eBook Gratuit

APPRENEZ tastypie

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#tastypie

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: tastypie

It is an unofficial and free tastypie ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official tastypie.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Commencer avec tastypie

Remarques

Cette section fournit une vue d'ensemble de ce qu'est goût et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets au sein de tastypie, et établir un lien avec les sujets connexes. La documentation de Tasteypie étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Examples

Installation ou configuration

Des instructions détaillées sur la mise en place ou l'installation de la goûts

Lire Commencer avec tastypie en ligne: https://riptutorial.com/fr/tastypie/topic/9618/commencer-avec-tastypie

Chapitre 2: Codage avec Tastypie

Introduction

Tastypie est un framework API webservice pour Django. Il fournit une abstraction pratique, mais puissante et hautement personnalisable pour créer des interfaces de type REST. Tastypie facilite l'exposition de vos modèles, mais vous donne un contrôle total sur ce que vous exposez, vous permettant d'abstraire la base de données autant que nécessaire. Tastypie facilite également l'intégration avec des sources de données non-ORM. Ainsi, Tastypie peut être utilisé avec des bases de données ORM et non-ORM.

Examples

Démarrage rapide

1. Ajouter tastypie à INSTALLED_APPS.
2. Créez un répertoire api dans votre application avec une **init** .
3. Créez un fichier <my_app> /api/resources.py et placez-y les éléments suivants:

```
from tastypie.resources import ModelResource
from my_app.models import MyModel


class MyModelResource(ModelResource):
    class Meta:
        queryset = MyModel.objects.all()
        allowed_methods = ['get']
```

Dans votre URLconf racine, ajoutez le code suivant (autour du code administrateur):

```
from django.conf.urls import url, include
from tastypie.api import Api
from my_app.api.resources import MyModelResource

v1_api = Api(api_name='v1')
v1_api.register(MyModelResource())

urlpatterns = [
  # ...more URLconf bits here...
  # Then add:
  url(r'^api/', include(v1_api.urls)),
]
```

Installation Tastypie

Tastypie peut être installé avec la gestion des paquets python, c.-à-d. `pip` ou nous pouvons directement extraire le code de Github

1. pip installation django-tastypie

2. Caisse de Github

Pourquoi Tastypie?

Il existe d'autres frameworks API pour Django. Vous devez évaluer les options disponibles et décider vous-même. Cela dit, voici quelques raisons communes pour goûter au goût.

- Vous avez besoin d'une API RESTful et utilise bien HTTP.
- Vous voulez soutenir des relations profondes.
- Vous ne voulez pas avoir à écrire votre propre sérialiseur pour rendre la sortie correcte.
- Vous voulez un framework API qui ait peu de magie, très flexible et qui s'adapte bien au domaine problématique.
- Vous souhaitez / avez besoin d'une sérialisation XML traitée de la même manière que JSON (et YAML est là aussi).

Pour développer une API avec Django, nous avons plusieurs options. Les principales options sont **Tastypie** et **Django Rest Framework** (DRF).

Qu'est-ce qui fait un framework API décent?

Ces fonctionnalités:

1. pagination
2. affichage des données avec validation
3. Publication de métadonnées avec des groupes de requêtes
4. Découverte de l'API
5. gestion correcte des réponses HTTP
6. mise en cache
7. sérialisation
8. étranglement
9. autorisations
10. authentification

Les structures API appropriées doivent également:

11. Très bonne couverture de test de leur code
12. Performance décente
13. Documentation
14. Une communauté active pour avancer et soutenir le cadre

Si vous prenez ces facteurs, il n'y a actuellement que deux frameworks API à utiliser (dépend des vues utilisateur), django-tastypie et django-rest-framework.

Quel est le meilleur? django-tastypie ou django-rest-framework?

Je dis qu'ils sont égaux. Vous ne pouvez tout simplement pas vous tromper avec l'un ou l'autre. Les auteurs et les communautés derrière eux sont actifs, le code est solide et testé. Et voici mes

pensées spécifiques sur les deux:

Tastypie:

Avantages:

- Facilité de démarrage et fonctionnalités de base OOB (out of the box)
- La plupart du temps, vous ne serez pas confronté à des concepts avancés de Django tels que les CBV, les formulaires, etc.
- Code plus lisible et moins de magie!
- Si vos modèles sont non-ORM, allez-y.

Désavantages:

- Ne suit pas strictement Django idiomatique (les philosophies de python et de django sont bien différentes)
- Probablement un peu difficile à personnaliser les API une fois que vous êtes gros
- Pas d'O-Auth

DRF:

Avantages:

- Suivez le django idiomatique. (Si vous connaissez le Django à l'envers, et très à l'aise avec CBV, Forms etc sans aucun doute, allez-y)
- Fournit une fonctionnalité REST prête à l'emploi à l'aide de ModelViewSets. Dans le même temps, offre un meilleur contrôle de la personnalisation à l'aide de CustomSerializer, APIView, GenericViews, etc.
- Meilleure authentification
- Plus facile d'écrire des classes d'autorisation personnalisées.
- Travaillez très bien et surtout très facilement pour le faire fonctionner avec des bibliothèques tierces et OAuth.
- DJANGO-REST-AUTH mérite d'être mentionné. LIBRARY for Auth / SocialAuthentication / Registration.

Désavantages:

- Si vous ne connaissez pas très bien Django, ne faites pas cela.
- La magie! Quelque temps très difficile de comprendre la magie.
- Parce que cela a été écrit au-dessus du CBV de Django, qui est lui-même très complexe par nature.
- A courbe d'apprentissage raide.

Premiers pas avec Tastypie

Tastypie est une application réutilisable (c'est-à-dire qu'elle repose uniquement sur son propre code et se concentre uniquement sur une API de type REST) et convient pour fournir une API à toute application sans avoir à modifier les sources de cette application.

Tous les besoins ne sont pas les mêmes, alors Tastypie fait de son mieux pour fournir de nombreux points d'ancrage afin de modifier ou d'étendre son fonctionnement.

Par exemple, nous allons ajouter une API à une application de blog simple.

La seule configuration obligatoire consiste à ajouter ' ***tastypie*** *' à votre* ***INSTALLED_APPS*** *. Ce n'est pas strictement nécessaire, car Tastypie ne dispose que de deux modèles non obligatoires, mais peut faciliter son utilisation.*

Voici myapp / models.py:

```
from tastypie.utils.timezone import now
from django.contrib.auth.models import User
from django.db import models
from django.utils.text import slugify


class Entry(models.Model):
    user = models.ForeignKey(User)
    pub_date = models.DateTimeField(default=now)
    title = models.CharField(max_length=200)
    slug = models.SlugField(null=True, blank=True)
    body = models.TextField()
```

Créer des ressources

L'architecture de style REST parle de ressources, donc une intégration sans surprise avec Tastypie implique la création de *classes de ressources* . Pour notre application simple, nous allons créer un fichier dans **myapp / api.py** , bien qu'ils puissent vivre n'importe où dans votre application:

myapp / api.py

```
from tastypie.resources import ModelResource
from myapp.models import Entry


class EntryResource(ModelResource):
    class Meta:
        queryset = Entry.objects.all()
        resource_name = 'entry'
```

Dans REST, tout est une ressource, les objets sont décrits comme des ressources. Ainsi, la création d'une API REST Tastypie implique la création d'une classe de ressources. Pour créer une ressource, nous devons sous-classer la classe ModelResource. Cette classe EntryResource vérifie tous les champs non relationnels du modèle Entry et crée ses propres champs Api. *Cela fonctionne comme le ModelForm de Django Forms.*

Connexion de la **ressource:** Après avoir créé la ressource, vous devez indiquer à Django qu'une ressource est créée en connectant notre source d'entrée à l'URL.

urls.py

```
from django.conf.urls import url, include
from myapp.api import EntryResource

entry_resource = EntryResource()

urlpatterns = [
    url(r'^blog/', include('myapp.urls')),
    url(r'^api/', include(entry_resource.urls)),
]
```

Maintenant, si nous vérifions sur **localhost: 8000 / api / entry /** nous obtiendrons la réponse de l'API.

*(Le **nom_ressource*** au *sein* de *la classe Meta est facultative. Si non fourni, il est généré automatiquement le nom* de *classe,* en *supprimant toutes* les *instances de ressources et la chaîne* en *minuscule. Donc **EntryResource deviendrait simplement entrée.** Voilà pourquoi dans l'URL* que *nous avons spécifié **../entry /**)*

Autorisation Tastypie

Jusqu'à présent, nous avons essayé d'obtenir une demande. *(Si vous devez essayer une autre requête HTTP, utilisez des outils de test API tels que curl ou postman)*

Si vous essayez d'envoyer un POST / PUT / DELETE à la ressource, vous vous retrouvez avec des erreurs «401 non autorisées». Pour plus de sécurité, Tastypie est livré avec la classe d'autorisation définie sur ReadOnlyAuthorization. Cela le rend sûr à exposer sur le Web, mais nous empêche de faire POST / PUT / DELETE. Actifions ceux-là.

myapp / api.py

```
from tastypie.authorization import Authorization
from tastypie.resources import ModelResource
from myapp.models import Entry


class EntryResource(ModelResource):
    class Meta:
        queryset = Entry.objects.all()
        resource_name = 'entry'
        authorization = Authorization()
```

Lire Codage avec Tastypie en ligne: https://riptutorial.com/fr/tastypie/topic/10640/codage-avec-tastypie

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec tastypie	Community
2	Codage avec Tastypie	AKHIL MATHEW