

Langages Web 1



Programmation coté serveur

Florent Nicart

Université de Rouen

2016–2017

Application Web

Définition (Application Web)

Logiciel applicatif manipulable par l'intermédiaire d'un navigateur web.

- Autre dénomination : « site web dynamique » ou « **WebApp** »,
- Les pages web sont des *vues* de l'état d'une application qui s'exécute de manière distante.
- Dans ce mode, les (représentation des) ressources renvoyées par le serveur web sont calculées et non plus stockée → programmation coté serveur.
- Exemple : webmails, logiciels de gestion de contenu (CMS), environnements numériques de travail, site de vente par correspondance, interface de configuration de micrologiciel (routeur) ou de service (CUPS), etc.

Le point

- Pour l'instant les éléments vus nous permettent d'implémenter une partie de l'interface de l'application (HTML+CSS+JavaScript/AJAX).

PROBLEME :

- ❶ le serveur Web (par défaut) ne renvoie que des ressources statiques,
- ❷ la communication entre l'interface et le serveur se fait par le protocole HTTP, pas d'observateur/observé,
- ❸ et celui-ci est sans état, c-à-d. que le navigateur et le serveur ne souviennent pas l'un de l'autre à chaque échange.

Contenu dynamique

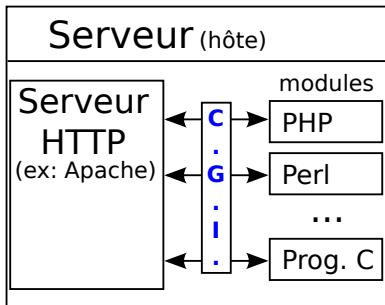
- Pour produire des ressources dont la représentation en fonction des actions de l'utilisateur, l'ajout de programmes est nécessaire en plus du serveur web (http) délivrant un contenu statique.

Plusieurs approches possibles :

- ① un processus autonome gérant les connexions *HTTP* et le parallélisme lui-même (ex : cups, micrologiciel d'un routeur/box internet),
- ② un processus invoqué à chaque requête par le navigateur avec l'interface C.G.I. (ex : perl, PHP, etc),
- ③ un serveur d'applications qui implémente le middleware (infrastructure communicante) et accueil des processus applicatifs (conteneur), par exemple *JEE* avec *JSP* (servlet) ¹.

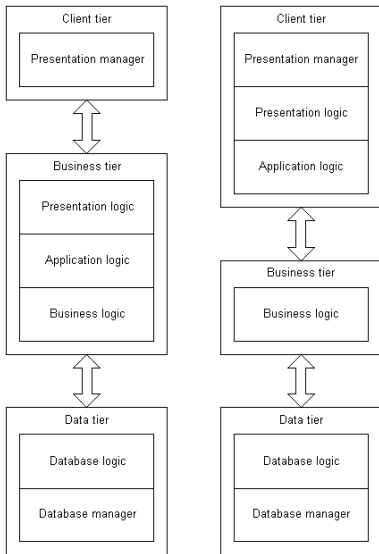
1. Java Enterprise Edition avec Java Server Page

Rappel C.G.I.



- Permet au programme d'accéder aux éléments de la requêtes (données + [évent.] en-tête HTTP),
- et de produire une réponse (représentation de la ressource + [évent.] en-tête HTTP).

Applications distribuées

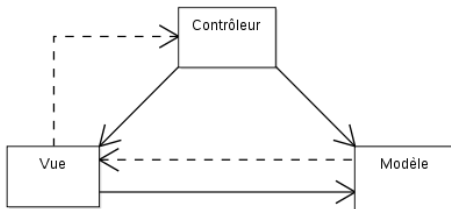


- Par définition, les applications web sont distribuées et ont des architectures **client / serveur** ou **2-tiers**.
- Lorsque l'on ajoute une couche de **persistance** (sauvegarde non volatile des données), le découpage en **3-tiers** devient naturel.
- Modèle en couche.
- Le modèle n -tiers existe.

Architecture logicielle

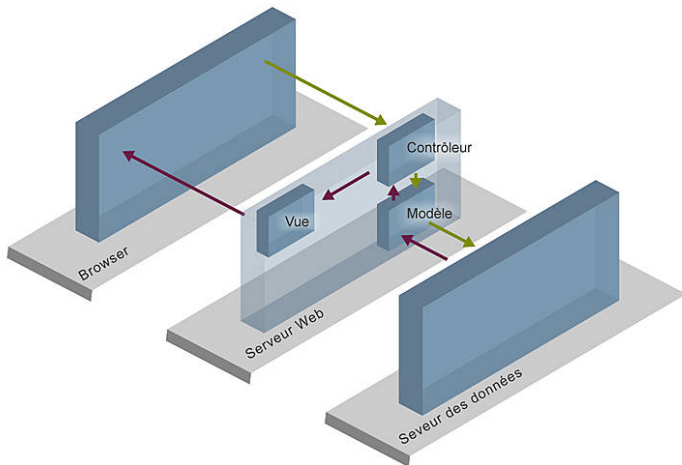
- Les applications peuvent être découpées d'un point de vue logiciel d'après des patrons de conception : MVC, PAC, ALV, MVC2, etc.
- Ce découpage découple les différents aspects d'une application,
- il rend le code modulaire, évolutif, flexible et réutilisable.

Rappel MVC



- Le **Modèle** : contient le modèle de données mais aussi la logique métier, la partie applicative qui réalise les transformations valides du modèle.
- La/les **Vue/s** : la partie interactive en contact avec l'utilisateur, inclu la logique de présentation.
- Le(s) **Contrôleur(s)** : orchestrent les interactions entre les vues (synchro) et le modèle (écriture).

MVC et tiers WEB



Le MVC en Web

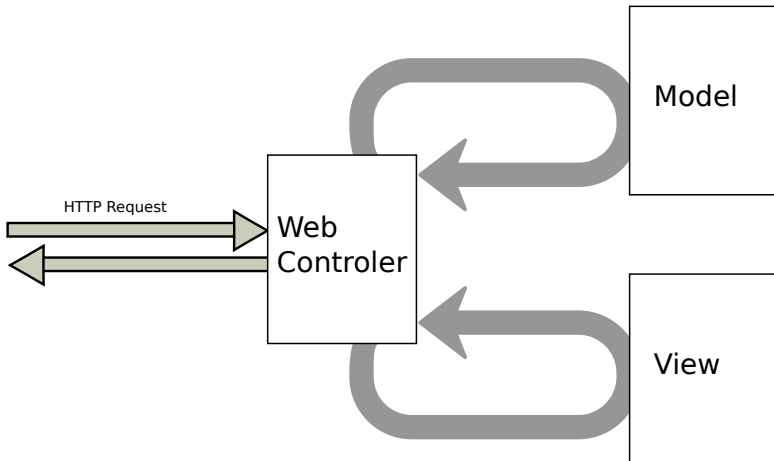
Schématiquement, coté serveur, on peut définir le modèle MVC de la manière suivante :

- **contrôleur(s)** : le ou les programmes qui reçoivent les requêtes en provenance de l'interface (scénario d'utilisation) ;
- **modèle** : les bibliothèques/classes qui encapsulent la gestion de l'état de l'application (variables de session et données persistantes) ;
- **vue** : les bibliothèques/classes permettant de mettre en forme le résultat d'un traitement (présentation des données, des erreurs, etc.).

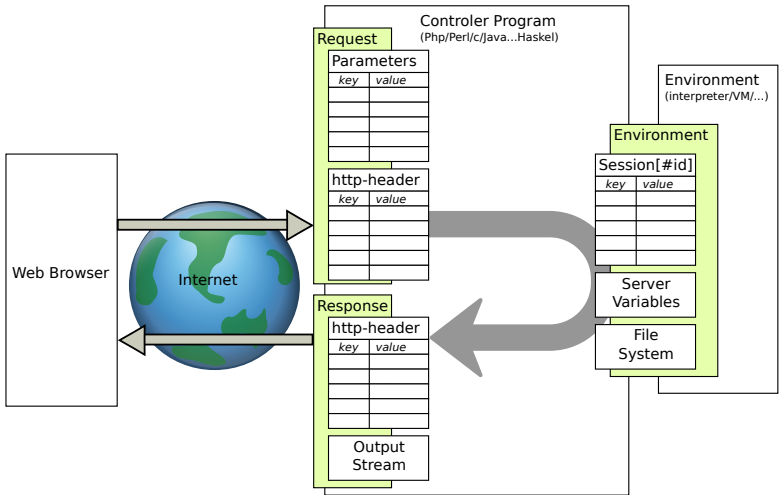
Le respect de ce modèle consiste en la séparation physique de ces aspects (classes mais également fichiers !).

Le MVC en Web...

...ressemble plutôt à ceci.



Environnement général



Applications Web avec états

Rappels

- Qu'elle soit interprétée ou compilée et/ou fonctionnant sur un serveur applicatif, une application Web n'a pas de processus permanent.
- Un processus est créé à réception de la requête HTTP dans lequel l'application s'exécute et se termine une fois la réponse produite.
- Les variables créés sont perdues.
- L'état de l'application a besoin d'être sauvegardé entre chaque requête...

Liaison d'une interface à un état de l'application

- Option 1 : véhiculer l'état dans les requêtes et les réponses,
- vers le serveur :
`/do.php?nom=toto&prenom=titi`

- vers le client :

```
1  ...  
2  <input type="text" name="nom" value="toto">  
3  <input type="text" name="prenom" value="titi">  
4  ...
```

- Inconvénients : peut être complexe à implémenter, générer un trafic important, l'état est maintenu coté client (pas de sécurité).

Liaison d'une vue à un état de l'application

- Option 2 : mémoriser dans des cookies,
- vers le client :

```
1 HTTP/1.1 200 OK
2 ...
3 Set-Cookie : NOM=toto ; expires=Wed, 13-Nov-2013 18:56:37
  GMT; path=/; domain=.monsite.org
4 Set-Cookie : PRENOM=titi ; expires=Wed, 13-Nov-2013
  18:56:37 GMT; path=/; domain=.monsite.org
5 ...
```

- vers le serveur :

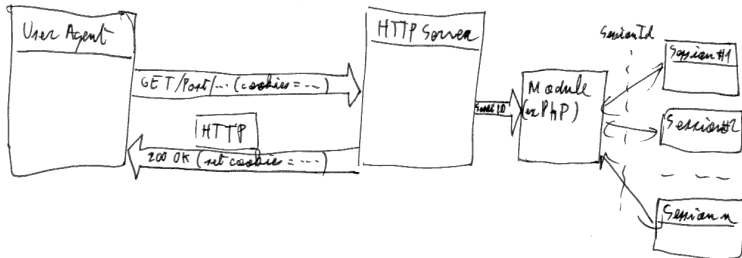
```
1 GET / HTTP/1.1
2 Host : monsite.org
3 ...
4 Cookie : NOM=toto ;NID=titi
5 ...
```

- Inconvénients : idem option 1 + envoi systématique.

Les sessions Web

- Les **sessions Web** sont la solution pour maintenir un état consistant d'une exécution d'un contrôleur à l'autre.
- L'environnement permet de retrouver un espace de stockage identique à l'invocation précédente.
- Tout ce qui y a été placé est retrouvé, tout le reste est perdu.
- L'espace est retrouvé grâce à un identifiant transmis systématiquement à chaque requête (en général par un cookie)...

Association d'un état à un utilisateur



Les sessions Web

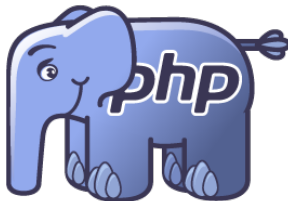
Considérations techniques

- Une session web est associée à un utilisateur : plus précisément à une instance d'un navigateur web (sur un compte système/sur une machine donnée).
- Elle est liée/crée à la demande du programme (ex : `session_start()`).
- Session Web \neq Session utilisateur !
- Une session en général = un cookie \Rightarrow question :
 - si le même utilisateur ouvre deux navigateurs différents sur le même site ?
 - Le même site sur deux ordinateurs différents ?
- S'il appelle le contrôleur `/main.php` puis `/admin.php` ?
- Sécurité : l'identifiant de session est sacré : génération, transmission, changement d'identité (*Session fixation*).

La partie modèle

On peut donc considérer que le modèle de l'application est composé des aspect suivants :

- la partie volatile (perdue à la fermeture de l'application) est représentées par les données de session,
- la partie permanente du modèle (la persistance) peut être réalisée par :
 - des fichiers plats (sérialisation),
 - un SGBD.
- en général, le modèle persistant est implémenté par une base de données et une couche d'abstraction (par objets : ORM²).



Le langage PHP

Introduction

- PHP est un langage de scripting coté serveur (code interprété),
- créé en 1994 par Rasmus Lerdorf (en Perl),
- réécrit en C et publié en 1995 : Personal Home Page Tools/Form Interpreter,
- repris en 1997, par deux étudiants, Andi Gutmans et Zeev Suraski qui redéveloppèrent un nouveau coeur pour la V3 : *PHP : Hypertext Preprocessor*³,
- modèle orienté objet réécrit à la version 5,
- Version actuel : 5.3.8, la prochaine est la v6.

Un (vrai) langage de scripting

- Amusant (hello.php) :

```
1 #!/usr/bin/php
2 blablabla
3 <?php
4     echo "coucou_la_planete\nNous_sommes_le_" .
5         date("j/m/Y")."\n";
6 ?>
7 bliblibli
```

- Dans un terminal :

```
florent@penpen:~/$ chmod u+x hello.php
florent@penpen:~/$ ./hello.php
blablabla
coucou la planete
Nous sommes le 13/11/2011
bliblibli
florent@penpen:~/$
```

PHP est un pré-processeur

Programmation serveur

Applications Web
Architectures
distribuées
MVC Web
Modèle général
Session Web

Le langage PHP

Introduction
Le langage
Syntaxe générale
Types
Opérateurs
Flots d'exécution
Les tableaux
Les chaînes
Fonctions
Classes et objets
Bibliothèques
C.G.I.
Cookies
Sessions
Frameworks

- À l'instar du préprocesseur des langages C/C++ (#include, etc ...)
- il est invoqué par le serveur web sur un fichier (.php),
- l'interpréteur PHP traite/transforme le contenu du fichier et transmet le résultat au serveur http,
- le serveur http récupère la sortie du script comme représentation de la ressource et (éventuellement) des données d'en-tête http

Syntaxe de PHP

- Le préprocesseur traite uniquement les instructions placées entre balises `<?php` et `?>`,
- mais le flux englobant peut être modifié (voir flots d'exécution plus loin...),
- les commentaires sont placés entre `/*` et `*/`, ou après `//` ou bien après `#` (et oui c'est un langage de script),
- le code est insensible à la casse, excepté pour les noms de variables (`$myvar` \neq `$MyVaR`),
- les variables sont préfixées systématiquement par `$`,
- et sont déclarées et typées implicitement :

```
1 $my_var = 710;^^| // my_var est un entier.  
2 $my_var = "sept_cent_dix"; // my_var est une chaine.
```


Types et constantes en PHP

Les types de base :

- entiers : `54`
- flottants : `54.3`
- chaînes : `"54"` (contenu interprété, ex `"54\n"`) ou `'54'` (contenu protégé),
- booléens : `false` ou `true`

Constantes symboliques :

- Définition : `define("PI", 3.14)`
- utilisation : `echo(PI)` (pas de `$`)

Fonctions sur les types :

- `isset($var)` : renvoie `true` si `$var` existe
- `unset($var)` : détruit `$var`
- `is_integer($var), is_string($var), ...` : renvoie `true` si `$var` est du type correspondant.

Opérateurs en PHP

- Arithmétiques : + - * / % ++ --
- Affectation : = .= += -= *= /= %=
- Comparaison : == < != > === <= !=> =>
- Logiques : and && or || xor !
- Conditionnel : ... ? ... : ...

Notes :

- . est l'opérateur de concaténation sur les chaînes,
- === et !== sont des comparaisons strictes : même type et, pour les tableaux associatifs, même ordre.

Flots d'exécution en PHP

Instructions conditionnelles

- Syntaxe :

```
1  if (cond) { ... }
2  elseif (cond) { ... }
3  else { ... }
```

- Exemple :

```
1  debut
2  <?php
3      if (1<2) {
4          ?>
5          Nous sommes le <b>
6          <?php
7              echo date("j/m/Y")."\n";
8          ?>
9          </b>
10         <?php
11             } else {
12                 ?>
13                 La fin des temps est arrivee !
14                 <?php
15                     }
16                 ?>
17                 fin
```

Résultat :

```
debut
Nous sommes le <b>
15/11/2011
</b>
fin
```

Flots d'exécution en PHP

Instructions conditionnelles 2

- Syntaxe du switch :

```
1  switch ( expr ) {  
2      case VALEUR1 :  
3          ...  
4      break ;  
5      case VALEUR2 :  
6          ...  
7      break ;  
8      default :  
9          ...  
10     break ;  
11 }
```

Flots d'exécution en PHP

Instructions d'itération

- Instruction **for** :

```
1  for ( init ; cond ; modif ) { ... }
```

- Instruction **foreach** :

```
1  foreach ( array_expression as $value ) { ... }  
2  foreach ( array_expression as $key => $value ) { ... }
```

- Instruction **while** :

```
1  while ( cond ) { ... }
```

- Instruction **do - while** :

```
1  do {  
2    ...  
3  } while ( cond ) ;
```

Flots d'exécution en PHP

Instructions d'itération – Exemple avec **for**

```
1  debut
2  <?php
3      for ($i=0; $i<6; $i++) {
4          echo $i."┐┐┐";
5      ?>
6  Vive le langage <b>PHP</b><br>
7  <?php
8      }
9  ?>
10 fin
```

Résultat :

```
debut
0 - Vive le langage <b>PHP</b><br>
1 - Vive le langage <b>PHP</b><br>
2 - Vive le langage <b>PHP</b><br>
3 - Vive le langage <b>PHP</b><br>
4 - Vive le langage <b>PHP</b><br>
5 - Vive le langage <b>PHP</b><br>
fin
```

Tableaux en PHP

Déclaration

- Tableaux classiques (clé entière) sont vus comme des tableaux associatifs (clé de type chaîne),
- clés et valeurs ne sont pas typées :

```
1 <?php
2 // Explicitement :
3 $stab = array(23, "coucou") ;
4 $asso = array("foo" => "bar", 12 => true) ;
5
6 // Ou implicitement :
7 $stab[0]=23 ;
8 $stab[1]="coucou" ;
9 $asso["foo"]="bar" ;
10 $asso[12]=true ;
11
12 echo $stab[0]; // 23
13 echo $stab[1]; // "coucou"
14 echo $asso["foo"]; // bar
15 echo $asso[12]; // true
16 ?>
```

Tableaux en PHP

Parcours de tableaux

- Parcours avec `for` et `foreach`

```
1 <?php
2 // Explicitement :
3 $tab = array(23, "coucou");
4 $asso = array("foo" => "bar", 12 => true);
5
6 // Parcours classique
7 for ($i=0; $i<count($tab); $i++) {
8     echo "$tab[$i]\n";
9 }
10
11 // Parcours avec foreach :
12 foreach ($tab as $value) { echo "$value\n"; }
13 foreach ($asso as $key => $val) { echo "$key=$val\n"; }
14 ?>
```


Tableaux en PHP

LE pointeur interne

- Les tableaux en PHP disposent d'un pointeur interne (sorte d'itérateur) :

```
1 <?php
2 $transport = array('foot', 'bike', 'car', 'plane');
3 $mode = current($transport); // $mode = 'foot';
4 $mode = next($transport);    // $mode = 'bike';
5 $mode = current($transport); // $mode = 'bike';
6 $mode = prev($transport);    // $mode = 'foot';
7 $mode = end($transport);     // $mode = 'plane';
8 $mode = current($transport); // $mode = 'plane';
9 ?>
```

- manipulable avec les fonctions : `reset($tab)` ,
`next($tab)` , `prev($tab)` , `current($tab)` ,
`key($tab)` , ...

Tableaux en PHP

Quelques fonctions utiles

- `sort($tab)` : trie les valeurs et réaffecte les indices,
- `asort($tab)` : trie les valeurs et ne réaffecte pas les indices,
- `rsort($tab)` : idem `sort` mais dans l'ordre inverse,
- `arsort($tab)` : idem `asort` mais dans l'ordre inverse,
- `ksort($tab)` : trie les indices,
- `krsort($tab)` : idem `ksort` mais dans l'ordre inverse,
- `usort($tab, $critere)`,
`uasort($tab, $critere)`,
`uksort($tab, $critere)` : trie selon un critère

Chaînes en PHP

Déclaration

- Constantes délimitées par ' (contenu non interprété) ou par " (contenu interprétable) :

```
1 $dollar=10;  
2 $s1 = 'the_price_is_$dollar'; // 'the price is $dollar';  
3 $s2 = "the_price_is_$dollar"; // 'the price is 10';
```

- L'opérateur de concaténation est ., strlen(\$str) donne la longueur de la chaîne et les crochets permettent de l'indicer :

```
1 $str="hello_" . 'world!';  
2 echo $str[strlen($str)-3]; // 'd';
```

Chaînes en PHP

Quelques fonctions utiles

- `str_repeat(ch, nb)` : répétition de `ch` `nb` fois,
- `strtolower(ch)` : conversion en minuscules,
- `strtoupper(ch)` : conversion en majuscules,
- `ucwords(ch)` : initiales en majuscules,
- `ucfirst(ch)` : 1re lettre en majuscule,
- `ltrim(ch, liste)` : suppression de caractères au début,
- `rtrim(ch, liste)` : suppression de caractères à la fin,
- `trim(ch, liste)` : suppression de caractères au début et à la fin,

Chaînes en PHP

Quelques fonctions utiles

- `strstr(ch, ch2)` : retourne une sous-chaîne de `ch`, allant de la première occurrence de `ch2` jusqu'à la fin de la chaîne (sensible à la casse),
- `stristr(ch, ch2)` : idem insensible à la casse,
- `substr(ch, indice, l)` : retourne la sous-chaîne de `ch` commençant à `indice` et de longueur `l`,
- `substr_count(ch, ssch)` : nombre occurrences sans chevauchement,
- `str_replace(oldssch, newssch, ch)` : remplacement toutes les occurrences,
- `strpos(ch, ssch), stripos(ch, ssch)` : position de la première occurrence

Chaînes en PHP

Fonctions de validation et d'échappement

Ces fonctions permettent de protéger des chaînes contre l'interprétation :

- **HTML / XML / XHTML** : `htmlspecialchars()`,
`htmlspecialchars_decode()`,
`html_entity_decode()`, `strip_tags()`,
- **URL** : `urlencode()`, `rawurlencode()`,
`urldecode()`, ...,
- **SGBD** : `pdo->quote()`,
`mysqli_real_escape_string()`,
`mysqli_bind_param()`, `mysqli_bind_result()`,
`mysql_escape_string()`,
`mysql_real_escape_string()`,
`pg_escape_string()`, `pg_escape_bytea()`,
`sqlite_escape_string()`, ...,

Chaînes en PHP

Fonctions de validation et d'échappement

- **dates** : `checkdate()`
- **Système** : `escapeshellcmd()`,
`escapeshellarg()`,
- **Chaînes** : `Quotemeta()`, `ctype_alnum()`,
`ctype_alpha()`, `ctype_cntrl()`,
`ctype_digit()`, `ctype_graph()`,
`ctype_lower()`, `ctype_print()`,
`ctype_punct()`, `ctype_space()`,
`ctype_upper()`,
`ctype_xdigit()`, `addslashes()`,
`addcslashes()`

Les fonctions en PHP

```
1 <?php
2 function double($n) {
3     $n *= 2;
4     return $n;
5 }
6 $x = 12;
7 echo "double_=" . double($x) . ", valeur=" . $x . "\n";
8 echo "double_=" . double(&$x) . ", valeur=" . $x . "\n";
9 ?>
```

Résultat :

double = 24, valeur = 12

double = 24, valeur = 24

- Les paramètres sont vus comme variables locales,
- passage des paramètres par valeur (par défaut),
- passage par référence à l'appel avec l'opérateur &.

Les fonctions en PHP

Portée des variables

```
1 <?php
2 function just_called() {
3     static $count1 = 0;      // Done only once.
4     global $count2;          // Maps to the global variable.
5     $count1++; $count2++;
6     echo "called with ($count1 , $count2)\n";
7 }
8
9 $count2=0;
10 just_called(); just_called();
11
12 echo "count1 : " . $count1 . "\n"; // Error
13 echo "count2 : " . $count2 . "\n";
14 ?>
```

Résultat :

called with (1, 1)

called with (2, 2)

PHP Notice: Undefined variable: count1 in function2.php on line

count1 :

count2 : 2

Classes et objets en PHP

- Syntaxe similaire à Java (héritage simple, interfaces, ...)

```
1 <?php
2 interface IA {
3     public function setVariable($val);
4 }
5
6 class SimpleClass extends Base implements IA, IB {
7     // properties :
8     private $var = 'a_default_value';
9
10    // methodes :
11    public function displayVar() {
12        echo $this->var; // Et non $var
13    }
14    public function setVariable($val) {
15        $this->var=$val;
16    }
17    ...
18 }
19
20 $s = new SimpleClass();
21 ?>
```

Objets en PHP

La pseudo variable *\$this*

- *This* dépend du contexte d'exécution :

```
1 <?php
2 class A {
3     function toto() {
4         if (isset($this)) echo '$this_is_defined(' .
5             get_class($this) . ")\n";
6         else echo "\$this_is_not_defined.\n";
7     }
8 }
9
10 class B {
11     function titi() { A::toto(); }
12 }
13
14 $a = new A(); $a->toto(); // defined (A)
15 A::toto(); // not defined
16 $b = new B(); $b->titi(); // defined (B)
17 B::titi(); // not defined
18 ?>
```

Classes et objets en PHP

Quelques fonctionnalités (en résumé)

- Modificateurs de portée : `public`, `private`, `protected` ;
- héritage simple ;
- Supporte les interfaces et l'héritage entre interface ;
- Les constantes de classe sont introduite avec **`const`** ;
- `static` introduit des méthodes ou propriétés de classe ;
- L'opérateur de résolution de portée `::` permet d'accéder aux membres *statiques* ou *constants*.
- `abstract` déclare une classe ou une méthode abstraite ;
- `final` interdit la surcharge d'une méthode ;

Classes et objets en PHP

Méthodes magiques

- Les méthodes magiques sont des méthodes qui sont appelées implicitement.
- Leur nom commence par `__` (deux soulignés bas),
- Elles peuvent être surchargée pour implémenter la fonctionnalité correspondante : `__construct()`, `__destruct()`, `__call()`, `__callStatic()`, `__get()`, `__set()`, `__isset()`, `__unset()`, `__sleep()`, `__wakeup()`, `__toString()`, `__invoke()`, `__set_state()` **et** `__clone()`

Classes et objets en PHP

Constructeurs et destructeurs

- Problème avec l'ancienne syntaxe :

```
1 class A {  
2     // Le constructeur de A :  
3     public function A() {  
4         ...  
5     }  
6     // Juste une fonction standard mais,  
7     // qui sera constructeur dans la classe B :  
8     public function B() {  
9         ...  
10    }  
11 }  
12  
13 class B extends A {  
14 }  
15  
16 // Utilise implicitement A::B() comme constructeur  
17 $b = new B() ;
```

Classes et objets en PHP

Constructeurs et destructeurs

- Privilégier la syntaxe à méthodes magiques :

```
1 <?php
2 class A {
3     // Le constructeur de A :
4     public function __construct() { ... }
5     // Le destructeur de A :
6     public function __destruct() { ... }
7 }
8
9 class B extends A {
10    // Le constructeur de B :
11    public function __construct() {
12        parent::__construct(); // Necessaire !
13    }
14    // Le destructeur de B :
15    public function __destruct() {
16        parent::__destruct(); // Necessaire !
17    }
18 }
```

Bibliothèques en PHP

- `require ("chemin/fichier")`
- `include ("chemin/fichier")`
- `require_once ("chemin/fichier")`
- `include_once ("chemin/fichier")`

Les variantes `include` provoquent des warnings au lieu d'erreurs en cas de problème, les variantes `_once` n'incluent le fichier que si celui n'a pas déjà été inclu.

PHP et C.G.I.

Récupération des données de formulaire

- La lecture des données transmises par un formulaire se fait par les tableaux `$_GET` (méthode GET), `$_POST` (méthode POST) et `$_FILES` (pour les fichiers envoyés).

```
1 // Exemple de donnees transmises :
2 // nom=Dupont&prenom=martin&age=10
3
4 // Lecture avec la methode POST :
5 if (isset($_POST[ 'nom' ])) {
6     $lenom=ucfirst($_POST[ 'nom' ]);
7     $leprenom=ucfirst($_POST[ 'prenom' ]);
8
9     if (is_int ($_POST[ 'age' ])) age=$_POST[ 'age' ];
10    else return ERROR
11    ...
```

PHP et C.G.I.

Opérations sur les en-têtes

- La fonction `getallheaders()` ⁴ permet d'accéder à l'en-tête HTTP de la requête :

```
1 <?php
2
3 foreach (getallheaders() as $name => $value) {
4     echo "$name :␣$value\n" ;
5 }
6
7 ?>
```

4. Alias de la fonction `apache_response_headers()`


PHP et C.G.I.

Opérations sur les en-têtes

- La fonction `header()` permet d'ajouter des informations à l'en-tête HTTP de la réponse.

Exemples :

```
1 <?php
2 header( ' Location :_http ://www.example.com/ ' ) ;
3 header( "HTTP/1.0 _404_Not_Found" ) ;
4 header( "Cache-Control :_no-cache ,_must-revalidate" ) ;
5 header( " Expires :_Sat ,_26_Jul_1997_05 :00 :00_GMT" ) ;
6 ?>
```

-  Les productions de l'en-tête doivent être faites avant la production de la ressource. `headers_sent()` permet de tester si l'en-tête a été envoyé :

```
1 <?php
2 echo ' Juste_coucou ! ' ;
3 header( ' Content-type :_texte / txt ' ) ; // Erreur !
4 ?>
```

Gestion des cookies

- Les cookies envoyés par le navigateur sont accessibles par la variable globale `$_COOKIE`.
- Un nouveau cookie peut être défini avec les fonctions :

```
1  bool setcookie ( string $name [, string $value note[, int $expire = 0 [,  
    string $path [, string $domain [, bool $secure = false [, bool  
    $httponly = false ]]]]] )  
2  ?>  
3  // Donnees brutes (sans encodage URL) :  
4  bool setrawcookie ( string $name [, string $value [, int $expire = 0 [,  
    string $path [, string $domain [, bool $secure = false [, bool  
    $httponly = false ]]]]] )
```

- équivalent à :

```
1  header ( "Set-Cookie : nom=toto ; expires=Wed, 13-Nov-2013 18:56:37GMT ; path  
    =/ ; domain=.monsite.org" );
```

- donc :

```
1  echo "voici la ressource";  
2  bool setcookie ("votreid", "XazeAzeaEZ"); // Erreur !
```

Gestion des sessions

- `session_start()` : association de la variable `$_SESSION` avec le tableau de variables associé à l'identifiant transmis par le client.
- Si aucun identifiant n'est transmis, création d'un nouveau tableau associé à un nouvel identifiant qui sera transmis au client dans la réponse⁵.
- Un nouveau cookie peut être défini avec les fonctions :

```
1 <?php
2 session_start() ;
3 // Utilisez $HTTP_SESSION_VARS avec PHP 4.0.6 ou plus ancien
4 if (!isset($_SESSION['count'])) {
5     $_SESSION['count'] = 0;
6 } else {
7     $_SESSION['count']++;
8 }
9 ?>
```

Gestion des sessions

Quelques fonctions de manipulation de sessions

- `session_start` : initialise/lie une session,
- `session_destroy` : détruit une session,
- `session_id` : lit et/ou modifie l'identifiant courant de session,
- `session_regenerate_id` : remplace l'identifiant de session courant par un nouveau⁶,
- `session_is_registered` : vérifie si une variable est enregistrée dans la session,
- `session_commit` : alias de `session_write_close`,
- `session_write_close` : écrit les données de session et ferme la session,

6. À utiliser systématiquement après *login* pour éviter une attaque du type **session fixation** !

Gestion des sessions

Quelques fonctions de manipulation de sessions

- `session_get_cookie_params` : lit la configuration du cookie de session,
- `session_set_cookie_params` : modifie les paramètres du cookie de session,
- `session_name` : lit et/ou modifie le nom de la session,
- `session_unregister` : supprime une variable de la session,
- `session_unset` : détruit toutes les variables d'une session...

Utilisation d'un Framework

Une courte présentation de Symfony

- Choix arbitraire, d'autres cadres comme *Zend* méritent d'être étudiés...
- Symfony permet d'accélérer le développement d'application web tout en imposant le modèle MVC.
- Il impose une organisation particulière des contrôleurs et des vues.
- Il offre deux environnements d'exécution : *développement* (avec outils de débogage/profilage) et *production*.
- Les *bundles* regroupent les ressources (fichiers php, style, javascript, images, ...) par modules.

Le cadriciel Symfony

Structure de l'application

```
app/  
    cache/  
    config/  
    logs/  
    Resources/  
bin/  
src/  
    Acme/  
        DemoBundle/  
        Controller/  
        Resources/  
...  
vendor/  
    symfony/  
    doctrine/  
...  
.
```

Le cadriceil Symfony

Les controleurs

- Le contrôleur est le programme d'entrée qui reçoit la requête.
- Symfony fourni un contrôleur par défaut,
- celui délègue le traitement aux sous-contrôleurs fournis par le développeur en fonction de l'url. Ex :
`http://host/controleur.php/action/param1/pa`
- Cet aiguillage s'appelle le **routing** et est décrit dans un fichier de configuration `app/config/routing.yml`

Le cadriceil Symfony

Les controleurs

- Exemple de fichier de configuration de pour le contrôleur :

```
1  # app/config/routing_dev.yml
2  _welcome :
3      pattern : /
4      defaults : { _controller : AcmeDemoBundle :Welcome :index
5                  }
6  _demo :
7      resource : "@AcmeDemoBundle/ Controller / DemoController .
8                php"
9      type :      annotation
10     prefix :    /demo
11 # ...
```

- la première règle fait pointer la racine du site vers la méthode `indexAction()` du contrôleur `Welcome` dans le bundle `AcmeDemoBundle`.

Le cadre Symfony

Les contrôleurs

- L'objectif d'un contrôleur est de recevoir une requête et de renvoyer une réponse.
- Les contrôleurs n'utilisent pas les variables globales de PHP vues précédemment (comme `$_GET`) mais utilisent les objets `request` et `response`.
- un (mauvais) exemple minimal :

```
1 use Symfony\Component\HttpFoundation\Response ;  
2 $name = $request->query->get( 'name' ) ;  
3 return new Response( 'Hello <b>'. $name. ' </b>', 200, array  
    ( 'Content-Type' => 'text/plain' ) ) ;
```

Le cadriceil Symfony

Les controleurs

- un (meilleur) exemple de contrôleur :

```
1 // src/Acme/DemoBundle/Controller/WelcomeController.php
2 namespace Acme\DemoBundle\Controller;
3 use Symfony\Bundle\FrameworkBundle\Controller\Controller;
4
5 class WelcomeController extends Controller {
6     public function indexAction() {
7         return $this->render('AcmeDemoBundle:Welcome:index.html.twig');
8     }
9 }
```

- Un nouveau contrôleur s'écrit facilement en dérivant la classe `Controller` fournie par Symfony.
- la méthode `render()` fait appel à la vue.

Le cadriceiel Symfony

Les controleurs

- un (meilleur) exemple de contrôleur :

```
1 // src/Acme/DemoBundle/Controller/WelcomeController.php
2 namespace Acme\DemoBundle\Controller;
3 use Symfony\Bundle\FrameworkBundle\Controller\Controller
4 ;
5 class WelcomeController extends Controller {
6     public function indexAction() {
7         return $this->render('AcmeDemoBundle:Welcome:index.
8             html.twig');
9     }
10 }
```

- Un nouveau contrôleur s'écrit facilement en dérivant la classe `Controller` fournie par Symfony.
- la méthode `render()` fait appel à la vue qui construit l'objet `Response`.

Le cadriciel Symfony

Les vues

- Dans Symfony, les vues sont construites par un moteur de templates (*Twig* invoqué par la méthode `render`).
- Celui-ci charge un template de réponse dans lequel il injecte les paramètres (optionnels) passés à `render ()` .
- Le template est un document HTML dans lequel les points d'insertion des données sont marqués dans un langage similaire à l'*expression language* de JSP.
- Les templates peuvent être étendus par héritage.

Le cadriceil Symfony

Les vues

- Exemple de template :

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>My Webpage</title>
5   </head>
6
7   <body>
8     <h1>{{ page_title }}</h1>
9
10    <ul id="navigation">
11      {% for item in navigation %}
12        <li><a href="{{ _item.href }}">{{ item.caption }}</a>
13          </li>
14      {% endfor %}
15    </ul>
16  </body>
17 </html>
```


Quelques références

- Manuel de PHP 
- Validation et protection des données 