Programmation Internet
Cours 3: structuration des applications WEB

Jean-Michel Ilié

-

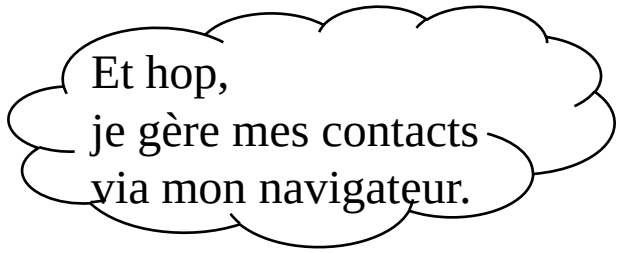
IUT Paris 5 - département informatique

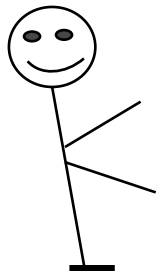
Construction modulaire

Une application web => développement d'un ensemble cohérent de services applicatifs, sous forme de **scripts serveurs** (PHP,…).

- Toute requête HTTP issue du navigateur correspondra à une demande d'exécution d'un service.
- Rappel : un service peut nécessiter plusieurs phases temporelles d'exécution correspondant chacune à un affichage sur le navigateur web.

Exemple de services





use case

Services à développer

- ✓ identification
- ✓Recherche d'une liste de contacts
- ✓Ajout d'un contact
- ✓Mise à jour [c]
- ✓Suppression [c]

Structuration des applications en services

Distinguer différents blocs de code :

- **1 Bloc de code pour chaque service de l'application**
  Les services sont développés dans des fichiers distingués éventuellement sous forme de fonctions PHP.

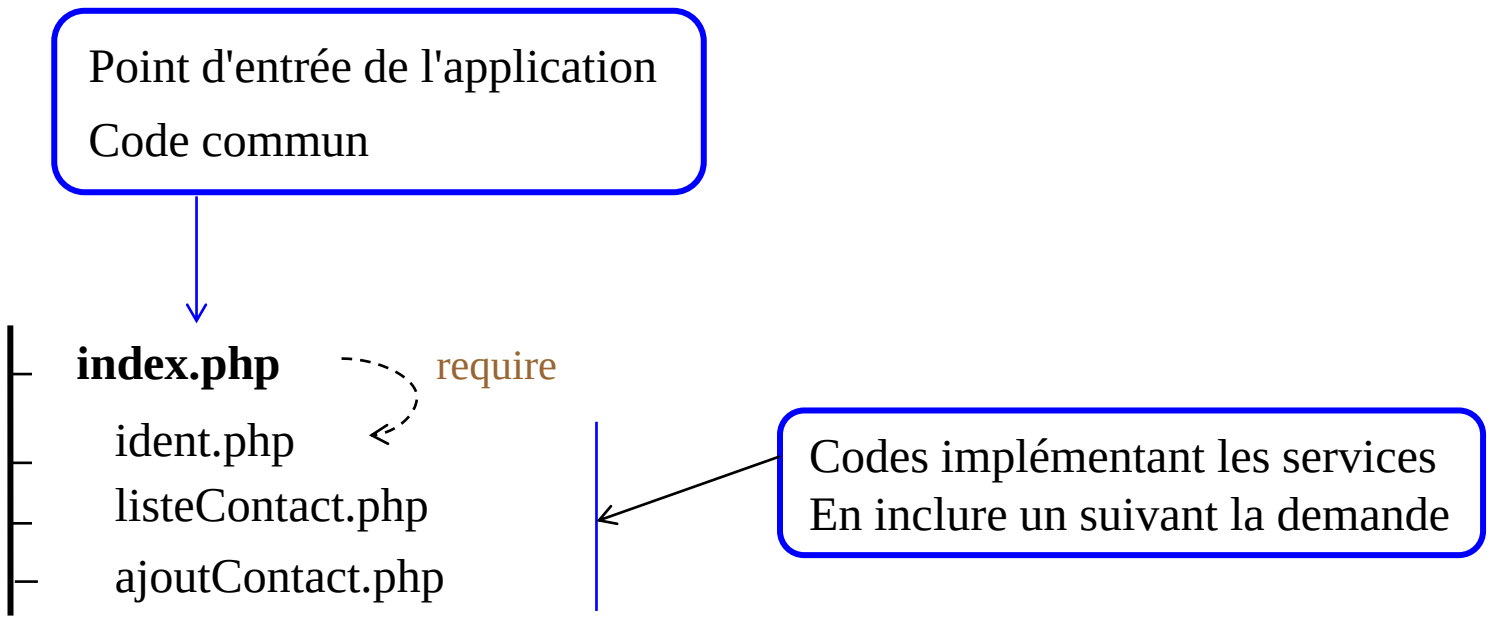
- **1 Bloc de code commun servant de frontal pour le navigateur**
  (fichier `index.php`)
  - ☞ Faire les traitements communs aux différents services :
    vérifier les accès aux services…
  - ☞ Déterminer le service désiré en fonction de la requête http
  - ☞ inclure le fichier implémentant le service
  - ☞ (si décrit en fonction) appel de la fonction du service

Structuration élémentaire en PHP

- un **fichier principal PHP par service de l'application**

- **un script générique servant de frontal** avec le navigateur
  (nom usuel : *index.php*)
  - pour réaliser les traitements communs et
  - inclure le fichier correspondant au service désiré

Exemple de structuration



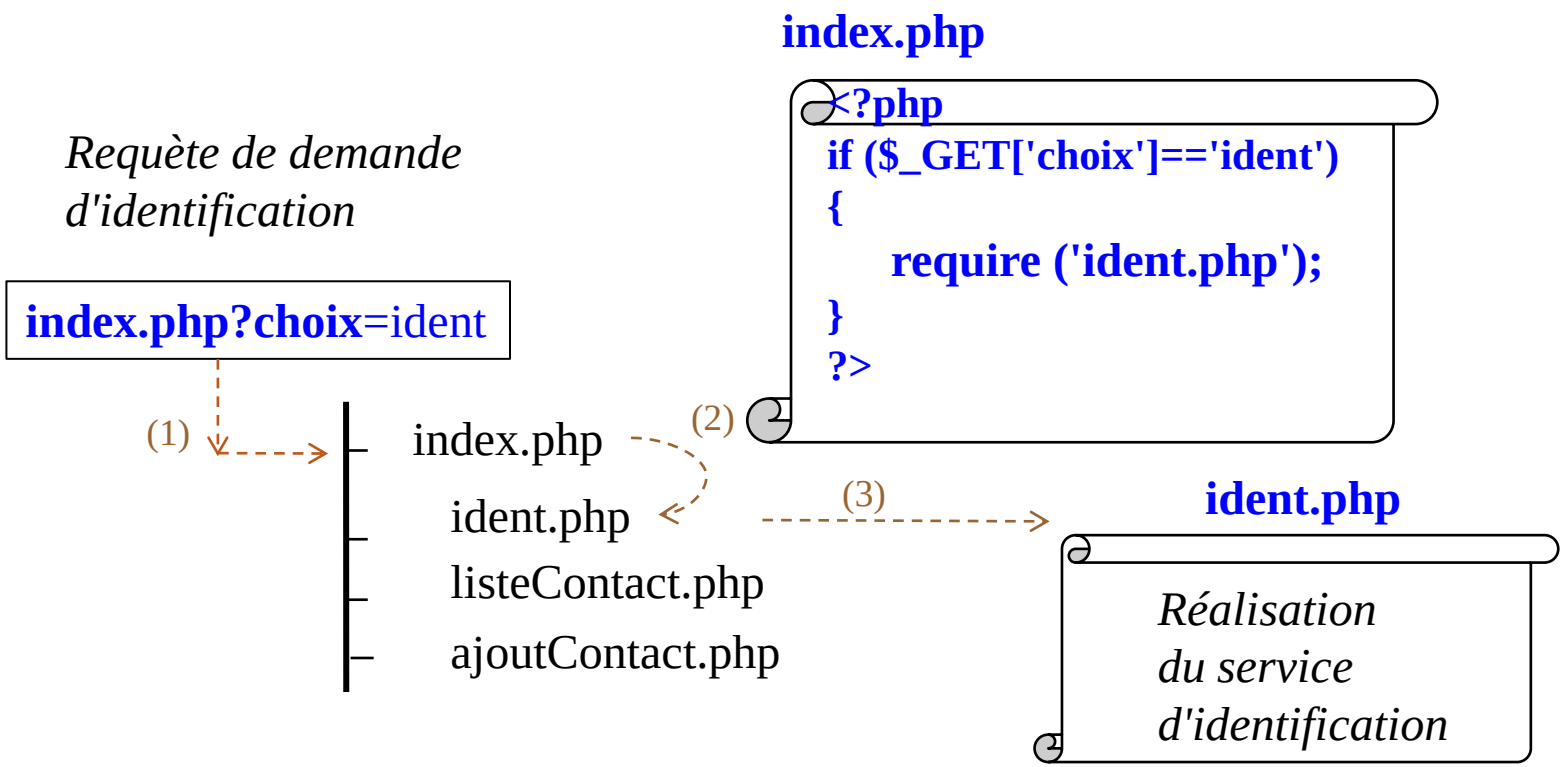
Requète HTTP d'exécution des services

- Toute requête HTTP à l'application invoque index.php
- Les requêtes sont paramétrés pour préciser le service et transmettre des données servant à la réalisation du service .

URL : **index.php?choix**=nom_du_service**&**data=valeur

- Les paramètres sont accessibles dans le tableau $_GET
- Rappel : les super-tableaux comme $_GET sont directement accessibles dans tout le code du script, y compris dans les fonctions.

Exemple d'invocation d'un service



Note : à tout niveau : $_GET et $_POST sont accessibles

Exemple de script de service invoquant index.php

Tant que l'identification n'est pas faite :
- *le formulaire de saisie est affiché*
- *le script ident.php analyse les données saisies*

(Prévoir l'affichage initial du formulaire vierge – cas count($_POST)==0).

```
//ident.php
<?php
        if  (count($_POST)==0)
           require('ident.tpl') ;
         else
            if  (! verif_ident($nom,$pass)) {
               $msg="erreur";
                require('ident.tpl') ;
               }
              else
header('Location:index.php?choix=accueil') ;
   }
?>
```

Template d'affichage du formulaire avec action=ident.php.

redirection vers la page d'accueil de l'application, via le navigateur.

Avantages et inconvénients de la structuration élémentaire en services

- conception et maintenance séparée des services
- interface unifiée (index.php)

- pas de séparation formalisée entre partie métier (l'accès aux données de l'application) et leur présentation (le html)

- aspect fichier : pas de regroupement de codes de services, comme avec les fonctions.

Principes de structuration MVC

Principe général de **séparation du code** :

- ☞ M : modele (Accès BD)
- ☞ V : vue (Affichage)
- ☞ C : contrôle regroupant les actions (utilise M et V)

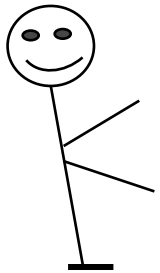
Principes de codage :

- Un fichier index.php pour l'interface avec le navigateur, **frontal**
- Développer les **services** en les regroupant par objets-métier (sur la base d'un ensemble de fonctions dites actions).
-Séparer le code de présentation et le code d'accès aux données métiers .

Exemple de services

Et hop,
je gère mes contacts
via mon navigateur.

Je suis administrateur
J'en fait profiter les
copains.



use case

Gestion des contacts

- ✓Ident
- ✓Recherche d'une liste de contacts
- ✓Ajout d'un contact
- ✓Mise à jour [c]
- ✓Suppression [c]

Gestion des utilisateurs

- ✓Ajout d'un utilisateur
- ✓Mise à jour [u]
- ✓Suppression [u]

Exemple d'objets-métiers à considérer
données **utilisateur**,
données **contact**, ….

Implémentation MVC en PHP

Des **scripts d'action** isolent le contrôle des services et exploitent des fonctions-**modèles** et **template d'affichage**

- C : ensemble de fonctions-service dites **actions**, regroupées en fichiers/classes de "**contrôle**",
- M : Ensemble de fonctions utilitaires, dites **modèles**, encapsulant les **accès SQL** aux tables de la base de données,
- V : Ensemble de fichiers **template d'affichage** (.tpl),

Schéma général d'un service web MVC

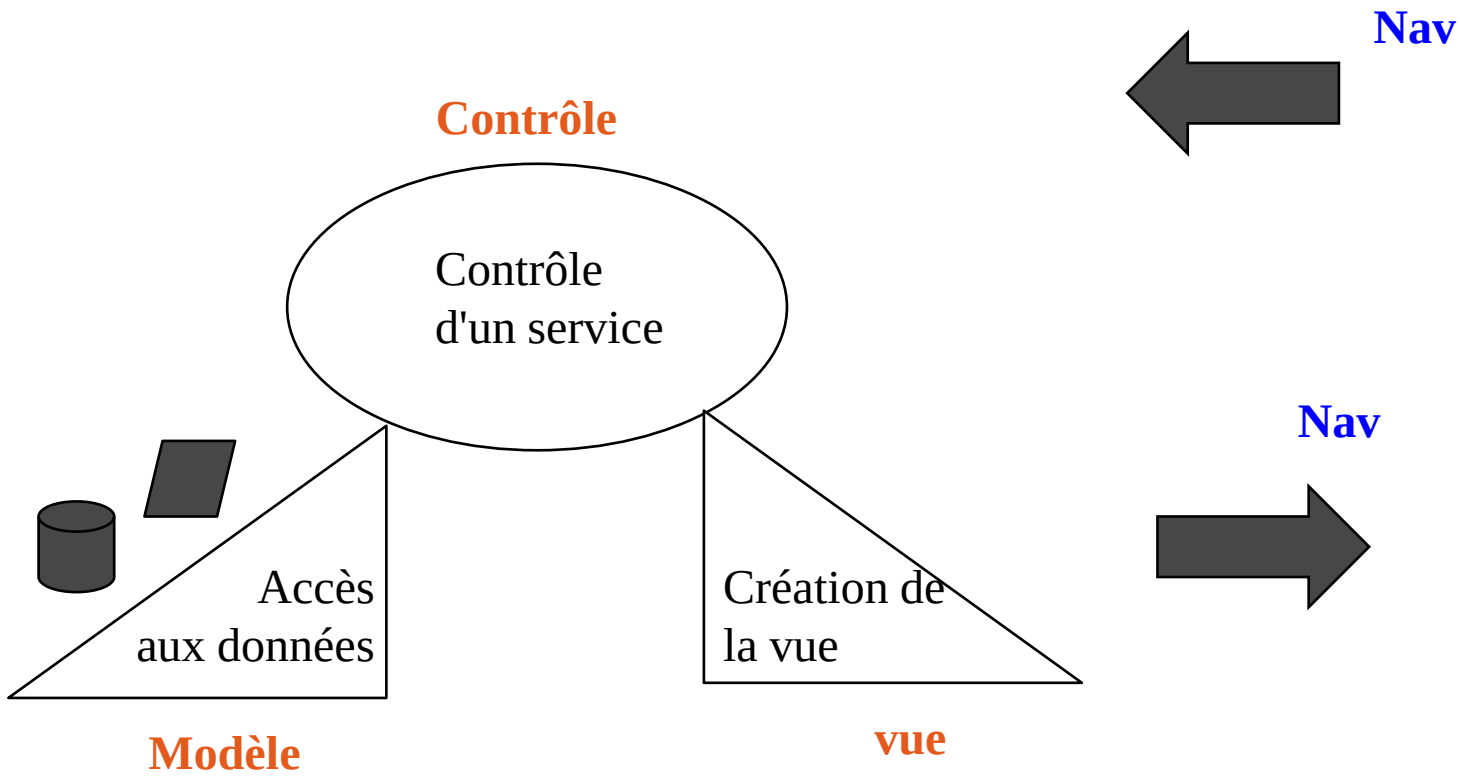
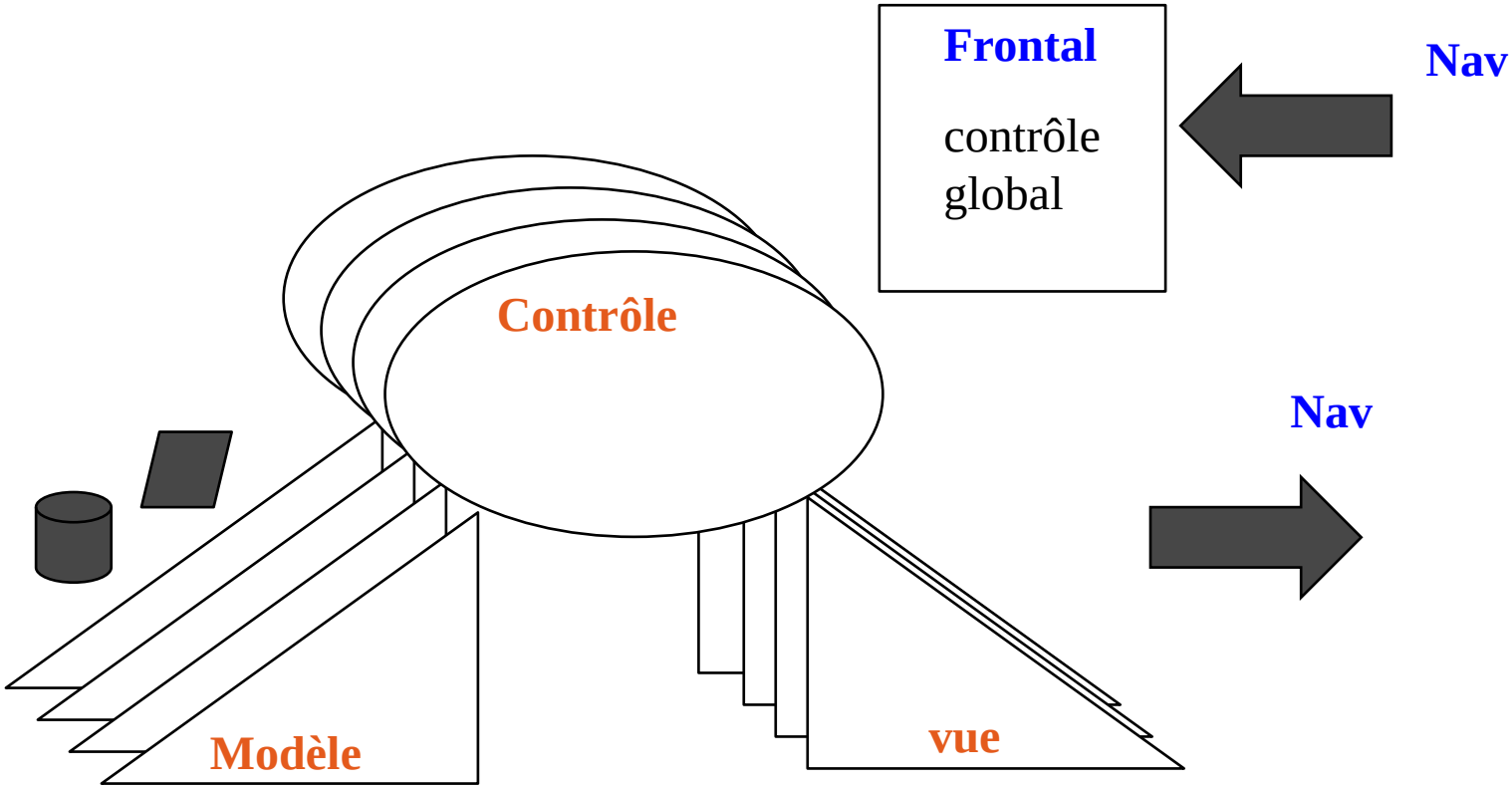
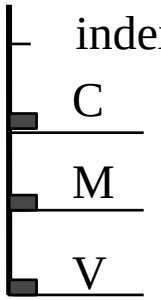


Schéma général d'une application web MVC



Principes de structuration d'une application MVC pour PHP

- **index.php** (frontal)
  pour l'interface avec le navigateur
- **3 répertoires** pour M, V, C



regroupe les fonctions implémentant les services (actions)

regroupe les fonctions modele

regroupe les template associés aux actions

- regroupement possible des fonctions par objet-métiers.

Structuration de l'application par regroupement des services

Les services sont développés en fonctions/méthodes dans des fichiers/classes distingants les objets métiers.

- gestion des **u**tilisateurs

Utilisateurs.php

function **ident**()
function **ajout_u**()
function **maj_u**()
function **destr_u** () …

- gestion des **c**ontacts d'un utilisateur :

Contact.php

function **liste_contacts**()
function **ajout_c**()
function **maj_c**()
function **destr_c** () …

Développement d'un service (action)

- Une ***action*** est la réalisation d'un service de l'application

  - **traiter les paramètres d'entrée** du service
  - **inclure un ou plusieurs fichiers modèle**
  - **initialise des variables PHP**
    en retour des **appels de fonctions modèles**
    (accès à une base, ….)
  - **inclur le template d'affichage** correspondant à l'action
    (dont possiblement, les affichages des valeurs des variables PHP).
  - **alternativement, rediriger l'exécution vers un autre service**.

Développement d'un service (action)

- Une ***action*** est la réalisation d'un service de l'application

  - **traiter les paramètres d'entrée** du service
  - **inclure un ou plusieurs fichiers modèle**
  - **initialise des variables PHP**
    en retour des **appels de fonctions modèles**
    (accès à une base, ….)
  - **inclur le template d'affichage** correspondant à l'action
    (dont possiblement, les affichages des valeurs des variables PHP).
  - **alternativement, rediriger l'exécution vers un autre service**.

! pour des questions de séparation de code claire entre M et V. le template **ne doit pas** invoquer directement les fonctions du modèle. C'est le rôle de l'action de faire l'intermédiaire.

Invocation des actions de l'application

Chaque URL doit avoir suffisamment d'information pour indiquer l'action à exécuter

- URL : prévoir au moins 2 paramètres spécifiques :
  - **controle** : relatif au fichier de contrôle, à inclure
  - **action** : relatif à l'action à lancer, au sein d'un fichier de contrôle
  - d'autres paramètres correspondent à des données pour le service.

Exemples :

```
http://...../index.php?controle=utilisateur&action=ident
http://...../index.php?controle=contact&action=listeContact&idU=3
```

Rôle (principal) du script d'interface : index.php

- **index.php** :
  - détermine le service invoqué en fonction des paramètres de l'URL
  - inclut le fichier de contrôle correspondant.
  - lance le service .

Url de requête avec 2 paramètres :
- contrôle : fichier développant des services
- action : nom d'une fonction contrôlant un service

```
//index.php
<?php
        $controle = $_GET['controle'];
        $action = $_GET['action'];
;

        require ("./C/" . $controle . ".php");
        $action();
?>
```

Exemple de script MVC service liste_contact

Action : lister les contacts d'un utilisateur

// ./C/contact.php

```
<?php
   function liste_contact() {
         $idnom = $_GET['id_U'];
         require ("M/contact_BD.php");
         $L =  read_Contacts($idnom);
         require ("V/liste-contact.tpl");
   }
?>
```

-1- Inclure un modèle

2- Inclure un template

$L : variable intermédiaire entre modèle et vue

// ./M/contact_BD.php

```
<?php
function read_Contacts($nom) {
      $req = "select * from contacts
                where NOM=`$nom`";
               ....
               return $Contacts;
}?>
```

// ./V/contact.tpl

```
<html><body><ul>
<?php foreach ($L as $c)
         echo "<li>" . $c['nom'] "</li>";
?>
</ul></body></html>
```

Exemple de structuration MVC pour 2 services