Django: Two Extreme Case Studies

Mike Biglan: CTO
Wiggins: Senior Developer

Concentric Sky
www.concentricsky.com

Wednesday, July 21, 2010
Overview

1. About the Projects
2. Speed of Development
3. Standards & Bootstrapping
4. Scaling
5. Integrations
About Concentric Sky

- Web, Mobile, Enterprise Java Development
- 35 people in Eugene, OR
- Our Django team has almost hit 10 people
- Django projects go well
- People enjoy using it
Two Extreme Deadlines

MichaelMoore.com: 5 weeks
Santa Fe Institute (santafe.edu): 6 weeks
Overview

1. About the Projects
2. Speed of Development
3. Standards & Bootstrapping
4. Scaling
5. Integrations
Process

- Owners of well-defined tasks
- Blockers identified/resolved ASAP
- Daily/routine checkins
- As issues come up, incorporate them into global process

Wednesday, July 21, 2010
Planning

- Data Schema (ERD) -> Django models
- Don’t overplan
- Vision to client/developers: sitemap, mockups, etc
- On same page?
- Early identification of changes; least costly
- Website Specification Document: Spreadsheet sitemap
Content & Migration

- Content components identified
- For each, owner for creating/moving content
- Structured content much easier
- Migration Methods: Python script, SQL script, manual to review site
Standards

- Standard project structure
- Standard project lifecycle
- Predict the questions/blockers as early as possible
Overview

1. About the Projects
2. Speed of Development
3. Standards & Bootstrapping
4. Scaling
5. Integrations
Bootstrapping: Traditional

- `django-admin.py startproject`
- clone latest project & prune
- clone a pre-pruned template, update several variables
Standard Project Structure: Why?

- Developers new to Django
- Developers new to project
- Easier to deploy, easier to build deployment scripts & environments
- Libs can be externals/sub-modules; isolated within each project
Djenesis

- Bootstraps a new project using a template
- Included: well-formed, web-based project template

http://code.google.com/p/djenesis/
Directory Structure

- **apps**: this project's apps (in PYTHONPATH)
- **lib**: 3rd party/helper libraries (in PYTHONPATH)
- **etc**: wsgi template file
- **mainsite**: the “main” project site
- **media**: css, img, js
- **templates**: with simple base.html
Mainsite

- **manage.py**: setup additional paths (top/lib/app)
- **settings.py**: primary settings, loads local
- **local_settings.py**: machine-dependent settings
- **urls.py**: In debug, serves static media; admin uncommented by default
Local Settings

- Multiple machines
- Multiple people
- Local, review, live servers
- Machine-dependent go in local_settings
- Not checked in to version control
Overview

1. About the Projects
2. Speed of Development
3. Standards & Bootstrapping
4. Scaling
5. Integrations
Deploying Django

http://www.djangobook.com/en/2.0/chapter12/
X-treme proxying

- Separate media server(s)
- Proxy dedicated database server(s)
- Load Balance app server(s)
- Use memcached for sticky sessions
- Cache your queries
CacheModel

- Quick, easy, does the heavy lifting
- Namespaced caching library
- Table-level caching with cache_key()
- Object-level caching with ns_cache_key()

http://code.google.com/p/django-cachemodel/
Book & Authors

Book

Author
from django.db import models
from django.core.cache import cache

class BookManager(models.Manager):
    def get_by_slug(self, slug, cache_timeout=900):
        cache_key = "book_by_slug_%s" % (slug,)
        book = cache.get(cache_key)
        if book is None:
            book = Book.objects.get(slug=slug)
            cache.set(cache_key, book, cache_timeout)
        return book

class Book(models.Model):
    slug = models.SlugField(max_length=128)
    name = models.CharField(max_length=128)
    objects = BookManager()

    def save(self, *args, **kwargs):
        super(Book, self).save(*args, **kwargs)
        cache.delete("book_by_slug_%s" % (self.slug,))

    def delete(self, *args, **kwargs):
        super(Book, self).save(*args, **kwargs)
        cache.delete("book_by_slug_%s" % (self.slug,))

>>> Book.objects.get_by_slug("my-book-slug")
<Book: Book Object>
from django.db import models
from django.core.cache import cache

class BookManager(models.Manager):
    def get_by_slug(self, slug, cache_timeout=900):
        cache_key = "book_by_slug_%s" % (slug,)
        book = cache.get(cache_key)
        if book is None:
            book = Book.objects.get(slug=slug)
            cache.set(cache_key, book, cache_timeout)
        return book

class Book(models.Model):
    slug = models.SlugField(max_length=128)
    name = models.CharField(max_length=128)
    objects = BookManager()

    def save(self, *args, **kwargs):
        super(Book, self).save(*args, **kwargs)
        cache.delete("book_by_slug_%s" % (self.slug,))

    def delete(self, *args, **kwargs):
        super(Book, self).save(*args, **kwargs)
        cache.delete("book_by_slug_%s" % (self.slug,))

>>> Book.objects.get_by_slug("my-book-slug")
<Book: Book Object>
from django.db import models
from django.core.cache import cache

class BookManager(models.Manager):
    def get_by_slug(self, slug, cache_timeout=900):
        cache_key = "book_by_slug_%s" % (slug,)
        book = cache.get(cache_key)
        if book is None:
            book = Book.objects.get(slug=slug)
            cache.set(cache_key, book, cache_timeout)
        return book

class Book(models.Model):
    slug = models.SlugField(max_length=128)
    name = models.CharField(max_length=128)
    objects = BookManager()

    def save(self, *args, **kwargs):
        super(Book, self).save(*args, **kwargs)
        cache.delete("book_by_slug_%s" % (self.slug,))

    def delete(self, *args, **kwargs):
        super(Book, self).save(*args, **kwargs)
        cache.delete("book_by_slug_%s" % (self.slug,))

>>> Book.objects.get_by_slug("my-book-slug")
<Book: Book Object>
from django.db import models
from cachemodel import models as cache_models

class Book(cache_models.CacheModel):
    slug = models.SlugField(max_length=128)
    name = models.CharField(max_length=128)

>>> Book.objects.get_by("slug", "my-book-slug")
<Book: Book Object>
from django.db import models
from cachemodel import models as cache_models

class Book(cache_models.CacheModel):
    slug = modelsSlugField(max_length=128)
    name = models.CharField(max_length=128)

    @cache_models.cached_method(900, 'authors')
def get_authors(self):
        return self.bookauthor_set.all()

class BookAuthor(cache_models.CacheModel):
    book = models.ForeignKey(Book)
    name = models.CharField(max_length=255)

    def flush_cache(self):
        super(BookAuthor, self).flush_cache()
        self.book.flush_cache()
Overview

1. About the Projects
2. Speed of Development
3. Standards & Bootstrapping
4. Scaling
5. System Integrations
System Integrations

- Google Search Appliance (GSA)
- Alfresco
- iCal Server
- Haystack/SOLR
Integrations

1) Identify (a) service and (b) push or pull

2a) Pull: Make a proxy model
   Use Caching

2b) Push: Make a web service