# dinnertime Documentation

***Release 1.0***

**myauthor**

February 06, 2013

# CONTENTS

Contents:

CHAPTER

ONE

# INSTALLATION

## 1.1 Pre-Requisites

- setuptools
- virtualenv

To install all of these system dependencies on a Debian-based system, run:

```
sudo apt-get install python-setuptools
sudo easy_install virtualenv
```

## 1.2 Creating the Virtual Environment

First, create a clean base environment using virtualenv:

```
virtualenv dinnertime
cd dinnertime
source bin/activate
```

If you are using virtualenvwrapper (recommended) then:

```
mkvirtualenv dinnertime
workon dinnertime
```

## 1.3 Installing the Project

Install the requirements and the project source:

```
sudo apt-get install python-memcache
python manage.py startapp accounts
    cd path/to/your/dinnertime/repository
pip install -r requirements.pip
pip install -e .
```

## 1.4 Customize the Project

Take a look at the settings file in dinnertime/settings/base.py

## 1.5 Configuring a Local Environment

If you're just checking the project out locally, you can copy some example configuration files to get started quickly:

```
cp dinnertime/settings/local.py.example dinnertime/settings/local.py
manage.py syncdb --migrate
```

## 1.6 Populate the Recipe Database

To download recipes from the yummly API add the correct key to yummly/management/commands/yummly_meta.py and run the following command:

```
python manage.py yummly_meta <model>
```

Make sure you set <model> to one of "Ingredient, Recipe, Course, Allergy or Diet"

## 1.7 Building Documentation

Documentation is available in `docs` and can be built into a number of formats using Sphinx. To get started:

```
pip install Sphinx
cd docs
make html
```

This creates the documentation in HTML format at `docs/_build/html`.

CHAPTER

TWO

# ENVIRONMENTS

When deploying to multiple environments (development, staging, production, etc.), you'll likely want to deploy different configurations. Each environment/configuration should have its own file in `dinnertime/settings` and inherit from `dinnertime.settings.base`. A `dev` environment is provided as an example.

By default, `manage.py` and `wsgi.py` will use `dinnertime.settings.local` if no settings module has been defined. To override this, use the standard Django constructs (setting the `DJANGO_SETTINGS_MODULE` environment variable or passing in `--settings=dinnertime.settings.<env>`). Alternatively, you can symlink your environment's settings to `dinnertime/settings/local.py`.

You may want to have different `wsgi.py` and `urls.py` files for different environments as well. If so, simply follow the directory structure laid out by `dinnertime/settings`, for example:

```
wsgi/
  __init__.py
  base.py
  dev.py
  ...
```

The settings files have examples of how to point Django to these specific environments.

CHAPTER

THREE

# DEPLOYMENT

## 3.1 Staging/Development

Fabric is used to allow developers to easily push changes to a previously setup development/staging environment. To get started, install Fabric by running the following command from within your virtual environment:

```
pip install fabric==1.4
```

So see a list of available commands, run the following command from within your project directory:

```
fab -l
```

Some common commands:

```
fab restart       # Restart the web server.
fab update        # Just update the repository.
fab push deploy   # Push, then fully deploy.
```

From the within the project directory, you can just run `fab [command]`. If you want to run fabric outside of the directory, use:

```
fab --fabfile /path/to/project/fabfile.py [command]
```

CHAPTER

FOUR

# INDICES AND TABLES

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