

# Cours SGBD

## TD – JavaScript, JSP, PHP/JSP et Oracle

22 novembre 2006

## 1 Introduction

### 1.1 Accès au serveur Oracle

Rappel de la manière d'accéder au serveur Oracle, pour les 2 configurations :

	Serveur	Port	Nom de base	URL de connexion JDBC
M1-MIAGE	miage	1521	dbmiage	<code>jdbc:oracle:thin:@miage:1521:dbmiage</code>
M1-INFO	s10	1521	orcl	<code>jdbc:oracle:thin:@s10:1521:orcl</code>

### 1.2 Serveur d'applications Java

Les pages JSP ne vont pas être interprétés, comme les pages PHP, par un module du serveur Web traditionnel (Apache), mais par un serveur Web distinct, incluant un serveur d'applications Java (Tomcat). La manière de mettre à disposition du serveur les fichiers JSP, et d'accéder aux pages Web interprétées, est similaire (on dépose les fichiers dans un répertoire particulier, et on consulte le résultat par un navigateur à une URL), mais pas identique. Vous aurez besoin des instructions suivantes pour les exercices des sections 2 et 4.

#### 1.2.1 M1-MIAGE

1. Se connecter par SSH sur `linux.isi.u-psud.fr`.
2. Aller dans le répertoire `/usr/jakarta-tomcat-5.5.7/webapps/senellar/jsp/m1/`
3. Créer un sous-répertoire dont le nom est votre login (par exemple, `toto`).
4. Tous les fichiers JSP créés doivent être placés dans ce répertoire, soit en les éditant directement au travers de la session SSH avec un éditeur en mode texte (`vim`, `pico`, `emacs -nw...`), soit en les copiant là avec l'utilitaire `scp`.
5. Un fichier `titi.jsp` placé dans le sous-répertoire `toto` sera alors consultable à l'URL :  
`http://linux.isi.u-psud.fr:8080/senellar/jsp/m1/toto/titi.jsp`

#### 1.2.2 M1-INFO

1. Accéder à l'URL `http://s2.ie2.u-psud.fr:8888/`. Chercher votre login dans la liste qui se trouve à la fin de cette page, et noter le numéro de port correspondant (par exemple, 22500, s'il y a marqué `s2.ie2.u-psud.fr:22500`).
2. Vous devez avoir un répertoire `public_tomcat` dans votre répertoire principal, faire `cd public_tomcat`.
3. Taper `cp -r ~/senellar/jsp/* ~/public_tomcat`
4. Éditer le fichier `build.properties` :
  - LOGIN doit être remplacé par le login Unix
  - PORT doit être remplacé par votre numéro de port noté plus haut
5. Taper `ant -Dmanager.password=mdp install` (où `mdp` est votre mot de passe Unix). Il ne devrait pas y avoir de message d'erreur.
6. Les fichiers JSP sont à placer dans le sous-répertoire `web`.
7. Après toute modification de fichiers, il faut mettre à jour les fichiers sur le serveur en tapant :  
`ant -Dmanager.password=mdp remove && ant -Dmanager.password=mdp install`  
à partir du répertoire `public_tomcat`.
8. Un fichier `titi.jsp` sera alors consultable à l'URL :  
`http://s2.ie2.u-psud.fr:22500/jsp/titi.jsp`  
(si 22500 est le numéro de port qui vous a été attribué).

## 2 Introduction à JSP

### 2.1 Page élémentaire

1. Créer un fichier titi.jsp et y mettre :

```
<%@page contentType="text/html" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html lang="fr">
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
    <title>Page de test</title>
  </head>
  <body>
    <h1><c:out value="Bonjour monde!" /></h1>
  </body>
</html>
```

Tester, en suivant les instructions de la section 1.2.

2. Ajouter une boucle affichant les nombres de 1 à 10 :
  - Au moyen d’une *scriptlet*, c’est à dire de code Java encadré de balises `<%` et `%>`. On utilisera `pageContext.getOut()` pour récupérer un objet sur lequel appeler la méthode `write()` pour écrire dans la page.
  - Au moyen de la JSTL.

### 2.2 Table de multiplication

En utilisant la JSTL (sans scriptlet, donc), créer une page JSP affichant la table de multiplication de 1 à 10 sous forme d’un tableau HTML. Ne pas oublier les en-têtes de ligne et colonne appropriés.

Comparer les codes JSP avec les codes PHP équivalents du TD précédent. Quelles similarités et différences y a-t-il ?

## 3 Introduction à JavaScript (fin de TD)

1. Récupérer sur la page du cours le fichier `test_js.html`. Faire en sorte que le fond de la page change à chaque fois qu’on clique sur un de ces boutons radio. On devra utiliser l’objet JavaScript `document.body.style.backgroundColor`.
2. Faire en sorte que le champ texte de `TestJS.html` disparaisse si la case à cocher est cochée, réapparaisse sinon. On aura besoin de la méthode `getElementById()` de l’objet `document`, ainsi que de la propriété `style.visibility` (avec les valeurs `hidden` ou `visible`) d’un élément. De plus, dans un gestionnaire d’événement d’une case à cocher, la propriété `checked` permet de savoir si celle-ci est cochée.

## 4 Accéder à Oracle depuis PHP et JSP

On va ici construire une application simple manipulant une base de données avec interface Web, que l’on écrira à la fois en PHP et en JSP, afin de manipuler ces deux langages et d’en voir les similarités et différences. L’application consiste à gérer les jetons et consommations de café d’un certain nombre de personnes.

1. Récupérer sur la page du cours le script SQL `cafe.sql` et l’exécuter dans `sqlplus`. Celui-ci crée une table `Cafe` comportant les champs suivants, et y ajoute deux entrées à la main :
  - `Identifiant`, chaîne de caractères identifiant uniquement la personne
  - `Jetons`, nombre de jetons de café dont dispose la personne (positif ou nul)
  - `Consommations`, nombre de cafés bus par cette personne (positif ou nul)

Le script crée également une procédure et une fonction qui seront utilisés à la fin de cet exercice.

2. Récupérer sur la page du cours la page HTML `form_cafe.html`, qui comporte un formulaire avec deux champs texte (identifiant et jetons). Ajouter à cette page un code JavaScript de validation du formulaire, qui vérifie que l’identifiant n’est pas vide, et que le nombre de jetons est bien un entier positif. On pourra pour cela utiliser les fonctions JavaScript suivantes :
  - `document.getElementById('toto')` renvoie l’objet du document HTML d’identifiant `toto`.

- `toto.value`, où `toto` est un objet HTML de type champ de formulaire, renvoie la valeur de ce champ.
- `text.length`, où `text` est une chaîne de caractères, renvoie la longueur de cette chaîne.
- `Number(text)`, où `text` est une chaîne de caractères, convertie cette chaîne en nombre. On peut tester si la conversion n'a pu avoir lieu en utilisant la fonction `isNaN` sur le résultat.
- `Math.floor` est la fonction partie entière.

Tester.

Copier la page dans `form_php.html` et `form_jsp.html` et positionner l'attribut `action` du formulaire respectivement à `ajout_personne.php` et `ajout_personne.jsp`. On placera les fichiers relatifs à PHP et ceux relatifs à JSP dans les endroits accessibles aux deux serveurs Web.

3. Créer `ajout_personne.php` et `ajout_personne.jsp` (avec scriptlet), qui ajoutent à la table `Cafe` une nouvelle personne telle qu'indiquée dans le formulaire, avec un compte de consommation nul initialement. La page HTML résultant contiendra juste un lien vers, respectivement `affiche_liste.php` et `affiche_liste.jsp`. Pour gagner du temps, on ne vérifiera pas que les données entrées sont cohérentes (ce qui est entré comme jetons est bien un nombre positif, l'identifiant n'est pas déjà présent dans la table, etc.), mais il faudrait le faire en pratique! On ne peut pas non plus à se fier à la validation JavaScript, qui ne peut être complète, et qui peut facilement être contournée. On fera donc attention à ne pas violer ces contraintes lors des essais des scripts. En écrivant la deuxième version, observer qu'on n'est pas obligé de tout retaper!

Modifier `ajout_personne.jsp` pour utiliser la JSTL plutôt qu'une scriptlet. Comparer. Pour les questions suivantes, nous n'utiliserons plus que la JSTL.

4. Créer `affiche_liste.php` et `affiche_liste.jsp`, qui affichent comme un tableau HTML le contenu de la table `Cafe`.
5. Ajouter à chaque ligne de `affiche_liste.php` et `affiche_liste.jsp` un bouton permettant de boire un café (ajouter une consommation, puis enlever un jeton). Écrire les `boire_cafe.php` et `boire_cafe.jsp` correspondants. Cette opération devra être une transaction.
6. Que se passe-t-il quand une contrainte est violée? Afficher un message d'erreur approprié dans `affiche_liste.php` et dans `affiche_liste.jsp`. On peut utiliser `error_reporting(0);`, en PHP, pour que PHP ne génère pas lui-même de message d'erreur. Vérifier que dans le cas de violation d'une contrainte, la transaction garantit que la base reste dans un état cohérent.
7. Ajouter un deuxième bouton *Ajouter un jeton* dans le tableau affiché par `affiche_liste.php` et `affiche_liste.jsp`. Le formulaire de ce bouton pointera vers `ajout_jeton.php` (respectivement `ajout_jeton.jsp`). Écrire ces deux scripts, qui se contenteront d'appeler la procédure stockée PL/SQL `AjouteJetons` pour l'identifiant en question. Pour JSP, comme cela n'est pas faisable avec la JSTL sans écrire de nouveaux tags, on utilisera une scriptlet.
8. À la fin de la page `affiche_liste.php` et `affiche_liste.jsp`, afficher le nombre de consommateurs à l'aide de la fonction stockée PL/SQL `NbConsommateursCafe`. Pour JSP, comme cela n'est pas faisable avec la JSTL sans écrire de nouveaux tags, on utilisera une scriptlet.