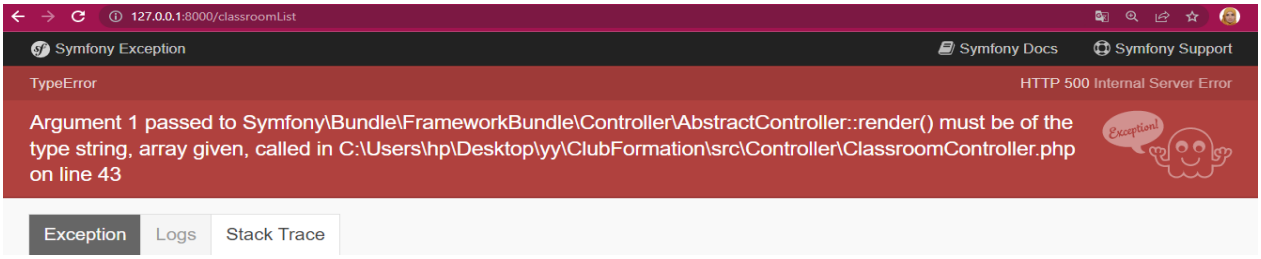
esprit Se former autrement HONORIS UNITED UNIVERSITIES	EXAMEN Semestre : 1 ☐ 2 ■ Session : Principale ■ Rattrapage ☐

Module: Technologies web 2.0
Enseignant(s): UP web
Classes : 3A2->3A27
Documents autorisés : OUI ☐ NON ■ Nombre de pages : 7 pages
Calculatrice autorisée : OUI ☐ NON ■ Internet autorisée : OUI ☐ NON ■
Date : 06/04/2022 Heure : 13h30 Durée : 1h30

Partie 1 : QCM (Une seule réponse est correcte) (8 points)

1. Quelles sont les étapes à suivre pour modifier la version du Twig ?
   A. Modifier le fichier composer.json et exécuter la commande twig:update
   B. Modifier le fichier twig.yml et exécuter la commande composer:update
   C. Modifier le fichier base.html.twig et exécuter la commande twig:update
   D. Modifier le fichier composer.json et exécuter la commande composer:update
2. Que contient le fichier .env dans un projet Symfony4 ?
   A. Les noms de la base de données et des tables
   B. Seulement le nom de la base de donnés
   C. Seulement les noms des tables
   D. Aucune de ces réponses n'est correcte
3. Quelle est la cause de cette erreur ?



   A. Il y a une erreur dans la méthode render dans le contrôleur.
   B. Il y a un problème dans le nom du fichier twig
   C. Il y a un problème dans le fichier twig.
   D. Il y a un problème dans la route
4. Dans un projet Symfony4, sous quel dossier se trouve le contrôleur frontal ?
   A. bin
   B. public
   C. src
   D. templates

5. Dans un projet Symfony4, sous quel dossier se trouvent les dépendances nécessaires au projet ?
    A. bin
    B. src
    C. vendor
    D. public

6. Est-il obligatoire que chaque méthode dans le contrôleur admette une route ?
    A. Oui, car la méthode est appelée par sa route
    B. Oui et doit être définie en utilisant les annotations
    C. Non, sauf pour les méthodes qui retournent une réponse TWIG
    D. Aucune de ces réponses n'est correcte

1. Que permet de faire L'ORM ?
    A. La relation entre les données orientées objet et les contrôleurs
    B. La relation entre les entités et les contrôleurs
    C. La relation entre les vues et les données relationnelles
    D. La relation entre les données orientées objet et les données relationnelles

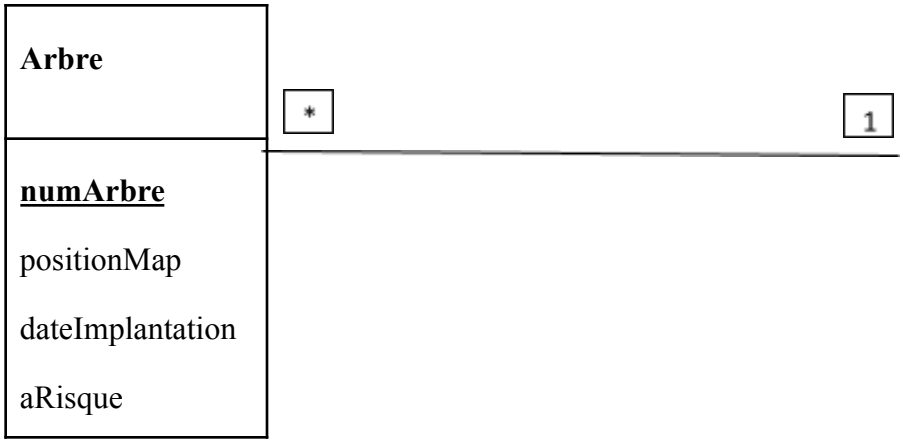
2. Quelle est la proposition qui représente une relation bidirectionnelle entre deux entités Student et Project de type OneToMany?
    A. On peut accéder seulement aux instances de Project à partir de Student.
    B. On peut faire $student->getProjects() mais on on peut pas faire $projects->getstudents()
    C. On peut faire $student->getProject() et $project->getstudents()
    D. Aucune de ces réponses n'est correcte

Partie 2 : Etude de cas (12 points)

Nous proposons de créer une application Web nommée « TREES » pour la gestion des arbres d'un parc. Chaque arbre possède un numéro unique et est examiné pour déterminer s'il est à risque de mourir. L'application est développée avec Symfony 4 et les données seront enregistrées dans une base de données de type MySQL.

Parc
Id nom adresse

Soit le diagramme de classe suivant :



- dateImplantation : champ date qui représente la date d'implantation de l'arbre
- aRisque : champ booléen qui décrit si l'arbre est à risque de mourir

1. Compléter le code suivant de l'entité Parc sachant que l'attribut id est la clé primaire, entier et s'incrémente automatiquement, l'attribut nom est une chaîne de caractère **[0.5 pt].**

Parc.php

```
      /**
 * @ORM\Id
 * @ORM\[1]
 * @ORM\[2](type="integer")
      */
private $id;
/**
 * @ORM\[2](type="string", length=255)
 */
 private $nom;
```

2. a. Quelles sont les étapes à faire pour faire le mapping des entités vers les tables de la base de données ? **[1pt]**
   b. Quel est le fichier qui sera généré après cette étape (de la question2.a) ainsi que son emplacement ? **[0.5pt]**
3. Ajouter le code nécessaire pour avoir une relation entre les deux entités**. [1pt]**

Arbre.php

```
/**
* @ORM\[1](targetEntity=Parc::class, [2] ="arbres")
*/
private $parc;
```

Parc.php

```
/**
* @ORM\ [3] (targetEntity=Arbre::class, [4]="parc")
*/
private $arbres;
```

4. Compléter le code nécessaire pour avoir le formulaire correspondant à l'entité Arbre comme indique la figure suivante **[1.5pts]**

Num arbre
Position map
Date implantation Jan 1 2017

ArbreType.php

```
->add('numArbre')
->add('positionMap')
->add('dateImplantation')
->add('aRisque')
->add('parc', [1]::class,
       [
           '[2]'=> [3]::class,
           '[4]'=>'[5]',
       ]);
->add(' enregistrer',[6]::class)
```

5. Compléter la fonction de mise à jour d'un arbre (updateArbre) dans le contrôleur. **[2pts]**

```
     /**
     * @Route("/arbre/update/{numArbre}", name="updateArbre")
     */
    public function updateArbre([1],[2])
    {
        $arbre = $this->getDoctrine()->getRepository([3])->[4];
        $form = $this->createForm(ArbreType::class,$arbre);
        $form->handleRequest([5]);
        if($form->isSubmitted())
        {
            $em = [6];
            $em->[7];
        }
        return $this->render('arbre/add.html.twig',['form'=>[8]);
    }
}
```

6. Compléter le code nécessaire afin d'afficher le numéro, la date d'implantation et le parc des arbres. **[2pts]**

arbreController.php

```
     /**
     * @Route("/getArbre", name="getArbre")
     */
    public function getArbres():Reponse
    {
        $rep = [1] ;
        $arbres = $rep->[2] ;
        return $this->render('arbre/list.html.twig',['arbres'=>[3]]);
    }
```

arbre/list.html.twig

```
<table border="1">
    <tr>
        <th>Numéro</th>
        <th>Date Implantation</th>
        <th>Parc</th>
    </tr>
    [4]
    <tr>
        <td>{{ c.numArbre }}</td>
        <td>{{ c.dateImplantation | [5]}}</td>
        <td>{{ [6] }}</td>
    </tr>
     [7]
</table>
```

7. Ecrire la fonction « deleteTree() » qui permet de supprimer un arbre qui est à risque de mourir et de rediriger vers la liste des arbres sinon retourner le message 'Erreur'.**[2 pts]**

```
/**
 * @Route("/arbre/delete/{numArbre}", name="deleteArbre")
*/
 public function delete($numArbre, ArbreRepository $rep, EntityManagerInterface $em){


}
```

8. Réécrire en corrigeant les erreurs de la fonction DQL « ParcStartByA() » suivante qui permet d'afficher l'ensemble des Parcs dont le nom commence par la lettre « «**A** » **[1.5pts]**

```
public function parcStartByA() {

$em = $this->getEntityManager();

 $query = $em->createQueryBuilder('SELECT p FROM Parc p WHERE p.nom LIKE :val')

              ->setParameter('val','A');

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

Bon Courage!