django-spaghetti-and-meatballs Documentation

Release 0.2.0

Samuel Spencer

February 29, 2016

Contents

Its a spicy meatball for serving up fresh hot entity-relationship diagrams straight from your django models.

Food puns aside, django-spaghetti-and-meatballs is a django app that extracts models and documentation from a projects `models.py` to build rich, live, interactive entity-relationship diagrams.

At the moment, it mines details from models:

- docstrings, so documentation that the same documentation can be reused in the modeling
- fields, including the fields datatype, help_text and cardinality
- foreign and many-to-many relationships (at the moment only many-to-many relationships get special styling, but one-to-one is coming soon)

And it does this for all models in a project across any django apps you choose - plus you can specify models to exclude as well.

CHAPTER 1

Why did you make this app?

There are a few really big use cases that I saw spaghetti-and-meatballs filling.

Firstly, for projects where generating user-facing diagrammatic representation of models helps with their understanding. Being able to generate this from the code means that it is *never* out of sync with the actual design.

Secondly, being able to easily and graphically see the relationships between models shows where some models are over- or under-connected. It also helps show where new connections can be useful, especially during development.

Lastly, it has been helpful during early stages of development as a prototyping and documentation tool. Getting a simple and live view of all of the models and their documentation has been a godsend on a recent project when working with non-developers. My preferred method of design and documentation is writing well documented code, which can be daunting for non-developers. By using django models as an input to making the entity-relationship diagrams it means that diagrams of the structure of a project and the code and documentation never get out of sync. It also meant that I (and others) could easily spot gaps in documentation. While some code is self-documenting, hovering over a model and still being confused about the existence of a field shows a definite need for explanation.

Contents:

1.1 Installing and configuring

1.1.1 Installing spaghetti

1. Install some spaghetti:

```
pip install django-spaghetti-and-meatballs
```

2. Add "`django_spaghetti`" to your `INSTALLED_APPS` setting like this:

```
INSTALLED_APPS = [
  ...
  'django_spaghetti',
]
```

3. Add a plate of spaghetti in your `urls.py` like so:

```
urlpatterns += patterns('',
  url(r'^plate/', include('django_spaghetti.urls')),
)
```

1.1.2 Configuring meatballs

`django-spaghetti-and-meatballs` takes a few options set in the `SPAGHETTI_SAUCE` variable from your projects `settings.py` file that make it *extra spicy*:

```
SPAGHETTI_SAUCE = {
  'apps':['auth','polls'],
  'show_fields':False,
  'exclude':{'auth':['user']},
  'show_proxy':True,
}
```

In the above dictionary, the following settings are used:

- `apps` is a list of apps you want to show in the graph. If its *not* in here it *won't be seen*.
- `show_fields` is a boolean that states if the field names should be shown in the graph or just in the however over. For small graphs, you can set this to *True* to show fields as well, but as you get more models it gets messier.
- `exclude` is a dictionary where each key is an `app_label` and the items for that key are model names to hide in the graph.
- `show_proxy` is boolean, if truthy proxy models will be shown and linked to their main model, otherwise they will be hidden. By default this is false.

If its not working as expected make sure your app labels and model names are all **lower case**.

1.2 Customising the output

`Django-spaghetti-and-meatballs` makes heavy use of the vis.js network library, so most customisation is either based on django template modification and overrides or changes to the vis.js settings. Knowing how to use both will make customisation a breeze, but below are some simple recipes to get you started.

1.2.1 Serving up a new template

The default template is a frugal dish that serves spaghetti with no extras. To serve the spaghetti in a fancier setting, just override the `django_spaghetti/plate.html` template in your projects `templates` directory.

You'll probably want your plate to look similar to that served by django_spaghetti, which can be viewed on github

However, the important things when making a plate are:

- make sure you import `vis.js` and `vis.css` *before* the script that creates the graph.
- make your your script loads the `meatballs` (nodes) and `spaghetti` (edges) with the safe django template filter

1.2.2 Changing how vis.js is loaded

By default `Django-spaghetti-and-meatballs` loads `vis.js` from the Cloudflare CDN, but this might not be appropriate for your project and may want to load it locally.

You can do this by following the above instructions and just changing the `extra_scripts` block.

1.2.3 Changing how vis.js lays out the graph

Dealing `vis.js` network customisation is beyond the scope of this project, but vis.js has comprehensive documentation of their network library available online. In the `plate.html` template you can make changes to the settings when you setup the graph. By default `django-spaghetti-and-meatballs` uses a hierarchical layout, but any setup should work. For example, to turn off hierarchical layout you can use the settings:

```
"layout": {
    hierarchical: false,
},
```

1.2.4 New flavours of meatballs

By default, `django-spaghetti-and-meatballs` shows a model with all its fields and the documentation from the docstring in a hover over pop-up pane. The layout for this box is included in the `django_spaghetti/meatball.html` template. To change how models are shown on hover, just override the `django_spaghetti/meatball.html` template in your projects `templates` directory.

You'll probably want your meatball to taste similar to that served by django_spaghetti, which can be viewed on github

For example, to show just the models name, number of fields and its documentation your template would look like this:

```
<div style="max-width:350px;white-space: normal;">
    <div style="border-bottom:1px solid black">
        Model: {{ model.model }}<br>
        # of fields: {{ fields|length }}
    </div>
    <tt style="white-space:pre-line">
        {{ model.doc }}
    </tt>
</div>
```

1.3 Contribute to django-spaghetti

They say that "too many cooks spoilt the broth", but thats not the case with open-source software. `django-spaghetti-and-meatballs` is a small and simple library that welcomes contributions from anyone. Just fork the project, and make a pull request.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search