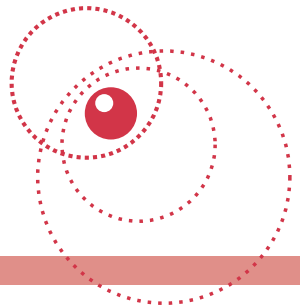

Développement Web (INF228)

Introduction à PHP

19 janvier 2015

Pierre Senellart

TELECOM ParisTech

Plan

PHP et HTML

- Script PHP : document HTML (par exemple), dans lequel est incorporé du code PHP.
- Le code PHP est à l'intérieur d'une pseudo-balise `<?php ... ?>` (ou `<? ... ?>` , ou `<?= ... ?>` qui est un raccourci pour `<? echo ... ?>`).

Exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

<html>
  ...
  <body>
  <h1><?php echo 2+2; ?></h1>
</body>
</html>
```

Syntaxe

- Syntaxe similaire à (et inspirée de) C, Java :
  - instructions séparées de points-virgules
  - instructions de base : `if` , `while` , `for` ...
  - opérateurs de base : `=` , `==` , `++` , `<=` ...
  - mêmes commentaires : `//` , `/* */` , mais `#` fonctionne également
  - littéraux similaires (mais voir plus loin l'interpolation)
  - etc.
- Mais des différences importantes :
  - nouvelles instructions (e.g., `foreach`), nouveaux opérateurs (`.` est la concaténation de chaîne)
  - variables non typées, non déclarées, de portée (par défaut) la fonction toute entière
  - variables toujours préfixées d'un signe « $ »

Interpolation

Les chaînes de caractère en PHP peuvent être :

- entourées par des apostrophes : `'Hello world'` . Elles se comportent comme les chaînes de caractère Java, mais les séquences d'échappement (`\n` . . .) ne sont pas disponibles.
- entourées par des guillemets doubles : `"Hello\n$world"` . Les séquences d'echappement sont autorisées, et les noms de variables sont interpolés (remplacés par leur valeur) En cas de valeur complexe, utiliser la syntaxe `{$var}` :

  `"La valeur est {$matrix[$i][$j]}"`

- Un tableau peut contenir des valeurs de types différents : entier, chaîne de caractères, etc.

Exemple

```
$tab[0] = "1er element";
$tab[1] = "2e element";
$tab[2] = 120;
```

- En PHP, l'affectation d'un indice à un nouvel élément est automatique : il correspond à la première cellule vide.

Exemple

```
$tab2[] = "1er element";
$tab2[] = "2e element";
$tab2[] = 120;
```

- L'initialisation d'un tableau peut également se faire à l'aide de la fonction `array`

Exemple

```
$tab3 = array("1er element","2e element",120);
```

- Les indices numériques sont remplacés par des chaînes de caractères appelées clefs.
- Pour un tableau donné, toutes les clefs doivent être différentes.
- La fonction `array` peut également être utilisée pour intialiser le tableau.

Exemple

```
$prenom["Belmondo"] = "Jean-Paul";
$prenom["Delon"] = "Alain";
$prenom["Deneuve"] = "Catherine";
$prenom = array("Balasko" => "Josiane", "Bourvil" => "");
```


- Le parcours d'un tableau associatif est plus complexe que celui des tableaux indicés : on ne peut pas se baser sur l'ordre des indices pour effectuer une boucle simple.
- On peut utiliser un curseur sur le tableau : sorte de flèche indiquant l'élément du tableau actuellement visé.
- Les fonctions `next` et `prev` permettent de déplacer le curseur initialement positionné sur le premier élément du tableau.
- Les fonctions `key` et `current` renvoient respectivement la clef et la valeur de l'élément courant.

Exemple (Afficher toutes les personnes de la liste prenom)

```
echo "Nom=".key($prenom).
     " Prenom=".current($prenom)."\n";

while(next($prenom)) {
  echo "Nom=".key($prenom)." Prenom=".current($prenom);
}
```

- Il est cependant plus facile de parcourir un tableau associatif à l'aide de l'instruction `foreach`.

Exemple

```
foreach ($prenom as $cle => $valeur) {
  echo "Nom=$cle Prenom=$valeur\n";
}
```

- Cette instruction peut également être utilisée pour des tableaux indicés.

Exemple

```
foreach ($tableau as $valeur) {
  ...
}
```

- Pour faciliter la visibilité d'un script PHP et pour rendre automatique l'exécution de certaines tâches répétitives, on peut définir des fonctions.
- La construction `list ( ... )` peut être utilisée pour récupérer dans plusieurs variables différentes une valeur de retour qui est un tableau.

Exemple

```
function Kenshin() {
  return array ("Kenshin Le Vagabond", "Nobuhiro Watsuki", 28);
}

function Addition($x,$y) {
  $somme = $x+$y;
  return $somme;
}

$z=Addition($x,$y);
list ($a,$b,$c)=Kenshin();
```

Mais aussi...

- Langage très (trop ?) riche
- Programmation orientées objet : `class`, `$object->field`... souvent utilisée par les frameworks PHP
- Très large bibliothèque de fonction, avec de nombreuses bibliothèques externes disponible
- Astuce : `http://php.net/fonction` donne la documentation de la fonction PHP *fonction*

Validation

- Un script PHP avec du code PHP incorporé au sein de code HTML n'est pas un document XHTML valide !
- Ce qu'il faut valider, c'est une (des) page(s) HTML produites par le script.
- Possibilité d'indiquer une URL au validateur du W3C. Malheureusement pas utilisable quand l'URL est privée. Possibilité dans ce cas de sauvegarder le fichier HTML produit, et de l'envoyer au validateur du W3C comme fichier local.
- Il n'y a pas de « validateur » PHP, mais les erreurs de syntaxe causeront des erreurs à l'exécution du script.

Plan

$_GET et $_POST

- Les paramètres HTTP peuvent être récupérées en PHP grâce au tableaux associatifs `$_GET` et `$_POST`.
- Les valeurs de ce tableau peuvent être des variables simples ou des tableaux indicés : ces derniers sont les paramètres à choix multiples dont on a suffixé le nom de `[]` dans le code HTML.

Exemple

```
echo "<p>Votre login est: " . $_POST["login"] . "</p>";
echo "<p>Vous avez coché les genres: ";
for($i=1;$i<=count($_POST['genre']);$i=$i+1) {
  echo $_POST['genre'][$i] . " ";
}
echo "</p>";
```

Les informations relatives aux fichiers transférés sont disponibles dans un tableau associatif $_FILES :

- les clés sont les noms des champs de formulaire d'où provient le fichier
- las valeurs sont des ensembles de propriétés (décrits comme des tableaux associatifs) décrivant le fichier reçu par le serveur auxquelles s'ajoute la propriété *error* qui permet de savoir si le transfert s'est bien déroulé

Exemple (dans un fichier *FormTransfert.html*)

```
<form enctype="multipart/form-data"
  action="TransfertFichier.php" method="post">
  ...
  <div>
    <label for="maPhoto">Choisissez un fichier :</label>
    <input type="file" name="maPhoto" id="maPhoto" />
  </div>
  ...
</form>
```

name	est le nom du fichier sur la machine du client
tmp_name	est le nom du fichier temporaire sur la machine du serveur
size	est la taille du fichier, en octets
type	est le type MIME du fichier, par exemple « image/gif »

Exemple (dans le fichier *TransfertFichier.php*)

```
$fichier=$_FILES['maPhoto'];
echo "Nom fichier client: ".$fichier['name']."<br />";
echo "Nom fichier serveur: ".$fichier['tmp_name']."<br />";
echo "Taille du fichier: ".$fichier['size']."<br />";
echo "Type du fichier: ".$fichier['type']."<br />";
```

name	est le nom du fichier sur la machine du client
tmp_name	est le nom du fichier temporaire sur la machine du serveur
size	est la taille du fichier, en octets
type	est le type MIME du fichier, par exemple « image/gif »

Exemple (dans le fichier *TransfertFichier.php*)

```
$fichier=$_FILES['maPhoto'];
echo "Nom fichier client: ".$fichier['name']."<br />";
echo "Nom fichier serveur: ".$fichier['tmp_name']."<br />";
echo "Taille du fichier: ".$fichier['size']."<br />";
echo "Type du fichier: ".$fichier['type']."<br />";
```

UPLOAD_ERR_OK pas d'erreur, le transfert s'est bien passé

UPLOAD_ERR_INI_SIZE le fichier transmis dépasse la taille maximale autorisée

UPLOAD_ERR_PARTIAL le fichier est transféré seulement partiellement

UPLOAD_ERR_NO_FILE aucun fichier n'a été transféré

Exemple

```
$codeErreur = $_FILES['maPhoto']['error'];

if($codeErreur!=UPLOAD_ERR_OK) {
  echo "<p>Erreur lors du transfert du fichier.</p>";
}
```

- La fonction PHP `copy($source,$destination)` permet de copier le fichier *source* vers *destination*. Important parce que le fichier temporaire pourra être détruit à la fin du script !
- Attention : le programme doit avoir les droits d'accès et d'écriture sur les répertoires dans lesquels les fichiers sont copiés.

Exemple

```
// Copie du fichier dans le répertoire PHOTOS
copy($_FILES['toto']['tmp_name'],"./PHOTOS/$id.jpg");
```

Plan

session : ensemble d'informations conservées tout au long d'une interaction avec les différentes pages d'un site Web

authentification : mécanisme permettant d'associer un identifiant (login) à un utilisateur d'un site Web, de manière à permettre de la personnalisation du contenu, ou de la gestion de droits sur une application Web ; habituellement, l'authentification est faite grâce à un mot de passe.

- Méthode d'autentification HTTP simple disponible au niveau du serveur Web, mais impose une modification de la configuration de celui-ci ; un peu lourd, difficilement connectable à un SGBD.
- Pas de gestion de session à proprement parler en HTTP. Alternatives :
  - Paramètres HTTP cachés dans l'URL (méthode GET). Mécanisme un peu lourd, puisque tous les liens doivent être changés pour incorporer ces paramètres.
  - Cookies.

- Informations, sous la forme de clés/valeurs, qu'un serveur Web demande à un client Web de conserver et de retransmettre à chaque requête HTTP.
- `setcookie($name,$value)` : demande client de stocker un cookie de nom $name et de valeur $value.
- `$_COOKIE` est un tableau associatif des cookies que le client a envoyé au serveur Web.

PHP fournit une abstraction de la gestion de session (utilisant un cookie, mais sans avoir à le gérer soi-même).

session_start() ouvre une session en cours, ou crée une nouvelle session s'il n'y a pas de session ouverte ; ceci est à placer au tout début du script PHP, ou en tous cas avant que quoi que ce soit n'ait été écrit dans la page, de manière à pouvoir modifier les en-têtes de la réponse HTTP.

session_destroy() termine la session en cours.

session_id() fournit un identifiant de la session en cours.

$_SESSION contient l'ensemble des paramètres de session (tableau associatif clé/valeur), disponibles dans les différentes pages Web de la même session.

- Un formulaire demande login et mot de passe.
- Un script de traitement de ce formulaire, contrôle que le login et le mot de passe sont corrects (par exemple à l'aide d'une table MySQL) :
  - Si c'est le cas, crée une session PHP (`session_start();`), y ajoute un paramètre nommé par exemple valid_user (`$_SESSION['valid_user']=1;`) et redirige vers une autre page.
  - Sinon, redirige vers la page de formulaire.
- Les autres pages (pages auxquelles les utilisateurs authentifiés et seulement eux ont accès) commençent par un `session_start();` et contrôlent si l'utilisateur est identifié (`if($_SESSION['valid_user']==1) { ... }`) et sinon redirigent vers la page de formulaire
- Une page de déconnexion appelle `session_destroy();`

Plan

Connecteurs pour SGBD

- Connecteurs (bibliothèques de fonctions) différentes pour les différents SGBD existants : MySQL, PostgreSQL, Oracle, etc.
- Pour MySQL, trois connecteurs existants : mysqli, PDO, mysql (déprécié)
- Possibilité de récupérer chaque ligne de résultat comme des tableaux associatifs, des tableaux nommés, ou des objets (voir doc)
- Possible d'utiliser une surcouche d'un framework (p. ex., ActiveRecord dans CodeIgniter)
- Ou d'utiliser un ORM complet (p. ex., Doctrine)

```php
<?php
// mysqli
$mysqli = new mysqli("example.com", "user", "password", "database");
$result = $mysqli->query("SELECT 'Hello, dear MySQL user!' AS _message
FROM DUAL");
$row = $result->fetch_assoc();
echo htmlentities($row['_message']);

// PDO
$pdo = new PDO('mysql:host=example.com;dbname=database', 'user',
'password');
$statement = $pdo->query("SELECT 'Hello, dear MySQL user!' AS _message
FROM DUAL");
$row = $statement->fetch(PDO::FETCH_ASSOC);
echo htmlentities($row['_message']);

// mysql
$c = mysql_connect("example.com", "user", "password");
mysql_select_db("database");
$result = mysql_query("SELECT 'Hello, dear MySQL user!' AS _message
FROM DUAL");
$row = mysql_fetch_assoc($result);
echo htmlentities($row['_message']);
?>
```

ActiveRecords dans CodeIgniter

```
 $this->db->select('*');
 $this->db->from('blogs');
 $this->db->join('comments', 'comments.id = blogs.id');
 $query = $this->db->get();

foreach ($query->result() as $row)
{
  echo $row->title;
}
```

cf. http://ellislab.com/codeigniter/user-guide/database/active_record.html

Déclaration d'objet dans l'ORM Doctrine

```php
<?php
/**
 * @Entity @Table(name="products")
 **/
class Product
{
    /** @Id @Column(type="integer") @GeneratedValue **/
    protected $id;
    /** @Column(type="string") **/
    protected $name;

    public function getId()
    { return $this->id; }
    public function getName()
    { return $this->name; }
    public function setName($name)
    { $this->name = $name; }
}
```

Utilisation d'objet dans l'ORM Doctrine

```
require_once "bootstrap.php";

$product = new Product();
$product->setName("toto");
$entityManager->persist($product);
$entityManager->flush();

$productRepository = $entityManager->getRepository('Product');
$products = $productRepository->findAll();

foreach ($products as $product) {
    echo sprintf("<li>%s</li>",
      htmlspecialchars($product->getName()));
}
```

Licence de droits d'usage

Contexte public } avec modifications

Par le téléchargement ou la consultation de ce document, l'utilisateur accepte la licence d'utilisation qui y est attachée, telle que détaillée dans les dispositions suivantes, et s'engage à la respecter intégralement.

La licence confère à l'utilisateur un droit d'usage sur le document consulté ou téléchargé, totalement ou en partie, dans les conditions définies ci-après et à l'exclusion expresse de toute utilisation commerciale.
Le droit d'usage défini par la licence autorise un usage à destination de tout public qui comprend :
– le droit de reproduire tout ou partie du document sur support informatique ou papier,
– le droit de diffuser tout ou partie du document au public sur support papier ou informatique, y compris par la mise à la disposition du public sur un réseau numérique,
– le droit de modifier la forme ou la présentation du document,
– le droit d'intégrer tout ou partie du document dans un document composite et de le diffuser dans ce nouveau document, à condition que :
    – L'auteur soit informé.

Les mentions relatives à la source du document et/ou à son auteur doivent être conservées dans leur intégralité.

Le droit d'usage défini par la licence est personnel et non exclusif.
Tout autre usage que ceux prévus par la licence est soumis à autorisation préalable et expresse de l'auteur : `sitepedago@telecom-paristech.fr`