TP noté symfony Timéo JACCOD TD1

Étape 1 : Initialisation du projet

```
>> C:\Users\timus\OneDrive\Documents\symfony\my_reservation_app>
Created database `symfony` for connection named default
```

Étape 2 : Création des entités

```
Do you want to automatically delete orphaned App\Entity\Reservation objects (orphanRemoval)? (yes/no) [no]:
> yes

updated: src/Entity/Reservation.php
updated: src/Entity/User.php

Add another property? Enter the property name (or press <return> to stop adding fields):
>
```

```php
#[ORM\Entity(repositoryClass: ReservationRepository::class)]
class Reservation
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(type: Types::DATETIME_MUTABLE)]
    private ?\DateTimeInterface $date = null;

    #[ORM\Column(length: 50)]
    private ?string $timeSlot = null;

    #[ORM\Column(length: 255)]
    private ?string $eventName = null;

    #[ORM\ManyToOne(inversedBy: 'reservations')]
    #[ORM\JoinColumn(nullable: false)]
    private ?User $user = null;

    public function getId(): ?int
    {
        return $this->id;
    }
```

```php
#[ORM\Entity(repositoryClass: UserRepository::class)]
class User
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(length: 180, nullable: true)]
    private ?string $email = null;

    #[ORM\Column(length: 255)]
    private ?string $password = null;

    #[ORM\Column]
    private array $roles = [];

    #[ORM\Column(length: 255)]
    private ?string $name = null;

    #[ORM\Column(length: 20)]
    private ?string $phoneNumber = null;
```

Étape 3 : Authentification et rôles

```yaml
security:
    # Remplacer 'encoders' par 'password_hashers' pour les versions récentes de Symfony
    password_hashers:
        App\Entity\User:
            algorithm: bcrypt

    # Configuration des firewalls
    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false

        main:
            form_login:
                login_path: login
                check_path: login
                username_parameter: email
                password_parameter: password
            logout:
                path: app_logout
            stateless: false  # Permet d'autoriser les utilisateurs non authentifiés

    # Rôles d'accès
    access_control:
        - { path: ^/admin, roles: ROLE_ADMIN }
        - { path: ^/user, roles: ROLE USER }
```

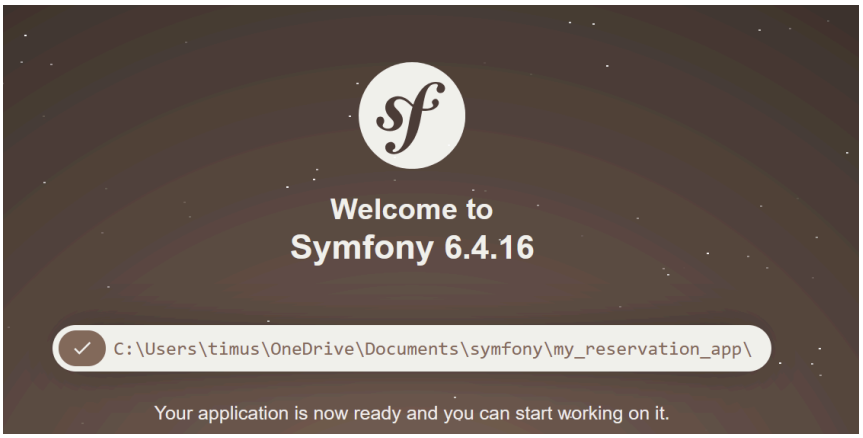
```php
// src/Controller/SecurityController.php
namespace App\Controller;

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

class SecurityController extends AbstractController
{
    /**
     * @Route("/login", name="login")
     */
    public function login(AuthenticationUtils $authenticationUtils)
    {
        // Récupère l'erreur de login s'il y en a
        $error = $authenticationUtils->getLastAuthenticationError();
        $lastUsername = $authenticationUtils->getLastUsername();

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

    /**
     * @Route("/logout", name="logout")
     */
    public function logout()
    {
        // Cette méthode ne sera jamais appelée
        // Car Symfony gère automatiquement la déconnexion
    }
```


Welcome to
Symfony 6.4.16
C:\Users\timus\OneDrive\Documents\symfony\my_reservation_app\
Your application is now ready and you can start working on it.

Étape 4 : Règles métier et validations

```php
namespace App\Entity;

use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Validator\Constraints as Assert;

#[ORM\Entity]
class Reservation
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column(type: 'integer')]
    private ?int $id = null;

    #[ORM\Column(type: 'datetime')]
    #[Assert\NotBlank(message: "La date est obligatoire.")]
    #[Assert\GreaterThan('now', message: "La réservation doit être faite au moins 24 heures à l'avance.")]
    private ?\DateTimeInterface $date = null;

    #[ORM\Column(type: 'string', length: 50)]
    #[Assert\NotBlank(message: "La plage horaire est obligatoire.")]
    private ?string $timeSlot = null;

    #[ORM\Column(type: 'string', length: 255)]
    #[Assert\NotBlank(message: "Le nom de l'événement est obligatoire.")]
    private ?string $eventName = null;

    #[ORM\ManyToOne(targetEntity: User::class)]
    #[ORM\JoinColumn(nullable: false)]
    private ?User $user = null;
```

```php
// src/Repository/ReservationRepository.php

namespace App\Repository;

use App\Entity\Reservation;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

class ReservationRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Reservation::class);
    }

    /**
     * Vérifie si une plage horaire est disponible
     */
    public function isTimeSlotAvailable(\DateTimeInterface $date, string $timeSlot): bool
    {
        $query = $this->createQueryBuilder('r')
            ->andWhere('r.date = :date')
            ->andWhere('r.timeSlot = :timeSlot')
            ->setParameter('date', $date)
            ->setParameter('timeSlot', $timeSlot)
            ->getQuery();

        $result = $query->getOneOrNullResult();

        return $result === null;  // Retourne vrai si aucune réservation n'existe
    }
}
```

```php
// src/Controller/ReservationController.php

namespace App\Controller;

use App\Entity\Reservation;
use App\Form\ReservationType;
use App\Repository\ReservationRepository;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class ReservationController extends AbstractController
{
    /**
     * Affiche la liste des réservations de l'utilisateur connecté
     *
     * @Route("/user/reservations", name="reservation_list")
     */
    public function list(ReservationRepository $repository): Response
    {
        $reservations = $repository->findBy(['user' => $this->getUser()]);

        return $this->render('reservation/list.html.twig', [
            'reservations' => $reservations,
        ]);
    }
```

Étape 5 : Gestion des fonctionnalités

```twig
{# templates/user/list.html.twig #}

{% extends 'base.html.twig' %}

{% block body %}
    <h1>Liste des utilisateurs</h1>

    <table>
        <thead>
            <tr>
                <th>Email</th>
                <th>Nom</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
            {% for user in users %}
                <tr>
                    <td>{{ user.email }}</td>
                    <td>{{ user.name }}</td>
                    <td>
                        <a href="{{ path('user_show', {'id': user.id}) }}">Voir</a>
                        <a href="{{ path('user_edit', {'id': user.id}) }}">Modifier</a>
                        <a href="{{ path('user_delete', {'id': user.id}) }}">Supprimer</a>
                    </td>
                </tr>
            {% endfor %}
        </tbody>
    </table>

    <a href="{{ path('user_edit', {'id': null}) }}">Ajouter un utilisateur</a>
{% endblock %}
```

	Table	Action	Lignes	Type	Interclassement	Taille	Perte
☐	doctrine_migration_versions	Parcourir Structure Rechercher Insérer Vider Supprimer	1	InnoDB	utf8_unicode_ci	16,0 kio	-
☐	reservation	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
☐	user	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	16,0 kio	-
	3 tables	Somme	1	InnoDB	utf8mb4_general_ci	64,0 kio	0 o