TEK-UP Ecole Supérieure Privée Technologie & Ingénierie

Ateliers Frameworfi (Symfony 6)

Jasser Jammeli
Jasserjammeli98@gmail.com

A.U. 2025-2026

Atelier 5

DQL

I.1. DQL

1. **Ajouter une nouvelle entité « Modele » en relation avec l'entité « Voiture ». Un Modèle possède plusieurs Voiture et une Voiture possède un seul Modèle. Un modèle est caractérisé par un libellé et un pays.**

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

    2 usages
    #[ORM\Column(length: 255, nullable: true)]
    private ?string $libelle = null;

    2 usages
    #[ORM\Column(length: 255, nullable: true)]
    private ?string $pays = null;

    /**
     * @var Collection<int, Voiture>
     */
    5 usages
    #[ORM\OneToMany(targetEntity: Voiture::class, mappedBy: 'modele')]
    private Collection $voitures;

    public function __construct()
    {
        $this->voitures = new ArrayCollection();
    }
```

I.1. DQL

add

```php
class ModeleRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Modele::class);
    }

    // + CREATE
    1 usage
    public function addModele(string $libelle, string $pays): Modele
    {
        $em = $this->getEntityManager();
        $modele = new Modele();
        $modele->setLibelle($libelle);
        $modele->setPays($pays);

        $em->persist($modele);
        $em->flush();

        return $modele;
    }
```

```php
#[Route('/modeles')]
final class ModeleController extends AbstractController
{
    // + CREATE : ajout direct d'un modèle
    #[Route('/add', name: 'modele_add')]
    public function add(ModeleRepository $repo): Response
    {
        $modele = $repo->addModele( libelle: 'Clio',  pays: 'France');

        return new Response( content: '✅ Modèle ajouté avec ID : ' . $modele->getId());
    }
```

I.1. DQL

Select

```php
1 usage
public function findAllModeles(): array
{
    $em = $this->getEntityManager();
    $query = $em->createQuery( dql: '
        SELECT m
        FROM App\Entity\Modele m
        ORDER BY m.libelle ASC
    ');
    return $query->getResult();
}
```

```php
// 📖 READ : liste tous les modèles
#[Route('/list', name: 'modele_list')]
public function list(ModeleRepository $repo): Response
{
    $modeles = $repo->findAllModeles();

    // Affichage simple dans la page
    $output = "<h2>Liste des modèles</h2><ul>";
    foreach ($modeles as $m) {
        $output .= "<li>ID: {$m->getId()} | Libellé: {$m->getLibelle()} | Pays: {$m->getPays()}</li>";
    }
    $output .= "</ul>";

    return new Response($output);
}
```

I.1. DQL

Update

```php
public function updateModele(int $id, string $libelle, string $pays): int
{
    $em = $this->getEntityManager();
    $query = $em->createQuery( dql: '
        UPDATE App\Entity\Modele m
        SET m.libelle = :libelle,
            m.pays = :pays
        WHERE m.id = :id
    ')
        ->setParameter( key: 'libelle', $libelle)
        ->setParameter( key: 'pays', $pays)
        ->setParameter( key: 'id', $id);

    return $query->execute();
}
```

Update

```php
#[Route('/update/{id}', name: 'modele_update')]
public function update(ModeleRepository $repo, int $id): Response
{
    // Mise à jour du libellé et du pays
    $rows = $repo->updateModele($id, libelle: 'Megane', pays: 'France');

    return new Response( content: "✏️ Modèle mis à jour : {$rows} ligne(s).");
}
```

I.1. DQL

Delete

```
1 usage
public function deleteModele(int $id): int
{
    $em = $this->getEntityManager();
    $query = $em->createQuery( dql: '
        DELETE FROM App\Entity\Modele m
        WHERE m.id = :id
    ')
        ->setParameter( key: 'id', $id);

    return $query->execute();
}
```

```
#[Route('/delete/{id}', name: 'modele_delete')]
public function delete(ModeleRepository $repo, int $id): Response
{
    $rows = $repo->deleteModele($id);

    return new Response( content: "🗑 Modèle supprimé : {$rows} ligne(s).");
}
```

I.1. DQL

1. **Ajouter les modifications nécessaires au niveau du formulaire d'ajout d'une voiture puis ajouter des nouveaux voitures avec l'url /addVoitures.**

```php
class VoitureForm extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder->add( child: 'serie',  type: TextType::class)
            ->add( child: 'dateMiseEnMarhe',  type: DateType::class)
            ->add( child: 'modele',  type: EntityType::class, [
                'class' => Modele::class,
                'choice_label' => 'libelle', // affichera le libellé du modèle dans la liste
                'placeholder' => 'Sélectionner un modèle',
            ])
            ->add( child: 'prixJour',  type: NumberType::class);

    }

    no usages
    public function getName(){
        return 'voiture';
    }

}
```

2. Créer une nouvelle méthode searchVoiture dans le VoitureRepository.php

```php
1 usage
public function findByModele(int $modeleId): array
{
    $em = $this->getEntityManager();

    $query = $em->createQuery( dql: '
    SELECT v, m
    FROM App\Entity\Voiture v
    JOIN v.modele m
    WHERE m.id = :modeleId
    ORDER BY v.serie ASC
')
        ->setParameter( key: 'modeleId', $modeleId);

    return $query->getResult();
}
```

2. Créer une nouvelle méthode voitureParModele dans le VoitureController.php

```php
#[Route('/voitures-par-modele', name: 'voitures_par_modele')]
public function voituresParModele(Request $request, VoitureRepository $vr,
                                  EntityManagerInterface $em): Response
{
    $modeleId = $request->query->get( key: 'modele'); // Récupère le modèle choisi via GET
    $voitures = [];

    if ($modeleId) {
        $voitures = $vr->findByModele((int)$modeleId);
    }

    // Récupérer tous les modèles pour le select
    $modeles = $em->getRepository(Modele::class)->findAll();

    return $this->render( view: 'voiture/voituresParModele.html.twig', [
        'voitures' => $voitures,
        'modeles' => $modeles,
        'selectedModele' => $modeleId
    ]);
}
```

I.2. DQL

filtrage

```
<h1>Voitures par modèle</h1>

<form method="get">
    <select name="modele">
        <option value="">-- Sélectionner un modèle --</option>
        {% for m in modeles %}
            <option value="{{ m.id }}" {% if m.id == selectedModele %}selected{% endif %}>{{ m.libelle }}</option>
        {% endfor %}
    </select>
    <button type="submit">Filtrer</button>
</form>

{% if voitures %}
    <h2>Résultats :</h2>
    <ul>
        {% for v in voitures %}
            <li>{{ v.serie }} | {{ v.dateMiseEnMarhe|date('Y-m-d') }} | {{ v.modele.libelle }} | {{ v.prixJour }}€</li>
        {% endfor %}
    </ul>
{% elseif selectedModele %}
    <p>Aucune voiture trouvée pour ce modèle.</p>
{% endif %}
```

I.2. DQL

filtrage