**WWCMS**

**Il “giro”**

1. La **index:**   
   1.1 carica il file autoload.php dalla cartella “vendor”:
2. **Autoload**, a sua volta:   
   a) carica il file **autoload\_real.php** dalla sottocartella “**composer**”  
   b) restituisce la funzione **getLoader()** della classe descritta dentro a autoload\_real
   1. crea un’istanza di **app**, definita dentro a lib
   2. lancia la funzione **run();**
3. **Classe App**

La classe App ha un solo metodo, “**run()**” che:

1. istanzia l’oggetto “**Route**”, prendendolo dalla cartella **Core**
2. con route lancia il metodo **mapFromRequest()**
3. **mapFromRequest()** utilizza dapprima una funzione di Symphony (**Request::createFromGlobals()**) che crea un oggetto (**$request**) contenente varie informazioni:  
   a) quelle che poi confluiscono nell’oggetto **$action**b) quelle che poi confluiscono nell’oggetto **$controller** (Compreso il **component**)  
   L’url, in sintesi, viene spezzato in 3 parti: **$component = array\_shift($array);   
   $controller = array\_shift($array);   
   $action = array\_shift($array);**Se il controller è **vuoto** viene definito come “**index**”  
     
   Se action è vuota viene definita come “**indexAction**”
4. istanzia l’oggetto “**Response”** della classe **Response**
5. istanzia l’oggetto “http://0.0.0.0:8080
6. “ della classe **ModuleFactory**”
7. lancia il metodo “**build()**” della classe “**ModuleFactory**” che utilizza come argomenti gli oggetti **Route e Response:**
   1. **Il metodo build** crea l’oggetto **$objectController**, utilizzando la Route, e poi lancia il metodo **setResponse**, utilizzando Response.  
      In altre parole build carica il **controller**, lancia il **metodo** e aggiunge l’output nella **Response.**

|  |
| --- |
| **ROUTE**: wwcms\Core\Route::\_\_set\_state(array( 'controller' => '\\Components\\Index\\Index', 'action' => 'indexAction', )) **RESPONSE**: wwcms\Core\Response::\_\_set\_state(array( 'container' => array ( ), 'type' => 'html', 'template' => 'default', 'view' => 'index.html', )) |

1. **Classe ModuleLogic**

La classe ModuleLogic definisce una serie di metodi atti a gestire i moduli:

-



Componenti = roba nostra di sistema (lib)

Moduli = di volta in volta creati per il progetto (app)

Request e Response: altri oggetti di S.

La url definisce