AP-5		TD n°3
J2EE		12 novembre 2013

Objectifs

Développement d'une application Web à partir d'un existant java non web
Listeners, Application
Séparation des couches MVC

Gestion des utilisateurs et des groupes

Une application de gestion d'utilisateurs et de groupes a été développée. Elle fonctionne pour l'instant en mode console. Vous devez en faire une version web de type client riche (avec ajax).
L'application en mode console existante a été développée en utilisant les consignes suivantes :

Contexte

Le développement à effectuer le sera dans le cadre d'une architecture applicative potentiellement multiple, mais à ce jour indéterminée : Web/Mobile/Client lourd…
Il s'agit de prévoir une gestion des utilisateurs, qui permettra de contrôler les accès à une application.

Contraintes fonctionnelles

Chaque utilisateur possède un login, et un mot de passe (password). Le login doit être unique pour permettre l'identification.
Les informations de l'utilisateur seront complétées par son nom, et son prénom (firstName et lastName).
Un utilisateur peut appartenir à un groupe.
Chaque groupe possède un nom, utilisé également comme identifiant textuel.
L'application devra permettre :
la gestion des utilisateurs :

- Ajout
- Modification
- Suppression
- Affectation à un groupe

La gestion des groupes :

- Ajout
- Modification
- Suppression

Contraintes techniques

Le développement sera effectué en java, en respectant les règles de développement de ce langage

- Les classes métier créées seront stockées dans un package **net.bo**
- Les classes techniques dans **net.technics**
- Les classes d'affichage de l'application dans **net.gui**

La persistance des données n'est pas à prévoir (pour l'instant).
Une méthode **loadData** permettra de charger des données exemples.

AP-5		TD n°3
J2EE		12 novembre 2013

Développement Web

Contraintes fonctionnelles :

Les contraintes fonctionnelles sont les mêmes que pour l'application java en mode console.

Contraintes techniques :

Vous reprendrez une première version du projet en important l'archive WAR **gestUser.war**.
Il faudra intégrer les classes du projet en mode console sous forme d'archive jar dans le projet web.
Les pages devront utiliser les fonctions javascript fournie dans le fichier forms.js fourni dans le projet.
On veillera particulièrement à séparer les vues, le modèle, et le contrôle.

- Les formulaires (*.jsp) seront à placer dans le dossier WebContent/forms, leur nom respectera toujours la forme suivante : frm**NomAction**.jsp
  frmDeleteUser.jsp pour le formulaire de suppression d'un utilisateur par exemple.
  Les vues contiendront le moins possible de code java.
- Les actions correspondant aux formulaires seront implémentées dans des servlets, stockées dans le package **web.action**, et porter un nom correspondant au formulaire qui leur est associé : do**NomAction**.act
  doDeleteUser.act pour l'action associée au formulaire **frmDeleteUser.jsp**.

Travail à faire

Implémenter les actions existantes du mode console dans l'application web, en respectant les contraintes ci-dessus.
Vous pourrez utiliser :

- La classe **MainController** dont le code est fourni et l'implémentation de la méthode **deleteUser**
- la javadoc des classes du projet en mode console
- la javadoc des classes du projet Web
- Les diagrammes de classe ci-joints

A faire :

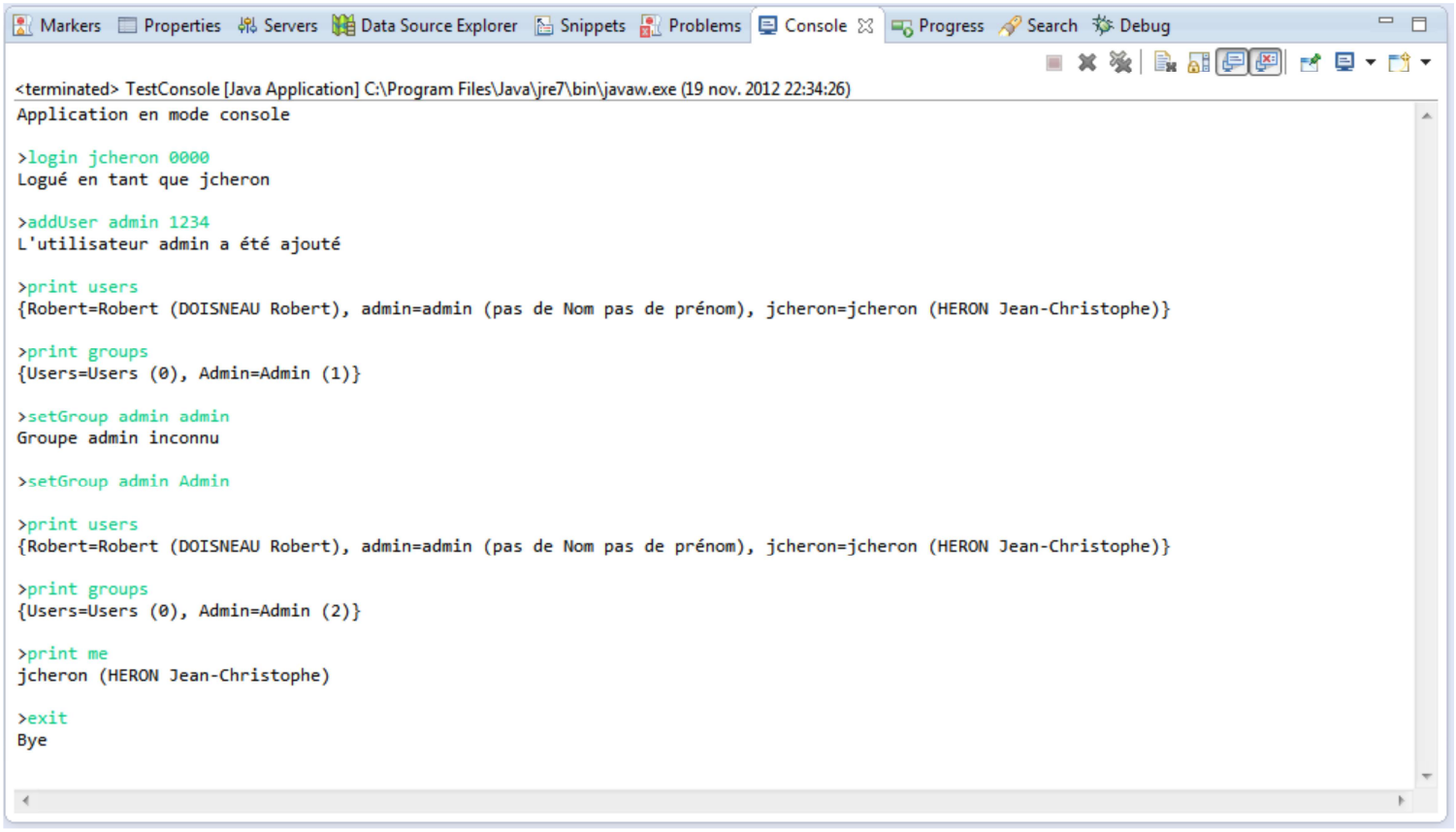
- Importer le fichier WAR pour créer le projet Web (File/Import/war file)
- Importer l'archive Zip pour créer le projet en mode console(File/import/existing project into workspace)
- Créer un jar à partir du projet console
- Incorporer le jar dans le dossier **WEB-INF/lib** du projet web
- Créer un listener sur la session Web nommé SessionStart.java (file/New/Listener) et ajouter y la création d'une variable de session nommée sessionApp

```
/**
* @see HttpSessionListener#sessionCreated(HttpSessionEvent)
*/
public void sessionCreated(HttpSessionEvent event) {
	session=event.getSession();
	session.setAttribute("sessionApp", new SessionApp() );
}
```

- Créer la servlet **MainController** et lui associer le mapping ***.do**
- Implémenter la méthode **deleteUser** et modifier la vue **frmDeleteUser.jsp**
- Déplacer les vues vers le dossier **WEB-INF.**

- Créer les méthodes suivantes dans la servlet **MainController** pour remplacer les jsp suivantes (sur le modèle de deleteUser) :
  - action/doPrintMe.jsp -> **public void printMe(request,response)**
  - action/doLogin.jsp -> **public void login(request,response)**
  - action/doAddUser.jsp -> **public void addUser(request,response)**
  - action/doDisconnect.jsp -> **public void disconnect(request,response)**
- Créer les méthodes (de la classe MainController) et les vues (dans WEB-INF/forms) correspondant aux fonctionnalités suivantes :
  - Ajout d'un utilisateur
  - Modification d'un utilisateur
  - Ajout d'un groupe
  - Modification d'un groupe
  - Affectation d'un utilisateur à un groupe

Exemple d'exécution en mode console de l'application :



Package net.bo

Package net.action