EBook Gratuito

APPRENDIMENTO

tastypie

Free unaffiliated eBook created from
Stack Overflow contributors.

#tastypie

Sommario

Di

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

Capitolo 1: Iniziare con tastypie

Osservazioni

Questa sezione fornisce una panoramica di ciò che è tasteypie, e perché uno sviluppatore potrebbe voler usarlo.

Dovrebbe anche menzionare qualsiasi argomento di grandi dimensioni all'interno di tasteypie e collegarsi agli argomenti correlati. Poiché la Documentation per tastypie è nuova, potrebbe essere necessario creare versioni iniziali di tali argomenti correlati.

Examples

Installazione o configurazione

Istruzioni dettagliate su come installare o installare tasteypie.

Leggi Iniziare con tastypie online: https://riptutorial.com/it/tastypie/topic/9618/iniziare-con-tastypie

Capitolo 2: Coding con Tastypie

introduzione

Tastypie è un framework API webservice per Django. Fornisce un'astrazione comoda, ma potente e altamente personalizzabile per la creazione di interfacce in stile REST. Tastypie rende semplice l'esposizione dei tuoi modelli, ma ti dà il controllo completo su ciò che esponi, consentendoti di estrarre il database quanto necessario. Tastypie semplifica inoltre l'integrazione con origini dati non ORM. Quindi Tastypie può essere utilizzato con i database ORM e Non-ORM.

Examples

Avvio veloce

1. Aggiungi tastypie a INSTALLED_APPS.
2. Crea una directory API nella tua app con un **init init** .py.
3. Crea un file <my_app> /api/resources.py e inserisci quanto segue:

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


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

Nel tuo URL root conf, aggiungi il seguente codice (intorno a dove potrebbe essere il codice amministratore):

```
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)),
]
```

Installazione di Tastypie

Tastypie può essere installato con la gestione dei pacchetti python, ovvero `pip` o possiamo controllare direttamente il codice da Github

1. pip installa django-tastypie

2. Pagamento da Github

Perché Tastypie?

Ci sono altri framework API là fuori per Django. È necessario valutare le opzioni disponibili e decidere autonomamente. Detto questo, ecco alcuni motivi comuni per tastypie.

- Hai bisogno di un'API che sia RESTful e usi bene HTTP.
- Vuoi sostenere relazioni profonde.
- NON si desidera scrivere il proprio serializzatore per rendere l'output corretto.
- Vuoi un framework API che abbia poca magia, molto flessibile e che si adatti bene al dominio del problema.
- Vuoi / hai bisogno della serializzazione XML che viene trattata in modo uguale a JSON (e c'è anche YAML).

Per sviluppare API con Django, abbiamo molte opzioni. Le opzioni principali sono **Tastypie** e **Django Rest Framework** (DRF).

Cosa rende un Framework API decente?

Queste caratteristiche:

1. paginatura
2. pubblicazione di dati con convalida
3. Pubblicazione di metadati insieme a querysets
4. Scoperta API
5. corretta gestione della risposta HTTP
6. caching
7. serializzazione
8. strozzamento
9. permessi
10. autenticazione

Anche i framework API corretti richiedono:

11. Copertura del test davvero buona del loro codice
12. Prestazione decente
13. Documentazione
14. Una comunità attiva per avanzare e supportare il framework

Se prendi questi fattori, al momento ci sono solo due framework API da utilizzare (dipende dalle visualizzazioni degli utenti), django-tastypie e django-rest-framework.

Qual è il migliore? django-tastypie o django-rest-framework?

Dico che sono uguali. Semplicemente non puoi sbagliare con nessuno dei due. Gli autori e le comunità dietro a entrambi sono attivi, il codice è solido e testato. E qui ci sono i miei pensieri

specifici su entrambi:

Tastypie:

vantaggi:

- Facile da utilizzare e funzionalità di base OOB (pronta all'uso)
- La maggior parte delle volte non avrai a che fare con concetti avanzati di Django come CBV, moduli ecc
- Codice più leggibile e meno magia!
- Se i tuoi modelli sono NON-ORM, fallo.

svantaggi:

- Non segue rigidamente Django idiomatico (la mente ben pitone e le filosofie del django sono molto diverse)
- Probabilmente è un po 'difficile personalizzare le API una volta diventato grande
- No O-Auth

DRF:

vantaggi:

- Segui il django idiomatico. (Se conosci il django dentro e molto comodo con CBV, Forms etc senza alcun dubbio vai a prenderlo)
- Fornisce la funzionalità REST fuori dalla scatola utilizzando ModelViewSets. Allo stesso tempo, offre un maggiore controllo per la personalizzazione utilizzando CustomSerializer, APIView, GenericViews ecc.
- Migliore autenticazione
- Più facile scrivere classi di permessi personalizzati.
- Funziona molto bene e, soprattutto, è molto semplice farlo funzionare con le librerie di terze parti e OAuth.
- DJANGO-REST-AUTH vale la pena menzionare LIBRARY per Auth / SocialAuthentication / Registration.

svantaggi:

- Se non conosci molto bene Django, non fare questo.
- Magia! Qualche tempo molto difficile per capire la magia.
- Perché è stato scritto in cima al CBV del django che sono a loro volta di natura piuttosto complessa.
- Ha una curva di apprendimento ripida.

Iniziare con Tastypie

Tastypie è un'app riutilizzabile (ovvero, si basa solo sul proprio codice e si concentra sulla fornitura di un'API in stile REST) ed è adatta per fornire un'API a qualsiasi applicazione senza dover modificare le origini di tale app.

Non tutti i bisogni sono uguali, quindi Tastypie si impegna a fornire un sacco di aggancio per scavalcare o estendere il modo in cui funziona.

> Ad esempio, aggiungeremo un'API a una semplice applicazione per blog.

*L'unica configurazione necessaria è l'aggiunta di **'tastypie'** al vostro **INSTALLED_APPS.** Questo non è strettamente necessario, in quanto Tastypie ha solo due modelli non richiesti, ma può facilitare l'utilizzo.*

Ecco 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()
```

Creare risorse

L'architettura in stile REST parla di risorse, quindi l'integrazione non sorprendente con Tastypie implica la creazione di *classi di risorse* . Per la nostra semplice applicazione, creeremo un file per questi in **myapp / api.py** , sebbene possano vivere ovunque nella tua applicazione:

frontend / api.py

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


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

In REST tutto è una risorsa, significa che gli oggetti sono descritti come risorse. Quindi per la creazione di un'API REST di Tastypie è necessario creare una classe di risorse. Per creare una risorsa, è necessario suddividere la classe ModelResource. Questa classe EntryResource controllerà tutti i campi non relazionali sul modello Entry e creerà i propri campi API. *Funziona proprio come ModelForm di Django Forms.*

Collegando la risorsa: Afrer crea la risorsa, dobbiamo dire al Django che una risorsa viene creata agganciando la nostra EntryResource all'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)),
]
```

Ora se controlliamo **localhost: 8000 / api / entry /** otterremo la risposta API.

*(Il **nome_risorsa** all'interno della classe Meta è facoltativo.Se non viene fornito, viene generato automaticamente fuori dal nome della classe, rimuovendo eventuali istanze di Risorsa e minuscole della stringa. Quindi **EntryResource diventerebbe solo una voce** . Ecco perché nell'URL abbiamo specificato **../entry /**)*

Autorizzazione Tastypie

Fino ad ora abbiamo provato a ottenere la richiesta. *(Se è necessario provare altre risorse HTTP, utilizzare alcuni strumenti di test API come curl o postman)*

Se provate a inviare un POST / PUT / DELETE alla risorsa, vi ritroverete a ricevere errori "401 non autorizzati". Per sicurezza, Tastypie viene fornito con la classe di autorizzazione impostata su ReadOnlyAuthorization. Questo lo rende sicuro da esporre sul web, ma ci impedisce di fare POST / PUT / DELETE. Let's abilitare quelli.

frontend / 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()
```

Leggi Coding con Tastypie online: https://riptutorial.com/it/tastypie/topic/10640/coding-con-tastypie

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con tastypie	Community
2	Coding con Tastypie	AKHIL MATHEW