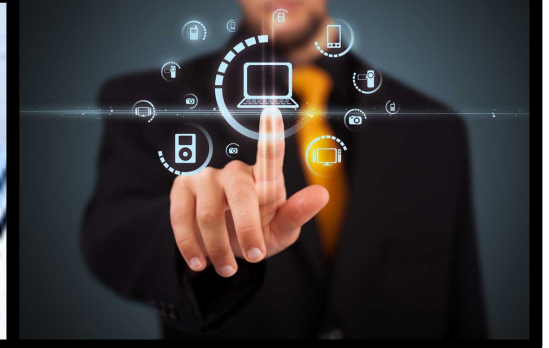


UNIVERSITE DE CORSE
Licence SPI 2ème année
Option Informatique

UE Programmation WEB Back End
CH2 - Organisation du code php



Evelyne VITTORI
vittori@univ-corse.fr

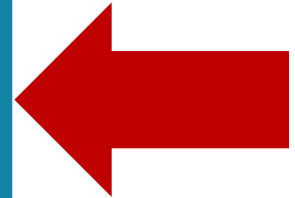
Plan du cours



CH1 – Fondamentaux
Programmation php



CH2 – Structuration et
organisation du code



CH3 – POO en php

Problématique de l'organisation du code php

- Comprendre comment structurer le code d'un projet php afin:

- De limiter la redondance de code

Principe DRY pour "Don't Repeat Yourself"

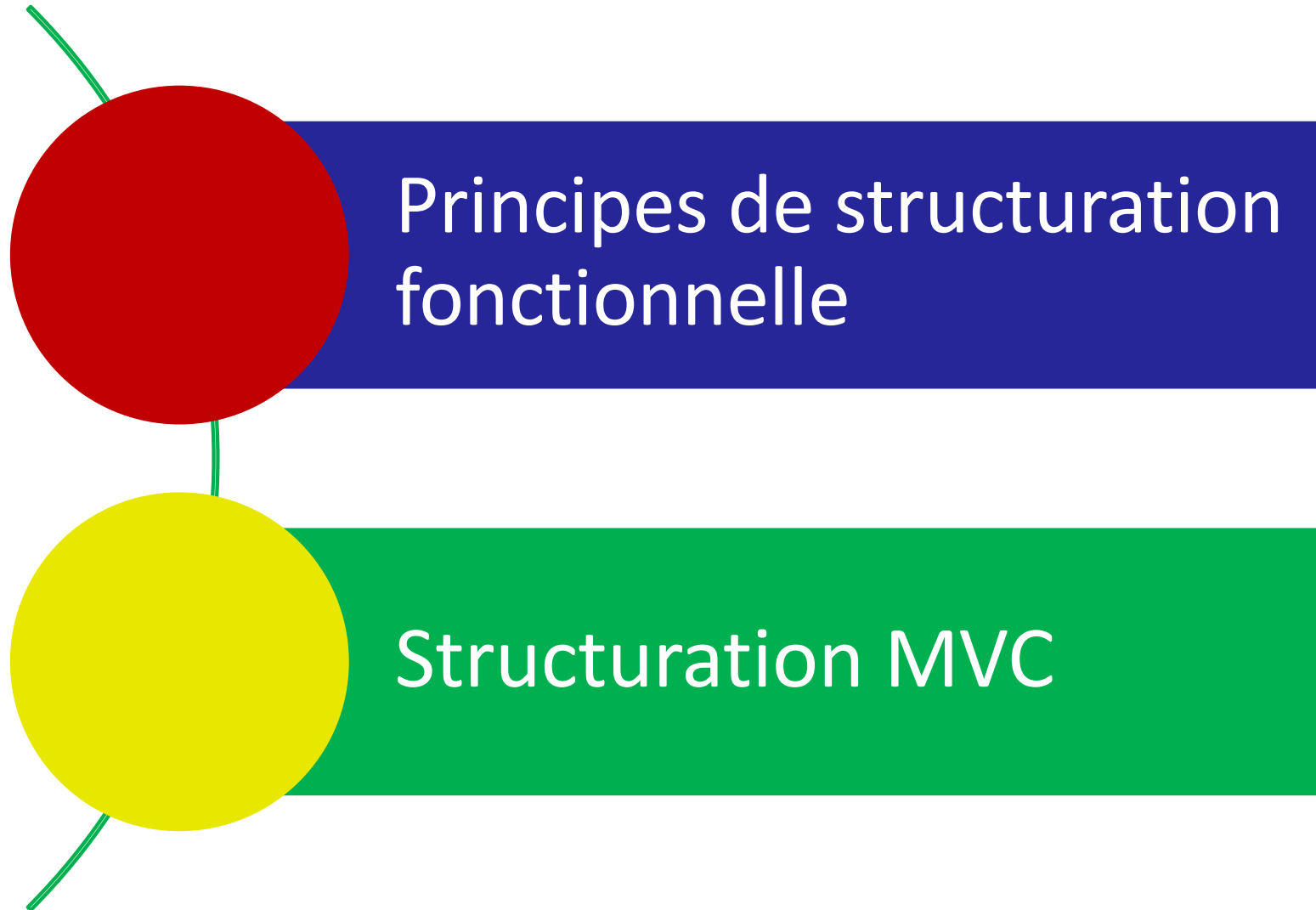
- D'assurer l'évolutivité de l'application

- Plusieurs approches possibles:

- Structuration fonctionnelle simple
 - Structuration MVC
 - *Structuration orientée objet et MVC OO*

Détaillée au
CH3

CH2 – Structuration et Organisation du code





1 - Principes de structuration fonctionnelle



Principes de structuration fonctionnelle

- Définir plusieurs fichiers et utiliser des includes (ou require):
 - `include` ('nomFichier.php') ou `require`('nomFichier.php')
 - `include_once`(« nom fichier php ») ou `require_once`('nomFichier.php')
 - Permet d'éviter les problèmes de duplication de fonctions en cas d'include multiples

`require` génère une erreur bloquante si le fichier à inclure n'est pas disponible

Principes de structuration fonctionnelle

- Isoler le code de connexion à la BD
 - Définir un fichier de configuration avec des constantes utiles
- Séparer au maximum le html du php
 - Définir un squelette de page réutilisable (avec des variables)
- Assurer un traitement des erreurs standardisé
- Définir des fichiers regroupant les fonctions par spécialité:
 - Accès à la BD

Fichier de configuration

- Fichier contenant des constantes utiles
(exemple : informations de connexion à la BD)

```
<?php

const DB_HOST = "localhost";
const DB_PORT = "8889";
const DB_NAME = "maBD";
const DB_USER = "root";
const DB_PWD = "root";

?>
```


Connexion à la base de données

- Isoler le code de connexion à la BD dans une fonction:

```
function getDatabase() {  
    static $db = null; //variable initialisée une seule fois  
    if ($db == null) {  
        $dsn = 'mysql:host=' . DB_HOST . '  
                ;port=' . DB_PORT . '  
                ;dbname=' . DB_NAME . ';charset=utf8';  
        $db = new PDO($dsn, DB_USER, DB_PWD);  
        // lever une exception si erreur  
        $db->setAttribute(PDO::ATTR_ERRMODE,  
            PDO::ERRMODE_EXCEPTION);  
    }  
    return $db;  
}
```

Remarque

le fichier php contiendra la ligne :
require 'config/db.php';

Définir un squelette de page paramétré (notion de template)

bonjour.php

```
<?php
$titre = "Un exemple
de page PHP simple";
$contenu = "Bonjour le monde";

include("squelette.php");
?>
```

HTML « à trous » avec des écho de variables php:

- \$titre
- \$contenu

Balise echo abrégée
équivalent de
<?php echo \$titre; ?>

Un exemple de page PHP simple

Bonjour tout le monde

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="utf-8" />
    <title><?php echo $titre; ?></title>
  </head>
  <body>
    <header>
      <h1><?= $titre ?></h1></a>
    </header>
    <div>
      <?= $contenu ?>
    </div>
  </body>
</html>
```

squelette.php

Traiter les erreurs de manière standardisée

- Fonction d'affichage des erreurs

```
function error($code, $msg) {  
    if ($code == XX) {  
        http_response_code(XX);  
        require 'vueErreurXX.php';  
    } else {...}  
    die();  
}
```

- Appel de la fonction **error**(xx, "mon message")

- Dans le traitement d'une exception

```
catch (Exception $e) {  
    error(xx, $e->getMessage()); ....
```

- Directement dans une fonction



2 - Architecture MVC

Une première approche
fonctionnelle

Exemple d'illustration


Application de gestion des produits

- Base de données
 - Une seule table Produits
- Fonctionnalités
 - Affichage de la liste des produits
 - Affichage d'un produit
 - Recherche d'un produit

Code à récupérer sur l'ENT :
exempleMVCCH2.zip

Base de données **ecommerce**

Une seule table Produit

Nom	Type
id_produit 	int(11)
titre	varchar(254)
descriptif	text
stock	int(11)
prix	decimal(10,0)
fabricant	varchar(254)

id_produit	titre	descriptif	stock	prix	fabricant
1	The 7th Continent Classic Edition	L'élément indispensable pour jouer au jeu. Elle vo...	145	59	Serious Poulp
2	7 Wonders	Prenez la tête de l'une des sept grandes cités du ...	54	43	Repos Production
3	Titles	Un jeu d'ambiance et de culture générale pour tous...	10	25	Vinca
4	Skull & Roses	Jeu d'ambiance, bluff, déduction : réussir deux dé...	5	16	Asmodee

Base de données eCommerce

- Le script de création de la base est donné dans le fichier bdeCommerce.sql

```
create database if not exists ecommerce;
use ecommerce;

drop table if exists produits;

create table produits (
  id_produit integer primary key auto_increment,
  titre varchar(254) not null,
  descriptif text not null,
  stock int not null,
  prix decimal not null,
  fabricant varchar(254) not null
) ENGINE=InnoDB DEFAULT CHARSET=utf8;;

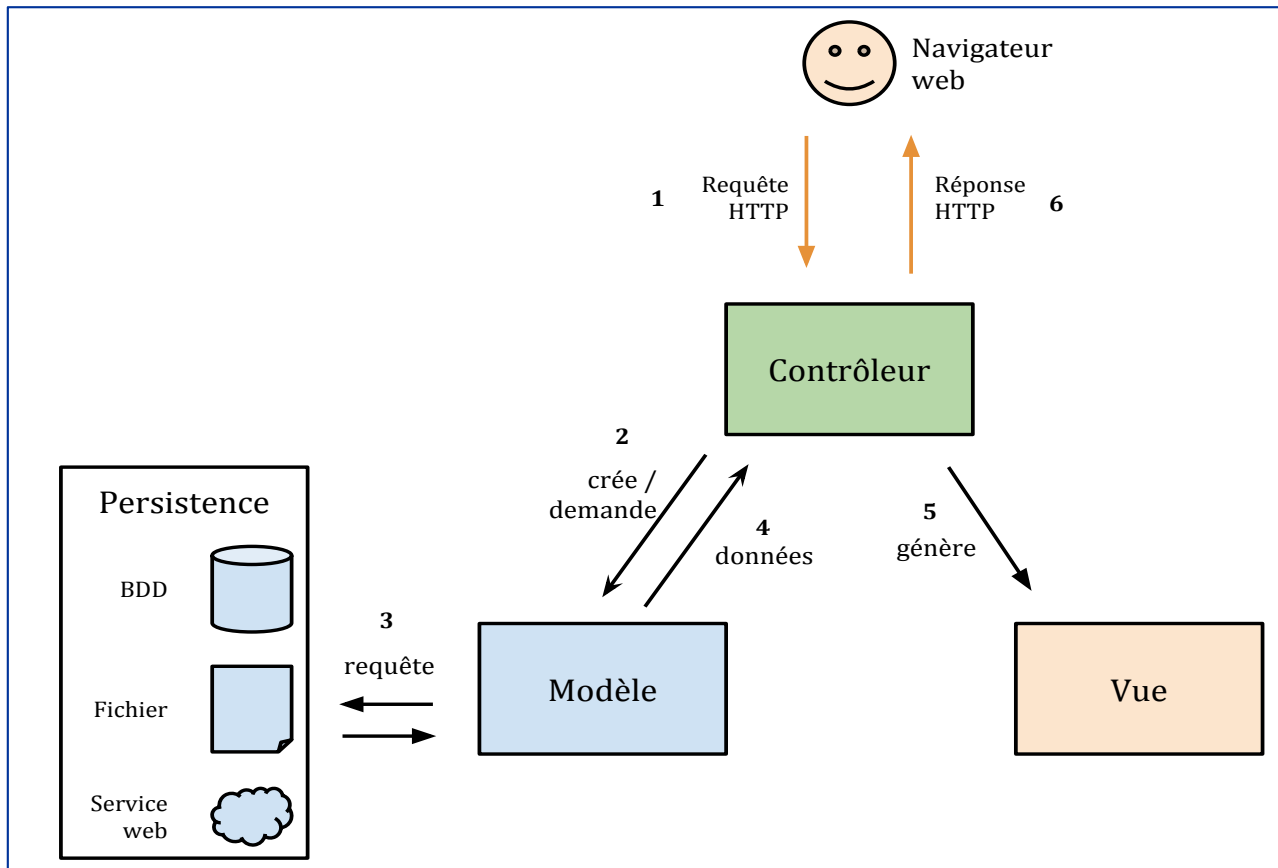
insert into produits(titre, descriptif, stock, prix, fabricant) values
('The 7th Continent Classic Edition', 'L'élément indispensable pour jouer au jeu',
('7 Wonders', 'Prenez la tête de l'une des sept grandes cités du monde Antique.',
('Titles', 'Un jeu d'ambiance et de culture générale pour tous les amateurs de',
('Skull & Roses', 'Jeu d'ambiance, bluff, déduction : réussir deux défis', 5, 16,
```


Pattern Modèle Vue Contrôleur

- Objectif : modularité, séparation des composants, évolutivité de l'application
- Le **Modèle**: gère l'accès aux données (BD, fichiers, service) de l'application
- La **Vue**: décrit l'affichage en fonction de l'état du modèle
- Le **Contrôleur**: gère les actions de l'utilisateur et les transmet au modèle

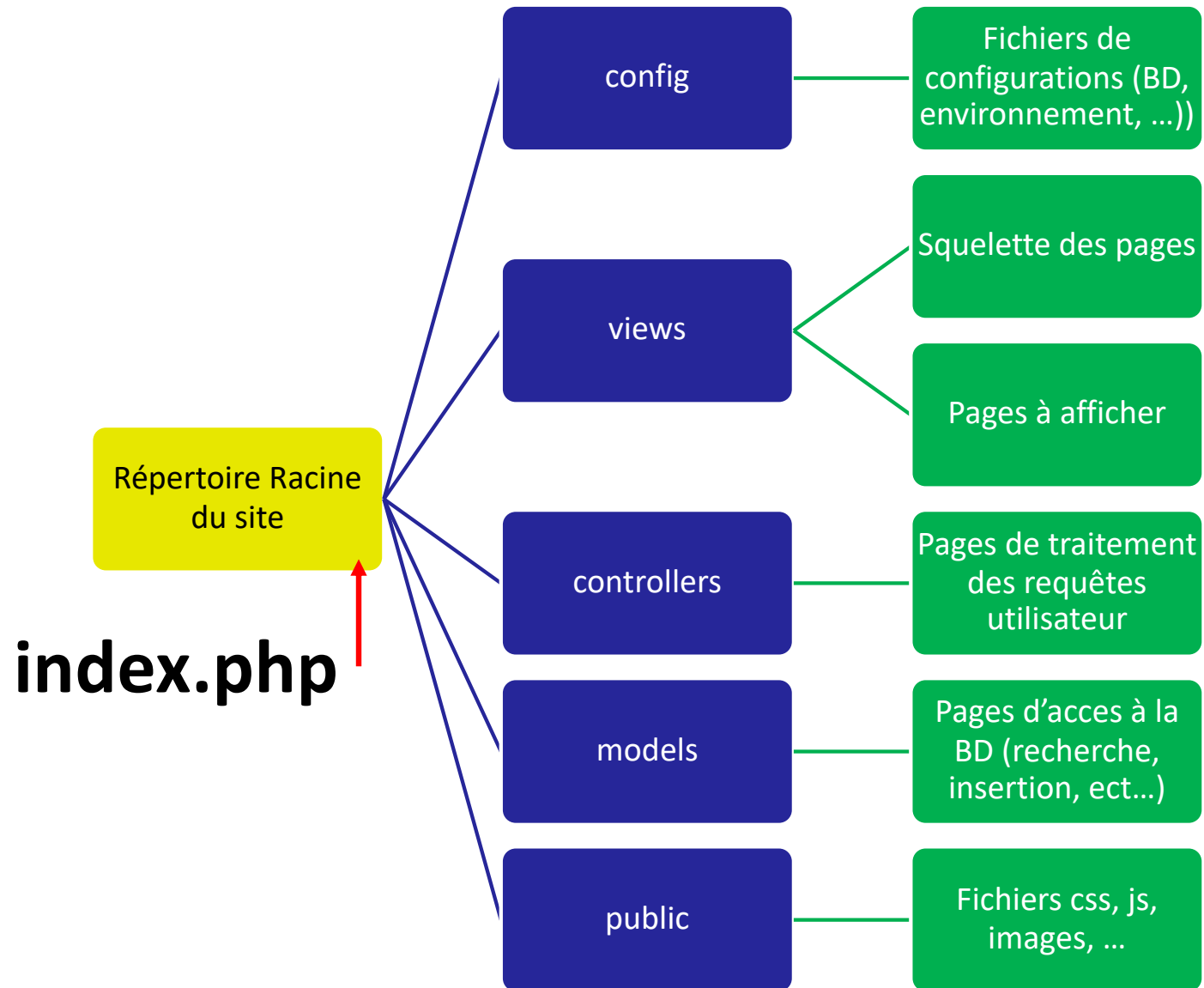
Pattern utilisé dans tous les
framework

Fonctionnement de l'architecture MVC

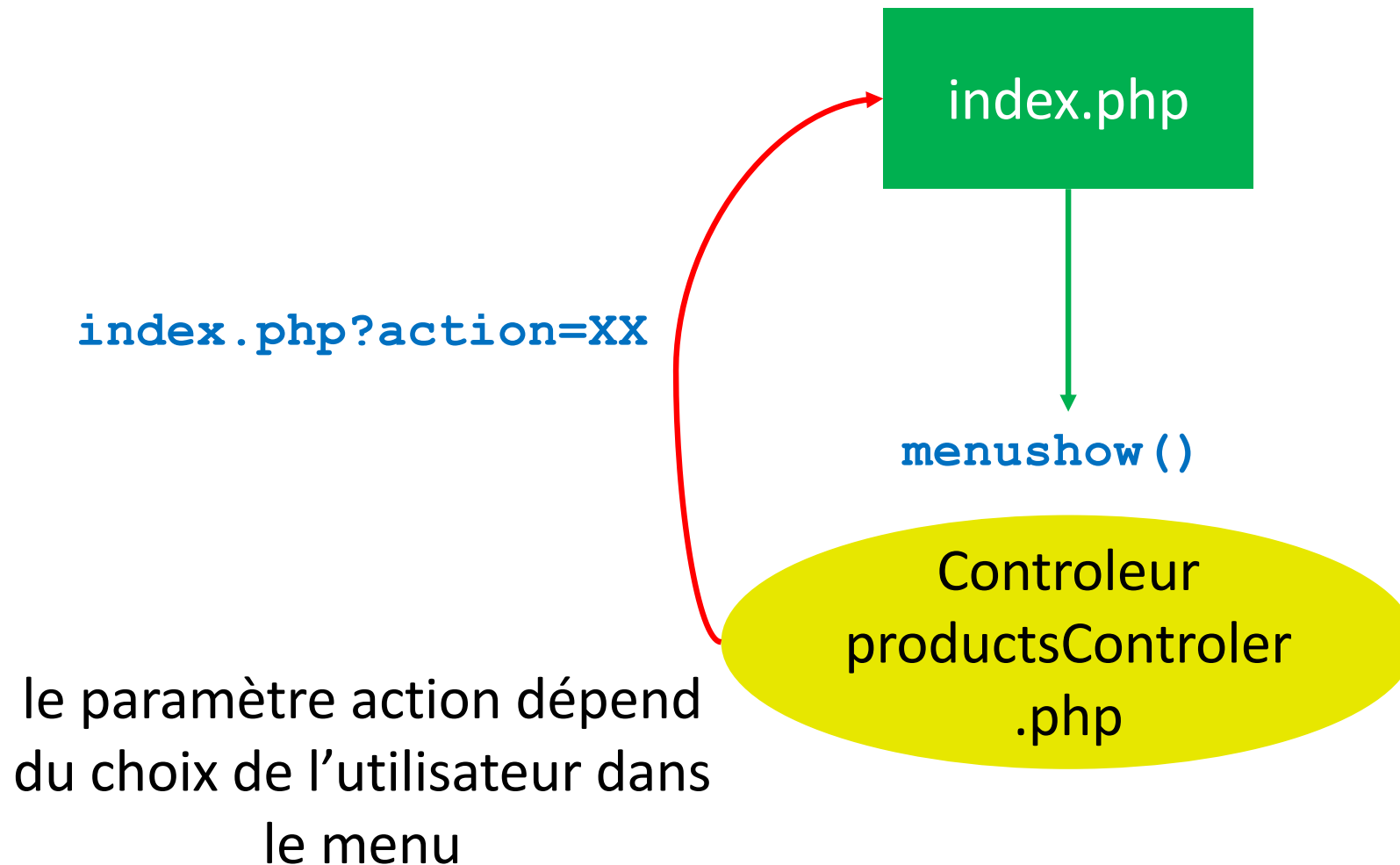


1. Le contrôleur reçoit une requête de l'utilisateur et vérifie ses paramètres
2. Le contrôleur demande aux modèles les données nécessaires .
3. Le modèle récupère les **données** demandées.
4. Le modèle renvoie les données **encapsulées** au contrôleur.
5. Le contrôleur génère la vue d'affichage des données.
6. Le contrôleur renvoie à l'utilisateur la vue générée.

Organisation des fichiers



Fichier d'accueil du site: index.php



Fichier d'accueil du site: index.php

```
<?php
```

```
ini_set('error_reporting', E_ALL) ;  
ini_set('display_errors', 1);  
require 'controllers/productsController.php';
```

le fichier controleur
productsController.php
est importé

- Le fichier index.php est appelé dans différents cas:
 - 1^{er} accès au site : sans aucun parametre Get
 - A partir d'une vue : paramètre Get « action »
 - products : affichage de la liste des produits
 - ajout: ajout d'un nouveau produit
 - recherche: recherche d'un produit à partir de son titre
 - product: affichage d'un produit (avec un deuxième paramètre id)

index.php (suite)

```
try {  
    if (isset($_GET['action'])) {  
        $action=$_GET['action'];  
        switch($action){  
            case 'products' :{ // GET /index.php?action=products  
                productsIndex();  
                break;  
            }  
            case 'product' : {  
                if (isset($_GET['id'])) { // GET /index.php?action=product&id=X  
                    productShow();  
                } else { // GET /index.php?action=product  
                    error(404, "/index.php?action=product");  
                }  
                break;  
            }  
            case 'ajout': {  
                ajout();  
                break;  
            }  
            case 'recherche': {  
                recherche();  
                break;  
            }  
            case 'menu' : {  
                menuShow();  
                break;  
            }  
            default: {  
                throw new Exception("Action non valide");  
            }  
        }  
    }  
    else { // GET /index.php  
        //1er appel d'index sans action  
        menuShow();  
    }  
} catch (Exception $e) {  
    error(500, $e->getMessage());  
}
```

Cas d'appel de la
page à partir des vues

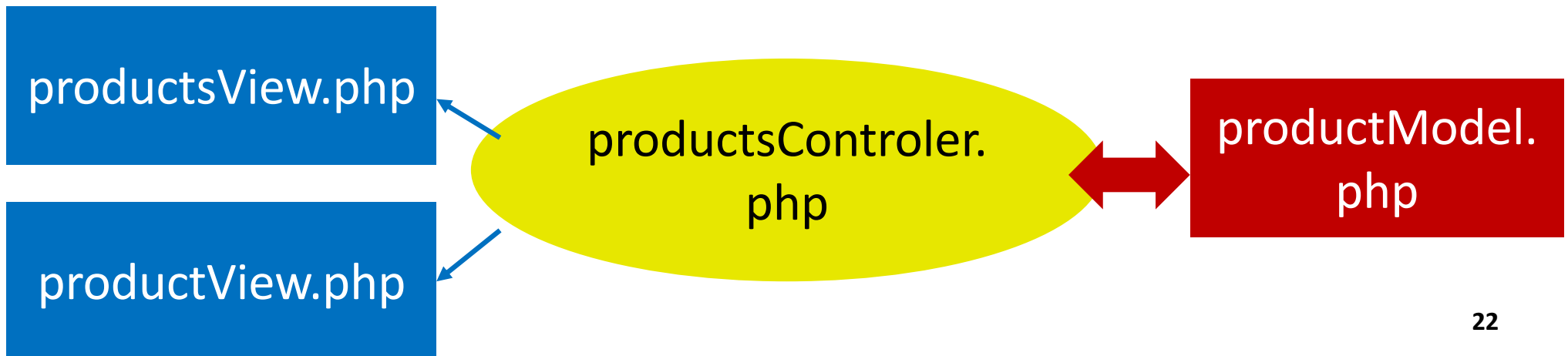
Fonctions
du
contrôleur

Appel initial

Contrôleur



- Fichiers php de coordination :
 - Déclenchent l'affichage des vues
 - Font appel aux modèles pour récupérer les données
- Exemple:
 - Contrôleur des produits (productsController.php)



Contrôleur **productsController.php**

```
<?php
require 'models/productModel.php';

function menuShow() {
    require 'views/menuView.php';
}

// Permet d'afficher une erreur
function error($code, $msg) {
    if ($code == 404) {
        http_response_code(404);
        require 'views/404.php';
    } else {
        http_response_code(500);
        require 'views/500.php';
    }
    die();
}
```

le fichier modèle
productModel.php
est importé

fonction
d'affichage de la
vue menu

fonction de gestion des erreurs

Contrôleur `productsControler.php` (suite)

```
function productsIndex() {  
    $products = getProductsModels();  
    require 'views/productsView.php';  
}
```

appel d'une fonction du modèle

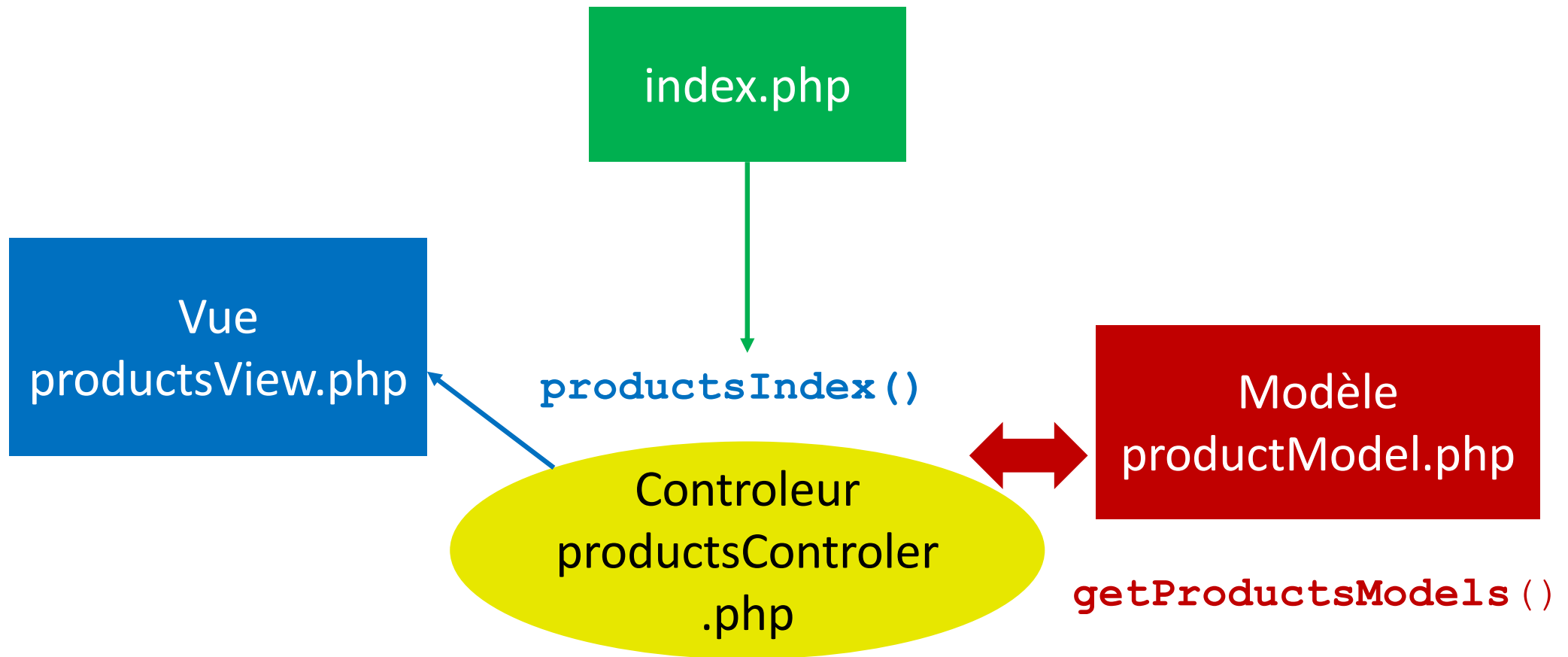
affichage de la vue des produits

```
function productShow() {  
    $id = intval($_GET['id']);  
    if ($id != 0) {  
        $product = getProductModel($id);  
        require 'views/productView.php';  
    } else {  
        error(404, "Produit " . $id );  
    }  
}
```

appel d'une fonction du modèle

affichage de la vue d'un produit

Choix de l'affichage de la liste des produits



Modèles

- Fichiers php contenant des fonctions d'accès à la base de données (ou à d'autres sources de données).
- Un modèle regroupe des fonctions par thème:
 - Exemple: modèle des produits (productModel.php)

Fonctions de manipulation de la table produit

- Fonction renvoyant tous les produits
- Fonction renvoyant un produit à partir de son id

Modèle des produits:

productModel.php

```
<?php

require 'config/db.php';

// Initialise une seule fois la connection à la base de donnée
function getDatabase() {
    static $db = null;
    if ($db == null) {
        $dsn = 'mysql:host=' . DB_HOST . ';port=' . DB_PORT . ';dbname=' . DB_NAME . ';charset=utf8';
        $db = new PDO($dsn, DB_USER, DB_PWD);

        // lever une exception si erreur
        $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }
    return $db;
}
```

Modèle des produits:

productModel.php (suite)

```
// Renvoie la liste des produits
function getProductsModels() {
    $db = getDatabase();
    $products = $db->query('select * from produits');
    return $products;
}

// Renvoie un produit selon son id
function getProductModel($id) {
    $db = getDatabase();
    $query = $db->prepare('select * from produits where id_produit=?');
    $query->execute(array($id));
    if ($query->rowCount() == 1) {
        return $query->fetch();
    } else {
        error(404, "Produit " . $id);
    }
}
```

Modèle des produits:

productModel.php (suite)

```
function ajoutProduit($titre,$descriptif,$stock,$prix,$fabricant){
    //cette fonction retourne un entier égal à 1 si l'ajout s'est bien déroulé
    $db = getDatabase();
    $req_ajout="INSERT INTO produits(titre,descriptif,stock,prix,fabricant) "
        . "VALUES('$titre','$descriptif',$stock,$prix,'$fabricant')";
    $res=$db->exec($req_ajout);
    return($res);
}

function rechercheProduit($titre) {
    $db = getDatabase();
    // $products = $db->query("select * from produits where titre='$titre'");
    $query = $db->prepare("select * from produits where titre='$titre'");
    $query->execute(array($titre));
    if ($query->rowCount() >= 1) {
        return $query;
    } else {
        error(404, "Produit " . $titre);
    }
}
```


Vues

- Fichiers php correspondant à l'interface utilisateur .

Les seuls fichiers contenant du html!!

- Un fichier squelette (layout.php)
- Un fichier php pour chaque page à afficher
 - Exemples:
 - Page d'affichage d'un produit (product.php)
 - Page d'affichage d'une liste de produits (products.php)
 - Pages d'affichage des erreurs (404.php, 500.php)

Squelette des vues

layout.php

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="public/style.css" />
    <title><?= $title ?></title>
  </head>
  <body>
    <header>
      <a href="index.php"><h1><?= $title ?></h1></a>
      <h2>Ce header est partagé par toutes les vues.</h2>
    </header>
    <hr/>
    <div>
      <?= $content ?>
    </div>
    <footer>
      Mon footer, partagé par toutes les vues également
    </footer>
    <hr/>
  </body> </html>
```

Vue d'affichage des produits

```
<?php $title = 'Mon E-Commerce - ' . "Produits"; ?>
```

```
<?php ob_start() ?>
```

productsView.php

```
<?php foreach ($products as $product): ?>
```

```
    <article>
```

```
        <a href="index.php?action=product&id=
            <?= $product['id_produit']; ?>">
```

```
            <h2>
```

```
                <?php echo $product['titre'] . ' (' .
                $product['prix']; ?> €)
```

```
            </h2>
```

```
        </a>
```

```
        <p>par <?php echo $product['fabricant'] ?></p>
```

```
    </article>
```

```
    <hr />
```

```
<?php endforeach ?>
```

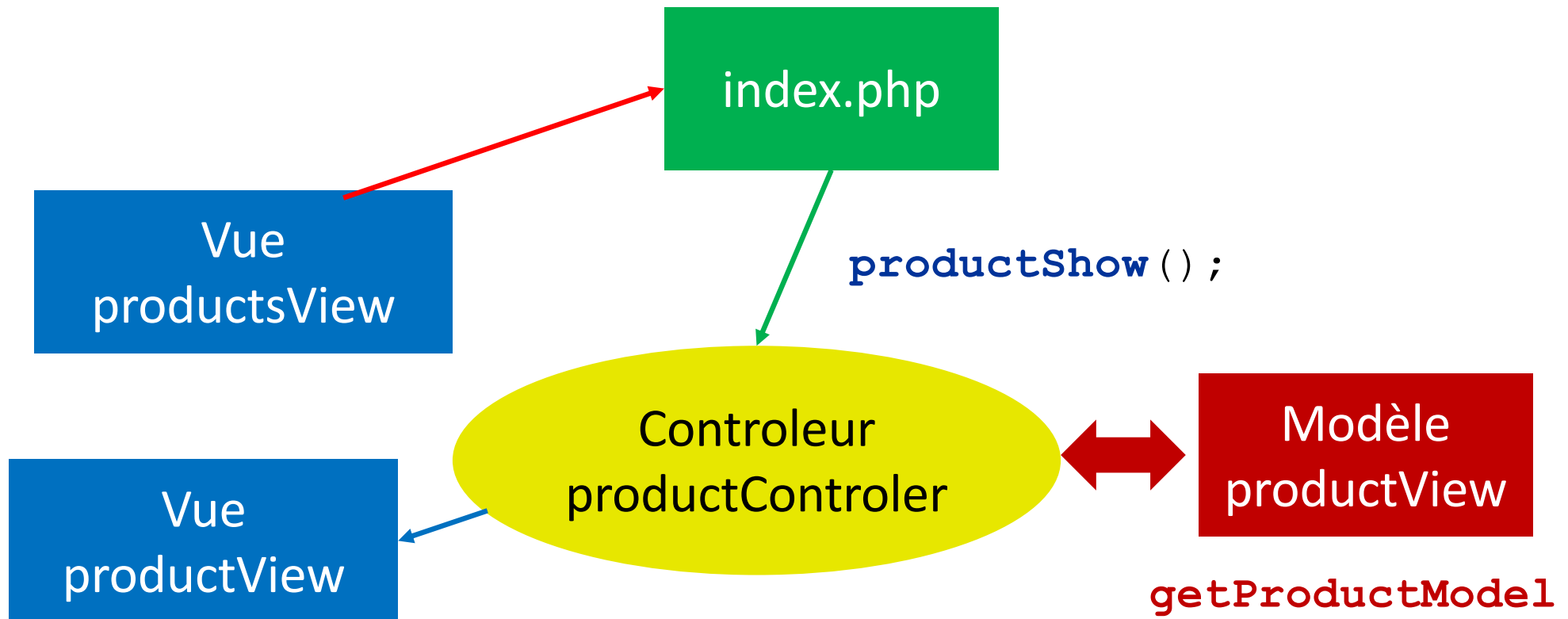
```
<?php $content = ob_get_clean(); ?>
```

```
<?php require 'layout.php'; ?>
```

lien vers index.php avec transmission
paramètres action et id en GET

Retour sur index.php à partir d'une vue

```
<a href="index.php?action=product&id=
    <?=$product['id_produit']; ?>">
```



Vue d'affichage d'un produit

```
<?php $title = 'Mon E-Commerce - ' . $product['titre'];
?>
<?php ob_start() ?>
<article>
    <header>
        <h2>
            <?php echo $product['titre'] . ' (' .
$product['prix']; ?> €)
        </h2>
        <p>par <?php echo $product['fabricant'] ?></p>
    </header>
    <p><?php echo $product['descriptif']; ?></p>
</article>
<hr />
<?php $content = ob_get_clean(); ?>

<?php require 'layout.php'; ?>
```

productView.php

Utilisation d'un tampon de sortie

- Placement du texte dans un tampon avant de l'envoyer au navigateur
- **ob_start**= démarre la temporisation de sortie (remplissage du tampon) , rien n'est envoyé au navigateur
- **ob_get_clean**: arrête la temporisation et renvoie le contenu du tampon

```
<?php ob_start(); ?> Début
<?php
while ($data = $posts->fetch())
{
    <div class="news">
        <h3>
            <?= htmlspecialchars($data['title']) ?>
            <em>le <?= $data['creation_date_fr'] ?></em>
        </h3>
        <p>
            <?= nl2br(htmlspecialchars($data['content'])) ?>
            <br />
            <em><a href="post.php?id=<?= $data['id'] ?>">Commentaires</a></em>
        </p>
    </div>
}
$posts->closeCursor();
<?php $content = ob_get_clean(); ?> Fin
```

Tout le code généré ici va dans \$content



Exercice CH2 –

- Récupérez sur l'ENT le fichier exempleMVCCH2.zip.
- Testez son fonctionnement.
- On vous demande d'ajouter la possibilité de supprimer un produit à partir de son id.
- Vous allez devoir:
 1. modifier le menu général (vue menuView)
 2. ajouter une vue
 3. modifier la page index.php
 4. ajouter une fonction dans le controleur
 5. ajouter une fonction dans le modèle

[Retour au menu](#)

Suppression produit

Exemple MVC Simple

Id du produit à supprimer :

Liens

- Une liste des bonnes pratiques php
 - https://fr.wikibooks.org/wiki/Programmation_PHP/Concevoir_du_code_de_haute_qualit%C3%A9
- Le tutoriel de Romain Lebreton sur MVC
 - <https://romainlebreton.github.io/ProgWeb-CoteServeur/tutorials/tutorial4.html>
- Le cours de Baptiste Pesquet détaillant la construction progressive d'une architecture MVC
 - <https://bpesquet.developpez.com/tutoriels/php/evoluer-architecture-mvc/#LII-C-1>