Programmation Web Serveur

Programmation orientée objet - Encapsulation

D'après les cours de Philippe Renevier

Fabien Givors

Université de Nice Sophia Antipolis

Département Informatique

fabien.givors@unice.fr

RETOUR SUR LA GÉNÉRATION DE MENUS

EXEMPLE : GÉNÉRATION DE MENUS

(Code source présent dans le cours 1)

- Une structure de données (tableau)
- Des fonctions pour les consulter / modifier

- Démarche :
  - Identification du code commun (fonctions)
  - Identification des liens données / fonctions
  - Factorisation
  - Amélioration (partagée) / Partage entre fichier
- Aspect Modulaire (pas de changement pour l’utilsiation)

RÉPÉTITION DE CODE (1/3)

- À la définition :

```
<?php
$formules = array(11.90, 18.90, 22.50);
$specialites = array();
$specialites[] = array(
  "nom"  => "Salade niçoise",
  "prix" => 6.50
);
$specialites[] = array(
  "nom"  => "Socca",
  "prix" => 3.00
);
$specialites[] = array(
  "nom"  => "Pissaladière",
  "prix" => 5.50
);
if($saison == "été")
{
  // Les prix de certains produits augmentent en été
  $formules[0] = 15.90;
  $specialites[1]["prix"] = 4.50;
}?>
```

RÉPÉTITION DE CODE (2/3)

- Pour l'affichage

```
<?php
for($i=0; $i<count($formules); $i++)
{
  echo "\t<li>Formule à ".$formules[$i]." euros.</li>\n";
}
?>
</ol>

<h2>Spécialités</h2>
<ul>
<?php
foreach($specialites as $specialite)
{
  echo "\t<li>La ".$specialite["nom"]." ";
  echo "est à ".$specialite["prix"]." euros.</li>\n";
}
?>
</ul>
```

RÉPÉTITION DE CODE (3/3)

- Types d'opérations

- Initialisation des données

- Ajout de données
- Modification des données
- Retrait de données

- Calculs sur les données

- Affichages des données

LIMITES DU DÉCOUPAGE

- Stockage dans des tableaux limités
  - Gestion d’une indexation commune…
- Problème accentué par les possibles échanges de données entre des fonctions…
  - Variables locales aux fonctions
  - Variables globales dangereuses (mauvaise utilisation : valeurs-initialisation, écrasement, etc.)
- Paradigme : association données et fonctions qui s’y rapportent

1. REGROUPER LES DONNÉES

```php
<?php
$formules = array(11.90, 18.90, 22.50);
$specialites = array();
$specialites[] = array(
  "nom"  => "Salade niçoise",
  "prix" => 6.50
);
$specialites[] = array(
  "nom"  => "Socca",
  "prix" => 3.00
);
$specialites[] = array(
  "nom"  => "Pissaladière",
  "prix" => 5.50
);
if($saison == "été")
{
  // Les prix de certains produits
augmentent en été
  $formules[0] = 15.90;
  $specialites[1]["prix"] = 4.50;
}?>
```

```php
class Menu{
      // données initiales
      private $nom;
      // données supplémentaires
      private $formules = array();
      private $specialites = array();
      // constructeur
      public function Menu($nom)
      {
                  $this->nom = $nom;
      }
…
}
// Instanciation
$menu_ete = Menu("Carte d'été");
$menu_hiver = Menu("Carte d'hiver");
…
```

2. IDENTIFIER LES « MÉTHODES » (=FONCTIONS)

```php
<?php
$formules = array(11.90, 18.90, 22.50);
$specialites = array();
$specialites[] = array(
  "nom"  => "Salade niçoise",
  "prix" => 6.50
);
$specialites[] = array(
  "nom"  => "Socca",
  "prix" => 3.00
);
$specialites[] = array(
  "nom"  => "Pissaladière",
  "prix" => 5.50
);
if($saison == "été")
{
  // Les prix de certains produits
augmentent en été
  $formules[0] = 15.90;
  $specialites[1]["prix"] = 4.50;
}?>
```

```php
class Menu{
     // données initiales
     private $nom;
     // données supplémentaires
     private $formules = array();
     private $specialites = array();
     …
     public function ajouteSpecialite($nom, $prix)
     {
               $this->specialite[] = array(
                   "nom"  => $nom,
                   "prix" => int_val($prix)
               );
     }
…
}
Création :

$menu = Menu("Carte d'été");
$menu->ajouteSpecialite("Socca", 3.00);
…
```

3. UTILISATION : SIMILAIRE AUX FONCTIONS

```
include "classMenu.inc";

$menu_ete = Menu("Menu d'été");
$menu_ete->ajouteSpecialite(... )
…
$menu_ete->afficheListeSpecialites();
$menu_ete->formulaireChoixFormule();
…
// ...
```

POUR ALLER PLUS LOIN...

- Fusion de menus
  - Dans un nouveau menu
  - Dans un menu existant (import)
- Ajout d'une carte
- Association des éléments de la carte aux formules

DES CLASSES POUR STRUCTURER LE CODE

CLASSE : INVERSION DU PROBLÈME

- Association données-fonctions dans des classes
  - Les fonctions deviennent des méthodes (vocabulaire)
  - Les données deviennent des champs de classe (vocabulaire)
- **Ces méthodes et ces champs de classe sont indissociables**
  - **Les méthodes ne fonctionnent qu'avec leurs paramètres et les valeurs ces champs de classe-là**
  - **Ces champs de classe ne sont directement accessible qu'à partir de ces méthodes-là**
- Classe = un modèle
  - de structuration de données (les champs de classes)
  - d'utilisation de ces champs de classes (les méthodes)
  - de publication : qu'est-ce qui est utilisable de l'extérieur, qu'est-ce qui ne l'est pas (ce qui est donc caché) ?

NOTION ET SYNTAXE

- Classe = modèle
  - Déclaration class Nom_De_Classe { }
- Un objet = une concrétisation du modèle
  - une instance (vocabulaire)
  - Respecte le modèle
    - Les champs de classe, mais avec des valeurs qui lui sont propres
    - Les méthodes restent inchangées
  - Dans le code de la classe, pour faire référence à l'objet : $this
  - Pour utiliser une méthode ou un champ de classe : $this->nom_du_champ ou $this->nom_de_la_methode()

EXEMPLE POUR UN « PANIER »

```
class Panier {
    // Eléments de notre panier
    private $items = array( );
      // possesseur du panier : le client !
    private $client ;

    // Ajout de $num articles de type $artnr au panier ; $artnr  est une référence
dans un catalogue
    public function add_item ($artnr, $num) {
          if ($num > 0) {
             if (isset($this->items[$artnr])) $this->items[$artnr] += $num;
             else $this->items[$artnr] = $num;
         }
    }

    /* … d'autres méthodes comme la suppression, obtenir une liste, etc. … */

    // savoir combien de type de produit sont dans le panier
    public function  get_number_product( ) {       return (count($this->items)) ;
    }

    // savoir combien de produit sont dans le panier
    public function  get_number_item( ) {
    $nb = 0;
    foreach ($this->items as $achat) { $nb += $achat ; }
    return $nb ;
    }
}
```

CRÉATION ET UTILISATION

- Création
  - $cart = new Panier();

- Accès aux méthodes d’une classe
  - $cart->add_item("diplome", 1);
    $cart->add_item("semestre", 2);

INITIALISATION

- Initialisation : dans le chargement de la classe et dans le constructuer
  - Chargement = valeur par défaut lors de la déclaration des champs de classe

- Le constructeur est la méthode qui est appelée automatiquement par la classe lorsque vous créez une nouvelle instance d'une classe a l'aide de l'opérateur new.
- La méthode constructeur a le même nom que la classe. Une méthode devient le constructeur si elle porte le même nom que la classe. Un constructeur peut avoir des paramètres avec ou sans des valeurs par défaut
  - Un seul constructeur (nombre de paramètre non discriminant)

```
class Panier {
    /* ... en complément ... */
    public function Panier($client)
        {
        $this->client = $client ;
        }
}
```

UTILISATION DES OBJETS DANS LES FONCTIONS/MÉTHODES

- Un objet peut utiliser un autre objet
- Une classe peut « typer » un paramètre d'une fonction/méthode ou d'un champ de classe

```
include "client.inc"; // contient la définition de la classe Client

class Panier {
   /* … en complément et correction … */
   private Client $client ;

   public function Panier(Client $client)
   {
   $this->client = $client ;
   }
}
```

PUBLIC / PRIVATE : CONTRÔLE L'ENCAPSULATION

- private
  - Ce qui ne sert qu'en interne
  - Ce qui ne doit pas être modifié de l'extérieur (cohérence des valeurs)
  - Ce qui ne doit pas être vu de l'extérieur (limité les dépendances)
- public
  - Ce qui est utilisé à l'extérieur
- Permettent de déterminer la partie visible (public) de l'encapsulation !

COMPLÉMENT SUR LES OBJETS

- Objet dans une string
  - Utilisation de {$...}
  - echo "il y a {$cart->get_number_item()} produits dans votre panier";

- Opérateur instanceof ou fonction is_a
  - $objet instanceof NomClasse
    - Retourne vrai si l'objet est bien un objet de la classe NomClasse
    - if ($cart instanceof Panier) {...}
  - is_a($objet, $nomClasse)
    - Retourne vrai si l'objet est bien un objet de la classe dont le nom est $nomClasse
    - if (is_a($cart, "Panier")) {...}

UTILISATION DES CLASSES / OBJETS

- Juste pour encapsulation
  - généralement un seul objet
- Juste pour structuration
  - Plusieurs objets
  - Simplification des méthodes/fonctions (moins d’indice, moins de if, etc.)
  - Exemple : générateur de tableau HTML

- *Avec héritage, polymorphisme, dynamique, etc.*

EXEMPLE SUR LA BARRE DE NAVIGATION

À PROPOS DU TP 6