Tervezési minták és a PHP 5

Nagy Gusztáv
nagy.gusztav@gamf.kefo.hu

Ajánlott előismeretek

- Programozási alapismeretek
- Osztály, objektum, öröklődés
- Objektumorientált tervezés

- PHP4

Miről lesz szó?

- Tervezési mintákról általában
- Néhány egyszerűbb minta
- Egy konkrét példa mintákra építve (egy általános honlap motor)

- PHP 5 OOP újdonságai

Mi a tervezési minta?

Objektumközpontú szerkezet, amely a gyakorlatban bevált módszerek újrafelhasználását teszi lehetővé.

Valaki egyszer leírta, én meg felhasználom ☺

Fontosabb minták

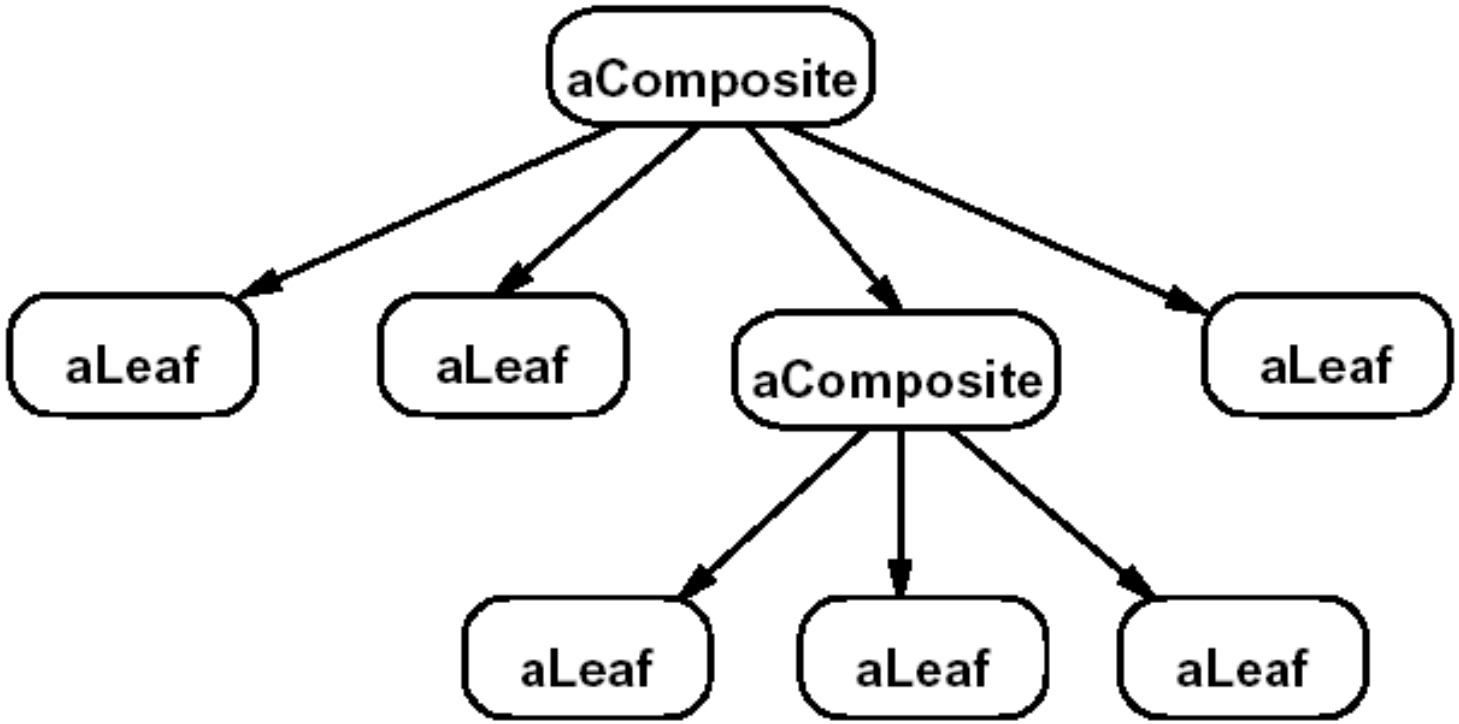
		Cél		
		Létrehozási	**Szerkezeti**	**Viselkedési**
Hatókör	**Osztály**	Gyártófüggvény	(Osztály)illesztő	Értelmező Sablonfüggvény
	Objektum	Elvont gyár Építő Prototípus Egyke	(Objektum)illesztő Híd Összetétel Díszítő Homlokzat Pehelysúlyú Helyettes	Felelősséglánc Parancs Bejáró Közvetítő Emlékeztető Megfigyelő Állapot Stratégia Látogató

Szerkezeti minták

Hogyan alkothatunk meg nagyobb szerkezeteket, hogyan tudjuk összekapcsolni az osztályokat, objektumokat magasabb szintű szolgáltatást nyújtó egységekké.

Összetétel minta

Faszerkezet építése 2 féle objektumból: primitív és összetett objektumból.



Illesztő (Adapter) minta

Adott osztály felületét az ügyfelek által igényelt felületté alakítja.

E nélkül a szolgáltatást nyújtó és igénybevevő osztályok nem tudnának kapcsolódni az interfészek különbözősége miatt.

Híd (Bridge) minta

Az elvont ábrázolást (interfészt) elválasztja a megvalósítástól, hogy a kettő egymástól függetlenül módosítható legyen.

Egyszerű példa: ha halmaz-szerű műveleteket akarunk végezni, ne drótozzuk a kódba egy láncolt listát, lehet, hogy később kiegyensúlyozott keresőfa lesz belőle.

Díszítő (Decorator) minta

Az objektumokhoz dinamikusan további felelősségi körök rendelhetők.

Példa: ablak kiegészítése görgetősávval.

Viselkedési minták

Algoritmusok és felelősségi körök objektumokhoz rendelése áll a középpontban.

Felelősséglánc minta

Egy kérelem lehetséges címzettjeit sorba állítja, amelyen a kérelem addig halad, amíg valamelyik objektum nem tudja kezelni a kérést.

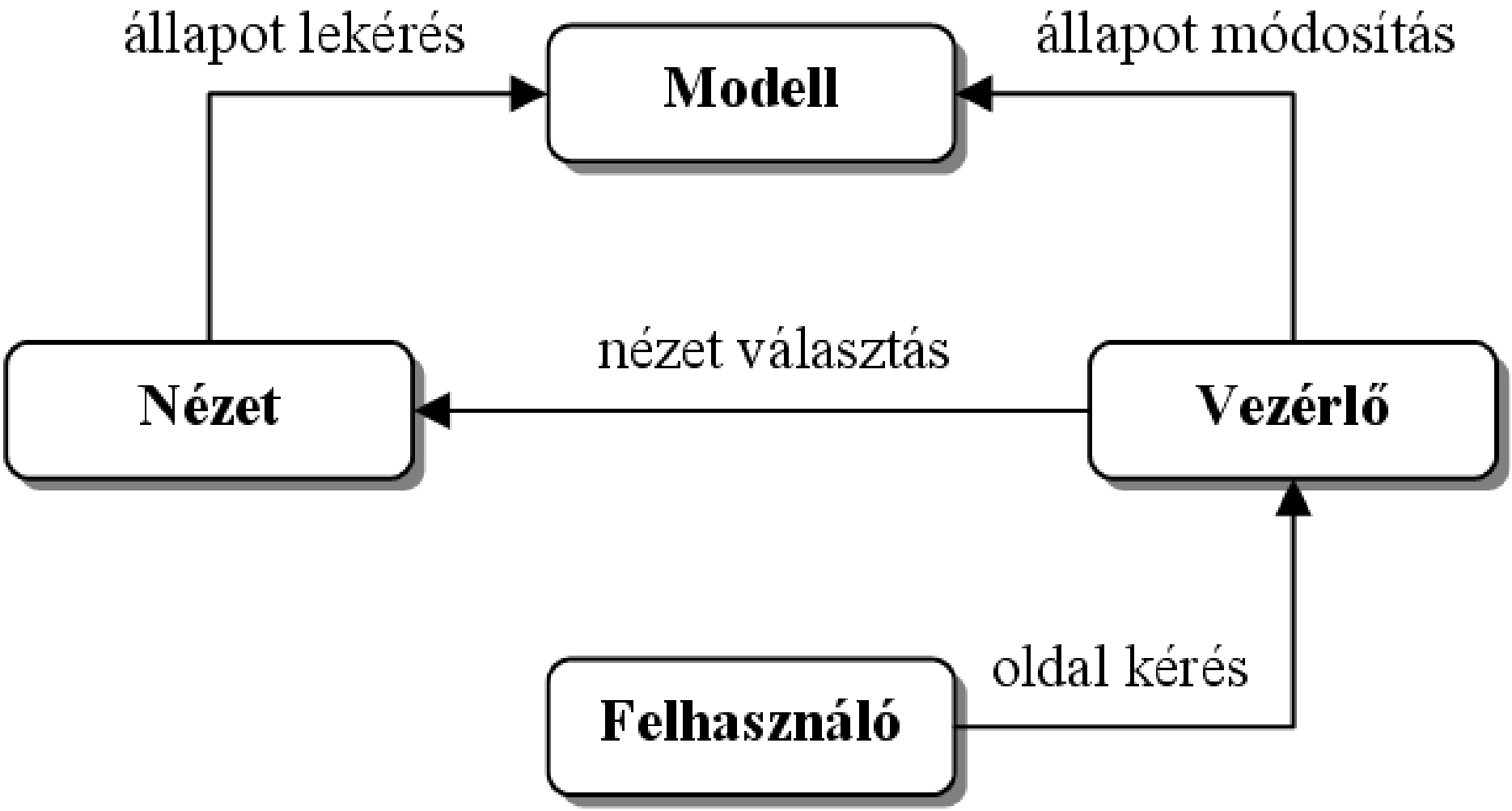
Példa: Grafikus alkalmazás esetén billentyűleütés, vagy egérművelet történt. Melyik komponens fog reagálni?

Megfigyelő (Observer) minta

Gondoskodni kell a kapcsolatban álló objektumok következetességének fenntartásáról úgy, hogy ne legyen túl szoros ez a kapcsolat.

Megoldás: a megfigyelő a megfigyeltnél regisztrálja magát, így a megfigyelt tudja, kiket kell értesíteni esemény bekövetkezésekor.

A Modell-View-Controller minta



Miért jó szétválasztani?

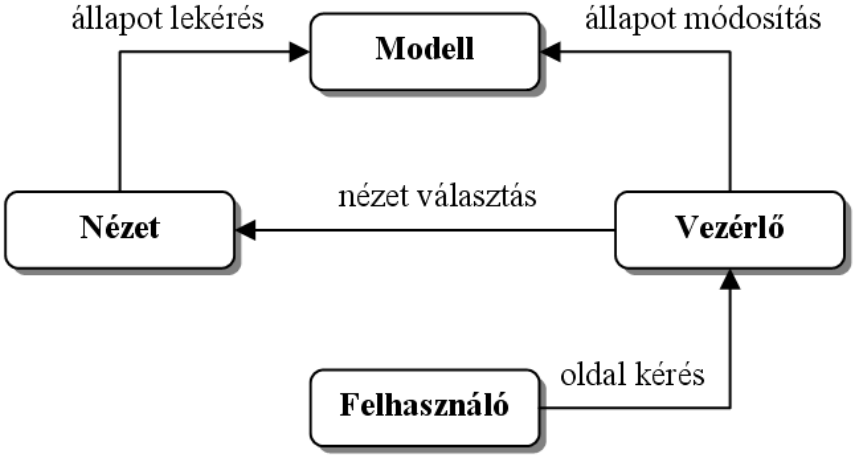
- Bármelyiket könnyen lecserélhetem, a többit nem kell változtatni.

Webfejlesztésben:

- Különböző nézetek kiszolgálása egyszerűen
- Adatábrázolástól független felépítés

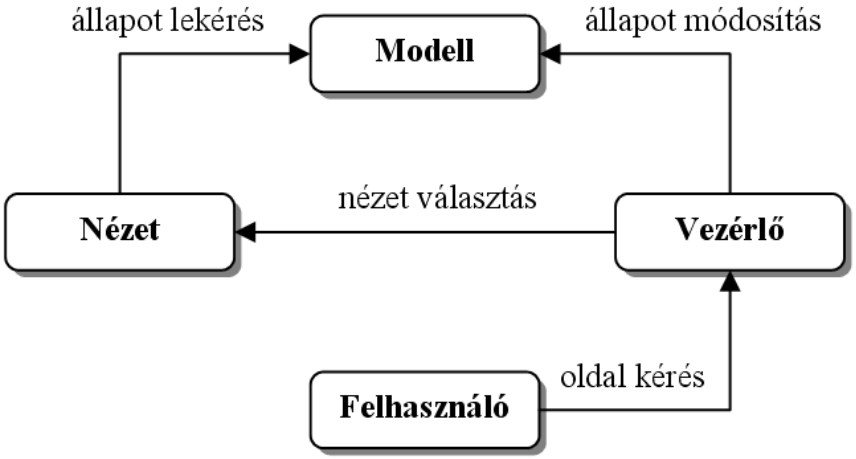
1. verzió: kapcsolatok

```
<?php
class Modell {
}
class Nezet {
  private $modell;
}
class Vezerlo {
  private $modell;
  private $nezet;
}
$modell  = new Modell();
$nezet   = new Nezet();
$vezerlo = new Vezerlo();
?>
```



- osztály
- objektum
- példányosítás
- adatelrejtés

2. verzió: kapcsolatok felépítése 1



- konstruktor neve
- =&
- paraméter típus

```
class Nezet {
  private $modell;
  function __construct(
      Modell & $modell) {
    $this->modell =& $modell;
  }
}
```

2. verzió: kapcsolatok felépítése 2

```
class Vezerlo {
  private $modell;
  private $nezet;
  function __construct() {
    $this->modell = new Modell();
    $this->nezet  = new Nezet
        ($this->modell);
  }
}
$vezerlo = new Vezerlo();
```

A Singleton (Egyke) minta

Garantálja, hogy az osztályból csak egy példányt lehessen létrehozni.

pl. a Vezerlo osztályból véletlenül sem tudunk több példányt létrehozni.

3. verzió: Egyke minta 1

```
class Vezerlo {
  private $modell;
  private $nezet;
  private function __construct() {
    $this->modell = new Modell();
    $this->nezet  = new Nezet
        ($this->modell);
  }
```

- privát konstruktor

3. verzió: Egyke minta 2

```
private static $peldany = null;
  public static function &
  peldanyosit() {
    if ( self::$peldany == null)
      self::$peldany = new Vezerlo();
    return self::$peldany;
  }
}
$vezerlo = Vezerlo::peldanyosit();
```

- statikus függvény, adattag
- self

A Factory (Gyár) minta

Három rokon minta:
- Abstract Factory (Elvont gyár),
- Builder (Építő) és
- Factory Method (Gyártó függvény)

Az objektumok példányosítását rugalmasan kezeljük:
csak futás közben derül ki, melyik osztály példánya lesz.

4. verzió: Többalakúság és Gyár minta

```
interface Modell {
  function alapNezet();
  function & fejlec();
  function & lablec();
  function & tartalom();
  function & balblokk();
  function & jobblokk();
}
```

- interfész

4. verzió: modell

```
class hierMenuModell implements Modell {
  function alapNezet() {
    // itt adatbázisból kellene kiolvasni
    return "Alap";
  }
  function & fejlec() { //...
  }
  function & lablec() { //...
  }
  function & tartalom() { //...
  }
  function & balblokk() { //...
  }
  function & jobblokk() { //...
  }
}
```

- implementáció

4. verzió: nézet

```
abstract class Nezet {
  private $modell;
  function __construct(Modell & $modell) {
    $this->modell =& $modell;
  }
  abstract function mutat();
}
```

- absztrakt függvény
- absztrakt osztály

4. verzió: nézetek

```
class AlapNezet extends Nezet {
  function mutat() {
    echo "<html><body>„
    echo "ok.</body></html>";
  }
}
class NyomtatasNezet extends Nezet {
  function mutat() {
    echo "<html><body>„
    echo "nyomtat...</body></html>";
  }
}
```

- öröklődés

4. verzió: vezérlő konstruktor

```
class Vezerlo {
  private $modell;
  private $nezet;
  private function __construct(
    Modell & $modell, Nezet & $nezet) {
    $this->modell =& $modell;
    $this->nezet  =& $nezet;
  }
  …
```

Már nem a példányosítás a konstruktor feladata.

4. verzió: vezérlő gyártás 1

```
public static function & gyart(
      $modelltip, $nezettip, $vezerlotip) {

    if ( self::$peldany == null) {
      $modellosztaly = $modelltip."Modell";
      $modell  = new $modellosztaly();

      if ( (!isset($nezettip)) ||
           (strlen($nezettip) == 0) )
        $nezettip = $modell->alapNezet();
      $nezetosztaly = $nezettip."Nezet";
      $nezet = new $nezetosztaly($modell);

...
```

4. verzió: vezérlő gyártás 2

```
        $vezerloosztaly =
          $vezerlotip."Vezerlo";
        self::$peldany =
          new $vezerloosztaly($modell,$nezet);
      } // if ( self::$peldany == null) vége
      return self::$peldany;
    }
    ...
}
$vezerlo = Vezerlo::gyart("hierMenu", "",
  "alap");
$vezerlo->mutat();
```

Összefoglalás

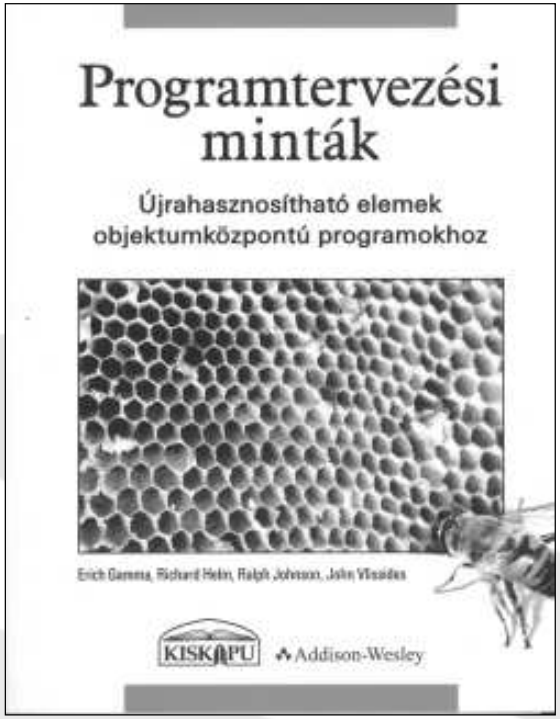
A kezdő programozó arra büszke, hogy az egész kódot ő írta. A profi arra, hogy a kódjának nagy része újra felhasznált.

Nem szégyen, sőt hasznos mások ötleteit ellesni tervezési szinten is. A tervezési mintákat pont ezért írják le az okosok.

Kellemes tanulást kívánok!

Források

- Programtervezési minták
  http://www.kiskapu.hu
  (C++, Smalltalk)



- PHP manual / Classes and Objects / Patterns
  http://www.php.net/manual/hu/language.oop5.patterns.php

Források

- Paul A.J. Braam: Design Patterns applied to Web Programming in PHP
  http://www.cs.vu.nl/~pajbraam/Essay_OOP.pdf
- Core PHP Programming, Chapter 29
  http://www.phptr.com/content/images/0130463469/samplechapter/0130463469_ch29.pdf
  (letölthető minta fejezet)