django-oscar-support Documentation

Release 0.1.0

Sebastian Vetter

February 24, 2014

Contents

Contents:

CHAPTER 1

Getting Started

Warning: *django-oscar-support* has been tested with Oscar 0.6 `master` (due to it's pre-release state) and only works with commit 4aea545de3 or later. Due to major changes in the stock record architecture and the customer account templates earlier version will not work with the instructions below.

1.1 Installation

The following few steps will walk you through setting up the support system within your existing Oscar project. If you don't have Oscar setup already, take a look at the documentation for Oscar on how to setup your shop.

Start off by installing the required package either from PyPI:

```
pip install django-oscar-support
```

or install the latest version from github using:

```
pip install git+https://github.com/tangentlabs/django-oscar-support.git
```

To make your Django project aware of the new package add it to your `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    ...
    'oscar_support',
    ...
)
```

and while you are in your `settings.py` (or whatever your settings file is called) add the following line to enable sensible default settings for *django-oscar-support*:

```
from oscar_support.defaults import *
```

Now run the migrations to make sure that all the required tables are created in the database. In most cases this will be something like but might be different depending on your project setup:

```
./manage.py migrate oscar_support
```

1.2 Integrating The UI

The support system hooks into two different places of Oscar: the profile of a customer and the dashboard for support agents. *django-oscar-support* provides a default UI for these two entry points but they need to be hooked into your

Oscar project. Let's start by adding the URL patterns to your current project by adding the following line into your `urls.py` file:

```
import oscar_support.urls

urlpatterns = patterns(''
    ...
    url(r'^', include(shop.urls)),  # Oscar's URL patterns
    url(r'^', include(oscar_support.urls)),
    ...
)
```

This will provide all the URLs required to interact with the support system. The next step is to make the dashboard UI available in the navigation using Oscar's `OSCAR_DASHBOARD_NAVIGATION` settings:

```
# import this if you want to make the label below translatable
from django.utils.translation import ugettext_lazy as _

....

OSCAR_DASHBOARD_NAVIGATION = OSCAR_DASHBOARD_NAVIGATION + [
    {
        'label': _("Support"),
        'icon': 'icon-comments',
        'url_name': 'support-dashboard:ticket-list',
    }
]
```

Note: Make sure that you aren't using `OSCAR_DASHBOARD_NAVIGATION +=` because this will cause duplicate entries due to a bug in Django that will only be fixed in version 1.6+

All that is left to do now is integrating the customer-facing part of the support system. The templates that come with *django-oscar-support* extend Oscar's customer account templates and should pick up the styles that are defined there following Oscar's template customisation process. If you are not sure how this works, take a look at How to customise templates in the Oscar docs.

Let's look at a simple way to add the customer-facing side of the support system into the customer's profile page. We can achieve that by integrating a link to the *ticket overview* page into the customer profile navigation. The first thing to do is to create `customer/baseaccountpage.html` in your template directory and add the following content:

```
{% extends "oscar/customer/baseaccountpage.html" %}
{% load i18n %}
{% load url from future %}

{% block standard_tabs %}
    {{ block.super }}
    <li>
        <a href="{% url 'support:customer-ticket-list' %}">{% trans "Support" %}</a>
    </li>
{% endblock %}
```

This adds a *Support* navigation element to the end of the profile navigation. If you are using a more customised set of templates it might be better to take a look at the templates in `oscar_support/templates` and integrate extended templates into your own template directory.

1.3 Configuring The RESTful API

django-oscar-support provides and uses a RESTful API for serveral resources such as the User. We use django-rest-framework (DRF) to provide the API URLs and render the data into a RESTful format. This also provides authentication mechanisms that can easily configured in the the settings file. We provide sensible defaults for this in `oscar_support.defaults` that you probably added to your settings already if you followed the instructions above.

The default configuration for DRF restricts API usage to session-based authentication and only provides JSON as serialisation format. We also disable the browsable API provided by DRF to avoid accidentally exposing it in production. You can change all of these settings by overriding the `REST_FRAMEWORK` dictionary in your settings file. For more info on settings available for DRF take a look at their excellent documentation.

Note: This is just the beginning, there's more docs to come.

CHAPTER 2

API Reference

Coming sooon....

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*