Developing reusable apps

James Bennett

PyCon Chicago, March 15, 2008

The fourfold path

- Do one thing, and do it well.
- Don’t be afraid of multiple apps.
- Write for flexibility.
- Build to distribute.

1

“

Do one thing, and do it well.

”

-- The UNIX philosophy

Application == encapsulation

Case study: user registration

Django snippets: Sign up

http://www.djangosnippets.org/accounts/register/

djangosnippets

Log in or register | Snippets | About

By author | By language | By tag | Highest-rated | Most bookmarked

Sign up

Username:

Email address:

Password:

Password (type again to catch any typos):

I have read and agree to the Terms of Service:

Register

Fill out the form to the left (all fields are required), and your account will be created; you'll be sent an email with instructions on how to finish your registration.

Snippets
By author
By language
By tag
Highest-rated

About
Powered by Django.
Learn more about this site.
Read the FAQ.

Features

- User fills out form, inactive account created.
- User gets email with link, clicks to activate.
- And that's it.

Some "simple" things aren't so simple.

Approach features skeptically

Should I add this feature?

- What does the application do?
- Does this feature have anything to do with that?
- No? Guess I shouldn't add it, then.

Top feature request in django-registration: User profiles

“

What does that have to do with user registration?

”

-- Me


No, You Can't Have a Pony
Not Yours

The solution?

django-profiles

- Add profile
- Edit profile
- View profile
- And that's it.

2

Don’t be afraid of multiple apps

The monolith mindset

- The “application” is the whole site
- Re-use is often an afterthought
- Tend to develop plugins that hook into the “main” application

The Django mindset

- Application == some bit of functionality
- Site == several applications
- Tend to spin off new applications liberally

Should this be its own application?

- Is it orthogonal to whatever else I'm doing?
- Will I need similar functionality on other sites?
- Yes? Then I should break it out into a separate application.

Case study: blogging

I wanted a blog

- Entries and links
- Tagging
- Comments with moderation
- Contact form
- “About” page
- Etc., etc.

I ended up with

- A blog app (entries and links)
- A third-party tagging app
- `contrib.comments` + moderation app
- A contact-form app
- `contrib.flatpages`
- Etc., etc.

Advantages

- Don’t keep rewriting features
- Drop things into other sites easily

```
urlpatterns += (‘’,
    (r’^contact/’, include(‘contact_form.urls’)),
)
```

But what about…

Site-specific needs

- Site A wants a contact form that just collects a message.
- Site B's marketing department wants a bunch of info.
- Site C wants to use Akismet to filter automated spam.

3

Write for flexibility

Common sense

- Sane defaults
- Easy overrides
- Don’t set anything in stone

Form processing

- Supply a form class
- But let people specify their own if they want

```
class SomeForm(forms.Form):
    ...

def process_form(request, form_class=SomeForm):
    if request.method == 'POST':
        form = form_class(request.POST)
        ...
    else:
        form = form_class()
    ...
```

Templates

- Specify a default template
- But let people specify their own if they want

```
def process_form(request, form_class=SomeForm,
                 template_name='do_form.html'):
    ...
    return render_to_response(template_name,
                              ...
```

Form processing

- You want to redirect after successful submission
- Supply a default URL
- But let people specify their own if they want

```
def process_form(request, form_class=SomeForm,
                 template_name='do_form.html',
                 success_url='/foo/'):
    ...
    return HttpResponseRedirect(success_url)
```

URL best practices

- Provide a URLConf in the application
- Use named URL patterns
- Use reverse lookups: `reverse()`, `permalink`, `{% url %}`

4

Build to distribute

So you did the tutorial

```
➜from mysite.polls.models import
  Poll
➜mysite.polls.views.vote
➜include(‘mysite.polls.urls’)
➜mysite.mysite.bork.bork.bork
```

Project coupling kills re-use

What is a "project"?

- A settings module
- A root URLConf module
- And that's it.

ljworld.com

- `worldonline.settings.ljworld`
- `worldonline.urls.ljworld`
- And a whole bunch of reused apps

What reusable apps look like

- Single module directly on Python path (`registration`, `tagging`, etc.)
- Related modules under a package (`ellington.events`, `ellington.podcasts`, etc.)
- No project cruft whatsoever

And now it's easy

- You can build a package with `distutils` or `setuptools`
- Put it on the Cheese Shop
- People can download and install

General best practices

- Be up-front about dependencies
- Write for Python 2.3 when possible
- Pick a release or pick trunk, and document that
- But if you pick trunk, update frequently

Templates are hard

- Providing templates is a big “out of the box” win
- But templates are hard to make portable (block structure/inheritance, tag libraries, etc.)

I usually don't do default templates

Either way

- Document template names
- Document template contexts

Be obsessive about documentation

- It's Python: give stuff docstrings
- If you do, Django will generate documentation for you
- And users will love you forever

Recap:

- Do one thing, and do it well.
- Don’t be afraid of multiple apps.
- Write for flexibility.
- Build to distribute.

Good examples

- django-tagging (Jonathan Buchanan, http://code.google.com/p/django-tagging/)
- django-atompub (James Tauber, http://code.google.com/p/django-atompub/)
- Search for “django” on code hosting sites

More information

- django-hotclub (http://groups.google.com/group/django-hotclub/)
- Jannis Leidel's django-packages (http://code.google.com/p/django-reusableapps/)
- Django sprint at PyCon

Questions?

Photo credits

- "Purple Sparkly Pony Ride" by ninjapoodles, http://www.flickr.com/photos/ninjapoodles/285048576/
- “Stonehenge #2” by severecci, http://www.flickr.com/photos/severecci/129553243/
- “sookiepose” by 416style, http://www.flickr.com/photos/sookie/41561946/
- "The Happy Couple" by galapogos, http://www.flickr.com/photos/galapogos/343592116/