Cours Spring Security

1 Présentation Générale

- Spring Security a été créé à partir du Framework libre Acegi dont le code source a été donné pour bâtir les fondations de Spring Security.

- Spring Security permet de sécuriser à la fois les invocations de ressources web (par le biais de « *Filtres* » HTTP), ainsi que des appels vers des méthodes en pure Java (par l'utilisation de *Proxy AOP* et de l'@nnotations @Secured(...Role...))

- L'ensemble des paradigmes standards/classiques en terme de sécurité est mis en œuvre par Spring Security, nous verrons dans le cadre de ce TP les principes suivants :

    - Authentification (Qui ?)
      (login, password, rôle, encryption).
      Elle peut se faire par : DAO, LDAP, JAAS (crypto, ...), ...
    - Habilitation (Quoi ?)
      (autorisation des accès aux ressources) par @nnotations ou Tag Library.
    - Configuration d'*Adapter* (Design pattern, famille de « Structuration ») pour le protocole http
      (log in/out, cookie, accessDenied, sessions concurrentes, X509, …).
    - @nnotations de sécurité sur les accès full Java.

- Spring Security utilise très largement le pattern « *Chain of Responsibility* » (famille « *Comportement* ») implémenté par des filtres HTTP, formant les maillons d'une chaine de vérification de droits (le tout déclaré dans le fichier web.xml de configuration principal de l'application).

  Cette chaine va permettre :

    - D'identifier la ressource concernée (appelée « *Candidate* ») par la requête HTTP entrante (en la passant d'ailleurs en lowerCase, FYI).
    - D'utiliser un système (en background) de « *Voters* » qui vont permettre de déterminer si la ressource est atteignable ou non (en fonction des ROLES défini pour la session de Spring Security affectée à l'utilisateur).

- Spring Security propose pour la gestion des droits au sein des pages web une TagLibrary (cf ex dans le TP), qui se charge en fonction des droits en session utilisatrice d'effectuer (ou non) le rendering de portions de page.

Remarques :

- Les logs de Spring Security, bien que très utiles dans le cadre d'un débug peuvent être lourd en terme de charge (tant physique que verbeuse) dans la console, ne pas hésiter à élever le niveau de Log de DEBUG à INFO…

- Spring Security offre un option « Remember me » pour conserver les informations de sessions utilisatrices dans des cookies ou alors dans une base de données (« *data-source-ref* » à déclarer dans le nœud XML « remember-me ».

2 Contenu du TP

- Migration du TP9 JSF + Reporting + Spring AOP, afin d'intégrer le Framework Spring Security :

    - *web.xml* : Ajout de la référence du fichier de context Spring Security + ajout d'un filtre *springSecurityFilterChain* associé au pattern d'URL /*, et pouvant effectuer du dispatch de requête HTTP.
    - *faces-config.xml* : Ajout d'un *PhaseListener* chargé de la gestion de l'authentification des utilisateurs.
    - *applicationContext-security.xml* : Fichier de configuration de Spring Security : déclaration de l'utilisation des *@nnotations @Secured* sur les méthodes Java, configuration de l'adapter du protocole HTTP, gestionnaire d'authentification.
    - *welcome.jsp* : Liens pour le login et le logout, affichage du nom du user logué dans le session sécurisée de Spring, gestion des liens d'accès aux pages selon le niveau d'authentification (minimum pour l'accès à la liste des produits, maximum pour la page des statistiques d'administration) -> utilisation de la TagLibrary.
    - *login.jsp, logoutSuccess.jsp, authenticationFailure.jsp, LoginBean.java* : le formulaire HTTP de login utilise des clés de champs spécifiques (*j_username*, …), les méthodes contrôleurs dispatchent les requêtes HTTP de login vers la chaine de traitement Spring Security.
    - *productList.jsp* : le lien d'accès au produit complet n'est disponible que pour les utilisateurs ayant le rôle ROLE_USER (cf context de sécurité XML et définition des users + rôles), sinon on affiche juste une chaine de caractère (et le navigation vers *productEdit.jsp* est inaccessible) -> utilisation de la TagLibrary.

- *IProductBean.java* : Déclaration d'une @nnotation @Secured sur la méthode *getPdf()*, qui ne sera accessible qu'aux utilisateurs loggués ayant le rôle ROLE_SUPERVISOR, sinon la page accessDenied.jsp (déclaré pour l'*adapter* du protocole HTTP, dans le fichier de sécurité XML, comme page par défaut) est renvoyée.