

Semestre : 1 ☐ 2 ☒

Session : Principale ☒ Rattrapage ☐

Module : Technologies Web2.0

Enseignants : UP Web

Classes : 3A1-3A19

Documents autorisés : OUI ☐ NON ☒ Nombre de pages : 7 pages

Calculatrice autorisée : OUI ☐ NON ☒ Internet autorisée : OUI ☐ NON ☒

Date : 07/06/2021

Heure : 13h30

Durée : 1h30

Partie 1 : QCM (8 points)

NB : Une seule réponse est correcte.

1. Dans un projet Symfony 4, dans quel fichier se trouvent les commandes ?
 - A. bin
 - B. console
 - C. app
 - D. src
2. Ou se situe l'**Entity Manager** dans doctrine ?
 - A. Entre l'entité et le contrôleur
 - B. Entre l'entité et la vue
 - C. Entre l'entité et la table dans la base de données
 - D. Aucune de ces réponses
3. Est-il obligatoire que chaque méthode dans le contrôleur admette une route ?
 - A. Oui, car la méthode est instanciée par sa route
 - B. Oui et doit être définie en utilisant les Annotations
 - C. Non, seules les méthodes ayant un affichage ont besoins d'une route
 - D. Aucune de ces réponses
4. Quelle opération ne peut-on pas réaliser avec le langage DQL ?
 - A. INSERT
 - B. SELECT
 - C. UPDATE
 - D. DELETE
5. Par quoi est appelé le paramètre positionnel dans DQL ?
 - A. Par son nom dans la requête :param
 - B. Par sa position dans la requête ?position
 - C. Par son nom dans la requête ?param

D. Par sa position dans la requête :position

6. L'annotation @ORM\GeneratedValue() sert à ?

- A. Préciser que le ou les attributs feront office de clé primaire.
- B. Indiquer que l'attribut est généré de façon automatique lors de l'insertion dans la base.
- C. Indiquer que la valeur de l'attribut doit être affectée avant l'insertion dans la base.
- D. Indiquer que la valeur de l'attribut doit être affectée après l'insertion dans la base.

7. Quel est le résultat de l'exécution de la route '/display' dans le bout de code ci dessous ?

```
/**
 * @Route("/display ", name="display")
 */
private function display()
{
    return new Response(' "Good Luck" ');
}
```

- A. Good Luck
- B. Erreur d'exécution
- C. 'Good Luck'
- D. "Good Luck"

8. Quel est le résultat du code Twig suivant?

```
{{ app.request.server.get("SERVER_NAME") }}
```

- A. Affiche "SERVER_NAME"
- B. Retourne la séquence de tous les éléments disponibles de la requête
- C. Affiche un paramètre d'une requête GET en utilisant la méthode get()
- D. Retourne le nom du serveur hôte qui exécute le script

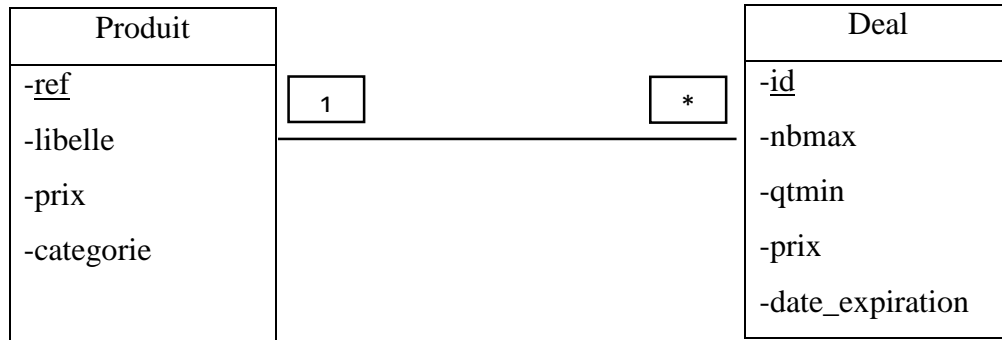
Partie 2 : (12 points)

Le concept d'achat groupé désigne, comme son nom l'indique, plusieurs personnes qui ont envie du même produit et qui peuvent l'acheter à un moindre prix si elles le commandent en nombre.

Un Produit est caractérisé par une référence, un libellé, un prix et une catégorie.

Chaque Deal a un identifiant. Il est composé d'un produit, la quantité minimum de produit à vendre, un nombre maximum d'acheteur, une date d'expiration et le prix.

Soit le diagramme de classe suivant :



NB :

- L'attribut **id** de la classe Deal est Auto Incrément
- L'attribut **ref** de la classe Produit n'est pas auto incrément
- L'attribut **date_expiration** désigne la date d'expiration d'un deal

1. Ajouter l'annotation nécessaire au début de chaque entité **(0.75pt)**

- Entité Produit

```
/**
 * @ORM\[1]( [2]= [3])
 */
class Produit
{
```

2. Ajoutez le code nécessaire pour avoir une relation entre les deux entités **(1.5 pts)**

- Entité Produit

```
/**
 * @ORM\[1](targetEntity=[2]::class, mappedBy=" [3] ") */
private $deals;
```

- Entité Deal

```
/**
 * @ORM\[4](targetEntity=[5]::class, inversedBy=" [6] ")
 * @ORM\JoinColumn(name="Produit", referencedColumnName="ref")*/
private $produit;
```

3. Dans la base de données la migration de la clé étrangère sera dans quelle table ? **(0.25 pt)**

4. Complétez le code source pour avoir le formulaire suivant qui permet d'ajouter un deal **(3.25 pts)**

Nbmax

Qtmin

Prix

Dateexpiration
 Jan ▼ 1 ▼ 2016 ▼

Produit Ceinture MK ▼

Figure 1:formulaire d'ajout d'un deal

-FormDealType

```
public function buildForm(FormBuilderInterface $builder, array $options)
{
    $builder
    ->add('nbmax')
    ->add('qtmin')
    ->add('prix')
    ->add('dateexpiration')
    ->add([1],EntityType::class,
    [
        'class'=>[2],
        'choice_label'=>[3],
        'multiple'=>[4] ])
}
```

-DealController

```
/**
 * @Route("/deal/Add" , name="d_add")
 */

public function Add([5] $req)
{
    [6]
    $form=$this->createForm([7]::[8], $deal);
    $form->add('Enregistrer', SubmitType::class);
    $form->handleRequest($req);
    if($form->isSubmitted() && $form->isValid())
    {
        $sem=[9];
        $sem->persist([10]);
        $sem->flush();

        return $this->redirectToRoute('d_afficher');
    }

    return $this->render('deal/adddeal.html.twig',[
```

```
'fadd'=>$form->createView()  
]);
```

-adddeal.html.twig

```
{{ form [11] }}
```

5. Compléter le code ci-dessous pour afficher la liste des deals dans le fichier twig (2 points)

- DealController

```
/**  
 * @Route("/deal/list" , name="[1]")  
 */  
  
public function list([2] $rep)  
{  
  
    $deals=$rep->findAll();  
    return $this->render('deal/listdeals.html.twig',['listDeals'=>[3]]);  
  
}
```

Remarque :

On désire afficher juste la date et les libellés de chaque produit dans la liste des deals

-listdeals.html.twig

```
[4]  
<tr>  
  
    <td>[5]</td>  
    <td> [6]</td>
```

</tr>

[7]

6. Complétez la méthode Query Builder « **Dealexpiree()** » qui permet d’afficher les deals expirés(dont leurs date est inférieure à la date système) (**1.75 points**)

- **DealRepository**

```
public function Dealexpires(){  
  
    $qb= $this->[1] ('d');  
    $qb ->[2] ('[3] <CURRENT_DATE()');  
    return $qb->[4]-> [5];  
}
```

- **DealController**

```
/**  
 * @Route("/deal/dealexpired")  
 */  
  
public function dealexpiredByQB(DealRepository $rep)  
{  
    $deals=[6]-> [7];  
    return $this->render('deal/listexpired.html.twig', [  
        'listExpired' => $deals  
    ]);  
}
```

7. Corrigez la méthode DQL « **DealByCategory()** » qui permet d’afficher les deals d’une catégorie donnée (**2.5 points**)

- **DealRepository**

```

public function DealByCategory($cat) {

    $entityManager=$this->getEntityManager();
    $query=$entityManager
        ->createQuery("SELECT * FROM APP\Entity\Deal d
            JOIN ON d.produit p WHERE p =:cat ")
        ->setParameter($cat);
    return $cat;

}

```

- DealController

```

/**
 * @Route("/deal/dealebyCat")
 */

public function dealebyCatDQL(DealRepository $rep)
{
    $deals=$rep->DealByCategory();
    return $this->render('deal/listcategory.html.twig', [
        'tab' => $deals
    ]);
}

```