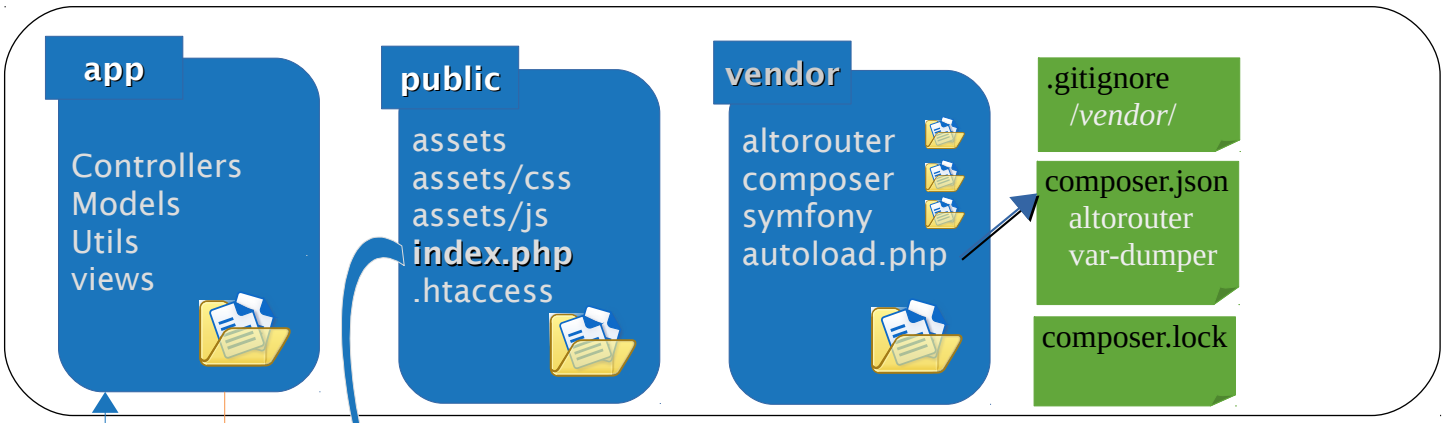
FrontController



app/
Application.php

FrontController

public/
Index.php

Point d'entrée

Le fichier *index.php* c'est notre point d'entrée, c'est ce fichier qui sera exécuté en premier. C'est lui qui va centraliser nos requêtes via **.htaccess**

Toutes les demande de fichier inconnus de nos routes seront redirigées vers le fichier index.php, sinon elles seront affichées.

```
require __DIR__.'/../app/Utils/...';
require __DIR__.'/../app/Models/...';
require __DIR__.'/../app/Controllers/...';

// Inclusion de mon FrontController
Require __DIR__ .'/../app/Application.php' ;
```

On a besoin d'inclure les dossiers du dossier *app*, car celui-ci comportera nos classes à utiliser dans le code.

On ajout à notre point d'entrée l'inclusion du fichier *Application.php* qui est notre **FrontController**, qui va démarrer notre site. C'est aussi une classe.

```
// Inclusion de Composer
require __DIR__.'/../vendor/autoload.php';

// Je créé une instance de mon application
// autremant dit de mon site web
$application = new Application() ;
```

Le fichier **autoload.php** permet de générer et d'utiliser nos dépendances, comme *altorouter* ou *var-dumper*, via le fichier *composer.json*

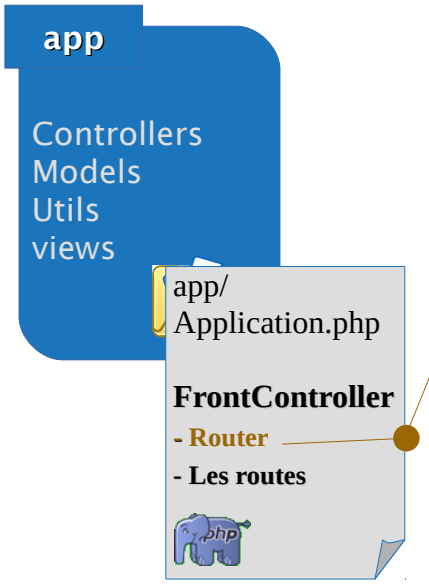
```
// Mise en place d'un FrontController séparé
de notre point d'entrée (index.php)

class Application
{

}
```

Pour pouvoir utiliser ma classe Application ou démarrer le site dans le **FrontController**, je dois instancier un objet de cette classe dans le fichier *index.php*
Cet objet sera ***$application***.

FrontController, Router



Le **Router** permet de définir un chemin comun aux différents fichiers appelés. le principal intérêt du routeur c'est de relier les différentes URL à différentes méthodes.

Dans la méthode **__construct**, on instancie un nouvel objet de la *classe AltoRouter*, on défini la partie commune au router et on la passe ensuite à la propriété *$router*, qui nous renvera un objet de la classe *AltoRouter* avec ses propriétés et ses méthodes. Cet objet est le routeur.

Faire un dump($this→router) de l'objet pour voir son contenu.

```php
// Mise en place d'un FrontController séparé de notre point d'entrée (index.php)
class Application
{

    // Mise en place d'un routeur
    // https://www.grafikart.fr/tutoriels/php/router-628
    // Je créé un attribut privé qui va contenir mon router
    private $router;

    // ma fonction __construct est une fonction dite magique
    // celle-ci est automatiquement appelée lorsque ma classe est instanciée
    // autrement dit lorsque l'on fait un new Application() dans inde.php
    public function __construct()
    {

        // On lance, on préconfigure AltoRouteur
        // J'instancie un nouvel objet à partir de AltoRouter
        $this->router = new AltoRouter();

        // Récupération de la BASE_URI en place via le .htaccess
        // la méthode trim() de PHP, permet d'être sur d'avoir une URI sans espace.
        // Si la clé BASE_URI dans le tableau $_SERVER est définie, alors supprimer les espaces sinon retourner à la racine
        $baseUrl = isset($_SERVER['BASE_URI']) ? trim($_SERVER['BASE_URI']) : '/';
        // faire un dump($_SERVER) permet de voir ce que l'on récupére dans notre super global qui est un tableau
            associatif
        // à la clé BASE_URI la valeur est :
        // « /Lunar_3-9-18/S06_API_31-10-18_12-11-18/E01-oFramework-filRougeSaison/ S06-E01_supp_oFramework-Rappel/public »
        // autrement dit le chemin jusqu'au point d'entrée 'index.php'. BASE_URI est la partie qui ne change pas dans notre URL.
        // URL = nom de domaine + BASE_URI + point d'entrée ou fichier ciblé.
        dump($_SERVER);
        // Définition de la base_URI à AltoRouter
        // Doc : http://altorouter.com/
        /**
         * C'est une méthode vraiment liée a AltoRouter
         * elle lui permet de séparer la partie fixe de la partie changeante dans l'url
         * donc dans le setBasePath on donne à AltoRouter la partie fixe
         * */
        // on passe la partie fixe(BASE_URI) à Altorouter. Ensuite cela nous permetra de définir la partie qui change, les routes.
        // on passe un argument à la méthode dont la valeur sera BASE_URI :
        // "/Lunar_3-9-18/S06_API_31-10-18_12-11-18/E01-oFramework-filRougeSaison/S06-E01_supp_oFramework-Rappel/public "
        // SetBasePath - Spécifie une URL de base pour que PHP interprète les URL relatives
        $this->router->setBasePath($baseUrl);
        dump($baseUrl);

        dump($this→router); // Aller page suivante pour voire le dump()

    }
}
```

Router

Le principe d'un Router en PHP, c'est de délocaliser la partie traitement des URL côté PHP .
`https://www.grafikart.fr/tutoriels/php/router-628`

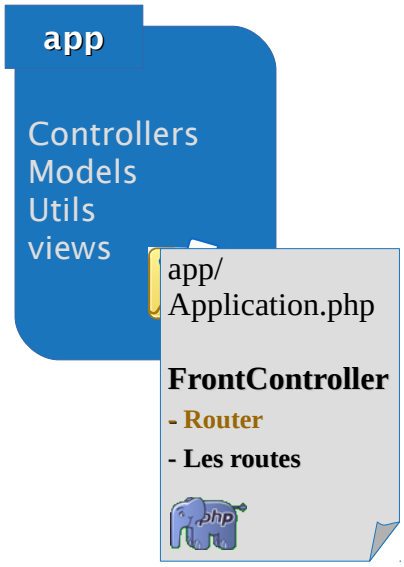
Ce que l'on souhaiterais c'est avoir des URL propres. Par exemple quand on consulte une page d'article, on peut avoir une URL de ce type :
`http://local.dev/Lab/Router/index.php?page=article&id=1&slug=salut-les-gens`

Pour améliorer la réécriture de ces URL, on va utiliser **.htaccess** ou autre, pour avoir une URL de ce type :
`http://local.dev/Lab/Router/article/salut-les-gens-3`

Pour éviter d'avoir à travailler sur ou dans **.htaccess**, car cela rend les modifications difficile. On va donc tout déléguer à PHP, c'est à dire que PHP recevra cette URL :
`http://local.dev/Lab/Router/article/salut-les-gens-3`

Et ? Il s'occupera de la parser(analyser, découper) pour savoir ce qu'il doit exécuter. Derrière il va chercher les codes, les paramètres et autres … - parser → https://fr.wikipedia.org/wiki/analyse_syntaxique

FrontController, Router #2



Dans la méthode **__construct**, on instancie un nouvel objet de la *classe AltoRouter*, on défini la partie commune au router et on la passe ensuite à la propriété *$router*, qui nous renvera un objet de la classe *AltoRouter* avec ses propriétés et ses méthodes. Cet objet est le routeur.

Faire un dump($this→router) de l'objet pour voir son contenu.

```php
// Mise en place d'un FrontController séparé de notre point d'entrée (index.php)
class Application
{

    // Mise en place d'un routeur
    // https://www.grafikart.fr/tutoriels/php/router-628
    // Je créé un attribut privé qui va contenir mon router
    private $router;

    // ma fonction __construct est une fonction dite magique
    // celle-ci est automatiquement appelée lorsque ma classe est instanciée
    // autrement dit lorsque l'on fait un new Application() dans inde.php
    public function __construct()
    {

        // On lance, on préconfigure AltoRouteur
        // J'instancie un nouvel objet à partir de AltoRouter
        $this->router = new AltoRouter();

        // Récupération de la BASE_URI en place via le .htaccess
        // la méthode trim() de PHP, permet d'être sur d'avoir une URI sans espace.
        // Si la clé BASE_URI dans le tableau $_SERVER est définie, alors supprimer les espaces sinon retourner à la racine
        $baseUrl = isset($_SERVER['BASE_URI']) ? trim($_SERVER['BASE_URI']) : '/';
        // faire un dump($_SERVER) permet de voir ce que l'on récupére dans notre super global qui est un tableau
            associatif
        // à la clé BASE_URI la valeur est :
        // « /Lunar_3-9-18/S06_API_31-10-18_12-11-18/E01-oFramework-filRougeSaison/ S06-E01_supp_oFramework-Rappel/public »
        // autrement dit le chemin jusqu'au point d'entrée 'index.php'. BASE_URI est la partie qui ne change pas dans notre URL.
        // URL = nom de domaine + BASE_URI + point d'entrée ou fichier ciblé.
        dump($_SERVER);
        // Définition de la base_URI à AltoRouter
        // Doc : http://altorouter.com/
        /**
         * C'est une méthode vraiment liée a AltoRouter
         * elle lui permet de séparer la partie fixe de la partie changeante dans l'url
         * donc dans le setBasePath on donne à AltoRouter la partie fixe
         * */
        // on passe la partie fixe(BASE_URI) à Altorouter. Ensuite cela nous permetra de définir la partie qui change, les routes.
        // on passe un argument à la méthode dont la valeur sera BASE_URI :
        // "/Lunar_3-9-18/S06_API_31-10-18_12-11-18/E01-oFramework-filRougeSaison/S06-E01_supp_oFramework-Rappel/public "
        // SetBasePath - Spécifie une URL de base pour que PHP interprète les URL relatives
        $this->router->setBasePath($baseUrl);
        dump($baseUrl);
        dump($this→router);
    }
}
```

```
AltoRouter {#2 ▼
  #routes: []
  #namedRoutes: []
  #basePath: "/Lunar_3-9-18/S06_API_31-10-18_12-11-18/E01-oFramework-filRougeSaison/S06-E01_supp_oFramework-Rappel/public"
  #matchTypes: array:6 [▼
    "i" => "[0-9]++"
    "a" => "[0-9A-Za-z]++"
    "h" => "[0-9A-Fa-f]++"
    "*" => ".+?"
    "**" => ".++"
    "" => "[^/\.]++"
  ]
}
```

https://drive.google.com/drive/folders/1lG9HUUh-RY0QKNtWBjc9cEsaZngLZUlY

FrontController, Router, Routes

app

Controllers
Models
Utils
views

app/
Application.php

FrontController
- Router
- Les routes

```php
// Mise en place d'un FrontController séparé de notre point d'entrée (index.php)
class Application
{

    // Mise en place d'un routeur
    // https://www.grafikart.fr/tutoriels/php/router-628
    // Je créé un attribut privé qui va contenir mon router
    private $router;

    // ma fonction __construct est une fonction dite magique
    // celle-ci est automatiquement appelée lorsque ma classe est instanciée
    // autrement dit lorsque l'on fait un new Application() dans inde.php
    public function __construct()
    {
        // On lance, on préconfigure AltoRouteur
        // J'instancie un nouvel objet à partir de AltoRouter
        $this->router = new AltoRouter();

        // Récupération de la BASE_URI en place via le .htaccess
        // la méthode trim() de PHP, permet d'être sur d'avoir une URI sans espace.
        // Si la clé BASE_URI dans le tableau $_SERVER est définie, alors supprimer les espaces sinon retourner à la racine
        $baseUrl = isset($_SERVER['BASE_URI']) ? trim($_SERVER['BASE_URI']) : '/';
        // faire un dump($_SERVER) permet de voir ce que l'on récupére dans notre super global qui est un tableau
            associatif
        // à la clé BASE_URI la valeur est :
        // « /Lunar_3-9-18/S06_API_31-10-18_12-11-18/E01-oFramework-filRougeSaison/ S06-E01_supp_oFramework-Rappel/public »
        // autrement dit le chemin jusqu'au point d'entrée 'index.php'. BASE_URI est la partie qui ne change pas dans notre URL.
        // URL = nom de domaine + BASE_URI + point d'entrée ou fichier ciblé.
        dump($_SERVER);
        // Définition de la base_URI à AltoRouter
        // Doc : http://altorouter.com/
        /**
         * C'est une méthode vraiment liée a AltoRouter
         * elle lui permet de séparer la partie fixe de la partie changeante dans l'url
         * donc dans le setBasePath on donne à AltoRouter la partie fixe
         * */
        // on passe la partie fixe(BASE_URI) à Altorouter. Ensuite cela nous permetra de définir la partie qui change, les routes.
        // on passe un argument à la méthode dont la valeur sera BASE_URI :
        // "/Lunar_3-9-18/S06_API_31-10-18_12-11-18/E01-oFramework-filRougeSaison/S06-E01_supp_oFramework-Rappel/public "
        // SetBasePath - Spécifie une URL de base pour que PHP interprète les URL relatives
        $this->router->setBasePath($baseUrl);

        // J'appelle ma méthode 'defineRoutes' qui se charge de remplir notre objet Altorouter avec nos routes
        $this→defineRoutes();
    }

    private function defineRoutes()
    {
        // toutes les routes seront placées dans une méthode réservée aux routes
        // map() -> méthode de AltoRouter -> http://altorouter.com/usage/mapping-routes.html
        // map('méthode', 'url', 'appel la méthode 'home' de MainController, 'nom de la route');
        // donc dès qu'on appellera l'URL '/ ' en GET, en gros dés qu'on affichera la home dans le navigateur, on appellera la
          méthode Home du MainController
        // Le 4eme argument est un nom qu'on donne à la route, il nous permettra par exemple de retrouver notre route ailleurs
          pour générer des lien en faisant $routeur->generate('nom_de_la_route')
        $this->router->map('GET', '/', 'MainController#home','main_home');
        // $this->router->map('GET', '/', 'MainController#home','main_home');
        // $this->router->map('GET', '/', 'MainController#home','main_home');
        // $this->router->map('GET', '/', 'MainController#home','main_home');

        dump($this→router);
    }
}
```

J'appelle la méthode *defineRoutes()* sur mon objet *AltoRouter* représenté par *$this*, c'est à dire que je vais charger les routes dans mon Router.

On défini les routes dans la méthode *defineRoutes()*.
On applique la méthode *map()* d'*AltoRouter* sur l'objet $router, en lui passant des arguments. Le 1er sera la méthode, le 2ième l'URL, le 3ième sera la méthode appelée du Controller et le 4ième sera le nom de la route.

En faisant un dump() de notre objet $router. On peux observer maintenant que nos propriétés *routes*, *nameRoutes* et *matchTypes* contiennent des tableaux de type associatifs avec des clés et des valeurs.

***dump($this→router)*, contient l'objet $router de la classe AltoRouter.**

On peut voire notre route definie dans la méthode *defineRoutes()*