esprit Se former autrement	**EXAMEN** Semestre : 1 ■ 2 □ Session : Principale ■ Rattrapage □

Module : Technologies Web2.0
Enseignants : UP Web
Classes : 3A20->3A38
Documents autorisés : OUI □ NON ■ Nombre de pages : 6
Calculatrice autorisée : OUI □ NON ■ Internet autorisée : OUI □ NON ■
Date : 05/02/2021 Heure : 09h00 Durée : 01h30

Partie 1 : QCM (8 points)

1- Afin d'utiliser des fonctions prédéfinies tel que « render » dans le contrôleur de quelle classe doit-on hériter ?

a- Controller

b- AppController

c- AbstractController

d- Aucune de ces réponses

2- Dans un projet symfony 4, quel fichier est considéré comme le contrôleur frontal « Front Controller » ?

a- bin/console.php

b- public/index.php

c- src/kernel.php

d- composer.json

3- Que contient le dossier Vendor dans un projet Symfony 4?

a- Les fichiers de configuration de Symfony

b- Les tests automatiques

c- La configuration de l'environnement du projet.

d- Les dépendances nécessaires au projet

4- Dans quel fichier on définit la version de TWIG ?

a- index.html.twig

b- .env

c- composer.json

d- symfony.lock

5- Quelle est la syntaxe utilisée pour parcourir un tableau Users dans le TWIG ?

a- {% for Users in User%}

b- {# for Users in User#}

c- {% for Users in Users%}

d- {# for Users in User#}

6- Quel est le rendu du code suivant ?

```
{% set products = [ ] %}
<table>
{%for product in products %}
 <tr>
<td>{{product.title|upper}}</td>
</tr>
 {%endfor%}
</table>
```

a- Une ligne contenant une colonne avec le titre en majuscule

b- Un tableau vide

c- Erreur

d- 3 lignes contenant une colonne avec le titre en majuscule

7- Quel est le résultat de code suivant {% set name = ' Mohamed' %} {{- name }} ?

a- erreur

b- Mohamed

c- 'Mohamed'

d- Aucune de ces réponses

8- Quel code doit-on indiquer sur un attribut dans une classe pour qu'il soit incrémenté automatiquement ?

a- @ORM\AutoIncrement(strategy="AUTO")

b- @ORM\GeneratedValue

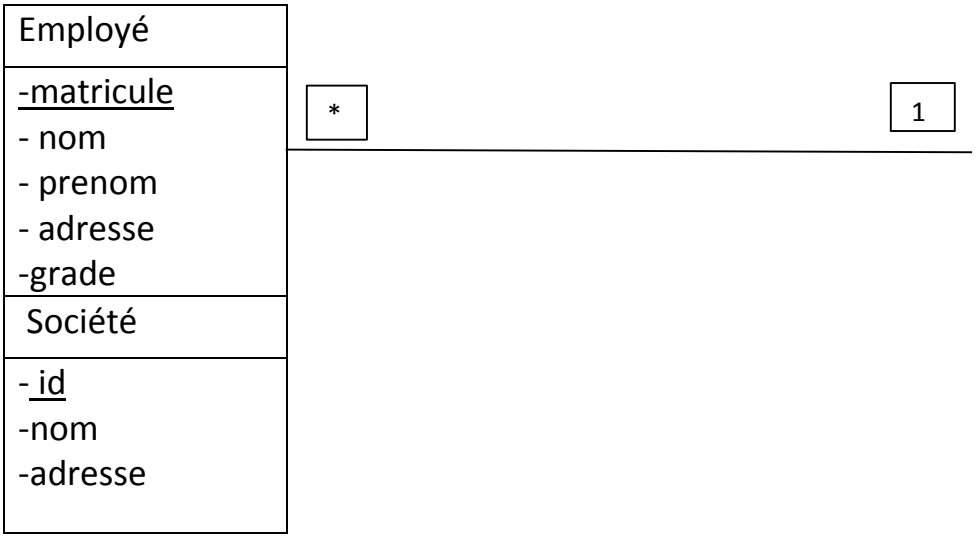
c- @ORM\Column(type="id", strategy="generated")

d- Aucune de ces réponses

Partie 2 : (12 points)

Nous proposons de créer une application Web développée avec Symfony 4 et les données vont être enregistrées dans une base de données MySQL qui permet de gérer les employés dans une société.

Travail demandé :



NB :

- Chaque Employé est associé uniquement à une seule Société
- Une Société peut avoir un ou plusieurs Employé
- L'attribut matricule et id sont Auto Incrément

1- Ajoutez le code nécessaire pour avoir une relation oneToMany entre les deux entités (2 points)

```
2- /**
3-      * @ORM\ManyToOne(targetEntity=Hopital::class, inversedBy="employer")
4-      */
5-     private $hopital;
```

- **Entité employé**

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

[1] employe
[2]societe
[3]societe
[4] employe
Ou
[1] societe
[2] societe
[3] employe
[4] employe

- **Entité société**

```
/**
* @ORM\ManyToOne(targetEntity= [3] ::class, inversedBy=" [4] ")
*/
private $societe;
```

6- Ajout d'un employé (3pts)

a- Compléter le code source pour avoir le formulaire d'ajout de la figure 1 (1.5 pts)

NB : le champ société est de type bouton radio

Figure 1 : formulaire d'ajout d'un employé

```
public function buildForm(FormBuilderInterface $builder, array $options)
  {
    $builder
      ->add('nom')
```

```
        ->add('prenom')
        ->add('adresse')
        ->add(' grade ')
        ->add([1]employe,EntityType[2]::class,[
           'class'=>Societe[3]::class,
           'choice_label'=> nom[4]
           'multiple'=> false[5],
           'expanded'=> true,
        ])
        ->add('save', SubmitType[6]::class);
    }
```

b- Compléter la fonction d'ajout (addEmploye) dans le contrôleur (1.5 pts)

```
            /**
             * @Route("/addE", name="addE")
             */
            public function addEmploye( Request $request)
            {
              $employes = new Employer()[1];
               $form=$this->createForm(EmployerType::class, $employes);
               $form->handleRequest($request [2] );
              if ($form->isSubmitted()) {
        $em= $this->getDoctrine()->getManger()[3];
                $em->persist ($employes)[4];
                $em-> flush()[5] ;
               return new Response('Ajout avec succès') ;
              }

              return $this->render('empolye/add.html.twig', [
                 'form' => Createview()[6],
              ]); }
```

7- Corrigez et complétez la fonction « updateEmploye() » afin de faire la mise à jour d'un employé (1.5 points)

```
/**
     * @Route("/update/{ matricule }", name="update")
     */
  public function updateEmploye (EmployerRepository $repository, $matricule, Request
$request)
  {
       $employe = $repository->find($matricule)
```

```
$form=$this->createForm(EmployerType::class, $employe);
$form->handleRequest($request);
      if ($form->isSubmitted()) {
$em=$this->getDoctrine()->getManger();
         $em->flush();
        return new Response("l'employé a été modifié avec succès");
      }

              return $this->render('empolye/update.html.twig', [

         'form' => Createview()]
   }
```

8- Complétez la fonction « getSocietes() » afin d'afficher la liste des Sociétés (1point)

```
          /**
             * @Route("/showSocietes", name="getSocietes")
             */
            public function getSocietes()
            {
          $repository =[1]$this->getDoctrine()->getReposotry(Societe::class)
                $societes =$repository-> findAll()[2];
                return $this-> render[3] ('societes/getAllsocietes.html.twig', [
                   'societes' => [4] $ societes
       ]);
```

9- Corrigez et complétez la fonction « deleteSociete() » qui permet de supprimer une société et de faire une redirection vers la liste des sociétés (1 point)

```
                /**
                 * @Route("/deletesociete/{id}", name="delete")
                 */
                public function deleteSociete($id, SocieteRepository $repository)
                {
          $societe= $repository->find($id)
          $em->remove($societe);
                    $em->flush();
                    return  $this->redirectToRoute('getSocietes') ;
       }
```

10- Compléter le code ci-dessous pour ajouter un lien de suppression d'une société dans le fichier twig (1 point)

```
{% for societe in societes %}
    {{ societe.id}}
    {{ societe.nom}}
    {{ societe.adresse}}
    <a href="{{path('delete',{' id[1]': societe.id [2]})}}">delete</a>
{%endfor%}
```

11- Ordonner le code suivant afin de faire une recherche des employés (1 point)

```
public function searchemploye($data){

   return
1- $this->createQueryBuilder('b')
2-      ->where('b.nom like :name')
3-      ->setParameter('name',$search)
4-          $search='%'.$data.'%';
5-      ->getResult()
6       ->getQuery();
}
```

4/1/2/3/6/5

12- Compléter la fonction DQL « getSecondSalaire() » qui permet de retourner le deuxième employé le plus rémunéré (1.5pts)

```
public function getSecondSalaire(EntityManagerInterface $em){

 $maxsalaire=$em-> createQuery [1] ('SELECT max[2] (e.salaire) from
App\Entity\Employee ');

   $res=$em-> createQuery [3] ('SELECT max[2] (e.salaire)  from App\Entity\ Employe e
where e.salaire != [4]:maxsalaire ');

$res-> setParameter ('maxsalaire', $maxsalaire->getSingleScalarResult());
return $res->getSingleScalarResult();
}
```