Servee Documentation

Release 0.6.0a1.dev22

Issac Kelly

October 06, 2015

Contents

Servee is a content-editing system for the front-end of your django based website.

Information is available by emailing issac@servee.com or visiting irc.freenode.net#servee

Contents:

CHAPTER 1

Introduction

Servee is a subclass of django.contrib.admin built for front-end editing. It keeps a separate registry from django.contrib.admin, and the tools are specifically built with non-technical users in mind. Many aspects of servee are intentionally basic to keep dependencies down so it can be used across a very wide variety of projects.

It's also meant to be tightly coupled with wysiwyg components for content editing. Right now TinyMCE is supported in Open Source, and we have a Redactor build as well.

Installation

First you should put servee in your environment:

```
pip install django-servee
pip install django-uni-form
```

or download and:

```
./setup.py develop
```

Then add servee to your installed apps.

At a minimum, you want to install *servee.frontendadmin*. This is the admin site. You probably also want our wysiwyg tools, and this particular branch runs Redactor. This converts your textareas to wysiwyg areas, and that's awesome (sometimes):

```
INSTALLED_APPS += [
    "servee.frontendadmin",
    "servee.wysiwyg",
    "uni_form" # Required
]
```

Then syncdb. Servee assumes that you're using either contrib.staticfiles or django-staticfiles (>=1.1) Make sure that you collectstatic in production.

It's important to add servee urls, and you probably want to use autodiscover:

```
from servee import frontendadmin
frontendadmin.autodiscover()
# ...
url(r"^servee/", include(frontendadmin.site.urls)),
```

You probably want several other packages as well:

```
pip install django-servee-redactor
pip install django-servee-uploadify
pip install django-servee-image
pip install django-servee-gallery
```

Add those to INSTALLED_APPS:

```
# before servee.wysiwyg, for static files finding
"servee_redactor",

"servee_document",
"servee_image",
"servee_gallery",
```

CHAPTER 3

Example Project

The Example Project will get you started with a working version of servee. It includes a sample image plugin, and django-flatpage support. Both of these are simple implementations to help you understand how you could make your own ModelInsert plugins or how to convert your own models to be FrontendAdmin editable.

3.1 Running it.

This is a bare django project + flatpages + required settings and installed apps for servee.

To build it, these instructions assume you have virtualenv + virtualenv wrapper installed.

Creating the Project:

```
mkvirtualenv servee
cd ~/Path/to/servee/
python manage.py develop

cd example_project/
python manage.py syncdb
python manage.py runserver
```

Login at http://localhost:8000/

Go to http://localhost:8000/ to edit it.

CHAPTER 4

WYSIWYG

The goal of the wysiwyg components of servee is to best emulate the final version of the site. The included engine is built on tinyMCE, which is already in the source tree. There's also a branch for Redactor.

The wysiwyg tools abstract the function calls of the engine so that it should be fairly simple to swap out the rendering engine if you prefer something like FCKeditor or WYMeditor.

4.1 Basic Tools

- Bold
- Italic
- Underline
- Blockquote
- Link
- Unlink
- Undo
- Redo
- Insert Horizontal Rule
- Edit HTML
- Insert
- Styles

4.2 Insert Dialog

This is how you insert media (pictures, videos, text) or inline representation of a model or models. You can create your own insert dialogs for things like forms, products, blog posts, or user profiles. This is populated through the inserts

CHAPTER 5

Inlines

Inlines are a very powerful tool for putting content inside another piece of content. The most comment use-case is to put a small representation of a Model (like an Image) onto your content area.

Included in the example_project is a very simple image insert + model, based on ModelInsert.

Inserts have three primary views:

- List: A list of available items to be inserted
- Detail: Shows More information about the item, as well as the button for rendering the insert onto the current content block.
- Render: Returns a partial HTML template that gets inserted onto the page at the cursor.

Model Inserts also work off of an idea that you can add one by only putting in the required fields For our Image example, the only required field is the *image = models.ImageField...* field.

We are using uploadify to allow you to add several images to the site at once.

If you don't specify an add_form when creating your ModelInsert class, it will create one for you based on the required fields.

Using Servee with django 1.2 is not straightforward, and won't really be supported, that being said I've made it work, and you can too.

- Django-staticifles is required, and must be configured to pull from directories named *static*
- You must write new templates for anything that begins {% url 'x' ... and replace it with {% url x ...

CHAPTER 6

Indices and tables

- genindex
- modindex
- search