Programmation Web Dynamique 3

Cours 1 – Introduction à la P.O.O.

Qu’est-ce que la POO?

- Pas seulement une autre syntaxe de programmation, mais bien un autre paradigme de programmation.
  - Une nouvelle façon de penser à un problème.
- Changement de focus :
  - Programmation procédurale -> quelle suite d’actions, de verbes mène à mon résultat?
  - Programmation orientée objet -> de quels éléments a besoin mon application pour arriver à ses fins?
- Ce n’est pas une *meilleure* façon de programmer, c’est une façon *différente* de programmer.

Concept d'abstraction

- Un premier concept important pour comprendre la programmation orientée-objet est le concept d'abstraction.
- On parle de niveau d'abstraction lorsque l'on veut définir un niveau sous lequel regarder un problème.
- Par exemple, lorsque l'on veut faire réchauffer de la nourriture, on ne soucie généralement pas du fonctionnement interne du four micro-ondes; on sait seulement que l'on doit démarrer le four, choisir une puissance et un temps, et mettre les ingrédients à l'intérieur.
- Ceux qui ont construit le micro-ondes, eux, regardent le problème sous un tout autre niveau d'abstraction : le fonctionnement interne du four.
- Il serait par contre complètement fou de demander à tous les cuisiniers de comprendre le fonctionnement interne d'un four micro-ondes. Lorsque l'on veut y faire réchauffer quelque chose, on le considère comme une boîte noire avec laquelle on interagit par l'interface. C'est pour prendre avantage de ce principe d'abstraction (on fait *abstraction* du fonctionnement interne du four) qu'a été développée la programmation orientée objet.

Concept de classification / modularité

- La programmation orientée objet est aussi basée sur le concept de classification.
- On a déjà vu, en programmation par fonctions, qu'il était souvent plus facile de séparer un problème en petites fonctions plutôt que de l'aborder dans un seul bloc.
- Il est souvent encore plus facile de conceptualiser un problème complexe en le séparant en plusieurs éléments qui interagissent entre eux, surtout si ces éléments réapparaissent dans plusieurs problèmes différents.
- De plus, lorsque certains éléments de ce problème changeront (les probabilités sont très fortes que cela arrive), il est plus compliqué de modifier la solution à ce problème si toute la solution est dans une même fonction. Si les éléments du problème sont séparés, il ne suffira qu'à redéfinir l'élément changé et ses relations avec les autres éléments, ce qui est souvent plus facile et plus évident à identifier.

Classes et objets

- Jusqu'à maintenant, vous avez vu comment regrouper des opérations ensemble sous forme de fonctions.
  - Ces fonctions, pour être utiles, devaient recevoir des données en paramètres. Il y avait donc une grande séparation entre données et opérations.
  - Pourquoi une si grande séparation?
    - Dans plusieurs cas, les fonctions pourraient être reliées directement aux données pour simplifier les choses.
    - Ex : toUpperCase est maintenant une fonction rattachée à l'objet String!
- Pour prendre avantage des concepts de classification et d'abstraction et de rattacher des fonctions avec des données, il faut être capable de créer des structures contenant à la fois des variables et des fonctions.
- C'est dans ce but que la programmation orientée objet amène le concept d'objet.

Classes et objets (suite)

- Qu'est-ce qu'un objet?
  - Un objet est une entité qui rassemble des données et des opérations pour accéder et modifier ces données.
  - De l'information et des fonctionnalités qui ont trait à cette information.
- Qu'est-ce qu'une classe?
  - Une classe est un modèle, un moule à partir duquel on peut créer des objets.
  - À partir d'une classe, il est possible de créer une infinité d'objets qui possèderont les mêmes caractéristiques avec des valeurs différentes. Tous ces objets fourniront aussi les mêmes opérations pour transformer leurs caractéristiques.
  - Vous pouvez faire un rapprochement avec la notion d'entité et d'occurrence.
  - **On dit alors qu'une classe est la description des objets, alors qu'un objet est une *instance* de cette classe.**

La notion d'objet

- Un objet aura donc des caractéristiques et des opérations (ou capacités).
- On nommera ces caractéristiques des *attributs*.
- Les opérations attachées à un objet seront des fonctions (telles que vues en programmation par procédure, comme dans les cours précédents) que nous nommerons des *méthodes*.
- Ces méthodes pourront accéder directement aux caractéristiques de l'objet. Théoriquement, il est toujours préférable de passer par les méthodes de l'objet pour accéder à ses caractéristiques.
  - Ceci découle du concept de l'encapsulation. Nous en parlerons plus tard.

Attributs

- Une classe possède donc des attributs (qu'on appelle aussi membres). Prenons par exemple une classe qui se nommerait Voiture, et qui servirait à construire, à définir des objets Voiture pour une course.
- Cette classe pourrait avoir des attributs comme :
  - Marque (la marque de la voiture)
  - Vitesse maximale
  - Consommation (en milles au galon)
  - Vitesse courante
  - Position
  - Etc.
- Ces attributs sont différents pour chaque instance de voiture créée. On les appelle donc les attributs d'instance.

Attributs (suite)

- Comme dans le cas des entités dans les bases de données, on définit toujours les attributs d'une classe en fonction du problème à résoudre.
  - Dans le cas d'une course présentée à la télévision, la vitesse courante d'une voiture peut être importante.
  - Dans le cas d'un inventaire de voitures à vendre pour un système d'encan en ligne, la vitesse courante de la voiture n'a aucune signification, aucune importance. Elle ne devrait donc pas être dans la liste d'attributs.
  - L'année de fabrication de la voiture est très importante dans ce même inventaire. Dans le cas de la course de F1? Probablement pas.

Méthodes

- En programmation orientée-objet, les fonctions membres d'une classe sont appelées des méthodes, et on les utilise généralement pour accéder, modifier ou utiliser des attributs d'une classe.
- Encore une fois, il est très important de définir les méthodes en fonction du problème à résoudre
- Dans le cas de la course :
  - Une méthode setVitesseMPH, par exemple, qui prendrait une vitesse en milles à l'heure et la transformerait en kilomètres/heure pour ensuite l'affecter à l'attribut Vitesse de notre objet Voiture.
  - Une méthode setConducteur, qui associerait un objet Conducteur à l'objet Voiture en question!
    - Nous verrons plus tard qu'il est possible pour un objet de contenir une référence à un autre objet!
- Dans le cas de l'encan :
  - Une fonction changeStatus pour faire passer de À Vendre à Vendu la voiture lorsqu'elle a été achetée par un usager de notre site Web.

Classes et objets en PHP

- Pour déclarer une nouvelle classe en PHP, il suffit d'utiliser le mot-clé class, suivi du nom de la classe que vous voulez déclarer. Ce nom ne doit pas commencer par un chiffre.
- Le code qui définit la classe doit ensuite être placé entre accolades.
- Ex :

  ```
  class Voiture
  {
  }
  ```

Utiliser des objets dans PHP

- Pour créer une instance d'une classe en PHP, on utilise le mot clé new, comme dans l'exemple suivant :
  - $maVoiture = new Voiture();
  - Nous verrons plus tard ce que ce mot clé new fait réellement.
- La syntaxe d'utilisation des objets en PHP est plutôt particulière. En effet, pour accéder à des méthodes ou des attributs d'un objet en PHP, on utilise l'opérateur « -> », qui est en fait un trait d'union suivi d'un opérateur « plus grand que ». Cet opérateur est l'équivalent du point « . » utilisé dans la plupart des langages de programmation.
  - echo $maVoiture->couleur; //utilisation d'un attribut
  - $maVoiture->setVitesseMPH(50); //appel d'une méthode

Méthodes dans PHP

- Pour créer une méthode dans PHP, on utilise le mot clé function, précédé du modificateur d'accès, et suivi du nom de la fonction, suivi des parenthèses qui vont contenir les arguments envoyés à la méthode. Le code qui doit être exécuté par la méthode est ensuite placé entre accolades.
- Ex : public function maMethode($argument1, $argument2) {

```
        // ...
}
```

- Il est possible pour une méthode, comme une fonction normale, de retourner une valeur à la fin de son exécution, avec le mot clé return.
- Ex :

```
public function afficheDescription()
{
    echo 'Ceci est la description de l'objet.';
}
```

- On appelle la méthode d'un objet que l'on a créé de la même manière que vu précédemment.
  - Ex : $monObjet->afficheDescription();
- De la même façon, les méthodes peuvent retourner quelque chose avec return.

```
public function getDescription()
{
    return 'Ceci est la description de l'objet.';
}
```

Attributs dans PHP

- On déclarer les attributs dans PHP en utilisant leur modificateur d'accès (public, private, protected) suivi du nom de la variable. Ceci est nouveau dans PHP 5 puisque les modificateurs d'accès n'existaient pas en PHP 4, et l'on devait utiliser le mot-clé var avant chaque déclaration d'attributs. Pour l'instant, nous utiliserons le modificateur d'accès public. Nous verrons au prochain cours comment l'encapsulation vient dicter l'utilisation des autres modificateurs d'accès.
- Ex :

```
class Voiture
{
  public $marque;
  public $prix;
  public $fabricant;
  public $annee;
}
```

- On utilise ensuite l'opérateur -> pour accéder aux attributs de cet objet.
  - Ex : $maVoiture->marque= « Caravan »;

Lier les méthodes et les attributs : Le mot clé $this

- Évidemment, le but de la programmation orientée est de lier les méthodes aux attributs, pour que la solution au problème soit dans un seul module.
  - Pour faire avancer une instance, un objet de la classe voiture avec la méthode avance(), par exemple, on a besoin de l'attribut vitesse de cette *même voiture*.
  - Comment faire pour accéder, à l'intérieur de la méthode d'une classe, à l'attribut de l'instance courante?
- Avec le mot clé $this!
- Voir exemple