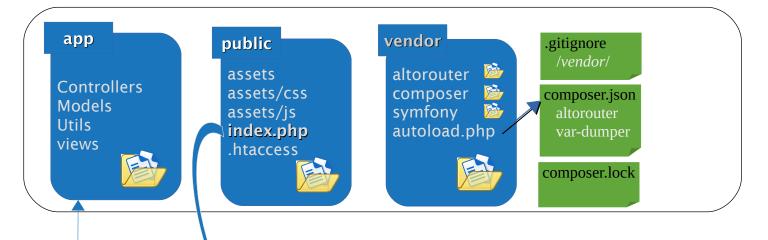
## Correspondance des routes : méthode run()



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** 

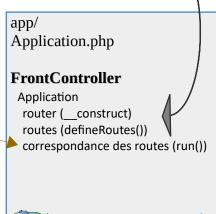
Toutes les demande de fichier inconnus de nos routes seront redirigées vers le fichier index.php, sinon elles seront affichées.

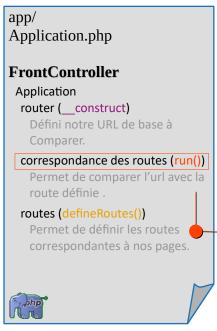
```
require DIR .'/../app/Utils/...';
require DIR .'/../app/Models/...';
require DIR .'/../app/Controllers/...';
// Inclusion de mon FrontController
Require __DIR__ .'/../app/Application.php' ;
// Inclusion de Composer
require DIR .'/../vendor/autoload.php';
// Je créé une instance de mon application
// autremant dit de mon site web
$application = new Application();
// On a initialisé ou préconfiguré le router grâce à la méthode
// on a appelé la méthode defineRoutes() dans le constructeur
// nous créons cette méthode , qui permet de créer nos routes
// on a maintenant nous routes, on va pouvoir rouler dessus.
// Je prend mon objet et lui applique une méthode
$application->run();
         Pour pouvoir suivre la bonne route et la
         parcourir, je vais appliquer une méthode
         (run()) sur mon objet $application.
         Ensuite je créais la méthode dans le
         FrontController.
```

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

Dans ma classe **Application** de mon **frontController**. Je préconfigure mon objet **\$router** de la classe **AltoRouter** dans la méthode **\_\_construct**.

Dans la méthode **defineRoute()** je défini la route utilisée par **AltoRouter**.





La méthode **match()** d'*AltoRouter* nous renvoi un tableau associatif avec 3 clés (*target*, *params*, *name*) correspondantes à des arguments passés dans la méthode *map()* de notre route définie dans la méthode defineRoutes().

On découpe la *clé 'target'* du tableau **\$match** en 2 entités avec la function *explode()* de php. Puis on réassigne chacune des valeurs dans **\$controllerName** et **\$methodeName** avec la fonction *list()* de php.

http://php.net/manual/fr/function.list.php http://php.net/manual/fr/function.explode.php

J'initialise un *variable* **\$params** qui pourra stocker à la *clé* '*params*' du tableau **\$match** une valeur.

```
array:3 [▼
    "target" => "MainController#home"
    "params" => []
    "name" => "main_home"
]

"MainController"

"home"
[]
```

Dans la *méthode run()* on a fait travailler *AltoRouter*, qui nous a permis de conclure, le *nom du controlleur... \$controllerName* à exécuter, le nom de la méthode... *\$methodeName* à exécuter et les paramètres éventuels à fournir (*\$params*).

```
// Mise en place d'un FrontController séparé de notre point d'entrée (index.php)
class Application
   private $router;
   public function _construct()
    $this->router = new AltoRouter();
    $baseUrl = isset($_SERVER['BASE_URI']) ? trim($_SERVER['BASE_URI']) : '/';
    $this->router->setBasePath($baseUrl);
    // J'appelle ma méthode 'defineRoutes' qui se charge de remplir notre objet Altorouter avec nos routes
    $this→defineRoutes();
   public function run()
       // je demande à AltoRouter d'essayer de "matcher" l'url actuelle ..
       // match() est une métode d'AltoRouter
       // On va faire correspondre l'URL qui est appelée, celle indiquée dans la barre d'URL du navigateur, donc une page du site.
       // Celle-si sera redirigée vers index.php, et comparée avec les routes déclarées et donc existantes dans la méthode
defineRoute()
       $match = $this->router->match();
       // la variable $match peut valoir :
       // false si aucune route ne correspond avec l'url actuelle
       // un tableau si une correspondance (= un "match") à été trouvée
       // Si on a une correspondance
      if ($match !== false) {
       // OU // if (is_array($match)){}
          // explode permet de découper une chaîne de caractères en fonction d'un délimiteur ici '#'
         // 'target' correspond à 'MainController#home
         // list() permet de regrouper un tableau dans des variables
         // Je viens donc découper ma variable $match['target'] suivant le #
         // et j'assigne les valeurs dans les 2 variables "$controllerName" et "$methodeName"
        list($controllerName, $methodeName) = explode('#',
         $match['target']);
            * Reviens à faire :
           * $target = explode('#', $match['target']);
            * $ControllerName = $target[0];
            * $methodeName = $target[1];
          // J'isole mes paramètres
         $params = $match['params'];
        // Si on a pas de correspondance ... ou la page 404
  	riangle } else {
          $controllerName = 'ErrorCOntroller';
          $methode = 'Error404';
          $params = array();
       }
        dump($match);
       dump($controllerName); // MainController
        dump($methodeName); // home
        dump($params);
                                        //[]
   private function defineRoutes()
     $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');
1
```