django-radio Documentation

Release 1.1.1

Iago Veloso Abalo

February 08, 2015

Contents

CHAPTER 1

Overview

RadioCo is a broadcasting radio recording scheduling system. RadioCo has been intended to provide a solution for a wide range of broadcast projects, from community to public and commercial stations.

Here are a few of the key features:

- designed to work with any web browser
- drag and drop scheduling calendar interface
- live shows can be recorded and published automatically
- complete authentication system (user accounts, groups, permissions)
- ...and much more

Note: This software is not for live streaming

CHAPTER 2

Release Notes

This document refers to version 1.1.1

CHAPTER 3

Table of contents

3.1 Quick start guide

The pages in this section of the documentation are designed to help you get started quickly and show how easy it is to work with RadioCo

The guides follow a logical progession and build on each other, so it's recommended to work through them in the order presented here.

3.1.1 Installing web application

This tutorial is written for Python 2.7 and Ubuntu 12.04 or later.

Installing on Ubuntu

If you're using Ubuntu (tested with 14.04), the following should get you started:

```
sudo apt-get install git-core python-dev python-pip python-virtualenv
```

Next, download the project and cd into it:

```
git clone https://github.com/iago1460/radioco
cd radioco
```

Create and switch to the virtualenv at the command line by typing:

```
virtualenv .
source bin/activate
```

Install the requirements:

```
pip install -r requirements.txt
```

Setup the database:

```
python manage.py migrate
```

Create a superuser:

```
python manage.py createsuperuser
```

Testing

Let's verify your installation

```
python manage.py runserver
```

Now that the server's running, visit http://127.0.0.1:8000/

> **Warning:** Don't use this server in anything resembling a production environment.

3.1.2 Installing recorder program

This tutorial is written for Python 2.7 and Ubuntu 12.04 or later.

Installing on Ubuntu

We'll get started by setting up our environment.

```
sudo apt-get install python-dev python-pip python-virtualenv git-core alsa-utils vorbis-tools
```

Next, download the project and create the virual environment:

```
git clone https://github.com/iago1460/django-radio-recorder.git
cd django-radio-recorder
```

Create and activate a virtual env:

```
virtualenv .
source bin/activate
```

Install the requirements:

```
pip install -r requirements.txt
```

Using your favorite text editor, configure the `settings.ini` file

Launch the program

```
python main.py
```

3.2 Reference

Technical reference material

3.2.1 Configuration

RadioCo has a number of settings to configure its behaviour.

Application Setup

These settings should be available in your `settings.py`.

USERNAME_RADIOCO_RECORDER

This specifies who is the user of the recorder program:

```
USERNAME_RADIOCO_RECORDER = 'RadioCo_Recorder'
```

Note: It's a good idea change this value for security reasons.

PROGRAMME_LANGUAGES

New in version 1.1

Default: A tuple of all available languages.

This specifies which languages are available for language selection in your programmes:

```
gettext_noop = lambda s: s

PROGRAMME_LANGUAGES = (
    ('es', gettext_noop('Spanish')),
    ('en', gettext_noop('English')),
)
```

You can see the current list of translated languages by looking in django/conf/global_settings.py (or view the online source).

Recorder Program Setup

The settings should be available in your settings.ini.

Communication Setup

In your Admin interface go to **Podcast Configuration**, copy **Recorder token** and put into the Recorder Program settings:

```
token:8fdde6d703c05773084ea83e5ec2da62637666a0 #for example
```

Modify the **url** in your Recorder Program settings:

```
url:http://yourdomain:80/api/1/
```

3.3 Development & community

RadioCo is an open-source project, and relies on its community of users to keep getting better.

You don't need to be an expert developer to make a valuable contribution - all you need is a little knowledge of the system, and a willingness to follow the contribution guidelines.

Remember that contributions to the documentation are highly prized, and key to the success of the project. Any time and effort you are willing to contribute is greatly appreciated!

3.3.1 Branch policy

- **master**: this is the current stable release, the version released on PyPI.
- **develop**: this branch always reflects a state with the latest delivered development changes for the next release.
- **feature branches**: these are used to develop new features.

3.3.2 Contributing Documentation

Perhaps considered "boring" by hard-core coders, documentation is sometimes even more important than code! This is what brings fresh blood to a project, and serves as a reference for old timers. On top of this, documentation is the one area where less technical people can help most - you just need to write semi-decent English. People need to understand you. We don't care about style or correctness.

Documentation should be:

- written using valid Sphinx/restructuredText syntax (see below for specifics) and the file extension should be `.rst`
- written in English (we have standardised on British spellings)
- accessible - you should assume the reader to be moderately familiar with Python and Django, but not anything else. Link to documentation of libraries you use, for example, even if they are "obvious" to you

Merging documentation is pretty fast and painless.

Except for the tiniest of change, we recommend that you test them before submitting.

Documentation markup

Sections

We use Python documentation conventions for section marking:

- # with overline, for parts
- * with overline, for chapters
- =, for sections
- -, for subsections
- ^, for subsubsections
- ", for paragraphs

Inline markup

- **use backticks - ``settings.py`` - for:**
  - literals
  - filenames
  - names of fields and other items in the Admin interface:
- **use emphasis - *Home* around:**
  - the names of available options in the Admin

- values in or of fields

References

Use absolute links to other documentation pages - `:doc:'/how_to/toolbar'` - rather than relative links - `:doc:'/../toolbar'`. This makes it easier to run search-and-replaces when items are moved in the structure.

3.4 Release notes & upgrade information

Some versions of RadioCo present more complex upgrade paths than others, and some **require** you to take action. It is strongly recommended to read the release notes carefully when upgrading.

It goes without saying that you should **backup your database** before embarking on any process that makes changes to your database.

3.4.1 1.1 release notes

What's new in 1.1

- Numerous updates to the documentation
- Updates to facilitate the starting-up (radioco_recorder_user and a empty shedule_board are self-created)
- Added custom languages to programmes (by default, all django languages)

How this affects you

If you're starting with a new installation, you don't need to worry about this. Don't even bother reading this section; it's for upgraders.

Activate your virtualenv and do the following in your project main directory:

```
pip install django-radio==1.1
python manage.py migrate
```

3.4.2 1.1.1 release notes

What's new in 1.1.1

- Added CKEditor to programme's synopsis, episode's summary and user's biography
- Added default settings to the project
- Added customisable footer

How this affects you

If you're starting with a new installation, you don't need to worry about this. Don't even bother reading this section; it's for upgraders.

Activate your virtualenv and do the following in your project main directory:

```
pip install django-radio==1.1.1
python manage.py collectstatic
python manage.py migrate
```

In your settings.py import default settings:

```
from radio.settings_base import *
```

3.5 Indices and tables

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