IUT2A Université Grenoble Alpes Département INFO

R4.01 - TP 06 – Sécurité
1 séance

6.0. Travail à réaliser – Vue d'ensemble

Terminez les TP précédents avant de réaliser cette étape.
L'objectif de cette séance est de mettre en place la sécurité sur votre application en suivant le processus présenté dans le cours 06… Ce sera rapide !

6.1. Commande `make:security` (cours 06 page 12)

- Exécutez la commande `php bin/console make:security:form-login`
- Attention à bien indiquer l'entité utilisée pour l'authentification : `App\Entity\Usager` (c'est parfois demandé par la commande… et parfois non !?)
- Cette commande crée (ou modifie) 3 fichiers qu'il va falloir adapter :
  - `src/Controller/SecurityController.php`
  - `templates/security/login.html.twig`
  - `config/packages/security.yaml`

6.2. Adapter `SecurityController` (cours 06 page 13)

- Si vous aviez auparavant mis en place la traduction sur votre site, modifiez les routes des deux contrôleurs **`login`** et **`logout`** présents dans la classe contrôleur **`SecurityController`** afin que ces routes gèrent l'internationalisation (elles devront avoir un paramètre **`_locale`**)

6.3. Adapter `login.html.twig` (cours 06 page 14)

- Modifiez le *template* **`login.html.twig`** pour que le formulaire d'authentification soit cohérent avec votre site (textes en français, style « bootstrap » pour les champs)

6.4. Adapter le formulaire `login.html.twig` (cours 06 page 15)

- Modifiez le *template* **`login.html.twig`** pour que le formulaire d'authentification soit cohérent avec votre site (textes en français, style « bootstrap » pour les champs)

6.5. Adapter la configuration `security.yml` (cours 06 page 15)

- Définissez la hiérarchie des rôles (**`ROLE_ADMIN`** et **`ROLE_CLIENT`**, même si nous n'utiliserons pas **`ROLE_ADMIN`**)
- Dans la section **`logout`** du firewall **`main`**, définissez la route vers laquelle l'utilisateur sera redirigé après une déconnexion
- Configurez soigneusement la section **`access_control`** afin que les routes suivantes (et seulement elles !) nécessitent une authentification avec un rôle **`ROLE_CLIENT`** :
- **`app_usager_index`** (contrôleur qui affiche la page d'accueil d'un usager authentifié)
- **`app_panier_commander`** (contrôleur qui transforme un panier en commande)

6.6. Refactoring : utiliser l'usager authentifié lors d'une commande

- Modifiez le contrôleur **`PanierController::commander`** afin que ce soit maintenant l'usager authentifié qui se voit attribuer la commande qui est passée (*et non plus par défaut l'usager numéro 1 comme c'était demandé auparavant*)
- Pour accéder à l'usager connecté dans un contrôleur : voir cours 06 page 17

6.7. Adapter l'interface

- Ajoutez un choix « authentification » dans le menu Usager de votre navBar
- Ajoutez un choix « déconnexion » dans le menu Usager de votre navBar
- Modifiez la barre de navigation et vos *templates* pour qu'ils s'adaptent selon qu'un utilisateur est authentifié ou pas :
    - Le bouton « passer commande » du panier ne doit être visible que si l'utilisateur est authentifié
    - Les choix proposés dans le menu Usager de la navBar doivent être activés ou désactivés en fonction de l'authentification
- Pour accéder à l'usager connecté dans un template Twig : voir cours 06 page 17

6.8. Facultatif

S'il vous reste du temps, développez :

- Une page qui permet à l'usager connecté de consulter la liste de ses commandes (accessible depuis le menu Usager de la navBar)

- Une page qui permet à l'usager connecté de consulter le détail d'une de ses commandes (accessible depuis la page ci-dessus)

- Pensez à bien sécuriser ces pages qui ne doivent être accessibles qui si l'usager authentifié possède le `ROLE_CLIENT`