



TEK-UP Ecole Supérieure Privée Technologie &
Ingénierie

Ateliers Framework (**Symfony 5**)

Wisseem ELJAOUED
wisseem.eljaoued@ensi-uma.tn

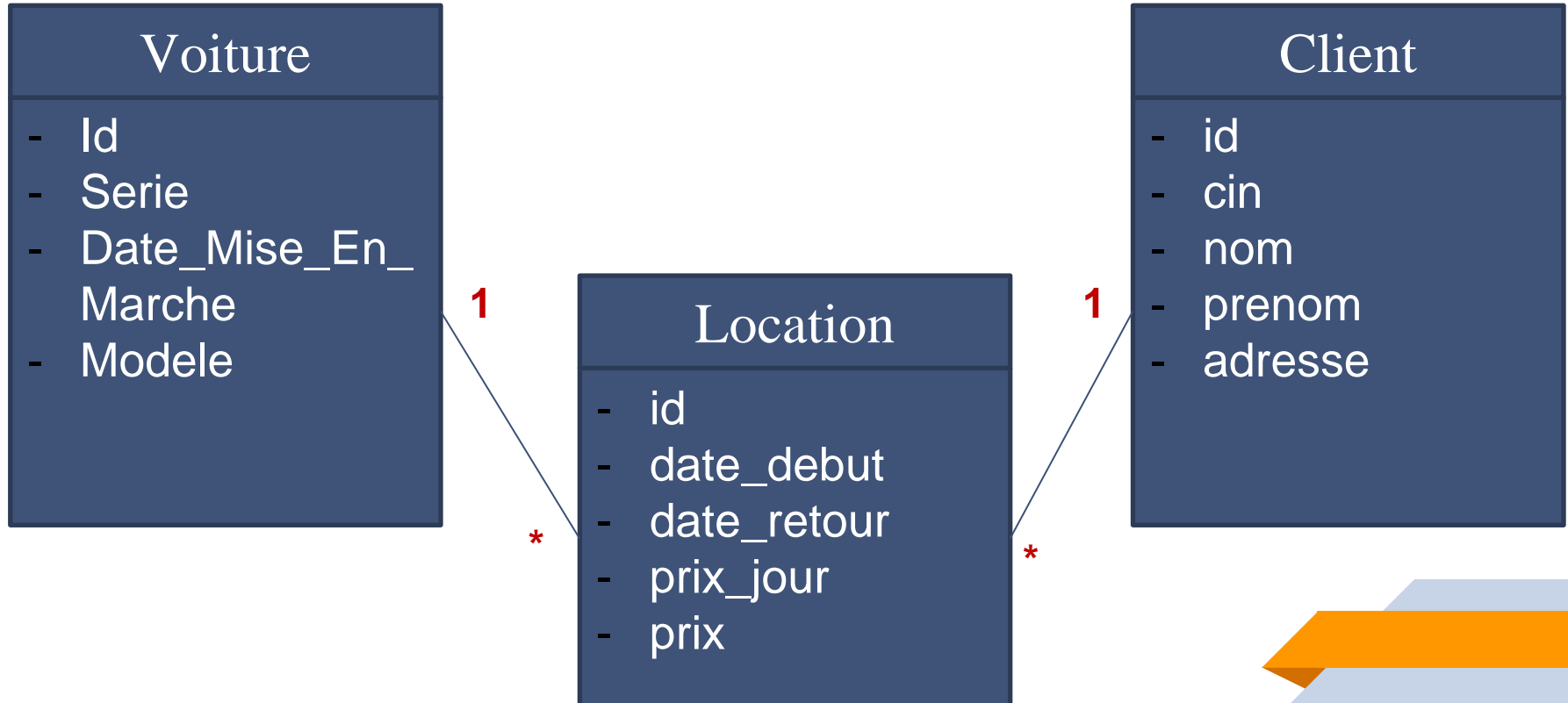
A.U. 2021-2022

Atelier 4

Doctrine

I.1. Etude de cas

- Nous allons travailler pendant ce TP avec l'étude de cas suivant:
 - « Location de voiture » représentée avec son diagramme de classe:



I.2. Création des entités

Maintenant nous devons créer nos **classes** sous le répertoire **Entity**

- Les attributs sont enrichis avec des **annotations** qui vont aider l'ORM à créer l'équivalent de la classe en table dans la base de données.
- Les annotations vont spécifier les types, les clés,...

Il faut générer les **getters** et les **setters** de tout les attributs (Menu>Code>Generate...)

Deux méthodes de création des entités:

- Manuellement (new PHP Class)
- La commande: **make:entity**

I.3. Configuration BD

- Les accès à la base de données ainsi qu'au serveur de messagerie sont centralisés dans le fichier App\env

```
DATABASE_URL="mysql://root:@127.0.0.1:3306/workshop1?serverVersion=5.7"
```

Serveur de BD

DB user

DB name

I.4. Génération de la BD

- Pour faire le mapping entre les entités et la base de données, nous devons créer des migrations à l'aide de la commande suivante:

`make:migration`

- Deuxièmement, exécuter ces migrations :

`doctrine:migrations:migrate`

En cas d'erreur, changer l'URL de la base de données dans le fichier `.env` :

- en éliminant la version du serveur comme suit:

```
DATABASE_URL="mysql://root:@127.0.0.1:3306/workshop1"
```

- Ou préciser la version exacte de votre serveur:

```
DATABASE_URL="mysql://root:@127.0.0.1:3306/workshop1?serverVersion=mariadb-10.4.14"
```

I.4. Génération de la BD

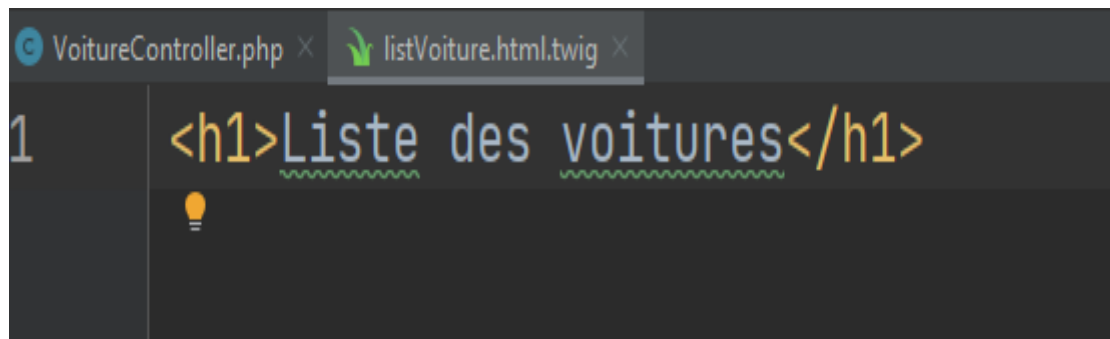
- Ajouter des voitures manuellement dans la table **Voiture**

 ▼				id	serie	date_mise_circulation	modele
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	1	1234	2021-08-10	BMW
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	2	5678	2021-07-12	FORD
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	3	9123	2020-12-08	AUDI

- Créer un contrôleur **VoitureController**
- Ajouter l'action **listVoiture**

```
/**
 * @Route("/voiture", name="voiture")
 */
public function listVoiture(): Response
{
    return $this->render(view: 'voiture/listVoiture.html.twig', [
    ]);
}
```


- Créer un dossier nommé **voiture** sous le répertoire **Templates**
- Créer le fichier **listVoiture.html.twig** sous le répertoire **voiture**



The screenshot shows a code editor with two tabs: 'VoitureController.php' and 'listVoiture.html.twig'. The 'listVoiture.html.twig' tab is active, displaying the following HTML code:

```
1 <h1>Liste des voitures</h1>
```

The code is written in a dark-themed editor. The text 'Liste des voitures' is underlined with a green wavy line, indicating a syntax or structure issue. A lightbulb icon is visible below the code, suggesting a suggestion or warning.

- Il faut charger la liste des voitures de la base de données et ceci dans l'action `listVoiture`

```
public function listVoiture(): Response
{
    $em= $this->getDoctrine()->getManager();
    $voitures = $em->getRepository( className: "App\Entity\Voiture")->findAll();

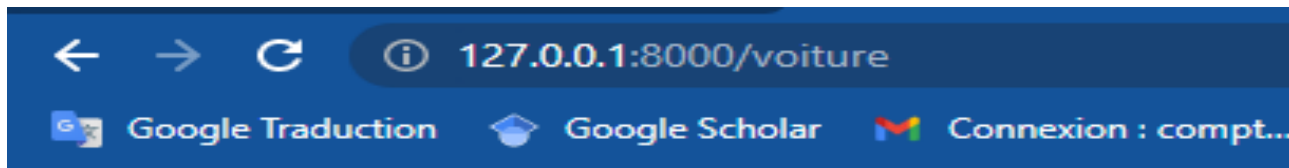
    return $this->render( view: 'voiture/listVoiture.html.twig', [
        "listeVoiture"=>$voitures
    ]);
}
```

Créer une
instance
de l'ORM

Récupérer
la liste de
tous les
modèles

- Maintenant nous allons personnaliser l'affichage de la liste dans la vue

```
<h1>Liste des voitures</h1>
<table border="1">
  <tr>
    <td>Serie</td>
    <td>Date M C</td>
    <td>Modele</td>
  </tr>
  {% for v in listeVoiture %}
    <tr>
      <td>{{ v.serie }}</td>
      <td>{{ v.dateMiseCirculation | date }}</td>
      <td>{{ v.modele }}</td>
    </tr>
  {% endfor %}
</table>
```



Liste des voitures

Serie	Date M C	Modele
1234	August 10, 2021 00:00	BMW
5678	July 12, 2021 00:00	FORD
9123	December 8, 2020 00:00	AUDI