Formation PHP 7
Symfony 4.3

Hajer ALAYA

Hajer.adahmeni@gmail.com

Du 15/07/2019 Au 19/07/2019

Plan

- Notions avancées
  - Sécurité et autorisation
  - Déploiement Sous WAMP

Sécurité et autorisation

Les notions d'authentification et d'autorisation

•**L'authentification** est le processus qui va définir qui vous êtes, en tant que visiteur.

L'autorisation est le processus qui va déterminer si vous avez le droit d'accéder à la ressource (la page) demandée.

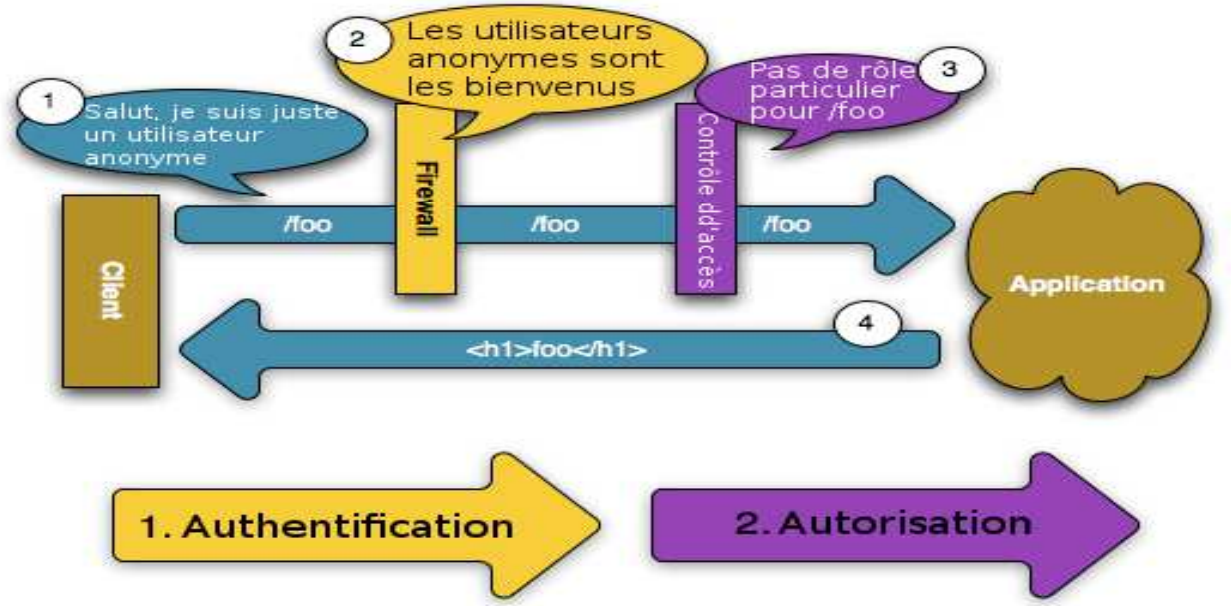


Schéma du processus de sécurité

Source : https://openclassrooms.com/fr/courses/3619856-developpez-votre-site-web-avec-le-framework-symfony/3624755-securite-et-gestion-des-utilisateurs

Sécurité et autorisation

Les notions d'authentification et d'autorisation

Lorsqu'un utilisateur tente d'accéder à une ressource protégée, le processus est finalement toujours le même, le voici :

1. Un utilisateur veut accéder à une ressource protégée ;
2. Le firewall redirige l'utilisateur au formulaire de connexion ;
3. L'utilisateur soumet ses informations d'identification (par exemple login et mot de passe) ;
4. Le firewall authentifie l'utilisateur ;
5. L'utilisateur authentifié renvoie la requête initiale ;
6. Le contrôle d'accès vérifie les droits de l'utilisateur, et autorise ou non l'accès à la ressource protégée.

Sécurité et autorisation

Système de gestion des utilisateurs avec Symfony 4

1. Créer un objet (Entité) User
2. Configuration de la sécurité de Symfony
3. Créer les pages nécessaires
4. Contrôle des accès

Sécurité et autorisation

Système de gestion des utilisateurs avec Symfony 4

1. Créer un objet (Entité) User

- Installer Doctrine

```
composer require symfony/orm-pack
composer require symfony/maker-bundle --dev
```

php bin/console make:user

- Configurer sa base de données

```
# .env
 # Éditez cette ligne
! DATABASE_URL="mysql://username:password@adresseserveur:port/bossiblog«
 # to use sqlite:
 # DATABASE_URL="sqlite:///%kernel.project_dir%/var/app.db"
```

```
php bin/console doctrine:database:create
```

- Créer notre Entité User

Sécurité et autorisation

Système de gestion des utilisateurs avec Symfony 4

1. Créer un objet (Entité) User

- Créer notre Entité User
  - Dans le dossier src\Entity Créer un fichier User.php
  - Configurer le comme suis

```
// src/Entity/User.php
<?php
namespace App\Entity;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Security\Core\User\UserInterface;
use Symfony\Component\Validator\Constraints as Assert;
use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
class User implements UserInterface, \Serializable {
    private $id; private $fullName;private $email;      private $password;
    private $roles = array();
```

```
public function __construct() {
      $this->isActive = true;
      // may not be needed, see section on salt below
      // $this->salt = md5(uniqid('', true));
  }
  public function getUsername() {
      return $this->email;
  }
  public function getSalt() {
      // you *may* need a real salt depending on your encoder
      // see section on salt below
      return null;
  }
  public function getPassword() {
      return $this->password;
  }
  function setPassword($password) {
      $this->password = $password;
  }
  public function getRoles() {
      if (empty($this->roles)) {
          return ['ROLE_USER'];
      }
      return $this->roles;
  }

  function addRole($role) {
```

```
public function __construct() {
      $this->isActive = true;
      // may not be needed, see section on salt below
      // $this->salt = md5(uniqid('', true));
  }

  public function getUsername() {
      return $this->email;
  }

  public function getSalt() {
      // you *may* need a real salt depending on your encoder
      // see section on salt below
      return null;
  }

  public function getPassword() {
      return $this->password;
  }
```

Système de gestion des utilisateurs avec Symfony 4

1. Créer un objet (Entité) User

- Créer notre Entité User
  - Préparer le fichier Yaml de mappage
  - retour à l'invite de commande et lancer :

```
php bin/console doctrine:schema:update --force
```

Cette commande créera votre table utilisateur en base de données

Sécurité et autorisation

Système de gestion des utilisateurs avec Symfony 4

1. Configuration de la sécurité de Symfony

- Activons maintenant la sécurité de Symfony

```
composer require symfony/security-bundle
```

- Maintenant configurons le pare-feu de Symfony. Ceci se fait dans le fichier **security.yaml**

```
security:
  # encoder
   encoders:
      App\Entity\User:
         algorithm: bcrypt

   # https://symfony.com/doc/current/security.html#where-do-users-come-from-user-providers
   providers:
      our_db_provider:
```

Système de gestion des utilisateurs avec Symfony 4

2. Configuration de la sécurité de Symfony

- **form_login** est l'attribut qui dis au pare feu de symfony que vous souhaitez authentifier les utilisateurs via un formulaire
- **login_path** définit la route vers le formulaire de connexion
- **check_path** la route vers le formulaire de vérification

Sécurité et autorisation

Système de gestion des utilisateurs avec Symfony 4

3. Créer les pages nécessaires

- Générer les formulaires

```
composer require symfony/form
composer require validator
```

- Dans le dossier src\Form

```
<?php namespace App\Form;
 use App\Entity\User;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Symfony\Component\Form\Extension\Core\Type\EmailType;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\Extension\Core\Type\RepeatedType;
use Symfony\Component\Form\Extension\Core\Type\PasswordType;
```

```
 class UserType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('fullName', TextType::class)
            ->add('email', EmailType::class)
            ->add('username', TextType::class)
            ->add('password', RepeatedType::class, array(
                'type' => PasswordType::class,
                'first_options'  => array('label' => 'Password'),
                'second_options' => array('label' => 'Repeat Password'),
            ))
        ;
    }

    public function configureOptions(OptionsResolver $resolver)
    {
        $resolver->setDefaults(array(
            'data_class' => User::class,
        ));
    }
```

Système de gestion des utilisateurs avec Symfony 4

3. Créer les pages nécessaires

- **Les routes**
  - Ouvrez votre fichier **routes.yaml** et configurez le comme suit:

```
index:
   path: /
   defaults: { _controller: 'App\Controller\IndexController::index' }
```

Système de gestion des utilisateurs avec Symfony 4

3. Créer les pages nécessaires

```
Admin:
    path: /Admin
    defaults: { _controller: 'App\Controller\IndexController::Admin' }

user_registration:
    path: /register
    defaults: { _controller: 'App\Controller\RegistrationController::registerAction' }

security_login:
     path: /login
     defaults: { _controller: 'App\Controller\SecurityController::login' }
```

Système de gestion des utilisateurs avec Symfony 4

3. Créer les pages nécessaires

- **Les Contrôleurs**
  - Créer IndexController
  - Créer RegistrationController
  - Créer SecurityController

Système de gestion des utilisateurs avec Symfony 4

3. Créer les pages nécessaires

- **Les vues**
  - Construire une vue twig dans votre dossier et appeler la home.html.twig

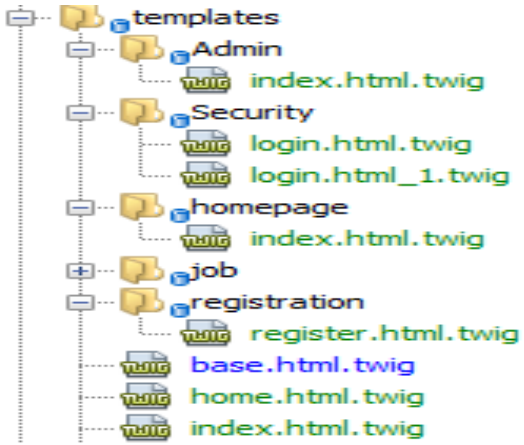
```
<form method="post"  action="{{ path('login') }}" class="home-form">
   <h2 class="form-signin-heading">
    Connection
  </h2>
  {% if error is defined %}
```

  - La Page d'inscription

Système de gestion des utilisateurs avec Symfony 4

3. Créer les pages nécessaires

- **Les vues**
  - Construire les vues twig dans votre dossier



Système de gestion des utilisateurs avec Symfony 4

4. Contrôle des accès

- Ouvre encore le fichier security.yaml et ajouter ces lignes

```
access_control:
   - { path: ^/admin, roles: ROLE_ADMIN }
   - { path: ^/, roles: IS_AUTHENTICATED_ANONYMOUSLY }
   - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
   - { path: ^/security, roles: IS_AUTHENTICATED_ANONYMOUSLY }
```

- Démarrer votre serveur web el lancer http://localhost:8000/:

Déploiement Sous WAMP

- Placer le projet symfony sous le dossier www de Wamp
- Exécuter sous invite dos la commande :

  composer require symfony/apache-pack
- Configuration des « vitual hosts » dans Wamp
  - exemple : configuration du projet Jobeet

    SiteJobeet

    Emplacement c:\wamp\www\jobeet\SiteJobeet

    Host : SiteJobeet

Déploiement Sous WAMP

- Editer le fichier **httpd-vhosts.conf** :
- C:\wamp\bin\apache\Apache2.2.11\conf\extra\httpd-vhosts.conf
- Ajouter le code suivant:

```
#SiteJobeet
<VirtualHost *:80>
    ServerAdmin localhost
    DocumentRoot "${INSTALL_DIR}/www/jobeet/public"
    ServerName SiteJobeet
    ServerAlias SiteJobeet
    ErrorLog "logs/siteA.localhost-error.log"
    CustomLog "logs/siteA.localhost-access.log" common
    Alias /sf "${INSTALL_DIR}/www/jobeet/lib/vendor/symfony/data/web/sf"
</VirtualHost>
```

Déploiement Sous WAMP

•Editer le fichier httpd.conf :

C:\wamp\bin\apache\Apache2.2.11\conf\httpd.conf

•Décommenter la ligne :

#Include conf/extra/httpd-vhosts.conf

•Editer le fichier hosts:

C:\WINDOWS\system32\drivers\etc\hosts

•Ajouter ces 1 ligne :

127.0.0.1 localhost SiteJobeet

Voilà, redémarrez Wamp, et vos sites sont accessibles aux adresses suivantes :

http://SiteJobeet/

Commandes Symfony 4.3


```
//ceration projet symfony
composer create-project symfony/skeleton joboffer
composer create-project symfony/ website-skeleton MonProjet
//demarrer symfony local server
cd joboffer
php -S 127.0.0.1:8000 -t public
//profiler
composer require symfony/profiler-pack --dev
// orm doctrine
composer require symfony/orm-pack
//mapping
php bin/console doctrine:database:create # pour créer la base de données
php bin/console doctrine:schema:create # pour créer la structure des tables
```

```
//fixures
composer config extra.symfony.allow-contrib true
composer require doctrine/doctrine-fixtures-bundle
dev
bin/console doctrine:fixtures:load
//twig
composer require symfony/twig-bundle
"composer require symfony/asset
```

Références

1. Site officiel PHP : https://www.php.net
2. Site Officiel Symfony : https://symfony.com/
3. FastPHP : Cours PHP Accéléré Version 0.9.6-Gérard Rozsa volgyi - mai 30, 2019 www.univ-orleans.fr/.../intra/tuto/php/FastPHP.pdf
4. PHP 7 - Développez un site web dynamique et interactif (2e édition) - Olivier HEURTEL - Date de parution : 09/2018 - ISBN : 978-2-409-01511-3- édition ENI : https://www.editions-eni.fr/open/mediabook.aspx?idR=34e6b661a247bbc86ee4cf0e0f163389
5. PHP 7 Cours et exercices - 2017, Jean Engels -ISBN : 978-2-212-67360-9- édition : Eyrolles https://www.eyrolles.com/Chapitres/9782212673609/9782212673609.pdf
6. Cours PHP 7, Programmation Objet : jan 2019 https://coursinformatiquepdf.com/telecharger-fichier.html
7. IDE PHP https://www.supinfo.com/articles/single/6272-top-10-ide-developpeurs-php
8. Installation wamp : https://alcatiz.developpez.com/tutoriel/installer-wamp-windows10/
9. https://sourceforge.net/projects/wampserver/files/
10. https://www.filehorse.com/download-netbeans/27672/old-versions/
11. Web Technologies – Prof. Dr. Ulrik Schroeder – WS 2010/111

Merci