La D è muta: introduzione a Django

Riccardo Magliocchetti

Torino Coding Society 28/09/15

whoami

Consultant

Free software developer

- maintainer: django-admin-bootstrapped, uwsgitop
- contributor: uwsgi, LibreOffice

Python

Highlights

- OO but functional friendly
- whitespace indentation
- late binding type system, strong types
- great standard library
- implementations: cpython 2 and 3, pypy, jython
- package manager: pip, https://pypi.python.org

>>> import this

The Zen of Python, by Tim Peters

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Readability counts.

[...]

Question

```
>>> langs = [
   csharp,
   java,
   javascript,
   php,
   python,
   ruby
]
>>> langs = sorted(langs,
                   key=lambda l: l.year)
```

Answer

```
>>> [(l.name, l.year) for l in langs]
[
   ('python', 1991),
   ('java', 1995),
   ('javascript', 1995)
   ('php', 1995),
   ('ruby', 1995),
   ('csharp', 2000)
]
```

Django

Facts

- Born in 2005 Lawrence, KS - USA
- MTV architecture framework (==MVC)
- Django Software Foundation
- BSD license
- Used in: Pinterest, Instagram, Bitbucket, Sentry

Trivia

Batteries included

- core: ORM, urls, templates (pluggable), i18n, migrations
- contrib apps: admin, auth, sessions, postgres (e.g. jsonb field)
- tons of third party apps and libs

A (silly) Todo application

Environment setup

```
$ sudo apt-get install python
python-virtualenv
$ virtualenv venv
$ . venv/bin/activate
(venv)$ pip install django
```

Project bootstrapping

```
$ django-admin startproject
djangotcs
$ find
./djangotcs
./djangotcs/urls.py
./djangotcs/__init__.py
./djangotcs/wsgi.py
./djangotcs/settings.py
./manage.py
./db.sqlite3
```

Let's do something

```
$ manage.py startapp todo
$ mkdir -p todo/templates/todo
$ find todo/
todo/
todo/migrations
todo/migrations/__init__.py
todo/__init__.py
todo/templates
todo/templates/todo
todo/views.py
todo/models.py
todo/admin.py
todo/tests.py
```

todo/models.py

```
# -*- coding: utf-8 -*-
from django.db import models

class Todo(models.Model):
    name = models.CharField(max_length=255)

    def __unicode__(self):
        return self.name

class TodoItem(models.Model):
    text = models.TextField()
    todo = models.ForeignKey(Todo)

    def __unicode__(self):
        return self.text
```

todo/forms.py

```
# -*- coding: utf-8 -*-
from django import forms
from .models import Todo, TodoItem

class TodoForm(forms.ModelForm):
    class Meta:
        model = Todo
        fields = ('name',)

class TodoItemForm(forms.ModelForm):
    class Meta:
        model = TodoItem
        fields = ('todo', 'text',)
```

todo/views.py

```python
from django.shortcuts import render, redirect, get_object_or_404
from .models import Todo, TodoItem
from .forms import TodoForm, TodoItemForm

def index(request):
    return render(request, "todo/index.html", {'todos': Todo.objects.all()})

def detail(request, todo_id):
    todo = get_object_or_404(Todo, pk=todo_id)
    return render(request, "todo/detail.html", {'todo': todo})

def new_todo(request):
    if request.method == 'POST':
        form = TodoForm(request.POST)
        if form.is_valid():
            todo = form.save()
            return redirect('todo:index')
    else:
        form = TodoForm()
    return render(request, "todo/new.html", {'form': form})

def add_item(request, todo_id):
    pass # same pattern as new todo :)

def remove_item(request, todo_id, item_id):
    pass # not enough vertical space sorry
```

todo/templates/todo/new.html

```
{% extends 'todo/base.html' %}

{% block title %}
Add a new todo
{% endblock %}

{% block content %}
<form action="" method="POST">
{% csrf_token %}
{{ form.as_p }}
<input type="submit" value="Add">
</form>
{% endblock %}
```

todo/urls.py

```
# -*- coding: utf-8 -*-
from django.conf.urls import url
from . import views

urlpatterns = [
    url(r'^$', views.index, name="index"),
    url(r'^add$', views.new_todo, name="new_todo"),
    url(r'^(?P<todo_id>\d+)$', views.detail,
name="detail"),
    url(r'^(?P<todo_id>\d+)/item/add$', views.add_item,
name="add_item"),
    url(r'^(?P<todo_id>\d+)/item/(?
P<item_id>\d+)/remove$', views.remove_item,
name="remove_item"),
]
```

todo/admin.py

```
# -*- coding: utf-8 -*-
from django.contrib import admin
from .models import Todo, TodoItem

class TodoItemInline(admin.StackedInline):
    model = TodoItem

class TodoAdmin(admin.ModelAdmin):
    inlines = [
        TodoItemInline,
    ]

admin.site.register(Todo, TodoAdmin)
```

djangotcs/settings.py (excerpt)

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'todo',
)

ROOT_URLCONF = 'djangotcs.urls'

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

djangotcs/urls.py

```
# -*- coding: utf-8 -*-
from django.conf.urls import patterns,
include, url
from django.contrib import admin

urlpatterns = patterns('',
    url(r'^admin/',
include(admin.site.urls)),
    url(r'^todo/', include('todo.urls',
namespace="todo")),
)
```

Migration time

```
$ ./manage.py makemigrations todo
Migrations for 'todo':
  0001_initial.py:
    - Create model Todo
    - Create model TodoItem
$ ./manage.py migrate
Operations to perform:
  Apply all migrations: admin, contenttypes, auth,
sessions, todo
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying sessions.0001_initial... OK
  Applying todo.0001_initial... OK
```

built-in http server with hot code reload

```
$ ./manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).


September 16, 2015 - 22:00:50

Django version 1.8.4, using settings
'djangotcs.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CONTROL-C.
```

A more serious stack

- web server: nginx
- application server container + lot more: uwsgi
- database: postgres
- data structure store / cache: redis

Modern Times

SPA

- (too?) trivial to add rest interface to models
  https://django-rest-framework.org
- react: server side jsx compiling (via nodejs)
  https://github.com/markfinger/python-react
- angular: trivial conflict with angular default $interpolateProvider

What about the realtime web?

- python 3.5 added async / await coroutines (or gevent)
- django not async friendly (especially ORM)
- tradeoff: offload realtime to another process with redis pub/sub
- please remember: 99% non-blocking is still blocking

Age of microservices

- do we need different kind of batteries?
- easy deploy as important as dev productivity?

Resources

Python resources

- Learn python the hard way
- PythonAnywhere: try python in the cloud
- ml python-it

Django resources

- django girls tutorial
- official tutorial
- ml django-it

Todo demo app

Incomplete, pull requests welcome! :)

https://github.com/xrmx/tcs-djangodemopap

Happy Hacking :)

Riccardo Magliocchetti

riccardo@menodizero.it

@rmistaken

http://menodizero.it

https://github.com/xrmx