAVADoc*

Toute la JavaDoc de l’application cliente a été rédigée par : Julien Dubois.

## *Hébergement*

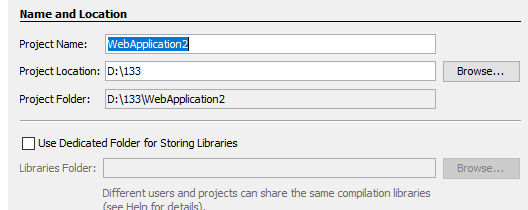
## *Installation du projet complet avec les 5 applications*

## *Outils, langages (versions, définitions, installations)*

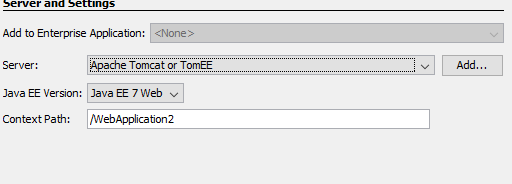
### Exercice 1



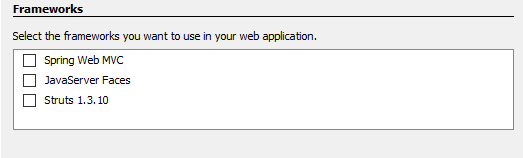
Création du projet dans Java Web/Web application



Nom et emplacement du projet



Ajout du server



On ne sélectionne rien dans la page suivante et on appuie sur finish

### Exercice 2

Le but de l’exercice est de confectionner un « login » via html et JSP.

Pour commencer, on ajoute les inputs permettant de prendre les valeurs :

<form method="post" action="checkLogin.jsp">

<label for="text">Votre Login :</label>

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

<label for="text">Votre Password</label>

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

<input type="submit" value="Log in">

</form>

Puis le corps de la page checkLogin.jds

<body>

<h1>JSP Controle votre Login</h1>

<% WrkDate wrkDate = new WrkDate("dd-MM-yyyy hh:mm:ss");

String laDate = wrkDate.donneDateCourante();%>

<p>la date du serveur est <%=laDate%></p><br><br>

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

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

%>

<p>

Votre Login est : <%=username%> <br>

Votre Password est : <%=password%> <br>

</p>

</body>

La classe workerDate :

import java.text.SimpleDateFormat;

import java.util.Date;

/\*\*

\*

\* @author DuboisA

\*/

public class WrkDate {

private String format;

public WrkDate(String format) {

this.format = format;

}

public String donneDateCourante(){

String date;

SimpleDateFormat formatdate = new SimpleDateFormat(format);

date = formatdate.format(new Date());

return date;

}

}

### Exercice 3

Pour commencer on doit créer un index.jsp avec ce code dans le body :

<%Stringcity="romont-fr";%>

< ahref="https://prevision-meteo.ch/meteo/localite/<%=city%>">

<img src="https://www.prevision-meteo.ch/uploads/widget/<%=city%>\_0.png"width="650"height="250"/>

</a>

Ensuite on va copier tout en haut du document :

**<%@page** import**=**"javax.xml.parsers.DocumentBuilderFactory"**%>**

**<%@page** import**=**"org.w3c.dom.Document"**%>**

**<%@page** import**=**"org.w3c.dom.NodeList"**%>**

**<%@page** import**=**"org.xml.sax.InputSource"**%>**

**<%@page** import**=**"org.json.JSONObject"**%>**

**<%@page** import**=**"java.io.\*"**%>**

**<%@page** import**=**"java.net.\*"**%>**

Puis on va mettre dans la première balise :

<% String city = "romont-fr";

URLConnection con = null;

try{

URL url=new URL("https://prevision-meteo.ch/services/json/" + city);

con=url.openConnection();

BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));

String inputLine;

String result="";

while((inputLine=in.readLine())!=null)

{

result+=inputLine;

}

in.close();

JSONObject results=new JSONObject(result.toString());

JSONObject city\_info=results.getJSONObject("city\_info");

out.println("Sunrise:"+city\_info.getString("sunrise")+"h<br>");

out.println("Sunset:"+city\_info.getString("sunset")+"h<br>");

out.println("Elevation:"+city\_info.getString("elevation")+"masl.<hr>");

JSONObject current=results.getJSONObject("current\_condition");

out.println("<table><thead>Temperature</td><td>WindSpeed</td></thead>");

out.println("<tbody><tr>"+current.getInt("tmp")+"°C<td>"+

current.getInt("wnd\_spd")+

"km/h</td></tbody></table>");

}catch(Exception e){e.printStackTrace();

out.println("<h2>Error while reading weather of"+city+"!</h2>");

}%>

### Exercice 4

Premièrement on va créer une classe WrkDB et la lier au index.jsp

[%@page import="WrkDB.WrkDB"%](mailto:%25@page%20import=%22WrkDB.WrkDB%22%25)

Body du index.jsp :

<% WrkDB wrkDB = new WrkDB();

String resultClasse ="";

String resultMatch="";

if (wrkDB.ouvrirDB()) {

resultClasse = wrkDB.getClasses();

resultMatch = wrkDB.getMatch();

}else{

resultClasse = "ouverture de DB impossible";

}

%>

<h1>les classe sont :</h1>

<p><%=resultClasse%></p>

<h2>les matchs sont :</h2>

<p><%=resultMatch%></p>

Classe WrkDB :

Méthode ouvrirDB :

public boolean ouvrirDB() {

boolean isOk = false;

if (conn==null) {

isOk = open();

}

return isOk;

}

Méthode open pour faire la connexion :

private boolean open() {

boolean ok = false;

String piloteOBDCPathDb = "jdbc:mysql://localhost:3306/133db" + "?zeroDateTimeBehavior=convertToNull";

try {

Class.forName("com.mysql.jdbc.Driver");

conn = DriverManager.getConnection(piloteOBDCPathDb, "root", "");

ok = true;

} catch (SQLException e) {

System.out.println(e.getMessage());

} catch (ClassNotFoundException e) {

System.out.println(e.getMessage());

} catch (Exception e) {

System.out.println(e.getMessage());

}

return ok;

}

Méthode close pour fermer la connexion :

private boolean close() {

boolean ok = false;

try {

if (conn != null) {

if (!conn.isClosed()) {

conn.close();

}

}

ok = true;

} catch (SQLException e) {

System.out.println(e.getMessage());

}

return ok;

}

Méthode getClassses pour avoir toutes les classes

public String getClasses() {

String resultat = null;

String commande = "select \* from t\_classe;";

try {

PreparedStatement prepared = conn.prepareStatement(commande);

ResultSet result= prepared.executeQuery();

resultat="";

while (result.next()){

resultat+=result.getString("pk\_classe") + " ";

}

} catch (Exception e) {

resultat = e.getMessage() + " Cause : "+ e.getCause();

}

return resultat;

}

Méthode getMatch pour avoir les matchs :

public String getMatch(){

String resultat = null;

String commande = "select \* from t\_match;";

try {

PreparedStatement prepared = conn.prepareStatement(commande);

ResultSet result= prepared.executeQuery();

resultat="";

while (result.next()){

resultat+=result.getString("pk\_match") + " " + result.getString("date\_match") + " " + result.getString("heure\_match") + "<br>" ;

}

} catch (Exception e) {

resultat = e.getMessage() + " Cause : "+ e.getCause();

}

return resultat;

}

### Exercice5

Lorsque les informations de loggin sont données et que l’input pour vérifier le loggin est donné, la jsp de vérification va effectuer ce bout de code et également l’import du bean :

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

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

<%

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

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

info.setUsername(username);

info.setPassword(password);

WrkCheckLogin wrk = new WrkCheckLogin();

if (wrk.loggCheck(info)) {

response.sendRedirect("maJspLogge.jsp");

}else{

response.sendRedirect("erreur.jsp");

}

%>

Qui fonctionne via la méthode du WrkCheckLogin qui fonctionne de la façon suivante :

public boolean loggCheck(OurUser user){

boolean islogged = false;

if ((user.getUsername().equals(username))&&(user.getPassword().equals(password))) {

islogged = true;

}

return islogged;

}

Qui utilise les attribut privates du WrkCheckLogin :

private String username = "Ash";

private String password = "Emf123";

Et les getters du beans OurUser :

public String getUsername() {

return username;

}

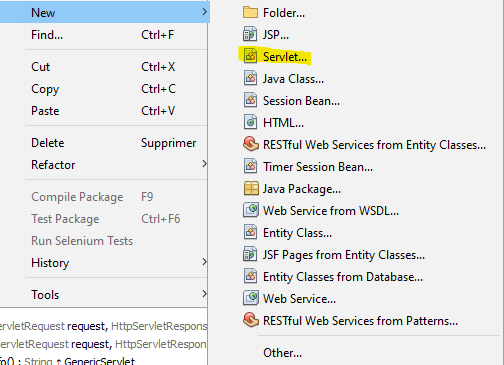
public String getPassword() {

return password;

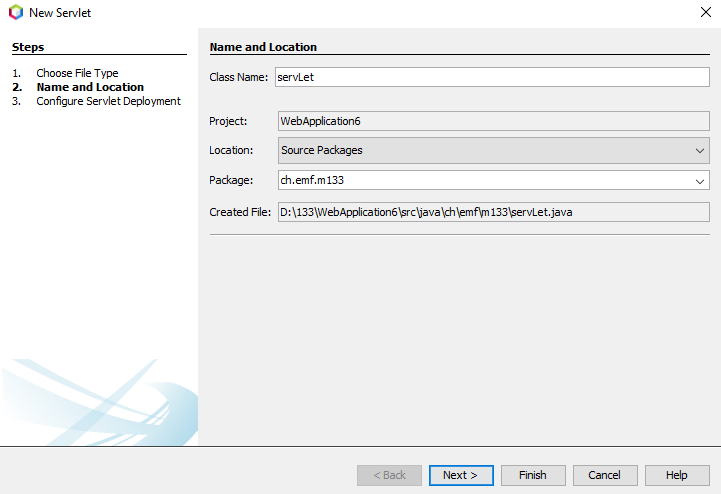
}

### Exercice6

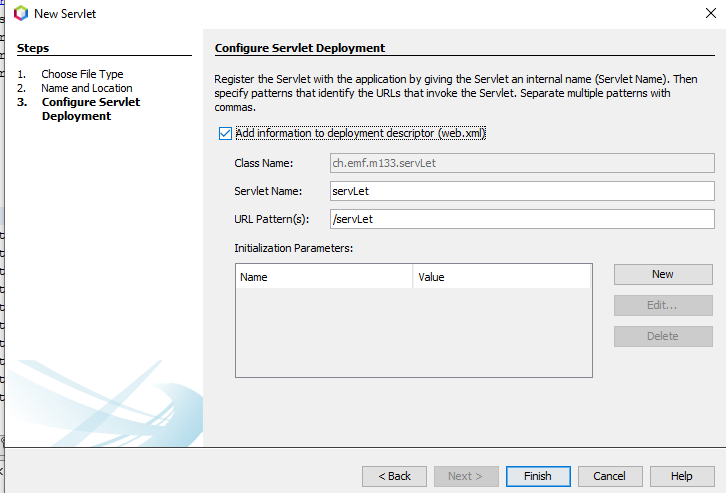
Création du ServLet



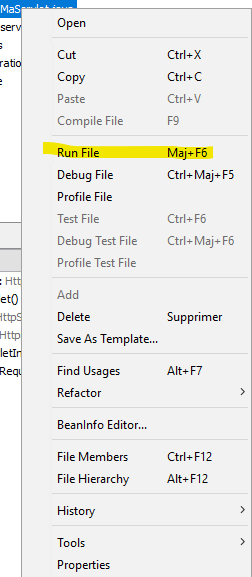
Dans cette page on met le nom du ServLet et son emplacement



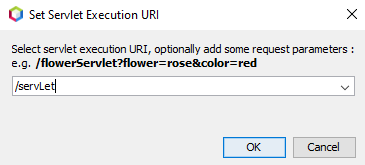
Ensuite dans cette page on doit cocher la case «Add informations »



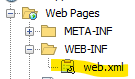
Pour pouvoir démarrer un servlet on doit faire un clic droit sur le fichier et appuyer sur « run file »



Ce qui donne son URL :



Qu’on retrouve dans le fichier XML :



A cet emplacement :



Pour continuer l’exercice on créé un login :

<html>

<head>

<title>Login Html et check Jsp</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<h1>Page de Login</h1>

<form method="post" action="/WebApplication6/MaServlet">

<label for="text">Votre Login :</label>

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

<label for="text">Votre Password</label>

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

<input type="submit" value="Log in">

</form>

</body>

</html>

Dans l’action de notre <form> on y met le chemin d’accès pour le Servlet qui se compose de /[nomDuProjet]/[URL qui est dans l’XML]

Et pour récupérer les informations du login :

String passwd = "ERROR";

String login = "ERROR";

if (request.getParameter("username") != "") {

login = request.getParameter("username");

}

if (request.getParameter("password") != "") {

passwd = request.getParameter("password");

}

/\* TODO output your page here. You may use following sample code. \*/

out.println("<!DOCTYPE html>");

out.println("<html>");

out.println("<head>");

out.println("<title>Servlet MaServlet</title>");

out.println("</head>");

out.println("<body>");

out.println("<h1>Servlet MaServlet at " + request.getContextPath() + "</h1>");

out.println("<p>votre login est: " + login + " et votre password est: " + passwd + "</p>");

out.println("</body>");

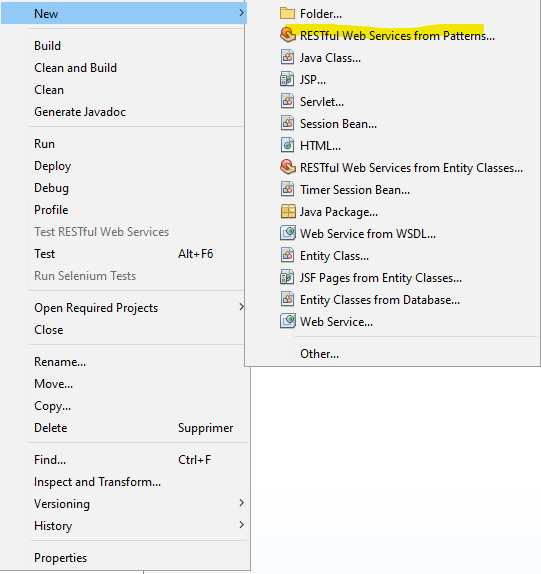
out.println("</html>");

On utilise request.getParameter() pour aller chercher les infos.

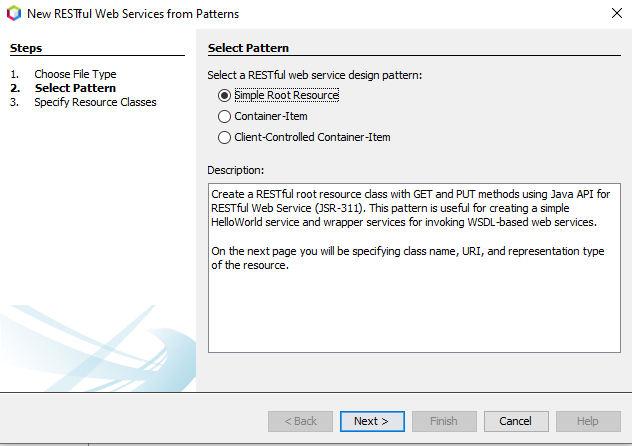
### Exercice 10

#### Serveur REST

Pour commencer, on créé un fichier de type RESTful Web Services from Patterns



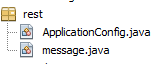
Dans la page suivante on laisse l’option Simple Root Ressource



Dans la page suivante on va y indiquer la nom du projet, la localisation, son package, le nom de sa classe et le path pour y accéder.



On peut voir que plusieurs fichiers ont été créé



Dans le fichier ApplicationConfig une seule ligne est utile, elle permet de spécifier le répertoire de toute nos application web du projet



Ensuite dans notre class message on va créer une méthode de type GET :

@GET

@Path("getMessage")

@Produces (javax.ws.rs.core.MediaType.TEXT\_PLAIN)

@Consumes(javax.ws.rs.core.MediaType.APPLICATION\_FORM\_URLENCODED)

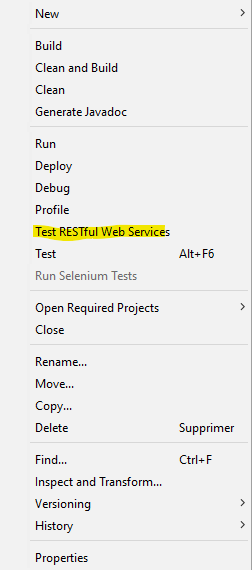
public String getMessage(){

return "Bonjour tout le monde !";

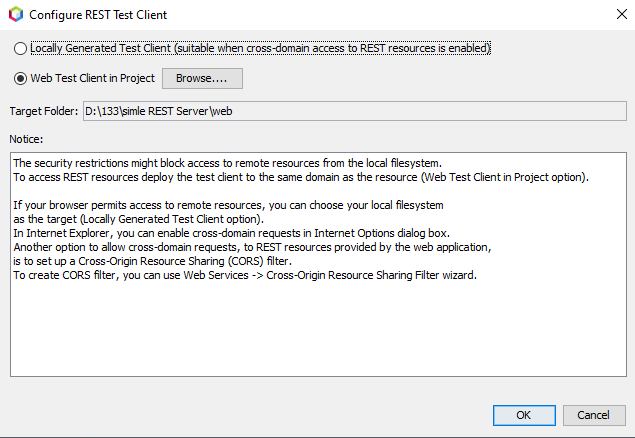
}

|  |  |
| --- | --- |
| @Path | Désigne l’emplacement pour accéder a l’application |
| @Produces | Si notre méthode ne renvoie rien en retour |
| @Consumes | Permet de spécifier le type MIME d’entrée de la méthode |

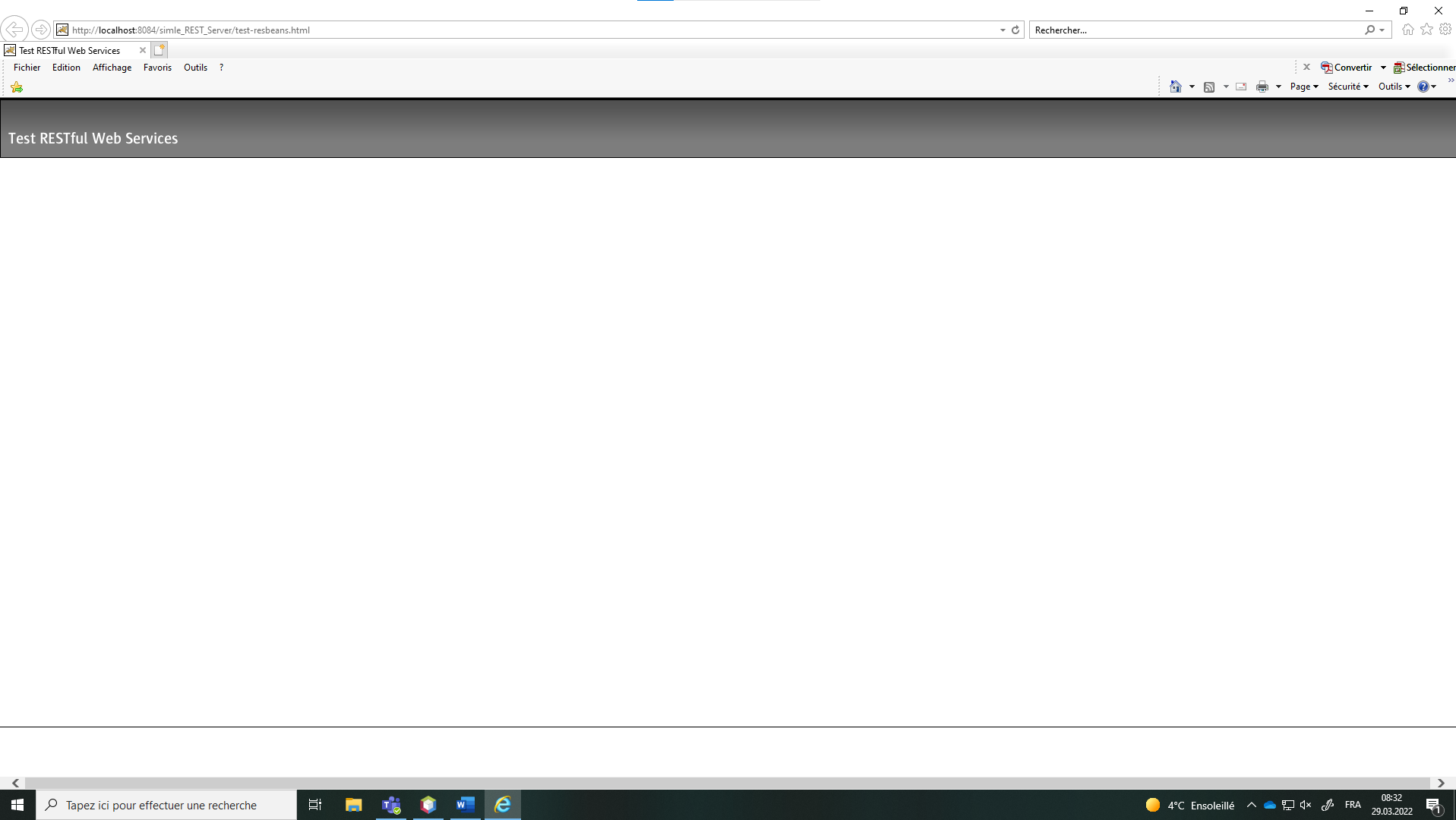
Test de l’application



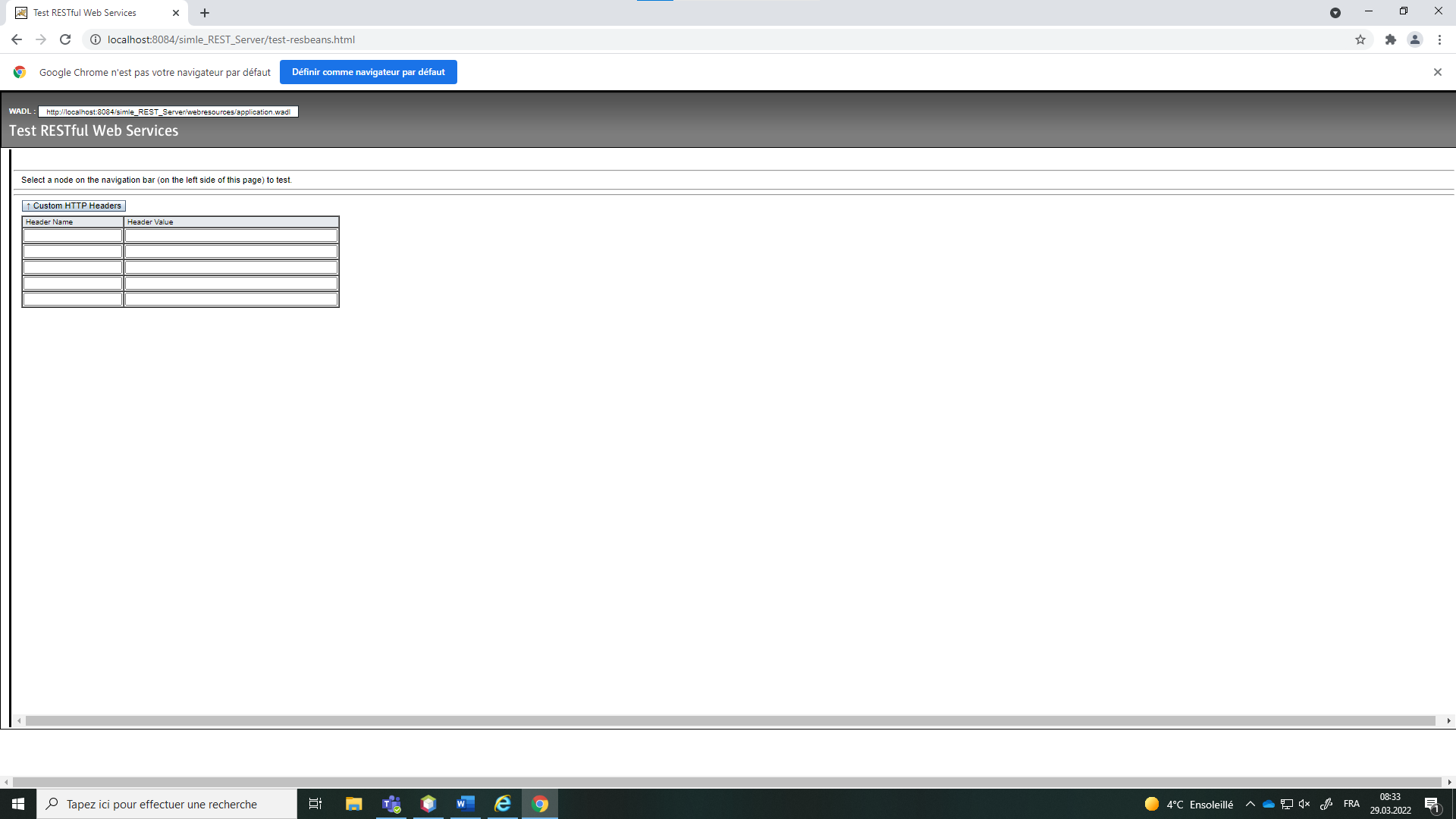
Ensuite on laisse les options par défaut



On arrive sur cette page



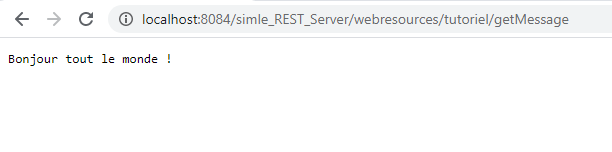
Mais internet explorer ne supporte pas très bien alors on passe sur chrome :



Si on veut accéder a notre méthode juste avec ce qu’on a mis dans le path ça ne vas pas marcher

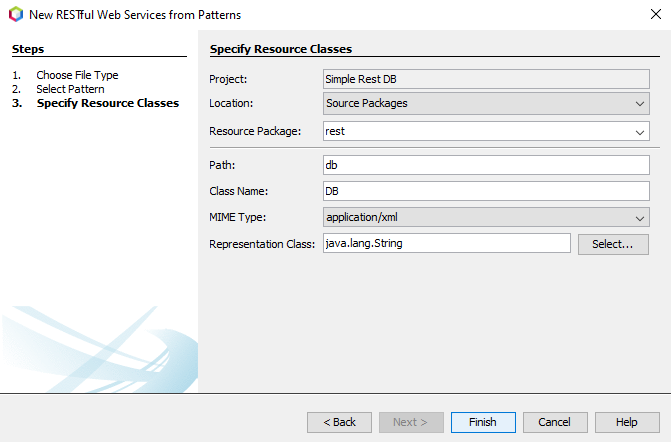


Du coup on doit bien spécifier le nom de la méthode

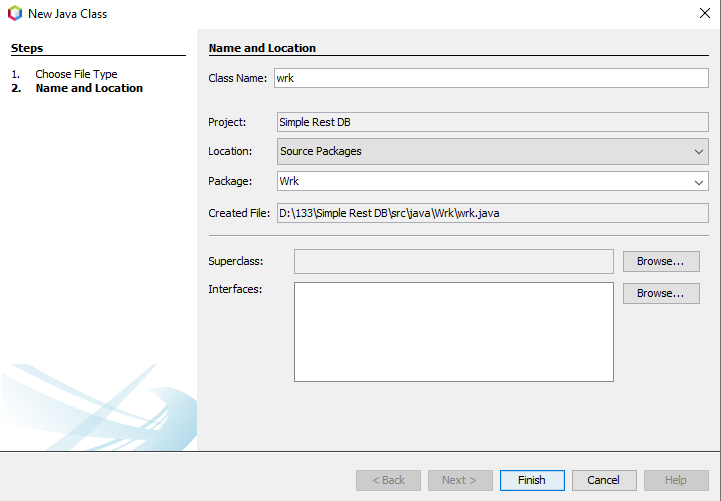


#### Création d’un server DB REST

On va refaire un fichier RESTful web Service from patterns



Ensuite on va créer une classe pour la gestion de la base de données



## *Auto-évaluations et conclusions*

### Apprentis dreosti

J’ai bien aimé ce module, pouvoir faire du web avec java est plutôt intéressant mais aussi très compliqué de mon point de vus que il y a beaucoup de matière nouvelle et dont on a pas l’habitude

### Apprentis Dubois

J’ai beaucoup apprécier se module, s’occuper d’une application Web plus complexe est vraiment sympathique mais tout autant compliqué. Je pense personnellement avoir beaucoup sur le fonctionnement des méthode http et sur le CSS. La matière était très diversifiée et compliqué mais compréhensible.