



Introduction à Composer et Symfony

composer, autoload, routage

Dans ce TD, vous allez vous familiariser avec les composants du standard des Frameworks PHP : Symfony en commençant par quelques composants de base.

Une introduction générale à la version actuelle de symfony se trouve ici :
<http://symfony.com/doc/current/index.html>

Exercice 1. composer, class-loader et http-foundation

Installons d'abord composer s'il n'est pas déjà présent sur notre OS. On va d'abord créer un répertoire `bin` à la racine de notre HOME :

```
cd
mkdir bin
cd bin
curl -s https://getcomposer.org/installer | php
```

Vérifiez la définition de vos variables d'environnement `http_proxy` et `https_proxy` dans votre `.bashrc`. Ajoutez également la ligne suivante à votre `.bashrc` :

```
export PATH=$PATH:~/bin
```

de manière à ce que tous les programmes installés dans le répertoire `bin` de votre HOME soient accessibles de n'importe où. Puis créez un répertoire de travail `sf-test` dans votre dossier de travail dans lequel vous allez lancer la commande suivante :

```
composer require twig/twig symfony/http-foundation
```

Ceci indique que nous n'installons pour l'instant que ces 2 composants. Observez le fichier `composer.json` qui a été créé dans le répertoire courant, ainsi que le répertoire `vendor` qui contient les composants installés. On peut aussi installer les composants en les spécifiant dans le fichier `composer.json` :

```
1  {
2    "require": {
3      "twig/twig": "^3.5",
4      "symfony/http-foundation": "^6.2"
5    }
6  }
```

Puis lancer la commande :

Compléments Web BUT 2 : PHP (TD n°4) 2022–2023

```
composer update
```

L'utilisation éventuelle de l'option `-o` de `composer` permet d'optimiser « au mieux » le chargement automatique des classes.

L'autoloader et les 2 principaux composants de `HttpFoundation` à savoir `Request` et `Response` sont prêts à l'emploi. Remarquez l'usage des espaces de nommages en PHP semblables à ceux du C++ ou aux `import` de packages en java.

```
1 <?php
2 require_once __DIR__.'vendor/autoload.php';
3
4 use Symfony\Component\HttpFoundation\Request;
5 use Symfony\Component\HttpFoundation\Response;
6
7 // $request = Request::createFromGlobals();
8
9 $request = Request::create('/essai.php?name=Zozo');
10
11 // URI demandee (sans les parametres)
12 $path=$request->getPathInfo();
13
14 // recup GET de variables
15 $nom=$request->query->get('name','World');
16 $prenom=$request->query->get('surname','Joe');
17 echo "Bonjour $surname $name<br/>";
```

On peut aussi récupérer d'autres informations sur le Client et fabriquer une réponse :

```
1 <?php
2 // recup variables SERVER
3 $host=$request->server->get('HTTP_HOST');
4
5 // recup COOKIES
6 $request->cookies->get('PHPSESSID');
7
8 // HTTP headers
9 $headers=$request->headers->get('host');
10 $content_type=$request->headers->get('content_type');
11
12 $method=$request->getMethod(); //GET, POST, PUT, DELETE ou HEAD
13 $langs=$request->getLanguages();
14 $IP == $request->getClientIp();
15 $response =
16 new Response($IP." ".$host." ".$path." ".$headers."
17 ".$content_type." ".$method." ".$nom." ".$langs[0]);
18 $response->send();
```

Exercice 2. installer twig

Compléments Web BUT 2 : PHP (TD n°4) 2022–2023

Pour installer twig, on fait simplement :

```
composer require twig
```

Voir TD5 pour son utilisation.

Exercice 3. créer une application SF

Pour créer un squelette d'application avec un nombre réduit de dépendances :

```
composer create-project symfony/skeleton hello-sf
```

Pour une application plus complète (ne le faites pas pour le moment)

```
composer create-project symfony/website-skeleton sf-full-project
```