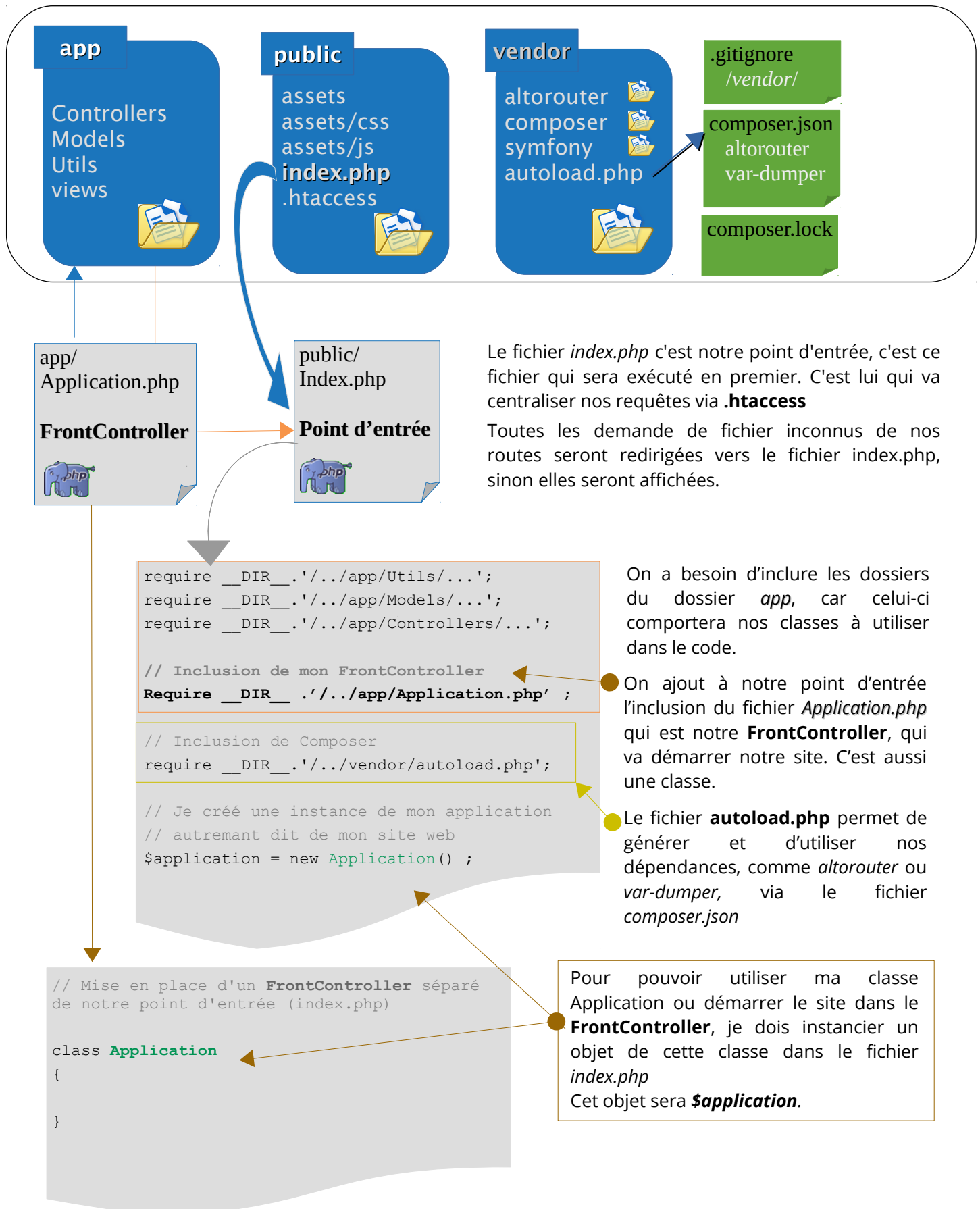
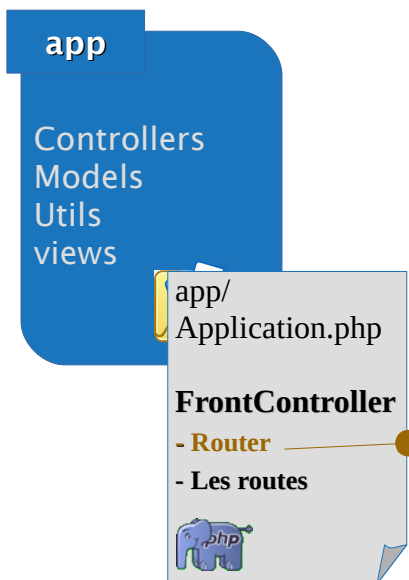


FrontController



FrontController, Router



Le **Router** permet de définir un chemin commun 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 définit la partie commune au router et on la passe ensuite à la propriété **\$router**, qui nous renverra 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.

```
// 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 crée un attribut privé qui va contenir mon routeur
    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 AltoRouter
        // 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 sûr 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($_SERVER) permet de voir ce que l'on récupère dans notre super global qui est un tableau associatif
        // à la clé BASE_URI la valeur est :
        // « /Lunar_3-9-18/S06_API_31-10-18_12-11-18/E01-oFramework-filRougeSaison/ S06-E01_supp_oFramework-Rappel/public »
        // autrement dit le chemin jusqu'au point d'entrée 'index.php'. BASE_URI est la partie qui ne change pas dans notre URL.
        // URL = nom de domaine + BASE_URI + point d'entrée ou fichier ciblé.
        dump($_SERVER);

        // Définition de la base_URI à AltoRouter
        // Doc : http://altorouter.com/
        /**
         * C'est une méthode vraiment liée à 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 permettra 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 voir 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 souhaiterait 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 :

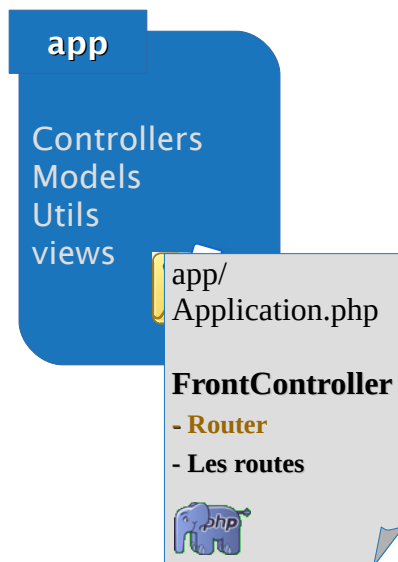
<http://local.dev/Lab/Router/article/salut-les-gens-3>

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 définit la partie commune au router et on la passe ensuite à la propriété `$router`, qui nous renverra un objet de la classe `AltoRouter` avec ses propriétés et ses méthodes. Cet objet est le router.

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

```
// 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 crée un attribut privé qui va contenir mon routeur
    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 index.php
    public function __construct()
    {

        // On lance AltoRouter
        // 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 sûr 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 récupère 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_suppl_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 à 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 permettra 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_suppl_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_suppl_oFramework-Rappel/public"
#matchTypes: array:6 [▼
    "i" => "[0-9]++"
    "a" => "[0-9A-Za-z]++"
    "h" => "[0-9A-Za-f]++"
    "*" => ".+?"
    "+" => ".++"
    "" => "[^/\\.]+?"
]
}
```