

EXAMEN

☐ ☐
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.

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

Produit		D
- <u>ref</u>	1	- <u>id</u>
-libelle		-nbmax
-prix		-qtmin
-categorie		-prix
		-date_expiration

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
{
```

[1] : Entity 0.25
[2] : repositoryClass 0.25
[3] : ProduitRepository

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;
```

- [1] : OneToMany 0.25
- [2] : Deal 0.25
- [3] : produit

- 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] : ManyToOne 0.25 [5] : Produit 0.25 [6] : deals 0.25

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 ▼

Enregistrer

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] ])
}
```

[5] : Request 0.25

[6]: \$deal=new Deal() ; 0.5

[7] : FormDealType 0.25

[8]: class 0.25

[9]: \$this->getDoctrine()->getManager();(0.5 pts)

[10]=\$deal 0.25

[11]=fadd 0.25

5. Compléter le

-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())
    {
        $em=[9];
        $em->persist([10]);
    }
    $em->flush();

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

```

```

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

```

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

-adddeal.html.twig

```
{{form [1]}}
```

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

DealController

[1] : d_afficher 0.25

[2] : DealRepository 0.25

[3] : \$deals 0.25

-listdeals.html.twig

[4] : {% for d in tab %} 0.25

[5]: {{ d.dateexpiration|date }} (0.5 pt)

[6] = {{ d.produit.libelle }} 0.25

[7]= {% endfor %}

- 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];  
}
```

[1] : createQueryBuilder 0.25

[2] :where 0.25

[3] : d.dateexpiration0.25

[4] : getQuery()0.25

[5]: getResult()0.25

- DealController

[6] : \$rep 0.25

[7]: Dealexpires();0.25

- 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

DealRepository

```
*: d 0.25
*: à supprimer On 0.25
*: p.categorie 0.25
*: setParameter('cat',$cat) 0.5
* : return $query->getResult();0.5
public function DealByCategory($cat) {
    $entityManager=$this->getEntityManager();
    $query=$entityManager
->createQuery("SELECT d FROM APP\Entity\Deal d
JOIN ON d.produit p WHERE p.categorie=:cat ")
->setParameter('cat',$cat);
return $query->getResult();
}
*: /{cat} 0.25
*: $cat 0.25
*: $cat 0.25

/** @param DealRepository $rep
 * @return Response
 * @Route("/deal/dealebyCat/{cat}")
 */
public function dealebyCatDQL($cat,DealRepository $rep)
{
    $deals=$rep->DealByCategory($cat);
return $this->render('deal/list.html.twig', [
]);
}
'tab' =>
$deals
```

```

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
]);
}

```