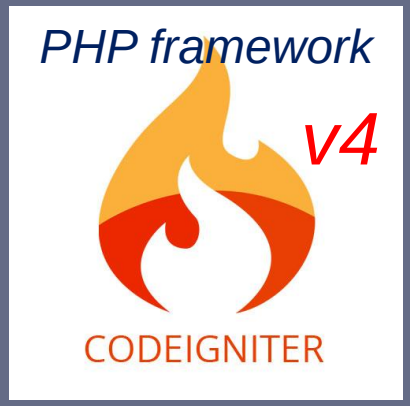
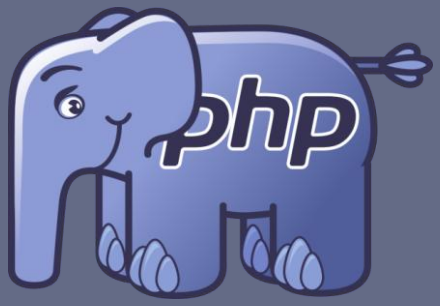




PHP > 7.2

MySQL > 5.1

CodeIgniter 4.0.4

Création de pages dynamiques

1/2

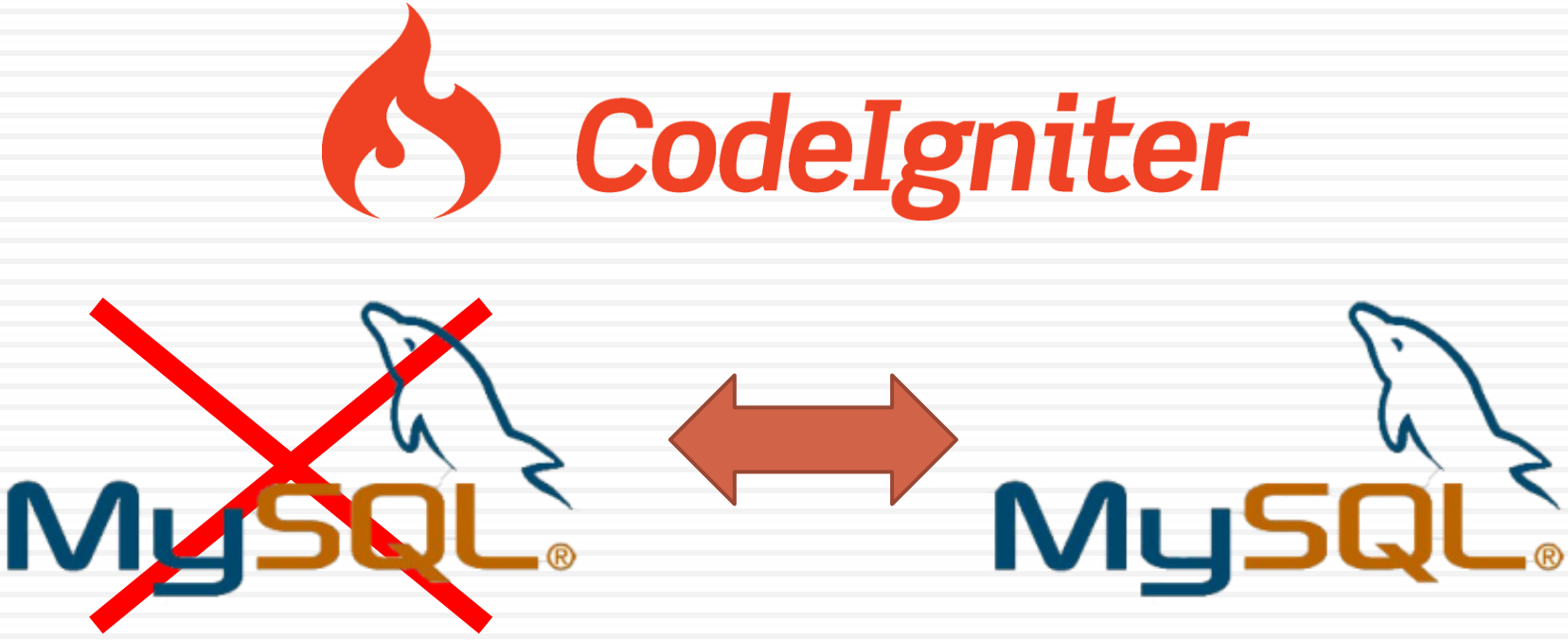
CI4_3

Gwénaël LAURENT – glaurent001@gmail.com

Plan

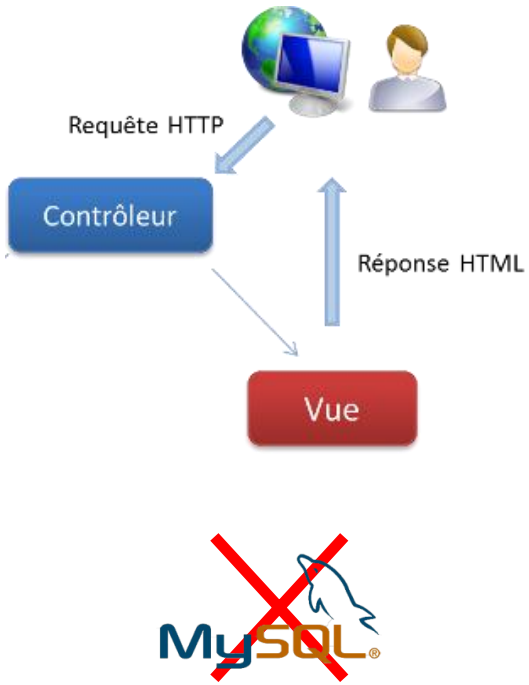
- Pages statiques vs dynamiques
- Création de la base de données
- Configuration du framework
- La page dynamique pour lister les conteneurs du Relais
- La liste des conteneurs avec pagination
- Afficher le détail d'un conteneur
- Gestion des erreurs

Pages statiques vs dynamiques

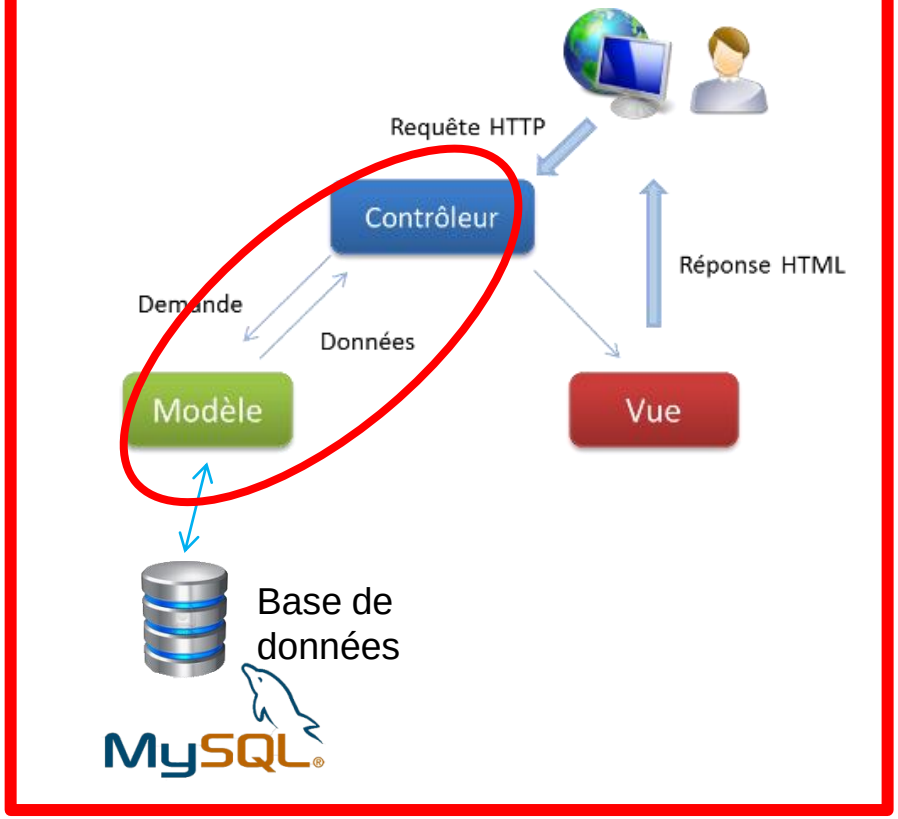


Pages statiques vs dynamiques

Page statique



Page dynamique



5

Création de la base de données

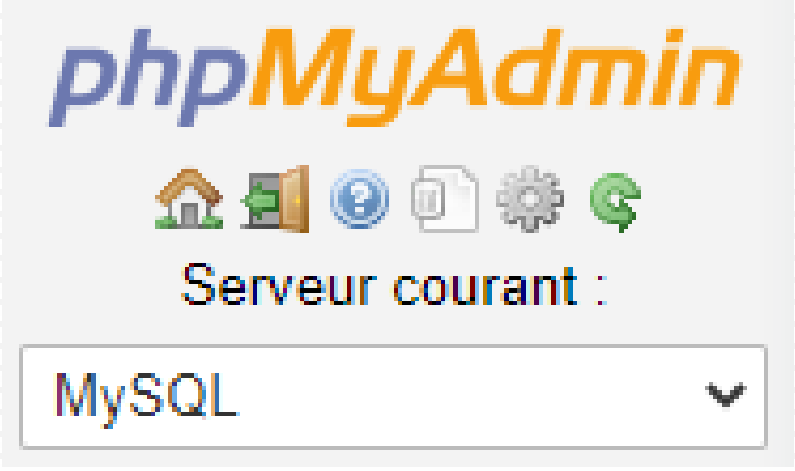
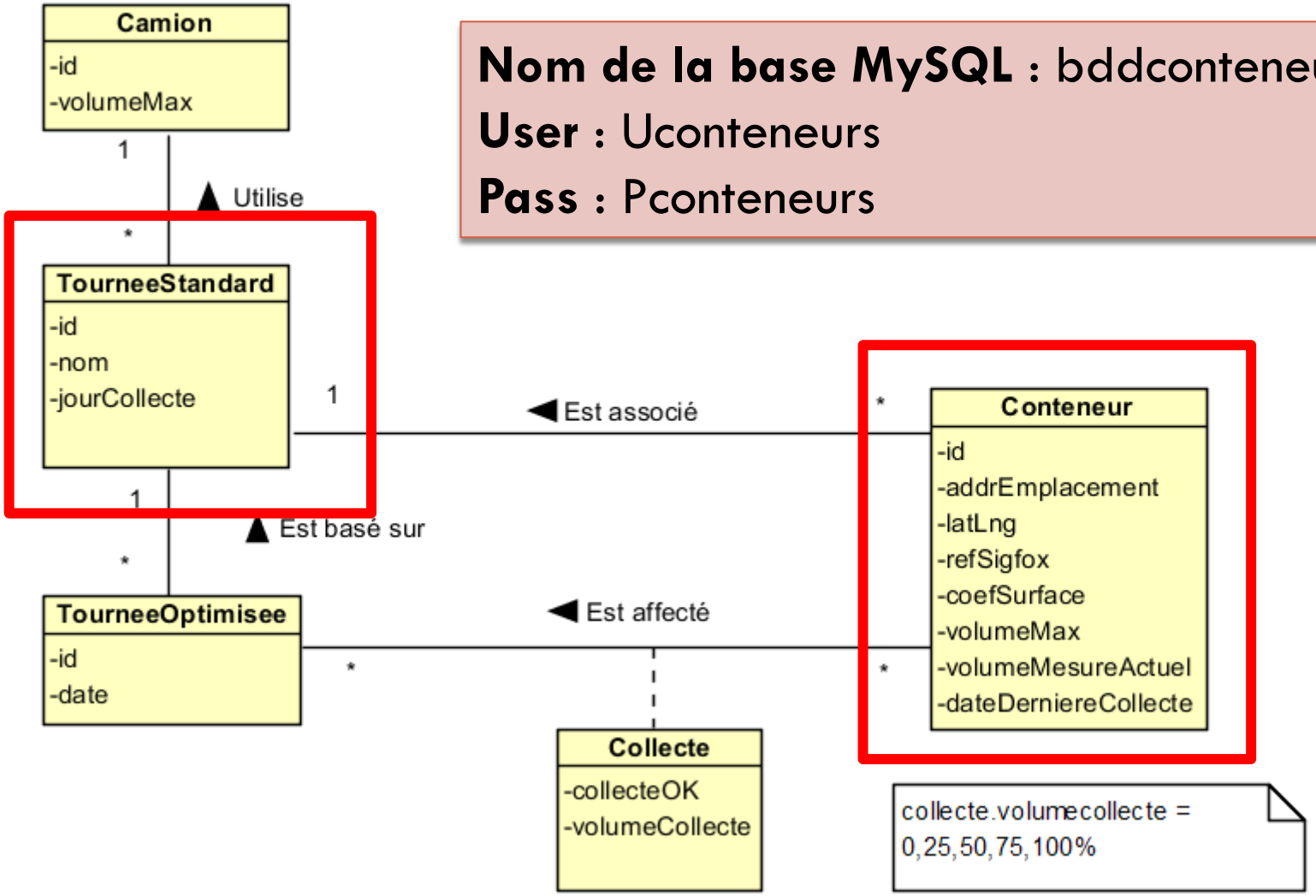


Schéma conceptuel de la bdd

Nom de la base MySQL : bddconteneurs
User : Uconteneurs
Pass : Pconteneurs



Données de la base « bddconteneurs »

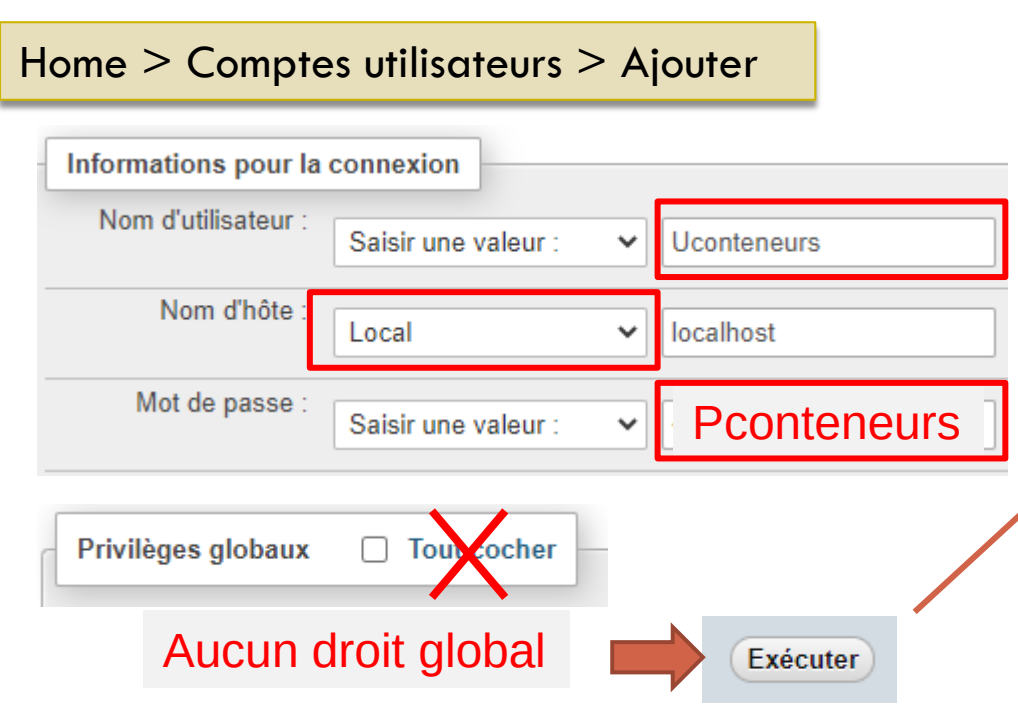
Id	CamionId	Nom	JourCollecte
3	1	Béthune	lundi
4	1	Armentières	mardi
5	2	Bailleul	mercredi
6	2	Estaires	jeudi

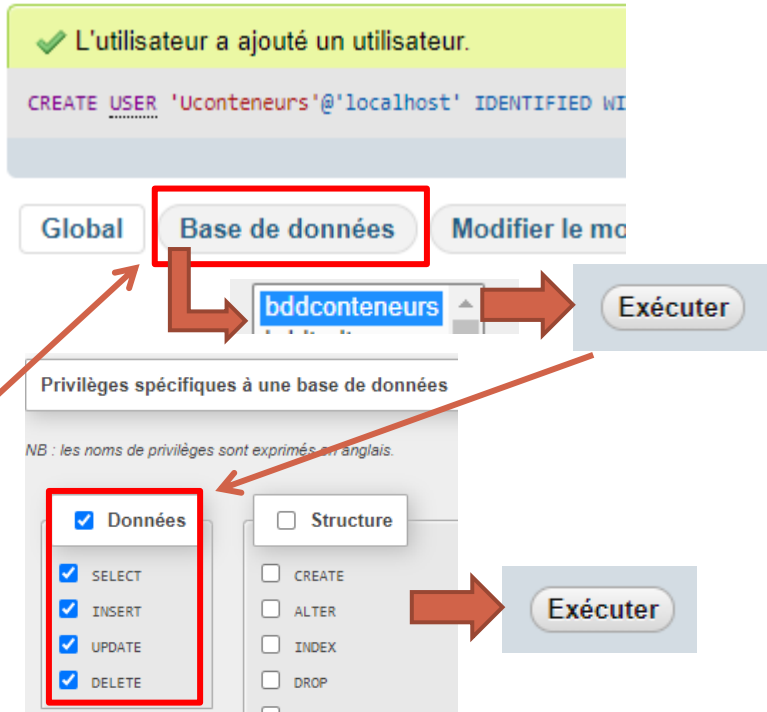
Id	TourneeStandardId	AddrEmplacement	LatLng
1	3	rue(s) Ferdinand Bart 62400 BÉTHUNE	50.534600,2.63
2	3	92 avenue du 8 mai 1945 62400 BÉTHUNE	50.531224,2.65
3	3	rue(s) de Lille 62400 BÉTHUNE	50.525624,2.65
4	3	rue(s) du Docteur Dhénin 62400 BÉTHUNE	50.521624,2.63
5	3	rue(s) de Fouquereuil 62232 FOUQUEREUIL	50.518274,2.60
6	3	chemin(s) du Paradis 62131 VERQUIN	50.509961,2.63
7	3	place Roger Salengro 62131 VERQUIN	50.500325,2.61

conteneur.TourneeStandardId est une clé étrangère vers *tourneestandard.Id*

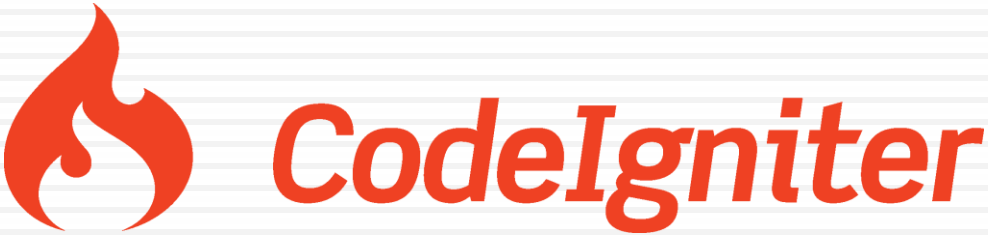
Création de la base de données

- Dans PhpMyAdmin
  - Importez le fichier « bddconteneurs.sql »
  - Créez l'utilisateur MySQL « Uconteneurs » et affectez ses droits CRUD seulement sur la base « bddconteneurs »





Configuration du framework



Configuration des accès à la base

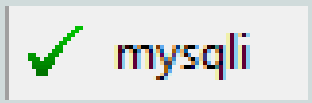
- Pour enregistrer les données de connexion à la base (*username, password, database name, etc.*) :
  - Modifiez le fichier **/app/Config/Database.php**
    - Adaptez les valeurs de l'attribut $default

/app/Config/Database.php

```
public $default = [
        'DSN'      => '',
        'hostname' => 'localhost',
        'username' => 'Uconteneurs',
        'password' => 'Pconteneurs',
        'database' => 'bddconteneurs',
        'DBDriver' => 'MySQLi',
        ...
        'charset'  => 'utf8',
        'DBCollat' => 'utf8_general_ci',
        ...
        'port'     => 3306,
];
```

Parce que le serveur PHP et le serveur MySQL sont sur la même machine

Ne pas oublier d'activer l'extension PHP « mysqli » dans WampServer



11

La page dynamique pour lister les conteneurs du Relais


```
php spark make:model Mconteneur --table conteneur --return array
```

Comment créer une page dynamique ?

- Construction en plusieurs étapes:
  1. Créer le squelette du contrôleur et de la vue comme une page statique
  2. Créer un modèle pour récupérer les données de la base (requête SQL)
  3. Modifier le contrôleur :
     1. Le contrôleur appelle le modèle
     2. Le contrôleur envoie les données à la vue
  4. Modifier la vue pour qu'elle affiche les données

URL pour afficher la liste des conteneurs

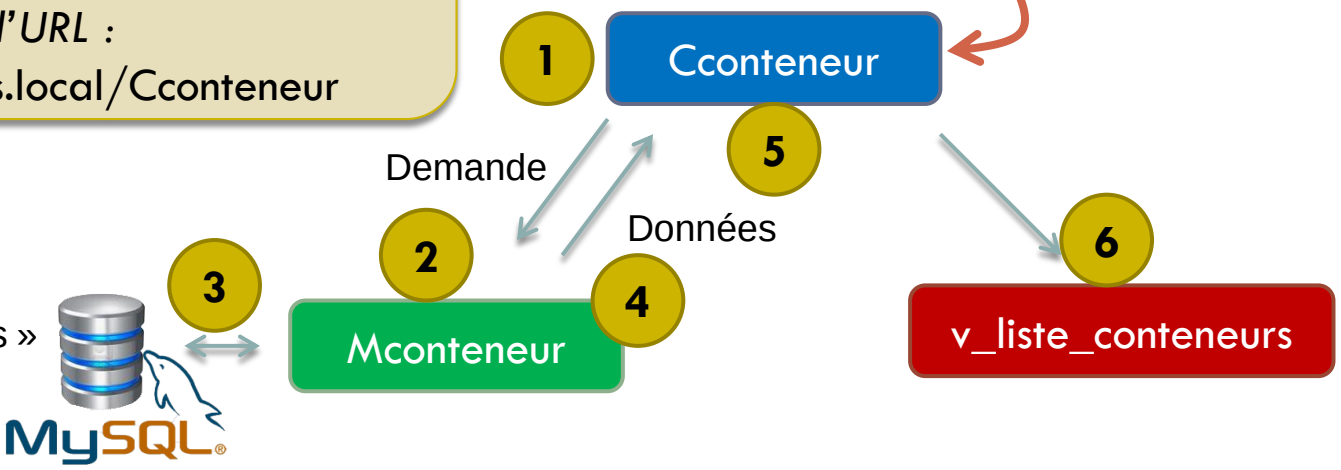
- URL d'accès à la page qui liste les conteneurs

`http://conteneurs.local/Cconteneur/index`

Nom du contrôleur

Nom de la méthode

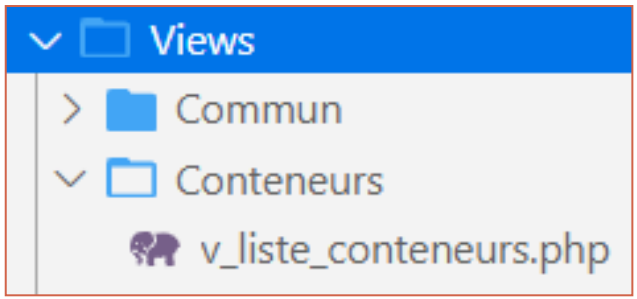
CodeIgniter instancie **Cconteneur**
et appelle sa méthode **index()**
méthode implicite si elle n'est pas présente dans l'URL :
http://conteneurs.local/Cconteneur



Remarque : Le nom du modèle et de la vue n'apparaissent pas dans l'URL

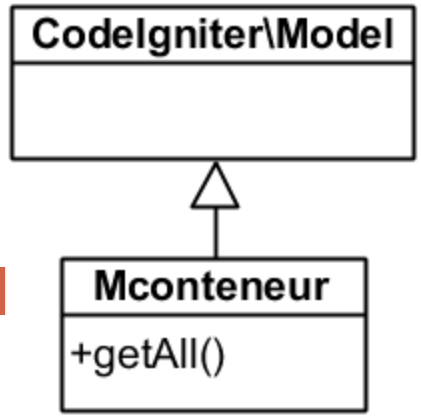
Créer le squelette du contrôleur et de la vue

- Créez une nouvelle page statique pour afficher la liste des conteneurs :
  - Créez le contrôleur /app/Controllers/**Cconteneur.php**
  - Créez la vue dans un sous dossier (*utile pour la suite de ce TP car il y aura une autre vue pour le même contrôleur*) /app/Views/**Conteneurs/v_liste_conteneurs.php**



  - Modifiez également les liens du menu en utilisant la fonction **base_url()** du helper URL

Généralités sur les modèles



- Règles d'accès à la base données :
  - Dans CodeIgniter, la connexion à la base et les requêtes SQL doivent se faire dans une **classe modèle** qui dérive de **CodeIgniter\Model**
  - On créera une méthode par requête SQL
  - Toutes les méthodes doivent retourner un tableau PHP
  - Les requêtes SQL utiliseront la classe « **Query Builder** »
    - Syntaxe propre à CodeIgniter
    - Syntaxe indépendante du type de SGBD
    - Protection contre les injections et attaques déjà intégrée

https://codeigniter4.github.io/userguide/database/query_builder.html

https://codeigniter4.github.io/userguide/models/model.html

Créer le modèle Mconteneur

/app/Models/Mconteneur.php

```php
<?php
namespace App\Models;
use CodeIgniter\Model;

class Mconteneur extends Model
{
  protected $table = 'conteneur';
  protected $primaryKey = 'Id';
  protected $returnType = 'array';


  public function getAll()
  {
    $requete = $this->select('Id, AddrEmplacement');
    return $requete->findAll();
  }
}
```

Toutes les classes Modèle doivent :

- être créées dans le dossier /app/Models
- faire partie du namespace App\Models
- importer le namespace CodeIgniter\Model
- dériver de CodeIgniter\Model
- Être associée à une table de la BDD

Les données retournées par les méthodes sont automatiquement converties en tableaux PHP

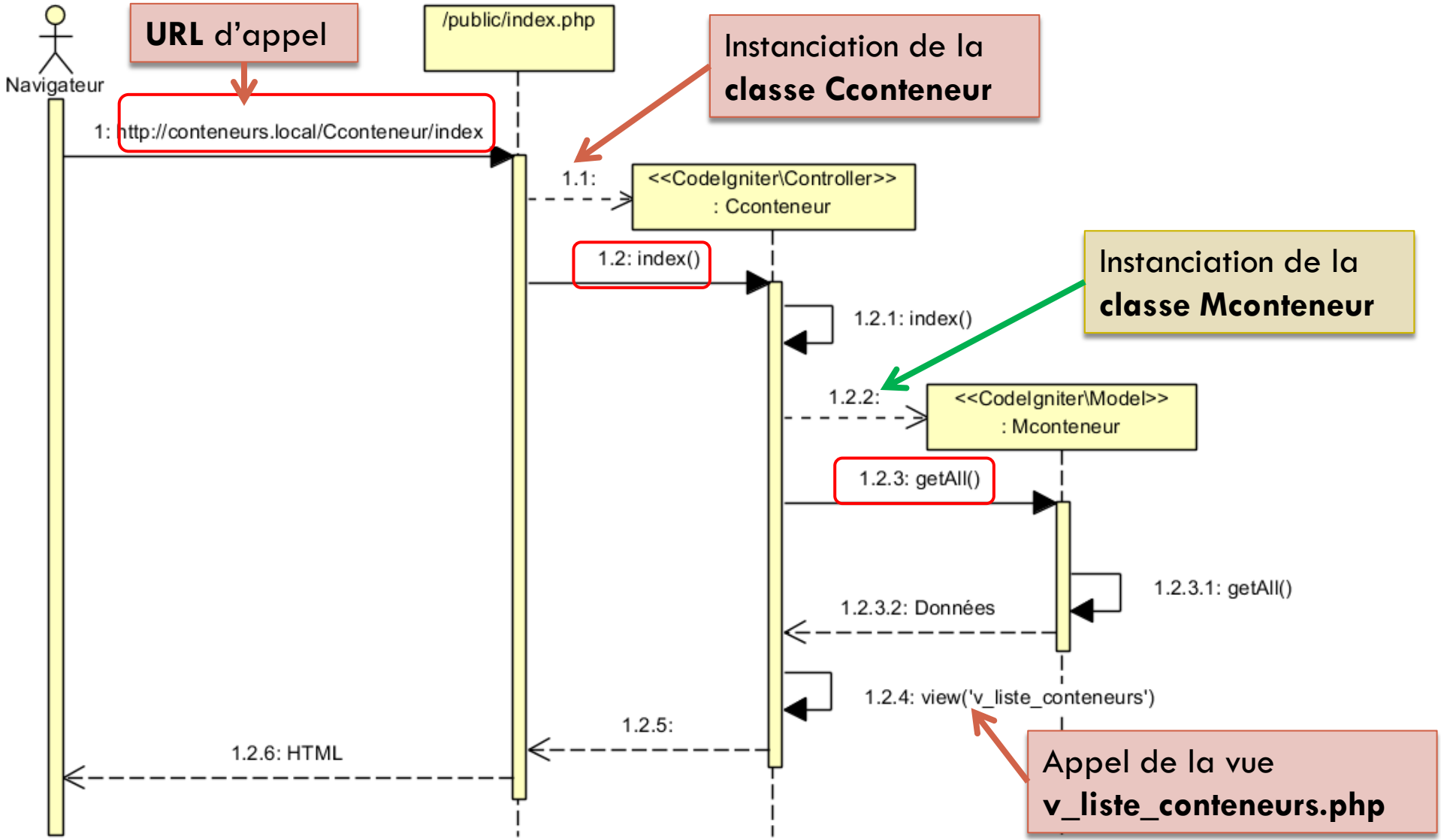
On choisit les colonnes de la table $this->table

On retourne toutes les lignes du resultset

Requête SQL correspondante :

SELECT `Id`, `AddrEmplacement` FROM `conteneur`

UML : fonctionnement pour afficher la liste des conteneurs



Modifier le contrôleur Cconteneur

- Dans Cconteneur, importez la classe Mconteneur

```
use App\Models\Mconteneur;
```

- Dans la **méthode index()** :
  - Créez un objet de la classe Mconteneur

```
$model = new Mconteneur();
```

  - Appelez la méthode getAll() du modèle

```
$data['result'] = $model->getAll();
```

  - Passez les données à la vue puis au template

```
$page['contenu'] =
        view('Conteneurs/v_liste_conteneurs', $data);
return view('Commun/v_template', $page);
```

Modifier la vue v_liste_conteneurs

- Les données sont reçues dans la variable **$result**

```
array(51)
  0: array(2)
    Id: "1"
    AddrEmplacement: "rue(s) Ferdinand Bart 62400 BÉTHUNE"
  1: array(2)
  2: array(2)
```

- La vue affiche les données grâce à une boucle **foreach**

```
<?php
    foreach ($result as $row) {
        echo $row['Id'] . "-";
        echo $row['AddrEmplacement'];
        echo "<br>";
    }
?>
```

Les conteneurs

1-rue(s) Ferdinand Bart 62400 BÉTHUNE
2-92 avenue du 8 mai 1945 62400 BÉTHUNE
3-rue(s) de Lille 62400 BÉTHUNE
4-rue(s) du Docteur Dhénin 62400 BÉTHUNE

La liste des conteneurs avec pagination

La pagination des données

- Tous les modèles disposent d'une bibliothèque qui gère la pagination des résultats : **pager**

1. **Dans le modèle**, à la place de findAll(), on renvoie les données avec :

```
return $requete->paginate(20); // 20 lignes par page
```

2. **Dans le contrôleur**, on passe une référence du pager à la vue

```
$data['pager'] = $model->pager;
```

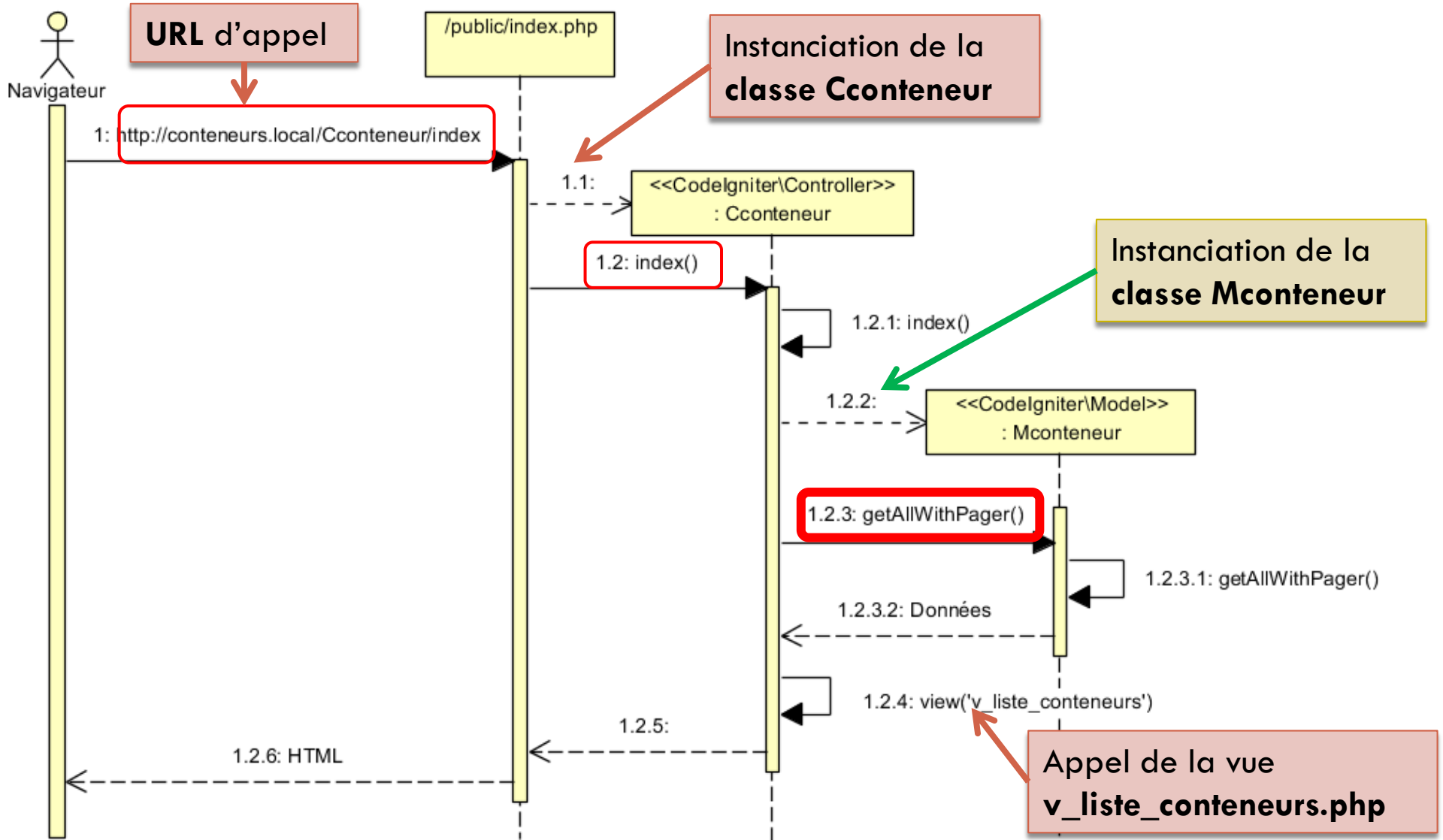
3. **Dans la vue**, on affiche les liens de navigation par page

```
<?php echo $pager->links(); ?>
```

- L'URL de la page intègre automatiquement la pagination

```
http://conteneurs.local/Cconteneur?page=3
```

UML : fonctionnement pour **paginer** la liste des conteneurs

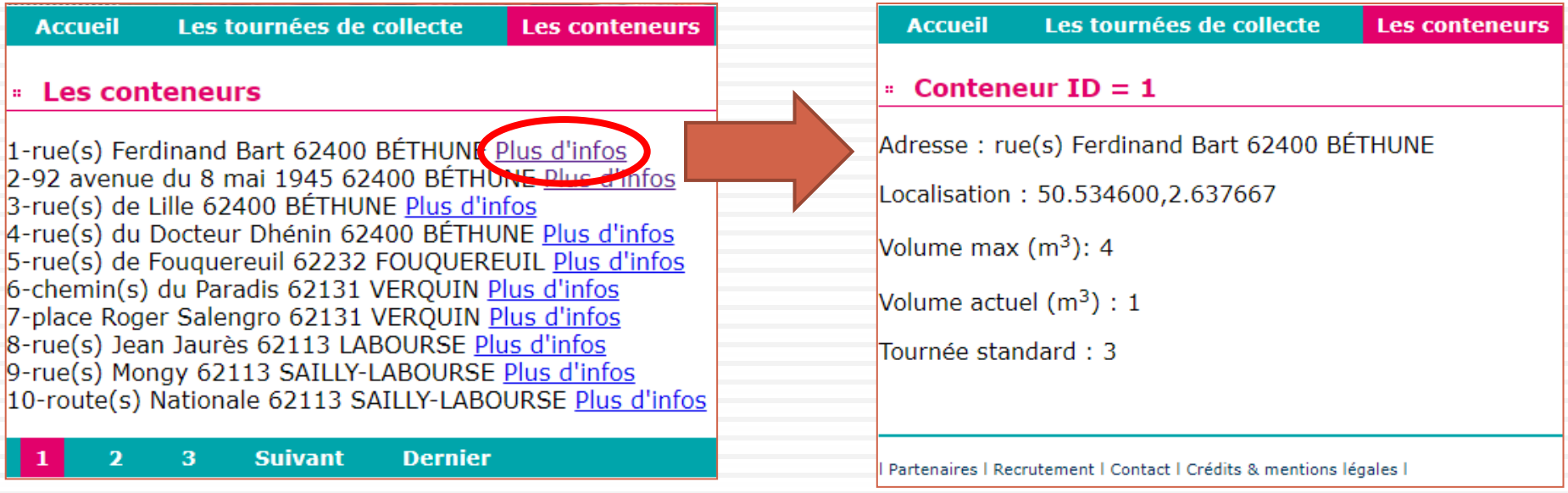


Activer la pagination

- Modifiez votre codage pour afficher la liste des conteneurs avec une pagination de 10 lignes par page
  - Dans le modèle **Mconteneur,** vous créerez une nouvelle méthode **getAllWithPager()**
    - Sélection des colonnes Id et Emplacement de la table conteneur
    - Renvoi des données par page de 10 lignes
  - Dans le contrôleur **Cconteneur,** modifiez la méthode **index()** pour prendre en compte la pagination
  - Afficher les **liens de pagination** dans la vue **v_liste_conteneurs**

24

Afficher le détail d'un conteneur



Modifier la liste des conteneurs

- Modifiez **v_liste_conteneurs** pour afficher un lien « *Plus d'info* » à côté de chaque conteneur (`<a href=...`)

rue(s) Ferdinand Bart 62400 BÉTHUNE Plus d'info
92 avenue du 8 mai 1945 62400 BÉTHUNE Plus d'info
rue(s) de Lille 62400 BÉTHUNE Plus d'info

- Ces liens doivent pointer vers ces URLs :

`http://conteneurs.local/Cconteneur/detail/XX`

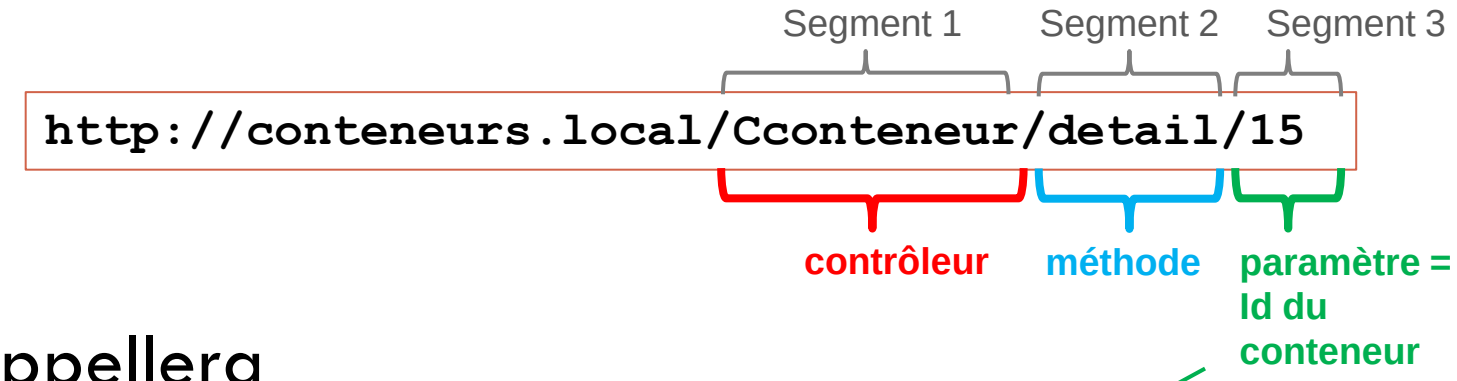
XX est à remplacer par la clé primaire du conteneur sélectionné

Id	TourneeStandardId	AddrEmplacement	LatLng
1	3	rue(s) Ferdinand Bart 62400 BÉTHUNE	50.534600,2.63
2	3	92 avenue du 8 mai 1945 62400 BÉTHUNE	50.531224,2.65
3	3	rue(s) de Lille 62400 BÉTHUNE	50.525624,2.65

Table : conteneur

Appel du contrôleur pour le détail

- Donc quand on cliquera sur un lien « *Plus d'infos* », on appellera cette URL

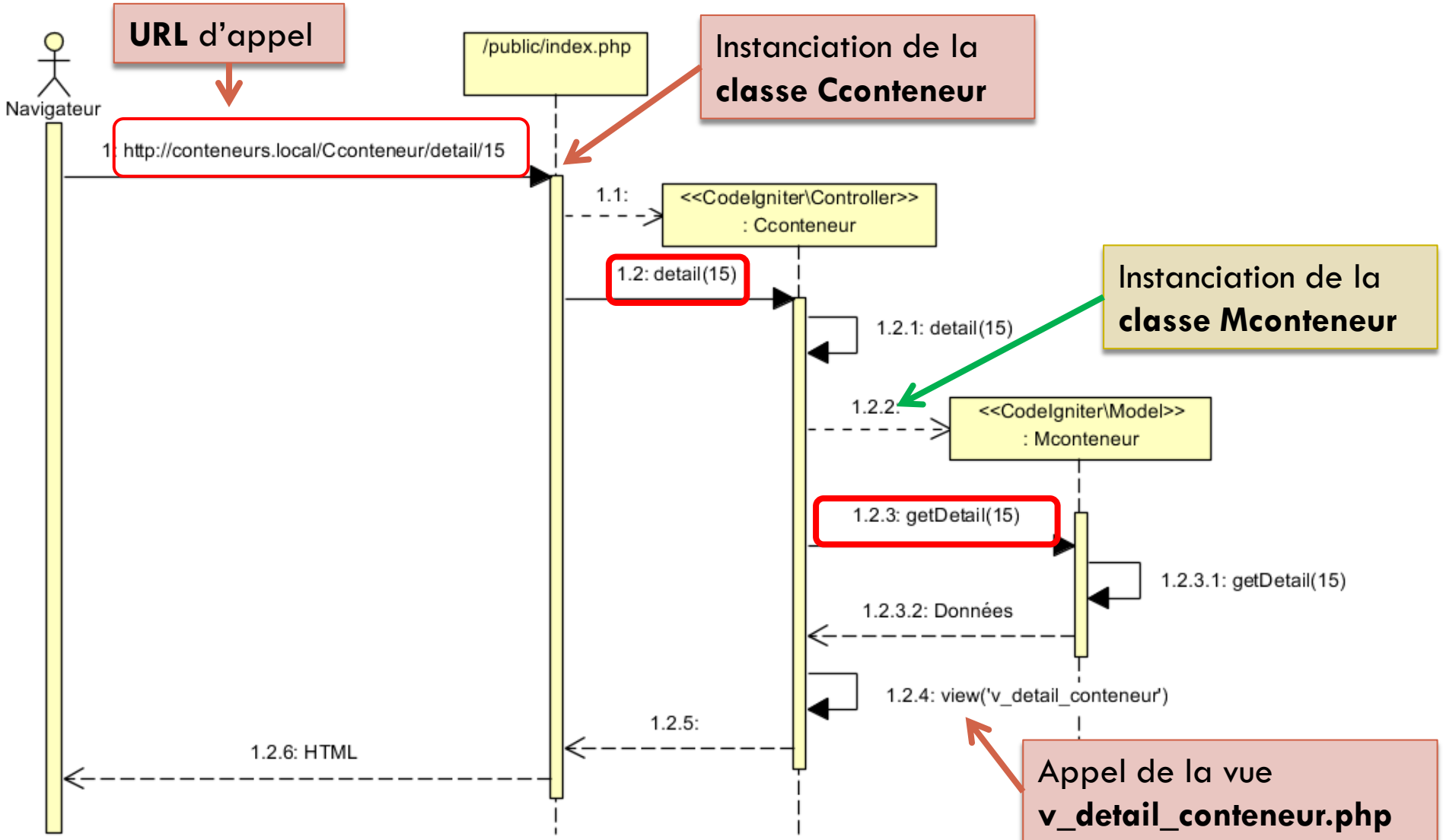


- Qui appellera automatiquement

Le 3ème segment est passé en paramètre de la fonction

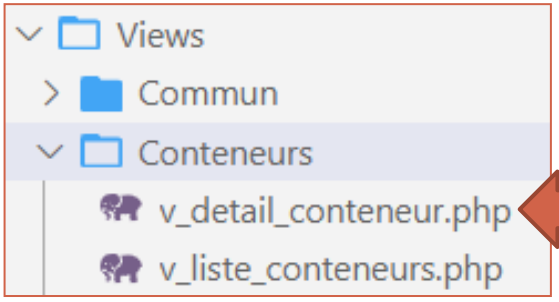
```
class Cconteneur extends Controller
{
    public function detail($prmId = null)
    {
        //$prmId vaudra 15 dans cet exemple d'URL
        // si pas de 3ème segment d'URL, alors $prmId = null
```

UML : fonctionnement pour afficher le détail d'un conteneur

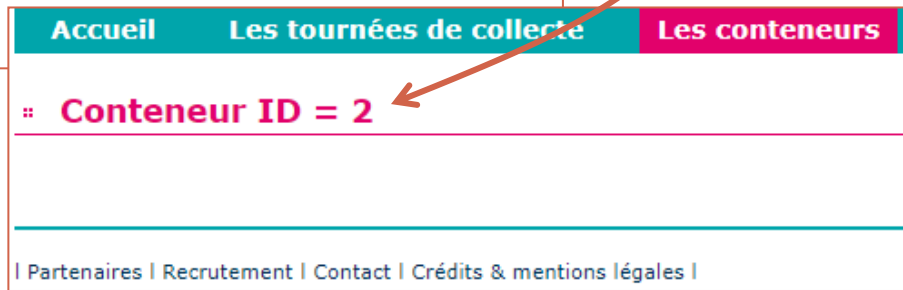


Codage du squelette contrôleur + vue

- Modifiez le contrôleur **Cconteneur**
  - Ajoutez une méthode `detail($prmId = null)`
- Créez une nouvelle vue pour afficher le détail d'un conteneur **v_detail_conteneur.php**
  - Dans le même dossier que la vue v_liste_conteneurs
  - Le titre de la page affichera l'ID du conteneur sélectionné




```
<article>
    <h1><?= $titre1; ?></h1>
    <p></p>
</article>
```



Codage du modèle

- Ajoutez une méthode dans le modèle **Mconteneur**

```
public function getDetail($prmId)
    {
        $requete = $this->select('*')
                        ->where(['Id' => $prmId]);
        return $requete->findAll();
    }
```

Le filtre « where » prend un **tableau associatif** en paramètre → on peut donc filtrer sur plusieurs critères

Remarque : Si on sélectionne tous les champs de la table, on peut omettre l'appel à la fonction select() :

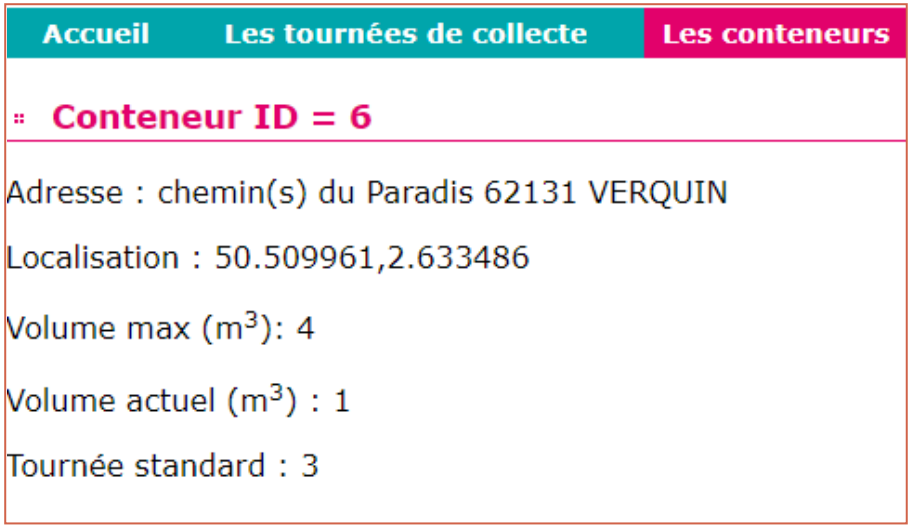
```
$requete = $this->where(['Id' => $prmId]);
return $requete->findAll();
```

On peut également écrire la requête sur une seule ligne :

```
return $this->where(['Id' => $prmId])->findAll();
```

Finaliser le contrôleur et la vue

- Finalisez le codage du contrôleur :
  - Appeler la méthode getDetail() du modèle en lui passant l'ID du conteneur
  - Passer les données à la vue dans $data['result']
- Finalisez le codage de la vue :
  - Afficher les données de $result (Attention, c'est un tableau de tableau)



Gestion des erreurs

Gérer les erreurs sur la page détail d'un conteneur

Puisque le fonctionnement du back-end est basé sur les URL, il faut toujours penser au cas où quelqu'un essaierait de se servir directement des URL pour avoir accès à nos données ou générer des erreurs :
→ Il faut **Sécuriser les URL**

- Il y a deux URL potentielles qui génèrent des erreurs dans la fonction **Cconteneur::detail()**
  - La fonction est appelée sans paramètre

    `http://conteneurs.local/Cconteneur/detail`

  - L'ID du conteneur n'existe pas dans la base

    `http://conteneurs.local/Cconteneur/detail/12345`

Modification du contrôleur pour gérer les erreurs

- Modifiez le code de la fonction **Cconteneur::detail()** pour afficher une page « 404 – File Not Found » si l'une des 2 erreurs précédentes apparait.
- CodeIgniter possède une « vue » pour afficher l'erreur 404. Pour retourner cette vue, il faut :
  - Importer la classe PageNotFoundException :

```
use \CodeIgniter\Exceptions\PageNotFoundException;
```

  - Déclencher l'affichage de la vue d'erreur 404 :

```
throw PageNotFoundException::forPageNotFound("votre message");
```

Ajustez votre message. Par exemple : "Ce conteneur n'existe pas !"
"Il faut choisir un conteneur !"

Personnaliser la page d'erreur 404

- La vue d'erreur 404 de CodeIgniter se trouve ici **/app/views/errors/html/error_404.php**
- Faites en une sauvegarde et modifiez l'original pour intégrer le modèle graphique du site comme ceci :

```
<?php
$page['page_title'] = "Cette page n'existe pas";
ob_start(); ?>
<article><h1>Cette page n'existe pas</h1><p>
                <?php if (!empty($message) && $message !== '(null)') : ?>
                        <?= esc($message) ?>
                <?php else : ?>
                        Désolé ! Cette page n'existe plus ou n'a jamais existée.
                <?php endif ?>
</p></article>
<?php
$page['contenu'] = ob_get_clean();
echo view('Commun/v_template', $page);
```

https://codeigniter4.github.io/userguide/general/errors.html > PageNotFoundException

Modifier les liens de pagination

- **Objectif** : les liens Précédent et Suivant se basent sur la page courante (et pas le groupe de page affiché)
  - Le fichier de vue utilisé pour afficher les liens de pagination est configuré dans app/Config/Pager.php
  - Modifiez le template 'default_full'

```
public $templates = [
'default_full'   => 'App\Views\Pagers\conteneur_full',
```

  - Créez la vue conteneur_full.php dans /app/Views/Pagers

```
<?php $pager->setSurroundCount(2) ?>
...
```

Après la 1ère ligne, collez dans ce fichier le dernier code donné à cette adresse : > Creating the View > links()
https://codeigniter4.github.io/userguide/libraries/pagination.html