TD-MVC-Struts

Conception et réalisation d'une application Web pour la gestion de notes.

Objectif : réaliser un système d'information permettant la gestion des notes des étudiants d'un établissement secondaire.

Description du système.

Un premier entretien avec le directeur de l'établissement, nous apporte les éclaircissements suivants sur les informations que nous devons stocker :

- il y a les enseignants, les étudiants et les modules.
- Un *enseignant* est identifié par son *nom* (chaîne de caractères), son *prénom* (chaîne de caractères), son *Numen* (Numéro d'identification national). Il a aussi un *login* (chaîne de caractères), et un *password* (chaîne de caractères).
- Un *etudiant* est identifié par son *nom* (chaîne de caractères), son *prénom* (chaîne de caractères), son numéro d'étudiant (*Netudiant*). Il a également un *login* (chaîne de caractères), et un *password* (chaîne de caractères).
- Un *module* a un *identifiant* (chaîne de caractères) et un *intitulé* (chaîne de caractères).
- Chaque module est enseigné par un seul enseignant, mais un enseignant peut enseigner plusieurs modules.
- Un étudiant a au plus une *note* (un réel) par module.

Notre deuxième entretien concernait l'aspect traitement : le directeur nous a confié que son souhait est d'avoir un système qui permet :

- à chaque enseignant de saisir les notes de ses étudiants, et
- à chaque étudiant de consulter ses propres notes,
- et ceci de n'importe qu'elle machine connectée au net.

Démarche et conception.

Armé par notre expérience sur les applications web et les entretiens réalisés, nous entamons notre réflexion sur la conception du système.

I. Pour la représentation des informations, quoi de mieux qu'une bonne vieille base de données sous oracle ! Nous allons donc utiliser notre outil pour la manipulation des BD sous oracle, *sqldevloper,* pour nous connecter à notre serveur, *miage03.dmiage.u-paris10.fr:1521:MIAGE***.** Nous écrivons ensuite les requêtes SQL de création de la base, après avoir conçu la schéma de la base avec l'ensemble des liens logiques entre les tables.

II. Pour la partie traitement, les discussions nous mènent, logiquement, à l'utilisation de la technologie **JEE** des **Sevlets/Jsp**, avec une architecture **MVC** mise en place par le *framework* **Struts**. Dans notre architecture, nous décidons que :

a. *le modèle* s'occupe de l'aspect métier de l'application. Il gère l'authentification d'un utilisateur. Si l'utilisateur fournit les bons login et mot de passe (comparés à ceux stockés dans la BD), l'authentification crée l'instance d'objet (*un bean*) correspondant à son statut : un objet *enseignant,* pour un enseignant et un objet *étudiant*, pour un étudiant. Chaque instance va contenir toutes les informations concernant l'utilisateur associé : pour un étudiant, elle comprendra toutes ses notes. Pour un enseignant, elle comprendra les modules enseignés ainsi que les étudiants associés à chaque module.

b. *La vue* s'occupe de l'interface avec l'utilisateur. Elle contient, entre autres,
   i. une page d'authentification (automatiquement chargé au lancement de l'application). Elle permet de saisir le login, le mot de passe, et le statut de l'utilisateur (enseignant ou étudiant) ;
   ii. une page d'accueil d'un étudiant. Elle permet de visualiser ses notes ;
   iii. une page d'accueil d'un enseignant. Cette page contient,
      1. la liste des modules enseignés ;
      2. un click sur l'un des modules de l'enseignant ouvre une nouvelle page contenant la liste des étudiants de ce module, avec une zone de saisie de la note de chacun. (PS : la zone doit contenir la note si elle a déjà été saisie auparavant).

c. *Le contrôleur* fait l'aiguillage des requêtes de l'utilisateur aux modèles qui peuvent les satisfaire.

d. Aussi, toute erreur susceptible de se produire dans le système doit être traitée d'une façon propre (i.e., exceptions, fichiers d'erreurs,…etc)

A vous de réaliser l'application !