Calloway Documentation

Release 0.4.2

calloway project

February 01, 2012

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION TO CALLOWAY

The underlying framework upon which Calloway resides is Django. It is written in Python and promotes adding functionality in pluggable "Applications" that can be reused in other projects. This isolation of functionality is what Calloway uses to combine other open source projects together to create a cohesive infrastructure.

1.1 Design Principles

Calloway is based on four principles:

- **Integration without dependency** Django pluggable apps should do one thing and only one thing and do it well. For example, a blog app, should not include tagging, categories, authors, and other things that might augment the core functionality of the blog. It should manage blogs and entries and make it easy to write them and deliver them.

  This allows you to build the core infrastructure like building blocks, adding in the functionality that you need in the way you need it.

- **User interface for content managers is important** The Django admin has spoiled us developers. While it is great in many respects, where it is lacking is often overlooked by developers. The content managers, the ones who keep your site up-to-date with the stuff users want, suffer.

  We want to promote and provide methods to improve the user experience for content managers, and promote applications that do so as well.

- **Easy customization through overrides and fallbacks** The methods that Django uses for handling settings and templates nails this idea. Templates are easily overridden one at a time, and it is easy to provide template to fallback on if no other is there.

- **Simple enough for a designer to do it** This is not meant to be derogatory, but a workflow advantage. A designer that is comfortable with the command line should be able to bootstrap a project, pick the building blocks of functionality and start designing the front end.

1.2 Don't Assume. It makes an ass out of you ... and someone else

We've been frustrated attempting to get supposed "pluggable" applications to work together; even our own. We realized that most of the time, it is because the application was written with some assumptions that are either not communicated, or don't work in a different situation.

Calloway aims to be assumption free. But it probably isn't right now. That's because *you* haven't tried it in *your* situation to discover an assumption not yet realized and fixed.

1.3 What Calloway Provides

- **Lists of application "bundles" that are known to work together.** Some of them we've written, most are really good third-party applications from the community. The application bundles are merely shortcuts and not required.

  You can see the list of application bundles at *Application bundles in settings*.

- **Dependency management.** With the applications that it knows about, it can generate a requirements file for `pip` to make installation of the packages easy.
- **Project templating.** The Django `startproject` command is nice for beginners, but is missing many things that you probably need and want. We provide an example project template, but encourage you to customize it to your heart's desire.

1.4 Structure of a Calloway Project

A fundamental Calloway principle is progressive enhancement. By providing good defaults, you only override what you need. You have access to individual parts, as well as a combined whole.

1. You create a project using a project template.
   - A project template is included, but you can modify it to your defaults
   - A script is used to walk you through the creation of the project
2. Your project includes Calloway as a dependency (that is typically defined already in the template project).
3. The `settings.py` in your project brings in Calloway's default settings, and specifies preset application bundles. Other applications or overrides of the defaults are also specified.
4. The `urls.py` in your project brings in Calloway's default routing, and overrides are added as necessary.
5. The template project includes common server configurations, customized by the `create_project` script.

CHAPTER

TWO

INSTALLATION

There are several parts to the Calloway project that are not dependent upon each other, but work great in tandem.

2.1 start_project.py script

The `start_project.py` script is an interactive command-line program for building a Django project from a template project, based on code from Eric Florenzano. The template project isn't anything special except that any placeholders for certain variables are replaced during the project creation.

View the script at https://gist.github.com/444408 or download the script directly from https://gist.github.com/gists/444408/download. Once you download the script, you will need to decompress it.

```
$ curl https://gist.github.com/gists/444408/download | tar -zxv

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
102  2058  102  2058    0     0   7629      0 --:--:-- --:--:-- --:--:--  9942
x gist444408-597f127451c5cd81e599753224b4e9fb133bd3eb/
x gist444408-597f127451c5cd81e599753224b4e9fb133bd3eb/start_project.py
```

You can move the script anywhere you want from here, however it is handy to rename the folder containing the script and download the *Project Template* into it.

```
$ mv gist444408-597f127451c5cd81e599753224b4e9fb133bd3eb project_template
$ cd project_template
```

2.2 Project Template

The example project template is meant to be a starting point. It is likely that you will have several templates for different types of sites or deployments. Download the `project_tmpl` code from github and decompress it. Alternatively, you can clone it using `git` if you wish to keep up-to-date with any changes we make to it.

```
$ curl -L  https://github.com/callowayproject/project_tmpl/tarball/master | tar -zvx

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 61440  100 61440    0     0   254k      0 --:--:-- --:--:-- --:--:--  254k

x callowayproject-project_tmpl-0dcbf69/
```

```
x callowayproject-project_tmpl-0dcbf69/.gitignore
x callowayproject-project_tmpl-0dcbf69/__init__.py
...
```

Make any changes to this template you wish. It's *your* project template now.

2.3 Calloway Application

The Calloway application is just like any other Django application. You can install it in any typical way, using `pip` or `easy_install` although the suggested method is to include Calloway in the requirements file in your project.

The example project template includes `calloway` as a requirement in `setup/requirements.txt`. When you execute `pip install -r setup/requirements.txt`, `pip` will install all other requirements (Django, for example) as well as Calloway. You can add the specific version of Calloway to your requirements file (e.g. `calloway==0.2`) if you want. We didn't want to tie the project template too closely with the Calloway Application version as the two parts will change at different rates.

CHAPTER

THREE

GETTING STARTED

1. Download the *start_project.py script*

```
$ curl http://gist.github.com/gists/444408/download | tar -zxv
$ mv gist444408-597f127451c5cd81e599753224b4e9fb133bd3eb project_template
$ cd project_template
```

2. Download the example *Project Template*

```
$ curl -L  https://github.com/callowayproject/project_tmpl/tarball/master | tar -zvx
$ mv callowayproject-project_tmpl-0dcbf69 project_tmpl
```

3. Execute:

```
$ python start_project.py
```

and answer the questions.

```
$ python start_project.py
Project name: sampleproject
Administrator e-mail address: admin@sampleproject.com
Destination directory (currently at /home/demo/project_template):
Project template directory [/home/demo/project_tmpl]: ./project_tmpl
Virtual environment name (e.g. sampleproject):
Copying /home/demo/quickstart/project_tmpl/__init__.py to /home/demo/quickstart/sampleproject/__
Copying /home/demo/quickstart/project_tmpl/manage.py to /home/demo/quickstart/sampleproject/mana
Copying /home/demo/quickstart/project_tmpl/menu.py to /home/demo/quickstart/sampleproject/menu.p
Copying /home/demo/quickstart/project_tmpl/settings.py to /home/demo/quickstart/sampleproject/se
Copying /home/demo/quickstart/project_tmpl/urls.py to /home/demo/quickstart/sampleproject/urls.p
Copying /home/demo/quickstart/project_tmpl/apps/media_storage.py to /home/demo/quickstart/sample
Copying /home/demo/quickstart/project_tmpl/apps/yourappshere.txt to /home/demo/quickstart/sample
Copying /home/demo/quickstart/project_tmpl/bin/ext-status.sh to /home/demo/quickstart/sampleproj
Copying /home/demo/quickstart/project_tmpl/bin/install.sh to /home/demo/quickstart/sampleproject
Copying /home/demo/quickstart/project_tmpl/bin/pull-ext.sh to /home/demo/quickstart/sampleprojec
Copying /home/demo/quickstart/project_tmpl/bin/push-ext.sh to /home/demo/quickstart/sampleprojec
Copying /home/demo/quickstart/project_tmpl/bin/upgrade.sh to /home/demo/quickstart/sampleproject
Copying /home/demo/quickstart/project_tmpl/conf/$$$$PROJECT_NAME$$$$.wsgi to /home/demo/quicksta
Copying /home/demo/quickstart/project_tmpl/conf/apache2-$$$$PROJECT_NAME$$$$ to /home/demo/quick
Copying /home/demo/quickstart/project_tmpl/conf/nginx-$$$$PROJECT_NAME$$$$ to /home/demo/quickst
Copying /home/demo/quickstart/project_tmpl/lib/yourlibshere.txt to /home/demo/quickstart/samplep
Copying /home/demo/quickstart/project_tmpl/setup/_bootstrap.sh to /home/demo/quickstart/samplepr
Copying /home/demo/quickstart/project_tmpl/setup/calloway_reqs.txt to /home/demo/quickstart/samp
Copying /home/demo/quickstart/project_tmpl/setup/requirements.txt to /home/demo/quickstart/sampl
Copying /home/demo/quickstart/project_tmpl/templates/404.html to /home/demo/quickstart/samplepro
Copying /home/demo/quickstart/project_tmpl/templates/500.html to /home/demo/quickstart/samplepro
```

```
Copying /home/demo/quickstart/project_tmpl/templates/base.html to /home/demo/quickstart/samplepr
Copying /home/demo/quickstart/project_tmpl/templates/site_base.html to /home/demo/quickstart/sam
Copying /home/demo/quickstart/project_tmpl/templates/admin/base_site.html to /home/demo/quicksta
Copying /home/demo/quickstart/project_tmpl/templates/admin/change_form.html to /home/demo/quicks
Copying /home/demo/quickstart/project_tmpl/templates/admin/positions/selector.html to /home/demo
Making the virtual environment (sampleproject)...
New python executable in sampleproject/bin/python
Installing distribute.....................................................................
virtualenvwrapper.user_scripts Creating /home/demo/.virtualenvs/sampleproject/bin/predeactivate
virtualenvwrapper.user_scripts Creating /home/demo/.virtualenvs/sampleproject/bin/postdeactivate
virtualenvwrapper.user_scripts Creating /home/demo/.virtualenvs/sampleproject/bin/preactivate
virtualenvwrapper.user_scripts Creating /home/demo/.virtualenvs/sampleproject/bin/postactivate
Searching for pip
Best match: pip 0.6.3
Processing pip-0.6.3-py2.6.egg
pip 0.6.3 is already the active version in easy-install.pth
Installing pip script to /home/demo/.virtualenvs/sampleproject/bin

Using /home/demo/.virtualenvs/sampleproject/lib/python2.6/site-packages/pip-0.6.3-py2.6.egg
Processing dependencies for pip
Finished processing dependencies for pip
Installing requirements...
Downloading/unpacking calloway
  Downloading calloway-0.1.10.tar.gz (10.4Mb): 10.4Mb downloaded
  Running setup.py egg_info for package calloway
    no previously-included directories found matching 'project'
    no previously-included directories found matching 'calloway_test'
Installing collected packages: calloway
  Running setup.py install for calloway
    no previously-included directories found matching 'project'
    no previously-included directories found matching 'calloway_test'
    changing mode of build/scripts-2.6/generate_reqs.py from 644 to 755
    changing mode of /Users/coordt/.virtualenvs/sampleproject/bin/generate_reqs.py to 755
Successfully installed calloway
Downloading/unpacking Django
  Downloading Django-1.3.tar.gz (6.2Mb): 6.2Mb downloaded
  Running setup.py egg_info for package Django
    warning: no files found matching '*' under directory 'examples'
Installing collected packages: Django
  Running setup.py install for Django
    changing mode of build/scripts-2.6/django-admin.py from 644 to 755
    warning: no files found matching '*' under directory 'examples'
    changing mode of /Users/coordt/.virtualenvs/sampleproject/bin/django-admin.py to 755
Successfully installed Django
Cleaning up...
```

The script copies the template project to the specified folder, replacing the placeholder elements in transit. Then it creates a virtualenv, installs pip and distribute, and finally executes an initial `pip install -r setup/requirements.txt` that should at least install Django and Calloway.

4. Switch to the virtual environment we specified in the script:

```
workon sampleproject
```

5. Customize the `settings.py` file, specifically changing the application bundles included in the `INSTALLED_APPS` setting. The example template uses:

```
INSTALLED_APPS = APPS_DJANGO13_BASE + \
    APPS_ADMIN + \
```

```
    APPS_STAFF + \
    APPS_REVERSION + \
    APPS_STORIES + \
    APPS_CALLOWAY_DEFAULT + \
    APPS_MPTT + \
    APPS_CATEGORIES + \
    APPS_COMMENT_UTILS + \
    APPS_FRONTEND_ADMIN + \
    APPS_MEDIA + \
    APPS_UTILS + \
    APPS_REGISTRATION + \
    APPS_TINYMCE
```

Remove any application bundles that you don't want or need, and add any others by adding them in an additional tuple like:

```
INSTALLED_APPS = APPS_CORE + \
    APPS_ADMIN + \
    APPS_STAFF + \
    ...
    APPS_TINYMCE + (
        "cheese_shop",
        "dead_parrot",
        "holy_grail",
    )
```

6. When Calloway is installed, it installs a stand-alone script to dynamically generate a `pip` requirements file based on the applications specified in `INSTALLED_APPS` that it knows about. It can populate or create a file specified or else it prints the requirements to standard out.

   ```
   generate_reqs setup/calloway_reqs.txt
   ```

7. If you have added any other Django apps or libraries, make sure you update the `setup/requirements.txt` file.

8. Finally, to install the new pieces:

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

   The example project template includes `-r setup/calloway_reqs.txt` at the end of its requirements file. Some packages may need to be compiled. After all the packages are installed you should see something like:

   ```
   Successfully installed BeautifulSoup critic django-admin-tools ...
   Cleaning up...
   ```

9. Synchronize your database, as you normally would:

   ```
   ./manage.py syncdb
   ```

CHAPTER

FOUR

MIGRATING AN EXISTING DJANGO PROJECT TO CALLOWAY

4.1 For Starters

To start off, make sure you are using `virtualenv` and `virtualenvwrapper`. If you have not already done so, install them:

```
pip install virtualenv virtualenvwrapper
```

Then create a new virtual environment for your new Calloway based project:

```
mkvirtualenv newproject
```

Note: Make sure to change your the shbang (first line) in your project's `manage.py` to point to the newly installed copy of python in your virtual environment. It should be something like: `#!/home/user/.virtualenvs/newproject/bin/python`

Activate your new virtual environment and install the Calloway app:

```
workon newproject
pip install calloway
```

This will install a load of dependencies along with it. To see the full list for future reference, you can generate a requirements file:

```
generate_reqs > requirements.txt
```

4.2 Settings

In your project's `settings.py` make the following changes. Add this bit to the top of the file so that the Calloway apps are on your `sys.path`:

```
import os, sys
import calloway

CALLOWAY_ROOT = os.path.abspath(os.path.dirname(calloway.__file__))
sys.path.insert(0, os.path.join(CALLOWAY_ROOT, 'apps'))
```

Then import the default Calloway settings:

```
from calloway.settings import *
```

Now you can alter your `INSTALLED_APPS` setting to include the Calloway application bundles. Prepend your local apps with the bundles like so:

```
INSTALLED_APPS = APPS_CORE + \
    APPS_ADMIN + \
    APPS_STAFF + \
    APPS_REVERSION + \
    APPS_STORIES + \
    APPS_CALLOWAY_DEFAULT + \
    APPS_MPTT + \
    APPS_CATEGORIES + \
    APPS_COMMENT_UTILS + \
    APPS_FRONTEND_ADMIN + \
    APPS_MEDIA + \
    APPS_UTILS + \
    APPS_REGISTRATION + \
    APPS_TINYMCE + \
    ( # Local apps here
        "cheese_shop",
        "dead_parrot",
        "holy_grail",
    )
```

Now you can adjust your media settings. Below is an example of how to setup the media where the `static` folder contains all of your project's assets and the `media` folder is where the new media is copied into for serving. For more information on media handling, checkout *Media Configuration*

```
try:
    from local_settings import MEDIA_URL_PREFIX
except ImportError:
    MEDIA_URL_PREFIX = "http://media.example.com/"
try:
    from local_settings import MEDIA_ROOT_PREFIX
except ImportError:
    MEDIA_ROOT_PREFIX = '/var/www/media'
try:
    from local_settings import MEDIA_ROOT
except ImportError:
    MEDIA_ROOT = os.path.join(MEDIA_ROOT_PREFIX, 'ugc')
try:
    from local_settings import STATIC_ROOT
except ImportError:
    STATIC_ROOT = os.path.join(MEDIA_ROOT_PREFIX, 'static')


MEDIA_URL = '%sugc/' % MEDIA_URL_PREFIX
STATIC_URL = "%sstatic/" % MEDIA_URL_PREFIX
STATIC_MEDIA_APP_MEDIA_PATH = STATIC_ROOT
STATIC_MEDIA_COPY_PATHS = (
    {'from': os.path.join(CALLOWAY_ROOT, 'media'), 'to': STATIC_ROOT},
    {'from': 'static', 'to': STATIC_ROOT},
)
STATIC_MEDIA_COMPRESS_CSS = not DEBUG
STATIC_MEDIA_COMPRESS_JS = not DEBUG
STATIC_MEDIA_PURGE_OLD_FILES = False
```

Then make sure you add the default Calloway templates:

```
TEMPLATE_DIRS = (
    'templates',
    ...
) + CALLOWAY_TEMPLATE_DIRS
```

The last bits you need to consider is middleware. Again here is an example of `MIDDLEWARE_CLASSES` that play nicely with Calloway:

```
MIDDLEWARE_CLASSES = (
    'django.middleware.cache.UpdateCacheMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.middleware.gzip.GZipMiddleware',
    'django.middleware.http.ConditionalGetMiddleware',
    'django.middleware.doc.XViewMiddleware',
    'django.contrib.redirects.middleware.RedirectFallbackMiddleware',
    'django.contrib.flatpages.middleware.FlatpageFallbackMiddleware',
    'pagination.middleware.PaginationMiddleware',
    'django.middleware.transaction.TransactionMiddleware',
    'ban.middleware.DenyMiddleware',
    'django.middleware.cache.FetchFromCacheMiddleware',
)
```

And finally there are some settings you could define in `local_settings.py` which should make life a bit easier including media:

```
CACHE_BACKEND = "dummy:///"
MEDIA_ROOT_PREFIX = 'media'
MEDIA_URL_PREFIX = '/media/'
MEDIA_ROOT = 'uploads'
ADMIN_MEDIA_PREFIX = '/media/static/admin/'
```

4.3 URLs

Now you can add the Calloway urlpatterns onto your existing patterns in `urls.py`:

```
from calloway.urls import urlpatterns as calloway_patterns

urlpatterns += calloway_patterns
```

If you also want the catch all categories app to start at the site root, add this line:

```
urlpatterns += patterns('', ('', include('categories.urls')))
```

Lastly you can setup a development media server to host your assets:

```
if settings.DEBUG:
    urlpatterns += patterns('django.views.static',
        (r'^media/static/(?P<path>.*)$', 'serve',
            {'document_root': settings.STATIC_ROOT, 'show_indexes': True}),
    )
```

CHAPTER

FIVE

TEMPLATE PROJECTS

5.1 The `create_project` script

Calloway is merely a glue between multiple open source projects and requires a host project.

CHAPTER

SIX

MEDIA CONFIGURATION

Calloway allows for a range of media serving options. While many Django users are used to having a single directory in which everything resides, others may need to have media hosted somewhere else, and if you allow for user uploads, you may want those stored somewhere different from your regular media.

6.1 Three Different Kinds of Media

All media is not created equal, and you may want to handle different media, well, differently. Here are a few types we've identified.

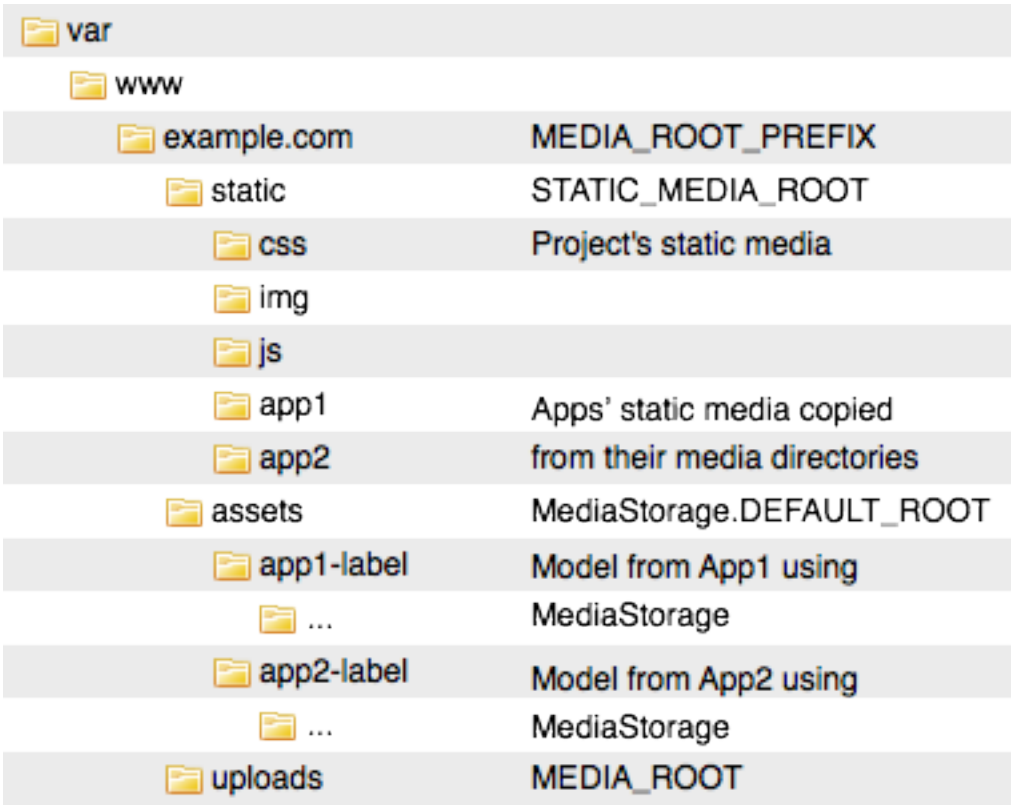
assets Assets are files that your staff uploads, such as photos, documents and audio recordings. Django 1.1 or greater allows for custom media storage, which Calloway can use to separate the staff uploads from applications such as Massmedia from user uploads from applications like a user profile.

static Static files are files specifically for the website, such as CSS, Javascript, and images. You should store these files in your repository. Many times you will want to combine some and minify or otherwise optimize others.

uploads You might not allow user uploads, but if you ever do, you will want to keep these files separate from others.

6.2 Media Management in Calloway

By default, Django wants to store all media in the same place. Segregating different types of media requires a bit of reconfiguration to make it work. The settings here assumes that static or website media is stored in the repository in a directory named `static`. Assets and uploads are stored in directories within a directory named `media` where `media` is *not* in the repository.



6.2.1 `MEDIA_URL_PREFIX` and `MEDIA_ROOT_PREFIX` settings

These settings aren't standard Django settings, but it makes it easier to configure the multiple urls we need. If you are going to store production media on a separate server or location from your development media.

The sample project template configures these:

```
try:
    from local_settings import MEDIA_URL_PREFIX
except ImportError:
    MEDIA_URL_PREFIX = "http://media.example.com/"
try:
    from local_settings import MEDIA_ROOT_PREFIX
except ImportError:
    MEDIA_ROOT_PREFIX = '/nfs-media/website/'
```

This configuration allows for media to be handled differently during development than deployment, and allows for custom media storage to utilize them.

6.2.2 `MEDIA_ROOT` and `MEDIA_URL` settings

This is the default place that Django looks for media. We are going to use this for user-generated content.

```
try:
    from local_settings import MEDIA_ROOT
except ImportError:
    MEDIA_ROOT = os.path.join(MEDIA_ROOT_PREFIX, 'ugc')
MEDIA_URL = '%sugc/' % MEDIA_URL_PREFIX
```

6.2.3 STATIC_MEDIA and STATIC_URL settings

This is a proposed addition to Django's default settings. Other projects, such as Pinax, are already using this convention for their website media. Static Media Manager has a context processor that makes STATIC_URL available to the template.

```
try:
    from local_settings import STATIC_ROOT
except ImportError:
    STATIC_ROOT = os.path.join(MEDIA_ROOT_PREFIX, 'static')
STATIC_URL = "%sstatic/" % MEDIA_URL_PREFIX
```

6.2.4 Massmedia storage setup

Massmedia allows for the configuration of a default storage system to be used for all types of media, and separate configurations to override the default for each type of media. In the apps directory is media_storage.py. This contains a custom file storage system for django. The following setting sets the correct storage place using the MEDIA_URL_PREFIX and MEDIA_ROOT_PREFIX settings.

```
MMEDIA_DEFAULT_STORAGE = 'media_storage.MediaStorage'
```

6.2.5 StaticMediaMgr settings

The tool we developed for managing application static media as well as site-specific static media is StaticMediaMgr. It can copy, compress, minify and join files in a configurable way.

First, we need to handle application media. StaticMediaMgr looks in every application in INSTALLED_APPS for a media directory. All of these directories need a place to copy. We will set it to the destination static media directory.

Note: This media is copied *before* the items in the static directory. This allows us to override a specific media's items similarly to overriding templates.

```
STATIC_MEDIA_APP_MEDIA_PATH = STATIC_ROOT
```

We can have StaticMediaMgr recursively copy multiple directories to different places.

```
STATIC_MEDIA_COPY_PATHS = (
    {'from': os.path.join(CALLOWAY_ROOT, 'media'), 'to': STATIC_ROOT},
    {'from': 'static', 'to': STATIC_ROOT},
)
```

Last, but not least, we must configure the compression. During development, we don't want it, but we do want it for production:

```
STATIC_MEDIA_COMPRESS_CSS = not DEBUG
STATIC_MEDIA_COMPRESS_JS = not DEBUG
```

6.2.6 Ignoring the right things

The .gitignore file should contain several things:

```
*.pyc
.svn
.DS_Store
local_settings.py
externals
pip-log.txt
dev.db
media2/
media/
```

CHAPTER

SEVEN

APPLICATION BUNDLES IN SETTINGS

Calloway's settings includes bundles of applications that are either dependent upon each other, or work well together. Some application bundles have suggestions for others that can augment their functionality, but are not required.

Application bundles are only for convenience. Feel free to mix and match applications.

7.1 calloway.settings.APPS_CORE

Warning: As of version 0.4 this application bundle is deprecated. In order to better handle core Django apps in different versions, this bundle has been broken into *calloway.settings.APPS_DJANGO_BASE*, *calloway.settings.APPS_DJANGO13_BASE*, and *calloway.settings.APPS_DJANGO_TEMPLATE_UTILS*.

These applications are considered core to any project and consist of Django `contrib` applictions.

Suggested application bundles: `APPS_TINYMCE`, `APPS_REVERSION` (for flatpages)

- `django.contrib.auth`
- `django.contrib.contenttypes`
- `django.contrib.sessions`
- `django.contrib.sites`
- `django.contrib.flatpages`
- `django.contrib.humanize`
- `django.contrib.comments`
- `django.contrib.markup`
- `django.contrib.redirects`

7.2 calloway.settings.APPS_DJANGO_BASE

Note: New in version 0.4

These are the core contrib apps included in the default settings for Django 1.0 - 1.2.

Suggested application bundles: `APPS_TINYMCE`, `APPS_REVERSION` (for flatpages)

- `django.contrib.auth`
- `django.contrib.contenttypes`
- `django.contrib.sessions`
- `django.contrib.sites`
- `django.contrib.flatpages`

7.3 calloway.settings.APPS_DJANGO13_BASE

Note: New in version 0.4

These are the core contrib apps included in the default settings for Django 1.3.

Suggested application bundles: `APPS_TINYMCE`, `APPS_REVERSION` (for flatpages)

- `django.contrib.auth`
- `django.contrib.contenttypes`
- `django.contrib.sessions`
- `django.contrib.sites`
- `django.contrib.flatpages`
- `django.contrib.messages`
- `django.contrib.staticfiles`

7.4 calloway.settings.APPS_DJANGO_TEMPLATE_UTILS

Note: New in version 0.4

When you want special help with templates, these contrib apps can help out.

- `django.contrib.humanize`
- `django.contrib.markup`
- `django.contrib.webdesign`

7.5 calloway.settings.APPS_ADMIN

The admin area of the project requires several apps. The `livevalidation` application provides real-time validation of forms, including the admin forms. `django-admin-tools` provides the custom skin.

Warning: In version 0.4, livevalidation has been removed from this bundle.

- admin_tools
- admin_tools.theming
- admin_tools.menu

- admin_tools.dashboard
- `django.contrib.admin`

7.6 calloway.settings.APPS_CALLOWAY_DEFAULT

Calloway includes no real applications, except `django_ext` which are small extensions to Django. Various utilities are included within this, including a case-insensitive authorization, requirements generator and a few others.

- `django_ext`

7.7 calloway.settings.APPS_MPTT

Django MPTT is an application that enhances other applications' ability to manage hierarchical data. It is required in *APPS_CATEGORIES* and *APPS_COMMENT_UTILS*.

- mptt

7.8 calloway.settings.APPS_STAFF

A specialized profile for staff members.

Suggested application bundles: *APPS_TINYMCE*

- staff

7.9 calloway.settings.APPS_REVERSION

Django Reversion is an app that enhances other applications. It allows you to view and revert to previous versions of records. Django Stories and Django Flatpages are currently configured to use it.

- reversion

7.10 calloway.settings.APPS_STORIES

Warning: Changed in version 0.4: Django Viewpoint (a blogging app) and Django Pullquote (for storing quotations) were removed.

A bundle of applications for creating news.

Suggested application bundles: *APPS_TINYMCE*, *APPS_REVERSION*

- stories
- positions
- news_sitemaps

7.11 calloway.settings.APPS_BLOGGING

Note: New in version 0.4

A blogging platform for one blog or many blogs.

Suggested application bundles: *APPS_TINYMCE*, *APPS_REVERSION*

- viewpoint

7.12 calloway.settings.APPS_CATEGORIES

A hierarchical category manager. Requires `:ref:`APPS_MPTT <apps_mptt>``.

- categories
- editor

7.13 calloway.settings.APPS_COMMENT_UTILS

Utilities for adding threaded comments to the default Django comments app and management of offensive comments. Requires *APPS_MPTT*

- mptt_comments
- offensivecontent

7.14 calloway.settings.APPS_FRONTEND_ADMIN

Allows using your admin forms in a regular template. Requires livevalidation in *APPS_ADMIN*.

- frontendadmin

7.15 calloway.settings.APPS_MEDIA

Warning: Changed in version 0.4: Django Tagging was moved to *apps_tagging*.

- massmedia

`calloway.settings.APPS_TAGGING` * tagging

7.16 calloway.settings.APPS_UTILS

- robots
- piston
- `ban`

- native_tags
- google_analytics
- hiermenu
- synagg
- uni_form
- critic
- mailfriend
- debug_toolbar
- pollit
- pullquote

7.17 calloway.settings.APPS_CACHING

- django_memcached
- versionedcache

7.18 calloway.settings.APPS_REGISTRATION

- registration
- custom_registration

7.19 calloway.settings.APPS_TINYMCE

- tinymce

CHAPTER

EIGHT

REFERENCE

8.1 Settings

CHAPTER

NINE

INDICES AND TABLES

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