
Documentation: dashboard professeur et développement dirigé par les tests

Version 0.1

Bryan Oberson

22 March 2015

1	Introduction	1
2	Explications des termes spécifiques	3
2.1	Framework	3
2.2	Classes et héritages	3
2.3	Diagramme UML	3
2.4	Wireframe	4
3	Fonctionnalités du dashboard	5
3.1	Ajouter un groupe	5
3.2	Gérer un groupe	6
3.3	Voir ses exercices	8
3.4	Changer de mot de passe	8
4	Documentation	11
4.1	Différentes technologies	11
4.2	Application au projet	12
4.3	Wireframes, ou “scénarios du site”	12
4.4	Modèles et diagrammes UML	12
4.5	Implémentation avec le reste du projet	13
5	Développement dirigé par les tests	15
5.1	Le test fonctionnel	15
5.2	Le test unitaire	15
5.3	Le cycle du développement dirigé par les tests	16
5.4	Gain de temps ?	16
6	Débuter un projet	17
6.1	Éléments requis	17
6.2	Premier test	17

Introduction

Dans ce travail, je vais tout d'abord documenter mon travail pratique. Ce travail consiste en la conception d'un dashboard pour les professeurs utilisant le framework Django tout en essayant de programmer avec le développement dirigé par les tests. Ce dashboard fera partie d'un projet plus grand : un site d'e-learning pour les mathématiques. Tous les termes spécifiques tels que framework seront expliqués par la suite.

Cette documentation parlera tout d'abord de l'intérêt de ce travail pratique. Il y aura aussi le wireframe du dashboard et un diagramme UML expliquant mes différents modèles Django. J'expliquerai par la suite comment j'ai utilisé les différentes technologies pour créer mon application et comment je pourrai l'intégrer au projet final (le site) en essayant de souligner les difficultés que je pourrai rencontrer pendant l'intégration.

Par la suite, j'essaierai de résumer les connaissances de bases relatives au développement dirigé par les tests. Ceci consistera à expliquer les différents termes et à surtout à expliquer comment cela fonctionne et quels sont les avantages.

Finalement, j'expliquerai comment créer une ébauche de dashboard (créer un dashboard complet étant trop long et trop compliqué à tester de par le nombre important de boutons et de fonctionnalités) en utilisant ce fameux développement dirigé par les tests

Explications des termes spécifiques

Pour être sûr que tout le monde comprenne de quoi nous allons parler, il va nous falloir expliquer le lexique spécifique à nos deux sujets : dashboard professeur et développement dirigé par les tests.

2.1 Framework

Un **framework**, ou **structure logicielle**, sert de base à un programme. En effet, c'est de par ses structures de base que l'on peut le plus facilement programmer. C'est donc une grande aide à tout informaticien. Les framework les plus connus seraient, par exemple, Django, Bootstrap ou Ruby On Rails.

Il est important de noter que les frameworks sont souvent spécialisés dans un langage informatique très précis. Par exemple, Django utilise Python, Bootstrap utilise HTML et CSS et Ruby on Rails utilise Ruby.

Un framework est en outre très utile pour la programmation orientée objet. Ce type de programmation permet d'établir des liens entre les différents objets. Le framework aide à créer des classes et surtout des héritages.

2.2 Classes et héritages

Les **classes** correspondent à un moule dans lequel on met différentes informations pour créer un objet. Par exemple, pour une voiture, on pourrait définir sa couleur, sa marque ou encore son âge. Tout ce qu'il reste à faire est de donner des valeurs à ces trois propriétés pour "créer" la voiture.

Le concept d'héritage n'est pas beaucoup plus compliqué. En effet, un héritage consiste à créer une classe en prenant comme base les caractéristiques d'une autre, ce qui nous permet d'en rajouter d'autre. Dans un cas qui nous intéresserait plus, un élève possède comme spécificités un nom, un prénom et une école. On peut grâce à ça créer une classe Professeur, qui aura les mêmes caractéristiques mais à laquelle on pourrait ajouter la branche qu'il enseigne.

2.3 Diagramme UML

Quand l'on possède de nombreuses classes, il devient indispensable de pouvoir voir rapidement quelles relations ces classes entreprennent entre elles. Pour ceci, le meilleur moyen de le faire est de créer un **diagramme UML**. Un diagramme schématise les objets et indique leurs différentes relations (relation simple avec une clé étrangère ou aussi plusieurs-à-plusieurs (ManyToMany)).

Il existe de nombreux programmes permettant de réaliser des diagrammes UML clairs et présentables, comme boUML, argoUML ou Poseidon.

2.4 Wireframe

Un **wireframe** est un schéma consistant à mettre en valeur les relations entre les différentes pages d'un site web. Cela explique par exemple ce qu'il se passe si l'on clique sur tel ou tel bouton. Cela consiste donc à pouvoir voir ou exactement l'on peut aller en manipulant les fonctionnalités du site.

Fonctionnalités du dashboard

3.1 Ajouter un groupe

La fonctionnalité de base de ce dashboard est la création de groupe. En créant un groupe, le professeur sera par la suite capable de le gérer en gérant les membres qui s’y trouvent mais aussi en y assignant des devoirs.

Pour créer un groupe, il suffit de se rendre sur Nouveau groupe, tout en bas dans le menu de gauche. Cette action fera apparaître le formulaire de création de groupe.

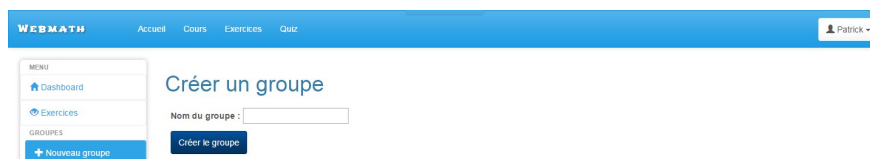


FIGURE 3.1 – Formulaire de création de groupe

La seule exigence présente lors de la création d’un groupe est le nom. Une fois le groupe créé, l’utilisateur actuel est automatiquement défini en tant que professeur pour le groupe.

Le groupe précédemment créé sera désormais affiché en permanence dans le menu de gauche du professeur, ce qui lui permet d’accéder à ses informations.

3.2 Gérer un groupe

3.2.1 Gérer les membres du groupe

Depuis cette page, le professeur peut gérer les membres qui sont actuellement enregistrés dans le groupe.

Il peut tout d'abord rajouter les élèves ou professeurs qu'il souhaite en entrant leur nom d'utilisateur dans le champ à disposition.

FIGURE 3.2 – Page d'administration d'un groupe

Si le nom d'utilisateur rentré correspond bien à un étudiant ou à un professeur, cet utilisateur sera rajouté dans la liste des membres.

FIGURE 3.3 – Ce à quoi ressemble la page une fois que des membres ont été rajoutés

Au contraire, si aucun utilisateur n'a été trouvé ou si l'utilisateur ne correspond pas au rôle qu'il lui est donné (par

exemple si c'est un professeur et qu'il a été ajouté aux étudiants), le site renverra un message d'erreur.

The screenshot shows the WEBMATH dashboard with a blue header containing 'Accueil', 'Cours', 'Exercices', and 'Quiz'. A user profile 'Patrick' is in the top right. On the left, a sidebar menu includes 'Dashboard', 'Exercices', '1EC03', and 'Nouveau groupe'. The main content area has a red button 'Supprimer la classe'. Below it, the 'Elèves' section shows a form to add a student with the name 'Jean-Paul'. A message states: 'Impossible d'ajouter cet élève. Veuillez vérifier que le nom d'utilisateur entré correspond bien à un élève.' Below this is a table of students:

Nom	Actions
Guillaume	<button>Retirer</button>

The 'Professeurs' section below shows a form to add a professor and a table of professors:

Nom	Actions
Patrick	<button>Retirer</button>
Alex	<button>Retirer</button>

The 'Derniers devoirs' section at the bottom has a form to assign a task, including a dropdown for 'Type de devoir' with options 'Exercice', 'Quiz', and 'Cours'.

FIGURE 3.4 – Message d'erreur retourné si l'utilisateur n'est pas valable

Une fois ajouté, un membre peut facilement être retiré du groupe grâce au bouton Retirer qui se trouve à côté de son nom.

3.2.2 Gérer un devoir

Un professeur peut bien évidemment donner des devoirs à son groupe.

Un devoir peut-être un exercice, un quiz ou un cours, tous les trois pouvant avoir été créé par un autre utilisateur.

Pour assigner un devoir, il suffit de savoir l'id de l'exercice, quiz ou cours, et de préciser grâce au menu à choix de quel type de devoir il s'agit.

The screenshot shows the 'Derniers devoirs' section. It includes a form to assign a task with fields for 'Id de l'exercice' (value: 5), 'Type de devoir' (dropdown with 'Exercice', 'Quiz', 'Cours'), and an 'Assigner le dev' button. Below is a table of tasks:

Nom	Type	Actions
Factorisation	Exercice	<button>Retirer</button>
Courte équation	Quiz	<button>Retirer</button>
Nombre de Viète	Cours	<button>Retirer</button>

FIGURE 3.5 – Différents champs à compléter pour assigner un devoir

Comme pour les fonctionnalités précédentes, si aucun exercice, quiz ou cours n'a pu être associé à l'id entrée, un message d'erreur sera renvoyé.

Un devoir peut être à tout moment retiré grâce au bouton Retirer à sa droite.

3.3 Voir ses exercices

Dans le menu de gauche, il y a un bouton nommé Exercices. C'est depuis cette page que le professeur pourra voir ses exercices, ses quiz et ses cours.

WEBMATH			
Accueil Cours Exercices Quiz			
Patrick			
MENU			
Dashboard			
Exercices			
GROUPES			
TECO3			
Nouveau groupe			

Liste des exercices			
Créer un exercice			
Exercice	Ajouté	Id	Actions
Factorisation	20 mars 2015 20:40:14	4	Supprimer
Equation du deuxième degré	20 mars 2015 20:40:31	5	Supprimer

Liste des quiz			
Créer un chapitre			
Quiz	Ajouté	Id	Actions
Courte équation	20 mars 2015 20:41:13	4	Supprimer

Liste des cours			
Créer un chapitre			
Cours	Ajouté	Id	Actions
Théorie sur le nombre d'or	20 mars 2015 20:41:43	4	Supprimer
Nombre de Viète	20 mars 2015 20:42:11	5	Supprimer

FIGURE 3.6 – Ce à quoi ressemble la page Exercices

Pour chaque activité que le professeur aura créé, il pourra voir le titre qu'il lui a donné, la date à laquelle il l'a créé et l'id qui lui sera utile s'il veut l'assigner en tant que devoir à un de ses groupes.

Il peut bien évidemment supprimer une activité en utilisant le bouton Supprimer se trouvant dans la dernière colonne du tableau.

Si le professeur souhaite créer une nouvelle activité, il n'a qu'à utiliser le bouton Créer en haut du tableau qui le redirigera directement au formulaire de création.

3.4 Changer de mot de passe

Peu importe sur quelle page il se trouve, le professeur peut accéder à un menu déroulant en haut à droite de cette page.

Dashboard amène le professeur sur l'accueil de son dashboard, Déconnexion le déconnecte et Profil l'amène sur un formulaire de changement de mot de passe.

Pour le modifier, le professeur n'a qu'à remplir les deux champs et à valider. Si tout a été rentré correctement, le mot de passe sera correctement modifié.

Au contraire, s'il y a une erreur, un message d'erreur sera retourné.

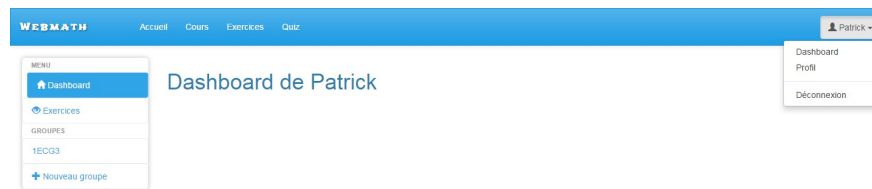


FIGURE 3.7 – Apparence du menu déroulant

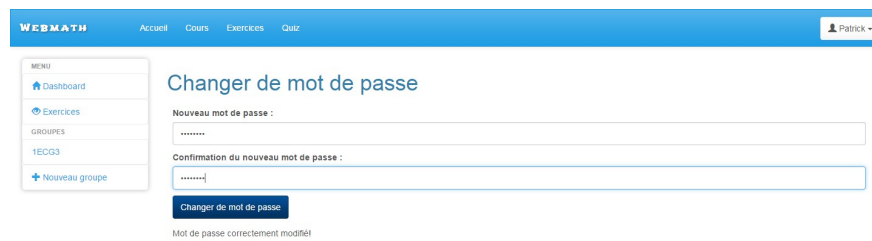


FIGURE 3.8 – Message pour confirmer que le changement de mot de passe a correctement eu lieu

WEBMATH Accueil Cours Exercices Quiz PAIDIK

MENU

- Dashboard
- Exercices
- GROUPES
- TEC33
- Nouveau groupe

Changer de mot de passe

Nouveau mot de passe :

Confirmation du nouveau mot de passe :

[Changer de mot de passe](#)

Mot de passe refusé. Vérifiez que les deux mots de passe entrés sont identiques ou que le mot de passe actuel correspond bien à votre mot de passe.

FIGURE 3.9 – Message d’erreur retourné si les champs n’ont pas correctement été remplis

4.1 Différentes technologies

Pour créer mon application (dashboard professeur), j'ai fait recourt aux technologies suivantes.

4.1.1 HTML5 (Hypertext Markup Language)

HTML est la base de toute page web. En effet, c'est ce qui nous permet d'afficher du texte, des images. Il est principalement utilisé avec CSS (pour la mise en page) et JavaScript (pour l'interactivité des pages).

J'ai utilisé HTML5 pour faire le frontal (plus connu sous le nom front-end). Le frontal correspond à ce que l'utilisateur voit. Il est opposé au back-end, qui est la partie qui travaille derrière mais que l'utilisateur ne voit pas.

4.1.2 CSS3 (Cascading Style Sheets)

CSS3 a été utilisé pour mettre en page mon code HTML. Cela a donc aussi contribué à la partie front-end de mon application. C'est CSS qui permet de modifier la police du texte, modifier la taille des images, ou encore de modifier la couleur du font. CSS permet donc de rendre une page potable pour l'oeil.

4.1.3 Bootstrap

Bootstrap est un framework qui contient du code HTML et CSS. J'ai utilisé Bootstrap comme base pour mon dashboard en utilisant le modèle *Charisma*¹. L'avantage de Bootstrap est qu'il offre des design adaptatifs, qui signifie que la mise en page va s'adapter à la taille de l'écran.

Voici un exemple d'adaptation :

Voici à quoi ressemble le dashboard sur un écran d'ordinateur.

Et voici ce que la page devient si l'on zoome ou si l'on réduit la taille de la fenêtre.

4.1.4 Django

Django est un framework spécialisé dans Python. C'est un des framework les plus utilisés. Nous avons choisi Django pour le projet car, travaillant sous Python, il est le plus documenté et le plus connu.

Grâce à Django, nous avons pu gérer des classes et des héritages, des vues et des urls.

1. <http://usman.it/free-responsive-admin-template/>

4.1.5 Git et Github

J'ai utilisé Git et Github pour garder une trace de l'avancement de mon travail et pouvoir rattraper toute erreur qui, dans le passé, m'aurait échappé et referait surface plus tard.

Grâce à Git, j'ai donc pu créer des commit, qui sont des entrées permettant au développeur de faire des sauvegardes et de voir à quel moment il a modifié certains éléments selon la légende qu'il a mit au commit. J'ai aussi pu aller chercher et remettre des informations sur le référentiel sur le site Github. En effet, il garde les copies des fichiers, ce qui permet de pouvoir les récupérer depuis n'importe quelle machine.

4.1.6 Cloud9

Enfin, j'ai utilisé Cloud9, un site qui permet de programmer sur un serveur se trouvant autre part, et de pouvoir y accéder depuis n'importe quelle machine.

La raison pour laquelle j'ai finalement favorisé la programmation sur Cloud9 plutôt que la programmation locale est le fait que, étant donné que je profite de programmer quand j'ai du temps et que je n'ai pas forcément une de mes machines, je peux le faire depuis l'ordinateur de quelque d'autre sans devoir télécharger multiples fichiers.

4.2 Application au projet

Dans le projet final, qui, rappelons-le, consistera en un site d'e-learning pour les mathématiques, ma partie pratique sera le dashboard professeur, donc l'endroit où les professeurs pourront gérer leurs exercices, leurs classes et leurs élèves. La raison pour laquelle cette partie du site est si importante est que sans un dashboard parfaitement opérationnel, il deviendrait chaotique pour les professeurs de gérer leurs différentes responsabilités.

Toutes les fonctionnalités ayant déjà été expliquées auparavant, je ne vais pas repasser dessus.

4.3 Wireframes, ou “scénarios du site”

4.4 Modèles et diagrammes UML

4.4.1 Modèles utilisés

Il y a tout d'abord notre classe Teacher, qui constitue bien évidemment le point central de notre application. La classe Teacher possède les propriétés suivantes : prénom, nom, adresse e-mail et l'école dans laquelle il enseigne. Il n'y a pas réellement d'autres propriétés à lui rajouter.

Pour qu'il y ait des professeurs, il faut aussi des élèves, c'est pourquoi j'ai une classe dénommée Student. Les propriétés ressemblent beaucoup à celles de Teacher étant donné que les deux classes correspondent à des classes d'utilisateur. Ces propriétés sont : prénom, nom, adresse e-mail, école et compétences de l'élève, établies par rapport à certains thèmes accomplis.

Finalement, ces deux classes sont réunies dans une classe nommée Group, qui pourrait s'apparenter à une classe d'école. En effet, la classe Group possède 1 à plusieurs professeurs, 1 à plusieurs élèves, un nom, des devoirs, des taux de réussites par rapport aux exercices réalisés en général, et une date de création.

4.4.2 Diagramme UML

Ce diagramme UML explique donc les relations entre les différentes pages de mon application, comment y accéder, que font les boutons, etc.

4.5 Implémentation avec le reste du projet

Comme déjà expliqué, ma partie, le dashboard professeur, servira de lieu de référence pour le professeur qui utiliserait le site. En effet, c'est seulement depuis cette partie qu'il pourra gérer tout ce qui le concerne.

Certaines fonctionnalités, tel que la création d'exercices ou de chapitres, feront appel aux applications de mes compagnons. En effet, le bouton devra aller chercher les formulaires correspondant.

Les problèmes que nous pourrions rencontrer lors de la mise en commun de nos différentes applications seraient des conflits, mais qui pourraient facilement être gérables si on y fait attention en cherchant les erreurs.

Note de bas de page

Développement dirigé par les tests

Le développement dirigé par les tests, grossièrement traduit de l'anglais Test Driven Development, est une technique de programmation utilisée par beaucoup de programmeurs.

En effet, c'est grâce à cette technique que l'on peut le mieux s'assurer de la fonctionnalité du site et de la simplicité du code. Dans un travail de longue haleine, cette méthode devient nécessaire pour ne pas être redondant dans son code et pour le rendre le plus clair possible.

Dans cette première partie, nous allons nous intéresser au côté théorique de ce type de programmation avant de se lancer dans un réel projet.

Le développement dirigé par les tests est composé de deux tests importants et bien différents.

5.1 Le test fonctionnel

Le test fonctionnel, comme son nom l'indique, cherche à tester la fonctionnalité du site. Plus précisément, il se met du côté de l'utilisateur du site. Il va par exemple vérifier que les titres et les textes apparaissent, mais il va aussi regarder si les différents boutons ou champs de textes marchent.

Voici un exemple de test fonctionnel :

Mettre un test fonctionnel plus tard

Nous pouvons déjà voir que les commentaires sont énormément présents dans ce test. En effet, la convention est que l'on crée un scénario pour expliquer exactement ce que l'on teste. Dans cet exemple, nous avons le professeur Jean-Paul qui entend parler d'un site pour apprendre les mathématiques où il y a un dashboard qui lui permet de gérer son travail. Il va donc essayer tous les boutons et toutes les fonctionnalités.

Evidemment, cela peut paraître obsolète, mais ça vous aide à vous y retrouver.

5.2 Le test unitaire

Le test unitaire, lui, se base plus sur le point de vue du programmeur. Si on aurait pu considérer le test fonctionnel comme un test externe, le test unitaire lui serait le test interne. Ce qu'il test ne sera jamais vu, car il permet de vérifier que le code est le plus propre et le plus simple possible.

Voici un exemple :

Test unitaire A rajouter un jour

5.3 Le cycle du développement dirigé par les tests

Quand on veut programmer à l'aide du développement dirigé par les tests, on tente de suivre un certain cycle :

1. Tout d'abord, il est **impératif** d'écrire un test avant même d'écrire n'importe quelle ligne de code. En effet, l'idée du Test Driven Development est d'être sûr qu'un test échoue avant d'écrire notre code. Le test peut être fonctionnel ou unitaire, cela dépendant de la partie de votre application que vous souhaitez développer. Vous allez évidemment recevoir un message disant que votre test a échoué, mais c'est normal étant donné que vous n'avez rien codé. Cela consiste donc à essayer un programme encore non-existant.
2. Ensuite seulement, le but est d'écrire un minimum de code possible pour que le test précédemment lancé fonctionne. Il faut seulement s'occuper de ce que le test cherchait à essayer, pas plus, pas moins.
3. Une fois que vous pensez avoir accompli ce que le test de la première étape vous disait avoir raté, vous pouvez relancer ce test. Si le résultat est positif, vous pouvez passer à l'étape suivante. Dans le cas contraire, il va vous falloir refaire l'étape 2 et 3 jusqu'à ce que le test soit positif.
4. L'étape finale : il va falloir réécrire et réstructurer le code précédent histoire qu'il soit plus agréable à l'oeil et le rendre plus lisible. Il faut faire très attention durant cette étape à ne rien rajouter ou enlever. Le code doit garder le même résultat.

Une fois que ces 4 étapes ont été effectuées, il ne reste évidemment qu'à recommencer avec une autre fonctionnalité de l'application, jusqu'à ce que celle-ci soit finie.

5.4 Gain de temps ?

En lisant ces 4 étapes répétitives, on ne peut que se demander si le Test Driven Development et son cycle compliqué est réellement un atout et un gain de temps pour le programmeur.

Il est clair que, sur un travail de petite taille, tout coder n'aurait pas énormément de sens, car tout peut être facilement essayable par soi-même. Dans le cas d'un travail d'une certaine consistance, ce n'est pas pareil. C'est uniquement en testant que l'on peut être sûr de son code, car cela signifie que notre code est valide, et devrait le rester.

Débuter un projet

Pour comprendre grâce à la pratique, nous allons au cours de ce travail établir un dashboard permettant à un professeur de gérer des classes et des exercices sur un site d'e-learning pour les mathématiques grâce à Django, un framework fonctionnant avec Python.

6.1 Éléments requis

Pour pouvoir parfaitement suivre ce guide, il nous faudra les éléments suivants afin de réaliser les tests :

- **Django 1.7** : en effet, il va vous falloir django, car c'est le framework que l'on va utiliser. Pour le télécharger, il suffit de taper la commande suivante avec pip :

```
sudo pip3 install django==1.7
```

Si vous utilisez Windows, vous pouvez enlever *sudo*.

- **Selenium** : Selenium est un outil permettant de gérer les navigateurs avec des commandes. Ceci nous sera utile pour les tests fonctionnels (ce terme sera expliqué plus tard). Encore une fois, il est possible de le télécharger grâce à pip :

```
sudo pip3 install --upgrade selenium
```

Encore une fois, le *sudo* n'est pas nécessaire sur Windows.

Note : il est important de toujours utiliser la dernière version de Selenium. En effet, une version dépassée peut facilement se comporter de façon non désirée.¹

Une fois les éléments nécessaires installés, vous pouvez passer à la suite.

6.2 Premier test

Le principe de base du Test Driven Development est d'écrire un test avant même de coder ce que le test doit vérifier. Pour notre exemple, on devrait vérifier qu'il y ait le plus basique des éléments sur notre site : Django. On va donc créer un test fonctionnel pour voir s'il y a bien le titre de Django sur la page d'accueil. Le test fonctionnel permet de nous assurer que notre site fonctionne et possède la fonctionnalité la plus optimale qu'il soit.

Commençons donc par écrire ce code :

```
1 from selenium import webdriver
2
3 browser = webdriver.Firefox()
4 browser.get("http://localhost:8000")
```

1. Inspiré de Test Driven Development With Python de Harry J.W. Percival

```
5
6  assert "Django" in browser.title
7
8  browser.quit()
```

Regardons une par une les lignes qui pourrait poser des problèmes de compréhensions :

1. Nous permet d'importer webdriver qui nous sera utile pour gérer les navigateurs web (dans ce cas, Firefox), nous permettant de les ouvrir, d'aller à une URL ou de les fermer.
6. Va basiquement nous dire si la page que l'on vient de charger (<http://localhost:8000>) contient le mot "Django" dans le titre.

Comme on peut s'y attendre, du moins si on se rappelle du but d'un test, le test ne marchera pas. En effet, comme dit précédemment, les tests sont faits pour évaluer quelque chose que l'on n'a pas encore fait, et pour nous aider à les faire le plus simplement possible.

Quand on test, il ne faut pas avoir les tests qui échouent comme quelque chose de mal. Dans certains cas (comme celui-ci), ces tests sont attendus et recevoir un *False* à la fin est donc un bon signe : notre test marche !

Note de bas de page