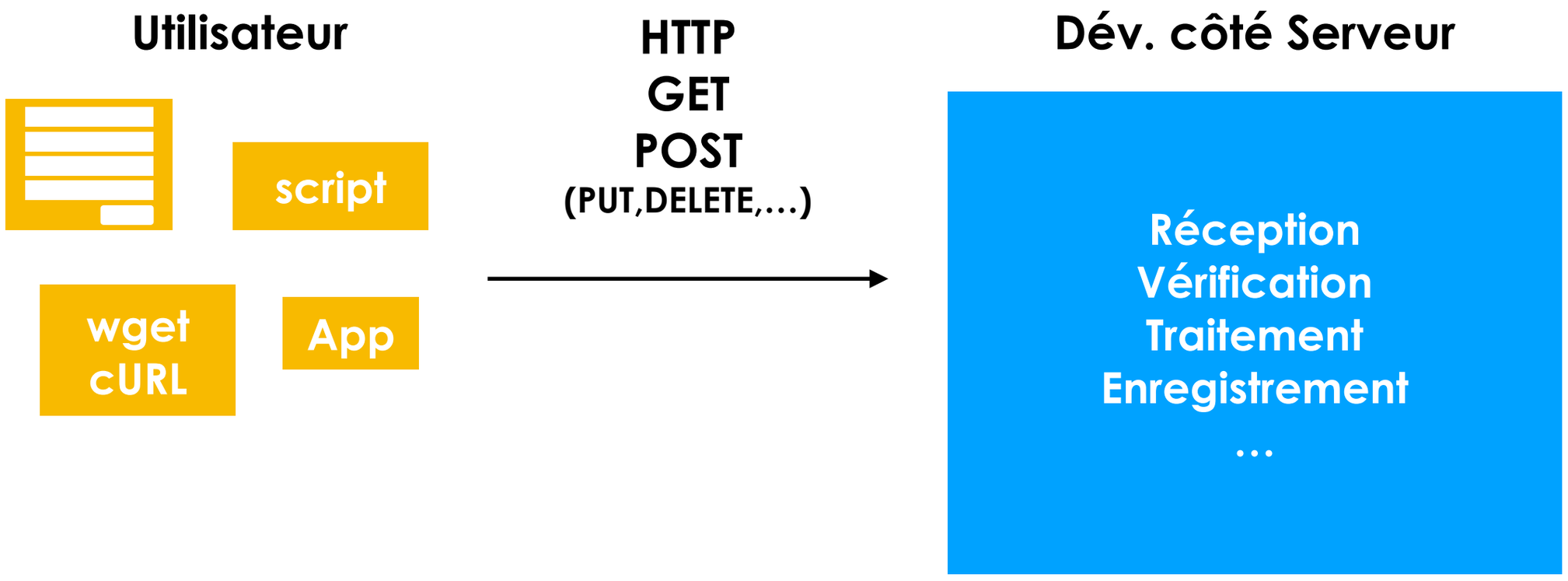
M3104

PHP
Développement Web côté serveur

david.durand@u-picardie.fr

Transmission de paramètres au script PHP

- permettre la saisie de données par l'utilisateur
- envoyés vers le serveur WEB pour traitement : authentification, enregistrement base de données, etc…



Formulaires
& Protocole HTTP

- GET : envoi des paramètres dans l'URL de la requête

```
GET  /script.php?nom=Smith&prenom=John
```

- POST : envoi des paramètres *après* la requête

```
POST /script.php
User-Agent:Mozilla 5.0 (Win NT ….)… OPR…
\n\n
nom=Smith&prenom=John
```

GET ou POST ?

- GET
  - paramètres visible dans l'URL
  - utilisés souvent pour passer des options à la page (tri, n° page, terme recherché)
    - limitée en taille
- POST
  - non visible dans l'URL
  - envoi possible de longs textes, fichiers, mots de passe
  - limite en taille = config du serveur, souvent >= 2Mo

Côté HTML

- Côté utilisateur : champs de formulaire HTML : texte, liste, cases à cocher, dates, données brut (fichiers)

- Possibilité de préciser le type de requête GET ou POST

```
<form action="url_page_de_traitement" method="GET/POST">
<input type="text" name="nom">
<input type="text" name="matiere">
<input type="date" name="limite">
<input type="submit" name="bouton" value="envoi">
</form>
```

Côté Serveur

- en PHP, deux tableaux associatifs contiennent respectivement les variables passées en GET et POST

```php
<?php
echo $_GET['nom'];
echo $_GET['matiere'];
echo $_GET['limite'];


?>
```

```php
<?php
echo $_POST['nom'];
echo $_POST['matiere'];
echo $_POST['limite'];


?>
```

```html
<form action="url_page_de_traitement" method="GET|POST">
<input type="text" name="nom">
<input type="text" name="matiere">
<input type="date" name="limite">
<input type="submit" name="bouton" value="envoi">
</form>
```

Formulaires
Côté Serveur

- Le tableau _REQUEST contient toutes les variables passées, GET ou POST

- Afficher toutes les variables transmises :

```
<?php
foreach($_REQUEST as $variable => $valeur)
  echo "<p>$variable : $valeur</p> ";

?>
```

```
<?php
print_r($_REQUEST);  // ou _GET, _POST
?>
```

Notion de framework

- PHP est un langage de script, à l'origine purement **procédural**.

- Habitudes de développement :
  - la structure du code de chaque page est calquée sur le rendu HTML souhaité

  - le code mélange différents aspects du projet : code html, code de traitement php, code d'accès à la base, javascript, css, contrôles d'accès, etc.

(Mauvaises) habitudes de code

recherche.php

```
<?php
include(...);
mysql_connect(...);
?>
<html>
...

<body>
     Recherche
</body>
</html>
```

resultat.php

```
<?php
include(...);
mysql_connect(...);
?>

<html>
...

<body>
     resultats de la recherche
</body>
</html>
```

index.php

```
<?php
include(...);
mysql_connect(...);
?>
<html>
...

<body>
     Page d'accueil
</body>
</html>
```

form_inscr.php

```
<?php
include(...);
mysql_connect(...);
?>
<html>
...

<body>
     Inscription
</body>
</html>
```

inscription.php

```
<?php
include(...);
mysql_connect(...);
mysql_query(...)
header("Location:index.php");
?>
```

(Mauvaises) habitudes de code

Liste des membres

Pseudo	Inscription
Spiderman	12/01/2001
Superman	24/05/2008
Batman	04/12/2003
Ironman	31/07/2007
...	
...	
...	

=

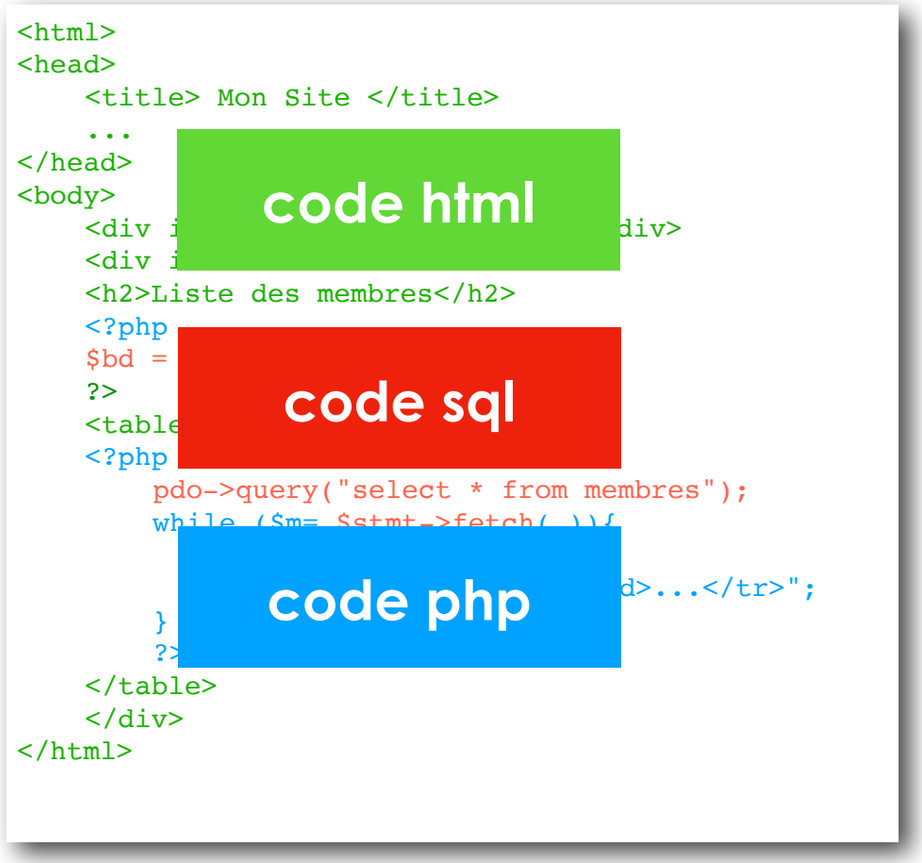
```
<html>
<head>
   <title> Mon Site </title>
   ...
</head>
<body>
   <div id='top'><h1>MonSite</h1></div>
   <div id='contenu'>
   <h2>Liste des membres</h2>
   <?php
   $bd = new PDO(...);;
   ?>
   <table>
   <?php
      pdo->query("select * from membres");
      while ($m= $stmt->fetch(…)){

         echo "<tr><td>${m[0]}</td>...</tr>";
      }
      ?>
   </table>
   </div>
</html>
```

Inconvénients

- beaucoup de code redondant (initialisations, code html, ...)
- difficulté de travailler en équipe
- maintenance de code fastidieuse

Remède

- identifier les parties du code à «factoriser»
- séparer en couches les différents aspects du code : il s'agit de s'orienter vers l'approche **MVC** (model-view-controller)

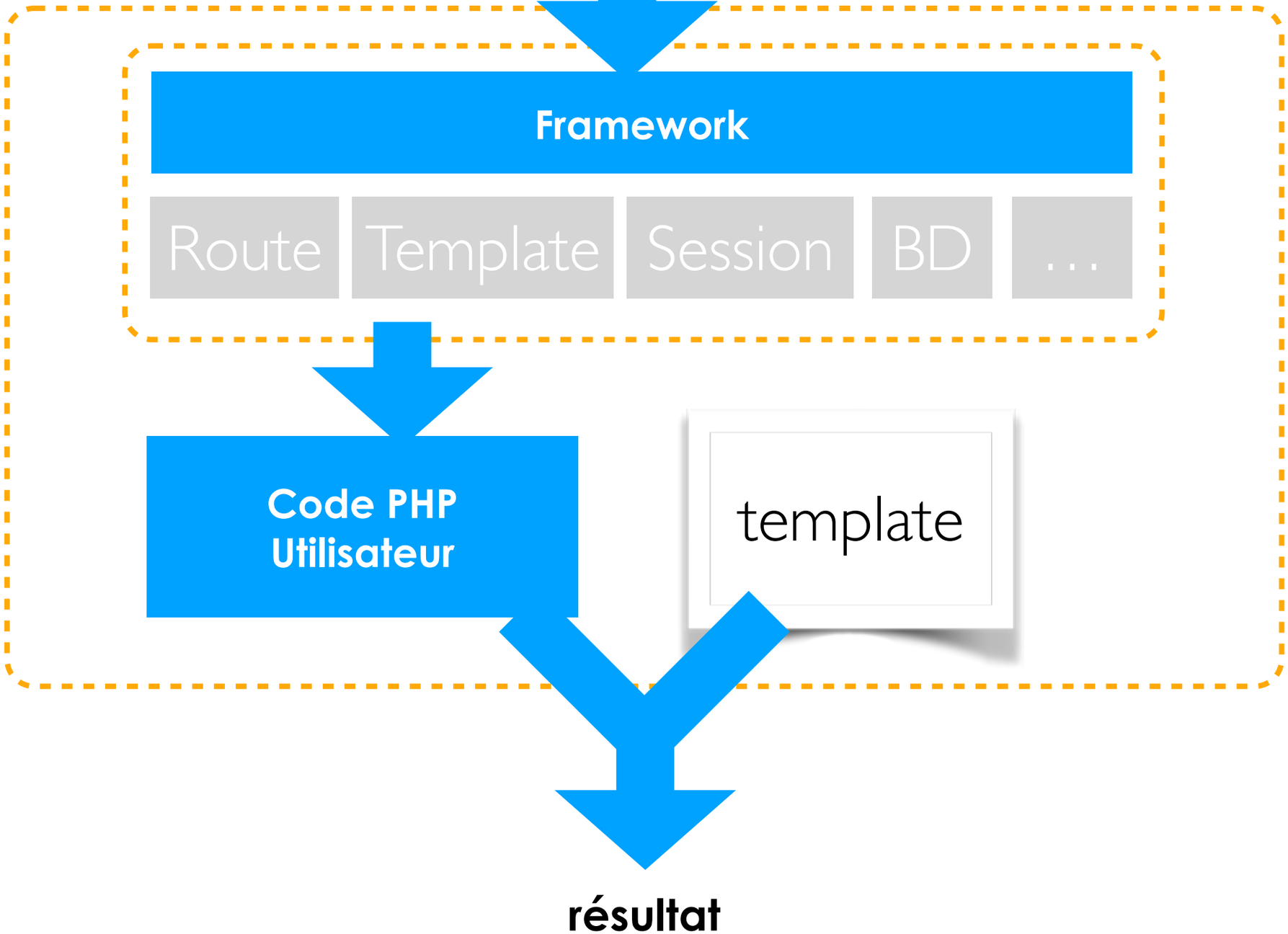


Framework

- **Objectif** :
  - éliminer le code et les tâches redondants
  - utiliser une structure de site qui gère les tâches d'initialisation et d'inclusions
  - gérer le chargement et la navigation des pages
  - séparer le rendu du code à l'aide de templates

Framework : principe

http://monsite.org/**index/recherche/liste**



Exemple : *Flight*

- Framework léger
- Associe une **route** à une **fonction**

```php
require 'flight/Flight.php';

Flight::route('/', function(){
  echo 'hello world!';
});

Flight::start();
```

- Doc : https://flightphp.com/learn

Exemple : *Flight*

```
http://monsite.org/index/recherche/liste
http://monsite.org/index/profil
http://monsite.org/index/inscription
http://monsite.org/index/user/david/9786
```

index.php?page=user&id=9786

~~echo~~

```php
// Assign template data
Flight::view()->assign('name', 'Bob');

// Display the template
Flight::view()->display('hello.tpl');
```