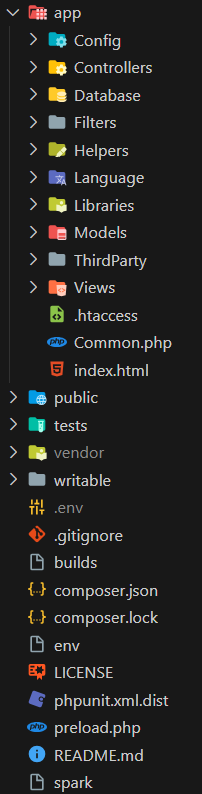
# Architecture d’un projet Code Igniter

Suite à la mise en place effectuée en première séance du TP3, vous devez tous avoir sur votre poste de travail un projet PHP copié sur le modèle fourni par CodeIgniter qui permet d’utiliser l’ensemble de ses fonctionnalités.

*⚡ Votre projet doit également être connecté au même dépôt Git par projet afin que chacun puisse contribuer au même code à la fin ! Si ça n’est pas le cas, se référer au sujet du TP3.*

La structure d’un projet Code Igniter est la suivante :

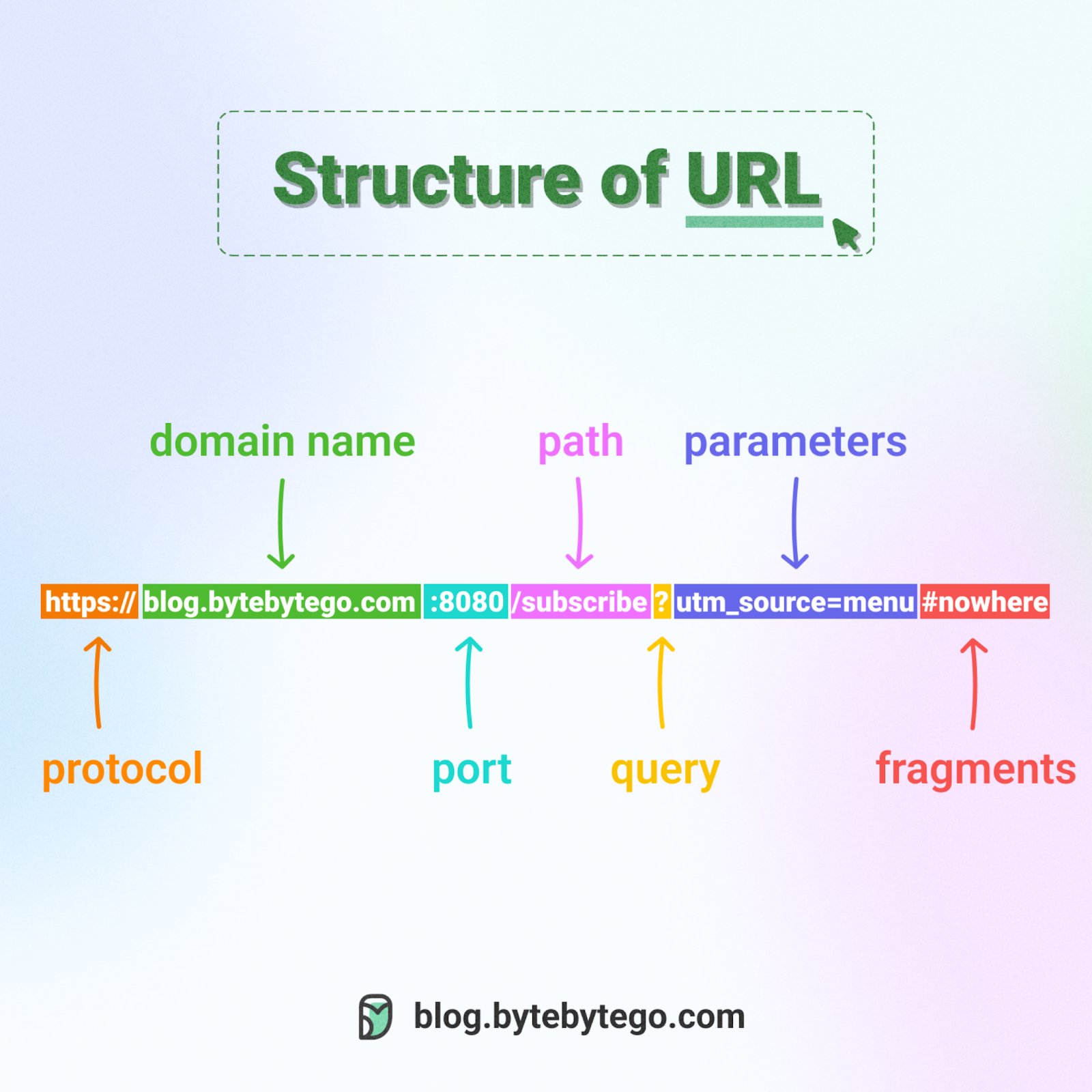


* Config: contient l’ensemble des informations de configuration de votre projet (quelles sont les routes de votre site qui sont consultables par vos utilisateurs et quel code elles exécutent, comment se connecter à la base de données, comment gérer les cookies, comment sécuriser votre site, etc…) Nous allons explorer en détail ces configurations au fil de notre TP !
* Controllers: contient l’ensemble des *controllers* qui vont implémenter la logique de notre site. C’est ici que nous allons détailler comment les pages doivent se comporter
* Database : contient des outils qui permettent de versionner votre base de données et de générer des données exemples. Nous reviendrons sur ces éléments plus tard.
* Filters : permet de générer des filtres qui empêchent les navigations à des pages non autorisées, afin de mettre en place de la sécurité
* Helpers : permet de définir des fonctions pratiques utilisables un peu partout dans notre projet appelées *helpers*. Nous reviendrons sur leur utilisation plus tard !
* Language : permet de mettre en place de l’internationalisation (créer un site multi-langues), on ne l’utilisera pas dans ce TP
* Libraries: permet de stocker les librairies externes à intégrer à Code Igniter. Nous ne les utiliserons pas dans ce TP
* Models: contient les classes *Models* représentant les données que notre site va manipuler. C’est ici que nous allons décrire les différentes entités que notre site va manipuler
* Views : contient les classes *views* qui représentent les vues de votre site, c’est-à-dire les templates HTML qui afficheront nos pages web.
* ThirdParty: contient les librairies externes qui ne s’intègrent pas avec Code Igniter, mais qui sont utilisées dans le backend. Nous ne les utiliserons pas dans ce TP

Je ne m’attarde pas sur le reste des fichiers pour le moment, on aura le temps de regarder ensemble comment ils fonctionnent.

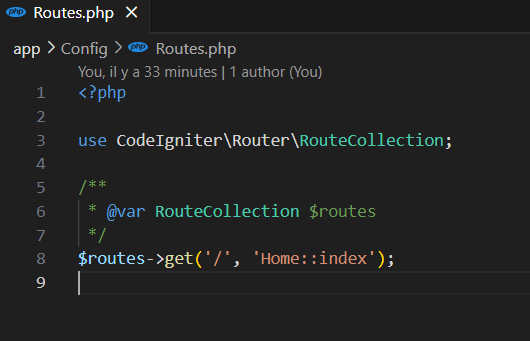
Commençons par construire ensemble notre première page en utilisant le design pattern **Model-View-Controller**. La notice de Code Igniter sur le design pattern MVC est consultable [ici](https://www.codeigniter.com/user_guide/concepts/mvc.html).

# Mise en place de votre première page

Pour rappel, une URL d’une ressource sur le web (que ce soit une page web, une API, une ressource quelconque) est structurée de la sorte :  **

Pour créer notre première page, nous allons dans un premier temps définir **la route** (c’est-à-dire, le chemin dans l’URL à partir de la base du site (protocole / domaine / port) qui permet d’y accéder.

Pour cela, nous allons **modifier notre configuration** dans le fichier Config/Routes.php:



Nous allons **déclarer une nouvelle route** en utilisant les méthodes de l’objet $routes:

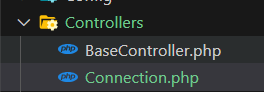
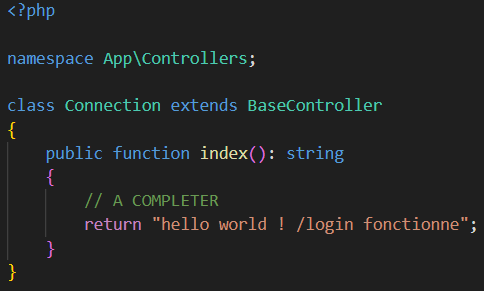


Nous décrivons ainsi au framework qu’il faut qu’il appelle la méthode index du controller Connection lorsque l’utilisateur navigue sur l’URL <protocole/domaine/ port>/login

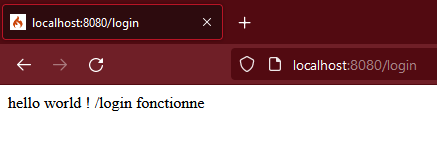
On utilise la méthode get() afin de préciser que notre site n'appellera le controller que lorsque la méthode HTTP GET est utilisée.

💡 *Plus d’infos sur la définition des routes :* [*URI Routing — CodeIgniter 4.4.6 documentation*](https://www.codeigniter.com/user_guide/incoming/routing.html#setting-routing-rules)

Notre route est déclarée, il faut désormais créer le controller Connection dans le fichier Controllers/Connection.php :

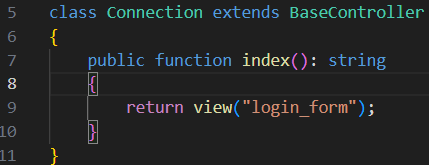
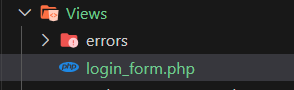


Ainsi, lorsque nous naviguons sur l’URL <http://localhost:8080/login>, le message suivant apparaît :



Cependant, nous n’écrivons pas directement notre code HTML dans notre controller mais dans une *view* (afin de séparer la logique de traitement et la définition de l’affichage de la page dans 2 classes distinctes).

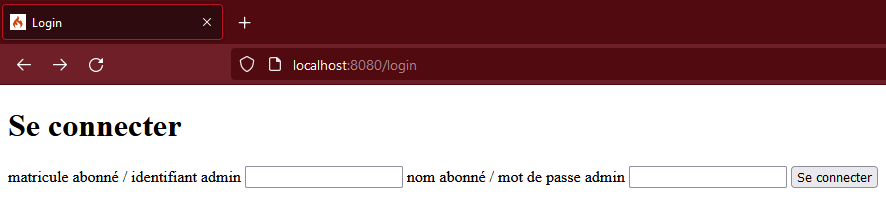
Nous allons donc créer une vue et l’utiliser dans le controller :



Dans le fichier login\_form.php, on écrit un template HTML simple (en affichant un titre et un formulaire composé de 2 inputs et un bouton pour valider) :

| <!DOCTYPE html>  <html lang="fr">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Login</title>  </head>  <body>  <h1>Se connecter</h1>  <form method="POST" action="/login">  <label for="login">matricule abonné / identifiant admin</label>  <input id="login" name="login" type="text" />  <label for="password">nom abonné / mot de passe admin </label>  <input id="password" name="password" type="password" />  <button type="submit">Se connecter</button>  </form>  </body>  </html> |
| --- |

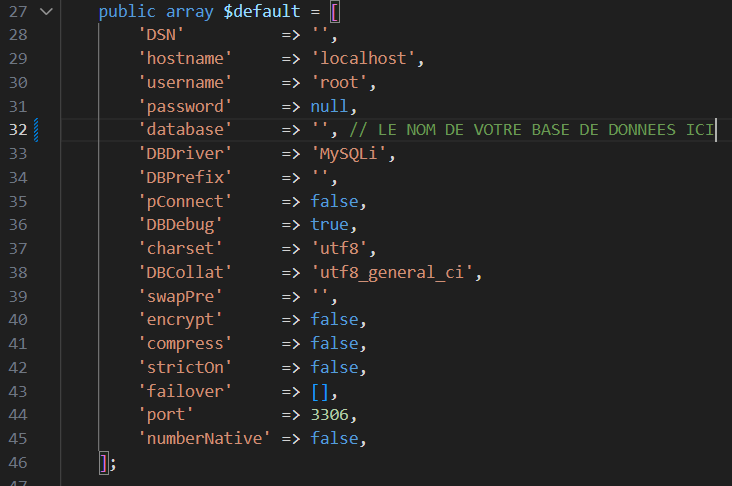
Notre page de login va donc être générée via la *view* login\_form, et si on recharge notre page, on peut voir le formulaire apparaître :



💡 *Pour le moment, on ne va pas mettre en place de style CSS, nous allons faire cela ensemble lors de la prochaine séance.*

# Connexion à une base de données

Maintenant que notre page s’affiche, nous allons devoir traiter le formulaire afin de pouvoir valider si les infos de connexion données sont correctes. Pour ce faire, nous allons éditer la configuration de connexion à la base de données de Code Igniter pour le faire interroger la base de données que vous avez créé lors de la mise en place du TP.  
Pour ce faire, nous allons modifier le fichier Config/Database.php et indiquer les informations de connexion à notre base de données ici :



Ainsi, les fonctionnalités de Code Igniter pour requêter des bases de données seront basées sur cette configuration.

💡 *Pour que votre base de données puisse être accessible, vous devez lancer Wamp !*

💡 *Vous trouverez la documentation complète sur la mise en place d ela configuration d’une base de données dans la documentation officielle de Code Igniter ici :* [Database Configuration — CodeIgniter 4.4.5 documentation](https://www.codeigniter.com/user_guide/database/configuration.html#setting-default-database)

# 

# Traitement du formulaire

## Définition d’une nouvelle route

Nous avons affiché une page web de connexion, mais nous n’avons pas encore fait le traitement qui va vérifier que les infos fournies par l’utilisateur lors de la connexion.

Pour ce faire, nous allons définir une nouvelle route qui va lire les données saisies et valider si le formulaire est bon pour l’administrateur.

Vérifier que votre formulaire est paramétré de la manière suivante :

<form method="POST" action="/login">

Puis dans la configuration des routes, ajouter une entrée qui va pointer sur une nouvelle méthode du controller Connection :

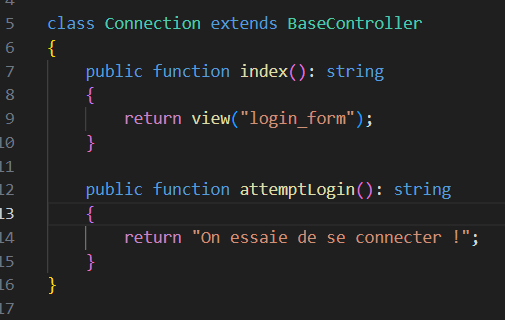


Notez qu’on utilise cette fois-ci la méthode routes->post() car on veut associer les requêtes HTTP POST sut /login à une autre logique que la méthode GET :

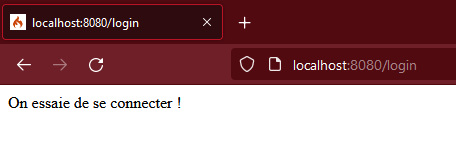
* GET /login sert à afficher le formulaire de connexion
* POST /login sert à le traiter

## Définition de la nouvelle méthode

Dans le controller Connection, nous allons rajouter la méthode attemptLogin :



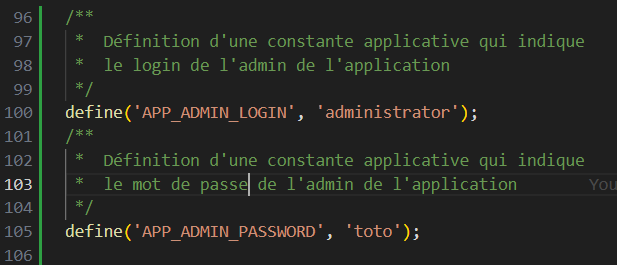
Si vous revenez sur votre formulaire, le clic sur le bouton **Se connecter** va afficher la page suivante :



Il faut maintenant récupérer le contenu du formulaire et le vérifier pour rediriger l’utilisateur vers une page d’accueil lorsqu’il est connecté.

## Définition de constantes

Nous avons défini que l’administrateur doit se connecter via des constantes que l’on va écrire en dur dans le code (pour notre TP, bien évidemment dans la vraie vie on a un système d’authentification plus élaboré !). Nous allons donc définir ces constantes dans la configuration de Code Igniter en ajoutant les entrées suivantes dans Config/Constants.php :



Ces constantes sont désormais définies et utilisables dans l’intégralité de notre application.

## Implémentation de la fonction

On peut donc revenir dans notre controller Connection.php et implémenter notre fonction de connexion. Code Igniter vous fournit des outils simples pour manipuler les requêtes et réponses à afficher à votre utilisateur, nous allons dans un premier temps utiliser l’objet request associé au controller pour récupérer les valeurs du formulaire et les contrôler :

public function attemptLogin(): string

{

$values = $this->request->getPost(['login', 'password']);

if (!empty($values) && $values['login'] == APP\_ADMIN\_LOGIN && $values['password'] == APP\_ADMIN\_PASSWORD) {

return "On a réussi à se connecter !";

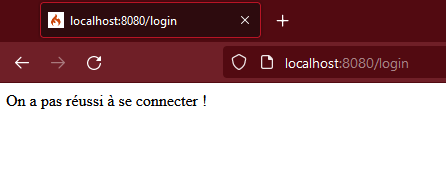
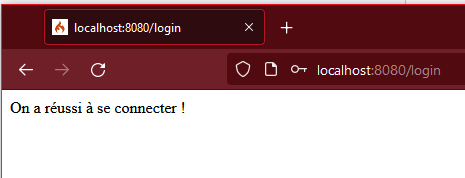
} else {

return "On a pas réussi à se connecter !";

}

}

Si vous retestez votre formulaire avec les valeurs indiquées dans les constantes définies, vous devez afficher le message de connexion OK, et l’autre message dans le cas contraire



💡 *Nous verrons ensemble comment organiser la gestion des formulaires de manière plus efficace dans les prochaines séances.*

Complétez votre controller pour qu’il vous redirige vers une nouvelle route “/home” à l’aide de la fonction [redirect()](https://www.codeigniter.com/user_guide/outgoing/response.html#redirect).

💡 *Nous verrons ensemble comment gérer l’authentification pour stocker la session utilisateur et mettre un cookie de connexion dans les prochaines séances*

# Utilisation de la base de données

Afin de pouvoir utiliser la base de données, Code Igniter fournit une méthode simple qui permet d’exécuter et de récupérer les résultats de la requête SQL fournie.

💡 *L’aide sur l’utilisation de la base de données est disponible ici :* [*Generating Query Results — CodeIgniter 4.4.5 documentation*](https://www.codeigniter.com/user_guide/database/results.html#getting-an-array-of-stdclass)

# Travail à faire

A l’aide des concepts présentés dans ce document, et en vous basant sur le sujet du TP3, travaillez en groupe afin de développer les premières pages du projet du TP3