Créer une API pour notre blog

La Et si nous voulions créer une application mobile liée à notre application ?

Ou laisser d'autres sites web utiliser nos incroyables articles ?

Pour cela, il va falloir créer une API pour notre site : rien de plus simple grâce au génial API Platform..

Vous avez dit API ?

Une API (alias Application Programming Interface) est, de manière un peu vulgarisée, une interface permettant aux développeurs d'utiliser des services mis en place par d'autres développeurs. C'est donc une sorte de portail permettant d'utiliser du code, déjà créé.

Dans notre cas, si nous voulons développer une application mobile qui a accès aux mêmes données que notre site web, il est indispensable de créer une API afin de pouvoir accéder depuis l'application web aux mêmes données et services que depuis notre site, tout en évitant de dupliquer le code « métier ».

Une API dans le web permet en effet d'exposer / d'échanger des informations via des formats (très souvent JSON ou XML). Ainsi, en développant une API pour notre blog nous pourrions exposer les articles créés pour qu'une potentielle application mobile (ou objet connecté, ou autre) les récupère et les utilise.

A noter qu'il existe différents standards d'API Web, mais aujourd'hui un des standard les plus utilisés est REST.

Pour exposer des données comme par exemple nos articles (on parle de « ressources » avec le standard REST), on créé des URLS (URI) permettant globalement de faire du CRUD c'est à dire d'afficher / modifier / créer ou supprimer les ressources (GET, POST, DELETE, PUT/PATCH).

Cette création peut être un peu fastidieuse et redondante, surtout si les ressources sont nombreuses. C'est pour cela qu'un framework comme API Platform a été créé.

Mettre en place une API avec API Platform

API Platform peut s'utiliser de plusieurs manières : en tant que composant server, c'est à dire avec un projet existant pour développer une API sur ce projet, ou de manière autonome. En effet API Platform Distribution est un ensemble de composants comprenant un espace d'admin créé en JS (React), un squelette d'API pour créer nos ressources à exposer et même un générateur de clients JS (React, Vue, Angular…) se connectant directement aux points d'API créés.

Dans notre cas, nous allons uniquement utiliser le composant serveur.

Pour l'installer :

composer require api

Une fois API Platform installé, il suffit d'aller sur l'url /api pour voir la documentation, c'est à dire les ressources exposées etc, générée automatiquement par l'outil Swagger !

Exposer une ressource

Pour créer les différentes URLS (URI) permettant d'afficher / créer / modifier / supprimer une ressource, rien de plus simple avec API Platform : il suffit de rajouter l'annotation @ApiResource au dessus de la classe de l'entité à exposer.

Pour nos articles par exemple :

```
…
use ApiPlatform\Core\Annotation\ApiResource;

/**
* @ORM\Entity()
* @ApiResource
*/

class Article
{
        /** * @ORM\Id
        * @ORM\GeneratedValue
```

```
     * @ORM\Column(type="integer")
     */
     private $id;
…
```

Et le tour est joué !

En rechargeant la page de documentation, les différentes urls (IRI) crées sont désormais listées et documentée. Votre API est mise en place !

Il est possible de voir les routes générées en utilisant :

```
bin/console debug:router
```

L'API est testable directement via la documentation, ou en utilisant un client tel que Postman.

Modifier une opération

Bien que très facile à mettre en place, l'API générée est tout à fait modifiable.
Il existe deux types d'opérations : les opération d'item (agissant sur une seule ressource) et les opérations de collection (agissant sur plusieurs ressources).

Pour n'activer que les méthodes GET sur les items et les collections :

```
…
* @ApiResource(
*         collectionOperations={"get"},
*         itemOperations={"get"}
* )
…
```

Il est également possible de modifier l'URL (IRI) utilisée pour la ressource, les paramètres etc :

```
…
```

```
* @ApiResource(
*          collectionOperations={"get"},
*          itemOperations={
*                  "get"= {
*                          "path"="blog/article/{id}"

*                  }
*         }
* )
…
```