Práctica Profesionalizante I

Unidad 10

Seguridad

Agenda

- Usuario
- Autenticación
- Control de acceso

Usuario

- Los permisos están vinculados a un usuario
- Debemos crear la clase Usuario
- Debe implementar UserInterface
- Suele ser una entidad de Doctrine
- Symfony provee el comando make:user

Usuario

```
$ php bin/console make:user
 The name of the security user class (e.g. User) [User]:
 > User

 Do you want to store user data in the database (via Doctrine)? (yes/no) [yes]:
 > yes

 Enter a property name that will be the unique "display" name for the user (e.g. email,
username, uuid) [email]:
 > email

 Will this app need to hash/check user passwords? Choose No if passwords are not needed or
will be checked/hashed by some other system (e.g. a single sign-on server).

 Does this app need to hash/check user passwords? (yes/no) [yes]:
 > yes

 created: src/Entity/User.php
 created: src/Repository/UserRepository.php
 updated: src/Entity/User.php
 updated: config/packages/security.yaml
```

Usuario

- Podemos modificar la entidad y agregar atributos

```
php bin/console make:entity
```

- Debemos crear las tablas en la base de datos

```
php bin/console make:migration
php bin/console doctrine:migrations:migrate
```

Usuario

Se modifica el archivo security.yaml

```
# config/packages/security.yaml
security:
    # ...

    providers:
        app_user_provider:
            entity:
                class: App\Entity\User
                property: email
```

Usuario

Realizar la práctica 1 de la unidad

Generación de contraseñas

- La clase usuario debe implementar la interfaz
  - PasswordAuthenticatedUserInterface
- En security.yaml debemos indicar el hasher

```
# config/packages/security.yaml
security:
    # ...
    password_hashers:
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
```

Generación de contraseñas

```
// src/Controller/RegistrationController.php
namespace App\Controller;

// ...
use Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;

class RegistrationController extends AbstractController
{
    public function index(UserPasswordHasherInterface $passwordHasher)
    {
        // ... e.g. get the user data from a registration form
        $user = new User();
        $user->setName('Robert');
        $plaintextPassword = 'mi_clave';

        // hash the password (based on the security.yaml config)
        $hashedPassword = $passwordHasher->hashPassword(
            $user,
            $plaintextPassword
        );
        $user->setPassword($hashedPassword);

        // ...
    }
}
```

Generación de contraseñas

Podemos codificar manualmente una contraseña ejecutando:

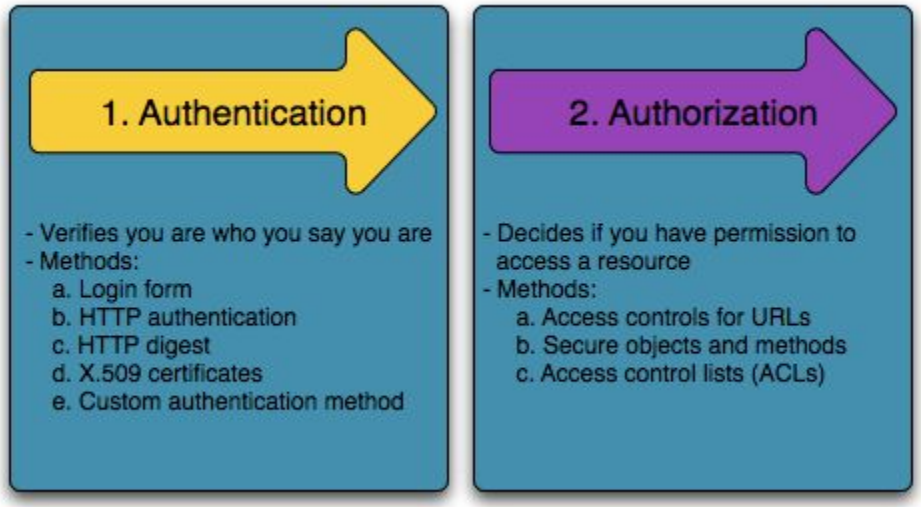
```
php bin/console security:hash-password
```

Usuario

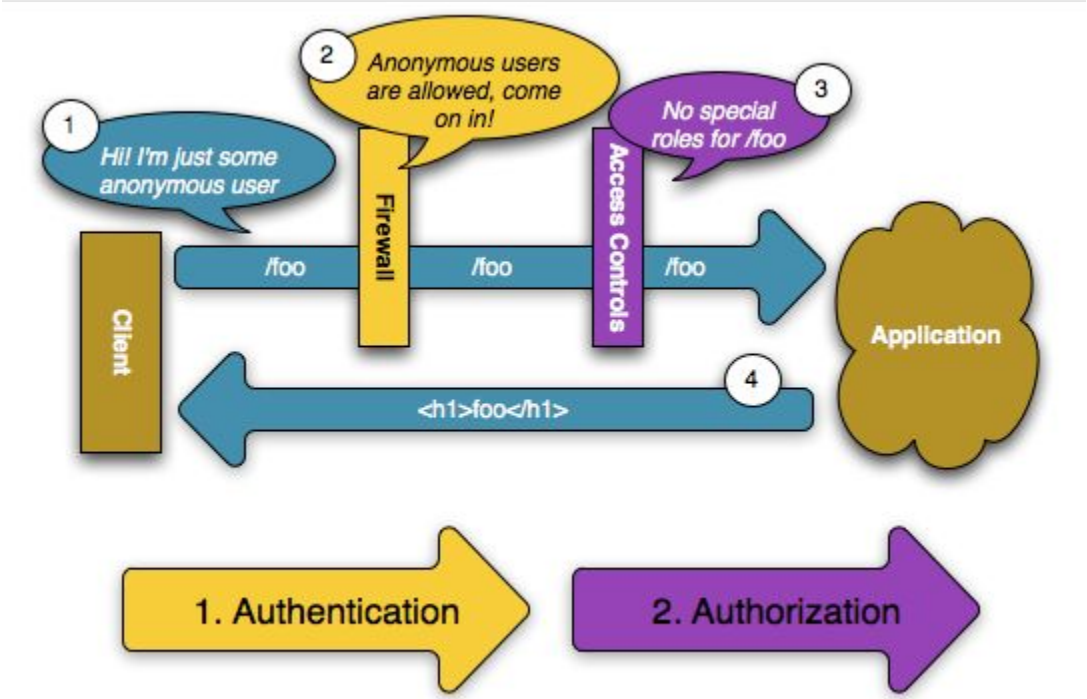
Realizar la práctica 2 de la unidad

Seguridad

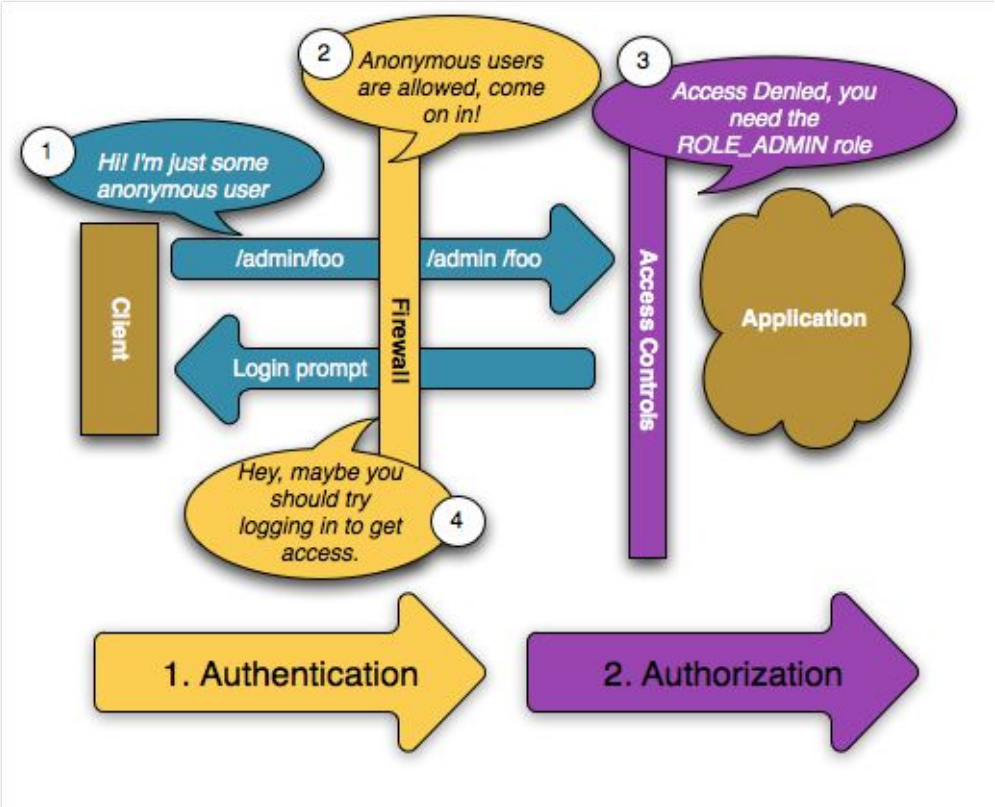
- Proceso de dos etapas



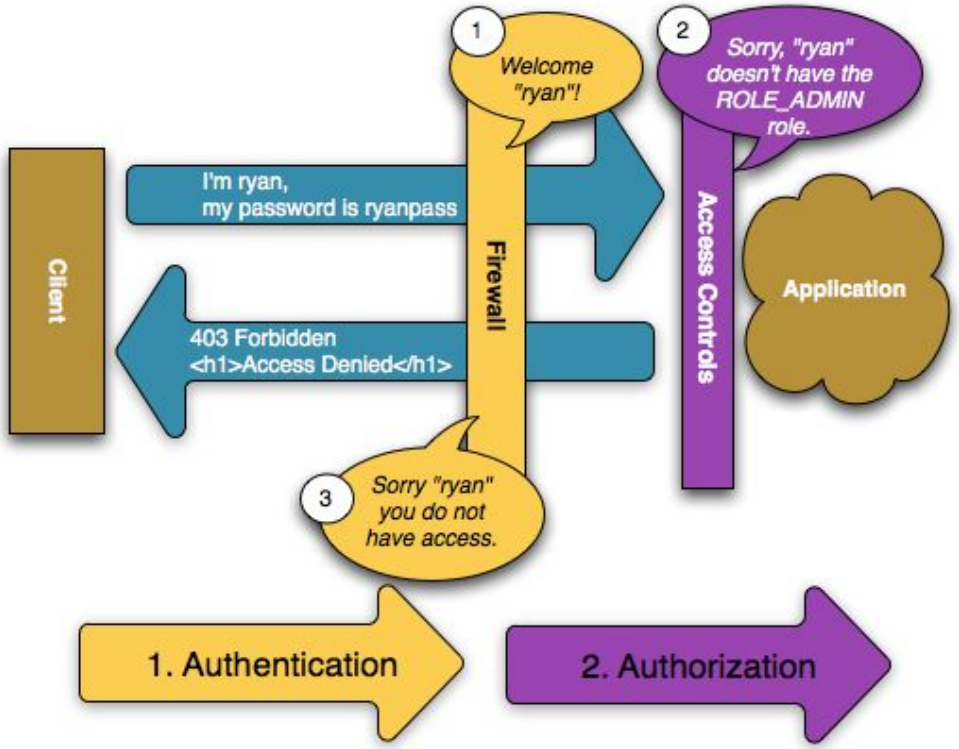
Seguridad


1
Hi! I'm just some anonymous user
2
Anonymous users are allowed, come on in!
3
No special roles for /foo
Firewall
Access Controls
Client
/foo
/foo
/foo
Application
4
<h1>foo</h1>
1. Authentication
2. Authorization

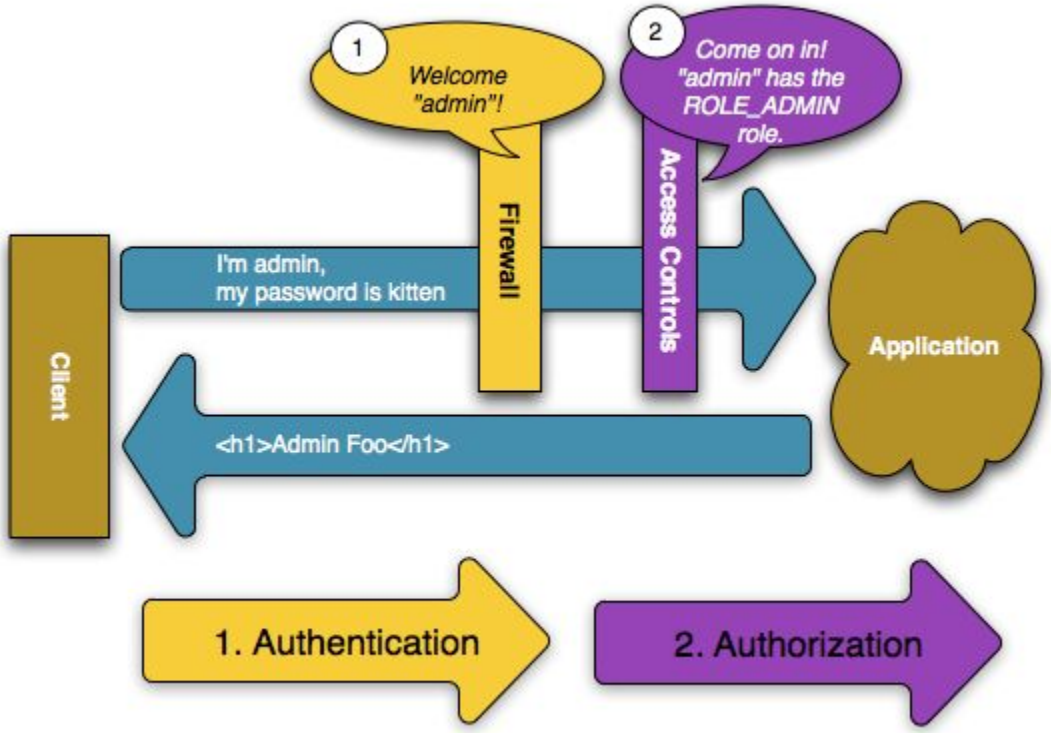
Seguridad


2
Anonymous users are allowed, come on in!
3
Access Denied, you need the ROLE_ADMIN role
1
Hi! I'm just some anonymous user
Client
/admin/foo
/admin /foo
Firewall
Access Controls
Application
Login prompt
Hey, maybe you should try logging in to get access.
4
1. Authentication
2. Authorization

Seguridad


1
Welcome "ryan"!
2
Sorry, "ryan" doesn't have the ROLE_ADMIN role.
Client
I'm ryan, my password is ryanpass
Firewall
Access Controls
Application
403 Forbidden
<h1>Access Denied</h1>
3
Sorry "ryan" you do not have access.
1. Authentication
2. Authorization

Seguridad


1
Welcome "admin"!
2
Come on in! "admin" has the ROLE_ADMIN role.
Firewall
Access Controls
Client
I'm admin, my password is kitten
Application
<h1>Admin Foo</h1>
1. Authentication
2. Authorization

Seguridad

```
# config/packages/security.yaml
security:
      password_hashers:
            Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface:
'auto'
      providers:
            app_user_provider:
                  entity:
                        class: App\Entity\Usuario
                        property: email
      firewalls:
            dev:
                  pattern: ^/(_(profiler|wdt)|css|images|js)/
                  security: false
            main:
                  lazy: true
                  provider: app_user_provider
      access_control:
            - { path: ^/login, roles: PUBLIC_ACCESS }
            - { path: ^/, roles: ROLE_USER }
```

Autenticación
Cortafuegos

```
# config/packages/security.yaml
security:
    # ...
    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            pattern: ^/
            provider: app_user_provider
```

Autenticación
Formulario de login

```
# config/packages/security.yaml
security:
    # ...
    firewalls:
        # ...
        main:
            pattern: ^/
            provider: app_user_provider
            form_login:
                login_path: app_login
                check_path: app_login
```

Formulario de login

```
// src/Controller/LoginController.php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\Security\Http\Authentication\AuthenticationUtils;

class LoginController extends AbstractController
{
    #[Route('/login', name: 'app_login')]
    public function login(AuthenticationUtils $authenticationUtils): Response
    {
        // get the login error if there is one
        $error = $authenticationUtils->getLastAuthenticationError();

        // last username entered by the user
        $lastUsername = $authenticationUtils->getLastUsername();

        return $this->render('security/login.html.twig', [
            'last_username' => $lastUsername,
            'error'         => $error,
        ]);
    }
}
```

Formulario de login

```
{# templates/security/login.html.twig #}
{% extends 'base.html.twig' %}

{# ... #}

{% block body %}
    {% if error %}
        <div>Usuario o contraseña incorrecta</div>
    {% endif %}

    <form action="{{ path('app_login') }}" method="post">
        <label for="username">Email:</label>
        <input type="text" id="username" name="_username" value="{{ last_username }}"/>

        <label for="password">Password:</label>
        <input type="password" id="password" name="_password"/>

        <button type="submit">login</button>
    </form>
{% endblock %}
```

Autenticación

Realizar la práctica 3 de la unidad

Autorización
Control de acceso

```
# config/packages/security.yaml
security:
    # ...

    firewalls:
        # ...
        main:
            # ...

    access_control:
        - { path: ^/login, roles: PUBLIC_ACCESS }
        - { path: '^/admin', roles: ROLE_ADMIN }
        - { path: '^/premium', roles: [ROLE_ADMIN, ROLE_PREMIUM] }
        - { path: ^/api/(post|comment)/\d+$, roles: ROLE_USER }
```

Autorización
Control de acceso

```php
// src/Entity/User.php

// ...
class User
{
    #[ORM\Column]

    private $roles = [];

    // ...
    public function getRoles(): array
    {
        $roles = $this->roles;
        // guarantee every user at least has ROLE_USER
        $roles[] = 'ROLE_USER';

        return array_unique($roles);
    }
}
```

Autorización

Realizar la práctica 4 de la unidad

Logout

```
# config/packages/security.yaml
security:
    # ...

    firewalls:
        main:
            # ...
            logout:
                path: app_logout
```

Logout

```
#[Route('/logout', name: 'app_logout')]
public function logout(): void
{
    // controller can be blank: it will never be called!
 throw new \Exception('Activate logout in security.yaml');
}
```

Logout

Realizar la práctica 5 de la unidad

Acceder al objeto usuario

```
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

class ProfileController extends AbstractController
{
    public function index(): Response
    {
        // returns your User object, or null if the user is not authenticated
        $user = $this->getUser();

        // Call whatever methods you've added to your User class
        // For example, if you added a getFirstName() method, you can use that.
        return new Response('Well hi there '.$user->getFirstName());
    }
}
```

Acceder al objeto usuario

```
{% if is_granted('IS_AUTHENTICATED_FULLY') %}
    <p>Email: {{ app.user.email }}</p>
{% endif %}
```

Acceder al objeto usuario

Realizar la práctica 6 de la unidad