

# Réalisation d'un site de contenu en base de données

# Sommaire

<b>Introduction</b> .....	3
<b>Les bases pour la réalisation du projet</b> .....	4
A. Récupération de la base projet_villes avec PhpMyAdmin... ..	4
B. Exercice d'application : un gestionnaire de contenu en base de données.....	13
C. Ajout d'une table Pays .....	21

Crédits des illustrations:  
© Fotolia, DR.

## Les repères de lecture



Retour au chapitre



Définition



Objectif(s)



Espace Élèves



Vidéo /  
Audio



Point important /  
À retenir



Remarque(s)



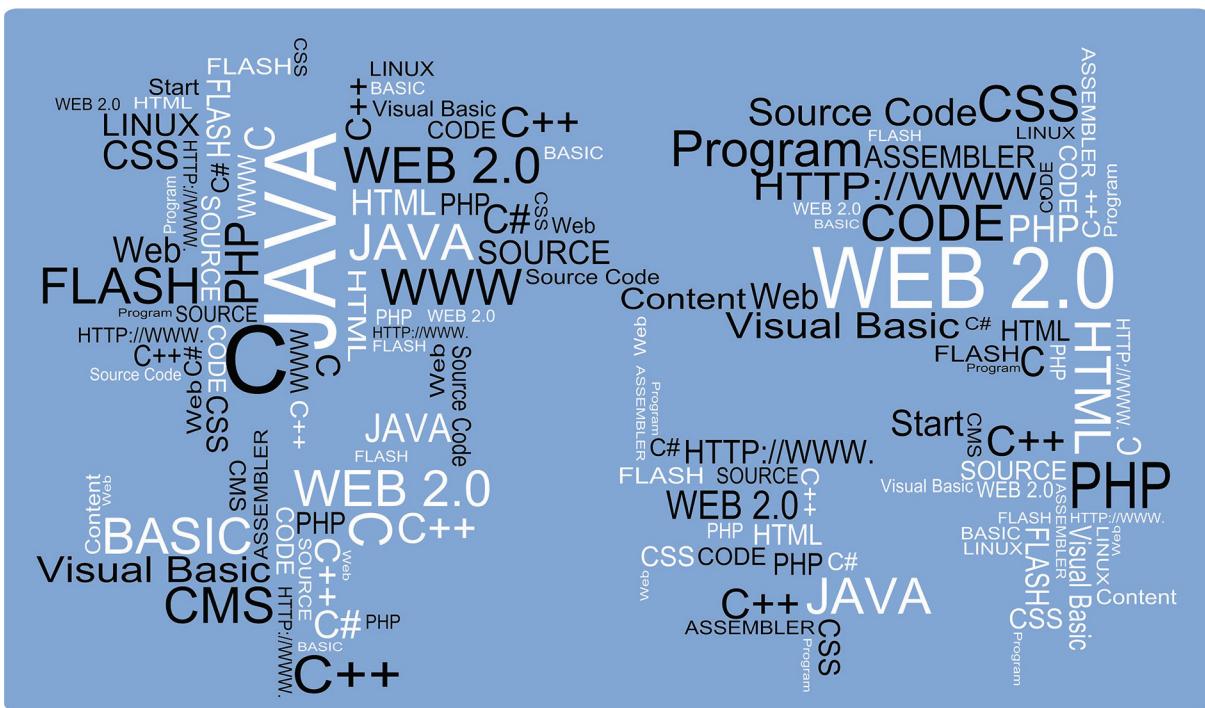
Pour aller  
plus loin



Normes et lois



Quiz



# Introduction

Notre connaissance actuelle de la relation PHP-MySQL nous permet dès à présent de développer un gestionnaire de contenu en utilisant une base de données. Ce type de besoin est un grand classique du développement web.

# Les bases pour la réalisation du projet

Nous continuons avec le thème des villes pour réaliser un mini-site simple constitué d'un menu de navigation, d'une page d'accueil et des pages permettant de consulter les informations sur les villes. Nous commencerons par les villes que nous avons déjà enregistrées dans la base de données. La mise en page sera simple et contiendra pour chaque ville le premier paragraphe fourni par la page Wikipedia correspondante.

Les contenus seront donc structurés comme suit:

- l'identifiant de la ville;
- le nom de la ville;
- un court texte de présentation.

La table ville contiendra donc trois champs:

1. le champ identifiant: ville\_id, dont le type sera INT
2. le champ nom: ville\_nom dont le type sera VARCHAR
3. le champ de contenu: ville\_texte dont le type sera TEXT.

## A. Récupération de la base projet\_villes avec PhpMyAdmin

Nous allons créer une nouvelle base de données mais puisque nous avons déjà enregistré les premiers noms de ville, nous allons les récupérer en les copiant avec PhpMyAdmin.

Créons la nouvelle base que nous appelons : projet\_villes\_site. Nous remarquons dans le menu de la colonne de gauche que PhpMyAdmin utilise les underscores pour créer une hiérarchisation des bases de données. Le mot « projet » est en effet considéré comme un préfixe et PhpMyAdmin affiche pour la base « projet\_villes » le lien « \_villes » et pour la base « projet\_villes\_site » le lien « \_villes\_site ».



Fig. 1 Liste des deux bases de données ayant le préfixe projet

Ensuite, pour récupérer le contenu de la base de données projet\_ville, nous cliquons sur le lien « \_villes » afin de nous trouver dans le contexte de cette base. Puis dans le menu du haut de page, nous cliquons sur l'onglet « Opérations ».



Fig. 2 Onglet Opérations du menu

PhpMyAdmin affiche alors une page contenant quatre zones :

- créer une nouvelle table sur la base projet\_villes;
- changer le nom de la base de données;
- supprimer la base de données;
- modifier l'interclassement (nous aborderons cette notion dans le prochain module);
- copier la base de données vers une autre base.

Nous choisissons de copier la base de données projet\_villes vers la nouvelle base. Nous saisissons donc le nom de la base destinataire projets\_villes\_site dans le champ de saisie et cochons les différentes propositions comme illustré ci-dessous.

The screenshot shows a configuration dialog for copying a database. At the top, it says "Copier la base de données vers:" (Copy database to:). Below that, a text input field contains "projet\_villes\_site". Under the heading "Type de copie" (Copy type), there are three radio buttons: "Structure seule" (Structure only), "Structure et données" (Structure and data), and "Données seulement" (Data only). The second option, "Structure et données", is selected. Below these are several checkboxes:

- Faire CREATE DATABASE avant la copie
- Ajouter DROP TABLE / DROP VIEW
- Inclure la valeur courante de l'AUTO\_INCREMENT
- Inclure les contraintes de clés étrangères
- Aller à la base de données copiée

At the bottom right of the dialog is a blue "Exécuter" (Execute) button.

Fig.3 Formulaire de copie de base de données

Passons en revue la liste de choix proposés :

Le choix « Structure et données » nous permet de récupérer toute la base : la structure est constituée par les champs (leur nom, leur type, etc.) et le contenu est constitué par les valeurs que nous avons enregistrées (identifiant et noms des villes, population, etc.).

Faire CREATE DATABASE avant la copie : PhpMyAdmin demande s'il doit créer la base destinataire. C'est inutile dans notre cas car nous l'avons déjà créé. Ajouter DROP TABLE / DROP VIEW : PhpMyAdmin demande s'il doit supprimer les tables de la base destinataire au cas où elles existent. C'est inutile dans notre cas car elles n'existent pas.

Inclure la valeur courante de l'AUTO\_INCREMENT : nous savons que l'AUTO-INCREMENT est une option pour que les valeurs du champ ville\_id soient automatiquement écrites par PhpMyAdmin. Nous choisissons donc de valider cette option.

Inclure les contraintes de clés étrangères : les clefs étrangères sont une notion que nous aborderons dans le prochain livret.

Aller à la base de données copiée : nous cochons cette option afin que PhpMyAdmin nous redirige vers la nouvelle base après validation.

Nous cliquons sur **Exécuter** pour valider nos choix.

Une fois la copie effectuée, nous vérifions en premier lieu grâce au fil d'Ariane que **nous sommes bien dans le contexte de la nouvelle base**. Nous constatons que la base de données « projets\_villes\_site » contient bien la table « villes » mais celle-ci contient le champ population dont nous n'avons pas besoin et ne contient pas encore le champ « ville\_texte » que nous souhaitons.

Cliquons sur l'onglet « Structures » et supprimons le champ « population ».



Fig.4 Suppression du champ population

Puis, ajoutons le champ « ville\_texte » au moyen du formulaire d'ajout de colonnes en lui attribuant le type TEXT.

Fig.5 Crédation du champ ville\_texte

Une fois cet ajout effectué, la table villes est créée et contient les trois champs voulus :

- ville\_id;
- ville\_nom;
- ville\_texte.

Récupérons pour chacune des cinq villes un des paragraphes proposés par Wikipedia et collons ces courts textes dans les champs correspondants comme vu précédemment (onglet Afficher, puis Tout cocher et modifier).

Maintenant que tous les contenus sont enregistrés et disponibles dans la base de données, nous pouvons développer le mini-site.

## 1. Les requêtes MySQL nécessaires au projet

La page d'accueil contiendra un menu de navigation ainsi qu'un simple texte « en dur ».

Cette expression « en dur » signifie que le texte en question ne sera pas issu de la base de données mais sera écrit directement dans le fichier PHP. Le menu de navigation doit lister les noms des villes sous forme de liens. Chaque lien renvoie vers la page de présentation de la ville.

Pour lister l'ensemble des villes, nous savons que nous pouvons utiliser une requête SELECT comme ceci :

```
SELECT ville_id, ville_nom FROM villes
```

Fig.6

Nous obtiendrons donc l'ensemble des noms de villes que nous pourrons intégrer dans une liste de lien avec les éléments <ul>, <li> et <a>.

Comme lors de l'exercice avec les tableaux de données PHP (arrays), l'identifiant sera utilisé pour créer une variable externe de type GET. L'URL de chaque page sera donc de la forme [http://localhost/villes\\_site/ville.php?id=1](http://localhost/villes_site/ville.php?id=1).

L'identifiant de la ville qui sera propagé dans l'URL sera récupéré pour être intégré dans la requête destinée à afficher les valeurs de la ville qui possède dans la table cet identifiant.

Cette requête sera également un SELECT contenant une clause WHERE :

```
SELECT ville_nom, ville_texte FROM villes WHERE id = 1
```

Fig.7

## 2. Le fichier index.php

Nous créons le répertoire « villes\_site ». Les fichiers utilisés seront index.php et ville.php.

Le fichier index.php servira de page d'accueil et contient un texte ainsi que le menu de navigation.

Le code du fichier index.php ne pose pas de problème particulier:

```
<!DOCTYPE html>
<html>
<head>
<title>Accueil</title>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<div>
<h1>Accueil</h1>
<p>Bienvenue sur le mini-site consacré aux villes.</p>
<p>Ce site utilise PHP et MySQL</p>
<p>Utilisez le menu de navigation pour consulter les pages du site.</p>
<p>Bonne visite !</p>
<?php
// 1. connexion
$mysqli = new mysqli('localhost', 'root', '', 'projet_villes_site');

// 2. requête
$result = $mysqli->query('SELECT ville_id, ville_nom FROM villes');

// 3.a. transformation en array avec fetch_array
while ($row = $result->fetch_array())
{
    // 3.b. création du nouvel array pour affichage hors de la boucle
    $villes[$row['ville_id']] = $row['ville_nom'];
}

// 4. Affichage
?>
</div>
<ul>
<li><a href="index.php">Accueil</a></li>
<?php foreach($villes as $id => $ville) : ?>
<li><a href="ville.php?id=<?php echo $id ?>"><?php echo $ville ?></a></li>
<?php endforeach ?>
</ul>
<?php
/* Libération des résultats */
$result->free();
/* Fermeture de la connexion */
$mysqli->close();
?>
</body>
</html>
```

Fig.8

### 3. Le fichier ville.php

Le fichier ville.php doit en premier lieu récupérer la variable externe `$_GET['id']` pour ensuite utiliser cette valeur afin de construire la requête.

```

<!DOCTYPE html>
<html>
<head>
<?php
// 1. récupération de la variable externe
$id = $_GET['id'];
// 2. connexion à la base
$mysqli = new mysqli('localhost', 'root', '', 'projet_villes_site');
// 3. requête.
// Concaténation de la requête avec la variable $id.
$result = $mysqli->query('SELECT ville_id, ville_nom, ville_texte FROM
villes WHERE ville_id = ' . $id );
// 4. création du nouvel array
$row = $result->fetch_array();
// 5. Affichage
$nom = $row['ville_nom'];
$texte = $row['ville_texte'];
?>
<title><?php echo $nom ?></title>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<div>
<h1><?php echo $nom ?></h1>
<p><?php echo $texte ?></p>
<?php
// 2. requête
$result = $mysqli->query('SELECT ville_id, ville_nom FROM villes');
// 3.a. transformation en array avec fetch_array
while ($row = $result->fetch_array())
{
    // 3.b. création du nouvel array pour affichage hors de la boucle
    $villes[$row['ville_id']] = $row['ville_nom'];
}
// 4. Affichage
?>
</div>
<ul>
<li><a href="index.php">Accueil</a></li>
<?php foreach($villes as $id => $ville) : ?>
<li><a href="ville.php?id=<?php echo $id ?>"><?php echo $ville ?></a></li>
<?php endforeach ?>
</ul>
<?php
/* Libération des résultats */
$result->free();
/* Fermeture de la connexion */
$mysqli->close();
?>
</body>
</html>

```

Fig.9

Résultats:

## Accueil

Bienvenue sur le mini-site consacré aux villes.

Ce site utilise PHP et MySQL

Utilisez le menu de navigation pour consulter les pages du site.

Bonne visite !

- [Accueil](#)
- [Paris](#)
- [Rome](#)
- [Berlin](#)
- [Moscou](#)
- [Madrid](#)

Fig.10 Page d'accueil du site

## Paris

Paris, ville la plus peuplée et capitale de la France, chef-lieu de la région Île-de-France et unique commune-département du pays, se situe au centre du Bassin parisien, sur une boucle de la Seine, entre les confluents avec la Marne en amont et avec l'Oise en aval. Ses habitants s'appellent les Parisiens. La ville est divisée en 20 arrondissements.

- [Accueil](#)
- [Paris](#)
- [Rome](#)
- [Berlin](#)
- [Moscou](#)
- [Madrid](#)

Fig. 11 Page de Paris

Nous constatons que le fichier ville.php permet de générer toutes les pages « villes » du site.

La génération de pages par une base de données s'appelle la **génération à la volée**. En effet, la page Paris ou la page Berlin sont créées uniquement lorsque l'internaute clique sur le lien correspondant. Il s'agit de pages dites **pages dynamiques**. Nous disons d'un site qu'il s'agit d'un **site dynamique** lorsque son contenu est enregistré dans une base de données, par opposition à un **site statique** dont chaque page est supportée par un seul fichier, comme par exemple les sites avec des pages HTML. Un fichier permettant de créer un nombre illimité de pages dynamiques s'appelle un **template** (ou modèle).

## 4. Optimisation du code avec la fonction require()

Comme nous le constatons les deux fichiers PHP contiennent du code identique:

- la connexion à la base de données ;
- le menu de navigation ;
- la libération des résultats et la fermeture de la connexion.

Nous utilisons la fonction native PHP require() qui permet d'inclure des fichiers dans un autre fichier. Ainsi nous n'écrirons qu'une seule fois ces lignes de code dans un fichier dédié que nous incluons dans le fichier principal et si nous devons les modifier, par exemple mettre à jour les paramètres de la base de données, nous n'aurons à le faire qu'une seule fois. Le code est optimisé et le développement et la maintenance sont simplifiés.

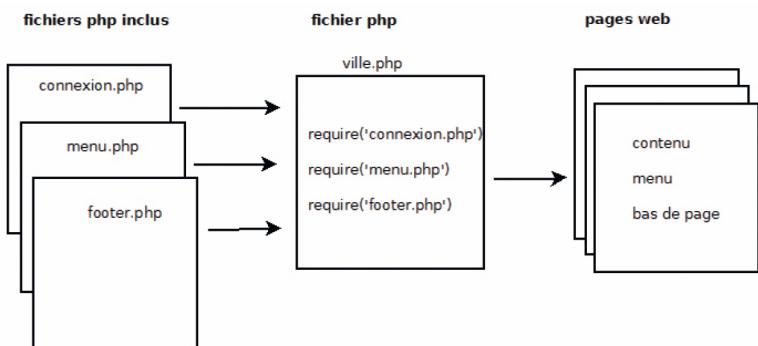


Fig. 12 Principe de l'inclusion avec la fonction require()

La gestion des inclusions de fichiers est une des bases du développement d'un site web.

## 5. Les fichiers de l'inclusion

Afin de distinguer les fichiers inclus des autres fichiers PHP, nous les préfixons avec le préfixe

« inc\_ » (inc pour inclus).

### inc\_connexion.php

```
<?php
/*
fichier inclus: inc_connexion.php
contient la connexion à la base de données
-----
$mysqli = new mysqli('localhost', 'root', '', 'projet_villes_
site');
```

Fig.13

### inc\_menu.php

```
<?php
/*
fichier inclus : inc_menu.php
contient le menu de navigation des villes
-----
// requête
$result = $mysqli->query('SELECT ville_id, ville_nom FROM villes');
// fetch_array
while ($row = $result->fetch_array())
{
    // création du nouvel array pour affichage ultérieur
    $villes[$row['ville_id']] = $row['ville_nom'];
}
// Affichage
?>
<ul>
<li><a href="index.php">Accueil</a></li>
<?php foreach($villes as $id => $ville) : ?>
<li><a href="ville.php?id=<?php echo $id ?>"><?php echo $ville ?></a></li>
<?php endforeach ?>
</ul>
```

Fig.14

### inc\_footer.php

```
<?php
/*
fichier inclus : inc_footer.php
contient la fermeture de la connexion et la fin de page HTML
-----
// Libération des résultats
$result->free();
// Fermeture de la connexion
$mysqli->close();
?>
</body>
</html>
```

Fig.15

## **index.php**

```
<?php require('inc_connexion.php'); ?>
<!DOCTYPE html>
<html>
<head>
<title>Accueil</title>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<div>
<h1>Accueil</h1>
<p>Bienvenue sur le mini-site consacré aux villes.</p>
<p>Ce site utilise PHP et MySQL</p>
<p>Utilisez le menu de navigation pour consulter les pages du site.</p>
<p>Bonne visite !</p>
</div>
<?php require('inc_menu.php') ?>
<?php require('inc_footer.php') ?>
```

Fig. 16

## **ville.php**

```
<?php require('inc_connexion.php'); ?>
<!DOCTYPE html>
<html>
<head>
<?php
// récupération de la variable externe
$Id = $_GET['id'];

// requête.
$result = $mysqli->query('SELECT ville_id, ville_nom, ville_texte
FROM villes WHERE ville_id = ' . $Id );

// création du nouvel array
$row = $result->fetch_array();

// variables destinées à l'affichage
$nom = $row['ville_nom'];
$texte = $row['ville_texte'];
?>
<title><?php echo $nom ?></title>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<div>
<h1><?php echo $nom ?></h1>
<p><?php echo $texte ?></p>
</div>
<?php require('inc_menu.php') ?>
<?php require('inc_footer.php') ?>
```

Fig. 17



La répartition du code est ici proposée à titre d'exemple. Le fichier intitulé footer.php (footer en anglais : bas de page) contient à la fois la fermeture de connexion MySQL et la fin de code HTML.



À noter que la fonction require() a un fonctionnement identique à la fonction include() ou la fonction require\_once() (inclure une seule fois).

Nous constatons que le code réparti dans plusieurs fichiers est beaucoup plus lisible et simple à maintenir.

Il est tout à fait envisageable de créer deux fichiers à inclure contenant respectivement ces portions de code, par exemple : inc\_close.php pour la fermeture de connexion et inc\_footer.php pour le code HTML.

De la même façon, un fichier inc\_head.html (head en anglais : en-tête), pourrait également être créé et contiendrait uniquement les en-têtes HTML.

## B. Exercice d'application: un gestionnaire de contenu en base de données

Maintenant que nous avons construit le mini-site avec la base de données, nous pouvons exploiter le fait qu'il est supporté par une base de données pour élaborer nous même un système de gestion de contenus. Le principe est de créer une interface simple qui permette de créer, de modifier ou de supprimer le contenu dans la base sans avoir à utiliser PhpMyAdmin, qui est un outil destiné au développeur et non un gestionnaire de contenu qu'un simple utilisateur peut prendre en main pour gérer le contenu d'un site web.

Nous pourrions réaliser une page contenant un formulaire qui permette d'enregistrer dans la base une nouvelle ville en saisissant simplement le nom de la ville et son texte de présentation.

Une autre page nous permettrait de modifier un contenu existant pour corriger une erreur de saisie ou mettre à jour des informations au moyen d'un formulaire pré-rempli.

Enfin, nous pourrions avoir à supprimer un enregistrement de la base de données, et un lien pointant vers une page PHP contenant la requête de suppression serait un moyen pratique d'effectuer cette opération.

Commençons par l'ajout de contenu en créant un fichier nommé ajout.php. Ce fichier va contenir un formulaire en mode POST constitué de deux champs:

- un champ pour la saisie du nom de la ville;

```
<input type="text" name="ville_nom" />
```

Fig. 18

- un textarea pour la saisie du texte

```
<textarea name="ville_texte" /></textarea>
```

Fig. 19

La validation du formulaire transmet donc la saisie de l'utilisateur dans deux valeurs externes de type POST (`$_POST['ville_nom']` et `$_POST['ville_texte']`) que nous allons affecter respectivement aux variables `$ville_nom` et `$ville_texte` afin de les utiliser dans une requête INSERT INTO destinée à ajouter dans la table un enregistrement constitué du nom de la nouvelle ville et de son texte de présentation.

```
INSERT INTO villes (ville_nom, ville_texte) VALUES ($ville_nom, $ville_texte);
```

Fig. 20

Lors de l'intégration de cette requête MySQL dans le code PHP, il faudra être attentif à la concaténation car les champs de type TEXT et VARCHAR nécessitent des doubles quotes dans les requêtes MySQL. Rappelons que pour PHP, une requête MySQL est du texte simple, comme le HTML.

La requête en PHP sera donc de la forme :

```
$mysqli->query('INSERT INTO villes (ville_nom, ville_texte)  
VALUES ("'. $ville_nom .'", "'. $ville_texte .'")')
```

Fig.21

Les double quotes permettent d'annoncer correctement à MySQL qu'il s'agit de chaînes de caractères (VARCHAR ou TEXT). Ensuite les simples quotes font partie de la concaténation imposée par l'utilisation de variables PHP dans la requête.

Contrairement aux requêtes SELECT, nous ne cherchons pas à afficher les valeurs de la base, et nous n'avons donc pas besoin de créer une variable \$result. Ensuite, et il s'agit d'un point important dans toute mise en oeuvre d'interaction utilisateur, nous avons besoin de confirmer que l'ajout a bien été effectué. Nous créons donc une condition if pour cette requête qui nous permet d'afficher un message de confirmation.

### Code basique complet du fichier ajout.php

```
<?php require('inc_connexion.php'); ?>  
<!DOCTYPE html>  
<html>  
<head>  
<link rel="stylesheet" type="text/css" href="style.css" />  
</head>  
<body>  
<?php  
// récupération des variables  
if(isset($_POST['submit_form']))  
{  
    $ville_nom = $_POST['ville_nom'];  
    $ville_texte = $_POST['ville_texte'];  
    // requête insert into  
    if ($mysqli->query('INSERT INTO villes (ville_nom, ville_texte)  
VALUES ("'.$ville_nom.'", "'.$ville_texte.'")'))  
    {  
        /* si la requête est effectuée, elle retourne le booléen TRUE et  
        donc le message pourra être affiché*/  
        $message = '<p class="message">L\'ajout de la ville '.  
        $ville_nom .' est effectué.</p>';  
    }  
}  
?>  
<div>  
<h1>Ajouter une ville</h1>  
<?php if(isset($message)) echo $message ?>  
<form method="post">  
<p>Nom de la ville : <input type="text" name="ville_nom" /></p>  
<p>Texte de présentation<br>  
<textarea name="ville_texte" cols="32" rows="8"/></textarea>  
</p>  
<p><input type="submit" name="submit_form" value="valider" /></p>  
</div>  
</body>  
</html>
```

Fig.22

Ce code est fonctionnel mais nécessite certaines optimisations. En effet, si nous validons le formulaire sans saisir de nom ou de texte de ville, des valeurs vides seront enregistrées dans la base de données.

Nous vérifions ce point avec la fonction `empty()`. D'autre part, au cas où MySQL n'enregistre pas notre saisie en ne prenant pas en compte la requête INSERT INTO, par exemple à cause d'une erreur sur les noms des champs de la table, nous allons également ajouter un message d'erreur.

Enfin, si nous ajoutons à nouveau la ville de Paris, alors nous aurons deux fois Paris dans la table. Il convient donc de vérifier également que la ville saisie n'est pas déjà dans la base. Nous pouvons effectuer une requête SELECT qui si elle retourne un résultat nous confirmera ou non l'existence de la ville dans la base.

Nous allons utiliser la commande dédiée count() de MySQL pour compter le résultat. Si Paris est déjà enregistré une fois dans la table, alors SELECT count() retournera la valeur 1. Nous effectuons le décompte en passant en argument le nom du champ ville\_id dans count() :

```
SELECT count(ville_id) FROM villes WHERE ville_nom = "Paris"
```

Fig.23

Cette requête MySQL retourne 1 et nous pouvons l'utiliser simplement dans notre code PHP comme suit : si la valeur est 0, la ville n'existe pas dans la table et nous pouvons alors procéder à l'insertion, si la valeur est supérieure à 0, alors nous n'effectuons naturellement pas l'insertion et nous affichons un message explicatif.

### Code PHP optimisé de ajout.php

```
<?php
// récupération des variables
if(isset($_POST['submit_form']))
{
    $ville_nom = $_POST['ville_nom'];
    $ville_texte = $_POST['ville_texte'];
    // vérification du contenu des variables
    if(empty($ville_nom) OR empty($ville_texte))
    {
        $message = '<p class="error">Vous devez saisir le nom d\'une ville et sa présentation.</p>';
    }
    else
    {
        // la ville existe-t-elle dans la base ?
        // effectuons une requête SELECT avec count()
        $result = $mysqli->query('SELECT count(ville_id)
        FROM villes WHERE ville_nom = "' . $ville_nom . '"');
        $row = $result->fetch_array();
        // $row[0] contient la valeur retournée par le count() de MySQL
        if($row[0] > 0)
        {
            $message = '<p class="error">La ville est déjà enregistrée.</p>';
        }
        else
        {
            // requête insert into
            if ($mysqli->query('INSERT INTO villes
            (ville_nom, ville_texte)
            VALUES
            ("' . $ville_nom . '", "' . $ville_texte . '")'))
            {
                $message = '<p class="message">L\'ajout de la ville
                '. $ville_nom .' est effectué.</p>';
            }
            else
            {
                $message = '<p class="error">L\'ajout de la ville '. $ville_nom .' n\'est pas effectué.</p>';
            }
        }
    }
}
```

Fig.24

Nous pouvons maintenant procéder à la gestion de la suppression des contenus dans la base. Dans un fichier liste.php, nous affichons donc la liste de toutes les villes sous la forme d'un lien qui pointera vers un fichier contenant la requête MySQL de suppression DELETE.

Le code de base permettant l'affichage de toutes les villes a déjà été utilisé dans le fichier inc\_menu.php et nous récupérons donc ce code. La différence est que nous allons utiliser cette liste de liens pour supprimer une ville ou bien la mettre à jour.

Nous gardons le lien vers ville.php qui permet de consulter le contenu, et nous ajoutons deux liens:

- un lien éditer qui pointe vers un fichier edition.php;
- un lien supprimer qui pointe vers un fichier suppression.php.

Chacun de ces liens comporte naturellement l'identifiant de la ville dans une variable externe de type \$\_GET['id'].

Le résultat pourra être celui-ci :

- Paris - [\[éditer\]](#) - [\[supprimer\]](#)
- Rome - [\[éditer\]](#) - [\[supprimer\]](#)
- Berlin - [\[éditer\]](#) - [\[supprimer\]](#)
- Moscou - [\[éditer\]](#) - [\[supprimer\]](#)
- Madrid - [\[éditer\]](#) - [\[supprimer\]](#)
- Marseille - [\[éditer\]](#) - [\[supprimer\]](#)

*Fig. 25 Liste des villes permettant l'édition ou la suppression*

Le fichier suppression.php contient trois parties:

- la récupération de l'identifiant (la variable de type GET);
- la requête de suppression qui utilise l'identifiant pour préciser à MySQL quelle est la ville à supprimer;
- un message de confirmation et un lien retour vers la page liste.php.

## Code du fichier suppression.php

```
<?php require('inc_connexion.php'); ?>
<!DOCTYPE html>
<html>
<head>
<?php
// récupération de la variable externe
$id = $_GET['id'];
// requête.
if($mysqli->query('DELETE FROM villes WHERE ville_id = ' . $id ))
{
    $message = '<p class="message">La ville a bien été supprimée
dans la base.<br><a href="liste.php">Accéder à la liste des
villes</a></p>';
}
else
{
    $message = '<p class="error">Erreur de la suppression.</p>';
}
?>
<title>Suppression</title>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<div>
<?php echo $message ?>
</div>
</body>
</html>
```

Fig.26

Nous pouvons à présent mettre en oeuvre l'interface d'édition. Celle-ci est destinée à modifier le nom et la présentation d'une ville. Elle contient donc le même formulaire que la page d'ajout mais les champs sont pré-remplis avec les informations issues de la base de données. La restitution des informations de la base dans le formulaire pré-rempli permet de modifier directement les informations puis de valider notre saisie pour mettre à jour la base.

### Modifier une ville

Nom de la ville :

Texte de présentation

Paris, ville la plus peuplée et capitale de la France, chef-lieu de la région Ile-de-France et unique commune-département du pays, se situe au centre du Bassin parisien, sur une boucle de la Seine, entre les confluents avec la Marne en amont et avec l'Oise

Fig.27 Page d'édition d'une ville

### Code du fichier edition.php

**Partie 1 :** la requête UPDATE utilise les informations transmises via le formulaire. Notons que l'identifiant de la ville ville\_id est également transmis.

```
<?php require('inc_connexion.php'); ?>
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<?php
/*
Partie 1 : gestion de la saisie et enregistrement
-----*/
// récupération des variables
if(isset($_POST['submit_form']))
{
    $ville_nom = $_POST['ville_nom'];
    $ville_texte = $_POST['ville_texte'];
    $ville_id = $_POST['ville_id'];
    // vérification du contenu des variables
    if((empty($ville_nom)) OR empty($ville_texte))
    {
        $message = '<p class="error">Vous devez saisir le nom d\'une
ville et sa présentation.</p>';
    }
    else
    {
        // requête UPDATE
        if ($mysql->query('UPDATE villes SET ville_nom = "'. $ville_
nom .'", ville_texte = "'. $ville_texte .'" WHERE ville_id = '.
$ville_id))
        {
            $message = '<p class="message">La mise à jour de la ville ' .
$ville_nom . ' est effectuée.</p>';
        }
        else
        {
            $message = '<p class="error">La mise à jour de la ville ' .
$ville_nom . ' n\'est pas effectuée.</p>';
        }
    }
}
}
```

Fig.28

**Partie 2 :** le formulaire est pré-rempli par la requête SELECT qui est identique à celle utilisée dans le fichier ville.php. Notons que l'identifiant ville\_id est récupéré depuis la base de données afin de transmettre l'identifiant de la ville dans la requête de mise à jour UPDATE.

La variable est placée dans le formulaire dans un champ input de type hidden, afin que l'utilisateur ne puisse pas la modifier.

```

/*
Partie 2 : récupération des informations de la base
et affichage dans le formulaire
-----
// récupération de la variable externe
$id = $_GET['id'];
// requête.
$result = $mysqli->query('SELECT ville_id, ville_nom, ville_texte
FROM villes WHERE ville_id = ' . $id );
// création du nouvel array
$row = $result->fetch_array();
// variables destinées à l'affichage
$nom = $row['ville_nom'];
$texte = $row['ville_texte'];
?>
<div>
<h1>Ajouter une ville</h1>
<?php if(isset($message)) echo $message ?>
<form method="post">
<p>Nom de la ville : <input type="text" name="ville_nom"
value="<?php echo $nom ?>" /></p>
<p>Texte de présentation<br>
<textarea name="ville_texte" cols="32" rows="8"/><?php echo $texte
?></textarea></p>
<input type="hidden" name="ville_id" value="<?php echo $id ?>" />
<p><input type="submit" name="submit_form" value="valider" /></p>
</form>
</div>
</body>
</html>

```

Fig.29

Nous avons donc créé les fichiers suivants permettant l'ensemble des manipulations attendues :

Tableau n°1

FICHIER	ACTION
liste.php	Liste les villes et permet l'accès à la mise à jour ou la suppression.
suppression.php	Supprime une ville.
edition.php	Édite une ville.
ajout.php	Ajoute une ville.

Il ne reste plus qu'à créer une page nommée admin.php et un menu de navigation qui va permettre d'accéder à l'ensemble de ces actions. La page admin.php contient l'inclusion par défaut du fichier liste.php. Elle sera également accessible depuis le mini-site. Nous verrons ultérieurement comment créer un accès réservé avec mot de passe à l'administration du site. Le code ne présentant pas de difficultés particulières, nous ne le détaillons pas.

## Accueil

Bienvenue sur le mini-site consacré aux villes.

Ce site utilise PHP et MySQL

Utilisez le menu de navigation pour consulter les pages du site.

Bonne visite !

- [Accueil](#)
- [Paris](#)
- [Rome](#)
- [Moscou](#)
- [Marseille](#)
- [Lille](#)
- [Administration](#)

Fig.30 Page d'accueil du mini-site

## Administration du mini-site des villes

L'administration du site vous permet d'ajouter une nouvelle ville au site ou de modifier ou supprimer une ville existante.

- [Accueil administration](#)

- [Ajouter une ville](#)

- [Voir le site](#)

Liste des villes :

- [Paris](#) - [éditer] - [supprimer]
- [Rome](#) - [éditer] - [supprimer]
- [Moscou](#) - [éditer] - [supprimer]
- [Marseille](#) - [éditer] - [supprimer]
- [Lille](#) - [éditer] - [supprimer]

Fig.31 Page d'accueil de l'administration du mini-site

Le mini-site tel que peuvent le consulter les internautes est généralement appelé le **front-office** (ou simplement le front). L'administration du mini-site qui est réservée au webmaster du site est appelée le **back-office**.

Le principe est que la base de données contienne le contenu du site tandis que l'administration du site permette de modifier les informations de la base.

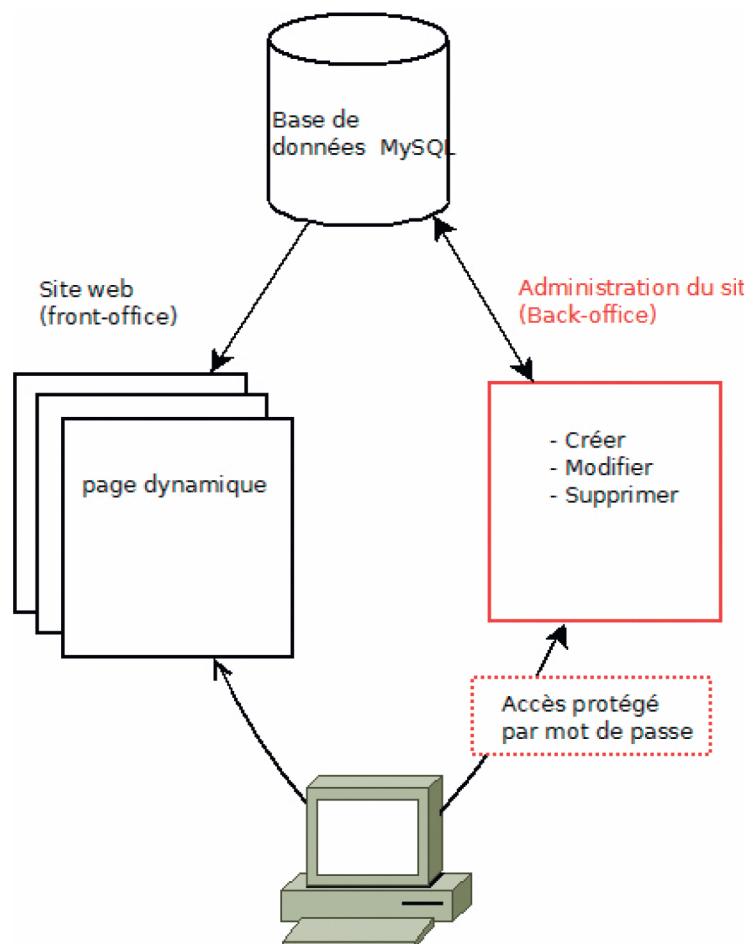


Fig.32 Principe d'architecture de site dynamique.

Cette architecture classique est typique du fonctionnement d'un site dynamique et toutes les solutions de gestion de site (Wordpress, Drupal, Magento, Prestashop, etc.), fonctionnent selon ce principe de CMS ou Content Management System (en français : système de gestion de contenus). Le CMS permet de créer, modifier ou supprimer des informations contenues dans la base. Pour information, les CMS sont dédiés à un type de besoin comme la gestion d'un blog ou d'un site avec Wordpress ou Drupal ou d'un site e-commerce avec Magento ou Prestashop, pour ne citer que les plus connus.

Nous constatons dans tous les cas que l'interface d'administration permet de modifier l'intégralité d'un site web et que donc son accès ne devrait pas être autorisé à tous les internautes. Il convient en effet de protéger cet accès avec un mot de passe. Le principe est que l'utilisateur puisse s'identifier comme étant le webmaster du site. Lorsqu'il souhaite accéder à l'administration, un formulaire lui demande de valider son identifiant et son mot de passe. S'il est reconnu, alors l'identification a réussi. À ceci s'ajoute le besoin de reconnaître lors de chaque action la personne autorisée à administrer le site. Mais il ne semble pas pertinent de demander le mot de passe avant chaque action. Il est donc utile de la reconnaître durant toute la session d'utilisation. C'est le mécanisme que nous étudierons dans le module consacré aux sessions.

## C. Ajout d'une table Pays

Pour finaliser l'exemple, nous ajoutons une table « pays » qui va nous servir à démontrer comment nous pouvons travailler avec deux tables. L'objectif est d'associer une ville à un pays, et d'afficher celui-ci lorsque nous affichons la ville. Nous aurons également besoin d'afficher la liste de toutes les villes et de leurs pays, enfin nous souhaitons afficher pour un pays la liste de toutes les villes qui y sont reliées. Nous allons commencer par modifier l'interface d'administration pour ajouter un pays à toutes les villes.

Commençons par ajouter à la base de données la table qui contient seulement deux champs en fonction des informations fournies dans cette requête de création de table.

```
CREATE TABLE `pays` (
  `pays_id` int(11) NOT NULL AUTO_INCREMENT,
  `pays_nom` varchar(15) NOT NULL,
  PRIMARY KEY (`pays_id`)
) ;
```

Fig.33

Insérons ensuite comme valeurs les noms des pays suivants : France, Espagne, Allemagne, Russie, Italie. Nous obtenons donc pour cette table les valeurs suivantes.

Tableau n°2 Table pays

PAYS_ID	PAYS_NOM
1	France
2	Espagne
3	Allemagne
4	Russie
5	Italie

Pour la mise à jour de la table villes, il nous faut en effet associer les villes et les pays.

Nous créons donc un nouveau champ pays\_id dans la table villes de manière à relier la table villes et la table pays. Nous avons vu ce principe en démarrant le cours sur les bases de données.

Ainsi nous avons pour la valeur Paris de la table villes ces informations:

PAYS_ID	VILLE_NOM	VILLE_TEXTE	PAYS_ID
1	Paris	Paris etc.	1

Grâce à ce nouveau champ pays\_id, nous savons que le pays\_id de Paris est 1 et donc que le pays de Paris est France, puisque le pays\_id de la table ville est égal au pays\_id de la table pays. C'est cette équivalence que nous allons finaliser pour l'ensemble des villes et qui nous permettra d'effectuer une requête à MySQL pour qu'il nous restitue les pays de chaque ville.



Attention: effectuer des sauvegardes de vos fichiers avant d'effectuer les modifications: copier-coller les fichiers dans un répertoire de sauvegarde (par exemple: backup\_villes) que vous pouvez zipper pour plus de précaution. Ainsi en cas de besoin vous avez toujours les fichiers originaux.

Dans l'administration, nous ajoutons une page permettant d'ajouter et modifier les pays (cf. exercice à réaliser) puis nous modifions les fichiers edition.php et ajout.php qui génèrent respectivement les pages de mise à jour et d'ajout des villes pour leur associer le pays.

## 1. Modification de la page de mise à jour d'une ville

Pour ajouter la gestion des pays aux villes à la page d'édition, nous effectuons cette simple requête:

```
SELECT pays.pays_id, pays_nom FROM pays INNER JOIN villes
WHERE villes.pays_id = pays.pays_id GROUP BY pays_nom ORDER BY
pays_nom
<?php foreach($pays_liste as $id => $pays) : ?>
<li><a href="pays.php?id=<?php echo $id ?>"><?php echo $pays ?></
a></li>
<?php endforeach ?>
```

Fig.34

Et nous en affichons le résultat au moyen d'une boucle foreach:

```
<?php foreach($pays_liste as $pays_id => $pays_nom) : ?>
<?php if($ville_pays_id == $pays_id) : ?>
<li class="bg_gris"> <input checked="checked" type="radio"
name="pays_id" value="<?php echo $pays_id ?>" /><?php echo $pays_
nom ?></li>
<?php else : ?>
<li> <input type="radio" name="pays_id" value="<?php echo $pays_id
?>" /><?php echo $pays_nom ?></li>
<?php endif ?>
<?php endforeach ?>
```

Fig.35

La condition if/else dans le code ci-dessus nous permet de montrer si un pays est déjà associé à la ville en utilisant l'attribut HTML checked. La variable \$ville\_pays\_id est obtenue par la modification de requête initiale d'affichage de la ville dans le fichier edition.php. Elle contient la valeur qui est enregistrée dans la base correspondant au champ pays\_id de cette ville.

```
$result = $mysqli->query('SELECT ville_id, ville_nom, ville_texte,
pays_id FROM villes WHERE ville_id = ' . $id );
$row = $result->fetch_array();
// variables destinées à l'affichage
$nom = $row['ville_nom'];
$texte = $row['ville_texte'];
$ville_pays_id = $row['pays_id'];
```

Fig.36

Il ne reste plus qu'à modifier la page permettant d'ajouter d'une ville (cf. exercice à réaliser).

Résultat:

## Modifier une ville

Nom de la ville :

### Texte de présentation

Paris, ville la plus peuplée et capitale de la France, chef-lieu de la région Ile-de-France et unique commune-département du pays, se situe au centre du Bassin parisien, sur une boucle de la Seine, entre les confluents avec la Marne en amont et avec l'Oise

- France
- Espagne
- Allemagne
- Russie
- Italie
- Belgique

Fig.37 Interface de mise à jour avec gestion des villes

Nous pouvons maintenant associer les villes et les pays, nous allons ajouter sur les pages du site le nom du pays pour chaque ville.

## 2. Afficher le nom du pays pour une ville

De la même manière que le fichier edition.php, nous modifions la requête existante pour récupérer le champ pays\_id de la table villes.

```
// requête.  
$result = $mysqli->query('SELECT ville_id, ville_nom, ville_texte,  
pays_id FROM villes WHERE ville_id = ' . $id );  
$row = $result->fetch_array();  
  
$nom = $row['ville_nom'];  
$texte = $row['ville_texte'];  
$ville_pays_id = $row['pays_id'];
```

Fig.38

Nous ajoutons la requête dans le fichier ville.php. Cette requête intègre l'identifiant pays\_ville\_id et retourne le nom du pays qu'il nous suffit d'intégrer.

```
$result = $mysqli->query('SELECT pays_id, pays_nom  
FROM pays WHERE pays_id = ' . $ville_pays_id );  
$row = $result->fetch_array();  
$pays_nom = $row['pays_nom'];
```

Fig.39

## 3. Afficher la liste de toutes les villes et de leurs pays

Nous souhaitons afficher dans le menu la liste des pays qui sont associés à une ou plusieurs villes et pour chacun de ces pays nous souhaitons créer un lien qui ouvrirait une page listant les villes en question.

Nous allons avancer pas à pas et commencer par lister pour toutes les villes les pays qui y sont associés.

Pour afficher la liste de toutes les villes et de leurs pays, nous avons besoin d'une requête MySQL permettant de relier (joindre) les informations de la table villes et les informations de la table pays. Cette requête **SELECT** qui s'appelle une jointure s'appuie sur la commande **INNER JOIN**.

La syntaxe est la suivante :

```
SELECT pays_nom, ville_nom  
FROM pays INNER JOIN villes  
WHERE villes.pays_id = pays.pays_id
```

Fig.40

Tableau n°3

Nous demandons à MySQL de nous retourner les valeurs des champs suivants : pays\_nom et ville\_nom depuis (FROM) la table pays jointe (INNER JOIN) à la table villes lorsque (WHERE) le champ pays\_id de la table villes est égal au champ pays\_id de la table pays.

MySQL va donc lister les valeurs des champs et à chaque fois vérifier si les identifiants pays\_id correspondent pour les deux tables.

Les champs pays\_id ont le même nom pour les deux tables, le fait d'ajouter un point entre le nom de la table et le nom du champ permet de spécifier précisément quel est le champ courant.

Testons cette requête dans l'interface SQL de PhpMyAdmin, le résultat est le suivant :

Tableau n°4

PAYS_NOM	VILLE_NOM
France	Paris
Italie	Rome
Russie	Moscou
France	Marseille
France	Lille

Nous avons avancé par rapport à notre besoin initial car nous sommes en mesure de lister tous les pays qui sont associés à une ville. Il suffit de limiter la requête au seul champ pays\_id pour ne plus afficher le nom des villes.

```
SELECT pays_nom FROM pays INNER JOIN villes WHERE villes.pays_id = pays.pays_id
```

Fig. 41

Le problème que nous rencontrons est en revanche le fait que nous aurons trois fois le pays France dans notre liste. Il serait préférable de n'avoir qu'une seule fois le lien et nous utilisons donc la commande GROUP BY de MySQL qui permet d'éliminer les doublons d'une requête.

Nous effectuons donc un GROUP BY sur pays\_nom.

```
SELECT pays_nom FROM pays INNER JOIN villes
WHERE villes.pays_id = pays.pays_id GROUP BY pays_nom
```

Fig. 42

Enfin, si nous souhaitons afficher dans l'ordre alphabétique le résultat, nous finalisons la requête avec le ORDER BY que nous avons déjà rencontré.

```
SELECT pays_nom FROM pays INNER JOIN villes
WHERE villes.pays_id = pays.pays_id GROUP BY pays_nom ORDER BY
pays_nom
```

Fig. 43

Notre requête est prête, nous la testons dans l'interface SQL de PhpMyAdmin puis l'intégrons en PHP dans le fichier inc\_menu.php. Puisque cette liste sera composée de liens qui serviront à transmettre l'identifiant du pays dans l'URL, nous constatons que la requête doit comporter l'identifiant pays\_id.

```
SELECT pays_nom FROM pays INNER JOIN villes
WHERE villes.pays_id = pays.pays_id GROUP BY pays_nom ORDER BY
pays_nom
```

Fig. 44

Nous créons ensuite le fichier pays.php qui contiendra le nom du pays ainsi que la liste des villes pour ce pays en question.

La requête est la même que la précédente hormis le fait que nous ajoutons avec AND le fait que pays\_id doit être égal à l'identifiant du pays transmis comme variable externe. (prenoms 1 par exemple):

```
SELECT pays_nom, ville_nom, ville_id FROM pays INNER JOIN villes  
WHERE villes.pays_id = pays.pays_id AND pays.pays_id = 1
```

Fig.45

Le code est donc:

```
$pays_id = $_GET['id'];  
  
// requête.  
$result = $mysqli->query('SELECT pays_nom, ville_nom, ville_id FROM  
pays INNER JOIN villes  
WHERE villes.pays_id = pays.pays_id AND pays.pays_id = ' . $pays_id  
>;  
  
while ($row = $result->fetch_array())  
{  
    $pays_nom = $row['pays_nom'];  
    $villes[$row['ville_id']] = $row['ville_nom'];  
}
```

Fig.46

Et il ne reste plus qu'à afficher les variables obtenues.

<p><b>Liste des villes pour le pays : France</b></p> <ul style="list-style-type: none"><li>• <a href="#">Paris</a></li><li>• <a href="#">Marseille</a></li><li>• <a href="#">Lille</a></li><li>• <a href="#">Rome</a></li><li>• <a href="#">Moscou</a></li><li>• <a href="#">Berlin</a></li><li>• <a href="#">Londres</a></li></ul>	<ul style="list-style-type: none"><li>• <a href="#">Paris</a></li><li>• <a href="#">Marseille</a></li><li>• <a href="#">Lille</a></li><li>• <a href="#">Rome</a></li><li>• <a href="#">Moscou</a></li><li>• <a href="#">Berlin</a></li><li>• <a href="#">Londres</a></li></ul> <p><b>pays</b></p> <ul style="list-style-type: none"><li>• <a href="#">Allemagne</a></li><li>• <a href="#">France</a></li></ul>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Fig.47 Exemple de page listant les villes associées au pays

Nous venons de réaliser une interface de gestion de contenus en back-office. Elle va permettre au gestionnaire de cette base d'intervenir sans avoir besoin d'être développeur.