Module **PWS3** **ORM Symfony suite**

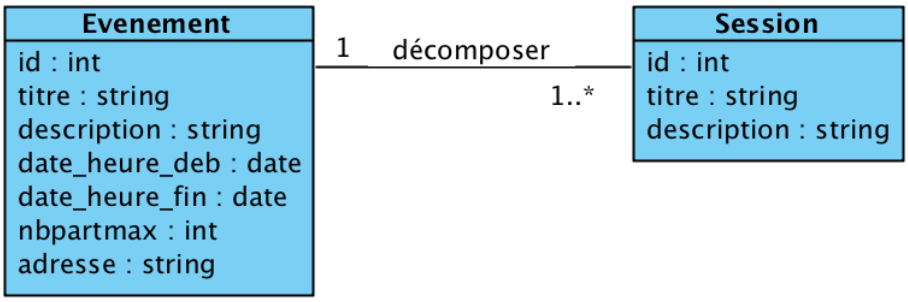
Les relations entre entités avec doctrine 2

http://symfony.com/doc/current/doctrine.html

Dans une relation entre deux entités dans doctrine, il y a toujours une entité propriétaire et une entité inverse.

Etape 1 : Mise en place d'une relation Many to One

Soit le modèle suivant :



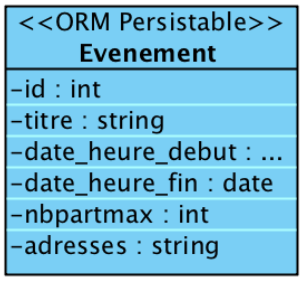
Ce qui se traduirait en schéma relationnel:

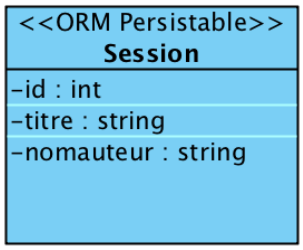
EVENEMENT (id, titre, description, date_heure_deb, date_heure_fin, nbpartmax, adresse)
Clé primaire : id

SESSION (id, titre, description, id_evenement)
Clé primaire : id

Dépendance de référence : SESSION.id_evenement Ref EVENEMENT.id

Et en entités ORM :





L'entité propriétaire est celle qui contiendrait l'attribut sur lequel portera la dépendance de référence vers l'autre entité l'autre entité. Notez que cet attribut id_evenement n'est pas dans l'entité c'est doctrine qui va gérer.
Entité Propriétaire : SESSION
Entité Inverse : EVENEMENT
Cette relation est un exemple de relation de type Many to One qui lie une entité A à plusieurs entités B.
Ici on peut ajouter plusieurs sessions à un événement mais une session n'est liée qu'à un seul événement, mais pourra éventuellement changer d'évènement.

https://www.doctrine-project.org/projects/doctrine-orm/en/2.7/reference/working-with-associations.html

Créez l'entité session avec le générateur : php bin/console make:entity

Ajoutez l'attribut privé $evenement de type relation Many to One

La première annotation met en place la relation many to one vers l'entité Evenement.

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

 Field type (enter ? to see all types) [string]:
 > relation

 What class should this entity be related to?:
 > Evenement

What type of relationship is this?
 ------------ ---------------------------------------------------------------------
  Type         Description
 ------------ ---------------------------------------------------------------------
  ManyToOne    Each Session relates to (has) one Evenement.
               Each Evenement can relate to (can have) many Session objects

  OneToMany    Each Session can relate to (can have) many Evenement objects.
               Each Evenement relates to (has) one Session

  ManyToMany   Each Session can relate to (can have) many Evenement objects.
               Each Evenement can also relate to (can also have) many Session objects

  OneToOne     Each Session relates to (has) exactly one Evenement.
               Each Evenement also relates to (has) exactly one Session.
 ------------ ---------------------------------------------------------------------

 Relation type? [ManyToOne, OneToMany, ManyToMany, OneToOne]:
 > ManyToOne

 Is the Session.evenement property allowed to be null (nullable)? (yes/no) [yes]:
 > no
```

Mise en place de l'association bidirectionnelle, on ne mettra pas en place la suppression des orphelins car une session peut basculer d'un évènement à un autre évènement.
Pour plus d'information sur ce point consultez : https://www.doctrine-project.org/projects/doctrine-orm/en/2.7/reference/working-with-associations.html

```
Do you want to add a new property to Evenement so that you can access/update Session objects from it - e.g. $evenement->getSessions()? (yes/no) [yes]:
>

A new property will also be added to the Evenement class so that you can access the related Session objects from it.

New field name inside Evenement [sessions]:
>

Do you want to activate orphanRemoval on your relationship?
A Session is "orphaned" when it is removed from its related Evenement.
e.g. $evenement->removeSession($session)

NOTE: If a Session may *change* from one Evenement to another, answer "no".

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

updated: src/Entity/Session.php
updated: src/Entity/Evenement.php
```

Observez les mises à jour sur les classes sessions et évènements.

Mettez à jour la base de données jeuxdicode créant et appliquant une nouvelle migration.

Notez que la colonne "evenement_id" dans la table "Session" est bien "not null", et que la dépendance de référence a bien été ajoutée.

Modifiez l'action detailevt($id) afin qu'elle récupère l'événement passé en paramètre de l'URL …../detailevt/id. Il faut pour cela demander au repository de l'entité Evenement de retourner l'événement de l'id concerné. Usage de la méthode find($id) du repository.

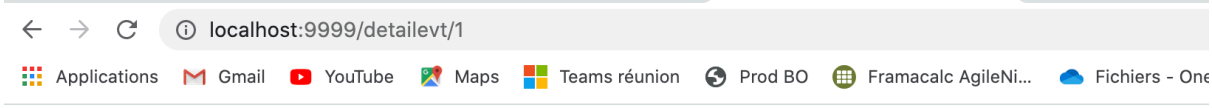
```php
/**
 * @Route("/detailevt/{id}", name="detailevt")
 */
    public function detailevt($id)
    {
        // acces aux services Doctrine, puis entityManager, puis Répository de l'objet Evenement :
        $repository=$this->getDoctrine()->getManager()->getRepository( className: 'App\Entity\Evenement');

        // récupération de l'evenement passé en paramètre
        $EvenementChoisi=$repository->find($id);

        return $this->render( view: 'evenement/detailevt.html.twig', [
            'EvenementChoisi'=>$EvenementChoisi,
        ]);

    }
```

Mettez à jour la vue pour afficher le titre, le nombre max de participants et testez en appelant directement la route paramétrée.



Titre : intro symfony!

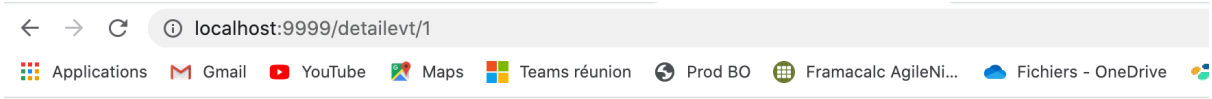
Nombre max participants: 12

Adresse : test

- On souhaite maintenant faire apparaître la liste des sessions de l'évènement concerné, nous allons pour cela demander au repository de Session de nous fournir la liste des sessions de cet événement.

```
// récupération de toutes les sessions associé à cet evenement :
$listeSessions=$EvenementChoisi->getSessions();
```

Créez des sessions pour votre évènements en base et mettez à jour la vue pour afficher le titre et le nom de l'auteur testez en appelant directement la route paramétrée.



Titre : intro symfony

Nombre max participants: 12

Adresse : test

Session : presentation ORM

Auteur : Cyrille Suire