2008

UFR Ingénieurs 2000

Vivien Boistuaud
Julien Herr

TP J2EE – GUIDE DE DEPLOIEMENT DE L'APPLICATION WEB DE GESTION DES COMPTES

Ce document a été réalisé par V. Boistuaud et J. Herr dans le cadre des travaux pratiques de J2EE dispensés par Olivier Planson, Ingénieur travaillant pour la société IBM en France.

Table des matières

Introduction

Lors des différents TP de J2EE dirigés par Olivier Planson, Vivien Boistuaud et Julien Herr ont réalisé, pas à pas, une *J2EE WebApplication* permettant de simuler l'interface de gestion des comptes d'une banque.

Cette application Web utilise les technologies JDBC, Servlet et JSP, mais ne met pas en œuvre les principes transactionnels, ni les principes de persistance d'objet, qui seront abordés dans les séances de travaux pratiques dirigés par Yves Mourrier.

Durant ces séances de TP, nous avons réalisé les sujets distribués jusqu'au numéro 10, dans la mesure où les sujets 11 et 12 ne nous avaient pas été distribués et où Yves Mourrier nous a fait travailler sur les sujets de TD EJB lors de la dernière séance.

Dans un premier temps, nous présentons brièvement le travail réalisé en TP ainsi que les pré-requis nécessaires à l'utilisation de notre *WebApp*. Dans un second temps, nous indiquons comment mettre en place la base de données nécessaire à l'utilisation de l'application. Enfin, nous présentons les différentes alternatives disponibles pour l'exécution de l'application : Déploiement direct sous Tomcat et Jetty, ainsi qu'importation et lancement sous Eclipse en tant que projet Tomcat.

1. Informations générales sur les TP de J2EE

Les TP de J2EE dirigés par Olivier Planson avaient pour but de nous familiariser avec trois technologies de la norme J2EE : JDBC, pour la connexion aux bases de données ; Servlet, pour la création de logique métier de traitement de requêtes web, et la technologie JSP pour la conception de pages dynamiques.

1. Travail réalisé en TP et fonctionnalités finales

Lors des séances de TP, nous avons réalisé les activités suivantes :

- Création d'une base de données PostgreSQL ;
- Connexion à PostgreSQL via JDBC ;
- Création d'un bootstrap[1] de création des tables en base de données ;
- Création d'un Bean de consultation et modification des données ;
- Création d'une Servlet de gestion des comptes ;
- Création des JSP présentant les données à l'utilisateur.

Pour réaliser ces différents modules de façon efficace, nous avons utilisé un environnement de développement intégré : Eclipse 3.2 (la version 3.3 fournie par l'Université de Marne-la-vallée ne permettant pas d'exploiter le plugin Tomcat de Sysdeo[2]).

L'application web réalisée est capable de créer une transaction monétaire sur un compte.

La section suivante décrit les IDE, plugins, applications et bibliothèques dont vous aurez besoin pour pouvoir exécuter notre application web dans un conteneur de Servlets, et éventuellement depuis un environnement de développement.

2. Pré-requis nécessaires au déploiement de l'application

Avant de pouvoir déployer l'application, il est nécessaire d'avoir :

- Un conteneur de Servlet préinstallé (tomcat ou jetty par exemple),
- Une copie du pilote JDBC correspondant à votre base de données, placée dans le shared/libs de votre conteneur de Servlet,
- Une copie de notre archive WAR,

[1] Petit programme d'amorçage qui permet d'en lancer un plus gros
[2] http://www.eclipsetotale.com/tomcatPlugin.html

Il est également nécessaire d'avoir une base de données crée et utilisable. C'est dans la section suivante que nous allons voir comment créer cette dernière, et les différentes alternatives qui vous sont proposées pour la peupler.

Enfin, pour l'utilisation depuis Eclipse décrite à la fin du document, il vous sera nécessaire d'installer le plugin Tomcat Sysdeo qui vous permettra d'exécuter Tomcat directement depuis Eclipse. Vous pourrez télécharger ce plugin à l'adresse : http://www.eclipsetotale.com/tomcatPlugin.html

2. Configuration de la base de données applicative

Pour fonctionner, notre application web a besoin d'avoir accès à une base de données dans laquelle elle peut stocker les informations sur les comptes et les transactions effectuées.

Notre application ayant été écrite en SQL89 et exploitant JDBC, il est tout à fait possible de créer la base de données sur n'importe quel SGBD (DB2, Oracle, MySQL, PostgreSQL…).

Cette section présente les moyens de configurer l'application pour qu'elle crée automatiquement en base de données les tables dont elle a besoin.

Dans l'exemple ci-après, le type de base de données retenu pour l'exemple est PostgreSQL.

1. Création d'une base de données PostgreSQL

Pour créer une base de données, nous utilisons l'outil Web phpPgAdmin[3] qui permet de gérer une base de données PostgreSQL via un simple client Internet. Nous prenons donc comme pré-requis que cet outil est installé et opérationnel.

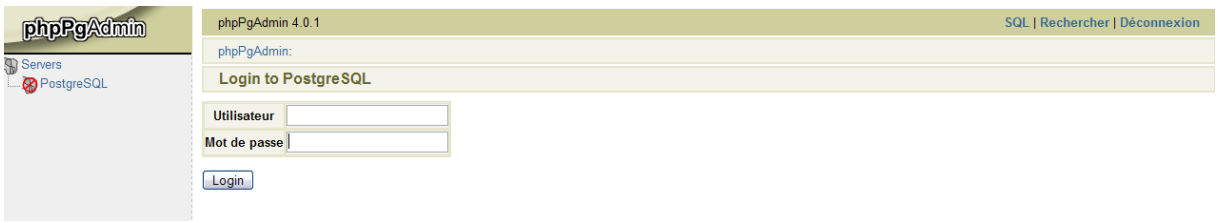

Figure 1 - Authentification à phpPgAdmin

Tout d'abord, nous devons nous authentifier à la base de données comme nous pouvons le voir dans la figure ci-avant.

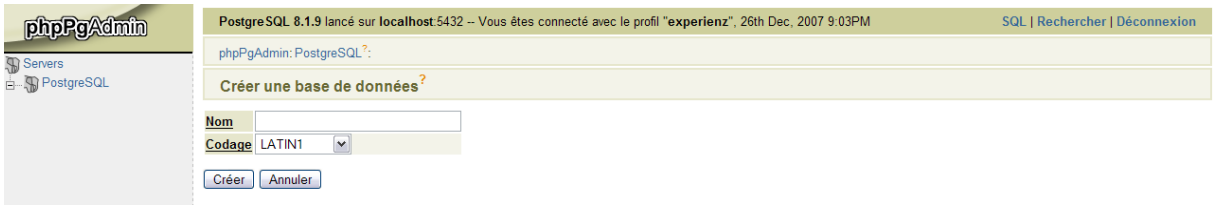

Figure 2 - Création d'une base de données

[3] http://phppgadmin.sourceforge.net

Nous devons ensuite créer la base de donnée en spécifiant le nom et le codage que nous allons utiliser (figure ci-avant). Il est recommandé d'utiliser un codage UNICODE (UTF-8 ou UTF-16 en fonction de la langue principale de stockage des valeurs textuelles).

Figure 3 - Création d'une table

Il ne reste plus qu'à créer les tables dont nous avons besoins en spécifiant le nom et le nombre de colonne que va posséder cette table (figure ci-avant).

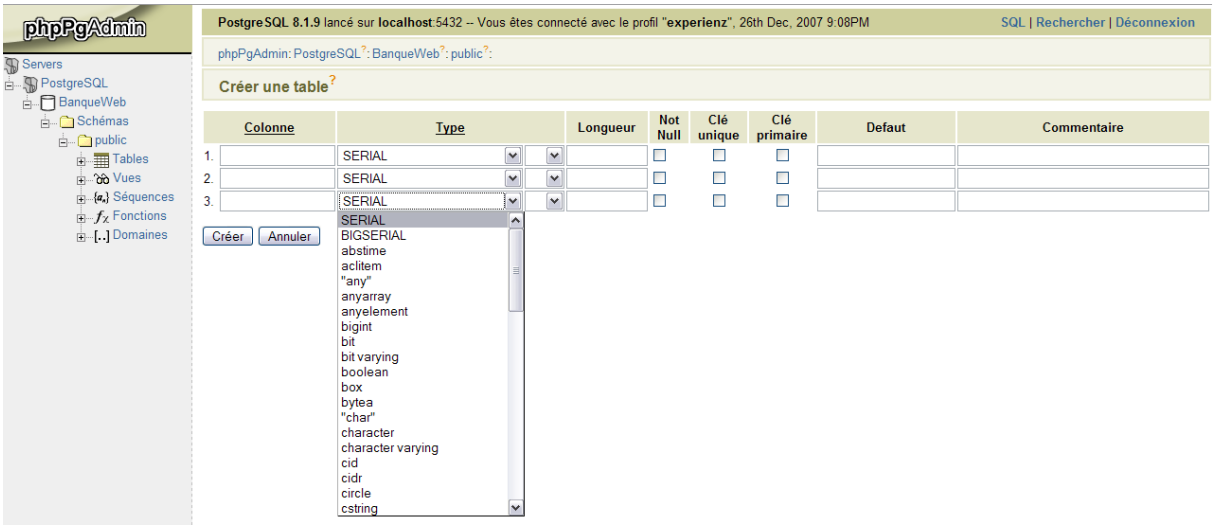

Figure 4 - Spécification des champs d'une table

Enfin, nous devons spécifier les champs de la table en indiquant le nom, le type ainsi que diverses contraintes (longueur, clé primaire, clé unique, valeur par défaut).

2. Fichier de description de la base de données

Afin d'améliorer le confort lors du déploiement de notre projet sur différent serveurs, il est possible de passer outre la procédure de création de table décrite précédemment.

En effet, nous avons développé durant les séances de TP un outil permettant de générer automatiquement les tables en base de données. Pour le lancer, vous aurez besoin de créer un fichier de configuration contenant les informations suivantes :

```
# Defines the JDBC driver class name
jdbc.classname = package.classname
jdbc.username = login
jdbc.password = password
jdbc.url = jdbc:drivername://dbhost.dbdomain.tld:port/dbname
```

Par exemple, notre configuration était la suivante lors des TP:

```
# db.properties example file by JH / VB
jdbc.classname = org.postgresql.Driver
jdbc.username = vboistua
jdbc.password = password
jdbc.url = jdbc:postgresql://sqletud.univ-mlv.fr:5432/vboistua_td4_exo1
```

Remarque : Si vous ne précisez pas les paramètres jdbc.username et.jdbc.password, aucune erreur ne sera affichée. En effet, vous pouvez également les préciser dans l'URL de connexion.

Par exemple :

```
jdbc.url = jdbc:postgresql://sqletud.univ-
mlv.fr/vboistua_td4_exo1?user=vboistua&password=password
```

Est équivalent à préciser les paramètres « `vboistua` » et « `password` » comme valeurs de `jdbc.username` et `jdbc.password`.

Notez de plus, que la base de données indiquée dans l'URL JDBC doit avoir été créée avant que le bootstrap ne soit lancé : elle ne sera pas créée automatiquement.

3. Lancement du Bootstrap de création automatique

Pour lancer le bootstrap de création automatique, il vous suffit de connaître l'emplacement sur votre système du pilote JDBC correspondant à votre base de données, ainsi que celui du WAR que nous avons fournit avec ce document.

Nous considèreront que le chemin et le nom du WAR se trouvent dans la variable d'environnement UNIX $IR_WAR et que le chemin et le nom du JAR contenant le pilote JDBC sont dans la variable $JDBC_DRIVER. Nous considèreront également que les réglages de base de données sont dans un fichier db.properties présent dans le répertoire courant d'exécution de la commande.

Pour lancer le créateur de base de données, il suffit alors d'exécuter la commande :

```
java –cp $IR_WAR:$JDBC_DRIVER gestionDB.CreationTable db.properties
```

Remarque : sous Windows, les variables d'environnement sont entourées de % au lieu de commencer par un $, et les chemins d'accès du classpath doivent être indiqués avec un point virgule (;) comme séparateur au lieu des deux points (:) utilisés dans l'exemple précédent.

Les commandes exécutées sont équivalentes au script SQL ci-après. Si vous n'arrivez pas à exécuter le Bootstrap correctement, vous pouvez exécuter les requêtes suivantes directement sur votre base de données pour créer les tables correspondantes :

```
CREATE TABLE COMPTE (NOACCOUNT CHAR(4) NOT NULL PRIMARY KEY, NOM
VARCHAR(20),PRENOM VARCHAR(20),SOLDE DECIMAL(10,2) NOT NULL);

CREATE TABLE OPERATION (NOACCOUNT CHAR(4) NOT NULL, DATE DATE NOT NULL DEFAULT
CURRENT_DATE, TIME TIME NOT NULL DEFAULT CURRENT_TIME, OP CHAR(1) NOT NULL,
VALEUR DECIMAL(10,2) NOT NULL);
```

Notez que ceci ne créera pas de données de démonstration. Si vous souhaitez saisir des informations fictives pour vérifier le fonctionnement de l'application, nous vous proposons de saisir les lignes ci-dessous :

```
INSERT INTO COMPTE(NOACCOUNT, NOM, PRENOM, SOLDE) VALUES ('0001', 'BOISTUAUD',
'Vivien', 1523.10);

INSERT INTO COMPTE(NOACCOUNT, NOM, PRENOM, SOLDE) VALUES ('0002', 'HERR',
'Julien', 2547.34);

INSERT INTO OPERATION(NOACCOUNT, OP, VALEUR) VALUES ('0001', '+', 1600);
INSERT INTO OPERATION(NOACCOUNT, OP, VALEUR) VALUES ('0001', '-', 76.90);
INSERT INTO OPERATION(NOACCOUNT, OP, VALEUR) VALUES ('0002', '+', 2600);
INSERT INTO OPERATION(NOACCOUNT, OP, VALEUR) VALUES ('0002', '-', 52.66);
```

3. Méthodes de déploiement

Une fois la base de données créée et initialisée, plusieurs méthods de déploiement vous sont proposées. Vous pouvez soit déployer directement dans un conteneur de Servlet, soit utiliser Eclipse pour la visualisation et l'édition des sources, ainsi que pour exécuter l'application web.

1. Déploiement sur Tomcat

Il existe plusieurs façons de déployer des applications web sur Tomcat. Dans un premier temps, nous décrivons la façon la plus simple et la plus rapide. Dans un second temps, nous expliquons comment personnaliser l'adresse de déploiement de l'archive web (WAR).

1. Déploiement automatique sans personnalisation de l'URL

En effet, dans sa configuration initiale, Tomcat déploie automatiquement toutes archives War déposées dans son répertoire « webapps » soit à son lancement, soit au court de son exécution pour les versions de Tomcat supérieures à 5.0 (lorsque le « hot-deploy » est activé, ce qui est le cas par défaut). Notez que vous pouvez changer le nom du répertoire en changeant la valeur de « appBase » dans le fichier conf/server.xml de Tomcat.

Il suffit donc de placer l'archive du projet dans le répertoire « appBase », d'attendre quelques instants, le temps que Tomcat déploie l'application. Il est ensuite possible de tester l'application Web par l'intermédiaire de l'url « http://servername:port/warname ».

Par exemple, si tomcat est lancé sur votre machine et répond sur le port http 8080, avec un war nommé boistuaud_herr_compte.war, alors l'adresse de déploiement sera : http://localhost:8080/boistuaud_herr_compte et la page d'accès à la Servlet principale est alors :

http://localhost:8080/boistuaud_herr_compte/GestionOperations

Remarque : Si vous essayez d'entrer un numéro de compte sans avoir configuré le WAR, un message vous demandant de contacter votre administrateur apparaîtra. Dans ce cas, décompressez le fichier WAR et modifiez le fichier WEB-INF/web.xml pour que les variables d'environnement jdbc.classname, jdbc.username, jdbc.password et jdbc.url contiennent les mêmes informations que dans le fichier db.properties précédemment

créé. De plus, vous devez placer dans le server/libs ou le common/libs de votre conteneur de Servlet le JAR correspondant à votre pilote JDBC.

Exemple :

```
  <env-entry>
    <env-entry-name>jdbc.url</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-
value>jdbc:postgresql://localhost:5432/banqueweb?user=postgres&amp;password=de
mo</env-entry-value>
  </env-entry>

  <env-entry>
    <env-entry-name>jdbc.classname</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>org.postgresql.Driver</env-entry-value>
  </env-entry>
```

Notez que vous devrez également recréer le fichier WAR avec le nouveau fichier modifié, à l'aide de n'importe quel utilitaire permettant de créer des archives au format ZIP.

2. Déploiement à une URL spécifique avec configuration manuelle

Pour configurer l'URL de déploiement, il suffit de laisser le WAR dans le répertoire appBase comme indiqué précédemment, et de rajouter une entrée <Context> dans le fichier conf/server.xml (ou le server XML que vous utilisez pour exécuter Tomcat) dans le nœud Server > Service Engine > Host correspondant au serveur et port sur lequel vous effectuez le déploiement.

La ligne à ajouter est de la forme :

```
<Context path="/Compte" reloadable="true" docBase="boistuaud_herr_compte"
workDir="boistuaud_herr_compte/work" />
```

Vous pouvez alors accéder à l'application depuis l'URL http://localhost:8080/Compte/GestionOperations

Notez toutefois que si vous laissez l'application dans votre répertoire appBase (webapps) alors elle sera accessible via les deux urls.

2. Déploiement sur Jetty

Le déploiement sur Jetty est similaire à celui sur Tomcat, dans la mesure où ils respectent tous deux les spécifications définies par Sun pour les normes Servlet et JSP.

Par défaut, le répertoire de déploiement automatique s'appelle aussi « webapps » et il suffit d'y placer notre WAR pour qu'il soit déployé au prochain démarrage de Jetty.

Pour définir une URL de déploiement personnalisé, ceci est spécifique à chaque conteneur de Servlet et/ou serveur d'application. Dans Jetty, il suffit de modifier le fichier etc/jetty.xml et d'ajouter un new <New> sous le nœud <Configure> comme suit :

```
<New id="boistuaud_herr" class="org.mortbay.jetty.webapp.WebAppContext">
    <Call name="setContextPath">
        <Arg>/Compte</Arg>
    </Call>
    <Call name="setWar">
      <Arg>webapps/Boistuaud_herr_compte.war</Arg>
    </Call>
</New
```

On peut alors appeler la web application sous le contexte /Compte du serveur Jetty.

3. Exécution sous Eclipse en tant que projet Tomcat

Nous allons maintenant décrire comment importer les sources de notre projet comme « Projet Tomcat » dans Eclipse et comment l'exécuter depuis Eclipse.

1. Importation dans Eclipse en tant que projet Tomcat

Nous considérons en pré-requis, comme déjà indiqué dans une section précédente, que le plugin Tomcat Sysdeo est correctement installé et fonctionnel.

Tout d'abord, il faut extraire les sources. Elles se trouvent dans le fichier WAR fournis. Pour les extraire, il suffit d'utiliser n'importe quel archiveur zip (exemeple : Winzip, Winrar, …) ; un fichier WAR étant une simple archive zippée avec une extension différente.

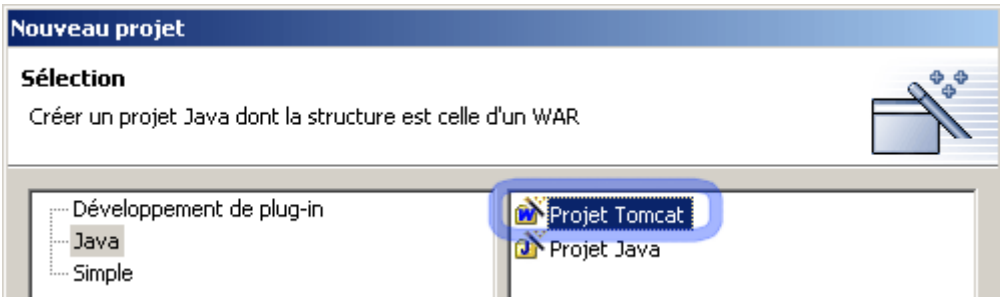

Figure 5 - Nouveau projet Tomcat

Dans Eclipse, nous devons créer un projet Tomcat disponible, comme le montre la figure ci-avant, dans le menu :

```
File > New > Project… > Java > Projet Tomcat
```

L'assistant permet ensuite de créer un projet Tomcat en indiquant le nom du projet ainsi que le nom du contexte Tomcat.

Il ne reste plus qu'à lier nos sources avec ce projet. Pour cela, nous devons copier les sources décompressées précédemment dans le répertoire WEB-INF/src de notre nouveau projet.

Notez que nous vous fournissons, par défaut, les fichiers nécessaires à l'autoconfiguration du plugin. Aussi, cette dernière étape ne devrait pas être nécessaire. Toutefois, il se peut, en fonction de la façon dont vous importez le projet, que vous ayez besoin de l'utiliser.

2. Configuration et exécution depuis Eclipse

Pour que notre projet soit déployable sur Tomcat et utilisable avec le plugin Tomcat de sysdeo, il est indispensable que nous définissions un nœud <Context> dans le fichier server.xml utilisé par le plugin. Si ce nœud n'est pas configuré, l'application ne sera pas déployée.

Pour cela, ouvrez la fenêtre de préférences d'Eclipse via le menu Window > Preferences. Dans la liste, sélectionnez le nœud nommé « Tomcat » et indiquez le chemin vers le server.xml que vous souhaitez utiliser, en général, il se trouve dans le répertoire conf de Tomcat.

Ajoutez alors un nœud <Context> comme décrit dans la section 3.1.2 de ce document, en remplaçant docBase et workDir par les valeurs correspondant au répertoire dans lequel se trouve votre projet eclipse et, respectivement, son sous répertoire work.

Vous pourrez alors démarrer Tomcat en utilisant l'icône correspondante dans la barre d'outils fournie par le plugin Sysdeo. N'oubliez pas de configurer le fichier WEB-INF/web.xml pour qu'il reflète vos réglages de base de données.