Développement Web 2
TP/TD - N°5

Exercice 1. Objectifs : *Comprendre le mécanisme des sessions*

1. Écrivez un script PHP qui ouvre une session, crée un cookie, et qui affiche ensuite un message de bienvenue.

2. Exécutez ce script, en démarrant d'abord le serveur PHP, et puis en demandant l'exécution du script via le navigateur.

3. Allez chercher les cookies stockés par le navigateur.
   Sur Mac, et avec Chrome, les cookies se trouvent dans le répertoire $\sim/Library/Application\ Support/Google/Chrome/Default/Cookies$.
   Sur Linux, et avec Firefox, les cookies peuvent se trouver dans le répertoire $\sim/.mozilla/firefox/*****.default/Cache$ ou dans le répertoire $\sim/.cache/mozilla/firefox/tsyb1hxz.default/Cache$.
   Les valeurs des cookies sont bien évidemment chiffrées et stockés dans un fichier $sqlite$.

4. Vérifiez que le cookie que vous avez créé a été bien enregistré par le navigateur dans votre répertoire local. Vérifiez également l'existence du cookie de session $PHPSESSID$ depuis $localhost:N°Port$.

Exercice 2. Objectifs : *Savoir utiliser les sessions PHP*

Dans cet exercice, nous allons réaliser un script de validation de formulaires de type $captcha$. Ce mécanisme utilisera les sessions PHP pour stocker une réponse secrète. L'idée est qu'un formulaire (qui pourrait être lui-même dans une vue) pourra contenir une balise input (de type texte) avec une étiquette, demandant une question. L'utilisateur saisit une réponse pour prétendre qu'il n'est pas un robot.

Pour ajouter cet input à un formulaire, on pourra appeler une fonction qui renvoie :

$$Captcha::renderHtml(\$nomDuCaptcha)$$

La fonction implémentera les fonctionnalités suivantes :

- Elle vérifiera si une session a été ouvert, et, si ce n'est pas le cas, elle ouvre cette session.
- Elle choisira, dans une liste de questions/réponses possibles, une demande/réponse spécifique.
- Elle stockera la réponse dans le tableau $_SESSION.
- La demande deviendra l'étiquette d'une balise input de type $text$ et de nom la valeur de la variable.
- Une chaine de caractère, contenant du code $html$ avec un input de type $text$ et de nom la valeur de la variable $\$nomDuCaptcha$, sera retounée.
- Lors de la validation du formulaire, on validera le $captcha$ par la fonction $Captcha::validate(\$nomDuCaptcha)$. Cette fonction retournera un booléen. Elle confrontera la valeur de $\$\_POST[\$nomDuCaptcha]$ avec la valeur stockée dans $_SESSION.
- Organisez ces deux fonctions dans une classe $Captcha$ (comme suggéré par la syntaxe $Captcha::\ldots.$)

Exercice 3. Objectifs : *Prendre en main Symfony - Percevoir le travail nécessaire à la réalisation d'une appli avec ce Framework*

Pour cet exercice, armez-vous des transparents du PDF $ch07\_symfony1.pdf$ et du code du projet contexte (que vous pouvez trouver sur le site du cours). Nous allons réorganiser ce code pour l'adapter au Framework Symfony. Nous ferons cela seulement pour une moitié du projet : l'inscription d'un nouvel utilisateur dans la base de données.

Procédez en suivant exactement ces étapes :

1. Créez un nouvel projet Symfony et paramétrez-le afin qu'il soit possible :
   - d'utiliser la base de données $sqlite$;
   - de définir des routes par des annotations dans le code des contrôleurs ;
   - d'utiliser des $templates\ php$.

2. Récupérez le fichier $create.sql$ depuis le projet $contexte$, et créez depuis ce fichier la base de données $sqlite$.

3. Créez, depuis la base de données les $entités$.

4. Ajoutez les setters (et getters) de toutes les propriétés des classes ainsi crées. Ces classes se trouvent dans le répertoire $src/AppBundle/Entity/$ Si vous utilisez $NetBeans$, vous pouvez trouver les informations sur comment effectuer ces ajouts en suivant ce lien :
   https://stackoverflow.com/questions/35524706/how-to-generate-php-getters-and-setters-in-netbeans

5. Créez un nouvel contrôleur $src/AppBundle/Controller/Users.php$, selon le schéma présenté en cours (n'oubliez pas les déclarations $namespace$ et use).

6. Adaptez le code du contrôleur $contexte/controllers/Users.php$ à Symfony : ajoutez au contrôleur $src/AppBundle/Controller/Users.php$ deux routes, l'une associée à la méthode $user\_new()$, l'autre associée à la méthode $users\_create()$.

7. Réécrivez le code de $user\_new()$ (utilisez la méthode $render()$ au lieu de $loader->load()$).

8. Réécrivez le code de $users\_create()$. Pour cela, prenez l'exemple de la méthode $addRegistration()$ du contrôleur $Registration$ vu en cours :
   - obtenez le $entityManager$,
   - créez un nouvel objet de la classe $User(s)$,
   - « settez » les propriétés de cet objet (on supposera que les données du post ont été filtrés).
   - utilisez le $entityManager$ pour « persister » l'objet.
   - « Flushez » ! ! !

9. Copiez les vues depuis $contexte/controllers/Users.php$ vers (un sous-répertoire de) $app/Resources/views/$ et adaptez les vues.

10. Testez le fonctionnement.