*Testez fonctionnellement votre application Symfony*

**Qu’est-ce qu’un test fonctionnel ?**

* Ecrire un test fonctionnel c’est établir un scénario utilisateur.
* Les tests fonctionnels servent à tester toutes les fonctionnalités de notre application de manière automatique.
* Un test fonctionnel est dit « boite noire » car il n’est pas nécessaire de connaitre le code pour l’effectuer.
* A la différence des tests unitaires les tests fonctionnels consistent à requêter une page de notre application et à en récupérer le résultat.
* Un test fonctionnel se décompose en quatre étapes :

1. Création d’un client HTTP (qui jouera le rôle du navigateur de l’utilisateur)
2. Envois d’une requête HTTP sur la page à tester
3. Récupérer la réponse HTTP
4. S’assurer que les éléments de la page testée sont bien tous présents (par l’écriture d’assertions)

On regroupe tout ça en l’écriture d’un « scenario d’utilisation ».

**Premiers tests fonctionnels**

* Pour savoir quoi tester il faut établir la liste des URLs de l’application. On peut également établir le code coverage de notre application et ne se concentrer que sur les méthodes des contrôleurs.
* On peut lister les route de notre application grâce à la commande : **« console debug:router »** à partir de la racine de notre projet. Les routes qui nous intéressent sont celles qui ne commencent pas par « \_ ».
* Pour avoir plus de détail sur une route (contrôleur et méthode associés) il suffit de la passer en argument : **« console debug:router routeName »**
* Le nom de la classe de test doit correspondre au nom du contrôleur testé suffixé par « Test » et l’arborescence du projet doit être respectée.

La classe WebTestCase simule le test d’un humain par un navigateur. C’est une instance de la classe FrameworkTest ce qui nous permet de faire des assertions : « **Symfony\Bundle\FrameworkBundle\Test\WebTestCase »**

* Pour créer un client : **« $client = static ::createClient() ; »**
* Pour envoyer une requête : **« $client->request(‘GET’, ‘/’) ; »**

**Interagissez avec votre application lors de tests**

Avec les tests fonctionnels il est possible de vérifier ce que renvoie la requête HTTP (code status, en-tête, etc…) mais également ce qui est présent sur la page. Pour cela on fait appelle au crawler.

Le crawler est un objet auquel on peut avoir accès après chaque requête et permet d’accéder à n’importe quel élément du DOM.

Le crawler nous permet également d’effectuer des tests sur les formulaires. Pour se faire il suffit de :

1. Récupérer le formulaire depuis le crawler : **« $form = $crawler->selectButton(‘Ajouter’)->form(); »**
2. Remplir les différents champs associés (sélectionnés par leur name) : **« $form[‘fieldName’] = value; »**
3. Envoyer le formulaire : **« $crawler = $this->client->submit($form); »**
4. Effectuer notre assertion en fonction des vérifications voulues.

Le crawler peut aussi servir à simuler des cliques sur des liens. Pour se faire il suffit de :

1. Récupérer le lien depuis le crawler : **«$link = $crawler->selectLink(param)->link(); »**

Le paramètre peut être l’id, une classe, ou le contenu textuel du lien cible.

1. Cliquer sur le lien cible : **« $crawler = $this->client->click($link); »**
2. Effectuer notre assertion en fonction des vérifications voulues.