Django Shared Schema Tenants Documentation

Release 0.1.0

Hugo Bessa

Oct 31, 2017

Contents

1 Django Shared Schema Tenants **3**

2 Installation **5**

3 Usage **7**
- 3.1 Instalation on Django . . . 7
- 3.2 Create new tenants . . . 7
- 3.3 Turning existing models into a Tenant Model . . . 7
- 3.4 Selecting tenant on requests . . . 8
- 3.5 Accessing current tenant . . . 9
- 3.6 Configuration options . . . 9

4 Contributing **11**
- 4.1 Types of Contributions . . . 11
- 4.2 Get Started! . . . 12
- 4.3 Pull Request Guidelines . . . 13
- 4.4 Tips . . . 13

5 Credits **15**
- 5.1 Development Lead . . . 15
- 5.2 Contributors . . . 15

6 History **17**
- 6.1 0.1.0 (2017-08-10) . . . 17

Contents:

CHAPTER 1

Django Shared Schema Tenants

A lib to help in the creation applications with shared schema without suffering

CHAPTER 2

Installation

At the command line:

```
$ easy_install django-shared-schema-tenants
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-shared-schema-tenants
$ pip install django-shared-schema-tenants
```

To use Django Shared Schema Tenants in a project, add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'shared_schema_tenants.apps.SharedSchemaTenantsConfig',
    ...
)
```

You also have to add TenantMiddleware to django *MIDDLEWARES*:

```
MIDDLEWARES = [
    # ...
    'shared_schema_tenants.middleware.TenantMiddleware',
    # ...
]
```

CHAPTER 3

Usage

Instalation on Django

To use Django Shared Schema Tenants in a project, add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'shared_schema_tenants.apps.SharedSchemaTenantsConfig',
    ...
)
```

Add Django Shared Schema Tenants's URL patterns:

```
from shared_schema_tenants import urls as shared_schema_tenants_urls


urlpatterns = [
    ...
    url(r'^', include(shared_schema_tenants_urls)),
    ...
]
```

Create new tenants

Run `python manage.py createtenant` to create you first tenant

Turning existing models into a Tenant Model

The models become tenant aware through inheritance. You just have to make your model inherit from `SingleTenantModelMixin` or `MultipleTenantsModelMixin` and you're set.

```
from shared_schema_tenants.mixins import SingleTenantModelMixin, MultipleTenantsModel

class MyModelA(SingleTenantModelMixin)
    field1 = models.CharField(max_length=100)
    field2 = models.IntegerField()

# ...

# 'default' tenant selected
instance = MyModelA(field1='test default tenant', field2=0)
instance.save()

# ...

# 'other' tenant selected
instance = MyModelA(field1='test other tenant', field2=1)
instance.save()

print(MyModel.objects.filter(field1__icontains="test"))
# prints only the instance with 'test other tenant' in field1

Obs.: For Django 1.8 and 1.9 you have to access the data by the active tenant through␣
↪:python:`MyModel.tenant_objects.all()` due to a `Django bug that was fixes in␣
↪version 1.10 <https://code.djangoproject.com/ticket/14891>`_
```

Selecting tenant on requests

Tenant site

If you access the site from a domain registered to a tenant, that tenant is automatically selected.

Tenant-Slug HTTP header

If the header `Tenant-Slug` could be found in the request, the tenant with that slug is automatically selected.

Forcing tenant selection

You can force tenant selection using set_tenant method.

```
from shared_schema_tenants.helpers import set_current_tenant

from .models import MyModel

def my_function():
    set_current_tenant('default')

    return MyModel.objects.all() # return only the models with tenant__slug='default'


Obs.: For Django 1.8 and 1.9 you have to access the data by the active tenant through␣
↪:python:`MyModel.tenant_objects.all()` due to a `Django bug that was fixes in␣
↪version 1.10 <https://code.djangoproject.com/ticket/14891>`_
```

Accessing current tenant

From Request

You can access the current tenant from the request.

```
def my_view(request):
    current_tenant = request.tenant
    # ...
```

From `get_current_tenant` helper

```
from shared_schema_tenants.helpers import get_current_tenant

def my_view(request):
    current_tenant = get_current_tenant()
    # ...
```

The models that inherit from `SingleTenantModelMixin` or `MultipleTenantsModelMixin` are also tenant aware. If you retrieve a collection from database with a tenant context in your request, your collection will already be filtered by that tenant.

Configuration options

To configure how Django Shared Schema Tenants works you can set a bunch of options in the SHARED_SCHEMA_TENANTS dictionary in django settings

SERIALIZERS

It's a dict where you can replace the serializers to be used in Django Shared Schema Tenants REST API endpoints. default value:

DEFAULT_TENANT_SLUG

In here you can define you default tenant (tenant to be use in case the middleware can't retrieve the tenant from the request)

default value: `'default'`

TENANT_SETTINGS_FIELDS

In here you define the fields in tenant setting. Every field is a dict and must have the followiing format:

The available types are `'number'`, `'string'`, `'boolean'`, `'object'` and `'list'`.

default value: `{ }`

TENANT_SETTINGS_FIELDS

In here you define the fields in tenant extra_data. This field is a dict and must have the following format:

The available types are `'number'`, `'string'`, `'boolean'`, `'object'` and `'list'`.

default value: { }

DEFAULT_SITE_DOMAIN

In here you define your default site domain.

default value: `'localhost'`

TENANT_HTTP_HEADER

In here you can defined which http header we should use to extract the tenant slug

default value: `'Tenant-Slug'`

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/hugobessa/django-shared-schema-tenants/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

Django Shared Schema Tenants could always use more documentation, whether as part of the official Django Shared Schema Tenants docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/hugobessa/django-shared-schema-tenants/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *django-shared-schema-tenants* for local development.

1. Fork the *django-shared-schema-tenants* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-shared-schema-tenants.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-shared-schema-tenants
$ cd django-shared-schema-tenants/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 shared_schema_tenants tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.5 and 3.6 and for PyPy. Check https://travis-ci.org/hugobessa/django-shared-schema-tenants/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ python -m unittest tests.test_shared_schema_tenants
```

CHAPTER 5

Credits

Development Lead

- Hugo Bessa <hugo@bessa.me>

Contributors

None yet. Why not be the first?

CHAPTER 6

History

0.1.0 (2017-08-10)

- First release on PyPI.