Anto59290

Zeste de savoir

Le framework Django passe en version 2.0

12 août 2019

Table des matières

Django est un des *framework* web les plus populaires sur Internet. Basé sur le langage Python, il fait fonctionner de nombreux sites que vous utilisez au quotidien, dont Zeste de Savoir. Aujourd'hui marque une grande étape de la vie du projet Django avec la sortie de la version 2.0. Dans cet article nous verrons de plus près les changements apportés par cette nouvelle version.

1. Django, Python et cycles de vie

Après la sortie en avril 2017 de la version 1.11 de Django, l'équipe du projet a entamé un nouveau cycle de développement pour la version 2.0. La version 1.11 est une version LTS qui sera encore supportée jusqu'en avril 2020, la 2.0 quant à elle est une *release* «classique» dont le support est annoncé jusqu'en avril 2019.

Pour chaque version, l'équipe de développement de Django suit le même cycle de développement :

— Une phase d'intégration de nouvelles fonctionnalités, corrections de bugs, améliorations.
— Une première version alpha (alpha 1). À partir de cette version aucune fonctionnalité ne peut être ajoutée au projet. Le reste du temps est donc consacré à la correction de bugs.
— Une version bêta (bêta 1) qui corrige les bugs remontés lors de la version alpha.
— Une ou plusieurs versions RC qui sont des versions quasiment prêtes pour la sortie définitive. Lors de la phase de RC, seulement les patch concernant les bugs critiques sont autorisés afin de stabiliser le comportement du *framework*.
— La sortie définitive est faite, le projet est disponible dans sa version stable utilisable en production.

Ce cycle se perpétue ainsi depuis 2015 et permet d'avoir une nouvelle version majeure du projet tous les 8 mois et une version LTS tous les deux ans (une version sur 3 est une LTS). Le cycle a été pensé ainsi afin de permettre d'apporter toujours des nouveautés au projet, tout en fournissant des versions stables à intervalle régulier. Notons que les LTS sont supportées durant 3 ans, ce qui laisse aux utilisateurs de Django un an pour passer d'une version LTS à une autre si ils ne souhaitent pas utiliser les versions intermédiaires.

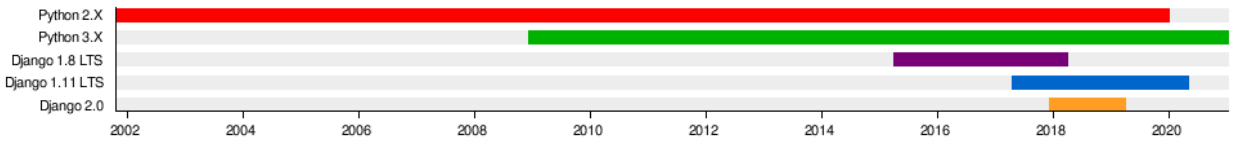

FIGURE 1. – Cycle de vie de Django et Python. On voit qu'entre deux versions LTS les développeurs ont une année pour mettre à jour leurs applications.

Django est un projet utilisant le langage de programmation open-source Python, qui lui-même suit son cycle de vie. L'écosystème Python est aujourd'hui à cheval entre deux versions majeure : la 2.X et la 3.X. La version 3.X est disponible depuis longtemps, mais de nombreuses applications n'ont pas encore migré vers la version 3.X. Ainsi en 2014, le support officiel de la version 2.X a été repoussé de 5 ans . La version 2.7 de Python est donc officiellement maintenue jusqu'en 2020. Le projet Django était jusqu'ici compatible avec les versions 2.X et 3.X de Python, la grande nouveauté de Django 2.0 est que celui-ci ne supporte plus Python 2.X.

> Django 2.0 supports Python 3.4, 3.5, and 3.6. We highly recommend and only officially support the latest release of each series. The Django 1.11.x series is the last to support Python 2.7.

Cela peut paraître anodin, mais c'est un grand changement pour la maintenance du projet. En effet le code ne devra plus être écrit que pour une version majeure de Python, cela va permettre une simplification du code a de nombreux endroits (exemple , autre exemple).

Enfin, la fin du support de la version 2.X de Python amène un changement au niveau des chaînes de caractères. En effet la version Python 2.X utilise les chaînes de caractères sous forme de byte-strings là où Python 3.X utilise des chaînes de caractères unicode. C'est-à-dire une chaîne de caractères représentée dans un certain encodage. À partir de Django 2.0 les développeurs du projet n'auront plus à se soucier de cette différence, car seules les chaînes de caractères unicodes seront supportées.

À noter que la fin de vie de Python 3.4 est programmée pour mars 2019, Django 2.0 sera donc la dernière version à supporter officiellement Python 3.4. Les développeurs souhaitant rester sur Python 3.4 devront donc s'orienter vers Django 1.11.

2. Quels changements ?

En dehors des améliorations liées aux changements de version, de nombreuses améliorations fonctionnelles sont aussi apportées par cette nouvelle version de Django. Regardons de plus près ce que la 2.0 nous réserve.

2.0.1. Nouvelle syntaxe de routage

Le routage (ou *routing* en anglais) est un mécanisme permettant d'associer à des URL des « pages » consultables par un visiteur. Chaque « page » est composée d'une vue qui contrôle la

logique et d'un *template* qui contrôle l'affichage et la mise en page du contenu. Les mécanismes de routages sont très populaires dans les *frameworks* web comme Django, Ruby on Rails ou Symfony. En effet ils permettent de gérer de manière simple et rapide le code à exécuter en fonction de l'URL demandée.

Dans sa version 2.0 Django a amélioré le routage afin de le rendre plus simple à utiliser. Dans les versions précédentes de Django, la syntaxe pour décrire une route était la suivante :

```
1 url(r'^articles/(?P<year>[0-9]{4})/$', views.year_archive),
```

Le premier argument était une expression régulière décrivant l'URL à laquelle la vue serait associée. Les expressions régulières sont des outils très puissants et permettent de définir avec une grande finesse les caractères autorisés dans une URL. L'inconvénient est en revanche la difficulté pour lire et écrire ces expressions qui utilisent leur propre langage. La nouvelle syntaxe proposée corrige ce défaut en rendant les routes plus lisibles :

```
1 path('articles/<int:year>/', views.year_archive),
```

La nouvelle syntaxe apporte aussi un autre avantage : en obligeant le développeur à préciser le type elle va automatiquement convertir la variable vers le bon type. Ainsi avec l'ancienne syntaxe la variable *year* aurait été reçue en tant que chaîne de caractères par la vue, elle sera maintenant automatiquement convertie en entier. Les types disponibles pour le routage sont les suivants :

— str : Les chaînes de caractères, à l'exclusion du caractère '/'. Si vous ne précisez pas de type, c'est celui-ci qui sera choisi par défaut.
— int : Les entiers positifs ou nuls.
— slug : Les slugs valides pour Django, constitués de lettres, chiffres ou de tirets
— uuid : Un identifiant unique dans un format compris par Python .
— path : N'importe quelle chaîne de caractères, y comprit le caractère '/'.

L'ancienne syntaxe reste disponible et compatible pour le moment, plus d'informations dans la documentation officielle .

2.0.2. L'administration en responsive design

L'un des modules les plus aimés des amateurs de Django est le module contrib.admin qui permet en quelques lignes d'obtenir une interface d'administration simple et puissante. Ce *back-office* est une vraie force car il permet d'obtenir de bons résultats très rapidement pour construire une interface d'administration. Jusqu'ici ce module n'était pas responsive et était donc difficilement utilisable sur tablette ou sur mobile. Un effort a été fait dans cette version 2.0 pour rendre cette interface *mobile-friendly*.

Ce n'est pas un changement énorme, mais il améliore tout de même beaucoup l'utilisabilité de ce module au quotidien.

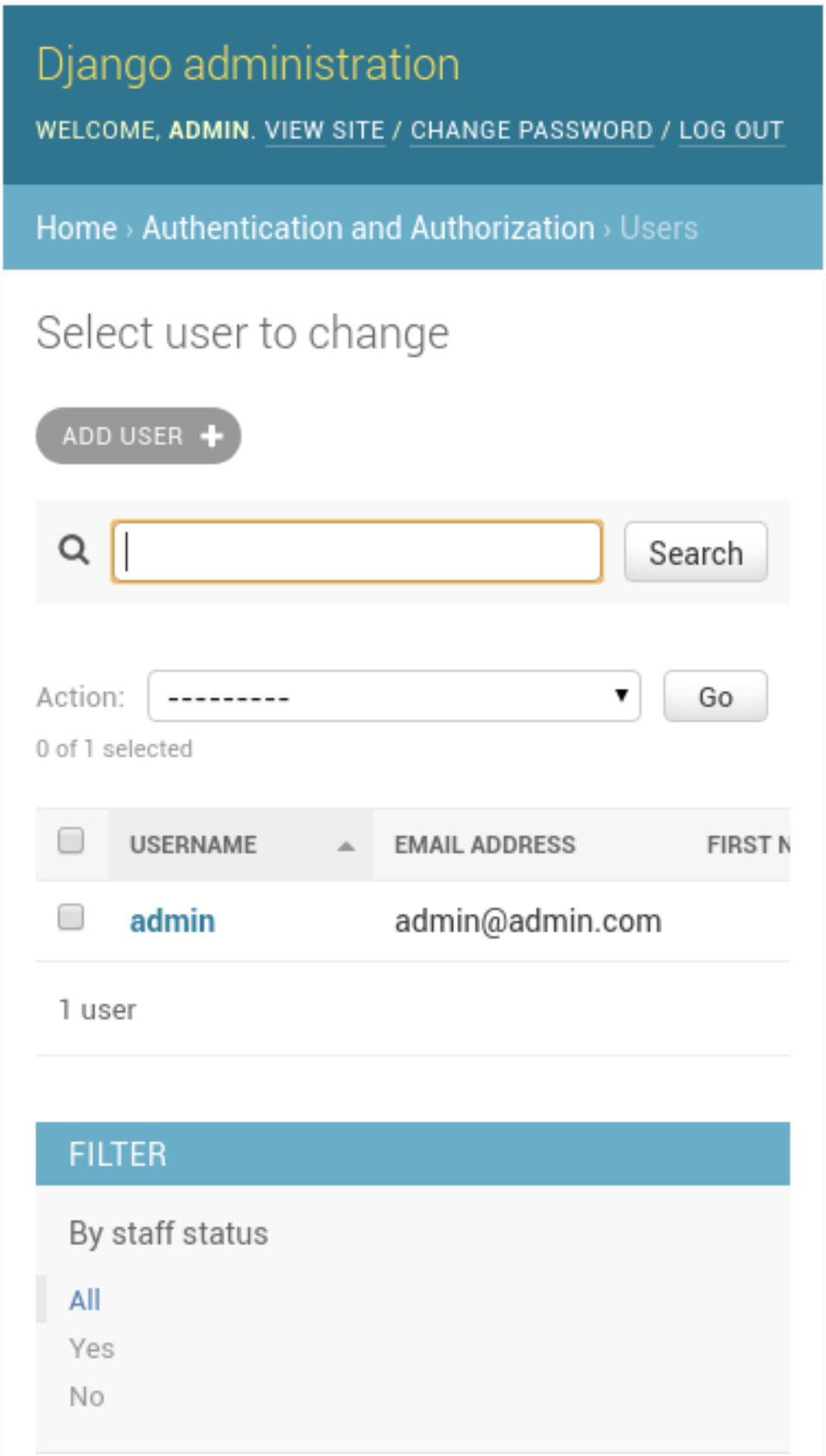

FIGURE 2. – La nouvelle console d'administration avec une interface en responsive design.

À noter au passage la possibilité d'utiliser le plugin jQuery Select2 dans l'interface d'administration . Ce plugin améliore les champs à choix multiples en permettant la recherche et en

allant chercher les informations de manière asynchrone.

2.0.3. Nouvelles fonctions d'agrégations

De nouvelles expressions ont aussi fait leur apparition dans l'ORM de Django, mais commençons par un bref rappel de la notion.

Un ORM est une technique permettant d'abstraire la gestion de la base de données dans une application. Ainsi plus besoin d'utiliser directement des requêtes SQL dans le code, on peut parler en utilisant la notion d'objets. Cette abstraction rend la manipulation de données plus simple et plus lisible. Un ORM permet en général de changer plus facilement de type de base de données, car l'ORM traduit vos requêtes objets en requêtes SQL comprises par votre base de données. Voici quelques exemples d'utilisation de l'ORM de Django :

```
1 Post.objects.all() # Récupére tous les posts
2 Post.objects.filter(public=True) # Uniquement les posts publics
3 Post.objects.filter(author=user) # Uniquement les posts de
      l'utilisateur
4 etc.
```

L'ORM de Django a été constamment enrichi au fur et à mesure des versions, ici c'est l'expression *Window* qui fait son apparition. Cette expression permet de réaliser des calculs (moyenne, minimum, maximum, etc.) sur une partition des données. La documentation propose l'exemple suivant : imaginons que dans une application de notations de films nous souhaitons annoter pour chaque film la moyenne des films du même studio. On peut réaliser une partition sur la condition «films du même studio» :

```
1 Movie.objects.annotate(
2    avg_rating=Window(expression=Avg('rating'),
3          partition_by=[F('studio')]))
```

Le résultat serait le suivant :

Film	Studio	Rating	Avg_Rating
Tempête de clémentines géantes	Studio Clem	3.0	4.0
Les zestes contre-attaquent	Studio Clem	5.0	4.0
Underworlf	Studio Lupin	1.0	2.5
Danse avec les louves	Studio Lupin	4.0	2.5
Carnets de lama	Studio Lama	3.5	3.5

On voit que pour chaque film on récupère la notation moyenne du studio. Il faut bien noter la différence avec la fonction group_by qui permet aussi d'annoter à l'aide de fonctions, même qui

ne renvoie qu'une ligne par partition. Avec un group_by sur le studio on aurait eu le résultat suivant

Studio	Avg_Rating
Studio Clem	4.0
Studio Lupin	2.5
Studio Lama	3.5

Il devient alors très simple de déterminer si un film est un bon ou un mauvais élève au sein de sa partition. Il est aussi possible d'annoter la partition avec plusieurs fonctions en même temps, par exemple pour récupérer la meilleure note ou la pire de la partition.

Nous avons fait le tour des nouvelles fonctionnalités les plus importantes de cette version. Je vous invite à consulter la page détaillant tous les changements pour en savoir plus sur toutes les modifications mineures qui ont été faites.

C'est tout pour cette nouvelle version de Django ! On peut dire pour conclure qu'il s'agit d'une version ne comprenant pas de grandes nouvelles fonctionnalités, mais essentielle pour l'avenir du projet. Pour la suite les développeurs reprendront le cycle de *release* en vue de préparer la version 2.1 qui devrait donc sortir dans 8 mois environ.

2.1. Sources

- Les notes de la release
- Un article sur le sujet sur Developpez.com
- La date de fin de support de la version 2.0 repoussée
- Annonce de la sortie de la RC1
- Calendrier de *release* de la version 2.0

2.2. Aller plus loin

- L'encodage avec Python, chez Sam et Max
- Article sur la sortie de Python 3.6

Un grand merci à entwanne pour sa relecture précieuse et merci à Gabbro pour la validation.

Liste des abréviations

LTS Long Term Support. 1

ORM Object Relational Mapping. 5

RC Release Candidate. 1