Ecole Supérieure Privée Technologies & Ingénierie

**Type d’épreuve :** Devoir Examen

**Enseignant :** Wissem ELJAOUED

**Matière :** Framework PHP (Symfony)

**Année Universitaire  :** 2022-2023 **Semestre :** 1

**Classe :** CII-2-GLSI-A-B-C-D, CII-2-DSEN-A

**Documents :** AutorisésNon autorisés

**Date : 04-01-2023 Durée :** 1h :30

**Nombre de pages :** 8 pages

**Exercice 01 : (10 pts)**

1. ***QCM (5 pts)***
2. Dans Symfony, la configuration des routes peut être écrite en\_\_\_\_\_? (d)  
   a) YAML  
   b) PHP  
   c) XML  
   d) Toutes les réponses sont vraies
3. Quel est le rôle du routeur dans Symfony ? (b)  
   a) Lister toutes les URL d’un site Internet  
   b) Associer des paramètres à une URL  
   c) Exécuter un contrôleur à partir d’une URL

d) Rediriger les URLs

1. En Twig, pour appliquer un filtre sur un attribut ***attribut*** d’un objet ***$objet***, quelle est la bonne  
   syntaxe ? (c)  
   a) {{ objet->attribut | filtre }}  
   b) {% objet.attribut | filtre %}  
   c) {{ objet.attribut | filtre }}  
   d) {{ filtre(objet.attribut) }}
2. Parmi les codes du contrôleur suivant, lequel supprime correctement une entité ? (a)

|  |  |
| --- | --- |
| **a)** $em= $this->getDoctrine()->getManager(); $salle = $em-> getRepository('NBundle:Salle') -> find($id); $em->remove($salle); $em->flush(); | **c)** $em= $this->getDoctrine()->getManager(); $em->remove(); $em->flush(); |
| **b)** $em= $this->getDoctrine()->getManager(); $salle = $em-> getRepository('NBundle:Salle') -> find($id); $em->remove ($salle); | **d)** $salle = $em-> getRepository('NBundle:Salle') -> find($id); $em->delete ($salle); $em->flush() |

1. Supposons qu’on a deux templates Twig : *enfant.twig* qui étend *parent.twig* qui contient un  
   block appelé ***body***. Si l'on surcharge ce block dans *enfant.twig*, comment retrouver le contenu  
   de ce block provenant de *parent.twig* ? (b)  
   a) {% parent() %}  
   b) {{ parent() }}  
   c) {% parent(body) %}  
   d) {{ parent(body) }}
2. ***Question/Réponse* (5 pts)**
3. Convertir le code PHP suivant en Twig.

|  |
| --- |
| <?php  echo ‘<ul>’;  if (count($list\_films) > 0) { foreach ($list\_ films as $film ) { echo ‘<li>’. $film->getNom().’</li>’ ; } } else { **?>** <li> Pas de film trouvé </li> **< ?php** }  echo ‘</ul>’; **?>** |

Méthode 1 :

<ul>

{% if list\_films | length > 0 %}

{% for film in list\_films %}

<li> {{ film.nom}}</li>

{% endfor %}

{% else %}

<li> Pas de film trouvé </li>

{% endif %}

</ul>

Méthode 2 :

<ul>

{% for film in list\_films %}

<li> {{ film.nom}}</li>

{% else %}

<li> Pas de film trouvé </li>

{% endfor %}

</ul>

1. Dessiner et expliquer l’architecture de Symfony en précisant tous les composants nécessaires.

Une image contenant texte, tableau blanc, écriture manuscrite, intérieur

Description générée automatiquement

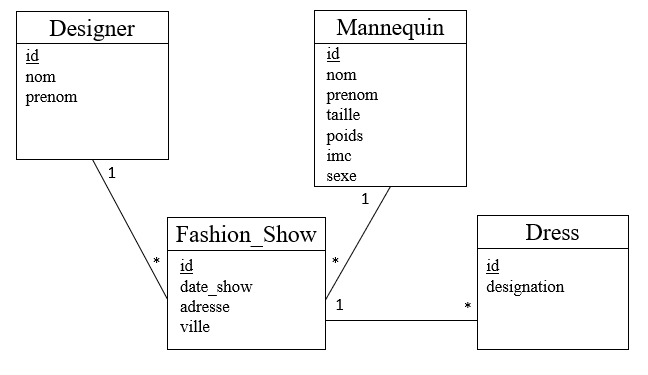
1. Comment faire la redirection vers la page qui a comme route « **liste\_chiens** » en utilisant la méthode **redirect** (sachant que **redirect** prend comme argument une URL)

Redirect($this->generateURL(“liste\_chiens”));

**Problème : (10 pts)**

Notre objectif est d’implémenter une application web qui gère des fashion shows en précisant pour chaque designer, ses défilés de mode (fashion shows) et les mannequins participants (en leurs associant les robes à porter). Dans ce cadre, on propose de créer une application avec Symfony 5.3.

Soit le diagramme de classes suivant :



1. ***Configuration des entités :***

***A.1.*** Compléter les annotations (et les attributs) manquantes dans les entités suivantes sachant qu’on peut naviguer entre les entités dans les deux sens :

|  |  |
| --- | --- |
| **<?php**  **namespace App\Entity;**  **use Doctrine\ORM\Mapping as ORM;**  #[ORM\Entity]  class **Designer**  **{**    #[ORM\GeneratedValue]  #[ORM\Id]  #[ORM\Column(type:integer)]    private **$id**;  #[ORM\Column(type:string)]  private **$nom**;  #[ORM\Column(type:string)]  private **$prenom**;  #[ORM\OneToMany(targetEntity:FashionShow)  private $fashionShows;  //Getters-Setters  **}** | **<?php**  **namespace App\Entity;**  **use Doctrine\ORM\Mapping as ORM;**  #[ORM\Entity]  class **FashionShow**  **{**  #[ORM\GeneratedValue]  #[ORM\Id]  #[ORM\Column(type:integer)]  private **$id**;    #[ORM\Column(type:date)]  private **$date\_show**;  #[ORM\Column(type:string)]  private **$adresse**;  #[ORM\Column(type:string)]  private **$ville**;  #[ORM\ManyToOne(targetEntity:Designer)  private $designer;  #[ORM\ManyToOne(targetEntity:Mannequin)  private $mannequin;  #[ORM\OneToMany(targetEntity:Dress)  private $dresses;  //Getters-Setters  **}** |

***B. Affichage de la liste des fashion shows :***

Soit ***FashionShowController***, le contrôleur responsable des actions de gestion des fashion shows.

Soit ***FashionShowRepository***, le repository de l’entité fashion shows.

***B.1.*** Terminer l’action ***listFS()*** qui permet de retourner la page Twig suivante : ***‘listFS.html.twig’*** (se trouve sous : ***Templates/***). Cette page Twig permet de retourner la liste des fashion shows sauvegardés dans notre base de données.

|  |
| --- |
| **<?php**  **namespace App\Controller;**  **use Symfony\Bundle\FrameworkBundle\Controller\Controller;**  class ***FashionShow*Controller** extends **Controller**  **{**  **#[Route(‘/listeFS’, name:’listeFS’)]**  public function **listFS(FashionShowRepository $fr)**  {  //Liste des fashion shows  **$fashionshows = $fr->findAll();**  **return** $this->render('/***listFS.html.twig*** ',  array(  “listeFashion”=>$fashionshows  ));  }  **}** |

***B.2.*** Compléter la page Twig suivante : ***‘listFS.html.twig’***. Pour Le Designer d’un fashion show, on veut afficher son **nom,** pour son mannequin, afficher son **nom** et pour les robes associées, afficher leurs **désignations**.

|  |
| --- |
| **<h1>** Liste des fashion shows **</h1>**  **<table border="1">**  **<tr>**  <td> id </td>  <td> date Show </td>  <td> adresse </td>  <td> ville </td>  <td> designer </td>  <td> Mannequin </td>  <td> Dresses </td>  **</tr>**    {% for f in listeFashion %}  **<tr>**  <td> {{f.id}} </td>  <td> {{f.dateShow}} </td>  <td> {{f.adresse }}</td>  <td> {{f.ville}} </td>  <td> {{f.designer.nom}} </td>  <td> {{f.Mannequin.nom}} </td>  <td> {% for d in f.dresses %}  {{d.designation}}  {% endfor %}  </td>  **</tr>**    {% endfor %}  **</table>** |

***C. Ajout d’un Mannequin :***

Pour ajouter un mannequin, nous avons besoin de créer un formulaire pour introduire ses données. Soit ***‘MForm’***, le formulaire d’ajout d’un mannequin.

Soit ***MController***, le contrôleur responsable des actions de gestion des mannequins.

***C.1.*** Terminer l’action ***addM()*** qui permet de:

* Rediriger vers la liste des mannequins ***Si*** l’ajout d’un mannequin est effectué correctement (sachant que le nom de la route de la liste des mannequins est ***listeMRoute***).
* ***Sinon***, Afficher le formulaire ***‘MForm’***en utilisant la page Twig suivante : ***‘addM.html.twig’*** (se trouve sous : ***Templates/M***).
* ***Calculer l’indice de masse corporelle (imc) avant d’ajouter un mannequin sachant que :***
  + IMC = poids/taille.

|  |
| --- |
| **<?php**  **namespace App\Controller;**  **//use**  class **MController** extends **Controller**  **{**  #[Route(‘/addM’, name :’addM’)]  public function **addM**(Request $r, EntityManagerInterface $em)  {    // Création d’un mannequin  $mannequin = new Mannequin() ;  // Création d’un formulaire  **$form** = $this->createForm (MForm::class, $mannequin);  **$form**->handleRequest($r);    if ($form->isSubmitted())  {  //Ajout du mannequin  $imc = $mannequin->getPoids()/$mannequin->getTaille();  $mannequin->setIMC($imc);  $em->persist($mannequin);  $em->flush();  **return** $this-> redirectToRoute(“listeMRoute”);  }  **return** $this->render('M/***addM.html.twig*** ',  array(  “formM”=>$form->createView()  ));  }  **}** |

***C.2.*** Compléter la page Twig suivante : ***‘addM.html.twig’***.

|  |
| --- |
| **<h1>**Ajout d’un mannequin **</h1>**  **<form** method=’post’>  {{form\_widget(formM)}}  **<input** type=’submit’ value=”addM”**/>**  **</form>** |

***D. Ajout d’un Fashion show :***

Soit ***‘*FS*Form’***, le formulaire d’ajout d’un fashion show.

***D.1.*** Compléter le formulaire ***‘*FS*Form’***:

Afficher les **noms des mannequins** pour **la liste déroulante des mannequins**.

Afficher les **noms des designers** pour **la liste déroulante des designers**.

Afficher les **designations des robes** pour **la liste déroulante des robes**.

|  |
| --- |
| **<?php**  **namespace App\Form;**  class **FSForm** extends **AbstractType**  **{**  public function **buildForm**(FormBuilderInterface $builder, array $options)  {  $builder  ->add(‘date\_show’, DateType ::class)  ->add(‘adresse’, TextType ::class)  ->add(‘ville’, TextType ::class)  ->add(‘designer’, EntityType::class, array( 'class' =>Designer::class, 'choice\_label'=>’nom’, 'expanded'=>false, 'multiple'=>false)  ->add(‘Mannequin’, EntityType::class, array( 'class' =>Mannequin::class, 'choice\_label'=>’nom’, 'expanded'=>false, 'multiple'=>false)  ->add(‘dresses’, EntityType::class, array( 'class' =>Dress::class, 'choice\_label'=>’designation’, 'expanded'=>false, 'multiple'=>true)  }  **}** |

***E. Doctrine Query Language (DQL) :***

***E.1.*** Ecrire une action php **nombreMannequins($ville)** permettant de calculer le nombre des mannequins qui ont participés à des fashion shows :

* à la ville **$ville**
* présentés par le designer **« Elie saab »**
* ont **« Gigi hadid »** parmi les mannequins participants et dont lesquels elle a porté exactement **2 robes**
* retourner que les mannequins **masculins**

|  |
| --- |
| Public function **nombreMannequins($ville, EntityManagerInterface $em)** {  $nbrM = 0;  $q = $em->creatQuery(‘  Select f from App\Entity\FashionShow f Join f.designer d Join f.mannequin m Join f.dresses dr  Where  f.ville = $ville and d.nom = ‘Elie’ and d.prenom = ‘saab’ and m.nom =’gigi’ and m.prenom=’hadid’ and m.sexe = ‘masculin’  ’);  $fashionshows = $q->getResult();  Foreach($fashionshows as f ){  If(count(f.dresses) == 2){  $nbrM ++;  }  }  Return $nbrM;  } |

**Annexe :**

Pour afficher une liste déroulante de type Entity dans un formulaire, utiliser le code suivant :

EntityType::class, array( 'class' => 'NomBundle\Entity\NomEntity', 'choice\_label'=>'NomLabel’, 'expanded'=>false, 'multiple'=>false)