Fannette SEKO - BTS SIO1 - 24 fev, 2025

Compte rendu du TP 1 PHP Cave à vins :

Page des vins

1) Documenter le code php et html afin d'expliquer chaque étape.

cf code.

2) Pourquoi ce code peut-il être considéré comme de la POO (Programmation Orientée Objet)?

Ce code peut est considéré comme de la POO car il assimile des notions de POO telles que la notion de classe, de constructeur , d'objets , de méthodes,, et de propriétés

3) Quelle portion du code gère la partie "métier" et quelle partie du code s'occupe plus précisément de l'affichage?

Partie métier :

```
class Wine
{
  // propriétés privés de  la classe  Wine
  private $name;
  private $vintage;
  private $price;
}
```

Partie affichage :

<h1>Wine List</h1>

```
<!-- création d'un tableau html pour afficher  la liste des vins -->
<table>
    <thead>
        <tr>
            <th>Name</th>
            <th>Vintage</th>
            <th>Price</th>
        </tr>
    </thead>
    <tbody>
        <!-- la boucle foreach parcourt le tableau $wineList et insère les valeurs dans le tableau html-->
        <?php foreach ($wineList as $wine): ?>
            <tr>
                <td><?php echo $wine->getName() ?></td>
                <td><?php echo $wine->getVintage() ?></td>
                <td><?php echo $wine->getPrice() ?></td>
            </tr>
        <?php endforeach; ?>
    </tbody>
</table>
```

4) Passer la page html à la validation w3c (W3C Markup Validation Service). Lister les éventuelles erreurs et avertissements puis les corriger.

Liste des erreurs :

1. **Warning**: Consider adding a lang attribute to the html start tag to declare the language of this document.
   From line 12, column 16; to line 12, column 28

$this↣**name** = **$name;**↩

For further guidance, consult Declaring the overall language of a page and Choosing language tags.

If the HTML checker has misidentified the language of this document, please file an issue report or send e-mail to report the problem.

2. **Error**: Saw <?. Probable cause: Attempt to use an XML processing instruction in HTML. (XML processing instructions are not supported in HTML.)

   At line 1, column 2

   <**?**php↩//creation

3. **Warning**: The document is not mappable to XML 1.0 due to a trailing hyphen in a comment.

   At line 12, column 15

   ↩ $this->name = $name;↩

4. **Error**: Non-space characters found without seeing a doctype first. Expected <!DOCTYPE html>.

   From line 12, column 16; to line 12, column 28

   $this↣**name** = **$name;**↩

5. **Error**: Element head is missing a required instance of child element title.

   From line 12, column 16; to line 12, column 28

   $this↣**name** = **$name;**↩

   Content model for element head:

   If the document is an iframe srcdoc document or if title information is available from a higher-level protocol: Zero or more elements of metadata content, of which no more than one is a title element and no more than one is a base element.

   Otherwise: One or more elements of metadata content, of which exactly one is a title element and no more than one is a base element.

6. **Error**: Stray doctype.

   From line 48, column 1; to line 48, column 15

   o")↩);↩?>↩<**!DOCTYPE html**>↩<html

7. **Error**: Stray start tag html.

   From line 49, column 1; to line 49, column 6

   YPE html>↩<**html**>↩↩<hea

8. **Error**: Stray start tag head.

   From line 51, column 1; to line 51, column 6

   >↩<html>↩↩<**head**>↩ <

9. **Error**: Element title not allowed as child of element body in this context. (Suppressing further errors from this subtree.)

   From line 52, column 5; to line 52, column 11

   head>↩ <**title**>Wine L

   Contexts in which element title may be used:

   In a head element containing no other title elements.

   Content model for element body:

   Flow content.

10. **Error**: Stray end tag head.

    From line 53, column 1; to line 53, column 7

    t</title>↩</**head**>↩↩<>↩

11. **Error**: Saw <>. Probable causes: Unescaped < (escape as <) or mistyped start tag.

    At line 55, column 2

    tle>↩</head>↩↩<>↩ <h1>Wine

12. **Error**: Saw <?. Probable cause: Attempt to use an XML processing instruction in HTML. (XML processing instructions are not supported in HTML.)

    At line 68, column 14

    >↩ <**?**php foreach ($

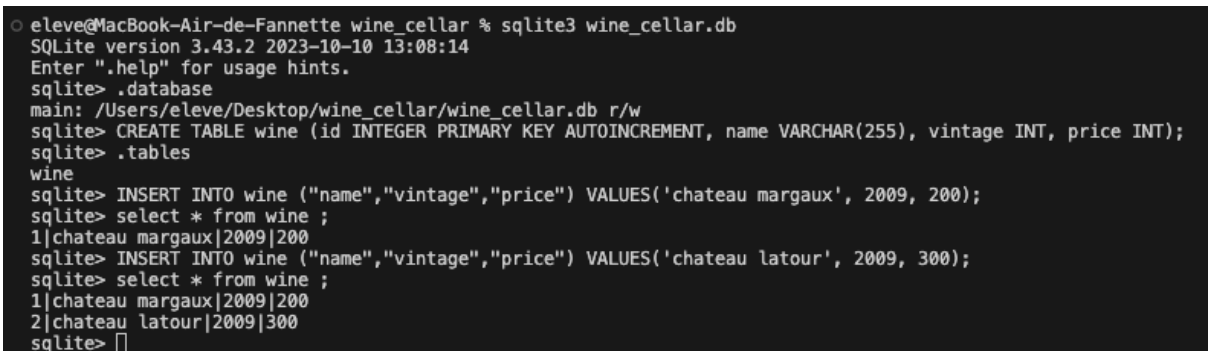
13. **Error**: Saw <?. Probable cause: Attempt to use an XML processing instruction in HTML. (XML processing instructions are not supported in HTML.)
    At line 70, column 26
    <td><?php echo $wine
14. **Warning**: The document is not mappable to XML 1.0 due to a trailing hyphen in a comment.
    At line 70, column 42
    php echo $wine->getName() ?></
15. **Error**: Saw <?. Probable cause: Attempt to use an XML processing instruction in HTML. (XML processing instructions are not supported in HTML.)
    At line 71, column 26
    <td><?php echo $wine
16. **Warning**: The document is not mappable to XML 1.0 due to a trailing hyphen in a comment.
    At line 71, column 42
    php echo $wine->getVintage() ?
17. **Error**: Saw <?. Probable cause: Attempt to use an XML processing instruction in HTML. (XML processing instructions are not supported in HTML.)
    At line 72, column 26
    <td><?php echo $wine
18. **Warning**: The document is not mappable to XML 1.0 due to a trailing hyphen in a comment.
    At line 72, column 42
    php echo $wine->getPrice() ?><
19. **Error**: Saw <?. Probable cause: Attempt to use an XML processing instruction in HTML. (XML processing instructions are not supported in HTML.)
    At line 74, column 14

5) La classe Wine permet-elle de modifier ses attributs après l'instanciation d'un objet? Argumenter votre réponse et, si nécessaire, modifier en conséquence la classe pour que cela soit le cas.

La classe Wine telle qu'elle est actuellement définie **ne permet pas de modifier ses attributs après l'instanciation d'un objet**. En effet, les propriétés name, vintage et price sont déclarées comme **privées** (private), ce qui signifie qu'elles ne peuvent être modifiées ou accédées directement depuis l'extérieur de la classe. De plus, la classe ne fournit pas de méthodes publiques (setters) pour modifier ces propriétés.

Pour permettre la modification des attributs après l'instanciation, il faut ajouter des **méthodes publiques (setters)** à la classe. Ces méthodes permettront de modifier les propriétés privées de manière contrôlée.

Base de données SQLite



```php

<?php
// Connexion à la base de données SQLite
$db = new SQLite3(filename: 'wine_cellar.db');
// Exécution de la requête SQL
$query = "SELECT * FROM wine WHERE price < 400";
$result = $db->query(query: $query);

?>

<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <title>Wine List</title>
    <style>
        table {
            width: 50%;
            border-collapse: collapse;
            margin: 20px 0;
        }

        table,
        th,
        td {
            border: 1px solid black;
        }

        th,
        td {
            padding: 10px;
            text-align: left;
        }
    </style>
</head>

<body>
    <h1>Wine List</h1>
    <!-- création d'un tableau html pour afficher  la liste des vins -->
    <table>
        <thead>
            <tr>
                <th>Id</th>
                <th>Name</th>
                <th>Vintage</th>
                <th>Price</th>
            </tr>
        </thead>
        <tbody>
            <!-- la boucle foreach parcourt le tableau $wineList et insère les valeurs dans le tableau html-->

            <?php while ($vin = $result->fetchArray(mode: SQLITE3_ASSOC)) {
                    echo "<tr>";
                    echo "<td>" . $vin['id'] . "</td>";
                    echo "<td>" . $vin['name'] . "</td>";
                    echo  "<td>" . $vin['vintage'] . "</td>";
                    echo "<td>" . $vin['price'] . "</td>";
                    echo "</tr>";
                }

                // Fermeture de la base de données
                $db->close();
                ?>
        </tbody>
    </table>
</body>

</html>
```

Import CSV

1) Expliquer ce qu'est une méthode statique (faire des recherches si besoin)

En PHP, une méthode statique est une méthode qui appartient à la classe elle-même, plutôt qu'à une instance spécifique de la classe. Cela signifie que vous pouvez appeler cette méthode sans avoir besoin de créer un objet de la classe.

2) Créer le fichier wine_cellar.csv ci-après dans votre répertoire. En s'inspirant du code de l'annexe 1, réaliser le code nécessaire à la lecture du fichier CSV que la méthode getWineListFromCSV() retourne un tableau d'objets Wine populé avec les données du fichier.

3) Transférer la classe Wine dans un fichier séparé nommé Wine.php que vous sauvegardez dans un répertoire nommé "model". Faire les modifications et inclusions nécessaires dans le fichier wine_list.php pour que la page fonctionne toujours à l'identique.

4. Définition du paradigme MVC

Le **MVC** (Modèle-Vue-Contrôleur) est un modèle architectural utilisé en développement logiciel pour séparer les préoccupations (separation of concerns). Il divise une application en trois composants principaux :

- **Modèle (Model)** : Gère les données et la logique métier. Il interagit avec la base de données et contient les règles de gestion des données.
- **Vue (View)** : Gère l'affichage des données. C'est la partie de l'application qui est visible par l'utilisateur (HTML, CSS, etc.).

- **Contrôleur (Controller)** : Gère les interactions entre le modèle et la vue. Il reçoit les entrées de l'utilisateur, interagit avec le modèle pour obtenir les données, puis les transmet à la vue pour affichage.

5. En quoi les modifications de la question 3 nous permettent-elles de nous rapprocher du paradigme MVC ?

En déplaçant la classe Wine dans un fichier séparé (Wine.php) dans un répertoire model, nous avons commencé à séparer les responsabilités :

- **Modèle (Model)** : La classe Wine représente le modèle. Elle gère les données des vins et la logique associée (comme la lecture des données depuis un fichier CSV).
- **Vue (View)** : Le fichier wine_list.php contient principalement le code HTML et CSS pour l'affichage des données. C'est la partie "vue" de l'application.
- **Contrôleur (Controller)** : Bien que nous n'ayons pas encore un contrôleur explicite, la logique de récupération des données depuis la base de données et leur passage à la vue est une étape vers la séparation des préoccupations.

Cette séparation nous rapproche du paradigme MVC en organisant le code de manière plus modulaire et en facilitant la maintenance et l'évolution de l'application.

Fannette SEKO - BTS SIO1 - 27 fév, 2025

Compte rendu du TP PHP 2 Accès backoffice et cybersécurité :

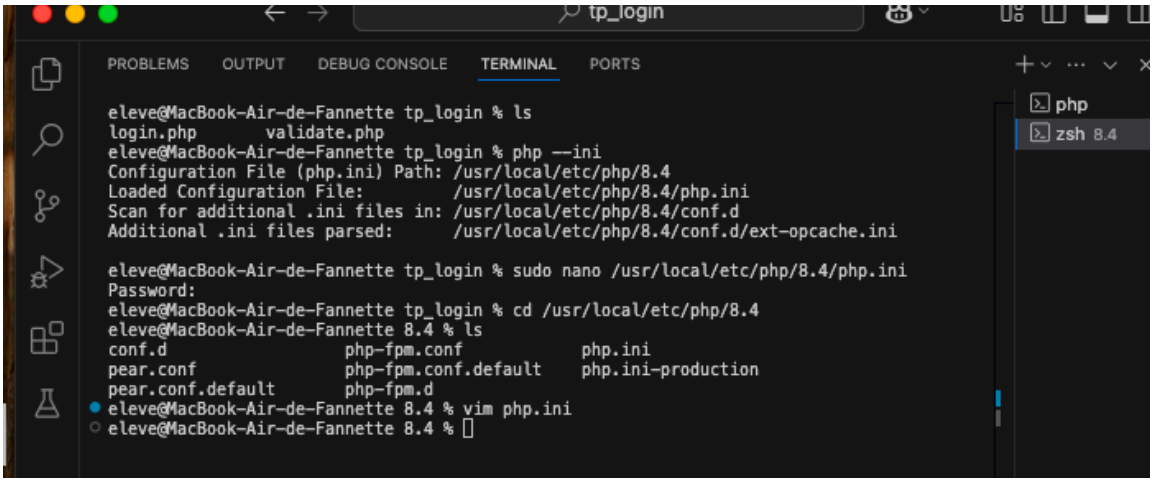
Maquette PHP/HTML

1) Insérer volontairement une erreur de syntaxe en php dans le fichier login.php.

<?php echo "Yes; ?>

Recharger la page. Que se passe-t-il?

Parse error: syntax error, unexpected end of file, expecting variable or "${" or "{$" in /Users/eleve/Desktop/tp_login/login.php on line 21



2) En déduire l'intérêt d'activer les options error_reporting et display_errors, via le fichier de configuration php.ini, dans le cadre des développements?

Activer les options error_reporting et display_errors dans le fichier php.ini est essentiel pendant le développement. Cela permet d'afficher les erreurs PHP directement dans le navigateur ou la sortie CLI, ce qui facilite le débogage en identifiant rapidement les problèmes tels que les erreurs de syntaxe, les variables non définies ou les appels de fonction incorrects. De plus, configurer error_reporting pour inclure tous les types d'erreurs (E_ALL) garantit que même les avertissements et les notices sont détectés, aidant ainsi à écrire un code plus robuste et à éviter les comportements inattendus

3) Quelle est la différence entre la méthode GET et la méthode POST, utilisée ici, pour le passage de paramètres lors de l'envoi de requêtes HTTP?

La méthode GET envoie les données dans l'URL, ce qui les rend visibles et adaptées pour des requêtes simples comme les recherches. Elle est limitée en taille et moins sécurisée. En revanche, la méthode POST envoie les données dans le corps de la requête, ce qui les rend invisibles et idéales pour envoyer des informations sensibles ou volumineuses, comme les formulaires de connexion dans notre cas. En résumé, GET est utilisé pour récupérer des données, tandis que POST est utilisé pour en envoyer de manière plus sécurisée.

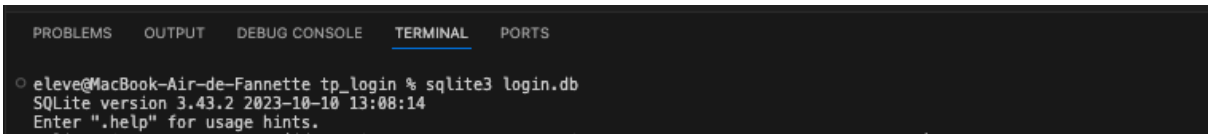
4) Pourquoi l'usage de la méthode POST est-elle plus sécurisée pour notre page de login?

L'usage de la méthode POST est plus sécurisé pour une page de login car les données (comme le nom d'utilisateur et le mot de passe) sont envoyées dans le corps de la requête HTTP, plutôt que dans l'URL. Cela signifie qu'elles ne sont pas visibles dans la barre d'adresse du navigateur, ni enregistrées dans l'historique ou

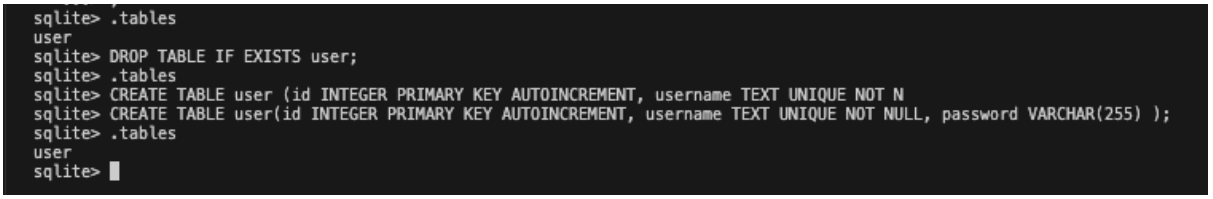
les logs du serveur. La méthode POST réduit les risques d'interception ou de fuite d'informations sensibles, offrant ainsi une meilleure protection pour les identifiants de connexion.

Base de données SQLite

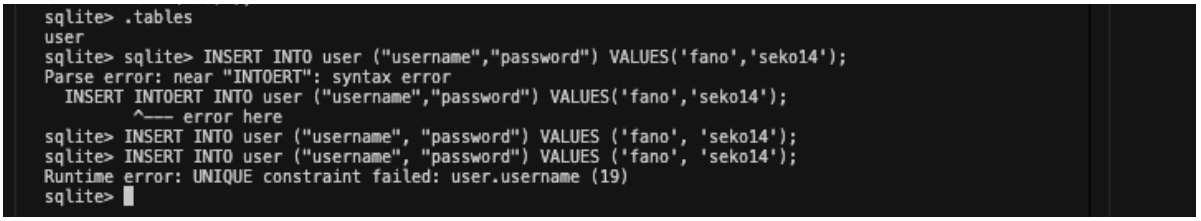
1) Créer une base de données login.db dans votre répertoire de travail.



2) Créer une table nommée user avec les champs suivants: username, password. Le champ username doit être unique.



3) Populer la table user avec un enregistrement admin/password. Quelle message d'erreur s'affiche si l'on essaie d'ajouter le même enregistrement à nouveau?



Cybersécurité

1) L’activation des messages d’erreur peut-elle poser un problème de sécurité sur un serveur en production? Expliquer pourquoi.

Oui, l'activation des messages d'erreur en production pose un risque de sécurité. Voici pourquoi :

- Exposition d'informations sensibles : Les messages d'erreur PHP peuvent révéler des détails sur la structure du serveur, les chemins des fichiers, les requêtes SQL, ou même des informations de configuration. Ces informations peuvent être exploitées par des attaquants pour identifier des vulnérabilités.
- Aide aux attaquants : Par exemple, une erreur SQL peut afficher une partie de la requête, ce qui permet à un attaquant de comprendre la structure de la base de données et de préparer une injection SQL.
- Expérience utilisateur dégradée : Les utilisateurs finaux n'ont pas besoin de voir des messages d'erreur techniques. Cela peut les perturber et nuire à la crédibilité du site.

Solution : En production, désactivez l'affichage des erreurs (display_errors = Off) et enregistrez-les dans un fichier de log (log_errors = On). Cela permet de surveiller les erreurs sans les exposer.

2) Sur notre serveur de test, les requêtes sont effectuées via le protocole HTTP. Effectuer une recherche sur internet afin d’expliquer, en quelques lignes, en quoi le protocole HTTPS sécurise les échanges dans le cadre d’une mise en production d’un site web.

Le protocole HTTPS (HyperText Transfer Protocol Secure) sécurise les échanges entre le client (navigateur) et le serveur en utilisant le chiffrement TLS/SSL. Voici ses avantages :

- Chiffrement des données : HTTPS chiffre les données échangées (comme les mots de passe, informations de paiement, etc.), empêchant les attaquants de les intercepter ou de les lire.
- Authentification du serveur : HTTPS garantit que le client communique bien avec le bon serveur, grâce à des certificats numériques. Cela évite les attaques de type man-in-the-middle.
- Intégrité des données : HTTPS assure que les données ne sont pas modifiées pendant leur transmission.
- Confiance utilisateur : Les navigateurs affichent un cadenas vert pour les sites HTTPS, ce qui renforce la confiance des utilisateurs.

Conclusion : Pour une mise en production, HTTPS est essentiel pour protéger les données sensibles, garantir la confidentialité et assurer la conformité aux standards de sécurité modernes.

3) Sur votre page de login, entrez maintenant le texte suivant comme nom d'utilisateur puis valider avec ou sans entrer un mot de passe…

' or 1==1; --

Que se passe-t-il?

Login successful!

Si vous entrez ' or 1==1; -- comme nom d'utilisateur, cela peut permettre une injection SQL. Voici ce qui se produit :

- La chaîne ' or 1==1; -- est interprétée comme une partie de la requête SQL.
- Le ' ferme la chaîne de caractères dans la requête SQL.
- Le or 1==1 rend la condition toujours vraie (1==1 est toujours vrai).
- Le ; -- termine la requête et commente le reste de la ligne, annulant toute vérification supplémentaire (comme le mot de passe).

Cela permet à un attaquant de se connecter sans connaître un nom d'utilisateur ou un mot de passe valide.

4) Écrire la requête SQL résultante et analyser ce qu'il s'est produit.

SELECT * FROM users WHERE username='' or 1==1; --' AND password='';

Analyse :

- username='' or 1==1 : La condition est toujours vraie, car 1==1 est vrai.
- -- : Tout ce qui suit est ignoré (commentaire SQL).
- Résultat : La requête renvoie tous les utilisateurs de la table users, permettant à l'attaquant de se connecter sans informations valides.

5) Proposer une idée de patch correctif du code php afin de sécuriser l'identification.

Pour éviter les injections SQL, utilisez des **requêtes préparées** (prepared statements) avec **PDO** ou **MySQLi**. Voici un exemple avec PDO :

Pourquoi cela fonctionne :

- Les **requêtes préparées** séparent les données (entrées utilisateur) de la structure de la requête SQL.
- Les entrées utilisateur sont traitées comme des données brutes et non comme du code SQL, ce qui empêche les injections.

Bonus : Autres bonnes pratiques

1. **Hachage des mots de passe** :
    - Utilisez des fonctions de hachage sécurisées comme password_hash() et password_verify() pour stocker et vérifier les mots de passe.

2. php

   Copy

   $hashed_password = password_hash($password, PASSWORD_DEFAULT);
3. **Validation des entrées** :
   - Validez et filtrez les entrées utilisateur pour éviter les caractères dangereux.
4. **Utilisation de HTTPS** :
   - Assurez-vous que toutes les communications sont chiffrées avec HTTPS pour protéger les données en transit.

En appliquant ces mesures, vous protégez votre application contre les injections SQL et autres attaques courantes

Bonus simulation de cyberattaque avec sqlmap

6) sqlmap (https://sqlmap.org/) est un outil de test de pénétration open source développé en Python. Il permet d'automatiser le processus de détection et d'exploitation de failles pour la prise de contrôle des serveurs de bases de données.
Sur quel type de failles opère-t-il?
Télécharger l'outil sqlmap sur le site et décompresser l'archive dans votre répertoire de travail sous le nom sqlmap.

7) Se mettre dans le répertoire sqlmap et créer le fichier texte suivant:

Fannette SEKO - BTS SIO1 - 03 mars, 2025

Compte rendu du TP3 PHP Cave à vins - Edition back office

Les formulaires

Maquette PHP/HTML

1- Test d'un formulaire de vin

1) Tester le code et expliquer ce qu'il se passe lorsque l'on valide le formulaire.

Lorsqu'on valide le formulaire les données sont enregistrées et on obtient un message de retour : **Wine was successfully edited!**

2) Quelle méthode est utilisée pour passer les paramètres dans la requête HTTP?

La méthode **Post**.

3) Faire une recherche et expliquer le rôle de la fonction preg_match() utilisée dans validation.php.

La fonction **preg_match()** est le plus souvent utilisée en cas de recherche de caractère.

La fonction preg_match() est couramment utilisée pour rechercher des motifs spécifiques dans une chaîne de caractères à l'aide d'une expression régulière (regex). Elle prend deux arguments principaux :

1. Une expression régulière (le motif à rechercher).
2. Une chaîne de caractères à vérifier.

La fonction retourne :

- 1 si une correspondance est trouvée.
- 0 si aucune correspondance n'est trouvée.
- false en cas d'erreur.

Dans le cas de notre formulaire, preg_match() joue un rôle de vérificateur. Elle permet de valider les données saisies par l'utilisateur avant de les traiter. Par exemple, si un champ ne respecte pas le format attendu, la fonction peut renvoyer un message d'erreur.

EXEMPLE :

```
// check if country field is valid
if (!preg_match("/^[a-zA-Z-'\s]+$/", $country)) {
  echo "Country field is not valid.";
  exit;
}
```

Explication de l'expression régulière

- ^ : Début de la chaîne.
- [a-zA-Z-'\s] : Autorise les lettres majuscules et minuscules (a-zA-Z), les tirets (-), les apostrophes (') et les espaces (\s).
- + : Au moins un caractère doit correspondre.
- $: Fin de la chaîne.

Ainsi, cette expression régulière vérifie que le champ country ne contient que des lettres, des tirets, des apostrophes et des espaces. Si ce n'est pas le cas, un message d'erreur est affiché.

4) Pour des besoins d'internationalisation, notre projet doit pouvoir être facilement traduit par la suite dans différentes langues. Dans fichier nommé translate_fr.php, franciser les champs de la page HTML et les messages en créant un tableau de traduction (clé/valeur) nommé translate. Modifier edit.php et validation.php en conséquence pour afin d'exploiter le tableau.

translate_fr.php :

```php
<?php
// Tableau de traduction en français
$translate = [
  // Libellés du formulaire
  'name_label' => 'Nom',
  'type_label' => 'Type',
  'country_label' => 'Pays',
  'region_label' => 'Région',
  'year_label' => 'Année',
```

```
    'submit_button' => 'Modifier le vin',

    // Options du type de vin
    'red' => 'Rouge',
    'white' => 'Blanc',
    'rose' => 'Rosé',

    // Messages d'erreur
    'error_all_fields_required' => 'Veuillez remplir tous les champs.',
    'error_name_invalid' => 'Le champ nom n\'est pas valide.',
    'error_type_invalid' => 'Le champ type n\'est pas valide.',
    'error_country_invalid' => 'Le champ pays n\'est pas valide.',
    'error_region_invalid' => 'Le champ région n\'est pas valide.',
    'error_year_invalid' => 'Le champ année n\'est pas valide.',
    'success_message' => 'Le vin a été modifié avec succès !',
];
?>
```

edit.php :

```
<!-- HTML Form -->
<?php

$lang = $_GET["lang"] ?? "fr";
```

```
$lang_path = "translate_" . $lang . ".php";

require_once($lang_path);
?>
<form action="validation.php" method="post">

  <label><?php echo $translate['name_label']; ?>:</label>
              <input type="text" name="name" placeholder="<?php echo $translate['name_label']; ?>"><br>
  <label><?php echo $translate['type_label']; ?>:</label>
  <select name="type">
     <option value="">--- <?php echo $translate['type_label']; ?> ---</option>
     <option value="Red"><?php echo $translate['red']; ?></option>
     <option value="White"><?php echo $translate['white']; ?></option>
     <option value="Rose"><?php echo $translate['rose']; ?></option>
  </select><br>

  <label><?php echo $translate['country_label']; ?>:</label>
              <input type="text" name="country" placeholder="<?php echo $translate['country_label']; ?>"><br>

  <label><?php echo $translate['region_label']; ?>:</label>
              <input type="text" name="region" placeholder="<?php echo $translate['region_label']; ?>"><br>

  <label><?php echo $translate['year_label']; ?>:</label>
              <input type="number" name="year" placeholder="<?php echo $translate['year_label']; ?>"><br>
```

```
  <input type="submit" value="<?php echo $translate['submit_button']; ?>">
</form>
```

validation.php :

```
<?php
$lang = $_GET["lang"] ?? "fr";

$lang_path = "translate_" . $lang . ".php";

require_once($lang_path);

$name = $_POST['name'];
$type = $_POST['type'];
$country = $_POST['country'];
$region = $_POST['region'];
$year = $_POST['year'];
// check if all fields are filled
if (
  empty($name) || empty($type) || empty($country) || empty($region) ||
  empty($year)
) {
  echo "error_all_fields_required";
  exit;
}
```

```
// check if name field is valid
if (!preg_match("/^[a-zA-Z-'\s]+$/", $name)) {
  echo "error_name_invalid";
  exit;
}
// check if type field is valid
if ($type != "Red" && $type != "White" && $type != "Rose") {
  echo "error_type_invalid";
  exit;
}
// check if country field is valid
if (!preg_match("/^[a-zA-Z-'\s]+$/", $country)) {
  echo "error_country_invalid";
  exit;
}
// check if region field is valid
if (!preg_match("/^[a-zA-Z-'\s]+$/", $region)) {
  echo "error_region_invalid";
  exit;
}
// check if year field is valid
if (!preg_match("/^[0-9]+$/", $year)) {
  echo "error_year_invalid";
  exit;
}
// edit wine information
// ...
// return success message
echo "success_message";
```

2- Les inputs

Les inputs dans un formulaire permettent de renseigner différents types de données.

1) Faire une recherche sur internet et lister tous les types de balises <input> qui existent en HTML.

- <input type=submit>
- <input type=text>
- <input type=password>
- <input type=radio>
- <input type=checkbox>
- <input type=date>
- <input type=range>
- <input type=hidden>
- <input type=number>
- <input type=url>
- <input type=email>
- <input type=month>
- <input type=time>
- <input type=image>
- <input type=color>
- <input type=file>

- <input type="datetime-local">
- <input type="week">
- <input type="search">
- <input type="tel">
- <input type="reset">
- <input type="button">

2) Pour chaque type de balise, rechercher un exemple de code.

1. **<input type="text">**
    a. **Description** : Champ de texte pour saisir du texte sur une seule ligne.
    b. **Exemple** :
    <input type="text" name="username" placeholder="Entrez votre nom">
2. **<input type="password">**
    a. **Description** : Champ de texte pour saisir un mot de passe (le texte est masqué).
    b. **Exemple** :
    <input type="password" name="password" placeholder="Entrez votre mot de passe">
3. **<input type="email">**
    a. **Description** : Champ pour saisir une adresse e-mail (validation automatique).
    b. **Exemple** :
    <input type="email" name="email" placeholder="Entrez votre e-mail">
4. **<input type="number">**
    a. **Description** : Champ pour saisir un nombre (avec des flèches pour augmenter/diminuer la valeur).

b. **Exemple** :

<input type="number" name="age" placeholder="Entrez votre âge">

5. **<input type="date">**
   a. **Description** : Champ pour saisir une date (avec un sélecteur de date).
   b. **Exemple** :

   <input type="date" name="birthdate">

6. **<input type="time">**
   a. **Description** : Champ pour saisir une heure (avec un sélecteur d'heure).
   b. **Exemple** :

   <input type="time" name="appointment_time">

7. **<input type="datetime-local">**
   a. **Description** : Champ pour saisir une date et une heure (sans fuseau horaire).
   b. **Exemple** :

   <input type="datetime-local" name="meeting_time">

8. **<input type="month">**
   a. **Description** : Champ pour saisir un mois et une année.
   b. **Exemple** :

   <input type="month" name="start_month">

9. **<input type="week">**
   a. **Description** : Champ pour saisir une semaine et une année.
   b. **Exemple** :

   <input type="week" name="vacation_week">

10. **<input type="url">**
    a. **Description** : Champ pour saisir une URL (validation automatique).

b. **Exemple** :

<input type="url" name="website" placeholder="Entrez votre site web">

c.

11. **<input type="search">**
    a. **Description** : Champ de recherche (peut inclure une icône de recherche).
    b. **Exemple** :

<input type="search" name="query" placeholder="Rechercher...">

12. **<input type="tel">**
    a. **Description** : Champ pour saisir un numéro de téléphone.
    b. **Exemple** :

<input type="tel" name="phone" placeholder="Entrez votre numéro de téléphone">

13. **<input type="color">**
    a. **Description** : Champ pour sélectionner une couleur (avec un sélecteur de couleur).
    b. **Exemple** :

<input type="color" name="fav_color">

14. **<input type="range">**
    a. **Description** : Curseur pour sélectionner une valeur dans une plage.
    b. **Exemple** :

<input type="range" name="volume" min="0" max="100">

15. **<input type="file">**
    a. **Description** : Champ pour téléverser un fichier.
    b. **Exemple** :

<input type="file" name="document">

17. **<input type="checkbox">**

a. **Description** : Case à cocher pour sélectionner une ou plusieurs options.
b. **Exemple** :
c. <input type="checkbox" name="subscribe" value="yes"> S'abonner à la newsletter

18. <input type="radio">

a. **Description** : Bouton radio pour sélectionner une seule option parmi plusieurs.
b. **Exemple** :

<input type="radio" name="gender" value="male"> Homme

1. <input type="radio" name="gender" value="female"> Femme

19. <input type="submit">

a. **Description** : Bouton pour soumettre un formulaire.
b. **Exemple** :

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

20.<input type="reset">

a. **Description** : Bouton pour réinitialiser un formulaire.
b. **Exemple** :

<input type="reset" value="Réinitialiser">

21. <input type="button">

a. **Description** : Bouton générique (souvent utilisé avec JavaScript).
b. **Exemple** :

<input type="button" value="Cliquez-moi" onclick="alert('Bonjour !')">

24. <input type="image">

a. **Description** : Bouton de soumission sous forme d'image.

b. **Exemple** :

<input type="image" src="submit.png" alt="Soumettre">

25. <input type="hidden">

a. **Description** : Champ caché pour stocker des données invisibles à l'utilisateur.

b. **Exemple** :

<input type="hidden" name="user_id" value="12345">

26.<input type="datetime-local">

a. **Description** : Permet de saisir une date et une heure (sans fuseau horaire).

b. **Exemple** :

<input type="datetime-local" name="meeting_time">

27. <input type="week">

a. **Description** : Permet de saisir une semaine et une année.

b. **Exemple** :

<input type="week" name="vacation_week">

i.

28. <input type="search">

a. **Description** : Champ de recherche (peut inclure une icône de recherche et un bouton pour effacer le texte).

b. **Exemple** :

<input type="search" name="query" placeholder="Rechercher...">

29.<input type="tel">

a. **Description** : Champ pour saisir un numéro de téléphone (peut être validé par le navigateur sur les appareils mobiles).

b. **Exemple** :

<input type="tel" name="phone" placeholder="Entrez votre numéro de téléphone">

30. <input type="reset">

a. **Description** : Bouton pour réinitialiser un formulaire (efface toutes les données saisies).

b. **Exemple** :

<input type="reset" value="Réinitialiser">

31. **<input type="button">**

a. **Description** : Bouton générique qui ne soumet pas le formulaire (souvent utilisé avec JavaScript pour déclencher des actions).

b. **Exemple** :

<input type="button" value="Cliquez-moi" onclick="alert('Bonjour !')">

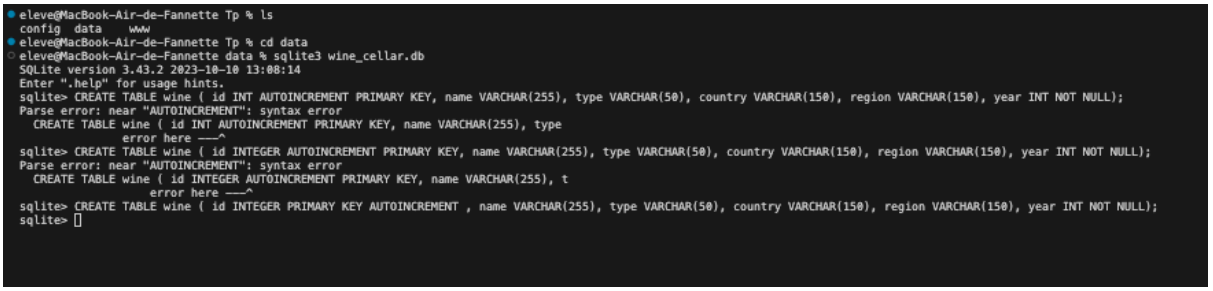
3- Base de données

1) Dans le répertoire data, créer une base de données nommée “wine_cellar.db”.

2) Pourquoi avoir revisité la structure de notre site web?

mvc

3) Dans la base de données, créer une table nommée wine qui reprend la structure d’un vin de notre maquette avec les champs suivants: id (identifiant unique, auto-incrémenté), name, type, country, region, year.



4) Nous désirons à présent ajouter les données saisies, dans le formulaire HTM dans la table wine de la base de données. Modifier le fichier validation.php en conséquence.

5) Ajouter le code suivant dans la partie formulaire du fichier edit.php…

Extrait de edit.php

<input name="id" type="hidden" value="<?php echo @$_GET['id'] ?>">

Recharger votre page web avec l'URL: http://localhost:8080/edit.php?id=23

Afficher le code source de la page (via clic droit de votre navigateur…) et expliquer ce qu'il se produit dans l'attribut value de cet input.

6) À quoi sert le caractère @ en PHP? -supprimer le et tester afin de voir ce qu'il se passe :

Le caractère `@` en PHP est utilisé pour supprimer les messages d'erreur ou d'avertissement qui pourraient être générés par une expression. Si vous supprimez le `@` et que `$_GET['id']` n'est pas défini, une notice PHP sera affichée.

7) Expliquer l'intérêt du type hidden après avoir visité le lien suivant:

Le type `hidden` permet de stocker des données dans un formulaire sans les afficher à l'utilisateur. Cela est utile pour passer des informations supplémentaires (comme un ID) entre les pages sans que l'utilisateur ne les voie ou ne les modifie.

8) Nous souhaitons que notre formulaire permette à la fois d’éditer un vin existant (id renseigné) ou de créer une nouvelle fiche (id non renseigné). Modifier le code source du fichier validation.php en conséquence et expliquer votre démarche

Fannette SEKO - BTS SIO1 - 10 mars, 2025

Compte rendu du TP Cave à vins - Le M de MVC (partie 1) :

Les cookies

1) Après avoir testé le code, expliquons le principe de fonctionnement des cookies et l'usage du tableau global $_COOKIE en PHP :

Principe de fonctionnement des cookies

Les cookies sont de petits fichiers texte stockés sur l'ordinateur de l'utilisateur par le navigateur web. Ils sont utilisés pour conserver des informations entre les sessions de navigation, comme les préférences de l'utilisateur, les identifiants de session, ou d'autres données utiles pour personnaliser l'expérience utilisateur.

Fonctionnement des cookies :

1. **Création** : Un cookie est créé côté serveur et envoyé au navigateur via l'en-tête HTTP Set-Cookie.
2. **Stockage** : Le navigateur stocke le cookie sur l'ordinateur de l'utilisateur.
3. **Envoi** : À chaque requête ultérieure vers le même domaine, le navigateur envoie automatiquement le cookie au serveur via l'en-tête HTTP Cookie.
4. **Expiration** : Les cookies peuvent avoir une durée de vie limitée (expiration) ou être supprimés manuellement par l'utilisateur ou le serveur.

Lorsqu'un utilisateur charge une page, un cookie peut être créé et enregistré sur son navigateur grâce à la méthode update_cookie(). Ce cookie est ensuite récupéré à l'aide de la méthode cookie($name), qui permet d'accéder à sa valeur en utilisant le tableau global $_COOKIE.

À chaque fois que l'utilisateur recharge la page, la méthode cookieIsSet($name) vérifie si le cookie existe déjà. Si c'est le cas, la méthode update_cookie() peut être utilisée pour mettre à jour la valeur du cookie ou prolonger sa durée de vie. Par exemple, si le cookie est configuré pour expirer après un certain délai (défini par Cookie::$delay), cette méthode permet de réinitialiser ce délai.

Enfin, si nécessaire, le cookie peut être supprimé à l'aide de la méthode clean_cookie($cookie_name). Cette méthode supprime le cookie du tableau $_COOKIE et envoie une instruction au navigateur pour qu'il supprime le cookie en définissant une date d'expiration dans le passé.

En PHP, le tableau global $_COOKIE est utilisé pour accéder aux cookies envoyés par le navigateur. Ce tableau est associatif, où les clés sont les noms des cookies et les valeurs sont les données stockées dans ces cookies.

2) Faire une recherche et expliquer à quoi sert le mot clé “require” en PHP.

Le mot-clé require en PHP est utilisé pour inclure et exécuter un fichier PHP dans un autre fichier. Il est similaire à include, mais avec une différence importante :

- Si le fichier spécifié n'est pas trouvé ou ne peut pas être inclus, require génère une **erreur fatale** (E_COMPILE_ERROR) et arrête l'exécution du script.
- En revanche, include génère seulement un avertissement (E_WARNING) et le script continue.

3) Faire une recherche et expliquer à quoi sert le mot clé “use” en PHP.

Le mot-clé use en PHP est utilisé pour :

1. **Importer des classes, interfaces, traits ou fonctions** dans un espace de noms (namespace) courant.
2. **Aliaser des classes ou des noms** pour éviter les conflits de noms ou simplifier l'utilisation de classes avec des noms longs.

4) Analyser et expliquer ce que fait la méthode statique afficher() de la classe Debogage.

La méthode statique afficher() de la classe Debogage est un outil de débogage qui permet d'afficher de manière lisible et structurée le contenu d'une variable ou d'une structure de données (comme un tableau ou un objet).

Fonctionnement :

- Elle prend un paramètre $value (la variable à afficher).
- Elle utilise var_dump() pour afficher le type et la valeur de la variable.
- Le résultat est encapsulé dans une balise <pre> pour un affichage formaté dans le navigateur.

5) Faire une recherche et expliquer ce qu'est une méthode statique.

Une méthode statique est une méthode qui appartient à la classe elle-même plutôt qu'à une instance de la classe. Elle peut être appelée sans créer d'objet de la classe.

Caractéristiques :

- Déclarée avec le mot-clé static.
- Accédée en utilisant l'opérateur :: (par exemple, Classe::methode()).
- Ne peut pas accéder aux propriétés ou méthodes non statiques de la classe (car elles nécessitent une instance).

6) Nous souhaitons encapsuler la gestion des cookies dans une classe dédiée de notre modèle. Pour ce faire, nous disposons d'une classe Cookie déjà précodée. Modifier le code de la page index.php afin qu'elle n'utilise plus directement le tableau $_COOKIE mais qu'elle passe par notre classe Cookie.

```php
<?php

require '../model/Cookie.php';

require '../model/Debogage.php';




use Model\Cookie;

use Model\Debogage;

// On vérifie si le cookie existe

if  (!Cookie::cookieIsSet('count')) {

   // Creation de la variable count
```

```
    $count = 1;

    // Note: Lorsqu'un cookie est créé depuis une page,

    // il ne devient disponible qu'à partir du chargement de la page

    // suivante car il faut que le navigateur envoit le cookie au serveur.

} else {

    // Incrementation de la variable count

    $count = Cookie::cookie('count') + 1;

}

// Mise à jour du cookie sur le navigateur

// Note: Un cookie dont la date d'expiration n'est pas précisée est enregistré

// dans la mémoire vive de l'ordinateur et non sur le disque dur. Il sera

// effacé à la fermeture du navigateur.

// https://www.php.net/manual/fr/function.setcookie.php

Cookie::update_cookie('count', $count);?>

<hr>

<?php

Debogage::afficher($_COOKIE);

?>

<hr>

<p>

    Salut visiteur! Vous avez chargé cette page

    <?php

    echo $count;

    ?>

    fois.

</p>
```

7) En quoi la classe Cookie pourrait-elle nous faciliter le codage si, par la suite, nous étions contraints d'utiliser des sessions (ex: besoin de plus de 4 Ko de stockage)?

La classe Cookie encapsule la gestion des cookies, ce qui facilite la transition vers les sessions si nécessaire :

- Il suffit de modifier les méthodes internes de la classe pour utiliser $_SESSION au lieu de $_COOKIE.
- Le reste du code (comme index.php) n'a pas besoin d'être modifié, car il utilise l'interface de la classe Cookie.

8) Vous pouvez constater que le code de la classe Cookie a été documenté avec une syntaxe bien particulière (/ */, @param, @return). Il s'agit d'un balisage pour le logiciel Doxygen. Faire une recherche. Expliquer le principe de cet outil et son intérêt lors du codage d'un projet en équipe.**

Doxygen est un outil de génération de documentation à partir du code source. Il analyse les commentaires structurés (comme /** */) pour produire une documentation au format HTML, PDF, etc.

Syntaxe Doxygen :

- /** ... */ : Commentaire de documentation.
- @param : Documente un paramètre de fonction.
- @return : Documente la valeur de retour.

Intérêt en équipe :

- Facilite la compréhension du code.
- Permet de générer une documentation à jour automatiquement.
- Standardise la documentation pour une meilleure collaboration.

Nos vins dans le modèle

Dans le cadre de notre cave à vins, une bouteille de vin constitue l'élément central de notre stock. Nous allons donc créer une classe à cet effet et l'enrichir.

1) Dans le répertoire "model", créer un fichier nommé "Bouteille.php" et y déclarer une classe Bouteille. -vous inspirer des classes Cookie et Debogage pour la syntaxe PHP-

```php
<?php

/*************************************************************************

****

*

* C L A S S E B O U T E I L L E

*

*/

// La classe fait partie de l'espace de nommage Model du paradigme MVC

namespace Model;
```

```php
/**
* Cette classe représente une bouteille de vin dans notre cave.
*/
class Bouteille
{
   // Attributs privés
   private $nom;
   private $type;
   private $pays;
   private $region;
   private $annee;

   // Constructeur (optionnel)
   public function __construct($nom, $type, $pays, $region, $annee) {
       $this->nom = $nom;
       $this->type = $type;
       $this->pays = $pays;
       $this->region = $region;
       $this->annee = $annee;
   }

   // Accesseurs (getters)
   public function getNom() { return $this->nom; }
   public function getType() { return $this->type; }
   public function getPays() { return $this->pays; }
   public function getRegion() { return $this->region; }
```

```php
    public function getAnnee() { return $this->annee; }



    // Mutateurs (setters)

    public function setNom($nom) { $this->nom = $nom; }

    public function setType($type) { $this->type = $type; }

    public function setPays($pays) { $this->pays = $pays; }

    public function setRegion($region) { $this->region = $region; }

    public function setAnnee($annee) { $this->annee = $annee; }

}
```

2) Ajouter le code nécessaire pour déclarer que la classe fait partie de l'espace de nommage Model.

namespace Model;

3) Ajouter les attributs privés suivant: nom, type, pays, region, année.

private $nom;

private $type;

private $pays;

private $region;

private $annee;

4) Ajouter les accesseurs et mutateurs correspondants.

```php
// Accesseurs (getters)

   public function getNom() { return $this->nom; }

   public function getType() { return $this->type; }

   public function getPays() { return $this->pays; }

   public function getRegion() { return $this->region; }

   public function getAnnee() { return $this->annee; }



   // Mutateurs (setters)

   public function setNom($nom) { $this->nom = $nom; }

   public function setType($type) { $this->type = $type; }

   public function setPays($pays) { $this->pays = $pays; }

   public function setRegion($region) { $this->region = $region; }

   public function setAnnee($annee) { $this->annee = $annee; }
```

5) Tester votre classe avec le fichier suivant et documenter le code.

```php
<?php

/**************************************************************************

****

*

* C L A S S E B O U T E I L L E

*

*/
```

```php
// La classe fait partie de l'espace de nommage Model du paradigme MVC

namespace Model;



/**

* Cette classe représente une bouteille de vin dans notre cave.

*/

class Bouteille

{

   // Attributs privés

   private $nom;

   private $type;

   private $pays;

   private $region;

   private $annee;



   // Constructeur (optionnel)

   public function __construct($nom, $type, $pays, $region, $annee) {

       $this->nom = $nom;

       $this->type = $type;

       $this->pays = $pays;

       $this->region = $region;

       $this->annee = $annee;

   }



   // Accesseurs (getters)

   public function getNom() { return $this->nom; }
```

```php
    public function getType() { return $this->type; }

    public function getPays() { return $this->pays; }

    public function getRegion() { return $this->region; }

    public function getAnnee() { return $this->annee; }



    // Mutateurs (setters)

    public function setNom($nom) { $this->nom = $nom; }

    public function setType($type) { $this->type = $type; }

    public function setPays($pays) { $this->pays = $pays; }

    public function setRegion($region) { $this->region = $region; }

    public function setAnnee($annee) { $this->annee = $annee; }

}
```

6) Quelle partie du code s'occupe ici de faire l'interface avec la partie Model de notre projet?

La classe `Bouteille` est la partie Model du projet. Elle encapsule les données et la logique métier liées à une bouteille de vin. L'interface avec le Model se fait via les accesseurs et mutateurs.

7) Quelle partie du code s'occupe de la partie View?

```php
foreach ($cave as $bouteile) {

      echo $bouteile->getNom() . "<br>";
```

8) Modifier la boucle foreach afin d'afficher tous les attributs de chaque vin dans un tableau HTML (<table>) avec les entêtes de colonnes: Nom, Type, Pays, Région, Année.

```php
<?php
   foreach ($cave as $bouteile) {
       echo '<tr>';
       echo $bouteile->getNom() . "<br>";
       echo $bouteile->getType() . "<br>";
       echo $bouteile->getPays() . "<br>";
       echo $bouteile->getRegion() . "<br>";
       echo $bouteile->getAnnee() . "<br>";
       echo '</tr>';
   }

   ?>
```

Fannette SEKO - BTS SIO1 - 06 mars, 2025

Compte rendu du TP Cave à vins - Le M de MVC (partie 2) :

1- Documentation du modèle

1. Faire une recherche et expliquer l'origine et le principe des tags PHPDoc.

Les tags PHPDoc sont des annotations utilisées dans le code PHP pour documenter les classes, méthodes, propriétés, fonctions, et autres éléments du code. Ils permettent de fournir des informations structurées et standardisées sur le code, ce qui facilite la compréhension, la maintenance, et l'utilisation de celui-ci. Ces tags sont souvent utilisés par des outils de documentation automatique comme **phpDocumentor**, mais aussi par des IDE (Environnements de Développement Intégrés) comme PHPStorm, VSCode, ou NetBeans pour fournir des fonctionnalités d'autocomplétion, de vérification de type, et d'aide à la navigation.

Origine des tags PHPDoc

PHPDoc est inspiré de **Javadoc**, un système de documentation utilisé en Java. L'idée est de permettre aux développeurs d'intégrer directement dans le code source des commentaires qui décrivent le comportement, les paramètres, les types de retour, et d'autres informations pertinentes. Ces commentaires sont ensuite extraits par des outils pour générer une documentation technique complète.

PHPDoc a été introduit dans la communauté PHP pour répondre au besoin de standardisation de la documentation, surtout dans les projets complexes où la compréhension du code est cruciale.

Principe des tags PHPDoc

Les tags PHPDoc sont intégrés dans des blocs de commentaires spéciaux qui commencent par `/**` et se terminent par `*/`. Ces blocs doivent précéder l'élément qu'ils documentent (classe, méthode, propriété, etc.). Chaque tag commence par le symbole `@` suivi d'un mot-clé qui indique le type d'information fournie.

Structure de base d'un bloc PHPDoc

```
/**
* Description courte de l'élément.
*
* Description détaillée de l'élément, si nécessaire.
*
* @tag1 valeur1
* @tag2 valeur2
* ...
*/
```

Tags PHPDoc courants

Voici quelques-uns des tags PHPDoc les plus couramment utilisés :

1. **@param** : Décrit un paramètre d'une fonction ou méthode.
    - Syntaxe : `@param type $nom_paramètre description`

Exemple :

```
/**
* @param int $id L'identifiant de l'utilisateur.
*/
function getUser($id) {
   // ...
}
```

2. **@return** : Décrit la valeur de retour d'une fonction ou méthode.
    - Syntaxe : `@return type description`

Exemple :

```
/**
* @return string Le nom de l'utilisateur.
*/
function getName() {
   return $this->name;
}
```

3. **@var** : Décrit le type d'une propriété de classe.
    - Syntaxe : `@var type description`

Exemple :

```
/**
* @var string Le nom de l'utilisateur.
*/
public $name;
```

4. **@throws** : Indique qu'une méthode ou fonction peut lancer une exception.
    - Syntaxe : `@throws type description`

Exemple :

```
/**
*   @throws   InvalidArgumentException   Si   l'identifiant   est
invalide.
*/
function getUser($id) {
   if (!is_int($id)) {
```

```
        throw new InvalidArgumentException("ID invalide");
    }
    // ...
}
```

5. **@deprecated** : Indique qu'un élément est déprécié et ne devrait plus être utilisé.
    - Syntaxe : `@deprecated version description`

Exemple :

```
/**
* @deprecated 1.2.0 Utilisez la méthode `newMethod()` à la
place.
*/
function oldMethod() {
    // ...
}
```

6. **@see** : Fait référence à un autre élément ou à une documentation externe.
    - Syntaxe : `@see référence`

Exemple :

```
/**
* @see https://example.com/docs
*/
function example() {
    // ...
}
```

7. **@since** : Indique depuis quelle version un élément est disponible.
    - Syntaxe : `@since version`

Exemple :

```php
/**
* @since 1.0.0
*/
function newFeature() {
   // ...
}
```

8. **@author** : Indique l'auteur de l'élément.
    - Syntaxe : `@author nom <email>`

Exemple :

```php
/**
* @author John Doe <john.doe@example.com>
*/
function myFunction() {
   // ...
}
```

Avantages des tags PHPDoc

1. **Documentation automatique** : Les outils comme phpDocumentor peuvent générer une documentation HTML ou PDF à partir des commentaires PHPDoc.
2. **Amélioration de l'autocomplétion** : Les IDE utilisent ces tags pour fournir des suggestions plus précises.

3. **Vérification de type** : Les tags comme `@param` et `@return` aident les IDE à détecter les erreurs de type.
4. **Maintenance facilitée** : La documentation intégrée au code permet de mieux comprendre le code, surtout dans les projets collaboratifs.

2. Documenter la classe Vin et ses attributs à l'aide de tags PHPDoc.

```php
class Vin
{

    //Atributs privés
    /**
     * Summary of nom
     * @var string le nom du vin
     */
    private $nom;
    /**
     * Summary of type
     * @var string le type du vin
     */
    private $type;
    /**
     * Summary of pays
     * @var string  le  pays du  vin
     */
    private $pays;
    /**
     * Summary of region
     * @var string la region du vin
     */
    private $region;
    /**
     * Summary of annee
     * @var int l'année de production du vin
     */
    private $annee;


    /**
     * Retourne  le  nom du  vin
     * Summary of getNom
     * @return string le  nom  du vin
     */
```

```php
public function getNom()
{
    return $this->nom;
}
/**
 * Définit  le  nom du vin
 * Summary of setNom
 * @param mixed $nom le  nouveau nom du  vin
 * @return void
 */
public function setNom($nom)
{
    $this->nom = $nom;
}
/**
 * Retourne le type du vin
 * Summary of getType
 * @return string le type du vin
 */
public function getType()
{
    return $this->type;
}
/**
 * Définit le type du vin
 * Summary of setType
 * @param mixed $type le nouveau type du vin
 * @return void
 */
public function setType($type)
{
    $this->type = $type;
}
/**
 * Retourne le pays du vin
 * Summary of getPays
 * @return string  le pays du vin
 */
public function getPays()
{
    return $this->pays;
}

/**
 * Définit  le pays du vin
```

```php
     * Summary of setPays
     * @param mixed $pays  le nouveau pays du vin
     * @return void
     */
    public function setPays($pays)
    {
        $this->pays = $pays;
    }
    /**
     * Retourne la région du vin
     * Summary of getRegion
     * @return string  la région du vin
     */
    public function getRegion()
    {
        return $this->region;
    }
    /**
     * Définit la région du vin
     * Summary of setRegion
     * @param mixed $region la nouvelle région du vin
     * @return void
     */
    public function setRegion($region)
    {
        $this->region = $region;
    }
    /**
     * Rettourne l'année de production du vin
     * Summary of getAnnee
     * @return int l'année de production du vin
     */
    public function getAnnee()
    {
        return $this->annee;
    }
    /**
     * Définit l'année de production du vin
     * Summary of setAnnee
     * @param mixed $annee la nouvelle année de production du vin
     * @return void
     */
    public function setAnnee($annee)
    {
        $this->annee = $annee;
```

```
    }
}
```

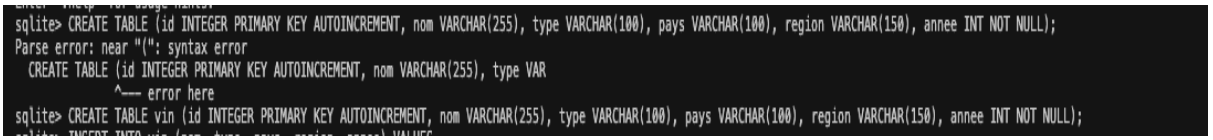
3. Que se passe-t-il lorsque l'on survole à la souris les noms des classes ou des méthodes d'un objet Vin dans le fichier index.html avec un éditeur moderne comme Visual Studio Code?

On obtient des informations sur la classe et les méthodes

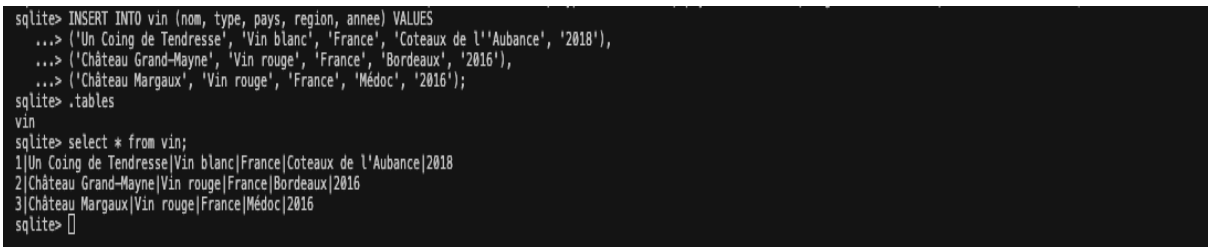
2- La base de données

1) Dans le répertoire data, créer une base de données nommée "wine_cellar.db".

2) Dans la base de données, créer une table nommée vin qui reprend la structure d'un vin de notre maquette avec les champs suivants: id (identifiant unique, auto-incrémenté), nom, type, pays, region, annee.



3) Populer la table vin avec trois vins contenant les mêmes informations que nos instances $uneBouteille, $uneAutreBouteille et $encoreEneAutreBouteille.



4) Créer le fichier suivant pour la gestion de la base de données.

Inclure la classe Debogage et BDD ainsi que leur namespace dans index.php et tester la méthode enteteDeTable() sur la table vin en faisant un simple echo dans la partie vue (View). Que se passe-t-il?

En testant la méthode enteteDeTable() sur la table vin, il retourne les entêtes (noms des colonnes de la table)

Nom	Type	Pays	Région	Année
Un Coing de Tendresse	Vin blanc	France	Coteaux de l'Aubance	2018
Château Grand-Mayne	Vin rouge	France	Bordeaux	2016
Château Margaux	Vin rouge	France	Médoc	2016

Liste des vins de la cave

5) Modifier la méthode de classe vins() pour qu'elle retourne un tableau d'objets Vin.

6) Modifier index.php pour utiliser les données retournées par la méthode vins().

$cave = BDD::vins();

7) Créer une méthode ajouterVin() dans la classe BDD avec pour paramètre d'entrée un objet Vin. Faire en sorte que la méthode insère le vin dans la table.

```
static public function ajouterVin(Vin $vin)
{
	$bdd = new SQLite3(BDD::$cheminDeLaBDD);
	$requete = "INSERT INTO vin (nom, type, pays, region, annee) VALUES (
	'" . $vin->getNom() . "',
	'" . $vin->getType() . "',
	'" . $vin->getPays() . "',
	'" . $vin->getRegion() . "',
	'" . $vin->getAnnee() . "'
	)";
	$bdd->exec($requete);
}
```

8) Que manque-t-il comme attribut dans la classe Vin afin de pouvoir faire le lien entre une instance et une ligne de la table vin de la base de données?

Il manque l'attribut `id` dans la classe `Vin` pour faire le lien entre une instance et une ligne de la table `vin`.

9) Ajouter cet attribut à la classe Vin ainsi que son assesseur et son mutateur.

```
private $id;

public function getId() { return $this->id; }
public function setId($id) { $this->id = $id; }
```

10) Créer une méthode supprimeVin() dans la classe BDD avec pour paramètre d'entrée un objet Vin. Faire en sorte que la méthode supprime le vin dans la table.

```
static public function supprimerVin(Vin $vin)
{
	$bdd = new SQLite3(BDD::$cheminDeLaBDD);
	$requete = "DELETE FROM vin WHERE id = " . $vin->getId();
	$bdd->exec($requete);
}
```

11) Quels sont les intérêts techniques d'encapsuler les accès à la base de données dans une classe dédiée telle que BDD?

- Centralisation : Toutes les requêtes SQL sont centralisées dans une seule classe, ce qui facilite la maintenance et les modifications.

- Sécurité : L'encapsulation permet de mieux contrôler les accès à la base de données et d'éviter les injections SQL.

- Réutilisabilité : Les méthodes de la classe `BDD` peuvent être réutilisées dans différentes parties de l'application.

- Abstraction : L'utilisation de la classe `BDD` permet de masquer les détails d'implémentation de la base de données, ce qui rend le code plus lisible et plus facile à comprendre.

Fannette SEKO - BTS SIO1 - 06 mars, 2025

Compte rendu du TP6 Cave à vins - Upload de fichier via un formulaire :

1- Fonctionnement du formulaire

1. Charger la page de test. Que contiennent les tableaux globaux $_POST, $_GET et $_FILES?

Les tableaux **$_POST, $_GET et $_FILES** sont vides (array(0)). Cela signifie qu'aucune donnée n'a été envoyée via un formulaire ou une requête HTTP au moment du chargement de la page.

2. Sélectionner un fichier quelconque sur votre disque dur (pdf, image, etc) via le bouton "Parcourir..." du formulaire puis cliquer sur le bouton "Envoyer". Que contiennent les tableaux globaux $_POST, $_GET et $_FILES?

Les tableaux globaux $_POST et $_FILES ne contiennent rien car la méthode utilisée pour envoyer le formulaire est GET (et non POST). Leurs tableaux sont vide mais $_GET contient le nom du fichier 1984.jpg

```
array(1) {
  ["fichier"]=>
  string(8) "1984.jpg"
}
```

3. Que contient l'url de votre navigateur?

Il contient le nom du fichier 1984.jpg

http://localhost:8080/index.php?fichier=1984.jpg

Cela signifie que le formulaire a été envoyé en utilisant la méthode **GET**, et le nom du fichier a été ajouté à l'URL en tant que paramètre de requête.

4. Les informations renvoyées par le formulaire en GET vous semblent-elles suffisantes? Pourquoi?

Non, les informations renvoyées par le formulaire en GET ne sont pas suffisantes pour gérer correctement l'envoi de fichiers. Voici pourquoi :

- **Limitation de la méthode GET** : La méthode GET ajoute les données du formulaire à l'URL, ce qui limite la taille des données pouvant être envoyées (la longueur maximale d'une URL est limitée par les navigateurs et les serveurs). Cela rend GET inadapté pour l'envoi de fichiers, qui peuvent être volumineux.
- **Absence de fichier dans $_FILES** : Avec la méthode GET, le fichier lui-même n'est pas envoyé au serveur. Seul le nom du fichier est transmis via l'URL, ce qui ne permet pas de récupérer le contenu du fichier.
- **Sécurité** : Transmettre des informations sensibles (comme des noms de fichiers) dans l'URL n'est pas sécurisé, car l'URL est visible dans l'historique du navigateur et peut être interceptée.

5. Modifier l'attribut method du formulaire en le passant à la valeur post. Recharger la page (attention à ne pas avoir de paramètre dans l'url). Refaire les actions de la question 2. Que contiennent maintenant les tableaux globaux $_POST, $_GET et $_FILES?

```
array(1) {
  ["fichier"]=>
  array(6) {
    ["name"]=>
    string(8) "1984.jpg"
    ["full_path"]=>
    string(8) "1984.jpg"
    ["type"]=>
    string(10) "image/jpeg"
    ["tmp_name"]=>
```

```
   string(79)
"/private/var/folders/mj/1gs4cwmd2m99v_5j7mlvs21c0000gp/T/phptkajqapln3m97P
881gJ"
   ["error"]=>
   int(0)
   ["size"]=>
   int(59949)
  }
}
```

- **$_POST** et **$_GET** sont vides car aucune donnée n'a été envoyée via ces méthodes.
- **$_FILES** contient des informations sur un fichier uploadé (1984.jpg), y compris son nom, son type, son emplacement temporaire sur le serveur, et sa taille.

2- Modification du contrôleur

1. Tester. Que se passe-t-il?

Warning: move_uploaded_file(./uploads/image.jpg): Failed to open stream: No such file or directory in /**Users**/**eleve**/**Desktop**/**php**/**upload**/**www**/**index.php** on line **21**

Warning: move_uploaded_file(): Unable to move "/private/var/folders/mj/1gs4cwmd2m99v_5j7mlvs21c0000gp/T/phpoas4nm2pl9861imXoBh" to "./uploads/image.jpg" in /**Users**/**eleve**/**Desktop**/**php**/**upload**/**www**/**index.php** on line **21**

2. Créer le répertoire “uploads” dans “www”. Tester à nouveau et documenter le code.

3- Vue conditionnelle

1. Modifier la vue pour afficher l'image avec le code suivant…

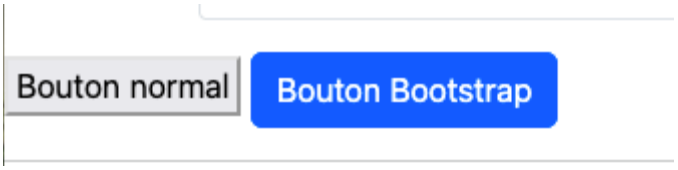
Nous souhaitons réaliser un affichage conditionnel de la vue. Faire en sorte que si $_FILES['fichier'] existe:
● Seule l'image est affichée.
● Le formulaire de téléversement n'est plus affiché.

Fannette SEKO - BTS SIO1 - 10 mars, 2025

Compte rendu du TP7 Cave à vins - Découverte de Bootstrap:

1- Un bouton

1. Quelle différence visuel observez-vous entre les deux boutons sur votre navigateur?

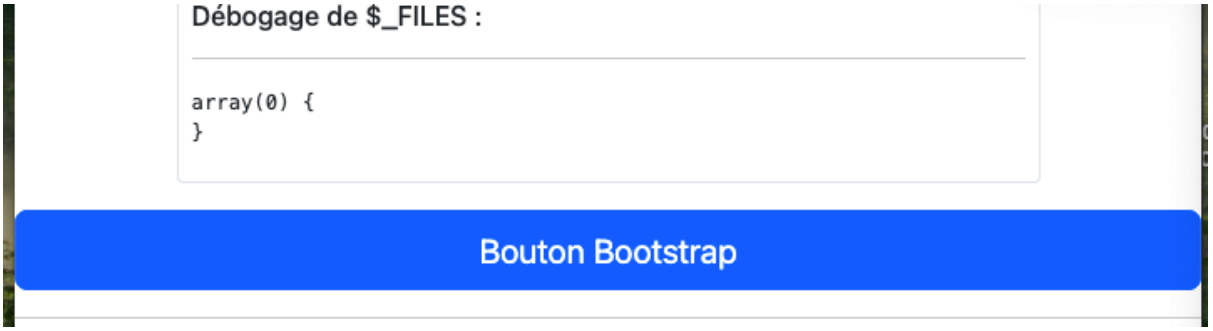


Le bouton bootstrap est customisé et plus beau que le bouton normal

2. Quel est le nom de l'attribut qui est utilisé pour customiser le bouton Bootstrap?

c'est l'attribut class="btn btn-primary"

3. Supprimer le bouton standard et modifier le bouton Bootstrap afin qu'il utilise maintenant toute la largeur de la page.
Ressource: https://getbootstrap.com/docs/5.3/components/buttons/



2- Un tableau

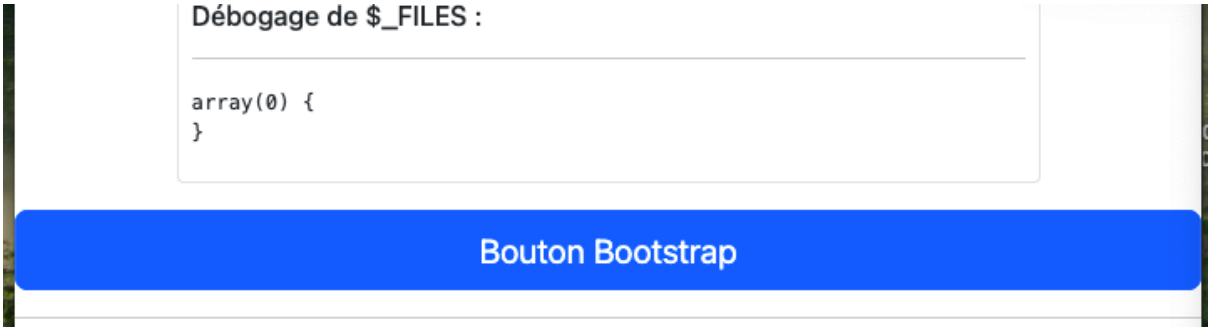
1. Proposer une stylisation à l'aide de Bootstrap.
Ressource: https://getbootstrap.com/docs/5.3/content/tables/

3- Une liste

1. Proposer une stylisation à l'aide de Bootstrap.

Ressource: https://getbootstrap.com/docs/5.3/components/list-group/

Nom	Type	Pays	Région	Année
Un Coing de Tendresse	Vin blanc	France	Coteaux de l'Aubance	2018
Château Grand-Mayne	Vin rouge	France	Bordeaux	2016
Château Margaux	Vin rouge	France	Médoc	2016



4- Un message d'alerte

1. Proposer une stylisation à l'aide de Bootstrap afin d'afficher une alerte en orange.
Ressource: https://getbootstrap.com/docs/5.3/components/alerts/

Attention! Rupture de stock sur ce millésime!

```php
<?php
/*******************************************************************************
*
* C O N T R O L L E R
*/
require('../model/Debogage.php');

use Model\Debogage;
// Ici je code le contenu du contrôleur (Controller).
?>
<!--------------------------------------------------------------------------
-- V I E W
-->
<!doctype html>
<html lang="en">

<head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <title>Modèle de base d'une page Bootstrap</title>
   <!-- Inclusion de la feuille de style CSS distante de Bootstrap -->
   <link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.cs
s" rel="stylesheet"

integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
crossorigin="anonymous">
</head>

<body>
   <!-- Ici je code le contenu de la vue (View). -->
   <div class="container">
       <div class="starter-template">
           <h1>Ceci est notre modèle de base</h1>
           <p class="lead">Utilisez ce document comme un moyen de démarrer
               rapidement tout nouveau projet.</p>
       </div>
       <?php
       Debogage::afficher($_POST, '$_POST');
       Debogage::afficher($_GET, '$_GET');
       Debogage::afficher($_FILES, '$_FILES');
?>
```

```
</div>

<table class="table table-dark table-striped">
    <thead>
        <tr>
            <th scope="col">Nom</th>
            <th scope="col">Type</th>
            <th scope="col">Pays</th>
            <th scope="col">Région</th>
            <th scope="col">Année</th>
        </tr>
    </thead>
    <tr>
        <th scope="row">
            Un Coing de Tendresse </th>
        <td>
            Vin blanc </td>
        <td>
            France </td>
        <td>
            Coteaux de l'Aubance </td>
        <td>
            2018 </td>
    </tr>
    <tr>
        <th scope="row">
            Château Grand-Mayne </th>
        <td>
            Vin rouge </td>
        <td>
            France </td>
        <td>
            Bordeaux </td>
        <td>
            2016 </td>
    </tr>
    <tr>
        <th scope="row">
            Château Margaux </th>
        <td>
            Vin rouge </td>
        <td>
            France </td>
        <td>
            Médoc </td>
```

```
            <td>
                2016 </td>
        </tr>
    </table>
    <hr>

    <h2>Les régions de nos producteurs</h2>
    <ul class="list-group">
        <li class="list-group-item">Coteaux de l'Aubance</li>
        <li class="list-group-item">Bordeaux</li>
        <li class="list-group-item">Médoc</li>
    </ul>
    <hr>

    <div class="alert alert-warning" role="alert">
        Attention! Rupture de stock sur ce millésime!
    </div>
    <hr>
    <div class="d-grid gap-2">
        <button class="btn btn-primary btn-lg" type="submit">Bouton
Bootstrap</button>
    </div>
    <hr>
    <!-- Inclusion de la bibliothèque distante Javascript Bootstrap et de Popper-->
    <scrip6

src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle
.min.js"

integrity="sha384-w76AqPfDkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTkF7CXv
N"
        crossorigin="anonymous"></scrip6>
</body>
</html>
```