SYMFONY

24/02/2021

1 PERSONNALISER LA PAGE D'ERREUR

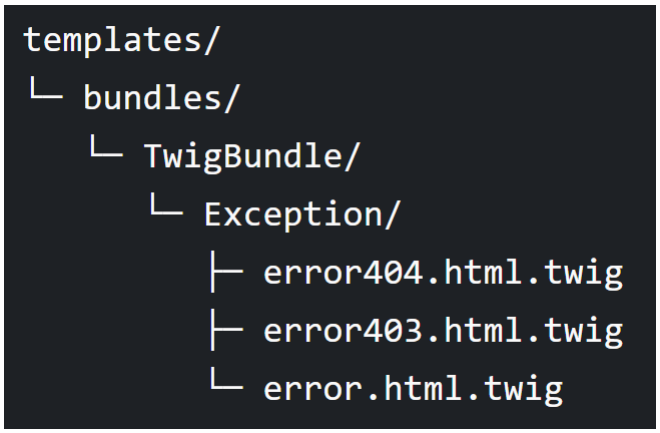
Il est possible d'utiliser TWIG pour personnaliser le message d'erreur. Pour cela il est nécessaire de s'assurer que les deux packages TwigBundle et TwigBridge sont installés :

```
symfony composer require symfony/twig-pack
```

Lorsqu'une erreur se produit Twig utiliser un renderer pour créer un template à afficher à l'utilisateur. Ce renderer utiliser le status code http pour déterminer le nom du template :

- Il cherche un template avec le nom de l'erreur, par exemple error500.html.twig
- S'il ne trouve pas le template du nom de l'erreur il "oublie" le code d'erreur et cherche le fichier d'erreur générique : error.html.twig

On peut surcharger ces templates en créer ses propres fichiers d'erreurs. Ils doivent se trouver dans le répertoire template, dans une arborescence dédiée à Twig :



2 GERER LES AUTORISATIONS DE CONNEXION : LES ROLES

2.1 DEFINITION

Une fois que l'utilisateur est connecté à votre site il faut désormais savoir comment lui autoriser ou lui refuser l'accès à certaines pages.

Ce processus dans Symfony est appelé l'autorisation et son but est de décider si l'utilisateur a le droit d'accéder à certaines ressources (URL).

Le processus d'autorisation se déroule en deux parties :

- L'utilisateur se voit attribué un rôle lorsqu'il se connecte
- On ajoute du code pour protéger certaines ressources (pages) qui requièrent un rôle particulier pour y accéder.

Quand un utilisateur se connecte il reçoit automatiquement le rôle ROLE_USER (voir dans la classe Utilisateur). La liste des rôles est un tableau stocké dans la base de données.

Le rôle ROLE_USER est donc le rôle par défaut de tous les utilisateurs ayant un compte. On peut ajouter d'autres rôles en respectant une règle : chaque rôle doit avoir un nom commençant par ROLE_ , ce que vous indiquez ensuite dépend de votre choix.

A noter que plutôt que de donner plusieurs rôles à un utilisateur vous pouvez mettre en place une hiérarchie de rôles dans le fichier securityyaml, par exemple :

```
role_hierarchy:
    ROLE_ADMIN:       ROLE_USER
    ROLE_SUPER_ADMIN: [ROLE_ADMIN, ROLE_ALLOWED_TO_SWITCH]
```

Cette hiérarchie se définit au niveau juste en dessous de la clé security:

Dans cet exemple un utilisateur ayant le ROLE_ADMIN possède aussi le ROLE_USER, et une utilisateur ayant le ROLE_SUPER_ADMIN possède donc le ROLE_ADMIN, le ROLE_USER (par héritage) et l ROLE_ALLOWED_TO_SWITCH.

2.2 MISE EN PLACE DES AUTORISATIONS

LA mise en place des autorisations peut se faire à différents endroits dans un projet Symfony.

Le premier endroit est le fichier security.yaml dans la section access_control.

Elle permet de définir des URL sous la forme d'expressions régulières qui ne seront accessibles que si l'utilisateur possède un rôle particulier.

```
access_control:
    # require ROLE_ADMIN for /admin*
    - { path: '^/admin', roles: ROLE_ADMIN }

    # or require ROLE_ADMIN or IS_AUTHENTICATED_FULLY for /admin*
    - { path: '^/admin', roles: [IS_AUTHENTICATED_FULLY, ROLE_ADMIN] }

    # the 'path' value can be any valid regular expression
    # (this one will match URLs like /api/post/7298 and /api/comment/528491)
    - { path: ^/api/(post|comment)/\d+$, roles: ROLE_USER }
```

Si on définit plusieurs pattern d'autorisations Symfony les vérifiera un par un en començant par le premier et dès qu'il trouve une correspondance s'arrêtera et appliquera la décision.

```
access_control:
    # matches /admin/users/*
    - { path: '^/admin/users', roles: ROLE_SUPER_ADMIN }

    # matches /admin/* except for anything matching the above rule
    - { path: '^/admin', roles: ROLE_ADMIN }
```

Ci-dessus seul le super admin a accès à l'URL /admin/users mais l'admin et le super admin auront accès à /admin

On peut aussi mettre en place l'authentification au sein du contrôleur soit avec des méthodes, soit avec des annotations.

Avec la méthode

```
$this->denyAccessUnlessGranted("Le Rôle");
```

ou

```
$this->denyAccessUnlessGranted('ROLE_ADMIN', null, Un
utilisateur a essayé d'accéder à la page sans le
ROLE_ADMIN');
```

Cette deuxième méthode laisse un message dans les log pour le responsable du site.

Lorsqu'un utilisateur essaye d'accéder à la ressource protégée il peut se passer deux choses :

Soit il n'est pas connecté, dans ce cas il est redirigé vers la page de connexion.

Soit il est déjà connecté mais n'a pas le droit d'accéder à la page, dans ce cas une error 403 est déclenchée.

On peut aussi utiliser les annotations.

Il faut importer la classe IsGranted :

```
use
Sensio\Bundle\FrameworkExtraBundle\Configuration\IsGranted;
```

Ensuite soit on met l'annotation au-dessus de la définition de la classe, dans ce cas ce sont toutes les ressources du contrôleur qui sont protégées et ont besoin du rôle demandé.

Ou on met l'annotation au-dessus d'une méthode et dans ce cas seule cette ressource est protégée, cela permet d'avoir plus de flexibilité dans la gestion des droits d'accès.

```
/**
 * @IsGranted("ROLE_ADMIN")
 */
```

Il est aussi possible de protéger certaines zones d'un template TWIG en fonction du rôle de l'utilisateur courant :

```
{% if is_granted('ROLE_ADMIN') %}
    <a href="...">Supprimer</a>
{% endif %}
```

Si on veut protéger une page en autorisant seulement les utilisateur connectés à y accéder on peut utiliser le statut IS_AUTHENTICATED_FULLY, il s'utilise comme un rôle mais ne s'attribue pas.

```
$this->denyAccessUnlessGranted('IS_AUTHENTICATED_FULLY');
```

Il existe plusieurs statuts :

- IS_AUTHENTICATED_REMEMBERED: Tous les utilisateurs connectés ont ce statut y compris ceux qui ont utilisé la fonctionnalité se sousvenir de moi (connexion par cookie)
- IS_AUTHENTICATED_FULLY: seuls les utilisateurs qui se sont vraiment connectés (sans utiliser l'authentification par cookie se souvenir de moi) possèdent ce statut
- IS_AUTHENTICATED_ANONYMOUSLY: tous les utilisateurs connectés ou non ont ce statut
- IS_ANONYMOUS: seuls les utilisateurs non connectés ont ce statut
- IS_REMEMBERED: seuls les utilisateurs ayant utilisé l'authentification par cookie (se souvenir de moi) ont ce statut.

Pour finir dans le contrôleur on peut récupérer l'utilisateur grêce à la méthode :

```
$this->getUser()
```

Dans un template :

```
{{ app.user }}
```

Par exemple pour afficher son email :

```
{{ app.user.email }}
```

2.3 NAVIGUER AVEC UN AUTRE UTILISATEUR

Il est parfois intéressant de parcourir le site en tant qu'un autre utilisateur (souvent avec moins de droits).

Pour cela il faut activer dans le firewall l'option switch_user à true.

```
firewalls:
    main:
        # ...
        switch_user: true
```

Ensuite pour afficher une page en tant qu'un autre utilisateur il vous faut ajouter à la fin de votre URL :

```
?_switch_user=username
```

Vous afficherez le site alors comme cet autre utilisateur.

Pour quitter cet utilisateur temporaire il faut ajouter dans l'URL :

```
?_switch_user=_exit
```

On peut aussi ajouter un lien dans le template pour quitter l'identité temporaire :

```
{% if is_granted('IS_IMPERSONATOR') %}
    <a href="{{ impersonation_exit_path(path('homepage') )
}}">Quiiter utilisateur temporaire</a>
{% endif %}
```

2.4 AJOUTER DES ROLES AUX UTILISATEURS

Si vous définissez une hiérarchie de rôles vous pouvez récupérer cette hiérarchie dans le contrôleur ou un formulaire :

Dans le contrôleur :

```
$roles = $this-
>getParameter('security.role_hierarchy.roles');
```

Dans un FormType :

```
$roles = $this->getParent('security.role_hierarchy.roles');
```

Dasn un formulaire de création d'utilisateur vous pouvez ajouter un champs non mappé pour ajouter un ou plusieurs rôles, par exemple :

```
->add('roles', ChoiceType::class, array(
    'attr'  =>  array('class' => 'form-control',
    'style' => 'margin:5px 0;'),
    'choices' =>
    array
    (
        'ROLE_ADMIN' => array
        (
            'Oui' => 'ROLE_ADMIN',
        ),
        'ROLE_PROF' => array
        (
            'Oui' => 'ROLE_TEACHER'
        ),
        'ROLE_ELEVE' => array
        (
            'Oui' => 'ROLE_STUDENT'
        ),
        'ROLE_PARENT' => array
        (
            'Oui' => 'ROLE_PARENT'
        ),
    )
    ,
    'multiple' => true,
    'required' => true,
    )
)
```