TP Spring n°8
Spring Security

Nous avons dans le TP précédent fait l'authentification d'un Client depuis la base de données. Nous allons améliorer cette solution sur plusieurs points :

- La personnalisation de la page de login/logout.
- Le cryptage du mot de passe en base (c'est mieux quand le framework s'appelle Spring Security).
- La définition des rôles.
- La définition des autorisations.

1° - La page de login

1° - Créez dans votre dossier jsp la page de login avec le thème de votre site et le formulaire de login, pour cela utilisez le taglib form :

```
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
```

2° - Ajoutez votre formulaire avec deux *input* pour le *username* et le *password*. Ces champs peuvent être *required*, ce qui évite de faire l'action lors du clic sur le bouton *submit*.

3° - Dans le controller, ajoutez la route /login pour le verbe GET, utilisation de l'annotation *GetMapping* ou alors de *RequestMapping*. Vous n'avez pas besoin de cette route avec le verbe POST car celle-ci sera interceptée automatiquement par la couche de spring security.

4° - Configurez la couche de sécurité en indiquant que la page de login c'est /login, et que la page de redirection en cas de succès, c'est la home soit /.

2° - Cryptage du mot de passe

Le mot de passe est actuellement "en clair" en base…

1° - Trouvez un moyen afin de mettre à jour les mots de passe existant cryptés, pour cela choisissez un algorithme de cryptage. Testez votre application.

Vos mots de passe ne doivent plus être en clair maintenant.

2° - Testez que votre authentification fonctionne avec votre mot de passe.

3° - Création des rôles

1° - Modifiez la configuration de votre base de données afin d'avoir une table *role* dans laquelle les rôles CLIENT et ADMIN sont définis.

L'idéal serait de renommer la table client en table *users* (car user est souvent mal géré par les SGBDs), mais on peut rester ainsi.
Par défaut, un client qui est en base a un rôle CLIENT. Ajoutez la table client_roles qui fait la composition entre un client et un rôle associé. En sachant que de base un client à un rôle CLIENT dans le cas d'un enregistrement (voir plus bas) mais peut aussi être ADMIN.

2° - Modifiez votre classe Client afin de récupérer les rôles de celui-ci.

3° - Testez via l'url de l'api : /api/clients afin de visualiser le Json avec les informations des rôles de utilisateurs.

4° - Authorisation

Oui mais avoir des routes du style /api/clients afin de ressortir tous les clients qui sont en base, c'est bien pour le test mais en réalité c'est vraiment pas top. Car si un client a connaissance de cette route, il peut y accéder, donc c'est quelque chose qui ne doit pas pouvoir se faire.

1° - Faite en sorte que seul un utilisateur avec le rôle ADMIN puisse accéder aux routes de l'api.

2° - Ajoutez une route qui est accessible seulement pour un utilisateur (client ou admin) qui a le username : "Jean-Pierre" par exemple. L'accès n'est pas autorisé pour un autre utilisateur authentifié avec un autre nom.