Django Tastypie Swagger Documentation

Release 0.1.3

Kyle Rimkus, Josh Bothun

Aug 03, 2017

Contents

1 Synopsis **1**

2 Usage **3**

2.1 Using `extra_actions` 4

2.2 Detecting required fields 5

2.3 Using plural names for resources 5

3 License **7**

4 About Concentric Sky **9**

CHAPTER 1

Synopsis

django-tastypie-swagger is a small adapter library to construct Swagger documentation from Tastypie resources.

This package provides two things:

1. An embedded instance of Swagger UI to point a URL to.
2. Automatic Resource Listing and API Declaration generation that is consumed by #1

CHAPTER 2

Usage

Install package:

```
pip install django-tastypie-swagger
```

Add to INSTALLED_APPS:

```
INSTALLED_APPS = [
    ...

    'tastypie_swagger',

    ...
]
```

Enable documentation for an api endpoint by adding a URL to your urlpatterns.

eg:

```
urlpatterns = patterns('',
    ...

    url(r'api/myapi/doc/',
      include('tastypie_swagger.urls', namespace='myapi_tastypie_swagger'),
      kwargs={
          "tastypie_api_module":"myapp.registration.my_api",
          "namespace":"myapi_tastypie_swagger",
          "version": "0.1"}
    ),

    ...
)
```

- The `namespace` is repeated on purpose to go around some limitations and should be unique amongst the other urls you have defined.

- The `tastypie_api_module` is either your Tastypie api instance or a string containing the full path to your Tastypie api instance.

To declare more than one endpoint, repeat the above URL definition and change the namespace.

Swagger documentation will be served up at the URL(s) you configured.

Using `extra_actions`

While most **ModelResource** based endpoints are good *as-is* there are times when adding additional functionality (like search) is required. In Tastypie the recommended way do to this is by overriding the `prepend_urls` function and returning a list of urls that describe additional endpoints. How do you make the schema map represent these endpoints so they are properly documented?

Add an attribute to the `Meta` class inside your **ModelResource** class called `extra_actions`. Following the Tastypie search example, here is how `extra_actions` should be defined:

```
class Meta:
    ...
    extra_actions = [
        {
            "name": "search",
            "http_method": "GET",
            "resource_type": "list",
            "description": "Search endpoint",
            "fields": {
                "q": {
                    "type": "string",
                    "required": True,
                    "description": "Search query terms"
                }
            }
        }
    ]
```

`extra_actions` is a list of dictionary objects that define extra endpoints that are unavailable to introspection.

Important: `extra_actions` feeds directly into the schema **for swagger**. It does not alter the Tastypie schema listing Tastypie provides.

Top level keys and meaning in the `extra_actions` dictionary:

- `name`: **Required**. Nickname of the resource.
- `http_method`: Defaults to `"GET"`. HTTP method allowed here as a string. Will be uppercased on output.
- `resource_type`: If this is declared as `"list"` then the endpoint **will not** include a `{id}` parameter in the uri or in the parameters list. This is applicable to endpoints such as the above example that filter or perform actions across many items. If `resource_type` is omitted and the `http_method` is `"GET"` then the endpoint will default to `"view"` and include a `{id}` parameter in the uri and parameter list.
- `summary`: Description of this endpoint.
- `fields`: **Optional** Dictionary of parameters this endpoint accepts.
- `responseClass`: **Optional** Should match the id of the corresponding model

- `model`: **Optional** Dictionary of parameters describing the model. `type` and `description` are the two required elements. `required` is optional and is False by default.

Example

```
'responseClass': 'blog-class',
'model':{
        'id': "blog-class",
        "properties": {
                "title": {
                        "type": "string",
                        "description": "This is the title. Max 80 caracters."
                        "required":True
                },
                "content": {
                        "type": "text",
                        "description": "Content of your post. Text up to 65000+"
                        "required":True
                }
        }
}
```

Field dictionaries are declared in a `{ "name": { [options dict] }` style. This is done for compatibility reasons with older versions of django-tastypie-swagger.

Warning: The structure of `fields` will likely change in future versions if Joshua Kehn continues committing.

Available keys and meaning for the `fields` dictionary:

- `type`: Defaults to `"string"`. Parameter type.
- `required`: Defaults to `False`.
- `description`: Defaults to `""` (empty string). Description of this parameter.

Detecting required fields

Tastypie 0.9.11 **ModelResource** fields do not respect the *blank* attribute on django model fields, which this library depends on to determine if a field is required or not.

You can use this ModelResource subclass as a workaround to this issue.

Using plural names for resources

It is possible to define a *plural* name for a resource, using this attribute in the class' `Meta`:

class ShoeResource(Resource): size = ... brand = ...

class Meta: resource_name = 'shoe' resource_name_plural = 'shoes'

CHAPTER 3

License

CHAPTER 4

About Concentric Sky

For nearly a decade, Concentric Sky has been building technology solutions that impact people everywhere. We work in the mobile, enterprise and web application spaces. Our team, based in Eugene Oregon, loves to solve complex problems. Concentric Sky believes in contributing back to our community and one of the ways we do that is by open sourcing our code on GitHub. Contact Concentric Sky at hello@concentricsky.com.