The Django Ecosystem

and you

Trey Hunner

Django Day

February 23, 2013

Where should I go to get help?

How I seek help on a Django-related problem:

1. Think of search terms that describe the problem
2. Search Duck Duck Go and/or Google
3. Search StackOverflow
4. Search documentation
5. Discover more search terms and start again at step 1

Refreshing your memory

You have a function/class but don't know how to use it.

1. For Django built-ins try http://django.me
2. Consult the documentation
3. Read the code

Refreshing your memory

You have a function/class but don't know how to use it.

1. For Django built-ins try http://django.me
2. Consult the documentation
3. Read the code
4. **Read the code**

How to read the code

How should I read the code?

Good	Find the code on Github and read in Firefox
Better	Find the code on your computer and read in your IDE
Best	Use “jump to definition” to open code in your IDE

How to read the code

How should I read the code?

Good	Find the code on Github and read in Firefox
Better	Find the code on your computer and read in your IDE
Best	Use “jump to definition” to open code in your IDE

Jumping to code definitions is supported in PyCharm, Wing IDE, Vim, and many other IDEs.

Stuff you should consume

- Books & Videos
- Newsletters & Mailing Lists
- IRC
- Blogs

Books & Videos

Books:

- Two Scoops of Django (not free, but really really useful)
- The Django Book (useful if you've never read it)

Videos:

- pyvideo.org (big index of Python talks)
- Marc Abramowitz's PyCon 2012 Notes

Mailing Lists

You should subscribe to:

- PyCoder's Weekly (you must subscribe to this)
- django-users list
- django-developers list (for Django devs to chat publicly)
- the mailing list of your favorite Django projects

IRC Rooms

First: Register a nickname on FreeNode.net.
Next: Join IRC channels:

- #django for general chatting and help (read the FAQ)
- #sandiegopython (there's always someone sitting in there)
- the IRC channels for your favorite projects. Examples:
    - #django-compressor
    - #restframework
    - #haystack
    - #fabric

Blogs to follow

- Django Weblog: the official blog
- Planet Django: reposts from everywhere

More blogs to read

These don't update frequently, but you should **read the archives**.

- Hackers Gonna Hack
- Procrastinating Dev
- Toast Driven
- Inside the head of PyDanny
- Hark! A Blog
- Jacob Kaplan-Moss' Blog
- Daniel Roseman's Blog
- Surfing in Kansas

I added just enough blogs to fit on this slide. There might be more.

Where do I go to find code?

- DjangoSnippets.org: small bits of code
- DjangoPackages.com: large directory of Django apps
- StackOverflow.com: answers often contain. . .
    - useful code snippets
    - links to useful Django apps
- GitHub
- Bitbucket

I found some code, should I use it?

I found some code on Github. Should I use it?

Yes	Why write new code when you can reuse code?
No	Someone else wrote it. It could be buggy.
Maybe	You need to evaluate your options.

How to evaluate code you found online

When evaluating new code ask...

1. Is it fresh?
2. Is it well-tested?
3. Is it popular?
4. Is it hackable?

Is the code fresh?

- Does this work in the newest version of Django?
- When was the last commit?
- When was the last mentioned of this package on blogs/StackOverflow?

Most relevant for full projects, but could apply to code snippets.

Is the code well-tested?

- Are there tests?
- Are they quality tests?
- What does the code coverage look like?

Most relevant for projects that don't have a very large community.

Is the code popular?

- How many stars or followers are there?
- How many forks, committers, closed issues and pull requests?
- Is it recommended on any blogs or Q & A websites?

Most relevant for large projects, especially when you wouldn't want to maintain your own fork of the code.

Is this code hackable?

- Do you understand how the code works?
- Would you be comfortable fixing bugs or adding features?
- Are the project maintainers accepting of contributions?
- Would you be willing to maintain your own fork of the code?

Most relevant for projects that may not fit your needs exactly.

Reasons you should write open source code

- You'll eventually find a bug. . . you might as well fix it
- You'll eventually discover a feature you want to add
- It will bring you wealth, fame, and happiness

Alternatives to open source

Some alternatives to contributing to open source projects:

- Monkey-patching the code to get the behavior you want
- Fixing/updating the code and maintaining your own fork
- Begging the community to fix your bugs and feature requests

How to contribute code

When contributing code to an open source project:

1. Tell the author what you're planning to do (optional)
2. Write tests that should pass after you're done
3. Write the code, asking the community for advice as needed

How to contribute code

When contributing code to an open source project:

1. Tell the author what you're planning to do (optional)
2. Write tests that should pass after you're done
3. Write the code, asking the community for advice as needed
4. Write all the tests you forgot to write in step 2

In Summary

1. Watch a video from PyVideos every day
2. Read Two Scoops of Django
3. Write code and stuff