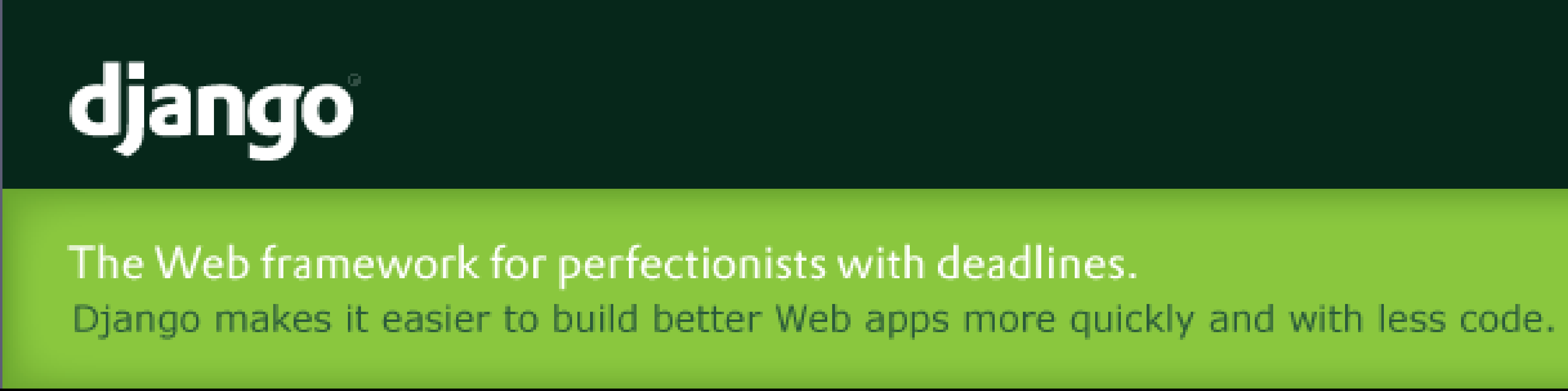
Introduction to Django Web Framework

Web application development seminar, Fall 2007
Tampere University of Technology

Jaakko Salonen <jaakko.t.salonen@tut.fi>
Jukka Huhtamäki <jukka.huhtamaki@tut.fi>


django

The Web framework for perfectionists with deadlines.

Django makes it easier to build better Web apps more quickly and with less code.

“Django – The MacGyver of Web Frameworks”

http://www.unessa.net/en/hoyci/2007/01/django-macgyver-web-frameworks/

“Django gets the big picture”

http://www.oreillynet.com/onlamp/blog/2006/07/django_gets_the_big_picture.html

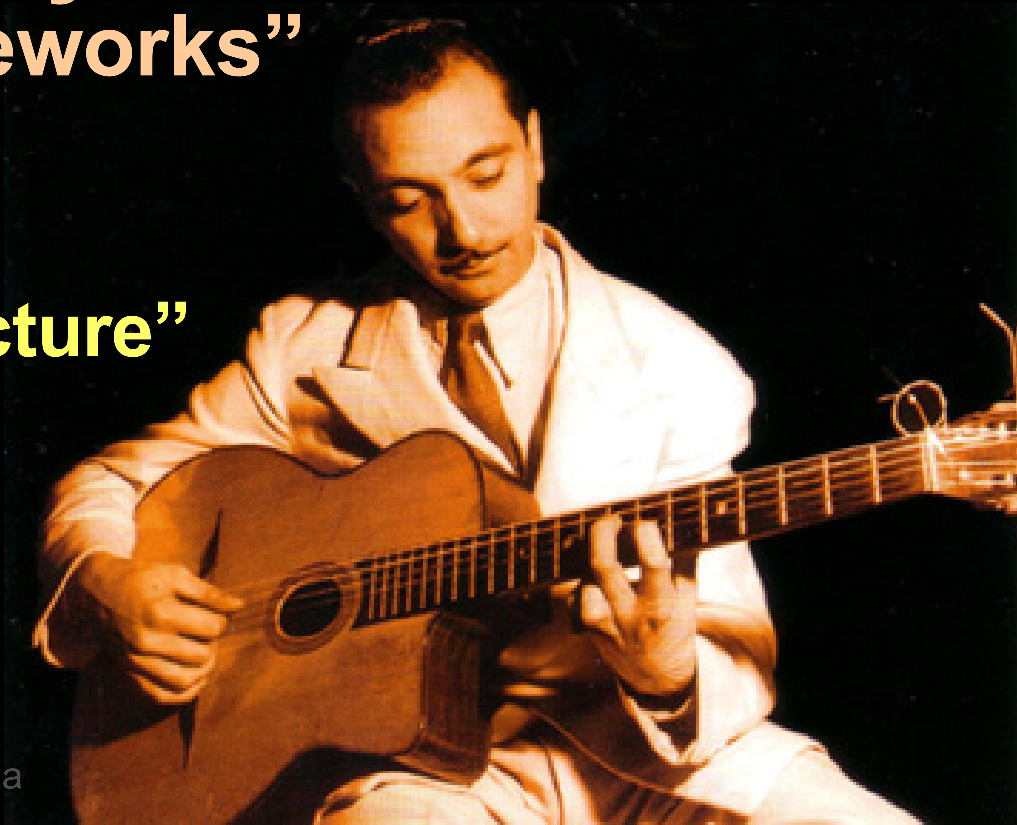


Image from: Wikipedia

In this presentation

- A short history of Django
- Key Philosophies
- Key Features
- An example: [Implementing] poll application in Django
- An example: Testing in Django
- Discussion and questions

A short history of Django

Originally developed at World Online as *a publishing framework*

Fall 2003	The developers at World Online switched from PHP to Python. Django was born.
Summer 2005	Django is open sourced, community begins to emerge
Fall 2007	Latest official version is still **0.96.1** Hundreds of sites use Django
[Near] future	”Final” 1.0 release with stabile API, along with a book (Holovathy and Kaplan-Moss 2007)

Key Philosophies

As according to (http://www.djangoproject.com/):

Loose Coupling

- Clear interfaces between different layers of the framework

- **Less code**
  - Especially by utilising Python's dynamic capabilities
- **Quick** [Web] **Development**
  - Focus on outcome, not on the details
- **Don’t Repeat Yourself**
  - Single placement for every distinct concept and/or data
- **An MTV(?) (Model-Template-View) framework**

  Rather than MVC (Model-View-Controller)

→ Emphasis on **reusability** and "**pluggability"** of components

Key Features (1/2)

As according to (http://www.djangoproject.com/):

- **Object-relational mapper (ORM)**
  For dynamic data-access API (cf. Ambler, 2006)
- **Polished administration interface for end-users**
  With configurable CRUD support (Create,Read,Update,Delete)
- **Elegant URL design**
  for parameter-free URIs hiding the technology, based on URL mapping (cf. Berners-Lee, 1998)
- **Template system**
  providing means to separate design, content and code

Key Features (2/2)

- **Built-in web application core features**
  - Authentication (session management, login/logout)
  - Authorisation (user rights and roles),
  - Multi-site support for single-source publishing and
  - Redeployable apps
- **Built-in Internationalisation (i18n) support**
  Based on ”translation strings”
- **Cache support**
  Instructions are provided for integrating Memcached (http://www.danga.com/memcached/) into a Django app
- **Built-in test framework** (doctests and unit tests)

Example: Poll application in Django (1/7)

Task: to implement a site that lets people view polls and vote in them (and manage polls)

A summary of Django's tutorial, for more comphrensive version, see http://www.djangoproject.com/documentation/tutorial01/

Example: Poll application in Django (2/7)

Step 1/3: Poll Model

polls/
    models.py
    views.py
templates/
    polls.html

```
from django.db import models

class Poll(models.Model):
    question = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
    class Admin:
        pass

class Choice(models.Model):
    poll = models.ForeignKey(Poll)
    choice = models.CharField(max_length=200)
    votes = models.IntegerField()
```

Example: Poll application in Django (3/7)

Step 2/3: Poll View (control logic for latest polls)

polls/
    models.py
    views.py
templates/
    polls.html

```
from mysite.polls.models import Poll
from django.http import HttpResponse

def index(request):
    latest_poll_list = Poll.objects.all().order_by('-pub_date')[:5]
    output = ', '.join([p.question for p in latest_poll_list])
    return HttpResponse(output)
```

Example: Poll application in Django (4/7)

Step 3/3: Poll Template (a representation)

```
polls/
        models.py
        views.py
templates/
        polls.html
```

```
{% if latest_poll_list %}
    <ul>
    {% for poll in latest_poll_list %}
        <li>{{ poll.question }}</li>
    {% endfor %}
    </ul>
{% else %}
    <p>No polls are available.</p>
{% endif %}
```

Example: Poll application in Django (5/7)

Database schema is **generated from** the model:

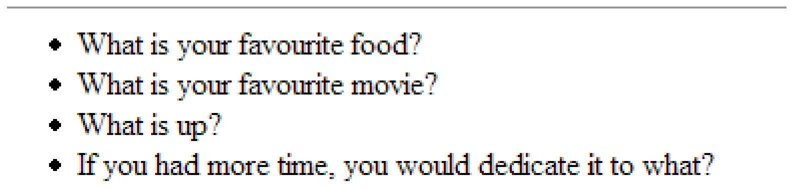
```
$ python manage.py syncdb
```

```
BEGIN;
CREATE TABLE "polls_poll" (
    "id" serial NOT NULL PRIMARY KEY,
    "question" varchar(200) NOT NULL,
    "pub_date" timestamp with time zone NOT NULL
);
CREATE TABLE "polls_choice" (
    "id" serial NOT NULL PRIMARY KEY,
    "poll_id" integer NOT NULL REFERENCES "polls_poll" ("id"),
    "choice" varchar(200) NOT NULL,
    "votes" integer NOT NULL
);
COMMIT;
```

Example: Poll application in Django (6/7)

..And test server is available instantly:

```
$ python manage.py runserver 8080
```

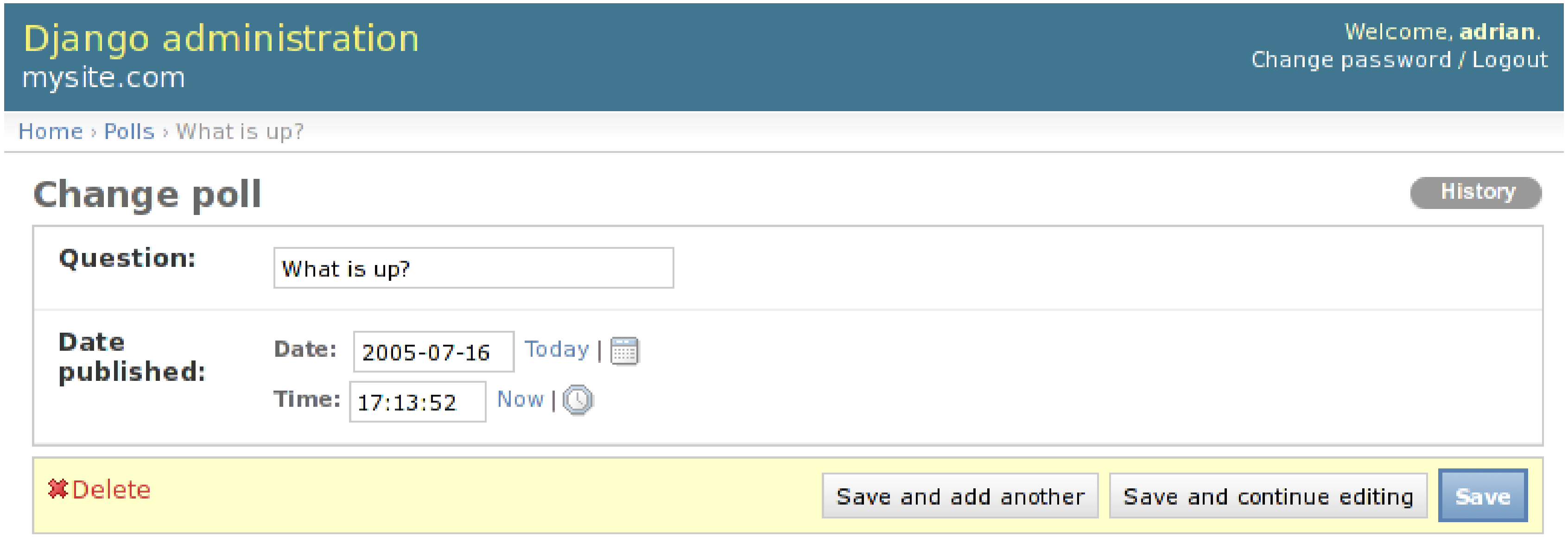


- What is your favourite food?
- What is your favourite movie?
- What is up?
- If you had more time, you would dedicate it to what?

```
<ul>
	<li>What is your favourite food?</li>
	<li>What is your favourite movie?</li>
	<li>What is up?</li>
	<li>If you had more time, you would dedicate it to what?</li>
</ul>
```

Example: Poll application in Django (7/7)

Poll administration? The user interface is readily generated!



Django administration
mysite.com

Welcome, **adrian**.
Change password / Logout

Home › Polls › What is up?

Change poll

Question: What is up?

Date published: Date: 2005-07-16 Today | Time: 17:13:52 Now |

Example: Inserting doctest to Poll Model

Doctests can be written directly to model definitions (models.py)

```
...
class Poll(models.Model):
    """
    >>> p = Poll(title=u'Your favourite movie')
    >>> p.question = 'What is your favourite movie?'
    >>> p.save()
    >>> p
    <Poll: 'Your favourite movie'>
    >>> p.question
    What is your favourite movie?
    """
...
```

Running the tests

```
$ python manage.py test
```

Discussion

- Content management is addressed by many features
  - ➔ A statement for that Django is designed for implementing content management systems

- Clear emphasis on agile web publishing
  - ...**even** in favor of framework genericity and extensibility

- Questions that have to be asked:
  - Is Django fit for Rich Internet Application (RIA) development?
  - What if database bindings are complex?
  - How much of the web application core features can be customised? (admin UI, authentication, etc.)

Questions

Question?
Comments?
Discussion!

Jaakko Salonen <jaakko.t.salonen@tut.fi>
Jukka Huhtamäki <jukka.huhtamaki@tut.fi>

References

[Ambler, 2006] “Mapping Objects to Relational Databases: O/R Mapping In Detail”.

[Berners-Lee, 1998] ”Cool URIs don't change”. Online article. Available at <http://www.w3.org/Provider/Style/URI>.

[Holovathy and Kaplan-Moss, 2007]. ”The Definitive Guide to Django: Web Development Done Right” . To be published.