Objectifs

1. Paramétrer et installer l'environnement nécessaire à l'exécution de Servlets.
2. Ecrire et tester des servlets rudimentaires.
3. Faire le lien entre une servlet et une base de données.

1 Installation de l'environnement

Vous êtes déjà en possession dans votre répertoire personnel d'un serveur tomcat10 (Core). Il s'agit du serveur Java qui nous sera utile pour faire du développement web.
Notez également l'emplacement et la version du `JDK` installé, généralement dans `/usr/lib/jvm`. Nous supposerons que vous avez installé une version 17 ou supérieure. Afin de vérifier la bonne installation vous pouvez ouvrir une console est exécuter `java -version`.

Vérifiez aussi que vous êtes en mesure d'appeler le compilateur en exécutant `javac -version`. Si ce n'est pas le cas, vous devez aussi configurer la variable `PATH` de votre système.

Vérifiez les variables d'environnement suivantes. Elles doivent contenir les chemins permettant d'accéder à toutes les classes et commandes nécessaires à la compilation.

— `CLASSPATH` : pour faciliter la compilation des servlets avec `javac`

— `JAVA_HOME` : qui sera utile au serveur web Tomcat (en général déjà configurée)

Si ce n'est pas déjà fait, configurez correctement deux variables d'environnement :

```
export CLASSPATH=.:~/tomcat/lib/servlet-api.jar:~/jars/postgresql-42.6.0.jar
export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64
```

1. Récupérez le fichier `vide.zip` et décompressez le dans le répertoire `webapps` de Tomcat. Il s'agit d'un contexte web basique.
2. Les scripts de lancement de Tomcat se trouvent dans `tomcat/bin`. Les fichiers `.bat` sont utilisés sous Windows alors que les fichiers `.sh` sont utilisés sous linux et MacOs. Il y en a deux scripts différents :
   (a) `startup.sh` lancera le serveur en tache de fond et placera les erreurs dans un fichier de log.
   (b) `catalina.sh run` lancera le serveur en avant plan affichant les erreur dans la console.
   Démarrez votre serveur Tomcat. Dans sa configuration standard ce serveur utilise le port 8080. Il faut donc s'assurer qu'aucun autre programme sur votre système n'utilise déjà ce port.
3. Si tout s'est bien passé, l'environnement nécessaire est maintenant configuré.

2 Test de la configuration

Lancez Firefox (ou autres) et accédez à votre serveur à l'adresse `http://localhost:8080`. La page de de garde de Tomcat doit s'afficher correctement.

Attention : Si vous utiliser un proxy (c'est le cas à l'université de Lille), il faudra soit le désactiver dans votre navigateur, soit indiquer que vous ne l'utilisez pas pour `localhost` dans `General/Paramètres réseau`.

3 Ma première page HTML

Dans le répertoire `tomcat/webapps/vide` créez une page HTML nommée `essai.html`.

```
<HTML>
<HEAD>
  <META http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
  <TITLE>Essai</TITLE>
</HEAD>
<BODY> Voici ma première page sur mon propre serveur </BODY>
</HTML>
```

1. Vous pouvez y accéder[1] par l'URL `http://localhost:8080/vide/essai.html`
2. Remplacez `UTF_8` par `Latin1` et rechargez la page. Que se passe til ?

Pour le moment, nous ne faisons que transmettre un fichier au navigateur. Voyons comment exécuter du code java à l'intérieur de Tomcat.

4 Ma première compilation

L'archive qui vous a été donnée contient une page `Test.html` qui appelle une servlet `Test.java`. Une erreur s'est glissée dans chacun de ces programmes. Faites en sorte que tous les liens fonctionnent correctement.

Ce test n'est pas anodin ! Si vous êtes arrivés ici, c'est que le serveur web, le moteur de servlets, le réseau, le proxy, les variables d'environnement système sont tous bien réglés et que vous savez compiler et exécuter une Servlet !

5 Création de ma propre servlet

Créez dans le répertoire `vide/WEB-INF/classes` la servlet `First.java`

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;

@WebServlet("/servlet-First")
public class First extends HttpServlet
{
  public void service( HttpServletRequest req, HttpServletResponse res )
  throws ServletException, IOException
  {
    res.setContentType("text/html;charset=UTF-8");
    PrintWriter out = res.getWriter();
    out.println( "<head><title>servlet first</title>" );
    out.println( "<META content=\"charset=UTF-8\"></head><body><center>" );
    out.println( "<h1>Test de ma Servlet</h1>" );
    out.println( "<h2>Super ! ça marche</h2>" );
    out.println( "</center> </body>" );
  }
}
```

1. Compilez cette servlet

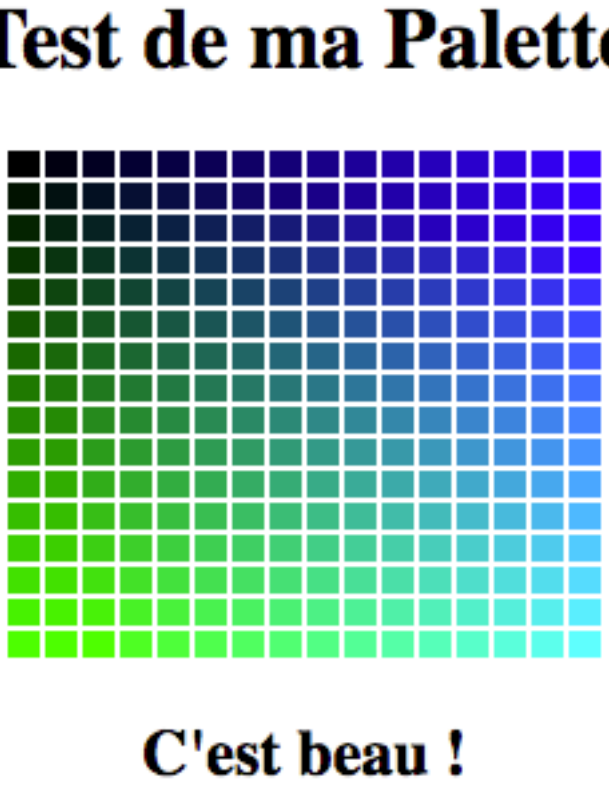
1. Tomcat autorise l'accès à tous les fichiers présents dans le répertoire du contexte. Par sécurité, il est par défaut impossible d'accéder au contenu du répertoire `WEB-INF`. Tout ce qui est dans ce répertoire est privé, sauf avis contraire (avec `@WebServlet` par exemple).

2. Redémarrez votre serveur [2]
3. Appelez la servlet via votre navigateur en utilisant son nom public [3] : `http://localhost:8080/vide/servlet-First`
4. Modifiez `text/html` (enlevez une lettre), recompilez et rechargez. Que se passe t-il ?

6 Génération de html

L'exemple précédent aurait évidemment pû être écrit à l'aide d'une page html statique. Lorsque l'on doit générer une grande quantité de code html ou calculer des données, l'utilisation d'une servlet devient nécessaire.

1. Ecrivez une servlet `Fibonacci.java` qui affiche les 30 premières valeurs de cette suite célèbre. La suite est définie par $f(n) = f(n-1) + f(n-2)$ avec $f(0) = 0$ et $f(1) = 1$. Pour le dire plus simplement, les deux premiers nombres sont 1, puis les suivants sont la somme des deux nombres précédents. Le résultat à obtenir doit donc être : 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040
2. Les couleurs peuvent être décrites en html à l'aide d'un code hexadécimal codé sur 3 ou 6 caractères représentant un mélange des trois couleurs primaires (rouge, vert, bleu). Ainsi, la couleur codée `#3f5` (rouge=3, vert=f, bleu=5) représente un mélange de 3/15e de rouge, 15/15e de vert et 5/15e de bleu. Sachant cela, écrivez une servlet `Palette.java` qui affiche un tableau html de 16 lignes (v allant de 0 à 15) et 16 colonnes (b allant de 0 à 15). Chaque cellule de ce tableau aura une couleur de fond dont la composante de rouge sera "0", les composantes verte et bleue seront fonction de la colonne (b) et de la ligne (v). Vous devriez obtenir quelque chose ressemblant à :



3. Ecrivez une servlet `Ascii.java` qui affiche une page contenant une table des codes ASCII de 32 à 255 (par exemple sur deux colonnes, la première avec le code et la seconde avec la valeur).
   Pour cet exercice, enlevez le `charset=UTF-8` de la méthode `setContentType` et de la balise html `META`.
   Si votre navigateur le permet, vous pouvez essayer de faire varier l'encodage des caractères pour voir des différences (essentiellement à partir du caractère 127).
4. Pensez au style ! Ajoutez à la servlet Ascii le style de table de la semaine dernière

2. Tomcat paramètre son propre CLASSPATH au moment du démarrage. Pour tout nouvel objet, il faut donc le redémarrer. Si vous modifiez une servlet déjà connue, ça n'est par contre plus nécessaire de redémarrer le serveur.
3. L'annotation `@WebServlet` permet de faire le lien entre un chemin indiqué dans le navigateur, et la classe java qui doit s'exécuter. Ce nom **doit** commencer par "/". L'url d'accès à la servlet sera composée du nom de la webapp suivi de ce nom.

7 Page web dynamique

1. Ecrivez une servlet `NouvelAn.java` qui affiche le nombre de secondes avant la prochaine année. Pour cela, vous pouvez importer et utiliser les objets des packages `java.time` et `java.time.temporal` fournis depuis Java 8.

   Par exemple :

   ```
   LocalDateTime today = LocalDateTime.now();
   LocalDateTime aprilFirst = LocalDateTime.of(2018, Month.APRIL, 1, 0, 0);
   Duration delay = Duration.between(today, aprilFirst);
   long seconds = delay.get( ChronoUnit.SECONDS );
   ```

   Chaque appel à cette servlet doit donc logiquement générer un affichage différent. Notez que ce n'était pas le cas des servlets écrites jusqu'à présent !

2. Pour le fun, ajoutez à cette page un "refresh" automatique

   ```
   <meta http-equiv=refresh content=2; >
   ```

8 Servlet en lien avec une base de données

Il est tout à fait possible, dans une servlet, d'afficher des données provenant d'un SGBD. C'est ce que font la majorité des applications web. Ceci se fait bien évidemment grâce à l'API JDBC, comme nous l'avons déjà fait.

Il faut tout d'abord placer le driver JDBC [4] dans le répertoire `WEB-INF/lib` de votre contexte web. Tomcat se charge d'ajouter tous les jar présents à cet endroit dans son classpath lorsqu'il démarre.

Exécutez les requêtes du fichier `foot.sql` sur votre base de données. Cela devrait créer des tables et des données pour la suite des exercices.

1. Ecrivez une servlet `ListeJoueurs.java` permettant d'afficher le contenu de la table JOUEURS, de la même manière que le programme `Select.java` écrit dans le TP précédent.
2. Affichez tout ceci dans un tableau html avec les titres de colonnes dans des balises `TH`.
3. N'oubliez pas votre style ;-) une ligne colorée sur deux, surbrillance de la ligne quand la souris passe dessus
4. Modifier le programme `ListeJoueurs` pour afficher tous les clubs, mais cette fois avec un tableau de joueurs par club. Chaque nom de club sera affiché dans une balise html `H2` avant les joueurs correspondants [5].
5. Mettre en rouge les joueurs qui ont plus de 30 ans au moment de l'affichage de la page.

4. ou un lien symbolique sur celui ci
5. Il est évidemment possible de trouver une solution pour le faire au moyen d'une seule requête SQL.