Бесплатная электронная книга

учусь

tastypie

Free unaffiliated eBook created from
Stack Overflow contributors.

#tastypie

Около

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

глава 1: Начало работы с tastypie

замечания

В этом разделе представлен обзор того, что такое tastypie, и почему разработчик может захотеть его использовать.

Следует также упомянуть о любых крупных предметах в рамках tastypie и ссылки на связанные темы. Поскольку Документация для tastypie является новой, вам может потребоваться создать начальные версии этих связанных тем.

Examples

Установка или настройка

Подробные инструкции по настройке или установке tastypie.

Прочитайте Начало работы с tastypie онлайн: https://riptutorial.com/ru/tastypie/topic/9618/начало-работы-с-tastypie

глава 2: Кодирование с помощью Tastypie

Вступление

Tastypie - это API-интерфейс webservice для Django. Он обеспечивает удобную, но мощную и настраиваемую абстракцию для создания интерфейсов типа REST. Tastypie упрощает демонстрацию ваших моделей, но дает вам полный контроль над тем, что вы раскрываете, позволяя вам абстрагировать базу данных столько, сколько необходимо. Tastypie также упрощает интеграцию с источниками данных, отличных от ORM. Следовательно, Tastypie можно использовать с базами данных ORM и без ORM.

Examples

Быстрый старт

1. Добавьте tastypie в INSTALLED_APPS.
2. Создайте каталог api в своем приложении с голой **init** .py.
3. Создайте файл <my_app> /api/resources.py и поместите в него следующее:

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


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

В корневой URLconf добавьте следующий код (вокруг которого может быть код администратора):

```
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)),
]
```

Установка Tastypie

Tastypie можно установить с помощью управления пакетами python, то есть с помощью `pip`

или мы можем напрямую проверить код от Github

1. pip install django-tastypie
2. Оформить заказ от Github

Почему Тастипи?

Для Django существуют другие интерфейсы API. Вам нужно оценить доступные варианты и решить сами. Тем не менее, вот некоторые общие причины тастипии.

- Вам нужен API, который RESTful и хорошо использует HTTP.
- Вы хотите поддерживать глубокие отношения.
- Вы НЕ хотите писать собственный сериализатор для правильного вывода.
- Вам нужна инфраструктура API, которая обладает малой магией, очень гибкой и хорошо отображает проблемную область.
- Вам нужна / нужна XML-сериализация, которая одинаково обрабатывается JSON (и YAML тоже есть).

Чтобы разработать API с Django, у нас есть много вариантов. Основными параметрами являются **Tastypie** и **Django Rest Framework** (DRF).

Что делает достойную API Framework?

Эти функции:

1. пагинация
2. размещение данных с проверкой
3. Публикация метаданных вместе с запросами
4. Открытие API
5. правильная обработка HTTP-ответа
6. кэширование
7. сериализация
8. дросселирование
9. разрешений
10. аутентификация

Для правильной структуры API также необходимы:

11. Действительно хорошее тестирование покрытия их кода
12. Достойная производительность
13. Документация
14. Активное сообщество для продвижения и поддержки рамок

Если вы примете эти факторы, на данный момент стоит использовать только две рамки API

(зависит от пользовательских представлений), django-tastypie и django-rest-framework.

Какой из них лучше? django-tastypie или django-rest-framework?

Я говорю, что они равны. Вы просто не можете ошибиться ни с одним. Авторы и сообщества, стоящие за обоими из них, активны, код прочный и проверен. И вот мои конкретные мысли о них обоих:

Tastypie:

Преимущества:

- Легко начать работу и предоставить базовые функции OOB (из коробки)
- В большинстве случаев вы не будете иметь дело с концепциями Advanced Django, такими как CBV, Forms и т. Д.
- Более читаемый код и меньше магии!
- Если ваши модели не имеют ORM, пойдите для этого.

Недостатки:

- Не следует строго следовать идиоматическому Django (ум хорошо питона и философии django совершенно разные)
- Наверное, немного сложно настроить API, как только вы пойдете
- Нет O-Auth

ФПИ:

Преимущества:

- Следуйте идиоматическому джанго. (Если вы знаете django наизнанку, и очень удобно с CBV, Forms и т. Д., Без всяких сомнений, идите на это)
- Предоставляет функциональность REST из коробки с помощью ModelViewSets. В то же время, обеспечивает более широкий контроль для настройки с помощью CustomSerializer, APIView, GenericViews и т. Д.
- Лучшая аутентификация.
- Легче писать пользовательские классы разрешений.
- Работайте очень хорошо и очень легко, чтобы он работал с сторонними библиотеками и OAuth.
- DJANGO-REST-AUTH стоит упомянуть БИБЛИОТЕКУ для Auth / SocialAuthentication / Registration.

Недостатки:

- Если вы не знаете Джанго очень хорошо, не идите на это.
- Магия! Некоторое время очень трудно понять магию.
- Потому что он был написан поверх CBV django, которые в свою очередь довольно сложны по своей природе.
- Имеет крутую кривую обучения.

Начало работы с Tastypie

Tastypie - приложение многократного использования (то есть оно опирается только на собственный код и фокусируется на предоставлении только API-интерфейса REST) и подходит для предоставления API любому приложению без необходимости изменять источники этого приложения.

Не все потребности одинаковы, поэтому Tastypie делает все возможное, чтобы обеспечить множество крючков для переопределения или расширения того, как это работает.

> Например, мы добавим API в простое приложение для блога.

*Единственная обязательная конфигурация - добавление « **tastypie** » к вашему **INSTALLED_APPS** . Это не является абсолютно необходимым, так как Tastypie имеет только две необязательные модели, но может облегчить использование.*

Здесь находится 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()
```

Создание ресурсов

Архитектура REST-стиля говорит о ресурсах, поэтому неудивительно интегрироваться с Tastypie связано с созданием *классов ресурсов* . Для нашего простого приложения мы создадим файл для них в **myapp / api.py** , хотя они могут жить в любом месте приложения:

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'
```

В REST все является ресурсом, значит объекты описываются как ресурсы. Таким образом, для создания REST API Tastypie требуется создание класса ресурсов. Чтобы создать ресурс, нам нужно подклассифицировать класс ModelResource. Этот класс EntryResource проверяет все нереляционные поля на модели ввода и создает свои собственные поля Api. *Это работает так же, как ModelForm Django Forms.*

Захват **ресурса:** создайте ресурс, нам нужно сообщить Django, что ресурс создается путем привязки нашего EntryResource к URL-адресу.

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)),
]
```

Теперь, если мы проверим **localhost: 8000 / api / entry /,** мы получим ответ API.

*(Имя **ресурса** в классе Meta не является обязательным. Если оно не указано, оно автоматически генерируется с именем класса, удаляя все экземпляры Resource и **уменьшая размер** строки. Таким образом, **EntryResource станет просто записью** . Вот почему в указанном нами URL- **адресе ../entry /**)*

Разрешение Tastypie

До сих пор мы пытались получить запрос. *(Если вам нужно попробовать другое HTTP-опрос, используйте некоторые инструменты тестирования API, такие как завиток или почтальон)*

Если вы попытаетесь отправить POST / PUT / DELETE на ресурс, вы получите ошибки «401 Unauthorized». Для безопасности Tastypie поставляется с классом авторизации, установленным для ReadOnlyAuthorization. Это позволяет безопасно открывать в Интернете, но не позволяет нам делать POST / PUT / DELETE. Давайте включим их.

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()
```

Прочитайте Кодирование с помощью Tastypie онлайн:
https://riptutorial.com/ru/tastypie/topic/10640/кодирование-с-помощью-tastypie

кредиты

S. No	Главы	Contributors
1	Начало работы с tastypie	Community
2	Кодирование с помощью Tastypie	AKHIL MATHEW