The Impact of Django

Armin Ronacher

djangocon europe 2011 — http://lucumr.pocoo.org/talks/

Traveling to the Past

What did the World look like in July of 2005?

The Year 2005

The initial release of Django was on

July 21th 2005

Do you remember?

- mod_python
- django.core.meta
- Magic all over the place
- TurboGears 1 — “Best of Breed”

django.core.meta

```
# Get a reference to the module the class is in, and
# dynamically add the new module to it.
app_package = sys.modules.get(new_class.__module__)
if replaces_module is not None:
    app_label = replaces_module[0]
else:
    app_package.__dict__[opts.module_name] = new_mod
    app_label = app_package.__name__[
        app_package.__name__.rfind('.')+1:]
```

django.core.meta

```
def _reassign_globals(function_dict, extra_globals, ns):
    new_functions = {}
    for k, v in function_dict.items():
        code = v.func_code
        new_globals = {'__builtins__': __builtins__,
                       'db': db.db, 'datetime': datetime}
        new_globals.update(extra_globals.__dict__)
        func = types.FunctionType(code, globals=new_globals,
            name=k, argdefs=v.func_defaults)
        func.__dict__.update(v.__dict__)
        setattr(ns, k, func)
        for new_k, new_v in new_functions.items():
            new_v.func_globals[k] = func
        new_functions[k] = func
```

What?

*"Note that we could have just left the extra methods in [...], but that would have meant that any code within the extra methods would *not* have access to module-level globals, [...]. In order to give these methods access to those globals, we have to deconstruct the method getting its raw "code" object, then recreating the function with a new "globals" dictionary."*

Good Times

```
class User(meta.Model):
    fields = (
        meta.CharField('username', 'user name', maxlength=60),
        meta.CharField('password', maxlength=100)
    )

    def save(self):
        assert is_secure_password(self.password)
        meta.Model.save(self)
```

Ambitions

“Django's been out as an unofficial pre-release for almost […] now, and it's about time to wrap things up to roll a 1.0 release.”

— Jacob Kaplan-Moss

Big Plans

“We've decided that Django, like Python itself, should put a very high priority on backwards compatibility.”

— Jacob Kaplan-Moss

But what did it have?

- an ORM. (albeit a different one)
- WSGI support
- reStructured Text in the Documentation
- the same template engine
- the Admin interface

ORM

Declaration > Reflection

Ideas

- Declare database tables in Python code
- Map each row in a table to an instance of a model

Not unique to Django

- … but the popularity of Django made many people use the same pattern
  - Everything is configured from Python
  - No XML, no INI files, no SQL schema files etc.

WSGI

Back when it was unpopular

2005

- Dec 7th: WSGI celebrates second birthday
- TurboGears does not support WSGI yet
- Django has WSGI Support in [169]

WSGI -> Rack

```
        * chris2 looks at pythons wsgi
<chris2> python has a lot of web frameworks, i think. at least they
         dont all duplicate common code
 <zedas> manveru: not really.  they want me to collaborate to make
         whatever they have official.
<mitsuhiko> chris2, zedas: +1 for a ruby wsgi
   <chris2> mitsuhiko: if you were to draft the python wsgi, what would
            you change?
   <chris2> kuja: more a common layer of web server interfacing
    <zedas> well keep in mind that when i say I want mongrel to push a
            wsgi for ruby frameworks, i don't mean copy their api.  it
            hink the python api has transactional problems and see
            bolted on.
<mitsuhiko> chris2: i would add support for unicode to it
```

Trunk Stability / Packaging

Django hates Packaging

Stability

- Django encouraged SVN checkouts
  - trunk was documented to be reasonable stable
  - A concept that many Python projects nowadays follow

/trunk @ 155

- ez_setup.py
- setup.py

/setup.py [3906]

"Changed setup.py to use standard distutils instead of setuptools. This means installing Django no longer requires a easy_install, setuptools or a working Internet connection, greatly simplifying things."

— *Adrian Holovaty*

Aftermath

- easy_install broken for 2 months
- Django went the safe way and bundles requirements

A Good Decision

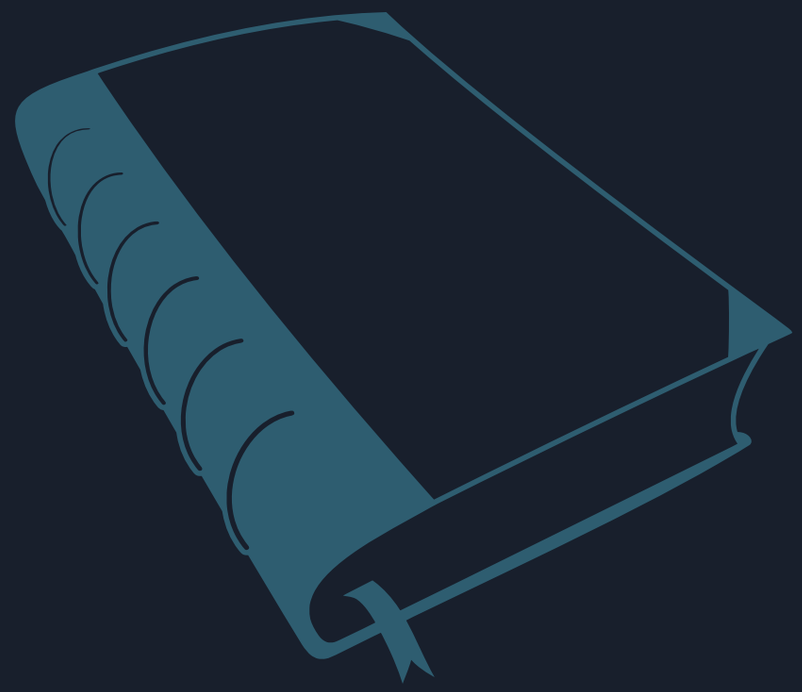
- TurboGears 1
  - Kid templates — author went to Ruby and maintenance stopped
  - CherryPy — was upgraded to a WSGI compliant server with changed API
  - SQLObject — SQLAlchemy came along and mostly replaced it. New API

Changing Situations

- Jannis Leidel / Carl Meyer working on pip
- Bundling was a good idea in 2005 for Django, that does not mean that this will continue to be the case now

Documentation

Setting a Trend



Python Docs in 2006

- Python documentation still based on pain, LaTeX and pain
- LaTeX sources -> PDF and via Perl to HTML

Convert like it's 1999

"These scripts and Makefile fragment are used to convert the Python documentation in LaTeX format to SGML. XML is also supported as a target, but is unlikely to be used."

— Guido van Rossum

Conversion Gems

"This is the really painful part of the conversion. Well, it's the second really painful part, but more of the pain is specific to the structure of the Python documentation and desired output rather than to the parsing of LaTeX markup."

— Guido van Rossum

Beginning of Sphinx

```
 <mitsuhiko> do you know the new djangobook webpage?
<birkenfeld> wow
 <mitsuhiko> something like that would be really cool to have for
             the python documentation
<birkenfeld> shouldn't be too hard
<birkenfeld> but the first thing would be to change the docs'
             format from latex to something better
 <mitsuhiko> rst!
 <mitsuhiko> :D
<birkenfeld> not really
<birkenfeld> I'm not sure if rst is powerful enough
 <mitsuhiko> i think most of the stuff the python documentation
             requires is possible with rst
<birkenfeld> hm, you could use roles for that
<birkenfeld> :module:`os.path`
```

Style

- Hand written prose documentation
- General structure of documentation
- Applies the “what's not documented is not implemented” rule

Templates

… you make the markup for the MTV

```
{% extends "awesome" %}
```

Django Inspired Templates

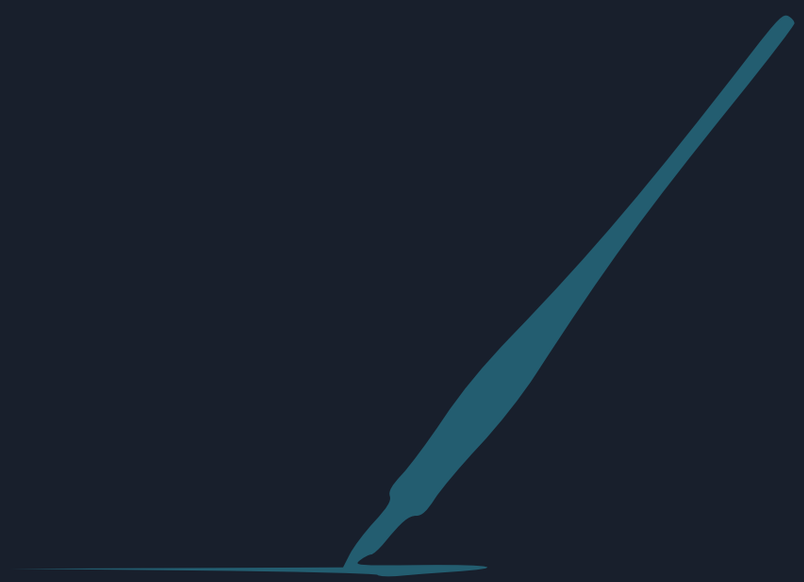
- Jinja (Python)
- Liquid Templates (Ruby)
- Jangod (Java)
- Twig (PHP)
- Dotiac (Perl)
- Djangode (JavaScript)
- NDjango (.NET)

New* Concepts

- Block based template inheritance
- Filters
- Trend towards less logic in templates
- Providing helpers for common operations in templates

Licensing

Because we believe in Open Source



Licensing on PyPI

- 2287 GPL packages (many mislabeled)
- 2475 BSD packages
- 1659 MIT packages
- 417 Apache packages

Django Classifier

- 111 out of 2287 GPL
- 708 out of 2478 BSD
- 181 out of 1659 MIT

Marketing / Management

Advertisement is Good, Management Important

stats on bitbucket

- 2993 repositories with Django in their repository name (1841 without forks)
- 14171 repositories with the language type set to Python. This has to happen manually.
- But only 22% of Django projects on bitbucket have their language set to Python.

stats on github

- 76571 repositories on github
- 9794 of those have “Django” in their name.
- public repositories only and includes forks.

PyPI Stats

- 15070 packages on PyPI
- 1171 of these packages have “Django” in the name
- 476 packages with various ways of writing “Zope” (z3c, zope, zopy etc.)

Bold Claim

- Django is the reason Python is getting that much attention lately. More than any other package on PyPI

Marketing Lite

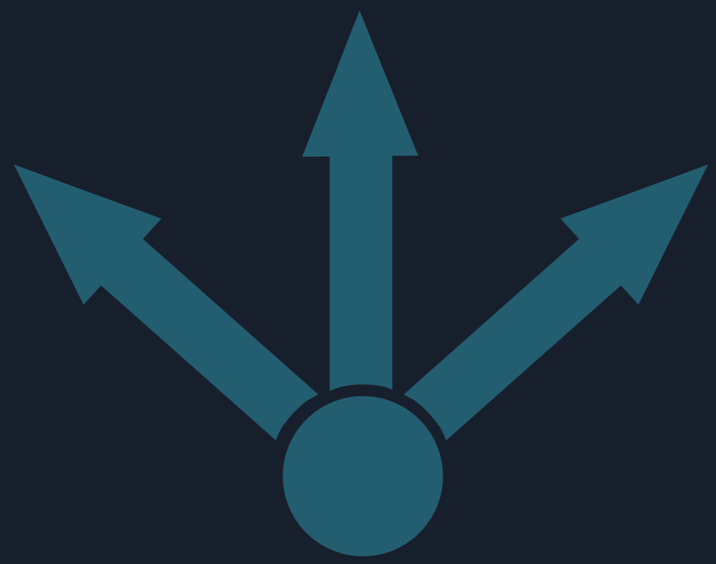
- Django had a beautiful and well structured website and documentation
- “The Web framework for perfectionists with deadlines”

Management

- Strong leadership
- Reasonable backwards compatibility policy*
- Testing, testing, testing

Bad Influence

not everything is great



Bad Examples

- django.conf.settings
- Magic

django.conf.settings

- Singleton
- Not exactly clear when it is imported
- No single entrypoint

Magic

- importing modules by name and then catching the import error
- find things by expecting them to be in a certain file with a certain name

9: