IFT1147
Programmation Serveur Web avec PHP

La POO en PHP

Plan

- Lecture du chapitre 20.
- Programmation par objets
  - Introduction à la POO
  - Objets et classes
  - Propriétés
  - Méthodes
  - Private
  - Héritage
  - Polymorphisme et héritage multiple

Programmation par objets

POO en PHP ce n'est pas encore ce que ça pourrait être, mais …

PHP 4 et PHP 5

- PHP 4 supportait déjà quelques rudiments de la programmation par objets:
  - Constructeurs
  - Héritage
- PHP 5 supporte
  - `private`, `public`
  - Destructeurs
  - *Method overloading*

Programmation procédurale vs POO

- Procédurale
  - Définitions de données
    - Définition de données pour les employés
    - Définition de données pour l'inventaire
  - Instructions de manipulation des données
    - Code de manipulation des données pour les employés
    - Code de manipulation des données pour l'inventaire
- POO
  - Le concept général de la POO permet de regrouper les données et les instructions de manipulation en unités conceptuelles

POO vs procédurale (suite)

- POO (suite)
  - Les employés
    - Définition de données
    - Code de manipulation des données
  - L'inventaire
    - Définition de données
    - Code de manipulation des données
  - En quoi cela est-il mieux que la programmation procédurale?
    - En regroupant les données et les programmes on s'assure d'avoir des programmes mieux adaptés (??)
    - On cache les détails d'implémentation interne des données et en même temps on s'assure d'une meilleure consistence.
    - On force le programmeur qui veut réutiliser le programme à utiliser les bons mécanismes de modification des données (mutateur/accesseur avec validation)
    - Plus facile de réutiliser les instructions et les données.
    - Etc...

POO

- Définition de nos types de données
- Terminologies
- Encapsulation
- Héritage
- Constructeurs et desctructeurs
- Définition en PHP

Définition de nos types de données

- Le concept principal consiste à définir nos propres types de données.
- Intégrer dans ces données des mécanismes pour les valider, les initialiser, les modifier et les consulter.
- Permettre la création d'objets utilisant ces types. Le processus de création de ces objets s'appelle l'instanciation.

Terminologies

- Classe: type de données défini par le programmeur.
- Objet: instance de la classe. On défini la classe une seule fois et on l'utilise plusieurs fois pour créer des objets.
- Donnée membre: aussi appelée attribut ou propriété et représente une valeur faisant partie des données de la classe.
- Fonction membre: aussi appelée méthode et représente une fonction permettant d'agir sur les données de la classe.
- Classe parent: c'est la classe utilisée pour dériver une nouvelle classe. On l'appelle aussi classe de base.
- Classe enfant: c'est la nouvelle classe dérivée à partir d'une classe parent.

Encapsulation

- L'encapsulation consiste à cacher les détails internes utiles au fonctionnement et à l'implémentation du type de données.
- Au niveau programmation on pourra décider de rendre disponible seulement certaines portions du type de données au monde extérieur et cacher le reste (les détails d'implémentation). De cette manière le concepteur qui décide de modifier la portion cachée peut le faire sans risque d'affecter les programmeurs l'utilisant car il est certain qu'ils ne voient pas cette portion.
- Il est également possible de forcer la modification des données en passant par un mutateur (fonction de modification) qui permettra de valider le changement avant qu'il soit effectuer.
- De la même manière il est possible de forcer la lecture en passant par un accesseur (fonction de lecture).

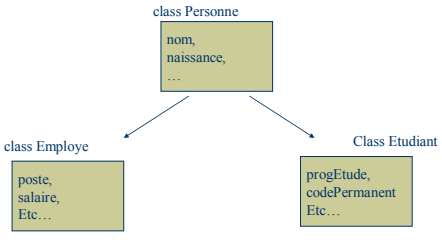
Héritage

- Le concept d'héritage est le plus important de la POO. Il permet la réutilisation de type de base défini par l'utilisateur tout en permettant de spécialiser le type.

L'héritage (suite)

- C'est la base des notions de réutilisation de composants logiciels.
- L'idée est de pouvoir définir (dériver) une nouvelle classe en se servant d'une classe existante (base).
- La classe dérivée hérite des membres de la classe de base tout en lui ajoutant de nouveaux.
- Il s'agit d'écrire de nouvelles classes plus spécifiques en se servant des définitions de base.
- Par exemple, nous pouvons dériver une nouvelle classe Employe en se servant de la classe de base Personne.
- Permet de définir la relation "est un". Par exemple, un employé est une personne ou encore un étudiant est une personne. Un cercle est une forme géo…

Généralités (héritage)



Constructeurs/destructeurs

- Un constructeur servira à initialiser le contenu d'un objet et même dans certains cas à allouer l'espace nécessaire aux données membres de l'objet.
- Le constructeur est appelé automatiquement lors de la création de l'objet, il est alors impossible de l'oublier.
- Un destructeur permet de libérer les ressources utilisées par un objet.

Programmation par objets en PHP

- La classe est l'élément de base de la programmation par objets. Elle est le modèle à partir duquel des instances peuvent être créées.
- Une classe est un conteneur pour des propriétés (variables) et des méthodes (fonctions). On dit propriétés et méthodes membres.

Exemple

```
class NomDeLaClasse {
     var $propriete1;
     var $propriete2;
     function __construct() {
          //constructeur
     }
     function fonction() {
        // méthode
    }
}
```

Note: avant PHP5 le constructeur utilisait le même nom que la classe. Mais on n'utilise plus cette méthode

Instanciation

- Afin d'instancier un objet, il faut utiliser le mot clé `new`

```
$maVar = new NomDeLaClasse();
```

- Afin de faire référence à une propriété de l'objet, il faut utiliser `->`

```
$maVar->prop = 23;
```


La classe Employe

```
class Employe {
  var $nom;
  var $salaire;
  function __construct($n, $s) {
    $this->nom = $n;
    $this->salaire = $s;
  }
  function toHTML() {
    return "<strong>Le nom:</strong><em>$this->nom</em>".
       "<strong>sal:</strong><em>$this->salaire</em>";
  }
}
$bob = new Employe( "Bob", 45000 );
echo $bob->toHTML();
```


Utilisation de $this

- Afin de faire référence à une propriété à l'intérieur de la classe, il faut utiliser `$this`

```
$this->age = 23;
```


Constructeur et destructeur en PHP

- Le constructeur est automatiquement appelé lors de l'instanciation d'un objet de la classe.
- Le constructeur est une méthode dont le nom est __construct( ) (notez le double souligné).
- En PHP4 le constructeur portait le même nom que la classe.
- Seulement le dernier constructeur défini sera utilisé.
- Le desctructeur est automatiquement appelé lorsqu'on détruit un objet. Le nom du destructeur est __destruct( ). Pour détruire un objet la fonction unset(objet) sera utilisée.

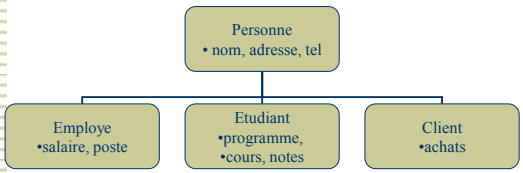
Le mot clé private

- Dans l'exemple précédent avec $bob utilisant la classe Employe les propriétés $nom et $salaire sont publiques (par défaut). Les propriétés sont accessibles aux utilisateurs de la classe.
- Il est alors possible de faire:
  `$bob->salaire = -12000;`
- En POO ce comportement n'est pas désirable, car les propriétés ne devraient être accessibles que par des méthodes de la classe. Cela permet un minimum de cohérence des données ainsi la méthode qui sert à modifier le salaire peut valider la nouvelle valeur avant de faire ces modifications.
- `private $nom;`
  `private $salaire;`
- Il est aussi possible de rendre privé des méthodes, qui ne pourront être utilisées qu'à l'intérieur de la classe.

L'héritage plus spécifiquement

- L'héritage consiste à réutiliser une classe de base (plus générale) pour en faire une version plus spécialisée.
- Par exemple, nous pourrions faire une classe Personne qui contient les propriétés nom et adresse pour ensuite dériver une nouvelle classe Employe à partir de Personne. Cette nouvelle classe permettrait d'ajouter la propriété salaire et les méthodes qui s'y rattache. Dans la même ordre d'idée nous pourrions dériver une autre classe Etudiant à partir de la classe Personne pour lui ajouter son dossier étudiant et les méthodes permettant de le manipuler.

L'héritage



Les classes dérivées héritent des propriétés nom, adresse et tel en plus de pouvoir utiliser les méthodes définies dans la classe de base

La classe Personne

```
class Personne {
  private $nom;
  private $adresse;
  private $tel
  function __construct($n, $a, $t) {
    $this->nom = $n;
    $this->adresse = $a;
    $this->tel = $t;
  }
  function toHTML() {
     // …
  }
}
$toto = new Personne( "toto", "123 rue Laval", "514…");
echo $toto->toHTML();
```

La classe dérivée Employe

```
class Employe extends Personne {
  private $salaire;
  private $poste;
  function __construct($n, $a, $t, $s, $p) {
    parent::__construct($n, $a, $t);
    $this->salaire = $s;
    $this->poste = $p;
  }
  function toHTML() {
     // …
  }
}
$bob = new Employe("toto","123…","514…",45000,"prog");
echo $bob->toHTML();
```


Méthodes statiques

- Certaines méthodes d'une classe peuvent être exécutées sans qu'on ait à créer une instance de celle-ci.
- Ces méthodes sont appelées *statiques*.
- Afin d'exécuter une méthode statique il faut la préfixer par le nom de sa classe, suivie de `::`

```
LaClasse::laMethode();
```


Conventions

- Le nom d'une classe débute généralement par une majuscule. Ceux des attributs et méthodes par une minuscule.
- Ne manipulez pas les attributs d'une classe à l'extérieur de celle-ci, mais définissez des méthodes pour l'accès:
  - Accesseurs: pour lire un attribut `getAtt()`
    `par exemple: echo bob->getNom();`
  - Mutateurs: pour modifier la valeur `setAtt($var)`
    `par exemple: bob->setNom("Larue");`

Avantages des objets

- L'implémentation d'une classe peut être changée sans que les appels à celle-ci aient à être modifiés.
- La programmation par objets permet de créer du code très modulaire et réutilisable.
- Elle permet de penser en termes d'objets du domaine de l'application.

Concrètement: `toHTML()`

- On programme souvent une méthode `toHTML()` qui permet d'obtenir l'affichage HTML de l'objet.
- Il est généralement avantageux de laisser cette méthode retourner une chaîne de caractères (donc, on n'utilise pas `echo` dans `toHTML()`).

Concrètement: `toHTML()`

- L'utilisation la plus simple de toHTML() est donc

```
echo $monObj->toHTML();
```

- Cette façon de faire simplifie l'utilisation de librairies de gestion de page qui deviennent de plus en plus la norme pour des projets d'envergure.

Concrètement: `$_GET` et `$_POST`

- Il est généralement avantageux de ne pas accéder directement `$_GET` et `$_POST` à l'intérieur d'objets si ceux-ci ne font pas partie de la couche de présentation (« interface HTML »).
- Il vaut mieux ajouter des paramètres aux méthodes et passer `$_GET` ou `$_POST` comme paramètres aux fonctions.

Concrètement: `$_GET` et `$_POST`

- Vous aurez alors des objets qui sont
  - indépendants du protocole HTTP
  - qui peuvent être utilisés dans n'importe quel contexte (web, application indépendante, script, etc.)
  - pour lesquels des tests unitaires sont faciles à concevoir (parce qu'on n'a pas besoin de créer un contexte « web »).

Polymorphisme et héritage multiple

- Le polymorphisme est supporté partiellement en PHP 5. La redéfinition d'une méthode dans une classe enfant permettra de surcharger ce nom et la méthode correspondante sera appelée en fonction du type de l'objet utilisé.
- L'héritage multiple n'est pas supporté en PHP 5.

Notions avancées de POO

- Membres protected
- Interfaces
- Constantes
- Classes abstraites
- Utilisation des méthodes de la classe de base
- Surcharge des méthodes

Pour aller plus loin

- La programmation par objets a permis de jeter de nouveaux regards sur la programmation
  - *Extreme Programming*
  - *Test-driven Development*
  - *Refactoring*
  - Catalogues de *Design Patterns*

Lectures intéressantes

- *Extreme Programming Explained*
  Kent Beck, Addison Wesley (2000)
- *Test-Driven Development By Example*
  Kent Beck, Addison Wesley (2003)
- *Design Patterns*
  Erich Gamma et al., Addison Wesley (1995)
- *Patterns of Enterprise Application Architecture*
  Martin Fowler, Addison Wesley 2003