**EBook Gratis**

# APRENDIZAJE tastypie

Free unaffiliated eBook created from
**Stack Overflow contributors.**

#tastypie

# Tabla de contenido

# Acerca de

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

# Capítulo 1: Empezando con tastypie

## Observaciones

Esta sección proporciona una descripción general de qué es tastypie y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro de tastypie y vincular a los temas relacionados. Dado que la Documentación para tastypie es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

## Examples

### Instalación o configuración

Instrucciones detalladas para configurar o instalar tastypie.

Lea Empezando con tastypie en línea: https://riptutorial.com/es/tastypie/topic/9618/empezando-con-tastypie

# Capítulo 2: Codificando con Tastypie

## Introducción

Tastypie es un marco de API de servicio web para Django. Proporciona una abstracción conveniente, pero potente y altamente personalizable para crear interfaces estilo REST. Tastypie facilita la exposición de sus modelos, pero le da un control total sobre lo que expone, lo que le permite abstraer la base de datos tanto como sea necesario. Tastypie también facilita la integración con fuentes de datos que no son ORM. Por lo tanto, Tastypie se puede utilizar con bases de datos ORM y no ORM.

## Examples

### Inicio rápido

1. Añade tastypie a INSTALLED_APPS.
2. Cree un directorio de API en su aplicación con un simple **init** .py.
3. Cree un archivo <my_app> /api/resources.py y coloque lo siguiente en él:

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


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

En su URLconf de raíz, agregue el siguiente código (alrededor de donde podría estar el código de administración):

```
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)),
]
```

### Instalación de Tastypie

Tastypie se puede instalar con la administración de paquetes de Python, es decir, `pip` o podemos verificar directamente el código de Github

1. pip instalar django-tastypie
2. Salida desde Github

## ¿Por qué Tastypie?

Hay otros frameworks API para Django. Necesita evaluar las opciones disponibles y decidir por sí mismo. Dicho esto, aquí hay algunas razones comunes para tastypie.

- Necesita una API que sea RESTful y que use bien HTTP.
- Quieres apoyar relaciones profundas.
- Usted NO quiere tener que escribir su propio serializador para hacer que la salida sea correcta.
- Desea un marco de API que tenga poca magia, sea muy flexible y se asigne bien al dominio del problema.
- Desea / necesita la serialización XML que se trata por igual a JSON (y YAML está ahí también).

Para desarrollar API con Django, tenemos muchas opciones. Las opciones principales son **Tastypie** y **Django Rest Framework** (DRF).

**¿Qué hace un marco de API decente?**

*Estas características:*

1. paginación
2. Publicación de datos con validación.
3. Publicación de metadatos junto con querysets
4. Descubrimiento de API
5. manejo adecuado de la respuesta HTTP
6. almacenamiento en caché
7. publicación por entregas
8. estrangulamiento
9. permisos
10. autenticación

*Los marcos API adecuados también necesitan:*

11. Muy buena cobertura de prueba de su código.
12. Rendimiento decente
13. Documentación
14. Una comunidad activa para avanzar y apoyar el marco.

*Si toma estos factores, en este momento solo hay dos marcos de API que vale la pena usar (depende de las vistas del usuario), django-tastypie y django-rest-framework.*

**¿Cuál es mejor? django-tastypie o django-rest-framework?**

*Yo digo que son iguales. Simplemente no te puedes equivocar con ninguno de los dos. Los autores y las comunidades detrás de ambos están activos, el código es sólido y probado. Y aquí están mis pensamientos específicos sobre ambos:*

Tastypie:

**Ventajas:**

- Fácil de comenzar y proporciona funcionalidades básicas OOB (listas para usar)
- La mayoría de las veces no tratará con conceptos avanzados de Django como CBV, formularios, etc.
- ¡Código más legible y menos magia!
- Si tus modelos no son ORM, hazlo.

**Desventajas:**

- No sigue estrictamente el Django idiomático (las filosofías de Python y Django son muy diferentes)
- Probablemente sea un poco difícil personalizar las API una vez que vayas a lo grande
- No O-Auth

DRF:

**Ventajas:**

- Siga idiomatic django. (Si conoces el django de adentro hacia afuera, y te sientes muy cómodo con el VBC, las formas, etc. sin ninguna duda).
- Proporciona funcionalidad REST fuera de la caja utilizando ModelViewSets. Al mismo tiempo, proporciona un mayor control para la personalización utilizando CustomSerializer, APIView, GenericViews, etc.
- Mejor autenticacion.
- Más fácil escribir las clases de permisos personalizados.
- Trabaje muy bien y, lo que es más importante, muy fácil para que funcione con bibliotecas de terceros y OAuth.
- Vale la pena mencionar a DJANGO-REST-AUTH BIBLIOTECA para Autenticación / Autentificación Social / Registro.

**Desventajas:**

- Si no conoces muy bien a Django, no vayas por esto.
- ¡Mágico! Algún tiempo muy difícil de entender la magia.
- Porque ha sido escrito sobre el CBV de django que a su vez es bastante complejo en su naturaleza.
- Tiene curva de aprendizaje empinada.

## Empezando con Tastypie

Tastypie es una aplicación reutilizable (es decir, se basa solo en su propio código y se centra en proporcionar solo una API de estilo REST) y es adecuada para proporcionar una API a cualquier aplicación sin tener que modificar las fuentes de esa aplicación.

No todas las necesidades son iguales, por lo que Tastypie se esfuerza por proporcionar muchos ganchos para anular o extender su funcionamiento.

> A modo de ejemplo, agregaremos una API a una aplicación de blog simple.

*La única configuración obligatoria es agregar '* ***tastypie*** *' a tu* ***INSTALLED_APPS*** *. Esto no es estrictamente necesario, ya que Tastypie tiene solo dos modelos no necesarios, pero puede facilitar el uso.*

**Aquí está 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()
```

**Creando Recursos**

La arquitectura de estilo REST habla de recursos, por lo que, como es lógico, la integración con Tastypie implica la creación de *clases de recursos* . Para nuestra aplicación simple, crearemos un archivo para estos en **myapp / api.py** , aunque pueden vivir en cualquier parte de su aplicación:

# 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'
```

En REST todo es un recurso, significa que los objetos se describen como recursos. Entonces, para crear una API REST de Tastypie se requiere crear una clase de recurso. Para crear un recurso, necesitamos subclase de la clase ModelResource. Esta clase EntryResource verificará

todos los campos no relacionales en el modelo de Entrada y creará sus propios campos de Api. *Esto funciona igual que el ModelForm de Django Forms.*

Conectando el **recurso:** Afrer creando el recurso, necesitamos informarle a Django que un recurso se crea al enganchar nuestro EntryResource a la 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)),
]
```

Ahora si revisamos **localhost: 8000 / api / entry /** obtendremos la respuesta de la API.

*(El nombre de **recurso** dentro de la clase Meta es opcional. Si no se proporciona, se genera automáticamente a partir del nombre de clase, eliminando cualquier instancia de Recurso y **bajando** la cadena. Por **lo** tanto, **EntryResource se convertiría en una simple entrada** . Por eso, en la URL que especificamos **../entry /** )*

### Autorización Tastypie

Hasta ahora hemos intentado obtener solicitud. *(Si necesita probar otra solicitud HTTP, use algunas herramientas de prueba de API como curl o cartero)*

Si intenta enviar un POST / PUT / DELETE al recurso, se encontrará con errores "401 no autorizados". Por seguridad, Tastypie se envía con la clase de autorización establecida en ReadOnlyAuthorization. Esto hace que sea seguro exponer en la web, pero nos impide hacer POST / PUT / DELETE. Vamos a habilitar esos.

# 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()
```

Lea Codificando con Tastypie en línea: https://riptutorial.com/es/tastypie/topic/10640/codificando-

con-tastypie

# Creditos

| S. No | Capítulos | Contributors |
|---|---|---|
| 1 | Empezando con tastypie | Community |
| 2 | Codificando con Tastypie | AKHIL MATHEW |