

PHP

LES MOTEURS DE TEMPLATES

L'objet de ce cours est de présenter les principes sous-jacents l'utilisation des moteurs de templates ainsi que leurs avantages et inconvénients. Dans le cadre du cours de programmation en PHP, nous terminerons en étudiant plus particulièrement deux moteurs de templates : Smarty & Mustache.

Les moteurs de templates

INTRODUCTION À LA NOTION DE GABARIT

Introduction aux moteurs de templates

Étude de cas

On veut mettre en œuvre une page php affichant une liste de personnes. Cette liste doit pouvoir être filtrée sur le nom.

Pour chaque personne l'on dispose des informations suivantes :

Un identifiant unique de la personne : id ;

son nom suivi de son prénom : nom ;

son adresse email ;

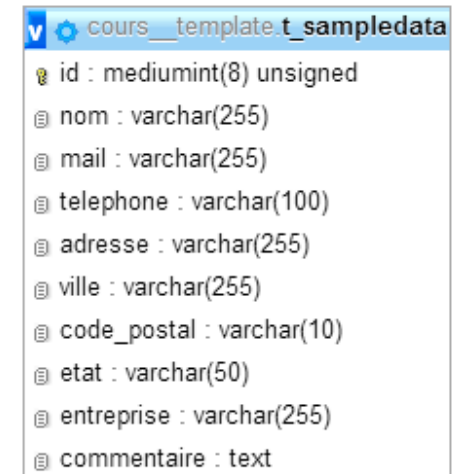
son numéro de téléphone : telephone ;

Son adresse postale : adresse, ville, code_postal, etat ;

Son entreprise : entreprise ;

Un commentaire : commentaire.

Ces données sont stockées en base de données dans une table nommée T_SampleData.



id	: mediumint(8) unsigned
nom	: varchar(255)
mail	: varchar(255)
telephone	: varchar(100)
adresse	: varchar(255)
ville	: varchar(255)
code_postal	: varchar(10)
etat	: varchar(50)
entreprise	: varchar(255)
commentaire	: text

Introduction aux moteurs de templates

Exemple de contenu

id	nom	mail	telephone	adresse	ville	code_postal	etat	entreprise	commentaire
1	Imelda Conway	a@ullamcorpereu.com	01 16 47 66 43	Appartement 511-4972 Quisque Avenue	Lisieux	31739	Lo	Purus In Foundation	ante. Maecenas
2	Dylan Gallegos	Vivamus@temporlorem.net	02 54 69 74 52	Appartement 696-3442 Lacus. Route	Brive-la-Gaillarde	57171	Li	Pellentesque Sed Corp.	semper pretium neque. Morbi
3	Lucy Moody	In.at.pede@Nam.org	02 19 73 89 05	Appartement 432-616 Lorem Av.	Pontarlier	84572	Fr	Tristique Senectus PC	Aliquam nisl. Nulla eu neque pellentesque massa lobortis ultrices.
4	Reed Castro	ante.dictum.mi@euismod.ca	06 55 71 83 71	CP 150, 9710 Vestibulum, Rue	Montauban	35304	Mi	At Velit Cras PC	Curabitur dictum. Phasellus in felis. Nulla
5	Maxwell Ruiz	eget@magnisdis.edu	05 44 26 83 68	CP 945, 9495 Gravida Rd.	Montb	15228	Fr	Maecenas Consulting	eget
6	Len Wagner	neque.Nullam.ut@ametmassa.com	01 16 70 91 19	144-5639 Diam. Rd.	Carcassonne	15506	La	Viverra Maecenas Iaculis Inc.	facilisis lorem tristique aliquet. Phasellus fermentum convallis ligula.
7	Uriel Harrison	ipsum.dolor.sit@tinciduntui.net	05 68 65 81 89	939-4275 Cubilia Av.	Besan	22422	Franche-Comt	Class Institute	pede
8	Venus Sherman	sit.amet.orci@atpede.com	09 73 86 04 90	677-932 Id Rue	Montlu	9093	Au	Magna Foundation	pretium et, rutrum non, hendrerit
9	Jescie Roy	Fusce.aliquam@Aliquamtincidunt.com	02 07 10 61 23	9338 In Rd.	Limoges	57533	Limousin	Lectus Convallis Est Industries	amet, faucibus ut, nulla. Cras eu tellus
10	Arsenio Tran	lorem@sociosqu.edu	02 28 35 15 90	444-4587 Lobortis Avenue	Rennes	32966	Brittany	Scelerisque Lorem Ipsum Company	enim. Etiam imperdiet
11	Alden Trujillo	auctor.Mauris.vel@mauris.ca	07 60 76 55 19	184-3383 Urna. Rue	Versailles	7379		Tempus Mauris Erat LLC	at fringilla
12	Kadeem Gaines	ad.litora@elitpretiumet.co.uk	06 15 70 68 46	569-6346 Mollis Rd.	Saintes	85237	Poitou-Charentes	Pharetra Corp.	lobortis,
13	Cleo Townsend	eu.tempor.erat@Vivamus.ca	01 94 56 59 10	718-3956 Duis Chemin	Limoges	41210	Li	In Inc.	primis

Introduction aux moteurs de templates

Code PHP – index.php

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Annuaire</title>
  </head>
  <body>
    <?php
      $filtre=(isset($_REQUEST['filtre'])?$_REQUEST['filtre']: "");
    ?>
    <form action="" method="post">
      <input type="text" name="filtre" value="<?=$filtre?>">
      <input type="submit" name="filtrer" value="Filtrer les noms">
    </form>
    <form action="" method="post">
      <input type="hidden" name="filtre" value="">
      <input type="submit" name="filtrer" value="Afficher toutes les entrées">
    </form>
    <table>
      <caption>Membres</caption>
      <tr><th>nom</th><th>mail</th><th>téléphone</th><th>adresse</th><th>ville</th>
      <th>code postal</th><th>etat</th><th>entreprise</th><th>commentaire</th></tr>
    <?php
      $bd_config=[
        'dsn' => 'mysql:host=localhost;dbname=cours__template;charset=utf8',
        'user' => 'root',
        'pwd' => 'root'
      ];
      $dbh = new PDO($bd_config['dsn'], $bd_config['user'], $bd_config['pwd']);

      foreach($dbh->query('SELECT * from t_sampledata'.(!empty($filtre)? " where nom like '%$filtre%'" : '')) as $r) {
        printf("<tr><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td></tr>",
          $r[1],$r[2],$r[3],$r[4],$r[5],$r[6],$r[7],$r[8],$r[9]);
      }
    ?>
  </table>
</body>
</html>
```

Introduction aux moteurs de templates

Résultat de l'appel de la page index.php

nom	mail	téléphone	adresse	ville	code postal	etat	entreprise	commentaire
Imelda Conway	a@ullamcorpereu.com	01 16 47 66 43	Appartement 511-4972 Quisque Avenue	Lisieux	31739	Lo	Purus In Foundation	ante. Maecenas
Dylan Gallegos	Vivamus@temporlorem.net	02 54 69 74 52	Appartement 696-3442 Lacus. Route	Brive-la-Gaillarde	57171	Li	Pellentesque Sed Corp.	semper pretium neque. Morbi
Lucy Moody	In at pede@Nam.org	02 19 73 89 05	Appartement 432-616 Lorem Av.	Pontarlier	84572	Fr	Tristique Senectus PC	Aliquam nisl. Nulla eu neque pellentesque massa lobortis ultrices.
Reed Castro	ante dictum mi@euismod.ca	06 55 71 83 71	CP 150, 9710 Vestibulum, Rue	Montauban	35304	Mi	At Velit Cras PC	Curabitur dictum. Phasellus in felis. Nulla
Maxwell Ruiz	eget@magnisdis.edu	05 44 26 83 68	CP 945, 9495 Gravida Rd.	Montb	15228	Fr	Maecenas Consulting	eget
Len Wagner	neque.Nullam.ut@ametmassa.com	01 16 70 91 19	144-5639 Diam. Rd.	Carcassonne	15506	La	Viverra Maecenas Iaculis Inc.	facilisis lorem tristique aliquet. Phasellus fermentum connullis ligula.
Uriel Harrison	ipsum.dolor.sit@tinciduntui.net	05 68 65 81 89	939-4275 Cubilia Av.	Besan	22422	Fr	Class Institute	pede
Venus Sherman	sit.amet.orci@atpede.com	09 73 86 04 90	677-932 Id Rue	Monthu	09093	Au	Magna Foundation	pretium et, rutrum non, hendrerit
Jesie Roy	Fusce.aliquam@Aliquamtincidunt.com	02 07 10 61 23	9338 In Rd.	Limoges	57533	Limousin	Lectus Connullis Est Industries	amet, faucibus ut, nulla. Cras eu tellus
Arsenio Tran	lorem@sociosqu.edu	02 28 35 15 90	444-4587 Lobortis Avenue	Rennes	32966	Brittany	Scelerisque Lorem Ipsum Company	erim. Etiam imperdiet
Alden Trujillo	auctor.Mauris.vel@mauris.ca	07 60 76 55 19	184-3383 Urna. Rue	Versailles	07379		Tempus Mauris Erat LLC	at fringilla
Kadeem Gaines	ad.litora@elitpretiumet.co.uk	06 15 70 68 46	569-6346 Mollis Rd.	Saintes	85237	Poitou-Charentes	Pharetra Corp.	lobortis,
Cleo Townsend	eu.tempor.erat@Vivamus.ca	01 94 56 59 10	718-3956 Duis Chemin	Limoges	41210	Li	In Inc.	primis
Eagan Lopez	Maecenas.iaculis@tinciduntpede.co.uk	02 58 26 46 01	2222 Adipiscing Rue	Sens	38194	Bo	At Inc.	parturient montes, nascetur ridiculus mms. Donec dignissim magna
John Holcomb	varius.Nam@Aeneaneget.com	08 77 47 94 89	Appartement 914-8610 Ornare Rue	Lorient	59803	Brittany	Urna Nec Company	consectetuer rhoncus. Nullam
Hayley Cabrera	ante ipsum@Vivamusmolestiedapibus.net	06 31 52 76 96	CP 569, 4989 Connullis, Chemin	Saint-Quentin	25297	Picardy	Mauris Ut Limited	velit. Pellentesque ultricies dignissim lacus. Aliquam rutrum lorem ac risus.
Cain Fernandez	ultrices.ornare@Namligula.org	08 39 80 41 29	1634 Est Avenue	Colomiers	97721	Mi	Lorem Luctus Ut Inc.	lectus, a sollicitudin orci sem eget massa. Suspendisse
Lila Shaffer	non.leo.Vivamus@Vivamusnonlorem.co.uk	05 13 64 22 21	664-7642 Nulla Rue	Vitrolles	60849	Provence-Alpes-C	Nec Urna Et Institute	venenatis lacus. Etiam bibendum fermentum metus. Aenean sed
Kalia Maldonado	ac.urna.Ut@necmalesuadaut.ca	02 48 45 24 98	219 Nulla Av.	Brive-la-Gaillarde	00059	Limousin	Nec Mauris Institute	consectetuer adipiscing elit. Curabitur sed tortor. Integer aliquam adipiscing lacus.
Arthur Goff	arcu@fringillaeuismod.ca	09 08 49 76 76	CP 463, 1304 Montes, Route	B	27734	La	Non Foundation	bibendum fermentum metus.
Denton Smith	posuere@duiFuscediam.ca	09 62 63 98 53	707-9826 Ipsum. Av.	Brive-la-Gaillarde	07610	Li	Amet Incorporated	Cras vulputate velit eu sem. Pellentesque ut ipsum ac
Lila Burke	vel.nisl.Quisque@necimperdietnec.com	02 99 62 94 76	CP 717, 6888 Cras Avenue	Auxerre	22697	Bo	Eu Odio Phasellus Incorporated	amet, consectetur adipiscing elit. Aliquam
Ivy Thompson	non.justo.Proin@commodoipsumSuspendisse.com	01 87 84 16 58	CP 663, 8192 Risus. Rd.	Mulhouse	75061	Alsace	Vitae Company	hendrerit neque. In ornare sagittis felis. Donec
Tashya Mclean	ut.mi@arcuVestibulum.net	09 34 42 78 24	765-8950 Vel Route	Limoges	71721	Limousin	Dolor Sit Amet Associates	interdum.
Doris Ingram	et ultrices.posuere@MorbimetusVivamus.net	01 16 25 35 90	CP 301, 8513 Faucibus Avenue	Charleville-M	80872	Ch	Vulputate LLP	interdum. Nunc sollicitudin commodo ipsum. Suspendisse
Olga Bray	a.magna>Lorem@mauris.edu	08 95 79 96 02	9047 Mauris. Chemin	Blois	18128	Ce	Laoreet Ipsum Curabitur Corp.	Cras dolor dolor, tempus non, lacinia
Tad Hull	lobortis.quis@risus.ca	01 35 63 71 76	Appartement 239-1089 A Ave	Moulins	50209	Auvergne	Magna Ltd	risus. Nulla eget metus eu erat semper rutrum. Fusce
Keaton Harrington	hendrerit.consectetuer@adipiscingelitCurabitur.com	02 74 83 54 96	974-118 Dolor. Route	Ajaccio	41797	Co	Mattis Cras Institute	elit. Pellentesque ultricies dignissim lacus. Aliquam
Justina Abbott	id.libero.Donec@varisorcin.org	01 98 06 31 93	9458 Aliquam Chemin	Niort	86986	Po	Lorem Luctus Ut Incorporated	tristique senectus et netus
Lawrence Marshall	libero@habitanmorbi.co.uk	07 63 96 61 95	3525 Vitae, Avenue	Dieppe	79920	Upper Normandy	Nascetur Company	dui
Amery Weaver	vitae@morbitristique.com	06 59 65 47 32	1153 In Ave	B	49030	Aq	Nunc Sed Limited	fringilla cursus purus. Nullam scelerisque
Zephania Bailey	sapien.imperdiet.ornare@Nunclaoreet.com	03 96 02 91 78	817-6864 Faucibus Chemin	Castres	17600	Mi	Eu Associates	vulputate dui,
Colleen Howard	vehicula@Vestibulumaccumsan.ca	06 86 37 86 16	244-5866 Ac Rd.	Moulins	77202	Au	Maecenas PC	elit, pharetra ut, pharetra sed, hendrerit a, arcu. Sed et
Lavinia Wall	mauris.eu.elit@estacmattis.ca	05 32 70 61 57	CP 450, 7173 Metus Av.	Antibes	76268	Provence-Alpes-C	Eros LLP	volutpat nunc sit amet metus. Aliquam erat volutpat.
Adara Burns	blandit.mattis@Donecvitae.com	06 34 01 28 26	Appartement 989-1392 Nisl. Rd.	Tours	27162	Ce	Eu Dolor PC	sit amet, dapibus id, blandit at, nisi.
Michelle Heath	ultrices.posuere@malesuadaiderat.com	09 21 18 66 66	3885 Erat Route	Paris	11756		Quisque Varius Nam Associates	non, egestas a,
Victor English	at.velit.Cras@quis.org	04 80 94 67 65	9323 Amet Ave	Wattrelos	76403	No	Imperdiet Ullamcorper Duis Institute	malesuada fames
Noel Dean	Integer@magnaSed.edu	06 57 31 24 61	CP 258, 5956 Varius. Rd.	Dole	07937	Fr	Libero Industries	mus. Proin vel nisl. Quisque fringilla euismod enim. Etiam gravida
Kimberley Hobbs	diam.dictum.sapien@justoProinnon.org	02 89 70 31 24	693-2415 Elit. Route	Troyes	45636	Champagne-Ardenne	Sed Industries	lobortis augue scelerisque mollis. Phasellus libero mauris, aliquam eu, accumsan
Lawrence Welch	neque@malesuadafamesac.org	08 36 08 42 31	476-3222 Sagittis Chemin	Tournefeuille	77784	Midi-Pyr	Lectus Pede Company	et, rutrum eu, ultrices sit

Introduction aux moteurs de templates

Résultat de l'appel après avoir spécifié un filtre

St

Filtrer les noms

Afficher toutes les entrées

Membres

nom	mail	téléphone	adresse	ville	code postal	etat	entreprise	commentaire
Reed Castro	ante.dictum.mi@euismod.ca	06 55 71 83 71	CP 150, 9710 Vestibulum, Rue	Montauban	35304	Mi	At Velit Cras PC	Curabitur dictum. Phasellus in felis. Nulla
Justina Abbott	id.libero.Donec@variusorciin.org	01 98 06 31 93 9458	Aliquam Chemin	Niort	86986	Po	Lorem Luctus Ut Incorporated	tristique senectus et netus
Lynn Webster	nec.tempus.scelerisque@iaculis.ca	02 98 41 38 57	CP 226, 5206 Sagittis Rue	Argenteuil	88173		Tellus Phasellus Limited	Sed auctor odio a purus. Duis elementum, dui
Germane Stokes	vehicula.aliquet@velitegestas.edu	07 32 47 82 00	133-1199 Orci. Rue	Bayonne	47182	Aquitaine	Ipsum Incorporated	lacus vestibulum lorem, sit amet ultricies sem magna
Jaquelyn Stuart	sed@lorem.edu	04 35 36 96 97	Appartement 872-3514 Parturient Chemin	Belfort	57281	Franche-Comt	Aliquet Foundation	erat vel pede blandit congue. In scelerisque scelerisque
Chastity Small	tempor.diam.dictum@mattisInteger.co.uk	05 21 46 60 74	2535 Ligula Rue	Besan	23311	Fr	Dolor LLC	enim, condimentum

Introduction aux moteurs de templates

Retour au code

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Annuaire</title>
  </head>
  <body>
    <?php
      $filtre=(isset($_REQUEST['filtre'])?$_REQUEST['filtre']: "");
    ?>
    <form action="" method="post">
      <input type="text" name="filtre" value="<?=$filtre?>">
      <input type="submit" name="filtrer" value="Filtrer les noms">
    </form>
    <form action="" method="post">
      <input type="hidden" name="filtre" value="">
      <input type="submit" name="filtrer" value="Afficher toutes les entrées">
    </form>
    <table>
      <caption>Membres</caption>
      <tr><th>nom</th><th>mail</th><th>téléphone</th><th>adresse</th><th>ville</th>
      <th>code postal</th><th>etat</th><th>entreprise</th><th>commentaire</th></tr>
    <?php
      $bd_config=[
        'dsn' => 'mysql:host=localhost;dbname=cours__template;charset=utf8',
        'user' => 'root',
        'pwd' => 'root'
      ];
      $dbh = new PDO($bd_config['dsn'], $bd_config['user'], $bd_config['pwd']);

      foreach($dbh->query('SELECT * from t_sampledata'.(!empty($filtre)? " where nom like '%$filtre%'" : '')) as $r) {
        printf("<tr><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td></tr>",
          $r[1],$r[2],$r[3],$r[4],$r[5],$r[6],$r[7],$r[8],$r[9]);
      }
    ?>
  </table>
</body>
</html>
```

Introduction aux moteurs de templates

Retour au code

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Annuaire</title>
  </head>
  <body>
    <?php
      $filtre=(isset($_REQUEST['filtre'])?$_REQUEST['filtre']: "");
    ?>
    <form action="" method="post">
      <input type="text" name="filtre" value="<?=$filtre?>">
      <input type="submit" name="filtrer" value="Filtrer les noms">
    </form>
    <form action="" method="post">
      <input type="hidden" name="filtre" value="">
      <input type="submit" name="filtrer" value="Afficher toutes les entrées">
    </form>
    <table>
      <caption>Membres</caption>
      <tr><th>nom</th><th>mail</th><th>téléphone</th><th>adresse</th><th>ville</th>
      <th>code postal</th><th>etat</th><th>entreprise</th><th>commentaire</th></tr>
    <?php
      $bd_config=[
        'dsn' => 'mysql:host=localhost;dbname=cours__template;charset=utf8',
        'user' => 'root',
        'pwd' => 'root'
      ];
      $dbh = new PDO($bd_config['dsn'], $bd_config['user'], $bd_config['pwd']);

      foreach($dbh->query('SELECT * from t_sampledata'.(!empty($filtre)? " where nom like '%$filtre%'" : '')) as $r) {
        printf("<tr><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td></tr>",
          $r[1],$r[2],$r[3],$r[4],$r[5],$r[6],$r[7],$r[8],$r[9]);
      }
    ?>
  </table>
</body>
</html>
```

Avantages:

- + facile et rapide à écrire ;
- + pas la peine de réfléchir.
Est-ce vraiment un avantage ?

Inconvénients:

- Le code mélange complètement php & html i.e. logique fonctionnelle et présentation ;
- La vue comporte une partie de la logique fonctionnelle.

Introduction aux moteurs de templates

Il est important de séparer la logique métier de la présentation, chacun son rôle :

- ⊙ **Logique métier <-> développeur**
- ⊙ **Présentation <-> intégrateur**

Cela permet à chacun de se concentrer sur sa tâche sans interférer sur celle de l'autre.

Peut-on opérer cette séparation en PHP avec le langage natif ?

Oui et non..., en tous cas, on peut clairement améliorer les choses.

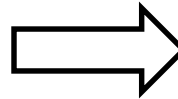
Introduction aux moteurs de templates

Utilisation de la syntaxe alternative

En PHP il existe une syntaxe alternative pour rassembler les instructions dans un bloc pour les fonctions de contrôle suivantes : if, while, for, foreach et switch.

Le principe consiste à remplacer l'accolade d'ouverture par deux points (:) et l'accolade de fermeture par: end<instruction>;

```
<ul>  
<?php foreach ($todo as $item) {  
    echo "<li>$item</li>";  
}?>  
</ul>
```



```
<ul>  
<?php foreach ($todo as $item): ?>  
    <li><?=$item?></li>  
<?php endforeach; ?>  
</ul>
```

Introduction aux moteurs de templates

Retour au code – sans la syntaxe alternative

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Annuaire</title>
  </head>
  <body>
    <?php
      $filtre=(isset($_REQUEST['filtre'])?$_REQUEST['filtre']: "");
    ?>
    <form action="" method="post">
      <input type="text" name="filtre" value="<?=$filtre?>">
      <input type="submit" name="filtrer" value="Filtrer les noms">
    </form>
    <form action="" method="post">
      <input type="hidden" name="filtre" value="">
      <input type="submit" name="filtrer" value="Afficher toutes les entrées">
    </form>
    <table>
      <caption>Membres</caption>
      <tr><th>nom</th><th>mail</th><th>téléphone</th><th>adresse</th><th>ville</th>
      <th>code postal</th><th>etat</th><th>entreprise</th><th>commentaire</th></tr>
    <?php
      $bd_config=[
        'dsn' => 'mysql:host=localhost;dbname=cours__template;charset=utf8',
        'user' => 'root',
        'pwd' => 'root'
      ];
      $dbh = new PDO($bd_config['dsn'], $bd_config['user'], $bd_config['pwd']);

      foreach($dbh->query('SELECT * from t_sampledata'.(!empty($filtre)? " where nom like '%$filtre%'" : '')) as $r) {
        printf("<tr><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td></tr>",
          $r[1],$r[2],$r[3],$r[4],$r[5],$r[6],$r[7],$r[8],$r[9]);
      }
    ?>
  </table>
</body>
</html>
```

Introduction aux moteurs de templates

Retour au code – avec la syntaxe alternative

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Annuaire</title>
  </head>
  <body>
    <?php
      $filtre=(isset($_REQUEST['filtre'])?$_REQUEST['filtre']: "");
    ?>
    <form action="" method="post">
      <input type="text" name="filtre" value="<?=$filtre?>">
      <input type="submit" name="filtrer" value="Filtrer les noms">
    </form>
    <form action="" method="post">
      <input type="hidden" name="filtre" value="">
      <input type="submit" name="filtrer" value="Afficher toutes les entrées">
    </form>
    <table>
      <caption>Membres</caption>
      <tr><th>nom</th><th>mail</th><th>téléphone</th><th>adresse</th><th>ville</th>
      <th>code postal</th><th>etat</th><th>entreprise</th><th>commentaire</th></tr>
      <?php
        $bd_config=[
          'dsn' => 'mysql:host=localhost;dbname=cours__template;charset=utf8',
          'user' => 'root',
          'pwd' => 'root'
        ];
        $dbh = new PDO($bd_config['dsn'], $bd_config['user'], $bd_config['pwd']);
        $people=$dbh->query('SELECT * from t_sampledata'.(!empty($filtre)? " where nom like '%$filtre%'" : ''))->fetchAll();
      ?>
      <?php foreach ($people as $p): ?>
        <tr><td><?=$p['nom']?></td><td><?=$p['mail']?></td><td><?=$p['telephone']?></td>
        <td><?=$p['adresse']?></td><td><?=$p['ville']?></td><td><?=$p['code_postal']?></td><td><?=$p['etat']?></td>
        <td><?=$p['entreprise']?></td><td><?=$p['commentaire']?></td></tr>
      <?php endforeach; ?>
    </table>
  </body>
</html>
```

Introduction aux moteurs de templates

Retour au code – avec la syntaxe alternative

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Annuaire</title>
  </head>
  <body>
    <?php
      $filtre=(isset($_REQUEST['filtre'])?$_REQUEST['filtre']: "");
    ?>
    <form action="" method="post">
      <input type="text" name="filtre" value="<?=$filtre?>">
      <input type="submit" name="filtrer" value="Filtrer les noms">
    </form>
    <form action="" method="post">
      <input type="hidden" name="filtre" value="">
      <input type="submit" name="filtrer" value="Afficher toutes les entrées">
    </form>
    <table>
      <caption>Membres</caption>
      <tr><th>nom</th><th>mail</th><th>téléphone</th><th>adresse</th><th>ville</th>
      <th>code postal</th><th>etat</th><th>entreprise</th><th>commentaire</th></tr>
      <?php
        $bd_config=[
          'dsn' => 'mysql:host=localhost;dbname=cours__template;charset=utf8',
          'user' => 'root',
          'pwd' => 'root'
        ];
        $dbh = new PDO($bd_config['dsn'], $bd_config['user'], $bd_config['pwd']);
        $people=$dbh->query('SELECT * from t_sampledata'.(!empty($filtre)? " where nom like '%$filtre%'" : ''))->fetchAll();
      ?>
      <?php foreach ($people as $p): ?>
        <tr><td><?=$p['nom']?></td><td><?=$p['mail']?></td><td><?=$p['telephone']?></td>
        <td><?=$p['adresse']?></td><td><?=$p['ville']?></td><td><?=$p['code_postal']?></td><td><?=$p['etat']?></td>
        <td><?=$p['entreprise']?></td><td><?=$p['commentaire']?></td></tr>
      <?php endforeach; ?>
    </table>
  </body>
</html>
```

Avantages:

- + facile et rapide à écrire ;
- + on arrive à identifier les parties PHP des parties HTML.

Inconvénients:

- Le code mélange toujours php & html ;
- La vue comporte une partie de la logique fonctionnelle.

Introduction aux moteurs de templates

Retour au code

On peut faire un peu mieux en réarrangeant le code PHP et en séparant d'avantage la partie présentation de la partie logique fonctionnelle.

Introduction aux moteurs de templates

Retour au code – avec la syntaxe alternative + en réarrangeant le code

```
<?php
$bd_config=[
    'dsn' => 'mysql:host=localhost;dbname=cours__template;charset=utf8',
    'user' => 'root',
    'pwd' => 'root'
];
$dbh = new PDO($bd_config['dsn'], $bd_config['user'], $bd_config['pwd']);

$filtre=(isset($_REQUEST['filtre'])?$_REQUEST['filtre']: "");
$people=$dbh->query('SELECT * from t_sampledata'.(!empty($filtre)?" where nom like '%$filtre%'" : ''))->fetchAll();
?>
<!doctype html>
<html lang="fr">
<head>
<meta charset="UTF-8">
<title>Annuaire</title>
</head>
<body>
<form action="" method="post">
<input type="text" name="filtre" value="<?=$filtre?>">
<input type="submit" name="filtrer" value="Filtrer les noms">
</form>
<form action="" method="post">
<input type="hidden" name="filtre" value="">
<input type="submit" name="filtrer" value="Afficher toutes les entrées">
</form>
<table>
<caption>Membres</caption>
<tr><th>nom</th><th>mail</th><th>téléphone</th><th>adresse</th><th>ville</th>
<th>code postal</th><th>etat</th><th>entreprise</th><th>commentaire</th></tr>
<?php foreach ($people as $p): ?>
<tr><td><?=$p['nom']?></td><td><?=$p['mail']?></td><td><?=$p['telephone']?></td><td><?=$p['adresse']?></td><td><?=$p['ville']?></td>
<td><?=$p['code_postal']?></td><td><?=$p['etat']?></td><td><?=$p['entreprise']?></td><td><?=$p['commentaire']?></td></tr>
<?php endforeach; ?>
</table>
</body>
</html>
```

Introduction aux moteurs de templates

Retour au code – avec la syntaxe alternative + en réarrangeant le code

```
<?php
$bd_config=[
    'dsn' => 'mysql:host=localhost;dbname=cours__template;charset=utf8',
    'user' => 'root',
    'pwd' => 'root'
];
$dbh = new PDO($bd_config['dsn'], $bd_config['user'], $bd_config['pwd']);

$filtre=(isset($_REQUEST['filtre'])?$_REQUEST['filtre']: "");
$people=$dbh->query('SELECT * from t_sampledata.(!empty($filtre)?" where nom like '%$filtre%':"')->fetchAll();
?>
```

logique fonctionnelle

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Annuaire</title>
  </head>
  <body>
    <form action="" method="post">
      <input type="text" name="filtre" value="<?=$filtre?>">
      <input type="submit" name="filtrer" value="Filtrer les noms">
    </form>
    <form action="" method="post">
      <input type="hidden" name="filtre" value="">
      <input type="submit" name="filtrer" value="Afficher toutes les entrées">
    </form>
    <table>
      <caption>Membres</caption>
      <tr><th>nom</th><th>mail</th><th>téléphone</th><th>adresse</th><th>ville</th>
        <th>code postal</th><th>etat</th><th>entreprise</th><th>commentaire</th></tr>
      <?php foreach ($people as $p): ?>
        <tr><td><?=$p['nom']?></td><td><?=$p['mail']?></td><td><?=$p['telephone']?></td><td><?=$p['adresse']?></td><td><?=$p['ville']?></td>
          <td><?=$p['code_postal']?></td><td><?=$p['etat']?></td><td><?=$p['entreprise']?></td><td><?=$p['commentaire']?></td></tr>
      <?php endforeach; ?>
    </table>
  </body>
</html>
```

présentation

Introduction aux moteurs de templates

On commence par traiter les données d'abords. On utilise la syntaxe alternative pour écrire la vue.

Avantages:

- + Bonne séparation de la logique fonctionnelle et de la couche de présentation
- + logique métier et affichage sont indépendant.
- + Il est possible à un graphiste et à un développeur de travail en parallèle, chacun sur ça partie.

Inconvénients:

- Le graphiste doit connaitre PHP.
- Risque de perversion de la partie fonctionnelle par du code écrit dans la couche présentation.

Les moteurs de templates

PRINCIPE

Principe d'un moteur de templates

Qu'est-ce qu'un moteur de templates ?

Un moteur de templates permet la séparation de la logique métier et de la logique de présentation i.e. il fait le lien entre les données et leur présentation tout en garantissant la séparation des scripts PHP et du code HTML.

Pour cela, on utilise un template pour décrire une vue. Le moteur de templates aura pour rôle d'y intégrer les données et d'en faire le rendu.

Un template (gabarit, ou encore patron en français) est un fichier combinant du code HTML (HTML, javascript et css) et du code spécifique au moteur de templates utilisé.

Il existe principalement deux types de moteurs de templates :

Ceux qui parsent – fonctionnent selon le principe search&replace

Ceux qui compilent – produisent du code PHP

Principe d'un moteur de template

Le moteur de templates va analyser le fichier `template.tpl` et le combiner aux données. La méthode employée dépend du type de moteur de templates.

S'il s'agit d'un moteur qui repose sur le parsing, le moteur de templates parcourt les fichiers de vue, identifie et remplace les marqueurs par les valeurs qui leurs sont associées.

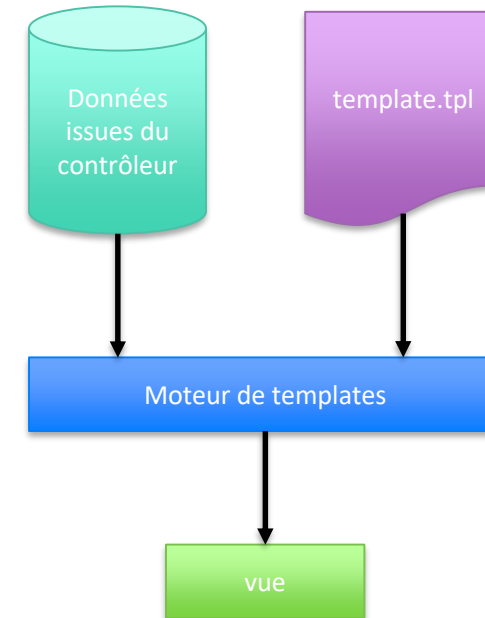
S'il s'agit d'un moteur qui fonctionne par compilation, le moteur compile les données et les vues pour produire un code PHP qui sera à son tour interprété.

Quelque soit le cas, cela est transparent pour l'utilisateur.

Principe d'un moteur de templates

L'utilisation d'un moteur de templates nécessite toujours les étapes suivantes :

1. Génération des données ;
2. Assignment des variables au template ;
3. Appel de la méthode permettant le rendu de la vue ;
4. Affichage de la vue retournée.



Principe d'un moteur de templates

Exemple

```
stdClass Object
(  
    [FAV_ID] => 5  
    [FAV_TITRE] => MétéoFrance  
    [FAV_LIEN] => http://france.meteofrance.com/  
    [FAV_DESCRIPTION] => Site web de METEO  
    France.  
)
```

```
<form method="post">  
  <div class="alert alert-danger">  
    <p>Voulez-vous vraiment supprimer ce favoris ?  
    <br>{$favoris->FAV_TITRE} :  
    <a href="{$favoris->FAV_LIEN}" target="_blank">{$favoris->FAV_LIEN}</a></p>  
    <p>{$favoris->FAV_DESCRIPTION}</p>  
    <div>  
      <input type="submit" name="action" value="delete">  
      <input type="submit" name="action" value="annuler">  
    </div>  
  </div>  
</form>
```



Supprimer un favori

Voulez-vous vraiment supprimer ce favoris ?
MétéoFrance : <http://france.meteofrance.com/>
Site web de METEO France.

Principe d'un moteur de templates

Templates ou pas templates ?

Avantages :

- + La séparation du PHP et du HTML permet une meilleure visibilité dans le code :
syntaxe propre et condensée dédiée au design.
code HTML beaucoup plus lisible.
- + Interdit le traitement dans les vues :
beaucoup plus difficile sans un moteur de templates, voir impossible.
- + Sécurité : le moteur de templates gère les échappements de chaînes.
- + La mise en cache offerte par certain moteurs permet d'économiser les ressources des serveurs.

Inconvénients :

- Chute de performances : retardement du chargement de la page (en fonction de la performance du moteur utilisé).
- Langage de templates spécifique : encore un langage à apprendre.
- Difficile de parser/debugger un template.
- Pas ou peu de coloration syntaxique dans les IDE.

Les moteurs de templates

EXEMPLE DE QUELQUES MOTEURS DE TEMPLATES

Les moteurs de templates

PHPLIB

PHPLib



<http://sourceforge.net/projects/phplib/>

PHPLib est une librairie qui embarque, entre autre, un moteur de templates. Ce moteur repose sur un parseur.

Le moteur est géré par une classe nommée Template qui permet de charger un gabarit et d'en remplir les marqueurs par des données arbitraires. Il supporte les variables scalaires mais aussi les tableaux via la notion de bloc.

L'un des avantages de ce moteur est qu'il est très peu permissif. Le gabarit est dans sa grande majorité du code HTML. Le marquage lui-même utilise les commentaires HTML.

PHPLib

Fichier index.php

```
<?php
require('template.inc');
$tpl = new Template('./');
$tpl->set_file('main','main.html');

$bd_config=[
    'dsn' => 'mysql:host=localhost;dbname=cours__template;charset=utf8',
    'user' => 'root',
    'pwd' => 'root'
];
$dbh = new PDO($bd_config['dsn'], $bd_config['user'], $bd_config['pwd']);

$filtre=(isset($_REQUEST['filtre'])?$_REQUEST['filtre']: "");
$people=$dbh->query('SELECT * from t_sampledata'.(!empty($filtre)?" where nom like '%$filtre%'" : ''))->fetchAll();

$tpl->set_block('main','person','bloc_people');
foreach($people as $p) {
    $tpl->set_var('nom', $p['nom']);
    $tpl->set_var('mail', $p['mail']);
    $tpl->set_var('telephone', $p['telephone']);
    $tpl->set_var('adresse', $p['adresse']);
    $tpl->set_var('ville', $p['ville']);
    $tpl->set_var('code_postal', $p['code_postal']);
    $tpl->set_var('etat', $p['etat']);
    $tpl->set_var('entreprise', $p['entreprise']);
    $tpl->set_var('commentaire', $p['commentaire']);
    $tpl->parse('bloc_people', 'person', TRUE);
}
$tpl->set_var('filtre',$filtre);
$tpl->pparse('resultat', 'main');
?>
```

PHPLib

Fichier main.html

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Annuaire</title>
  </head>
  <body>
    <form action="" method="post">
      <input type="text" name="filtre" value="{filtre}">
      <input type="submit" name="filtrer" value="Filtrer les noms">
    </form>
    <form action="" method="post">
      <input type="hidden" name="filtre" value="">
      <input type="submit" name="filtrer" value="Afficher toutes les entrées">
    </form>
    <table>
      <caption>Membres</caption>
      <tr><th>nom</th><th>mail</th><th>téléphone</th><th>adresse</th><th>ville</th>
        <th>code postal</th><th>etat</th><th>entreprise</th><th>commentaire</th></tr>
      <!-- BEGIN person -->
      <tr><td>{nom}</td><td>{mail}</td><td>{telephone}</td>
        <td>{adresse}</td><td>{ville}</td><td>{code_postal}</td>
        <td>{etat}</td><td>{entreprise}</td><td>{commentaire}</td></tr>
      <!-- END person -->
    </table>
  </body>
</html>
```

PHPLib

Avantages

- + Mise en œuvre très simple
4 fonctions suffisent
- + Marquage peu permissif
`{ } <!-- BEGIN truc --> <!-- END truc -->`
- + Peut travailler sur de très grande quantité de données.
- + Facile à déboguer

Inconvénients

- Pas de système de cache
Le temps nécessaire au parsing peut devenir assez vite handicapant
- Peu de fonctionnalités
Le traitement des données doit être intégralement réalisé avant la phase d'assemblage. Cela peut conduire à une confusion dans le code de la logique fonctionnelle et de la logique de présentation.

Les moteurs de templates

SMARTY

Smarty



<http://www.smarty.net/>

Smarty est un moteur de templates qui fonctionne par compilation

Smarty lit les templates et crée des scripts PHP à partir de ces derniers. Une fois créés, ils sont exécutés. Il n'y a pas d'analyse coûteuse de template à chaque requête, et les templates peuvent bénéficier des solutions de cache PHP.

Smarty permet de séparer la logique métier de la logique de présentation

Contrairement à PHPLib, les templates peuvent contenir des traitements, du moment qu'ils sont relatifs à la présentation.

Par exemple : Inclure d'autres templates, alterner les couleurs des lignes d'un tableau, mettre du texte en majuscule, parcourir un tableau de données pour l'afficher, etc.

On peut résumer le fonctionnement de Smarty ainsi:

1. lecture du script PHP ;
2. lecture du template ;
3. Compilation ;
4. création du script PHP à partir des deux autres ;
5. exécution du code généré.

Smarty

Toutes les balises Smarty sont entourées de délimiteurs.

Par défaut, ils sont { et }, mais ils peuvent être modifiés.

Commentaire :

```
{* ceci est un commentaire *}
```

Variable

Les variables de template commencent par un signe dollar (\$). Elles peuvent contenir des nombres, des lettres et des underscores, tout comme une variable PHP.

```
{ $foo } <-- affiche une variable simple (qui n'est pas un tableau ou un objet)
```

```
{ $foo[4] } <-- affiche le 5ème élément d'un tableau indexé
```

```
{ $foo.bar } <-- affiche la clé "bar" d'un tableau, identique à $foo['bar'] en PHP
```

```
{ $foo.$bar } <-- affiche la valeur de la clé d'un tableau, identique à $foo[$bar] en PHP
```

```
{ $foo->bar } <-- affiche la propriété "bar" de l'objet
```

```
{ $foo->bar() } <-- affiche la valeur retournée par la méthode "bar" de l'objet
```

Smarty

Les balises Smarty affichent une variable ou invoquent une fonction. Elles sont appelées lorsqu'elles sont entourées, ainsi que leurs paramètres, des délimiteurs Smarty. Par exemple : {nomfonction attr1='val' attr2='val'}.

Exemple :

```
{if $logged_in}
    Bonjour, <font color="{#fontColor#}">{$name}</font>
{else}
    Bonjour, {$name}!
{/if}
```

Les paramètres

La plupart des fonctions attendent des paramètres. Les paramètres des fonctions Smarty sont très proches des attributs des balises HTML. Les valeurs numériques n'ont pas besoin d'être entourées par des guillemets, par contre, ces guillemets sont recommandées lors de l'utilisation de chaînes de caractères. Des variables peuvent aussi être utilisées en tant que paramètres, et ne doivent pas être entourées de guillemets.

Smarty

Modificateurs de variables

Les modificateurs de variables peuvent être appliqués aux variables, fonctions utilisateurs ou chaînes de caractères. Pour appliquer un modificateur de variable, tapez une valeur suivie de | (pipe) et du nom du modificateur.

Exemples :

{* applique un modificateur à une variable *}

```
{ $titre|upper }
```

{* modificateur avec paramètres *}

```
{ $titre|truncate:40:'...' }
```

Smarty

Les modificateurs suivants sont disponibles dans smarty :

capitalize regex_replace

cat replace

count_characters spacyfy

count_paragraphs string_format

count_sentences strip

count_words strip_tags

date_format truncate

default upper

escape wordwrap

indent etc...

lower

nl2br

Smarty

Combiner des modificateurs de variables

Vous pouvez appliquer un nombre quelconque de modificateurs à une variable. Ils seront invoqués dans l'ordre d'apparition, de la gauche vers la droite. Ils doivent être séparés par un | (pipe).

Exemples :

```
{ $titreArticle }
```

```
{ $titreArticle | upper | spacyfy }
```

```
{ $titreArticle | lower | spacyfy | truncate }
```

```
{ $titreArticle | lower | truncate:30 | spacyfy }
```

```
{ $titreArticle | lower | spacyfy | truncate:30: ". . . " }
```

Smarty

Smarty est fourni en standard avec plusieurs fonctions natives. Ces fonctions sont partie intégrante du moteur de Smarty.

Parmi les plus utiles, on peut citer :

{assign} {if},{elseif},{else}

{block} {include}

{capture}{literal}

{extends} {nocache}

{for} {while}

{foreach},{foreachelse}

Smarty

{foreach}...{foreachelse}

{foreach} est utilisé pour parcourir un simple tableau associatif

Exemple :

```
<ul>
{foreach $myPeople as $value}
  <li>{$value}</li>
{/foreach}
</ul>
```


Smarty

{literal}

Les balises `{literal}` permettent à un bloc de données d'être pris tel quel, sans qu'il ne soit interprété par Smarty. Très pratique lors de l'emploi d'éléments tels que javascript, accolades et autres que peuvent confondre le moteur de template.

Exemple :

```
{literal}  
    function bazzy {alert('foobar!');}  
{/literal}
```

Smarty

{if},{elseif},{else}

L'instruction `{if}` dans Smarty dispose de la même flexibilité que l'instruction PHP `if`, avec quelques fonctionnalités supplémentaires pour le moteur de template. Tous les `{if}` doivent être utilisés de pair avec un `{/if}`. `{else}` et `{elseif}` sont également des balises autorisées. Toutes les conditions et fonctions PHP sont reconnues, comme `||`, `or`, `&&`, `and`, `is_array()`, etc.

```
{if $name eq 'Fred'}  
    Bienvenue, Monsieur.  
{elseif $name eq 'Wilma'}  
    Bienvenue m'dame.  
{else}  
    Bienvenue, qui que vous soyez.  
{/if}
```

{* Un exemple avec l'opérateur or *}

```
{if $name eq 'Fred' or $name eq 'Wilma'}  
    ...  
{/if}
```

{* même chose que ci-dessus *}

```
{if $name == 'Fred' || $name == 'Wilma'}  
    ...  
{/if}
```

```
{* les parenthèses sont autorisées *}  
{if ( $amount < 0 or $amount > 1000 ) and  
$volume >= #minVolAmt#}  
    ...  
{/if}  
  
{* vous pouvez également faire appel aux  
fonctions PHP *}  
{if count($var) gt 0}  
    ...  
{/if}  
  
{* vérifie si c'est un tableau. *}  
{if is_array($foo) }  
    .....  
{/if}  
  
{* vérifie si la variable est nulle. *}  
{if isset($foo) }  
    .....  
{/if}
```

Smarty

{include}

Les balises {include} sont utilisées pour inclure des templates à l'intérieur d'autres templates. Toutes les variables disponibles dans le template réalisant l'inclusion sont disponibles dans le template inclus.

Exemple :

```
<html>
<head>
  <title>{$title}</title>
</head>
<body>
{include file='page_header.tpl'}

{* body of template goes here, the $tpl_name variable
  is replaced with a value eg 'contact.tpl'
*}
{include file="$tpl_name.tpl"}

{* using shortform file attribute *}
{include 'page_footer.tpl'}
</body>
</html>
```

Smarty

L'objectif de ce cours n'est pas de réécrire la documentation de Smarty, je vous invite à aller la consulter :
<http://www.smarty.net/docs/en/>

Smarty

Retour au code – fichier index.php

```
<?php
require('/libs/Smarty/Smarty.class.php');
$smarty = new Smarty;
$smarty-> caching = true;
$smarty-> cache_lifetime = 120;

$bd_config=[
    'dsn' => 'mysql:host=localhost;dbname=cours__template;charset=utf8',
    'user' => 'root',
    'pwd' => 'root'
];
$dbh = new PDO($bd_config['dsn'], $bd_config['user'], $bd_config['pwd']);

$filtre=(isset($_REQUEST['filtre'])?$_REQUEST['filtre']:"");
$people=$dbh->query('SELECT * from t_sampledata'.(!empty($filtre)?" where nom like '%$filtre%':""));
$smarty->assign("filtre",$filtre, true);
$smarty->assign("people",$people);
$smarty->display(main.tpl,"$filtre");
?>
```

Smarty

Retour au code – fichier main.tpl

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Annuaire</title>
  </head>
  <body>
    <form action="" method="post">
      <input type="text" name="filtre" value="{{ $filtre }}">
      <input type="submit" name="filtrer" value="Filtrer les noms">
    </form>
    <form action="" method="post">
      <input type="hidden" name="filtre" value="">
      <input type="submit" name="filtrer" value="Afficher toutes les entrées">
    </form>
    <table>
      <caption>Membres</caption>
      <tr><th>nom</th><th>mail</th><th>téléphone</th><th>adresse</th><th>ville</th>
        <th>code postal</th><th>etat</th><th>entreprise</th><th>commentaire</th></tr>
      {foreach $people as $p}
        <tr><td>{{ $p.nom }}</td><td>{{ $p.mail }}</td><td>{{ $p.telephone }}</td><td>{{ $p.adresse }}</td><td>{{ $p.ville }}</td>
          <td>{{ $p.code_postal }}</td><td>{{ $p.etat }}</td><td>{{ $p.entreprise }}</td><td>{{ $p.commentaire }}</td></tr>
      {/foreach}
    </table>
  </body>
</html>
```

Smarty, utilisation de l'héritage

{extends}

Les balises {extends} sont utilisées dans les templates enfants lors de l'héritage de template pour étendre les templates parents.

Exemple :

```
{extends file='parent.tpl'}
```

```
{extends 'parent.tpl'}  {* short-hand *}
```

Smarty, utilisation de l'héritage

{block}

La balise {block} est utilisée pour définir une zone nommée dans un template. Ceci est utile lors de l'emploi de l'héritage de template.

Exemple :

Fichier parent.tpl

```
<html>
  <head>
    <title>{block name="title"}Default Title{/block}</title>
    <title>{block "title"}Default Title{/block}</title>  {* short-hand *}
  </head>
</html>
```

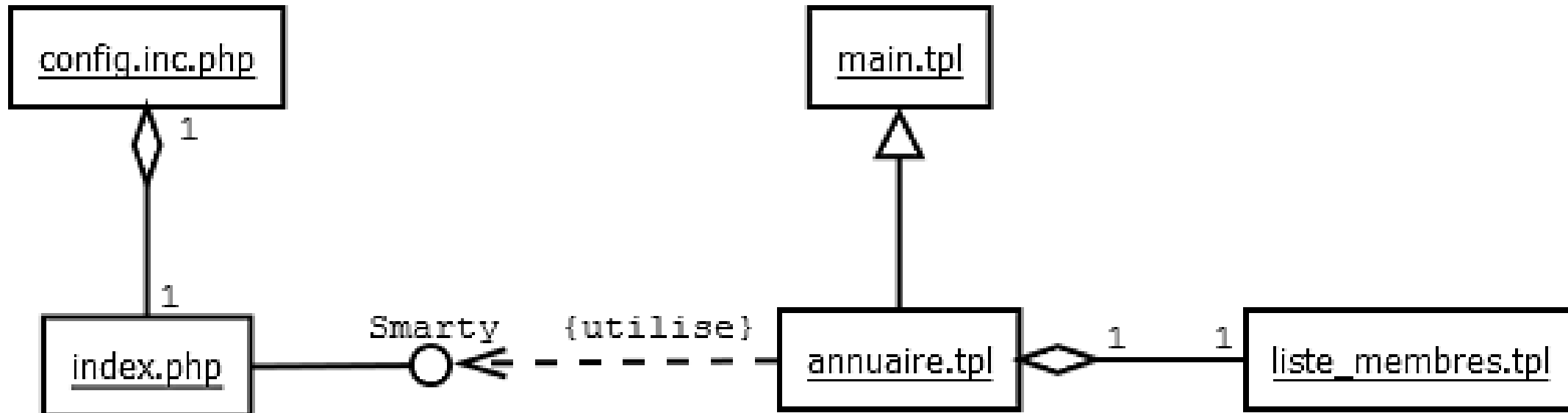
Fichier child.tpl

```
{extends file="parent.tpl"}
{block name="title"}
Page Title
{/block}
```


Smarty, utilisation de l'héritage

Retour au cas d'application : l'annuaire

Utilisation de l'héritage et de l'inclusion pour organiser les vues.



Smarty, utilisation de l'héritage

Retour au code fichier index.php

```
<?php
require_once('../libs/Smarty/Smarty.class.php');
require_once('config.inc.php');
$smarty = new Smarty;
$smarty->caching = true;
$smarty->cache_lifetime = 120;
$smarty->setTemplateDir('templates/');
$dbh = new PDO($config['bdd']['dsn'], $config['bdd']['user'], $config['bdd']['pwd']);

$filtre=(isset($_REQUEST['filtre'])?$_REQUEST['filtre']: "");
$people=$dbh->query('SELECT * from t_sampledata'.(!empty($filtre)?" where nom like '%$filtre%'" : ''))->fetchAll();
$smarty->assign("site_name",$config['site_name']);
$smarty->assign("filtre",$filtre, true);
$smarty->assign("people",$people);
$smarty->display('annuaire.tpl',$filtre);
?>
```

fichier config.inc.php

```
<?php
$config['site_name'] = "Annuaire";
$config['bdd'] = [
    'dsn' => 'mysql:host=localhost;dbname=cours__template;charset=utf8',
    'user' => 'root',
    'pwd' => 'root'
];
?>
```

Smarty, utilisation de l'héritage

Retour au code

fichier main.tpl

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>{$site_name}</title>
  </head>
  <body>
    {block
name=main_content}{/block}
  </body>
</html>
```

fichier annuaire.tpl

```
{extends 'main.tpl'}
{block name=main_content}
  <form action="" method="post">
    <input type="text" name="filtre" value="{$filtre}">
    <input type="submit" name="filtrer" value="Filtrer les noms">
  </form>
  <form action="" method="post">
    <input type="hidden" name="filtre" value="">
    <input type="submit" name="filtrer" value="Afficher toutes les entrées">
  </form>
  {include 'liste_membres.tpl' membre=$people}
{/block}
```

fichier liste_membres.tpl

```
<table>
  <caption>Membres</caption>
  <tr><th>nom</th><th>mail</th><th>téléphone</th><th>adresse</th><th>ville</th><th>code
postal</th><th>etat</th><th>entreprise</th><th>commentaire</th></tr>
  {foreach $membre as $m}
    <tr><td>{$m.nom}</td><td>{$m.mail}</td><td>{$m.telephone}</td><td>{$m.adresse}</td><td>{$m.ville}</td>
    <td>{$m.code_postal}</td><td>{$m.etat}</td><td>{$m.entreprise}</td><td>{$m.commentaire|truncate:30:'...'}</td></tr>
  {/foreach}
</table>
```

Smarty, utilisation de l'héritage

Avantages

- + Compilateur particulièrement rapide
- + Très bonne capacité de mise en œuvre de la logique de présentation
Nombreux filtres et fonction, possibilité d'étendre par plugin.
- + Système de cache efficace

Inconvénients

- Mise œuvre pas toujours évidente
Nécessite de la pratique et la documentation
- Marquage un peu trop proche de PHP (peut être un avantage, cela dépend).
- Difficile à déboguer si peu d'expérience avec ce moteur

Smarty, utilisation de l'héritage

On trouve actuellement de nombreux autres moteurs de templates, chacun ayant leurs avantages et inconvénients. On peut entre autre citer :

Twig

<http://twig.sensiolabs.org/>



Plus moderne que Smarty, un peu plus lent mais moins permissif

RainTPL

<http://www.raintpl.com/>



Très rapide, comprends le strict minimum, le tout dans un seul fichier.

Les moteurs de templates

UTILISATION AVANCÉE DES MOTEURS DE TEMPLATES

Aspect clients/serveur

Jusque là on c'est intéressé aux moteurs de templates côté serveur i.e. exécuté sur le serveur avant l'envoi de la page.

Il existe des moteurs de template côté client. On peut citer entre autre :

Handlebar <http://handlebarsjs.com/>

EJS (Embedded JavaScript) <http://embeddedjs.com/>

dust <http://akdubya.github.io/dustjs/>

pure <http://beebole.com/pure/>

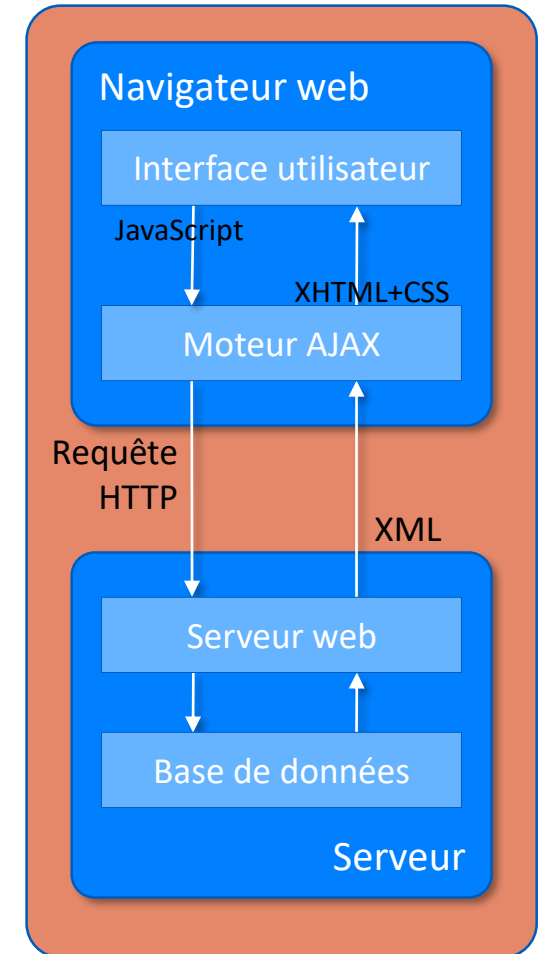
Mustache <http://mustache.github.io/>

Templates et AJAX

Ces moteurs de templates sont essentiellement utilisés en complément de l'AJAX pour intégrer plus facilement, les données récupérées via en objet XMLHttpRequest, à la page html.

Même si l'on a utilisé des templates côté serveur, il faudrait donc réécrire, avec une autre syntaxe, d'autres contraintes, etc. nos templates pour le moteur côté client...

OU PAS !



Les moteurs de templates

MUSTACHE

Mustache

<http://mustache.github.io/>



« Logic-less templates.

Available in Ruby, JavaScript, Python, Erlang, PHP, Perl, Objective-C, Java, .NET, Android, C++, Go, Lua, ooc, ActionScript, ColdFusion, Scala, Clojure, Fantom, CoffeeScript, D, and for node.js. »

Fichier initial

```
Hello {{name}}  
You have just won ${{value}}!  
{{#in_ca}}  
Well, ${{taxed_value}}, after taxes.  
{{/in_ca}}
```

YAML ou JSON

```
{  
  "name": "Chris",  
  "value": 10000,  
  "taxed_value": 10000 - (10000 * 0.4),  
  "in_ca": true  
}
```

+

Fichier final

```
Hello Chris  
You have just won $10000!  
Well, $6000.0, after taxes.
```

Mustache

Mustache utilise le format YAML ou JSON pour décrire les données. Ces données sont ensuite combinée à la vue via l'utilisation de marqueurs spécifiques.

La philosophie utilisée permet d'utiliser Mustache dans différent contexte (html, xml, script, etc..). En outre, il est aisé de l'utiliser à la sortie de requêtes de type XPath.

Dans l'exemple qui suit, on va utiliser le moteur Mustache implémenté en php pour gérer les vues côté serveur et le moteur Mustache écrit en javascript pour les interpréter côté client, lorsque cela est possible. Le tout devra utiliser les mêmes vues sans avoir à les réécrire spécifiquement pour le serveur ou pour le client.

Mustache

Retour au code – index.php

```
<?php
require '../libs/src/Mustache/Autoloader.php';
Mustache_Autoloader::register();
$m = new Mustache_Engine(array(
    'loader' => new Mustache_Loader_FilesystemLoader(dirname(__FILE__)),
));

$bd_config=[
    'dsn' => 'mysql:host=localhost;dbname=cours__template;charset=utf8',
    'user' => 'root',
    'pwd' => 'root'
];
$dbh = new PDO($bd_config['dsn'], $bd_config['user'], $bd_config['pwd']);

$filter=(isset($_REQUEST['filter'])?$_REQUEST['filter']:"");
$people=$dbh->query('SELECT * from t_sampledata'.(!empty($filter)?" where nom like '%$filter%'":""));
echo $m->render('main', array('people' => $people, 'filter' => $filter));
?>
```

Mustache

Retour au code – main.mustache

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Annuaire</title>
  </head>
  <body>
    <form action="" method="post">
      <input type="text" name="filtre" value="{{ filtre }}">
      <input type="submit" name="filtrer" value="Filtrer les noms">
    </form>
    <form action="" method="post">
      <input type="hidden" name="filtre" value="">
      <input type="submit" name="filtrer" value="Afficher toutes les entrées">
    </form>
    <table>
      <caption>Membres</caption>
      <tr><th>nom</th><th>mail</th><th>téléphone</th><th>adresse</th><th>ville</th>
        <th>code postal</th><th>etat</th><th>entreprise</th><th>commentaire</th></tr>
      {{# people}}
        <tr><td>{{ nom }}</td><td>{{ mail }}</td><td>{{ telephone }}</td><td>{{ adresse }}</td><td>{{ ville }}</td>
          <td>{{ code_postal }}</td><td>{{ etat }}</td><td>{{ entreprise }}</td><td>{{ commentaire }}</td></tr>
      {{/people}}
    </table>
  </body>
</html>
```

Mustache

Tout comme Smarty, il est possible avec Mustache de gérer des sous-templates et de les insérer par la suite. Ceux-ci sont nommés partial.

base.mustache

```
<h2>Names</h2>
{{#names}}
  {{> user}}
{{/names}}
```

user.mustache

```
<strong>{{name}}</strong>
```

index.php

```
<?php
$m = new Mustache_Engine(array(
    'loader' => new Mustache_Loader_FilesystemLoader(dirname(__FILE__) . '/views'),
    'partials_loader' => new Mustache_Loader_FilesystemLoader(dirname(__FILE__) . '/views/partials'),
));
?>
```

Mustache

Retour au code – index.php

```
<?php
require '../libs/src/Mustache/Autoloader.php';
Mustache_Autoloader::register();
$m = new Mustache_Engine(array(
    'loader' => new Mustache_Loader_FilesystemLoader(dirname(__FILE__).'/views'),
    'partials_loader' => new Mustache_Loader_FilesystemLoader(dirname(__FILE__).'/views/partials'),
));

$bd_config=[
    'dsn' => 'mysql:host=localhost;dbname=cours__template;charset=utf8',
    'user' => 'root',
    'pwd' => 'root'
];
$dbh = new PDO($bd_config['dsn'], $bd_config['user'], $bd_config['pwd']);

$filter=(isset($_REQUEST['filter'])?$_REQUEST['filter']:"");
$people=$dbh->query('SELECT * from t_sampledata'.(!empty($filter)?" where nom like '%$filter%'":""));
$people->fetchAll();

echo $m->render('main', array('people' => $people, 'filter' => $filter));
?>
```

Mustache

Retour au code fichier main.mustache

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Annuaire</title>
  </head>
  <body>
    <form action="" method="post">
      <input type="text" name="filtre" value="{{ filtre }}">
      <input type="submit" name="filtrer" value="Filtrer les noms">
    </form>
    <form action="" method="post">
      <input type="hidden" name="filtre" value="">
      <input type="submit" name="filtrer" value="Afficher toutes les entrées">
    </form>
    {{> liste_membres }}
  </body>
</html>
```

fichier liste_membres.mustache

```
<table>
  <caption>Membres</caption>
  <tr><th>nom</th><th>mail</th><th>téléphone</th><th>adresse</th><th>ville</th><th>code postal</th><th>etat</th><th>entreprise</th><th>commentaire</th></tr>
  {{# people}}
    <tr><td>{{ nom }}</td><td>{{ mail }}</td><td>{{ telephone }}</td><td>{{ adresse }}</td><td>{{ ville }}</td>
      <td>{{ code_postal }}</td><td>{{ etat }}</td><td>{{ entreprise }}</td><td>{{ commentaire }}</td></tr>
  {{/people}}
</table>
```

Mustache

[Début d'un long et passionnant dialogue entre le prof et les étudiants...]



M@ri82> Ok...Mustache est donc un moteur de templates et on s'en sert un peu comme Smarty mais il est plus simple (logic-less) et fait moins de chose...je ne vois pas le rapport avec la choucroute.



Killer_du_75> Ouai klr



Professeur> Il ne faudrait pas oublier que Mustache est écrit en PHP, Javascript, Java, Perl, C++, Ruby, Python... Il est donc techniquement possible d'utiliser les mêmes templates côté client et côté serveur.



M@ri82> Mais messieurs, je croyais que l'on ne pouvait pas lire de fichier en javascript ? Comment on va charger la vue ?

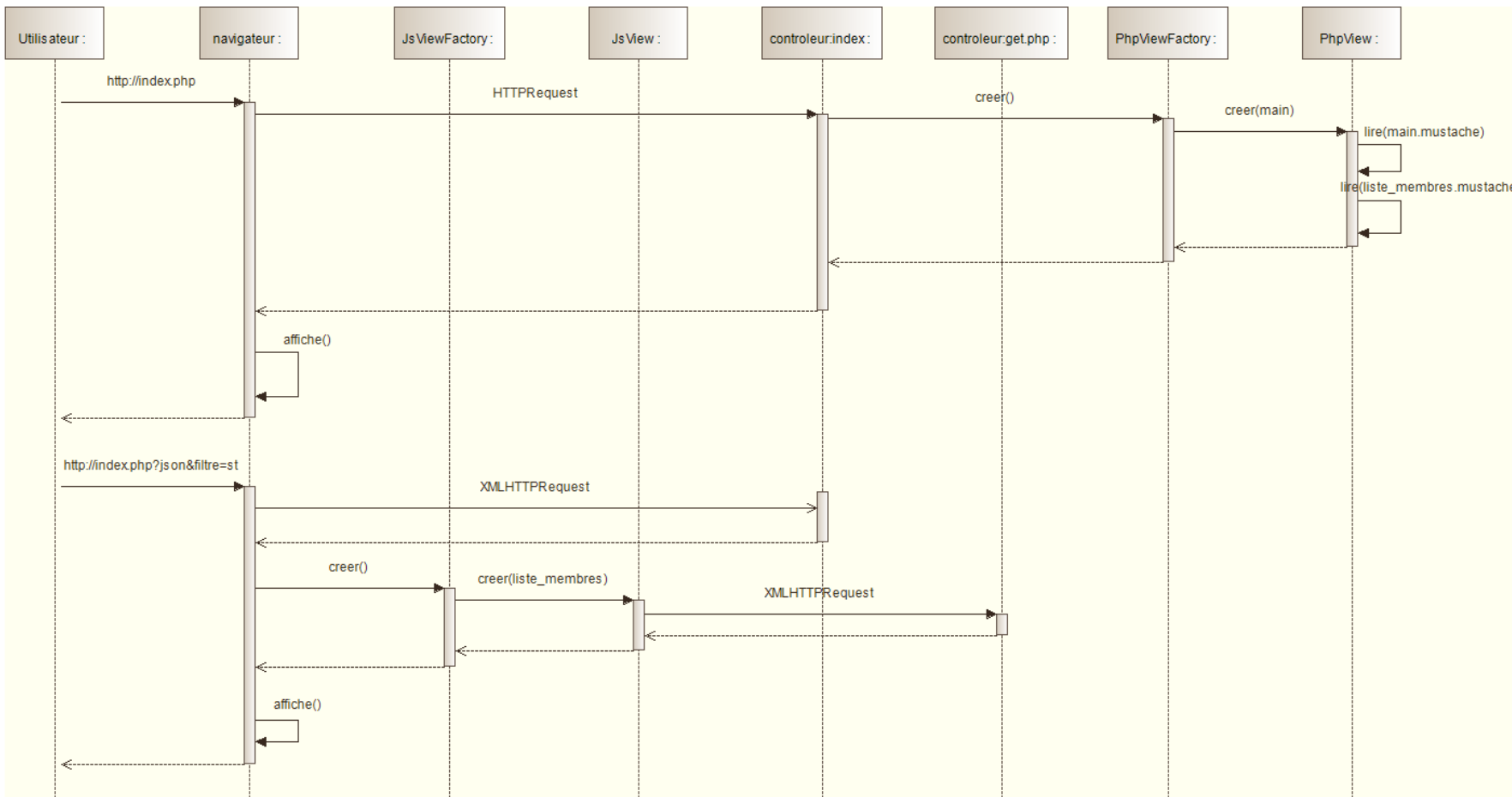


Professeur> On va écrire un script PHP qui va s'en charger. Oui, oui, PHP car le fichier est sur le serveur
« Killer_du_75 » -_- . Et on va utiliser AJAX pour appeler ce script.

Mustache

Un dessin et se sera plus clair : enfin peut-être...

Diagramme de séquence de notre exemple



Mustache

fichier main.mustache

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Annuaire</title>
    <script type="text/javascript" src="js/mustache/mustache.js"></script>
    <script type="text/javascript" src="js/sha1.js"></script>
    <script type="text/javascript" src="js/tools.js"></script>
    <script type="text/javascript" src="js/annuaire.js"></script>
  </head>
  <body>
    <form id="onFiltre" action="" method="post">
      <input type="text" name="filtre" value="{{ filtre }}">
      <input type="submit" name="filtrer" value="Filtrer les noms">
    </form>
    <form id="onDisplayAll" action="" method="post">
      <input type="hidden" name="filtre" value="">
      <input type="submit" name="filtrer" value="Afficher toutes les entrées">
    </form>
    <div id="annuaire">{{> liste_membres }}</div>
  </body>
</html>
```

fichier liste_membres.mustache

```
<table>
  <caption>Membres</caption>
  <tr><th>nom</th><th>mail</th><th>téléphone</th><th>adresse</th><th>ville</th><th>code postal</th><th>etat</th><th>entreprise</th><th>commentaire</th></tr>
  {{# people}}
    <tr><td>{{ nom }}</td><td>{{ mail }}</td><td>{{ telephone }}</td><td>{{ adresse }}</td><td>{{ville }}</td>
    <td>{{ code_postal }}</td><td>{{ etat }}</td><td>{{ entreprise }}</td><td>{{ commentaire }}</td></tr>
  {{/people}}
</table>
```

Mustache

fichier index.php

```
<?php
require '../libs/src/Mustache/Autoloader.php';
Mustache_Autoloader::register();
$m = new Mustache_Engine(array(
    'loader' => new Mustache_Loader_FilesystemLoader(dirname(__FILE__).'./views'),
    'partials_loader' => new Mustache_Loader_FilesystemLoader(dirname(__FILE__).'./views/partials'),
));
$bd_config=[
    'dsn' => 'mysql:host=localhost;dbname=cours__template;charset=utf8',
    'user' => 'root',
    'pwd' => 'root'
];
$dbh = new PDO($bd_config['dsn'], $bd_config['user'], $bd_config['pwd']);
$filtre=(isset($_REQUEST['filtre'])?$_REQUEST['filtre']: "");
$people=$dbh->query('SELECT * from t_sampledata'.(!empty($filtre)?" where nom like '%$filtre%':""));
$data = array('people' => $people, 'filtre' => $filtre);

if (isset($_REQUEST['json'])) echo json_encode($data);
else echo $m->render('main', $data);
?>
```

fichier get.php

```
<?php
$filename=(isset($_REQUEST['view'])?$_REQUEST['view']: "");
if (!empty($filename)) {
    if (file_exists("views/partials/$filename")) include "views/partials/$filename";
}
?>
```

Mustache

fichier annuaire.js

```
function displayAll(event) {
    TOOLS.stopEvent(event);
    TOOLS.getCached('get.php?view=liste_membres.mustache', function (tpl) {
        TOOLS.XMLHttpRequest("index.php?json", null, function (json) {
            data=JSON.parse(json);
            tbl=document.getElementById("annuaire");
            form=document.getElementById("onFiltre");
            form["filtre"].value='';
            tbl.innerHTML=Mustache.render(tpl,data);
        }, {method: 'post', asynchrone: true});
    });
}

function displayFiltre(event) {
    TOOLS.stopEvent(event);
    form=document.getElementById("onFiltre");
    TOOLS.getCached('get.php?view=liste_membres.mustache', function (tpl) {
        TOOLS.XMLHttpRequest("index.php?json&filtre="+form["filtre"].value, null, function (json) {
            datas=JSON.parse(json);
            tbl=document.getElementById("annuaire");
            tbl.innerHTML=Mustache.render(tpl,datas);
        }, {method: 'post', asynchrone: true});
    });
}

function main() {
    TOOLS.addListener(document.getElementById('onDisplayAll'),'submit',displayAll);
    TOOLS.addListener(document.getElementById('onFiltre'),'submit',displayFiltre);
}

TOOLS.addListener(window, 'load', main);
```

Mustache

Avantages

- + Permet d'uniformiser les traitements sur les vues côté client et serveur
- + Pas permissif (logic-less)
- + Très simple à déboguer

Inconvénients

- Pas de logique de présentation (logic-less)
- Nécessite la mise en place d'un cache côté client (html5 peut combler ce problème) pour diminuer les accès serveur.

Le mot de la fin

Mustache est limité dans le cadre d'une approche où l'on mixe le rendu côté serveur et côté client. Néanmoins, cela permet de se rendre compte de ce que l'on peut faire.

En outre, il existe des portage Javascript de :

Smarty : jSmart (<https://code.google.com/p/jsmart/>) ;

Twig : twig.js (<https://github.com/justjohn/twig.js>).

POUR ALLER PLUS LOIN

AngularJS

<https://angularjs.org/>

Framework Javascript Open-source qui étend le langage HTML.

Permet de replacer la logique de présentation là où elle doit-être : sur le client.

Très utilisé pour les SPA (Single Page Application) mais pas seulement

Orienté vers la programmation déclarative