

Ma Première Application Web

par
David Nogent



Qu'est-ce qu'une application web ?

Une application web désigne un logiciel applicatif hébergé sur un serveur et accessible via un navigateur web. Contrairement à un logiciel traditionnel, l'utilisateur n'a pas besoin de l'installer sur son ordinateur. Il lui suffit de se connecter à n'importe quel navigateur web. Les technologies utilisées pour développer une application web sont les mêmes que celles employées dans la création de site internet.

Qu'est-ce que Java ?

Java est langage de programmation de haut niveau orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems.

Que sont les Java Server Pages ?

Les JSP sont une technologie Java qui permet la génération de pages web dynamiques. Elle permet de séparer la présentation sous forme de code HTML et les traitements écrits en Java.

Qu'est-ce qu'un site web dynamique ?

Un site web dynamique est un site dont les pages sont générées à la volée en fonction d'une demande de l'utilisateur. Le contenu d'une page web dynamique peut varier en fonction d'informations qui ne sont connues qu'au moment de sa consultation. A l'inverse, le contenu d'une page statique reste la même à chaque consultation.

Les sites web dynamique sont souvent associés à une base de données.

Qu'est-ce qu'une base de données ?

Une base de données est conteneur stockant des données telles que des chiffres, des dates, des mots, pouvant être retraités par des moyens informatiques pour produire une information.

Pour concevoir notre première application web, nous aurons besoin :

- **JavaSE Development Kit**
 - o [Java Downloads | Oracle](#)
- **Serveur Tomcat**
 - o [Apache Tomcat® - Apache Tomcat 10 Software Downloads](#)
Je vous suggère de télécharger la version « installer » qui vous permettra de configurer Tomcat de manière automatique.
- **MySQL 8** (serveur de base de données)
 - o [MySQL :: Download MySQL Community Server](#)
 - o [MySQL :: Download Connector/J](#)
- Le Bloc-notes de Windows ou un IDE comme Netbeans ou Eclipse.

Dans un premier temps, nous allons concevoir la première page de notre application web. Il s'agira qu'une page HTML sur laquelle nous allons mettre un formulaire permettant à un utilisateur de se connecter. Le nom de cette page sera index.html

Nous y ferons figurer une image, notre formulaire de connexion et un lien vers une page pour créer un compte.

Dans un second temps, nous allons créer une base de données sur MySQL pour le système de compte.

Dans un troisième temps, nous ferons quelques manipulations dans les dossiers de Tomcat.

Vous êtes prêts ? Alors, ouvrons notre bloc-notes.

PREMIERE PARTIE :

CONCEPTION D'UNE APPLICATION WEB

PAS A PAS



Dans un premier temps, nous n'allons pas nous préoccuper de la mise en page.

index.html

```
<html>
<head>
<title>My 1st Java Application</title>
</head>
<body>
<p></p> // insertion d'une image
<div>
<form action="connexion.jsp" method="post"> // formulaire qui va envoyer les saisies vers une page JSP appeler connexion.
<p>LOGIN NAME : <input type="text" name="name" /></p> // saisie du nom de connexion
<p>PASSWORD : <input type="password" name="pw" /></p> // saisie du mot de passe
<p><input type="submit" value="CONNEXION" /></p>
</form>
</div>
</body>
</html>
```

Sauvegardons ce petit travail dans un document que nous allons appeler « index.html ». Mettons-le pour le moment sur le bureau windows ou dans Mes Documents.

Maintenant, allons dans le dossier du serveur Tomcat.

Nom	Modifié le	Type	Taille
bin	24/12/2023 23:15	Dossier de fichiers	
conf	24/12/2023 23:16	Dossier de fichiers	
javadoc	26/12/2023 16:02	Dossier de fichiers	
lib	26/12/2023 16:02	Dossier de fichiers	
logs	26/12/2023 15:22	Dossier de fichiers	
temp	24/12/2023 23:15	Dossier de fichiers	
tld	26/12/2023 16:02	Dossier de fichiers	
webapps	26/12/2023 15:23	Dossier de fichiers	
work	24/12/2023 23:16	Dossier de fichiers	
LICENSE	08/12/2023 00:31	Fichier	61 Ko
NOTICE	08/12/2023 00:31	Fichier	3 Ko
RELEASE-NOTES	08/12/2023 00:31	Fichier	7 Ko
tomcat	08/12/2023 00:31	Fichier ICO	22 Ko
Uninstall	08/12/2023 00:31	Application	86 Ko

Voici mon installation. Si vous avez laissé l'installateur créer le chemin par défaut, votre installation est différente. Pour éviter, les confusions, j'appellerai le dossier principal de Tomcat TOMCAT_HOME.

Si votre dossier principal ressemble à mon image (avec éventuellement un ou deux dossiers en moins), alors vous avez trouvé votre dossier TOMCAT_HOME.

Dans le dossier « lib » de TOMCAT_HOME, placez le fichier **mysql-connector-j-8.2.0.jar** que vous avez téléchargé.

Une fois ce fichier placé, redémarrer le serveur si ce dernier n'est pas éteint.

Allons, maintenant, dans le dossier « webapps » de TOMCAT_HOME. C'est le dossier qui contient toutes nos applications webs.

Dans ce dossier, créez le dossier « My1stJavaApp » et placez-y votre fichier « index.html ». (faites un copier-coller)

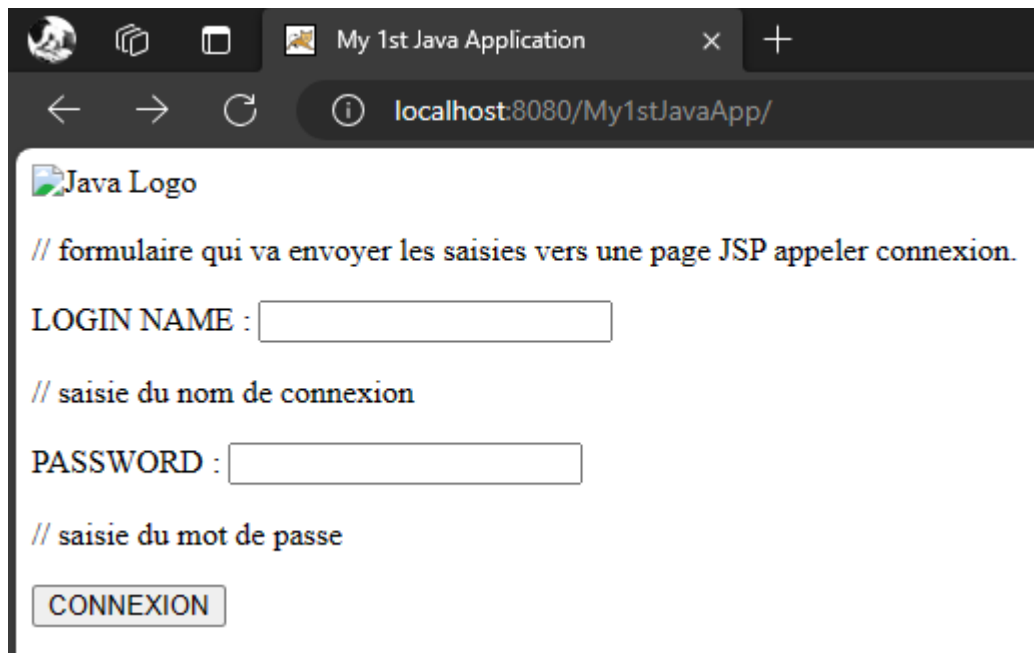
Tapez en suite dans votre navigateur web : <http://localhost:8080/My1stJavaApp> et voyez ce qui se passe.

A ce stade, le serveur nous sort un message d'erreur. La raison ? Il manque un dossier important. Ce qui caractérise une application web, c'est la présence, dans le dossier de notre application, d'un dossier « WEB-INF ». Il est même probable que Tomcat ait supprimé le dossier My1stJavaApp.

Désactivez le serveur Tomcat. Dans le dossier « webapps », recréez le dossier « My1stJavaApp » et dans ce dernier, créez le dossier « WEB-INF ».

Dans « My1stJavaApp », enregistrez notre page « index.html » et retapez <http://localhost:8080/My1stJavaApp>

Voici, ce que nous obtenons :



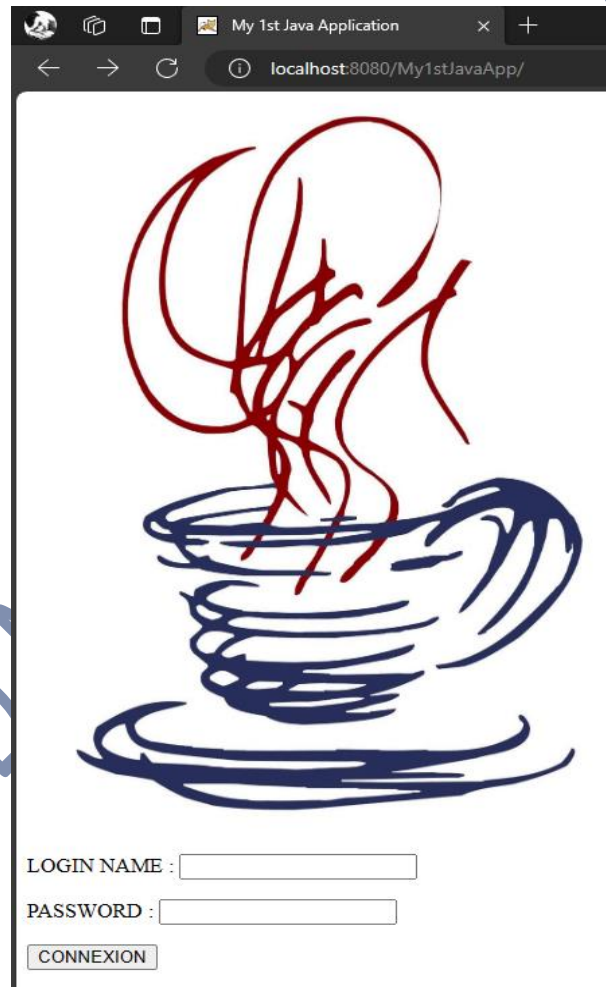
The screenshot shows a web browser window with the title 'My 1st Java Application'. The address bar displays 'localhost:8080/My1stJavaApp/'. The page content includes a 'Java Logo' icon, a comment '// formulaire qui va envoyer les saisies vers une page JSP appeler connexion.', a 'LOGIN NAME :' label followed by a text input field, a comment '// saisie du nom de connexion', a 'PASSWORD :' label followed by a text input field, a comment '// saisie du mot de passe', and a 'CONNEXION' button.

Il nous faut ajouter une image et cacher mes commentaires. Dans « My1stJavaApp », mettez une image qui se nomme « java.png »

Dans « index.html », remplacez « `` » par « `` »

Là où j'ai mis des commentaires avec //, mettez `<!--` en début de commentaire et `-->` en fin de commentaire.

Et maintenant, voici un résultat d'affichage convenable.



Notre code **index.html** ressemble maintenant à

```
<html>
<head>
<title>My 1st Java Application</title>
</head>
<body>
<p></p> <!-- insertion d'une image -->
<div>
<form action="connexion.jsp" method="post"> <!-- formulaire qui va envoyer les saisies vers une page JSP appeler
connexion. -->
<p>LOGIN NAME : <input type="text" name="name" /></p> <!-- saisie du nom de connexion -->
<p>PASSWORD : <input type="password" name="pw" /></p> <!-- saisie du mot de passe -->
<p><input type="submit" value="CONNEXION" /></p> <!-- bouton de validation du formulaire -->
</form>
</div>
</body>
</html>
```

Maintenant, nous allons devoir créer notre base de données pour la suite du développement.

Si vous n'avez jamais utilisé MySQL, ni étudié le langage SQL, je vous recommande la lecture des deux publications « **Installation de MySQL 8** » et « **Introduction au langage SQL par la pratique 2.0** ».

Dans le bloc-notes, recopiez les instructions suivantes et enregistrez-les dans un fichier « my_java_app.sql ».

```
DROP DATABASE IF EXISTS my_java_app;
CREATE DATABASE my_java_app;
USE my_java_app;

CREATE TABLE utilisateurs (refUser INT AUTO_INCREMENT PRIMARY KEY,
                           nomUser VARCHAR(120),
                           prenomUser VARCHAR(120),
                           emailUser VARCHAR(120),
                           passwordUser VARCHAR(120),
                           adresseUser VARCHAR(250),
                           codepostalUser VARCHAR(10),
                           villeUser VARCHAR(200));
```

La connexion à MySQL se fera avec l'utilisateur root et le mot de passe défini lors de l'installation, sur le port 3307 sur ma machine.

Connectez-vous à MySQL via la console windows et exécutez le fichier « **my_java_app.sql** ».

Pour exécuter le fichier, tapez : **source my_java_app.sql**

(lire ou relire le tutoriel « **Introduction au langage SQL par la pratique 2.0** » pour voir les détails de cette partie)

Maintenant que nous avons créé notre base de données, nous allons procéder à l'écriture d'une nouvelle page HTML qui va contenir un nouveau formulaire pour créer un nouvel utilisateur et une page JSP qui va établir la connexion avec la base de données pour insérer les données concernant ce nouvel utilisateur. Une fois que ces pages seront opérationnelles, nous insérerons un lien sur notre page index vers cette nouvelle page HTML.

Appelons cette nouvelle page « **nouvel_utilisateur.html** » et la page JSP « **enregistrement_utilisateur.jsp** ».

nouvel_utilisateur.html

```
<html><head><title>My 1st Java Application</title></head>
<body>
<div>
<table><tr><td></td>
<td>
<form action="enregistrement_utilisateur.jsp" method="post"> <!-- formulaire qui va envoyer les saisies vers une page
JSP appeler enregistrement_utilisateur. -->
<p>NOM : <input type="text" name="nom" /></p> <!-- saisie du nom de famille -->
<p>PR&Eacute;NOM : <input type="text" name="prenom" /></p> <!-- saisie du prénom -->
<p>COURRIEL : <input type="text" name="email" /></p> <!-- saisie du courriel -->
<p>MOT DE PASSE : <input type="password" name="pw" /></p> <!-- saisie du mot de passe -->
<p>RUE : <input type="text" name="address" /></p> <!-- saisie de la rue -->
<p>CODE POSTAL : <input type="text" name="cp" /></p> <!-- saisie du code postal -->
<p>VILLE : <input type="texte" name="ville" /></p> <!-- saisie de la ville -->
<p><input type="submit" value="CR&Eacute;ER UN COMPTE" /></p> <!-- bouton de validation du formulaire -->
</form></td></tr></table> <!-- nous avons insérer notre image et le formulaire dans un tableau -->
</div>
</body>
</html>
```

enregistrement_utilisateur.jsp

Nous allons procéder à l'écriture de notre page JSP en plusieurs étapes.

D'abord, nous posons la base comme s'il s'agissait qu'une page HTML :

Entre les balises « **<body>** » et « **</body>** », nous allons mettre les balises « **<%** » et « **%>** » qui contiendront nos scripts Java.

Avant de commencer à envisager l'enregistrement de notre nouvel utilisateur, nous allons dans un premier temps récupérer les données saisies par le formulaire et l'afficher dans notre navigateur.

Pour récupérer le nom saisi dans le formulaire, nous taperons la ligne suivante :

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

// ici, nous récupérons le contenu saisi dans le formulaire à l'aide de **request.getParameter("nom_du_champ_saisi")** et on place la valeur récupérée dans une variable « nom » qui est une chaîne de caractères (string=chaîne)

et, pour afficher le contenu sauvegarder dans la variable nom, tapons la ligne :

out.println(nom) ;

Avant d'aller à la page suivante pour voir le code complet de la page, essayez d'écrire les lignes pour les autres variables (prénom, adresse, etc.)

```
<html>
<head>
<title>My 1st Java Application</title>
</head>
<body>

</body>
</html>
```

```
<html>
<head>
<title>My 1st Java Application</title>
</head>
<body>
<%
//récupération des valeurs du formulaire
String nom = request.getParameter("nom");
out.println(nom);
%>
</body>
</html>
```

Voici le résultat de notre petit travail :

```
<html>
<head>
<title>My 1st Java Application</title>
</head>
<body>
<%

// récupération des valeurs du formulaire que l'on place dans des variables
String nom = request.getParameter("nom");
String prenom = request.getParameter("prenom");
String email = request.getParameter("email");
String password = request.getParameter("pw");
String adresse = request.getParameter("adresse");
String cp = request.getParameter("cp");
String ville = request.getParameter("ville");

// affichage du contenu des variables sur ma page JSP
out.println(nom);
out.println(prenom);
out.println(email);
out.println(password);
out.println(adresse);
out.println(cp);
out.println(ville);

%>
</body>
</html>
```

Notre code affiche les données sur une seule ligne

Maintenant, nous allons passer à la seconde phase de notre page « **enregistrement_utilisateur.jsp** », c'est-à-dire l'enregistrement des données dans notre base de données « **my_java_app.sql** ».

D'abord, supprimons nos lignes « **out.println()** ; » qui ne nous sont plus utiles à ce stade.

Maintenant, nous devons établir la connexion avec la base de données. Pour cela, nous chargeons un pilote permettant la connexion (c'est le fameux fichier **mysql-connector-j-8.2.0.jar** que je vous ai fait mettre dans le dossier « **lib** » de **TOMCAT_HOME**).

Et pour cela, nous tapons les ligne suivantes :

```
// chargement du pilote JDBC pour mysql :
Class.forName("com.mysql.jdbc.Driver");
// ouverture de la connexion avec mysql sur le port 3307, mettez 3306 si port par défaut, remplacez MyJavaApp par votre MySQL login et password :
java.sql.Connection cnx = java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3307/my_java_app?user=MyJavaApp&password=MyJavaApp");
// preparation et execution de la requete SQL :
java.sql.Statement req;

// définition de la requête
String requete = "INSERT INTO utilisateurs VALUES(" + null + ", '" + nom + "', '" + prenom + "', '" + email + "', '" + password + "', '" + adresse + "', '" + cp + "', '" + ville + "')";

req = cnx.createStatement(); // initiation de la requête
req.executeUpdate(requete); // exécution de la requête
req.close(); // fermeture de la requête
cnx.close(); // fermeture de la connexion à la base de données
```

Et pour ne pas avoir une page blanche en guise de résultat, nous ajoutons ces quelques lignes pour avoir un affichage

```
out.println("BRAVO ");
out.println(prenom);
out.println(" !");
out.println("<br />");
out.println("Votre compte a &eacute;t&eacute; parfaitement cr&eacute;");
out.println("<br />");
out.println("Vous allez &ecirc;tre redirig&eacute; vers la page d'accueil pour vous connecter");
out.println("<meta http-equiv='refresh' content='10; url=index.html' />");
```

Comme nous avons ajouté le pilote dans le dossier « lib » de TOMCAT_HOME, notre connexion a fonctionné et l'insertion des données a pu se faire. Toutefois, il peut être utile de placer notre pilote **mysql-connector-j-8.2.0.jar** dans un sous-dossier « lib » du dossier **WEB-INF** de notre application web. Imaginons que vous devez la déployer sur un autre serveur, cela pourrait être fort utile.

ATTENTION, je tiens à souligner que pour le moment, nous concevons des pages pas à pas. Elles ne sont pas définitives. Mon but est de vous aider à vous familiariser avec le développement web qui est un processus long.

Nous essayons dans un premier de mettre en application nos idées de manière simple.

Une fois que nos idées seront palpables. Nous améliorerons le code et peaufinerons une mise en page sympathique.

Dans notre page « **index.html** », nous allons placer un lien vers la page « **nouvel_utilisateur.html** » en dessous du formulaire (après la balise **</form>**).

**
**

Création d'un nouveau compte

Dans notre formulaire, nous allons changer ce qui concerne le « login ». Nous utiliserons l'adresse mail pour nous connecter.

Remplacez donc la ligne

<p>LOGIN NAME : <input type="text" name="name" /></p> <!-- saisie du nom de connexion -->

par

<p>EMAIL : <input type="text" name="email" /></p> <!-- saisie de l'email de connexion -->

puis sauvegarder.

Maintenant, nous allons créer la page « **connexion.jsp** » et une page « **mon_compte.jsp** »

Dans la page « **connexion.jsp** », nous allons établir une connexion à MySQL, et vérifiez que les données saisies sont identiques à celles figurant dans la base de données. Si oui, nous autoriserons la connexion et nous redirigerons l'utilisateur vers la page « mon_compte.jsp ». Dans le cas contraire, la connexion sera refusée et l'utilisateur sera renvoyé vers la page d'accueil.

Entre les balises « **<%** » et « **%>** », nous insérons le code suivant :

// récupération des valeurs du formulaire de la page index.html que l'on place dans des variables

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

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

// chargement du pilote JDBC pour mysql

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

// ouverture de la connexion avec mysql sur le port 3307

java.sql.Connection cnx =

java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3307/my_java_app?user=MyJavaApp&password=MyJavaApp");

// preparation et execution de la requete SQL

java.sql.Statement req = cnx.createStatement();

java.sql.ResultSet rs = req.executeQuery("SELECT * FROM utilisateurs");

while(rs.next()){ // tant qu'il y a des données dans la base faire ce qui suit

// place les données de la base correspondantes à Email et Password dans des variables

String db_login = rs.getString("emailUser");

String db_password = rs.getString("passwordUser");

String db_prenom = rs.getString("prenomUser");

// procède à des vérifications

if (email == db_login || db_login.equals(email))

// condition 1 : si email saisi = email base de données faire...


```
{  
    if (password == db_password || db_password.equals(password))  
        // condition 2 : si password saisi = password base de données faire...  
        {  
            out.print("<meta http-equiv='refresh' content='10; url=mon_compte.jsp' />");  
            out.print("Bonjour ");  
            out.print(db_prenom); // la connexion réussie, on salut l'utilisateur  
            out.print(" !");  
            out.print("<br />");  
            out.print("<br />");  
            out.print("Vous allez être redirigé vers la page principale<br />");  
            out.print("dans quelques secondes.<br />");  
            out.print("Veuillez patienter...");  
        }  
        else { // si condition 2 non remplie faire...  
            out.print("ACCÈS REFUSÉ, SERVICE");  
            out.print("<br/><br/>");  
            out.print("Vous devez entrer votre email et votre mot de passe<br/>");  
            out.print("pour accéder à cette partie!<br/>");  
            out.print("<a href='index.html'>Revenir en arrière.</a>");  
        }  
    }  
}
```

```
    }  
}  
else { // si condition 1 non remplie faire...  
out.print("ACC&Egrave;S R&Eacute;SERV&Eacute;");  
out.print("<br/><br/>");  
out.print("Vous devez entrer votre email et votre mot de passe<br/>");  
out.print("pour acc&egrave;der à cette partie!<br/>");  
out.print("<a href='index.html'>Revenir en arri&egrave;re.</a>");  
}  
} // fin de while  
// fermeture de la connexion et des requêtes  
rs.close();  
req.close();  
cnx.close();
```

Maintenant, nous allons écrire la page « **mon_compte.jsp** ». Elle sera similaire à la page « **connexion.jsp** » mais nous n'allons pas effectuer de vérification. Nous allons juste récupérer les données contenues dans la base et les afficher car pour le moment, je n'ai aucune idée de ce qu'il adviendra de cette page par la suite.

Nous reprenons le même code mais allons changer ce qui suit le « **while** », tout le reste sera identique, du moins, pour le moment.

Voici la partie que nous changeons pour faire notre page « **mon_compte.jsp** ».

```
while(rs.next()){ // tant qu'il y a des données dans la base faire ce qui suit
// placer les données de la base correspondantes à Email et Password dans des variables
String db_login = rs.getString("emailUser");
String db_password = rs.getString("passwordUser");
String db_prenom = rs.getString("prenomUser");
String db_nom = rs.getString("nomUser");
String db_ref = rs.getString("refUser");

out.println("Bienvenue sur votre page principale");
out.println("Vous &ecirc;tes l'utilisateur #");
out.println(db_ref);
out.println(".");
out.println("<br />");
out.println(db_nom);
out.println(" ");
out.println(db_prenom);

} // fin de while
```

Au cas, où vous n'auriez pas tout suivi dans le processus d'écriture, regardez les codes sources de notre application figurant dans le fichier « **My1stJavaApp1.0.zip** » et comparer avec ce que je vous ai demandé de faire tout au long de ces pages.

Pour le moment, le code est simpliste. Ne faut-il pas commencer par de petites choses pour ensuite attaquer des choses plus grandes, difficiles ou plus complexes ?

Ici, s'achève donc la version 1.0 de notre application. Dans les prochaines pages, nous allons modifier le code pour lui donner un design si je peux dire les choses ainsi. N'hésitez pas à refaire cette application à votre sauce, avec votre propre base de données, vos propres variables. C'est ainsi que vous vous approprierez les connaissances liées au développement web.

DEUXIEME PARTIE :

MISE EN PAGE, DESIGN ET AMELIORATION DU CODE

Présentation des feuilles de style et de JavaScript





Voici notre page d'accueil « **index.html** » avec un nouveau design.

Pour cela, nous avons utilisé la technologie des feuilles de style. Nous avons également amélioré le code source avec JavaScript...

Cette version 1.1 de la page d'accueil a été complètement repensée.

J'ai imbriqué notre formulaire dans un jeu de tableaux.

En HTML, on crée un tableau de la façon suivante :

```
<table> <!-- début du tableau -->
<tr> <!-- début de la première ligne -->
<td>colonne 1</td><td>colonne 2</td>
</tr> <!-- fin de la première ligne -->
<tr> <!-- début de la seconde ligne -->
<td>colonne 1</td><td>colonne 2</td>
</tr> <!-- fin de la seconde ligne -->
<tr>
<td colspan="2">colonne 1 + colonne 2 fusionnées</td>
</tr>
</table> <!-- fin du tableau -->
```

Bien entendu, vous l'avez compris, notre tableau ne va pas contenir des données mais des éléments permettant la mise en page.

Schématiquement, cela donne :

TITRE DU PROJET

Mon tableau principal :

```
<table>
<tr>
<td> notre image java </td> <td> insertion d'un second tableau dans notre tableau principal
</td>
</tr>
</table>
<table>
<tr>
<td> il s'agit de mettre en forme avec un tableau le formulaire du connexion
</td>
</tr>
</table>
```

C'est assez simpliste mais ça vous donne une bonne idée de ce que j'ai fait pour mettre en place certains éléments.

On peut également donner des noms à un tableau avec `<fieldset><legend>Mon tableau</legend></fieldset>`

Exemple :

```
<fieldset>
<legend>TITRE DE MON TABLEAU</legend>
<table>
<tr>
<td></td>
<tr>
</table>
</fieldset>
```

Mise à part quelques « **align="center"** » (ou bien **left** ou **right**), tout le reste du design recourt aux technologies des feuilles de style.

Une feuille de style (ou **Cascade Style Sheet**) est un document *.css qui ressemble à cela :

```
body {
toutes les données mettant en page le corps principal de la page
}

p {
toutes les données mettant en page ce qui est placé dans une balise <p></p>
}

#titre {
ici, il s'agit d'un élément personnalisé par exemple <p id="titre"> </p>
}
```

On peut faire énormément de chose avec les feuilles de style dès lors que l'on prend l'habitude de s'en servir et d'exploiter leurs possibilités.

a { } va s'occuper de mettre du style sur ****

a : hover { } va s'occuper de changer le style de **** lorsque vous passez la souris au-dessus du lien

body {

font-family: Gotham, "Helvetica Neue", Helvetica, Arial, sans-serif; /* style de police */

color:#FFF; /* couleur de la police */

font-size:18px; /* taille de la police */

font-weight:bold; /* caractère gras */

background:url(..images/background.png); /* image d'arrière plan */

background-color:#3CF; /* couleur de l'arrière plan */

font-variant:small-caps; /* caractère en lettre capitale */

}

Lorsque la feuille de style est créée, on l'appelle dans le document HTML avec la balise **<LINK>** que l'on place entre les balises **<head>** et **</head>** :

<link rel="stylesheet" type="text/css" href="css/my_style_app.css" media="screen" />

On peut utiliser le même document ***.css** pour toute l'application web ou faire un document ***.css** pour chaque page, tout dépend de ce que vous avez envie de faire.

Comme vous le savez, notre page d'accueil est un formulaire de connexion vers une page JSP. Si notre page JSP contrôle les données du formulaire, il est bon en développement web de contrôler un formulaire avant que soit envoyée la requête. Et pour cela, on utilise du JavaScript ! Comme cela, on évite d'envoyer la requête inutilement si le formulaire est vide.

On peut mettre notre script de contrôle soit dans la page HTML, soit comme pour la feuille de style, créer un document *.js qui va contenir le code JavaScript.

```
function RecommandationW3C() {  
var email = document.forms["Connexion"]["email"]; // récupération des données du formulaire que l'on place dans des variables  
var pw = document.forms["Connexion"]["pw"];  
if (email.value == "") { // contrôle de l'email  
alert("Veuillez saisir votre adresse email !");  
email.focus();  
return false;  
}  
if (email.value.indexOf("@", 0) < 0) {  
alert("Mettez une adresse email valide !");  
email.focus();  
return false;  
}  
if (email.value.indexOf(".", 0) < 0) {  
alert("Mettez une adresse email valide !");  
email.focus();  
return false;  
}  
  
if (pw.value == "") { // contrôle du mot de passe  
alert("Veuillez saisir votre mot de passe !");  
pw.focus();  
return false;  
}  
return true;  
}
```

Pour importer notre script dans notre page, on le fait à l'aide de la balise **<SCRIPT>** :

```
<script type="text/javascript" src="javascript/index_connexion_control.js"></script>
```

que l'on place à la fin du code HTML juste avant la balise **</body>**.

Il faut néanmoins modifier un peu notre code HTML au niveau de la balise **<FORM>** :

```
<form name="Connexion" action="connexion.jsp" onsubmit="return RecommendationW3C()" method="post">
```

Maintenant, vous avez tous les éléments pour exécuter notre petit script.

Les recommandations du World Wide Web Consortium !

Le W3C est un organisme de standardisation, fondé en 1994, chargé de promouvoir la compatibilité des technologies du net : HTML, XHTML, XML, PNG, SVG etc... Le Consortium compte 452 membres au 1^{er} octobre 2021.

La page « **index.html** » et notre feuille de style ont été vérifiées sur réciproquement sur :

[The W3C Markup Validation Service](#) pour le HTML

et [Service de validation CSS du W3C](#) pour la feuille CSS

Maintenant, il nous faut faire la même chose avec les autres pages HTML de l'application et personnalisé les pages JSP que nous allons voir dans les pages qui suivent.

Modifier les autres pages va aller plus vite que la première grâce aux feuilles de style.



Voici ce que donne la page « **connexion.jsp** ». L'essentiel du travail provient de la feuille de style. De quelques copiés-collés ici et là. Très peu de modifications de la partie écrite en java. On a copié le code HTML de la page « index.html » (titre, copyright) et on a placé les messages du code java dans des balises « **<p>** » qui sont pris en charge par notre document CSS.

```
out.print("<p id='message'>");
```

// les lignes qui étaient déjà écrites (on corrige éventuellement les fautes de frappes ou de code au passage)

```
out.print("</p>");
```

et dans la feuille de style, on ajoute les lignes suivante :

```
#message {  
    text-align:center;  
}
```

Pour la page « nouvel_utilisateur.html », je suggère de procéder comme pour la page « **index.html** » et pour la page « **enregistrement_utilisateur.jsp** », il faudra procéder comme pour la page « **connexion.jsp** ».

Mais, j'ai eu une petite idée qui me pousse à modifier la base de données. En effet, même si, pour le moment, je ne sais pas ce que je vais faire de ce projet d'application qui me sert pour le moment de tutoriel, je me suis dit que ce serait sympa d'enregistrer la date de création du compte de l'utilisateur.

Pour cela, nous allons ajouter une colonne à la table « **utilisateurs** » de notre base de données de cette manière :

```
mysql> ALTER TABLE utilisateurs ADD date VARCHAR(20);  
Query OK, 0 rows affected (0.60 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

Inutile de faire des modifications à la page HTML qui va créer le compte car c'est la page JSP qui va se charger de nous donner la date du jour et de l'enregistrer dans la base en même temps qu'elle va enregistrer les données de l'utilisateur.

Ajoutons d'abord la nouvelle variable dans la requête de la page :

```
String requete = "INSERT INTO utilisateurs VALUES(" + null + ", " + nom + ", " + prenom + ", " + email + ", " + password + ", " + adresse + ", " + cp + ", " + ville + ", " + aujourd8 + ")";
```

Pour que notre page JSP puisse afficher la date du jour, nous allons utiliser les classes « **Date** » et « **SimpleDateFormat** ». La première se charge d'obtenir la date. La seconde va nous la donner dans un format souhaité.

Au-dessus de la balise « **<HTML>** », nous écrivons la ligne suivante :

```
<%@page language="java" import="java.util.Date" import="java.text.SimpleDateFormat" %>
```

Grâce à cette ligne, nous faisons savoir à notre programme java qu'il a besoin de ces objets, et il va les chercher dans JAVA_HOME pour pouvoir les manipuler.

Il nous reste plus qu'à écrire ces quelques lignes au début du code java (après la balise « **<%>** ») :

```
SimpleDateFormat today = new SimpleDateFormat("dd/MM/yy HH:mm:ss"); // nous définissons un format pour la date
```

```
Date date = new java.util.Date(); // récupération de la date du jour
```

```
String aujourd8 = today.format(date); // on applique la format demandé à la date du jour et on place le résultat dans une variable
```

Sur le formulaire de la page « **nouvel_utilisateur.html** », il faudra appliquer également le code javascript en l'adaptant au formulaire.

Pour que la création d'un nouveau compte soit parfaite, il faudra empêcher l'utilisation d'un mail préexistant dans la base de données sinon, notre programme, tel que nous l'avons écrit va chercher toutes les occurrences du mail en question et du coup, nous avons des incohérences durant la connexion.

Pour ce faire, nous allons utiliser un argument qu'on retrouver très souvent dans les servlets et autres programmes java.

Après avoir récupéré les valeurs saisies dans le formulaire, nous écrivons

try {

On se connecte à la base de données à l'aide d'un **SELECT**

à l'aide de **if (rs.next()) { }** si la valeur de l'email existe alors on va afficher un message informant de la préexistence de ce mail dans notre base **else { }** sinon, nous allons procéder à une autre connexion à la base de données pour créer le nouveau compte

}

par contre, nous devons ajouter un autre argument pour éviter d'avoir un message d'erreur

finally {

Java exige la présence de **finally** si on utilise un **try** mais nous ne sommes pas obligés de nous en servir

}

Maintenant, je vous invite à regarder les codes sources de l'archive « **MyJavaApp1.1.zip** » afin de voir en détail les modifications apportées aux pages.

Le but de ce petit tutoriel n'est pas de vous apprendre à développer en Java mais de vous donner des clés pour bien débuter dans la conception d'une application web.

Le code que je vous ai présenté est avant tout simple. Mais, il peut être amélioré. On peut ajouter une gestion des erreurs (que je ne n'ai voulu ajouter pour ne pas trop vous gaver de connaissance).

Dans tous les cas, vous avez maintenant de quoi faire vos premiers pas dans le développement web.