Langages de l'internet

Web dynamique et projet WEB

Bruno Mascret

CPE Lyon
Techniques et langages de l'Internet Avancés

Plan

Plan

Introduction
Présentation du module

Objectif du module :

- découvrir les outils et techniques du web dynamique ;
- savoir les utiliser et les mettre en œuvre ;
- acquérir une méthodologie de développement web.

Peu de cours, beaucoup de pratique...

Introduction
Présentation du module

Différents types de TPs vous seront proposés :

1. TPs de découverte d'application du cours : destinés à vous familiariser avec les bases des langages et à vous apprendre les bonnes pratiques liées à leur utilisation ;
2. TPs de mise en pratique concrète : ces TPs « fil rouge » seront directement liés à un problème concret pour lequel il vous sera demander de proposer des solutions en utilisant les compétences et connaissances acquises en cours et durant la réalisation des TPs de découverte et d'application du cours ;

L'enchaînement des TPs se fera directement en lien avec le cours.

Introduction
Présentation du module

- séance 1 : cours 1 (2h) : Architectures PHP, templates, méthodologie de conception
- séance 2 : cours 2 (2h) : Ajax
- séance 3 : cours 3 (2h) : Web services et architecture REST

Introduction
Présentation du module

deuxième partie (TLI avancé)			
	ETI	IRC	
Cours	1	6	Architecture et méthodologie PHP/Templates
TP	1	7	Architecture et méthodologie
TP	2	8	Templates/MVC en PHP
Cours	2	7	Le « vrai » Ajax
TP	3	9	JS Ajax/PHP
Cours	3	8	Web Services/REST
TP	4	10	Web Services/REST
TP	5	11	Web Services/REST
TP	6	**fin**	Fin Web Services/REST

Introduction
Ressources

Pour ce module :

- Un support de cours (celui là)
- Un livret interactif des TPs détaillé
- Un guide détaillé
- Une webographie

Me contacter : bruno.mascret@cpe.fr

Introduction
Auteurs

Cours : **Bruno Mascret**

TPs : **Bruno Mascret**

Intervenants : **Bruno Mascret, Clémence Lop, Mohamed Sallami, Charles Perrin, Timothé Bordiga, Jonas Pauthier**

Plan

Architecture dynamiques

Il y aura exécution de code **côté serveur**

- CGI
- Server APIs (Apache modules, ISAPI)
- ASP, PHP, JSP
- Serveurs d'application (Zope, *groupware)
- Requêtes en arrière–plan : Javascript et AJAX

Exemple

- Javascript avec Ajax (Objet HttpRequest) et PHP

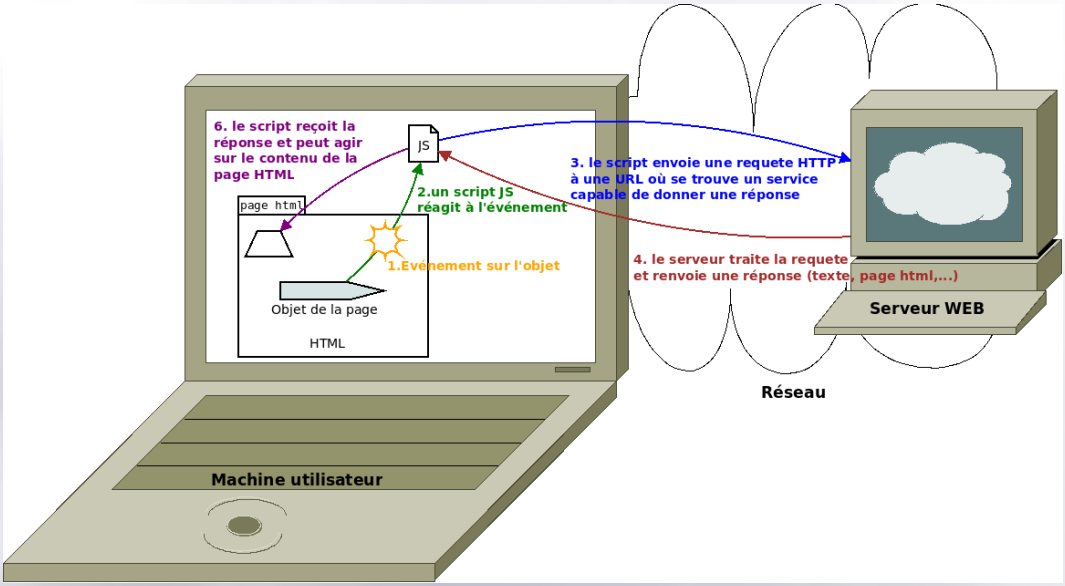


FIGURE : Principe du fonctionnement d'AJAX

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5 : exemple

```php
/*
* Line3D.php
* Represents one Line in 3-dimensional space using two Point3D objects.
*/
class Line3D
{
    private $start;
    protected $end;
    public function __construct($xCoord1=0, $yCoord1=0, $zCoord1=0, $xCoord2=1, $yCoord2=1,
        $zCoord2=1)
    {
        $this->start = new Point3D($xCoord1, $yCoord1, $zCoord1);
        $this->end = new Point3D($xCoord2, $yCoord2, $zCoord2);
    }
    /*
     * calculate the length of this Line in 3-dimensional space.
     */
    public function getLength()
    {
        return sqrt(
            pow($this->start->x - $this->end->x, 2) +
            pow($this->start->y - $this->end->y, 2) +
            pow($this->start->z - $this->end->z, 2)
        );
    }
```

Exemple venant de `http://php.net/manual/fr/language.oop5.basic.php`

Les classes en PHP 5 : exemple

```
1
2
3       /*
4        * The (String) representation of this Line as "Line3D[start, end, length]".
5        */
6       public function __toString()
7       {
8           return 'Line3D[start=' . $this->start .
9               ', end=' . $this->end .
10              ', length=' . $this->getLength() . ']';
11      }
12  }
13
14  /*
15  * create and display objects of type Line3D.
16  */
17  echo '<p>' . (new Line3D()) . "</p>\n";
18  echo '<p>' . (new Line3D(0, 0, 0, 100, 100, 0)) . "</p>\n";
19  echo '<p>' . (new Line3D(0, 0, 0, 100, 100, 100)) . "</p>\n";
20
21  ?>
```

Les méthodes commençant par __ sont des méthodes préexistantes et communes à tous les objets (comme les méthodes de la classe Object en java). On les appelle les méthodes « magiques » (!) en PHP.

Voir `http://php.net/manual/fr/language.oop5.php` et `http://php.net/manual/fr/language.oop5.magic.php`

PHP : particularités (1/2)

- Inclusion d'autres fichiers par `include()`, `require`, include_once et require_once
- Variables globales non visibles des fonctions, mot clé `global`

```
1 function test() {
2 global $var;
3 echo $var;
4 }
5 $var = "Hello world";
6 test();
```

- Certaines variables sont *superglobales* (cf. POST, GET, SESSION...)
- Constantes

```
1 define("CONSTANTE", "Bonjour le monde.");
2 echo CONSTANTE; // affiche "Bonjour le monde."
3 echo Constante; // affiche "Constante" et une note.
```

- Les fonctions peuvent être appelées par leur nom :

```
1 function salut()
2 {
3 echo 'bonjour';
4 }
5 $a = 'salut';
6 $a(); // affiche bonjour
```

PHP : particularités (1/2)

- Inclusion d'autres fichiers par `include()`, `require`, include_once et require_once
- Variables globales non visibles des fonctions, mot clé `global`

```
1 function test() {
2 global $var;
3 echo $var;
4 }
5 $var = "Hello world";
6 test();
```

- Certaines variables sont *superglobales* (cf. POST, GET, SESSION...)
- Constantes

```
1 define("CONSTANTE", "Bonjour le monde.");
2 echo CONSTANTE; // affiche "Bonjour le monde."
3 echo Constante; // affiche "Constante" et une note.
```

- Les fonctions peuvent être appelées par leur nom :

```
1 function salut()
2 {
3 echo 'bonjour';
4 }
5 $a = 'salut';
6 $a(); // affiche bonjour
```

PHP : particularités (1/2)

- Inclusion d'autres fichiers par `include()`, `require`, include_once et require_once
- Variables globales non visibles des fonctions, mot clé `global`

```php
function test() {
global $var;
echo $var;
}
$var = "Hello world";
test();
```

- Certaines variables sont *superglobales* (cf. POST, GET, SESSION...)
- Constantes

```php
define("CONSTANTE", "Bonjour le monde.");
echo CONSTANTE; // affiche "Bonjour le monde."
echo Constante; // affiche "Constante" et une note.
```

- Les fonctions peuvent être appelées par leur nom :

```php
function salut()
{
echo 'bonjour';
}
$a = 'salut';
$a(); // affiche bonjour
```

PHP : particularités (1/2)

- Inclusion d'autres fichiers par `include()`, `require`, include_once et require_once
- Variables globales non visibles des fonctions, mot clé `global`

```php
function test() {
global $var;
echo $var;
}
$var = "Hello world";
test();
```

- Certaines variables sont *superglobales* (cf. POST, GET, SESSION...)
- Constantes

```php
define("CONSTANTE", "Bonjour le monde.");
echo CONSTANTE; // affiche "Bonjour le monde."
echo Constante; // affiche "Constante" et une note.
```

- Les fonctions peuvent être appelées par leur nom :

```php
function salut()
{
echo 'bonjour';
}
$a = 'salut';
$a(); // affiche bonjour
```

PHP : particularités (1/2)

- Inclusion d'autres fichiers par `include()`, `require`, include_once et require_once
- Variables globales non visibles des fonctions, mot clé `global`

```
1 function test() {
2 global $var;
3 echo $var;
4 }
5 $var = "Hello world";
6 test();
```

- Certaines variables sont *superglobales* (cf. POST, GET, SESSION...)
- Constantes

```
1 define("CONSTANTE", "Bonjour le monde.");
2 echo CONSTANTE; // affiche "Bonjour le monde."
3 echo Constante; // affiche "Constante" et une note.
```

- Les fonctions peuvent être appelées par leur nom :

```
1 function salut()
2 {
3 echo 'bonjour';
4 }
5 $a = 'salut';
6 $a(); // affiche bonjour
```

Intégration PHP/HTML

- Insertion de code PHP dans HTML
  Technique la plus simple, mais **à proscrire dès que le volume de code augmente !**

```
1  <html>
2  <head>
3  <title>Configuration PHP</title>
4  </head>
5
6  <body>
7  <h1>Configuration PHP</h1>
8
9  <? phpinfo(); ?>
10
11 </body>
12 </html>
```

- Variables prédéfinies
  - Variables génériques (similaires à CGI : `$HTTP_USER_AGENT`, `$REMOTE_ADDR`, ...
  - Arguments de requête (POST, GET et COOKIES) dans des tableaux globaux :

```
1  $_GET['mavariable'],
2  $_POST['mavariable'],
3  $_COOKIE['moncookie']
```

PHP avancé
Les variables super globales

- Accessibles PARTOUT...
- ...ce qui ne veut pas dire qu'il soit recommandé de les utiliser partout ! ! !

Principales variables superglobales :

- $_SERVER : valeurs données par le serveur (très nombreuses !).
- $_ENV : variables d'environnement données par le serveur. ex :"USER", nom de l'utilisateur
- $_SESSION : variables pour un client qui sont stockées sur le serveur le temps de sa présence sur le site.
- $_COOKIE : variables enregistrés sur l'ordinateur du client. Pas de durée de fin de vie.
- $_GET : données envoyées en paramètres dans l'URL.
- $_POST : données de la page encapsulées dans la requête (mais visible avec les bons outils quand même !)
- $_FILES : liste des fichiers envoyés par la page (majoritairement via formulaire).

PHP avancé
Les variables super globales

Utilisation

```php
<?php

$_COOKIE[preference] = array();
$_COOKIE['preference'][couleur] = 'vert';

$_SESSION['nom'] = 'Mascret';
$_SESSION['mail'] = $_POST['mail'];

if(isset($_GET['action'])){
  if($_GET['action']=="menu"){
    affiche_Menu();
  }
  elseif($_GET['action']=="cmd"){
    afficheCommande();
  }
}
else{

  echo "rien a faire";
}
?>
```

PHP avancé
Les variables super globales

Utilisation

```php
<?php
  echo "<h1>Serveur</h1>";
  print_r($_SERVER);
  displayTab($_SERVER);
  echo "<h1>Env</h1>";
  print_r($_ENV);
  displayTab($_ENV);

  echo "<h1>POST</h1>";
  print_r($_POST);
  displayTab($_POST);

  echo "<h1>GET</h1>";
  print_r($_GET);
  displayTab($_GET);

  function displayTab($t){
    echo "<ul>";
    foreach($t as $k=>$v){
      echo "<li>".$k.": ".$v."</li>";
    }
    echo "</ul>";

  }

?>
```

PHP avancé
Les sessions

Hors programme, étudié en détail en spécialité....

- Permettent de maintenir une pseudo-connexion
- Optimisation
- Calculs facilités
- Gains en temps... si utilisées !
- Utilisent explicitement la superglobale $_SESSION et implicitement $_COOKIE

PHP avancé
Les sessions

Exemple : Déroulement d'une session

1. Ouverture de session : *session_start()* demande à PHP de générer un numéro unique (PHPSESSID) en hexadécimal (bf4c57deff89006ca44347dfe9egf899 par exemple) qui est transmis au client lors de la réponse par un cookie. A ce stade :
   - le serveur dispose chez lui du numéro de session et des données associées ;
   - le client dispose d'un cookie avec son identifiant.
2. Les scripts PHP sont alors en mesure de créer et stocker des informations sous forme de variables dans l'espace session **côté serveur** (chaînes, nombres, tableaux, etc.) ;
3. A chaque nouvelle requête client, celui–ci transfert son identifiant. **Attention : Il est indispensable d'appeller *session_start()* à chaque nouvelle requête !**. Le programme PHP est en mesure grâce à cet identifiant de retrouver les données chez lui.
4. Après un certain temps d'inactivité, ou lorsque l'utilisateur est déconnecté (*session_destroy()*), les données sont supprimées sur le serveur. Il faut alors commencer une nouvelle session.

PHP avancé
Les sessions

Exemple : Déroulement d'une session

1. Ouverture de session : *session_start()* demande à PHP de générer un numéro unique (PHPSESSID) en hexadécimal (bf4c57deff89006ca44347dfe9egf899 par exemple) qui est transmis au client lors de la réponse par un cookie. A ce stade :
   - le serveur dispose chez lui du numéro de session et des données associées ;
   - le client dispose d'un cookie avec son identifiant.
2. Les scripts PHP sont alors en mesure de créer et stocker des informations sous forme de variables dans l'espace session **côté serveur** (chaînes, nombres, tableaux, etc.) ;
3. A chaque nouvelle requête client, celui–ci transfert son identifiant. **Attention : Il est indispensable d'appeller *session_start()* à chaque nouvelle requête !**. Le programme PHP est en mesure grâce à cet identifiant de retrouver les données chez lui.
4. Après un certain temps d'inactivité, ou lorsque l'utilisateur est déconnecté (*session_destroy()*), les données sont supprimées sur le serveur. Il faut alors commencer une nouvelle session.

PHP avancé
Les sessions

Exemple : Déroulement d'une session

1. Ouverture de session : *session_start()* demande à PHP de générer un numéro unique (PHPSESSID) en hexadécimal (bf4c57deff89006ca44347dfe9egf899 par exemple) qui est transmis au client lors de la réponse par un cookie. A ce stade :
   - le serveur dispose chez lui du numéro de session et des données associées ;
   - le client dispose d'un cookie avec son identifiant.
2. Les scripts PHP sont alors en mesure de créer et stocker des informations sous forme de variables dans l'espace session **côté serveur** (chaînes, nombres, tableaux, etc.) ;
3. A chaque nouvelle requête client, celui–ci transfert son identifiant. **Attention : Il est indispensable d'appeller *session_start()* à chaque nouvelle requête !**. Le programme PHP est en mesure grâce à cet identifiant de retrouver les données chez lui.
4. Après un certain temps d'inactivité, ou lorsque l'utilisateur est déconnecté (*session_destroy()*), les données sont supprimées sur le serveur. Il faut alors commencer une nouvelle session.

PHP avancé
Les sessions

Exemple : Déroulement d'une session

1. Ouverture de session : *session_start()* demande à PHP de générer un numéro unique (PHPSESSID) en hexadécimal (bf4c57deff89006ca44347dfe9egf899 par exemple) qui est transmis au client lors de la réponse par un cookie. A ce stade :
   - le serveur dispose chez lui du numéro de session et des données associées ;
   - le client dispose d'un cookie avec son identifiant.
2. Les scripts PHP sont alors en mesure de créer et stocker des informations sous forme de variables dans l'espace session **côté serveur** (chaînes, nombres, tableaux, etc.) ;
3. A chaque nouvelle requête client, celui–ci transfert son identifiant. **Attention : Il est indispensable d'appeller *session_start()* à chaque nouvelle requête !**. Le programme PHP est en mesure grâce à cet identifiant de retrouver les données chez lui.
4. Après un certain temps d'inactivité, ou lorsque l'utilisateur est déconnecté (*session_destroy()*), les données sont supprimées sur le serveur. Il faut alors commencer une nouvelle session.

PHP avancé
Les sessions

Exemple simple

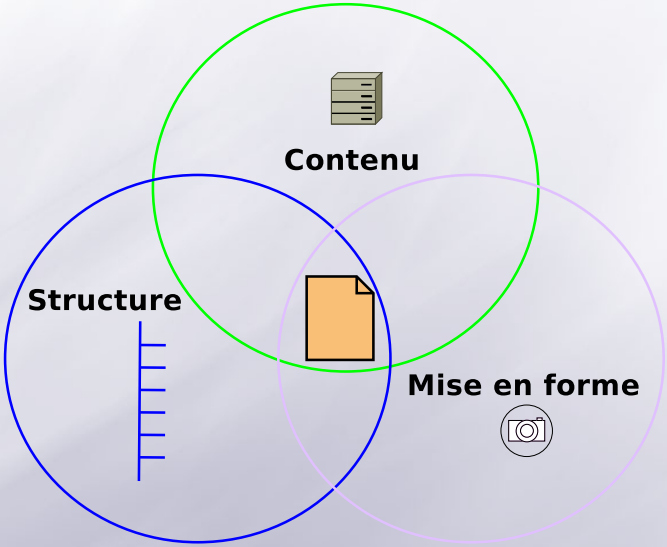
```php
<?php
session_start();

$_SESSION['prenom'] = 'Bruno';
$_SESSION['nom'] = 'Mascret';
$_SESSION['mail'] = $_POST['mail'];
?>
```

PHP avancé
Les sessions

Exemple complet

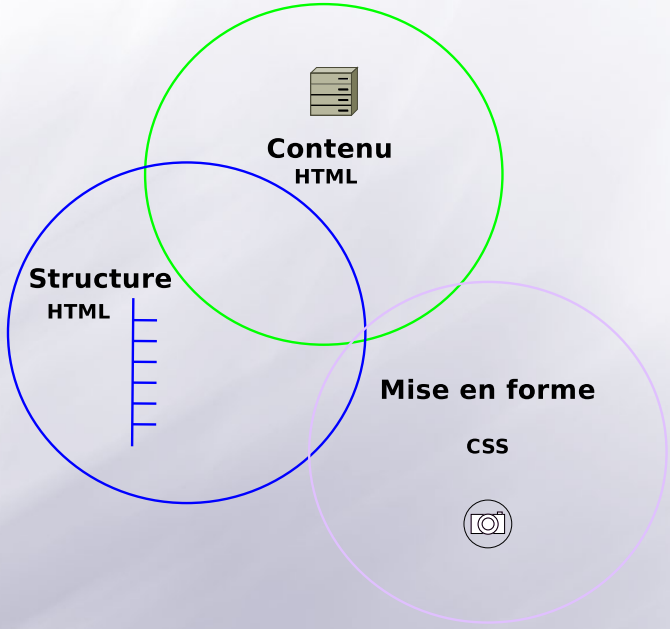
```php
<?php
  session_start();

  if(isset($_SESSION['nom'])){
    echo "Session ouverte:".$_REQUEST['SESSION_NAME'];
  }
  else{
    if(verifLoginPass($_POST['login'], $_POST['pass'])){
      //identifiant et pass ok

      $_SESSION['prenom'] = $_POST['prenom'];
      $_SESSION['nom'] =  $_POST['nom'];
      $_SESSION['mail'] = $_POST['mail'];
    }
    else{
      echo "Erreur";
    }
  }

?>
```

Problématique



Comment minimiser les zones de recouvrement ?

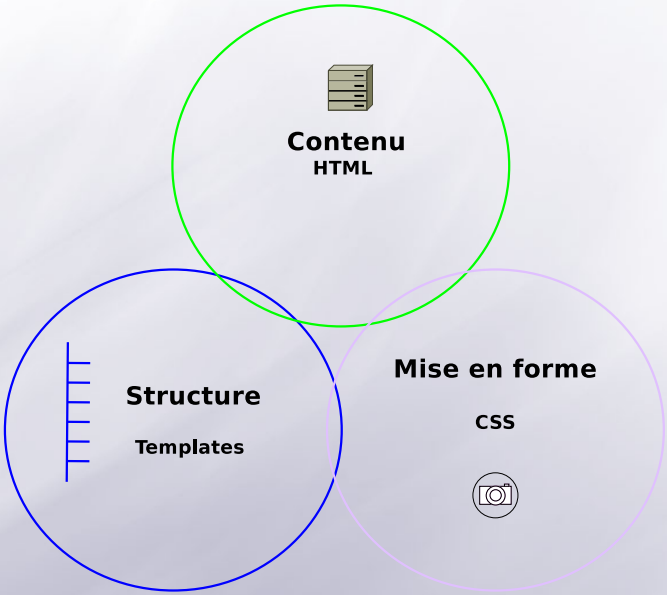
Problématique



Utilisation de CSS pour dissocier la présentation du contenu et de sa structure (TLI 1ère partie)

Plan

Réponse des templates à la problématique



Utilisation de templates (patrons) pour dissocier la structure du contenu.
Ces mécanismes de templates nécessitent l'utilisation d'un interpréteur dynamique côté serveur.
Il n'y a PAS de standards de langages de template.

Templates

Une première technique de template

```
1  <html>
2  <head>
3  <title>Configuration PHP</title>
4  </head>
5
6  <body>
7  <?php include(fragments/menu.tpl); ?>
8  <h1>Prmeière Partie</h1>
9
10 <?php include(fragments/partie1.tpl); ?>
11 </body>
12 </html>
```

Principe : inclure des fragments de fichiers HTML.

Inconvénient : pas de dynamisme !

Templates
Une deuxième technique de template

```php
<?php
....
$menuItems = array('partie1', 'partie2'....);
$titre="Partie 1";

$parser->assign("titre", $titre);
$parser->assign("items", $menuItems);

$parser->parse("partie1.tpl");

$parser->render();
?> ...
```

Principe : inclure des fichiers PHP partageant des variables contextuelles.
Inconvénient : on reste en PHP, il n'y a pas de langage de template à proprement parler.
De plus, les variables sont globales !

Templates

Une deuxième technique de template

```
partie1.tpl:


<html>
<head>
<title><?php echo $titre?></title>
</head>

<body>
<nav>
<ul>
<?php for-each($items as $i){echo "<li>".$i."</li>" ?>
</ul> ...
```

Principe : inclure des fichiers PHP partageant des variables contextuelles.
Inconvénient : on reste en PHP, il n'y a pas de langage de template à proprement parler.
De plus, les variables sont globales !

Templates
Une troisième technique de template

```
1  <?php
2  ....
3  $menuItems = array('partie1', partie2'....);
4  $titre="Partie 1";
5
6  $parser->assign("titre", $titre);
7  $parser->assign("items", $menuItems);
8
9  $parser->parse("partie1.tpl");
10
11 $parser->render();
12 ?>
13 ...
```

Principe : utiliser un langage dédié et "vrai" parseur.

Templates
Une troisième technique de template

```
partie1.tpl:

<html>
<head>
<title>{$title}</title>
</head>
<body>
{* Inclusion du fichier de menu *}
{include file='menu.tpl'}

menu.tpl:

<ul>
 {foreach from=$items item=foo}
 <li>{$foo}</li>
 {/foreach}
</ul>
...
```

Principe : utiliser un langage dédié et "vrai" parseur.

Plan

PHP avancé
Les Bases de données

- Un des points fort de PHP : lier serveur web et serveur sql
- à l'origine, une API par SGBD (mysql_connect()...)
- inconvénients : portabilité, dépendance vis–à–vis du SGBD
- solution : PDO (inclus par défaut depuis PHP 5).

PHP avancé
PDO : principes

- fonctionnement objet
- déclaration du SGBD à la connexion
- possibilité de paramétrer les requêtes
- possibilité de préparer les requêtes
- protection contre les injections

Inconvénients : problème de compatibilité de certaines instructions non standard SQL

PHP avancé
PDO : exemple

```
1
2  class BD{
3
4  private $dbName = "cinema";/*mettre le nom de votre base de donnée*/
5  private $pass = "tp"; /*donnez le mot de passe de votre bd */
6  private $user = "root"; /*donnez le nom d'utilisateur de la bd (probablement root)*/
7
8  private function getDB(){
9  $db = null;
10 try{
11 $db = new PDO('mysql:host=localhost;dbname='.$this->dbName, $this->user, $this->pass,array(
       PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8"));
12 }
13 catch (Exception $e){
14 die('Erreur : ' . $e->getMessage());
15 }
16 return $db;
17 }
```

PHP avancé
PDO : exemple

```
1
2  public function requete($sql){
3  $resu = null;
4
5  $db = $this->getDB();
6
7  foreach  ($db->query($sql) as $row) {
8  $resu[] = $row;
9  }
10 return $resu;
11 }
12 }
```