2022

Test technique

Développeur d’applications PHP / Bases de données / Système



**Nom du candidat : ANTHONY Edouard**ATTENTION : document corrigé

**Date du test : 09/06/2022**ATTENTION : document corrigé

# Partie bases de données MySQL – cas 1

Nous considérerons pour l’ensemble de cette partie les tables suivantes :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table « **livre** » :   |  |  |  |  | | --- | --- | --- | --- | | livre\_id | livre\_prix | livre\_titre | livre\_genre | | 1 | 40 | Le glaive magique | 1 | | 3 | 40 | Gaffes en gros | 1 | | 4 | 40 | Lagaffe nous gâte | 1 | | 5 | 45 | QRN sur Bretzelburg | 1 | | 6 | 80 | Tour de manège | 3 | | 7 | 45 | Le spectre aux balles d'or | 1 | | 8 | 30 | La bonne chanson | 2 | | 9 | 50 | La jeune Parque | 2 | | 10 | 50 | Michel Strogoff | 4 | | 11 | 50 | La Serpe d'or | 1 | | 12 | 70 | Toujours aimer | 3 | | 13 | 70 | Toujours aimante | 3 | | 14 | 72 | Toujours aimé | 3 | | Table « **lib\_genre** » :   |  |  | | --- | --- | | genre\_id | genre\_libelle | | 1 | Bande Dessinée | | 2 | Poésie | | 3 | Roman de gare | | 4 | Roman | | 5 | Science-Fiction | |

## Effectuer la requête permettant d’afficher le livre numéro 7

Réponse :

select \*

from livre

inner join lib\_genre on livre.livre\_genre = lib\_genre.genre\_id

where livre\_id = 7

Réponse : SELECT \* FROM livres WHERE livre\_id = 7;

## Effectuer la requête permettant d’afficher les livres supérieurs à 50€

Réponse :

Select \*

From livre

Where livre\_prix > 50

Réponse : SELECT \* FROM livres WHERE libre\_prix > 50;

## Effectuer la requête permettant d’afficher les livres comprennant la chaine « tou »

Réponse :

Select \*

From livre

Where livre\_titre like "%tou%"

Réponse : SELECT \* FROM livres WHERE livre\_titre LIKE 'tou%';

## Effectuer la requête permettant d’afficher les poésies supérieurs à 40€

Réponse :

Select \*

From livre

Where livre\_genre = 2

And livre\_prix > 40

Réponse : SELECT \* FROM livres WHERE livre\_genre = 2 AND livre\_prix > 40';

## Donner le **résultat** de cette requête :

## « SELECT livre\_titre, genre\_libelle FROM livre

## INNER JOIN lib\_genre ON livre\_genre = genre\_id

## WHERE livre\_prix IN (72,80) ; »

Réponse :

Tour de manège Roman de gare

Toujours aimé Roman de gare

Réponse :

* Toujours aimé - Roman de gare
* Tour de manège - Roman de gare

## Effectuer la requête permettant d’afficher le nombre de livres par genre et le total de prix par cette catégorie

Réponse :

Select livre.livre\_genre, count(\*), sum(livre\_prix)

From livre

inner join lib\_genre on livre.livre\_genre = lib\_genre.genre\_id

Group by livre.livre\_genre

Réponse : SELECT COUNT(\*), SUM(livre\_prix) FROM livres GROUP BY livre\_genre;

# Partie bases de données MySQL – cas 2

Nous considérerons pour l’ensemble de cette partie les tables suivantes :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table « **eleves** »   |  |  |  | | --- | --- | --- | | id | nom | prenom | | 1 | Dupont | Albert | | 2 | Martin | Sophie | | 3 | Boucher | Alain | | Table « **notes** » :   |  |  |  | | --- | --- | --- | | id\_eleve | matiere | note | | 1 | MATHS | 12.4 | | 1 | HISTOIRE | 7.5 | | 1 | PHYSIQUE | *NULL* | | 2 | MATHS | 8 | | 2 | HISTOIRE | 11 | | 3 | MATHS | *NULL* | | 3 | HISTOIRE | 13.5 | |

## Lister pour chaque élève sa moyenne dans chacune des matières

Réponse : *Question étrange au vu des données car chaque élève n’a qu’une note max par matière !*

Select eleves.id, eleves.nom, eleves.prenom, notes.matiere, round(avg(note),1)

From eleves

Inner join notes on eleves.id = notes.id\_eleve

Group by eleves.id, notes.matiere

Group by eleves.id, notes.matiereRéponse : SELECT nom, prenom, AVG(note) FROM eleves INNER JOIN notes ON id=id\_eleve GROUP BY id\_eleve;

## Afficher les informations de l’élève ayant la plus grosse note en histoire

Réponse :

Select nom, prenom, matiere, max(note)

From eleves

Inner join notes on eleves.id = notes.id\_eleve

where matiere = "HISTOIRE"

group by matiere

having max(note)

Réponse : SELECT nom, prenom, MAX(note) FROM eleves INNER JOIN notes ON id=id\_eleve WHERE matiere = ‘HISTOIRE’;

## Qu’est ce qu’une clé primaire ? Quelle est son utilitée ?

Réponse :

C’est un champ de typer INTEGER qui est unique, et qui permet d’avoir un élément court et discriminant pour spécifier un enregistrement d’une table.

## A quoi servent les index ? Dans quel cas doit-on les utiliser ?

Réponse :

Ils servent à spécifier un enregistrement d’une table avec un élément court généralement de typer INTEGER.

Ensuite si on ordonne (cf. trie) la table par cet index, on optimise la rapidité de recherche au sein de cette table.

# Partie PHP procédural

## Remplacer le mot « World » par « Universe »  via une fonction PHP :

$string = "Hello World !" ;

Réponse :

str\_replace("World", "Universe", $string);

Réponse : str\_replace("World", "Universe", $string);

## Afficher le mombre de caractère de cette chaine, puis l’afficher en majuscule.

$string = "Hello World !" ;

Réponse :

echo strlen($string);

echo strtoupper($string);

Réponse : echo strlen($string); echo strtoupper($string);

## Soit la déclation suivante :

$a = 15 ; $b = 12 ; $c = 1 ;

## Que va afficher :

if($a == 15) echo 'hello';

else echo 'world';

Réponse :

hello

Réponse : hello

## Que va afficher :

echo ($b < 12) ? 'hello' : 'world';

Réponse :

world

Réponse : world

## Que va afficher :

if($c <= 10 && $b%2 != 0) echo 'hello';

else echo 'world';

Réponse :

world

Réponse : world

## Que va afficher :

if($a >= $b + $c)

echo 'hello' ;

else

echo 'world' ;

Réponse :

hello

Réponse : hello

Réaliser le code PHP qui permet d’afficher cette liste au format html sachant que les élèves sont stockés dans un tableau nommé $aEleve :

<ul>

<li class=’impair’>Eleve 1</li>

<li >Eleve 2</li>

<li class=’impair’>Eleve 3</li>

….

</ul>

Réponse :

$aEleve = ['Pierre', 'Paul', 'Hélène', 'Paul', 'Jeanne'];

$compteur = 1;

echo "<ul>";

foreach ($aEleve as $unEleve) {

if ($compteur % 2 == 0) {

echo "<li>" . $unEleve . "</li>";

} else {

echo "<li class=’impair’>" . $unEleve . "</li>";

}

$compteur++;

}

echo "</ul>";

# Partie PHP orienté objet

## Soit la déclation suivante :

class eleve{

var $id;

var $nom;

var $prenom;

public function \_\_construct() {

$this->nom = "dupont";

$this->prenom = "jean";

}

public function f1() {

$this->nom = "perret";

}

private function f2() {

$this->prenom = "marc";

}

protected function f3() {

$this->nom = "perret";

$this->prenom = "marc";

}

}

## Que va afficher :

$my\_eleve = new eleve();

echo $my\_eleve->nom.' '.$my\_eleve->prenom.'<br/>';

Réponse :

dupont jean

Réponse : dupont jean

## Que va afficher :

$my\_eleve = new eleve();

$my\_eleve->f1();

echo $my\_eleve ->nom.' '.$my\_eleve ->prenom.'<br/>';

Réponse :

perret jean

Réponse : perret jean

## Que va afficher :

$my\_eleve = new eleve();

$my\_eleve->f2();

echo $my\_eleve ->nom." ".$my\_eleve ->prenom.'<br/>';

Réponse :

ERREUR : Call to private method eleve::f2() from context '' in C:\wamp64\www\test-mds\test-mds.php on line 30

* En effet, on ne peut pas appeler une méthode « private » en dehors de la classe

Réponse : Fatal error: Call to private method

## Que va afficher :

$my\_eleve = new eleve();

$my\_eleve->f3();

echo $my\_eleve ->nom ." ".$my\_eleve ->prenom.'<br/>';

Réponse :

ERREUR : Call to protected method eleve::f3() from context '' in C:\wamp64\www\test-mds\test-mds.php on line 30

* En effet, on ne peut pas appeler une méthode « protected » en dehors de la classe ou des classes héritières

Réponse : Fatal error: Call to protected method

# Connaissances générales

## Donner une définition au modèle « MVC ». Quels moyens techniques permettent de le réaliser ?

Réponse :

MVC est un standard qui veut dire Modèle Vue Controller.

Le Modèle est les objets (entités = classes + tables en BDD).

La Vue est « le fichier » de présentation des informations à l’utlisateur sur une interface graphique.

Le Controller est le code qui fait la liaison entre la Vue et les objets.

Les Framework PHP que je connais, Silex et Symfony, utilisent ce formalisme (mais aussi beaucoup d’autre Framework).

Sur Symfony on trouve classiquement :

- les controller dans /src/Controller

- les entités (modèle) dans /src/Entity

- les vues dans /templates

## Quels sont les bonnes pratiques que vous connaissez dans l’intégration d’email ?

Réponse :

J’ai utilisé Mailer sur Symfony 5.

Pour la configuration des mails :

Il faut configurer le SMTP dans le fichier « .env » avec la key ‘MAILER\_DSN= …. »

En environnement de PROD (par exemple sur Heroku), il faut créer une variable d’environnement avec cette key

Pour la présentation des mails :

J’ai utilisé Twig avec HTML, CSS, Bootstrap pour présenter les choses joliment.

## Réaliser un formulaire qui demande le nom, le prénom, la date de naissance, si la personne procède (possède ?) un animal (boutons radio), son pays d’origine (select) et qui transmettra le tout sur une page enregistrement.php

Réponse :

Formulaire.html :

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <form action="enregistrement.php" method="POST">

        <p><label for="nom"> Nom :</label>

            <input type="text" required="required" pattern="[a-zA-Z'-]{2,50}" name="nom" size='30' maxlength='50'

                value="" />

        </p>

        <p><label for="prenom"> Prénom :</label>

            <input type="text" required="required" pattern="[a-zA-Z'device-width]{2,50}" name="prenom" size='30'

                maxlength='50' value="" />

        </p>

        <p>

            <label for="date-naissance"> Date de naissance :</label>

            <input type="text" id="date-naissance" name="date-naissance" size='15' maxlength='15' autocomplete="off"

                placeholder="Format jj/mm/aaaa" pattern="[0-3]{1}[0-9]{1}/[0-1]{1}[0-9]{1}/\d{4}" value=""

                required="required" />

        </p>

        <div>

            <span class="label">Possédez vous un chien ? :</span>

            <input type='radio' name="verif-possede-chien" value="oui" />

            <label for="un-chien"">Oui</label>

            <input type='radio' name=" verif-possede-chien" value="non" />

            <label for="pas-de-chien">Non</label>

        </div>

        <br>

        <label for="pays">Pays :</label>

        <select name="pays">

            <option value="France">France</option>'

            <option value="Espagne">Espagne</option>'

            <option value="Allemagne">Allemagne</option>'

        </select>

        <br>

        <br>

        <input type="submit" value="Enregistrer">

    </form>

</body>

</html>

enregistrement.php :

<?php

echo "Voici les données que vous vous apprétez à insérer en base :";

var\_dump($\_POST);

**Comment procédez-vous pour mettre à jour la structure d’une BDD par rapport à vos entités ?**

Sur Symfony :

On modifie l’entité sous Visual Studio Code (VSC) ou on tape cette commande sur le terminal de VSC :

php bin/console make:entity User (dans le cadre d’une entité « User »)

L’assistant va alors nous propose de rajouter des champs, et on rajoute un champ (si c’est la modification demandée).

Ensuite on lance la commande suivante dans le terminal de VSC :

php bin/console make:migration

On vérifier la migration créée, et on ajuste si besoin le SQL.

Puis on lancer la migration avec cette commande :

symfony console doctrine:migrations:migrate (ou symfony console d:m:m)

Il faut impérativement utiliser doctrine et ca commande associées :   
php app/console doctrine :schema :update –force ( --dump-sql pour afficher les requêtes sql executées)  
On peut utiliser create à la place de update la première fois.

**Quel est l’intérêt d’un service dans symfony2 ?**

Je ne connais que Symfony 5

Je suppose qu’un service est une fonction qui permet d’être reconnue partout par le container de services, et qui est donc utilisable de n’importe où dans le Framework.

De structurer et ranger le code afin d’en faciliter la maintenance.  
Egalement de rendre chaque fonction indépendante les unes des autres et pouvoir les faire évoluer indépendamment sans risquer de casser le fonctionnement d’un autre service (principe SOLID)