RAPPORT

Numéro de Référence
#AZR8933

(Document de 7 pages)

Résumé

Rapport devoir jee

Mots Clés

Rapport, JEE

Université Aix Marseille

SECTION DES REDACTEURS

Nom	Prénom	Contribution
De Barros	Sylvain	Rédacteur
Bernardini	Mickael	Rédacteur

CONTACTS

Nom	Prénom	Email	Fonction
Bernardini	Mickael	mickael.bernardini@gmail.com	Testeur
De Barros	Sylvain	Sylvain099@hotmail.Fr	Testeur

HISTORIQUE DES MODIFICATIONS

Modifications	Date	Version	Approbateur de la diffusion
Ecriture	10/12/2017	1.0	

TABLE DES MATIERES

1. COMMENT INSTALLER

1.1. Installation d'eclipse

Se rendre sur le site http://www.eclipse.org/downloads/eclipse-packages/ pour télécharger eclipse JEE.

1.2. Installation de tomcat et intégration à eclipse

Suivre les instructions sur le site : http://jean-luc.massat.perso.luminy.univ-amu.fr/ens/jee/tp-jsp1.html

2. BEANS

Il y a deux types de Bean, les groupes et les personnes. Un groupe a pour seuls attributs un identifiant et un nom. Le bean person modélise une personne en base de données, il y a donc l'identifiant de celle-ci, le nom, le prénom, l'adresse mail, le site web de la personne, sa date d'anniversaire, son mot de passe et le groupe auquel elle appartient. Une personne ne peut se trouver que dans un seul groupe.

3. IDATABASE

Cette interface fait le lien base de données sans se soucier de l'implémentation sous-jacente. La seule méthode est celle permettant de fournir une connection base de données, peu importe laquelle.

4. IDAOUTILS

Avec cette interface polymorphe nous avons abstrait le traitement base de données des Beans. Elle fournit les méthodes d'ajout et de modification en BDD des Beans. Elle permet aussi de fournir les vues base de données pour les différentes opérations avec les ResultSet comme par exemple la sélection d'une seule entrée ou de filtrer la base selon des paramètres. C'est notamment cette interface qui permet le système de pagination qui fait des appels de type "record" et aussi les fonctions de recherche des groupes et des personnes.

Elle est implémentée par deux classes car nous avons seulement deux beans, les personnes et les groupes. Chaque implémentation définit ses propres requêtes BDD.

5. IDAOPERSON

Cette interface est le point final de modification des beans en base de données. L'implémentation utilise des méthodes privées de types polymorphes qui vont venir s'appuyer sur les implémentations de l'interface IDaoUtils. Ces méthodes permettent l'utilisation des fonctions de recherche, pagination, ajout et modification pour les différents beans. La plupart des méthodes fournies par cette implémentation sont seulement des wrappers utilisant l'abstraction fournie par IDaoUtils. IDaoPerson fournit donc les méthodes de récupération de taille utile à la pagination, les différentes méthodes de recherche sur les groupes et les personnes, et le retour de ceux-ci.

6. DIRECTORYMANAGER

La couche métier rajoute seulement la contrainte que l'utilisateur doit être authentifié pour accéder à toutes les fonctionnalités offertes par l'annuaire de personnes.

7. HASHAGE

Dans le cadre de la sécurisation de la base de données nous avons utilisé l'algorithme de hashage Sha3. Le hashing de mot de passe est une mesure de sécurité qui intervient une fois que la base de données ait été perdue. A partir de ce moment là, la méthode "classique" pour l'attaquant pour retrouver les mots de passes initiaux est une attaque de type brute-force, donc essayer toutes les combinaisons possibles. C'est pour cela qu'un hash lent lui posera plus de soucis qu'un hash rapide.

Keccak est une fonction de hachage cryptographique conçue par Guido Bertoni, Joan Daemen, Michaël Peeters et Gilles Van Assche à partir de la fonction RadioGatún. SHA-3 est issu de la "NIST hash function competition" qui a élu l'algorithme Keccak le 2 octobre 2012. Elle n'est pas destinée à remplacer SHA-2, qui n'a à l'heure actuelle pas été compromise par une attaque significative, mais à fournir une autre solution à la suite des possibilités d'attaques contre les standards MD5, SHA-0 et SHA-1. Keccak est une fonction éponge dans laquelle les blocs du messages sont soumis à un XOR avec des bits initiaux, ensuite permutés de manière réversible. L'algorithme Keccak tel que soumis initialement est utilisé par exemple dans la crypto monnaie Ethereum, il diffère de l'algorithme spécifié par le NIST car il a spécifié la manière de compléter le message lorsque sa longueur n'est pas égale à la taille requise en entrée par la fonction éponge.

8. SPRING

8.1. Connection

Connection est un contrôleur de notre application son rôle est d' authentifier un utilisateur. Pour cela il doit vérifier si l'utilisateur est authentifié. Si cela s'avère faux il lui renvoie une page avec un formulaire pour qu'il puisse entrer un identifiant et un mot de passe. Ensuite il doit vérifier si l'identifiant et le mot de passe sont bien enregistrés en base de données. Si cela est validé il doit sauvegarder son authentification et lui donner accès au reste de l'application en affichant la liste des groupes.

8.2. Groups

Le contrôleur de groupe se charge de lister les groupes présents en base des données. Cette action est disponible seulement pour un utilisateur connecté. S'il ne l'est pas il sera redirigé sur la page de login. Sur la page des groupes sont affichées les fonctions de recherche. Cette liste est affichée avec des paramètres de pagination réglables utilisables en requête GET.

8.3. Persons

Le contrôleur de personnes se charge de lister les personnes présentes en base des données avec l'identifiant du groupe. Cette action est disponible seulement pour un utilisateur connecté. S'il ne l'est pas il sera redirigé sur la page de login. Cette liste est affichée avec des paramètres de pagination réglables utilisables en requête GET.

8.4. Search

Search est un contrôleur de notre application il se charge d'afficher le résultat d'une recherche demandée par l'utilisateur. Il est appelé quand un utilisateur valide un formulaire de recherche que ce soit pour rechercher une personne ou un groupe. Avant d'afficher la liste correspondant à la recherche il vérifie que l'utilisateur est bien authentifié si ce n'est pas le cas il le redirigera vers la page de connection.

9. TEST

9.1. Database

Les classes de tests contenues dans le package database sont les tests unitaires ou des tests de performances sur nos appels base de données. Pour effectuer nos tests unitaires nous avons utilisé dbUnit pour éviter de corrompre notre base de données de production et pour pouvoir vérifier plus précisément nos résultats.

9.2. Métier

Le package métier contient les tests sur notre couche métier. Notre couche métier se basant sur des classes DAO (Database Access Object) nous avons simulé les appels aux classes DAO à l'aide de JMockit. Nos tests vérifient donc que notre couche métier appelle les bonnes fonctions de la couche DAO et ne corrompt pas les résultats.

9.3. Spring

Le package spring contient les tests sur tous nos contrôleurs. Nous avons simulé les variables serveur comme Binding result à l'aide de JMockit. Nous avons aussi simulé les appels à notre couche métier, cette double simulation nous permet de savoir en cas d'erreur que seul notre contrôleur est responsable. Pour tester nos contrôleurs nous avons couvert tous les chemins possibles et vérifié que les pages renvoyées étaient bien celles attendues selon les cas. Nous avons aussi vérifié que l'intégrité des données retournées par nos contrôleurs aux vues JSP correspondent à celles renvoyées par notre couche métier.