COURS 7 - Adopter un style de programmation clair

Lorsque votre site prend de l'importance, le code devient vite illisible et incompréhensible si vous ne pensez pas à l'organiser . Mais comment organiser son code de manière à être le plus clair possible ? Il existe de nombreuses solutions, chacune ayant ses avantages et ses inconvénients. L'une des solutions fréquemment retenue est de dissocier la vue c'est à dire le code xHTML visible, du traitement des données c'est à dire de l'accès aux bases de données et le contrôle des résultats. On parle alors de modèles MVC : Modèle Vue Contrôleurs .

Présentation succincte du modèle MVC : Modèle Vue Contrôleur

Au lieu de tout mettre dans un seul fichier et ainsi de rendre votre code peu clair, le modèle MVC divisera le tout en trois fichiers.

1- Le modèle
Ce fichier ne contiendra que des fonctions. Le modèle a pour but de gérer l'organisation des données. **Vous ne pourrez effectuer des requêtes SQL que dans ces fonctions !** Le contrôleur et la vue ne devront contenir aucun appel à la base de données.

2 - La vue
La vue contient le code xHTML. C'est la seule partie qui doit en contenir. Les requêtes SQL étant effectuées dans le modèle et le contrôle des données dans le contrôleur, le code PHP devra se contenter d'afficher les résultats.

3 – Le contrôleur
C'est le fichier qui contient toute la logique du code. Vous commencerez généralement par y inclure votre modèle . Vous procéderez à des vérifications d'ordre général (comme les autorisations), puis vous pourrez appeler des fonctions avant d'envoyer les résultats à la vue.

Le modèle MVC sera abordé par la suite, notamment le rôle des modèles et des contrôleurs, mais nous pouvons déjà organiser notre code pour éviter à la vue d'avoir à gérer toutes les requêtes SQL, et ainsi commencer à construire nos contrôleurs.

Exemple d'organisation de votre code

A partir de la table salariés :

idsalaries	nom	prenom	date_naissance	date_embauche	salaire	service
1	Cuset	Jean	1966-06-02	1995-05-15	2400	informatique
2	Dian	Charles	1972-08-30	2007-04-01	1800	commercial
3	Paco	Alain	1958-04-25	2002-06-06	2100	commercial
4	Perez	Amanda	1972-01-11	1999-03-11	1700	comptable
5	Jeanu	Thierry	1970-07-11	1998-01-01	1600	comptable
6	Taque	Stephane	1965-04-08	2008-02-01	2000	informatique
7	Devos	Yahn	1965-07-08	2009-02-01	2400	informatique
8	Tettu	Sylvain	1975-07-11	2006-02-01	2000	comptable

On souhaite obtenir la vue suivante :

id	nom	prenom	date-naissance	date-embauche	salaire	service
1	Cuset	Jean	1966-06-02	1995-05-15	2400	informatique
2	Dian	Charles	1972-08-30	2007-04-01	1800	commercial
3	Paco	Alain	1958-04-25	2002-06-06	2100	commercial
4	Perez	Amanda	1972-01-11	1999-03-11	1700	comptable
5	Jeanu	Thierry	1970-07-11	1998-01-01	1600	comptable
6	Taque	Stephane	1965-04-08	2008-02-01	2000	informatique
7	Devos	Yahn	1965-07-08	2009-02-01	2400	informatique
8	Tettu	Sylvain	1975-07-11	2006-02-01	2000	comptable

nombre de salaries : 20

salaire moyen : 2355

Cette vue peut être obtenue d'une manière classique avec le code suivant :

```
<?php

  $conn = mysqli_connect('localhost','root','sio','salaries');
  if ($conn->connect_error) {
      die("Connection failed: " . $conn->connect_error); }

  $req1 = "SELECT * FROM salaries ;";
  $result1 = mysqli_query($conn,$req1) or die (mysqli_error($conn));
?>
<table border=2>
  <th>id</th><th>nom</th><th>prenom</th><th>date-naissance</th>
  <th>date-embauche</th><th>salaire</th><th>service</th>
<?php
  echo "<tr>";
  while ( $ligne=mysqli_fetch_array($result1)){
     echo "<tr>";
        echo "<td>".$ligne['idsalaries']."</td>";
        echo "<td>".$ligne['nom']."</td>";
        echo "<td>".$ligne['prenom']."</td>";
        echo "<td>".$ligne['date_naissance']."</td>";
        echo "<td>".$ligne['date_embauche']."</td>";
        echo "<td>".$ligne['salaire']."</td>";
        echo "<td>".$ligne['service']."</td>";
     echo "</tr>";
  }
?>
</table>
<?php
  $req2 = "SELECT count(*) FROM salaries ;";
  $result2 = mysqli_query($conn,$req2) or die (mysqli_error($conn));
  $ligne=mysqli_fetch_array($result2);
  echo "<p>nombre de salariés : ".$ligne[0]."</p>";

  $req3 = "SELECT avg(salaire) FROM salaries ;";
  $result3 = mysqli_query($conn,$req3) or die (mysqli_error($conn));
  $ligne=mysqli_fetch_array($result3);
  echo "<p>salaire moyen : ".round($ligne[0])."</p>";

?>
```

Ce code fonctionne mais il est difficile à maintenir et il peut vite devenir incompréhensible, si on souhaite par exemple afficher en plus du nombre de salariés, le salaire minimum, maximum etc.. Il faut donc essayer de le réorganiser .

Réorganisation du code :

Dans un premier temps, la vue n'a pas à gérer l'accès à la base de données. Si votre site comporte plusieurs vues, vous allez devoir intervenir dans chaque vue en cas de modification des paramètres de connexion. Il est préférable de créer un fichier contenant les paramètres de connexion que l'on va inclure dans chaque vue avec un **include** ou mieux avec un **require_once :**

```
require_once('connexion.php');
```

connexion.php

```
<?php

   $conn = mysqli_connect('localhost','root','sio','salaries');
   if ($conn->connect_error) {
     die("Connection failed: " . $conn->connect_error); }

?>
```

La vue contient le code xHTML. C'est la seule partie qui doit en contenir. Les requêtes SQL doivent être effectuées ailleurs, la vue doit se contenter d'afficher les résultats.
Il faut don créer un nouveau fichier qui va contenir toutes les fonctions qui se chargeront des requêtes SQL.
Par exemple, si on souhaite afficher le nombre de salariés, on va définir une fonction que le nommera getNbSalaries qui retournera le résultat de la requête suivante :

```
SELECT count(*) FROM salaries
```

fonctions.php

```
<?php


  function getNbSalaries($conn){
     $req = "SELECT count(*) FROM salaries ;";
     $result = mysqli_query($conn,$req) or die (mysqli_error($conn));
     $ligne=mysqli_fetch_array($result);
     return $ligne[0];
 }
?>
```

Et dans notre vue, on va placer avant tout code HTML, les données que l'on souhaite afficher :

```
<?php
  require_once("connexion.php");
  require_once("fonctions.php");


  $nbSalaries = getNbSalaries($conn);
?>

<p>Nombre de salariés : <?php echo $nbSalaries ; ?> </p>
```

Elle permet d'éviter les problèmes de concaténation et logiquement c'est au code HTML d'inclure du PHP et non l'inverse :

```
echo "<p>nombre de salariés : ".$nbSalaries."</p>" ;
```

Lorsque la requête SQL retourne plusieurs lignes, la fonction devra retourner un tableau contenant toutes les lignes. Par exemple si l'on souhaite afficher la liste et les attributs de tous les salariés

```
function getNbSalaries($conn){
     $req = "SELECT count(*) FROM salaries ;";
     $result = mysqli_query($conn,$req) or die (mysqli_error($conn));
     $ligne=mysqli_fetch_array($result);
     return $ligne[0];
 }

function getAllSalaries($conn){

    $listeSalaries = array();
    $req = 'SELECT * FROM salaries';

    $result = mysqli_query($conn,$req) or die (mysqli_error($conn));
    while($ligne = mysqli_fetch_assoc($result)) {
       $listeSalaries[]=$ligne;
    }
    return $listeSalaries;
  }

…...
```

- On initialise un tableau : `$listeSalaries = array() ;`
- On parcourt chaque ligne du résultat de la requête :
  `while($ligne = mysqli_fetch_assoc($result))`
- On ajoute au tableau chaque ligne : `$listeSalaries[]=$ligne ;`

La variable `$listeSalaries` est donc un tableau à deux dimensions, que l'on parcourt ainsi : (voir cours sur les tableaux associatifs)

```
<?php foreach ($listeSalaries as $leSalarie): ?>
  <tr>
    <td><?php echo $leSalarie['idsalaries']; ?></td>
    <td><?php echo $leSalarie['nom']; ?></td>
    <td><?php echo $leSalarie['prenom']; ?></td>
    <td><?php echo $leSalarie['date_naissance']; ?></td>
    <td><?php echo $leSalarie['date_embauche']; ?></td>
    <td><?php echo $leSalarie['salaire']; ?></td>
    <td><?php echo $leSalarie['service']; ?></td>
  </tr>
<?php endforeach; ?>
```

Dans notre vue :

```
 <?php
  require_once("connexion.php");
  require_once("fonctions.php");


  $nbSalaries = getNbSalaries();
  $listeSalaries = getAllSalaries() ;
?>

<table border=2>
  <th>id</th>
  <th>nom</th>
  <th>prenom</th>
  <th>date-naissance</th>
  <th>date-embauche</th>
  <th>salaire</th>
  <th>service</th>

  <?php foreach ($listeSalaries as $leSalarie): ?>
  <tr>
    <td><?php echo $leSalarie['idsalaries']; ?></td>
    <td><?php echo $leSalarie['nom']; ?></td>
    <td><?php echo $leSalarie['prenom']; ?></td>
    <td><?php echo $leSalarie['date_naissance']; ?></td>
    <td><?php echo $leSalarie['date_embauche']; ?></td>
    <td><?php echo $leSalarie['salaire']; ?></td>
    <td><?php echo $leSalarie['service']; ?></td>
  </tr>
<?php endforeach; ?>
 <?php endforeach; ?>

</table>
    <p>Nombre de salariés : <?php echo $nbSalaries ; ?> </p>
```

Notez que pour le ***foreach*** , il n'y a pas d'accolades :

- l'accolade de début est remplacée par **:**
- l'accolade de fin est remplacée par un ***endforeach ;***.
- Toutes les structures de contrôles listées ci-dessus ont des syntaxes de fermetures similaires : *endif*, *endfor*, *endforeach* et *endwhile*

Travail à faire :

1 – Dans votre répertoire **public_html** , créer un dossier **mvc2016**.

2 – Télécharger l'archive **mvc2016.zip** et l'extraire dans votre dossier **mvc2016**. Cette archive contient plusieurs fichiers notamment le script listeSalaries.php et salaries.sql.

3 – Avec phpMyAdmin, **créer une base salaries** et exécuter le script **salaries.sql**. Vous pouvez ensuite tester le script **listeSalaries.php**.

4 – Modifier le script **listeSalaries.php** pour séparer la vue du traitement des données. Vous allez devoir créer un fichier **connexion.php** et un ficher **fonctions.php** qui contiendra toutes les fonctions utiles à la vue.

4 – Écrire les fonctions qui vont vous permettre d'obtenir la vue suivante :

id	nom	prenom	date-naissance	date-embauche	salaire	service
1	Cuset	Jean	1966-06-02	1995-05-15	2400	informatique
2	Dian	Charles	1972-08-30	2007-04-01	1800	commercial
3	Paco	Alain	1958-04-25	2002-06-06	2100	commercial
4	Perez	Amanda	1972-01-11	1999-03-11	1700	comptable
5	Jeanu	Thierry	1970-07-11	1998-01-01	1600	comptable
6	Taque	Stephane	1965-04-08	2008-02-01	2000	informatique
7	Devos	Yahn	1965-07-08	2009-02-01	2400	informatique
8	Tettu	Sylvain	1975-07-11	2006-02-01	2000	comptable
9	Triet	Jacques	1968-11-08	2005-10-01	3000	informatique
10	Jolivet	Olivier	1973-02-23	2000-11-06	1800	commercial

Nombre de salariés : 20

Salaire moyen : 2355

Salaire minimum : 1600

Salaire maximum : 5000

nombre de salariés par service :

commercial 6

comptable 5

informatique 9

5 – Ajouter des liens pour supprimer et modifier un enregistrement :

id	nom	prenom	date-naissance	date-embauche	salaire	service	delete	update
1	Cuset	Jean	1966-06-02	1995-05-15	2400	informatique	delete	update
2	Dian	Charles	1972-08-30	2007-04-01	1800	commercial	delete	update
3	Paco	Alain	1958-04-25	2002-06-06	2100	commercial	delete	update
4	Perez	Amanda	1972-01-11	1999-03-11	1700	comptable	delete	update
5	Jeanu	Thierry	1970-07-11	1998-01-01	1600	comptable	delete	update

Ajouter salariés

6 – Sur le modèle de la gestion des utilisateurs, vous allez créer les formulaires qui vont vous permettre d'ajouter ou de mettre à jour les données d'un salarié.

7 – Pour supprimer un salarié vous allez faire apparaître une fenêtre de confirmation en javascript (voir correction gestion des utilisateurs) .

8 – Pour la gestion des dates d'embauche et naissance utiliser datepicker et faites tous les contrôles que vous jugerez nécessaires, par exemple on ne doit pas pouvoir embaucher un salarié de moins de 18 ans, la date de naissance doit être inférieure à la date d'embauche etc .

Pour les plus avancés :

Améliorer la présentation (éventuellement utiliser bootstrap pour la présentation).