Intégration à une application web

Contexte

Il s'agit du même contexte que celui du TP précédent.
L'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).

Activités du référentiel concernées :

- D1.1 - Analyse de la demande
- D1.2 - Choix d'une solution
- D1.3 - Mise en production d'un service
    - A1.3.1 Test d'intégration et d'acceptation d'un service
- D4.1 - Conception et réalisation d'une solution applicative

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 **gestUserGroup.war**.
Il sera nécessaire 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.

Missions

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

- le modèle fourni par le formulaire frmAddUser.jsp et la servlet associée : doAddUSer.act
- 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

|h SessionStart.java

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

- Transformer les jsp suivantes en servlet. Tester ensuite la connexion au site.
  - action/doPrintMe.jsp → doPrintMe.act
  - action/doLogin.jsp → doLogin.act
- Créer les actions et les formulaires 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

Objets ajax utilisés :
Requête ajax : classe Forms.Ajax

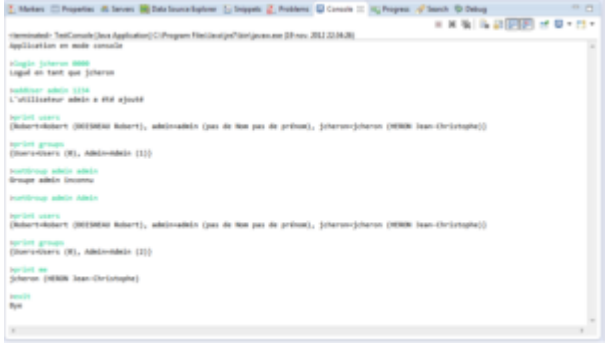
```
var req=Forms.Ajax(String idDiv, String url [,String params,var func, Object
indicator]);
```

Principales méthodes :

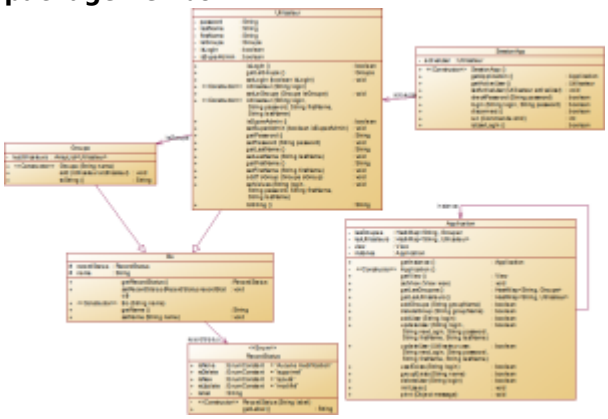
```
    req.get();
    req.post();
    req.postForm(formName);
```

voir http://api.kobject.net/doc/doku.php?id=javascript:ajx

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



package net.bo



package net.action



From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/slam4/tp2?rev=1353762397

Last update: **2019/08/31 14:38**