POLYTECH TOURS
Département Informatique

PROGRAMMATION WEB DI3

PHP

Cyrille FAUCHEUX cyrille.faucheux@etu.univ-tours.fr

Année 2010-2011

POLYTECH TOURS
Département Informatique

2 Historique

Historique

- 1969 : invention d'internet
  - Permet d'échanger des données de façon fiable
    - Protocoles propriétaires/spécifiques
    - Serveurs propriétaires
    - Clients propriétaires
      - Tout ça Cobol/Ada/Fortran/C/…

Historique

- 1989 : invention du Web
  - On sait comment transférer et afficher des données de façon standardisée
    - Protocole standardisé (HTTP)
    - Clients standardisés (navigateurs + HTML)
    - Serveurs spécifiques
      - C/Perl

Historique

- 1993 : Le NCSA (National Center for Supercomputing Applications) développe un serveur web disposant d'une interface permettant de faire appel à d'autre programmes
  - Gestion des requêtes HTTP
  - Permet de déléguer la génération des pages à des « modules » écrits dans d'autres langages plus adaptés/moins contraignants (ex. Perl pour le traitement des formulaires)
  - Cette interface sera normalisée en 2004 : CGI (Common Gateway Interface)
  - Ce serveur sera à la base d'Apache

Historique

- 1994 : Rasmus Lerdorf veut avoir des stats sur son CV en ligne
  - Perl : langage à tout faire
    - Possibilité d'écrire un langage plus adapté.
- 1995 : PHP/FI v1.0
  - Scripts Perl permettant d'interpréter un langage ressemblant au Perl,
  - Puis réécriture du parseur en C.
- 1997 : PHP/FI v2.0
  - Nouvelle réécriture en C,
  - 50 000 domaines l'utilise.

Historique

- 1998 : PHP v3.0
  - Nouvelle réécriture : « PHP d'aujourd'hui »,
  - Rasmus n'est plus le « seul » développeur !
- 1999 : PHP v4.0
  - Nouvelle réécriture : + performant, + modulaire.
- 2004 : PHP v5.0
  - Programmation orientée objet
- Aujourd'hui : PHP v5.3

Le PHP

- Licence
- Cas d'utilisation
- Fonctionnement

Le PHP

- Initialement
  - ***P**ersonal **H**ome **P**age/**F**orm **I**nterpreter*
- Maintenant
  - ***P**HP: **H**ypertext **P**rocessor*
- Licence *Open Source*
- Fonctionne sur toutes les plateformes

Le PHP

- PHP est un langage de script
  - Langage qui est interprété sans être compilé
  - Domaines d'utilisation
    - Sites web : la plus courante,
    - En ligne de commande (CLI) : écriture de scripts (maintenance, administration, …),
    - Programmes : création d'interfaces graphiques.
  - Modes d'exécution
    - Mode interprété (pour sites web),
    - Mode précompilé (extension, pour les sites web),
    - Mode compilé (programmes).

Le PHP

- Fonctionnement
  - L'internaute demande une ressource,
  - Le serveur Web (ex : Apache), décode la requête et identifie le script responsable de la génération de la page,
  - La main est passée à l'interpréteur PHP qui exécute le script
    - Interrogation de bases de données.
  - Le serveur web récupère tout ce qui a été écrit sur la sortie standard → page HTML,
  - Cette page HTML est renvoyée au client.

12

Outils nécessaires

Outils nécessaires (pour du dev Web)

- Serveur HTTP (ex. Apache),
- Interpréteur PHP,
- Module permettant au serveur HTTP d'utiliser l'interpréteur PHP (ex. libapache2-mod-php5),
- Les librairies dont vous aurez besoin
  - Ex. php5-mysql

Outils nécessaires (pour du dev Web)

- Installation
  - Installer et configurer tous les composants
    - Facile
  - Serveur tout en un (pas pour de la production)
    - EasyPHP
    - {W, L, M}amp
      - Windows/Linux/Mac Apache MySql PHP
    - {W, L, M}app
      - Windows/Linux/Mac Apache PostgreSQL PHP
    - …

Outils nécessaires

- Environnement de développement
  - PhpDesigner,
  - NetBeans,
  - Eclipse,
  - PHPEdit,
  - Notepad++,
  - …
- Outils annexes
  - FirePHP,
  - Xdebug,
  - …

POLYTECH TOURS
Département Informatique

16

Syntaxe du langage PHP

Syntaxe

- L'interpréteur lit un fichier PHP puis génère un flux en respectant les règles suivantes :
  - Un bloc de code PHP est un groupe continu de lignes encadrés par deux balises `<?php` et `?>`,
  - Toute ligne située à l'extérieur de ces balises sera envoyée telle quelle dans le flux de sortie,
  - Toute ligne située à l'intérieur de ces balises est considérée comme une instruction PHP et sera interprétée afin d'écrire dans le flux de sortie,
  - Les instructions PHP n'apparaissent pas dans le flux généré.

Syntaxe

- Le code HTML est en général écrit hors des balises de PHP,
- Les balises PHP peuvent intervenir à tout moment dans du code HTML.

Syntaxe

- Exemple

```
<!DOCTYPE html>
<html>
   <head>
        <title>Welcome to my website.</title>
   </head>
   <body>
        <h1>My article</h1>
        <p>Hello</p><br />
        <p><?php echo "World"; ?></p>
   </body>
</html>
```

Syntaxe

- Syntaxe proche du C
  - Commande PHP : se termine par un « ; »

```
echo "Hello World";
```

  - Commentaires (non visible par le navigateur)

```
/* … */ (multiligne)
// …
```

POLYTECH
TOURS
Département Informatique

21

Les variables

Les variables

- Les variables portent toutes le préfixe « $ »,
- Composée de caractères alphanumériques ou « _ » (sauf pour le premier caractère qui ne peux pas être un chiffre).
  - `$foo = "Hello world !";`
- Non typées
  - `$bar = 42;`
- Types disponibles
    - Int, float, boolean, string, array, object.

La portée des variables

- Déclarée dans le corps du script : portée globale

```
<?php
  $a = 1;
  …
```

- Dans le corps d'une fonction : portée **locale**

```
function test()
{
  echo $a; // Erreur, portée locale, $a
  n'existe pas
}
```

La portée des variables

- Mot clé `global`

```
<?php
$a = 42;

function test() {
  global $a; // Indique que l'on fait
  référence à la variable globale $a
  echo $a; // OK
}
…
```

La portée des variables

- Mot clé `static`

```
function incrementCount() {
  static $count = 0;
  return ++$count;
}
```

Les variables

- Changement de types (*cast*)
  - Il est possible de convertir une variable en un type prédéfini grâce au *cast*.
  - Exemple :
    - `$str = "12";`
    - `$nb = (int)$str;`

Les variables

- Quelques fonctions utiles
  - `empty($var)` : renvoie vrai si la variable est vide (`0`, `NULL`, `""`, `[]`)
  - `isset($var)` : renvoie vrai si la variable existe (contient autre chose que `NULL`)
  - `unset($var)` : détruit la variable (existence d'un *Garbage Collector*)
  - `gettype($var)` : renvoie le type de la variable
  - `is_long()`, `is_double()`, `is_int()`, `is_float()`, `is_numeric()`, …

Les variables

- Les constantes
  - On utilise l'instruction `define`

```
define("APP_VERSION", "1.2.3");
if(defined("APP_VERSION")){
  print("La version de l'application est "
  .APP_VERSION);
}
```

  - Remarques:
    - On accède aux constantes par leur identificateurs sans $,
    - Php sensible à la casse pour les fonctions et les variables mais pas pour les constantes,
    - Operateur de concaténation « . ».

Les variables variables


```php
<?php
    $a = "hello";
    $$a = "world"; // $hello = "world"
    …
```

Les variables superglobales

- Définies par le serveur (mot clé `global` inutile)
  - $_GET : données issues d'un formulaire
  - $_POST : données issues d'un formulaire
  - $_COOKIE : cookie de l'utilisateur
  - $_REQUEST : contient $_GET, $_POST, $_COOKIE
  - $_ENV : environnement dans lequel est exécuté le script
  - $_SESSION : données de session associées à l'utilisateur
  - $_FILES : données issues d'un envoi de fichier depuis un formulaire.

Les constantes prédéfinies

Name	Description
__LINE__	Le numéro de ligne dans le fichier courant.
__FILE__	Le chemin d'accès au fichier courant.
__DIR__	Chemin d'accès au dossier contenant le fichier courant.
__FUNCTION__	Le nom de la fonction en cours d'exécution.
__CLASS__	Le nom de la classe utilisée.
__METHOD__	Le nom de la méthode en cours d'exécution.
__NAMESPACE__	L'espace de nom actuel.

Les chaines de caractères

- Guillemets doubles
  - `$s = "My name is \"$name\".";`
    - Interprète les caractères spéciaux et les variables.
- Guillemets simples
  - `$s = 'My name is \"$name\".'; //KO`
    - N'interprète pas les caractères spéciaux (ex. « \n ») ni les variables.
- Syntaxe HEREDOC
  - Permet de définit une constante qui délimite un paragraphe, pas besoin de guillemets

```
echo <<<EOT
  My name is "$name".
EOT;
```

Les chaines de caractères

- Concaténation de chaines
  - `$var = "Hello " .$world;`
  - `$var = "Hello ";`
  - `$var .= "world !";`

Les tableaux/*hash*

- Déclaration

```
$tab = Array();
$tab = Array(1, 2, 3);
$tab = Array(1, "deux", 3);
$tab = Array("prenom" => "Bob", "Nom" =>
  "Léponge");
```

- Insertion dans un tableau
  - `$tab[3] = "Joe";`
  - `$tab[] = " Joe"; // Insertion à la fin`
  - `$tab["nom"] = "Lindien";`

Les structures de contrôle

- If
- Switch
- For
- While
- …

Les structures de contrôle

- If then else

```
if() {
  …
} elseif {
  …
} else {
  …
}
```

Les structures de contrôle

- Switch
  - Compare des entier, strings, …

```
switch($variable){
  case 1:
    …
    break;
  case 2:
    …
    break;
  default:
    …
    break;
}
```

Les structures de contrôle

- For/foreach

```
for(var $i = 0; $i < 5; $i++) {
  ...
}

foreach($tableau as $membre ) {
  ...
}
foreach($tableau as $key => $value ) {
  ...
}
```

Les structures de contrôle

- While/do… while

```
while(condition) {

}

do {

} while(condition);
```

Les structures de contrôle

- Nouvelle syntaxe

```
for(var $i = 0; $i < 5; $i++):
  ...
endfor;

if():
  …
else:
  …
endif;
```

Les fonctions

Les fonctions

- Déclaration

```
<?php
  function printHello() {
      echo "Hello world !";
  }
  function hello($world) {
      echo "hello" .$world;
  }
```

- Appel d'une fonction

```
  hello("world");
```

Les fonctions

- Passage de paramètre par référence

```
<?php
  function increment(&$nb) {
     ++$nb;
  }

  $i = 0;
  increment($i);
  increment($i);
  echo $i; // Affiche 2
```

Les fonctions

- Retourner une valeur

```
function hello($world) {
    return "hello" .$world;
}
```

Les fonctions

- Attention, certaines fonctions ne sont pas « vraiment » des fonctions !
  - Ex. `echo "Hello world";`
  - Mais `echo("Hello world");` fonctionne aussi.
  - Idem pour `print`.

Les opérateurs

Les opérateurs

- Les opérateurs arithmétiques (`$y = 5`)

Opérateur	Description	Exemple	Resultat
+	Addition	`$x = $y+2`	`$x = 7`
-	Soustraction	`$x = $y-2`	`$x = 3`
*	Multiplication	`$x = $y*2`	`$x = 10`
/	Division	`$x = $y/2`	`$x = 2.5`
%	Modulo	`$x = $y%2`	`$x = 1`
++	Pré-incrémentation	`$x = ++$y`	`$x = 6 ($y = 6)`
++	Post-incrémentation	`$x = $y++`	`$x = 5 ($y = 6)`
--	Pré-incrémentation	`$x = --$y`	`$x = 4 ($y = 4)`
--	Post-incrémentation	`$x = $y--`	`$x = 5 ($y = 4)`

Les opérateurs

- Les opérateurs d'assignement (`$x=10, $y=5`)

Opérateur	Exemple	Equivalent à	Résultat
=	`$x=$y`		`$x = 5`
+=	`$x+=$y`	`$x = $x+$y`	`$x = 15`
-=	`$x-=$y`	`$x = $x-$y`	`$x = 5`
*=	`$x*=$y`	`$x = $x*$y`	`$x = 50`
/=	`$x/=$y`	`$x = $x/$y`	`$x = 2`
%=	`$x%=$y`	`$x = $x%$y`	`$x = 0`

Les opérateurs

- Les opérateurs de comparaison (`$x = 5`)

Opérateur	Description	Exemple
`==`	Egalité	`$x == 5 est vrai` `$x == "5" est vrai`
`===`	Egalité et type	`$x === 5 est vrai` `$x === "5" est faux`
`!=`	Différence	`$x != 8 est vrai`
`>`	Supérieur	`$x > 8 est faux`
`<`	Inférieur	`$x < 8 est vrai`
`>=`	Supérieur ou égal	`$x >= 8 est faux`
`<=`	Inférieur ou égal	`$x <= 8 est vrai`

Syntaxe

- Les opérateurs logiques (`$x=6, $y=3`)

Opérateur	Description	Exemple
`&&`	ET logique	`($x < 10 && $y > 1) vrai`
`and`	ET logique	`($x < 10 and $y > 1) vrai`
`||`	OU logique	`($x==5 || $y==5) est faux`
`or`	OU logique	`($x==5 or $y==5) est faux`
`!`	NON logique	`! (x==y) est vrai`

- « && » et « || » sont prioritaires sur « and » et « or ».

Syntaxe

- Les opérateurs binaires ($x = 0b00000010)

Opérateur	Description	Exemple
<<	Décalage à gauche	$x << 1 vaut 0b00000100
>>	Décalage à droite	$x >> 1 vaut 0b00000001
&	ET binaire	$x & 0b00000011 vaut 0b00000010
|	OU binaire	$x | 0b00000001 vaut 0b00000011
^	XOR binaire	$x ^ 0b00000011 vaut 0b00000001
~	NON binaire	~$x vaut 0b11111111111111111111111111111101

Les exceptions

Les exceptions

- Une exception peut être un entier, une chaîne ou un objet.
- Lancer une exception : `throw "Erreur";`
- Attraper une exception :

```
try {
    …
}
catch($err) {
    …
}
```

Inclusion de fichiers

Inclusion de fichiers

- `include("config.php");`
  - Evalue le fichier
- `require("config.php");`
  - Idem, mais renvoie une erreur si le fichier n'existe pas.
- `include_once(), require_once()`
  - N'inclue qu'une fois un fichier si plusieurs appels sont faits.

Les sessions

Les sessions

- Permettent de garder en mémoire des informations relatives à l'utilisateur
- Appel à `session_start()` (avant toute écriture sur la sortie standard)
  - Un identifiant de session est généré est stocké dans le cookie.
  - Cet identifiant sera automatiquement récupéré à chaque appel de cette fonction
- Appel à `session_destroy()`
  - Supprime les données et l'identifiant de session

POLYTECH TOURS
Département Informatique

58

La programmation orientée objet

La POO

Méthode	Description
`class`	Déclaration de classe
`interface`	Déclaration d'une interface
`const`	Déclaration de constante de classe
`function`	Déclaration d'une méthode
`public/protected/private`	Accès (par défaut « public »)
`new`	Création d'un objet
`self`	Désigne la classe elle-même
`parent`	Désigne la classe parent
`static`	Appel statique
`extends`	Héritage de classe
`implements`	Implémentation d'une interface
`$this`	L'instance en cours
`__construct`	Nom du constructeur

La POO

- Définition d'une classe

```
class MyClass {


}
```

- Instanciation d'une classe

```
$inst = new MyClass();
```

La POO

- Les attributs

```
class MyClass {
  private $myAttr1;
  private $myAttr2;
}
```

- Constante et variable de classe

```
class MyClass {
  const $myConst = 42;
  static $myStaticVar;
}
```

La POO

- Portée des éléments
  - `private` : accessible seulement de l'intérieur de la classe,
  - `public` : Accessible aussi depuis l'extérieur de la classe,
  - `protected` : Accessible depuis les classe héritées.

La POO

- Les méthodes

```
class MyClass {
  private function myMethod() {

  }
}
```

- Appel d'une méthode

```
$inst = new MyClass();
$inst->myMethod();
```

La POO

- Méthode statique

```
class MyClass {
  public static function myStaticMethod() {

  }
}
```

- Appel d'une méthode statique

```
MyClass::myStaticMethod();
```

La POO

- Le constructeur et le destructeur

```
class MyClass {
  function __construct() {
  }
  function __destruct() {
  }
}
```

- Ils ne renvoient rien !
- Le destructeur ne prend pas de paramètre.

La POO

- L'héritage

```
class foo {

}
class bar extends foo {

}
```

- La classe `bar` a accès aux fonctionnalités `public` et `protected` définies dans `foo`.

La POO

- Résolution de la portée
  - Accès aux méthodes/attributs du parent
    - `$this` définit l'instance actuelle de l'objet (méthodes et attributs),
    - `parent` définie la classe parente (méthodes statiques et constantes),
    - `self` définie la classe actuelle.
    - Accès à une élément du parent
      - `$parent::$myAttr;`
      - `$parent::myFunc();`
    - Penser a appeler le constructeur(resp destructeur) du parent dans le constructeur (resp.) de la classe qui hérite.

La POO

- Les interface
  - Une interface définie un ensemble de méthodes que doit **implémenter** une classe

```
interface MyInterface {
  public function myFunc();
}
class MyClass implements MyInterface {
  public function myFunc() {
  …
   }
}
```

La POO

- Les classes abstraites
  - Similaire aux interface,
  - Déclarée grâce au mot clé `abstract`,
  - Les méthodes non implémentées doivent être déclarées avec le mot clés `abstract`.
  - Différence : une classe qui hérite d'une chasse abstraite est aussi du type de cette classe. Ce n'est pas le cas pour une interface.

Les bases de données

Les bases de données

- Permet de stocker des données structurées
- Volume très important
- Interrogation de la base de données grâce au langage SQL
- Différents types de bases de données : MySQL, SQL SERVER, …

Les bases de données

- L'accès à une base de données est composé de plusieurs étapes :
  - Ouverture de la connexion au serveur,
  - Choix de la BDD sur le serveur (parfois au moment de l'ouverture de la connexion),
  - Préparations de requêtes,
  - Exécution des requêtes et récupérations des résultats,
  - Déconnexion de la base.

Les bases de données

- Pour se connecter à une bd il faut : le nom du serveur, le login/mot de passe, le nom de la base.
- Fonctions de connexion :
  - `mysql_connect($server,$user,$password)` : permet de se connecter au serveur $server en tant qu'utilisateur $user avec le mot de passe $password, retourne l'identifiant de connexion si succès, FALSE sinon,
  - `mysql_select_db($base[,$id])` : permet de choisir la base $base, retourne TRUE en cas de succès, sinon FALSE,
  - `mysql_close([$id])` : permet de fermer la connexion,
  - `mysql_pconnect` : idem que `mysql_connect` sauf que la connexion est persistante, il n'y a donc pas besoin de rouvrir la connexion à chaque script qui travaille sur la même base.
- Les identifiants de connexions ne sont pas indispensables, ils permettent juste de lever toutes ambigüités (ex. interrogation de plusieurs bases dans le même script).

Les bases de données

- Exemple de connexion

```
if( $id = mysql_connect("localhost", "foobar", "0478"))
{
   if($id_db = mysql_select_db("gigabase")){
      echo "Succès de connexion.";
      /* code du script … */
   } else {
      die("Echec de connexion à la base.");
   }
   mysql_close($id);
} else {
   die("Echec de connexion au serveur de base de
     données.");
}
```

Les bases de données

- Pour envoyer une requête à une base de données, il existe la fonction : `mysql_query($str)` qui prend pour paramètre une chaîne de caractères qui contient la requête écrite en SQL et retourne un identificateur de résultat ou FALSE si échec

- Exemple :

```
$result = mysql_query("SELECT address FROM users WHERE
    name=\"$name\"");
```

- Cet exemple recherche l'adresse de l'utilisateur portant pour nom la valeur de la chaîne `$name`. L'identificateur de résultat $result permettra à d'autres fonctions d'extraire ligne par ligne les données retournées par le serveur

Les bases de données

- Extraction des données :
  - `mysql_fetch_row($result)` : retourne une ligne de résultat sous la forme d'un tableau. Les éléments du tableau étant les valeurs des attributs de la ligne. Retourne FALSE s'il n'y a plus aucune ligne.
- Exemple :

```
$request = "SELECT * FROM users";
if($result = mysql_query($request)) {
   while($ligne = mysql_fetch_row($result)) {
      $id = $ligne[0];
      $name = $ligne[1];
      $address = $ligne[2];
      echo "$id - $name, $address <br />";
   }
} else {
   echo "Erreur de requête de base de données.";
}
```

- Ici, on accède aux valeurs de la ligne par leur indice dans le tableau

Les bases de données

- Extraction des données :
  - `mysql_fetch_array($result)` : retourne un tableau associatif. Les clés étant les noms des attributs et leurs valeurs associées leurs valeurs respectives. Retourne FALSE s'il n'y a plus aucune ligne.
- Exemple :

```
$request = "SELECT * FROM users";
if($result = mysql_query($request)) {
   while($ligne = mysql_fetch_array($result)) {
      $id = $ligne["id"];
      $name = $ligne["name"];
      $address = $ligne["address"];
      echo "$id - $name, $address <br />";
   }
} else {
   echo "Erreur de requête de base de données.";
}
```

- Ici, on accède aux valeurs de la ligne par l'attribut dans le tableau associatif

Les bases de données

- PDO est une API générique (couche d'abstraction)
  - Simplicité lors d'un changement de SGBD
  - Changement d'un seul paramètre
  - Reste du code identique
  - Pas nécessaire d'apprendre le fonctionnement de chaque API spécifique

Le fichier php.ini

Configuration par le fichier php.ini

- Comportement de PHP dicté par sa configuration
- Habituellement placé dans le même répertoire que PHP
- Modifiable à partir de tout éditeur de texte

Configuration par le fichier php.ini

- disable_functions
  - Désactive certaines fonctions dans les scripts PHP
- disable_classes
  - Idem mais pour les classes
- max_execution_time
  - Temps d'exécution maximum (30 sec par défaut)
  - set_time_limit()
    - Modifie cette option pendant l'exécution d'un script

Configuration par le fichier php.ini

- error_reporting
  - Définit le niveau d'erreur qui doit être filtré par PHP (fichiers de log)
- display_errors
  - affichage des erreurs dans la sortie standard
- post_max_size
  - Taille maximum des données que PHP accepte depuis un formulaire POST

Configuration par le fichier php.ini

- upload_max_filesize
  - Taille maximum d'un fichier depuis un formulaire POST
- allow_url_fopen
  - Utiliser des fichiers sur d'autres serveur WEB
  - Attention à traiter ce qu'ils contiennent
- allow_url_include
  - Permet d'inclure du code provenant d'un autre serveur
  - Déconseillé car dangereux et gourmand

Configuration par le fichier php.ini

- Fonctions mail
  - Windows
    - SMTP
      - Adresse du serveur SMTP à utiliser
    - smtp_port (25 par défaut)
    - sendmail_from (facultatif)
      - Adresse par défaut en tant qu'émetteur des e-mails
  - UNIX
    - Sendmail_path
      - chemin jusqu'au programme d'envoi d'e-mails

Les alternatives

Les alternatives

- ASP.Net
- Ruby/Ruby on Rails
- Python/Django
- Perl
- Java/JEE
- …

Conclusion

Conclusion

- PHP est un langage de script encore d'actualité
- Permet la réalisation d'applications complexes
- Excellent choix face à d'autres technologies

- Manuel PHP : http://www.php.net/manual/en/manual.php