

Django: Python per il web

Sporchiamoci le mani!

Di seguito è riportata la demo che è stata fatta durante la presentazione “Django: Python per il web” nel Linux Day 2008 di Perugia. L'esempio riportato è solo leggermente modificato e stato preso da <http://docs.djangoproject.com/en/dev/intro/overview/>.

Le convenzioni tipografiche utilizzate sono le seguenti:

- le righe che iniziano con il carattere \$ indicano comandi da eseguire al terminale;
- le righe che iniziano con >>> o ... indicano comandi da eseguire nella shell di Python;
- le righe scritte con carattere a spaziatura fissa che non rientrano nei casi precedenti sono codice da scrivere in un file;
- *le righe in corsivo indicano l'output di un comando eseguito nella shell di Python;*
- tutto il resto è testo aggiuntivo per commentare le operazioni fatte.

PASSO 1: CREAZIONE DI UN PROGETTO

```
$ django-admin.py startproject DemoDjango
$ cd DemoDjango
```

PASSO 2: Configurare il database (sqlite3)

```
$ vi settings.py
```

Cambiare le seguenti righe (il file database richiede un percorso assoluto):

```
DATABASE_ENGINE = 'sqlite3'
DATABASE_NAME = '/home/pippo/DemoDjango/database.db'
```

Sincronizzare il database:

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

Inserire admin/ciao come credenziali per l'amministratore.

PASSO 3: Creare l'applicazione

```
$ python manage.py startapp giornale
$ vi DemoDjango/settings.py
```

Aggiungere 'DemoDjango.giornale' a INSTALLED_APPS

PASSO 4: Creazione del modello

```
$ vi giornale/models.py
```

Inserire il seguente codice:

```
class Reporter(models.Model):
    full_name = models.CharField(max_length=70)

    def __unicode__(self):
        return self.full_name

class Article(models.Model):
    pub_date = models.DateTimeField()
    headline = models.CharField(max_length=200)
    content = models.TextField()
    reporter = models.ForeignKey(Reporter)

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

PASSO 5: Creazione delle tabelle per il modello

Quali query SQL verranno eseguite nel database per creare le tabelle di supporto alla nostra applicazione?

```
$ python manage.py sql giornale
```

Bene, eseguiamole!

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

PASSO 6: Giocare con l'API per l'accesso ai dati

Aprire una shell Python con l'ambiente già impostato per Django.

```
$ python manage.py shell
```

```
>>> from giornale.models import Reporter, Article
```

Verificare che non ci sono ancora reporter nel database:

```
.
>>> Reporter.objects.all()
[]
```

Creazione di un nuovo reporter:

```
>>> r = Reporter(full_name='James Bond')
```

Salvare il reporter sul database:

```
>>> r.save()
```

Ora il nostro reporter ha un ID:

```
>>> r.id
1
```

Ora il reporter è nel database:

```
>>> Reporter.objects.all()
[<Reporter: James Bond>]
```

Django fornisce un'API per effettuare ricerche molto completa:

```
>>> Reporter.objects.get(id=1)
<Reporter: James Bond>
```

```
>>> Reporter.objects.get(full_name__startswith='James')
<Reporter: James Bond>
```

```
>>> Reporter.objects.get(full_name__contains='Bon')
<Reporter: James Bond>
```

```
>>> Reporter.objects.get(id=2)
Traceback (most recent call last):
```

```
...
DoesNotExist: Reporter matching query does not exist.
```

L'eccezione sollevata quando non viene trovato un oggetto può essere gestita mediante i comuni meccanismi che Python mette a disposizione:

```
>>> try:
...     Reporter.objects.get(id=2)
... except:
...     print "Non trovato..."
Non trovato...
```

Creazione di un articolo:

```
>>> from datetime import datetime
>>> a = Article(pub_date=datetime.now(), headline='Linux Day
2008', content='Woohoo!', reporter=r)
>>> a.save()
```

Ora l'articolo è nel database:

```
>>> Article.objects.all()
[<Article: Linux Day 2008>]
```

Accedere al reporter di un articolo è molto semplice:

```
>>> pippo = a.reporter
>>> pippo.full_name
'James Bond'
```

Viceversa, gli oggetti di tipo Reporter possono accedere a tutti i loro articoli:

```
>>> pippo.article_set.all()
[<Article: Linux Day 2008>]
```

Si possono effettuare ricerche basate anche su campi di oggetti collegati tra loro da relazioni:

```
>>> Article.objects.filter(reporter__full_name__startswith="Ja")
[<Article: Linux Day 2008>]
```

Per modificare un oggetto basta cambiare i suoi attributi e richiamare save() per rendere le modifiche persistenti:

```
>>> r.full_name = 'Emilio Fede'
>>> r.save()
```

```
>>> Article.objects.all()[0].reporter.full_name
u'Emilio Fede'
```

Cancellazione di un oggetto con delete(): l'integrità referenziale è rispettata:

```
>>> r.delete()
>>> Reporter.objects.all()
[]
>>> Article.objects.all()
[]
```

PASSO 7: Giocare con l'interfaccia di amministrazione

```
$ vi settings.py
```

Aggiungere 'django.contrib.admin' alle INSTALLED_APPS:

```
$ python manage.py syncdb
$ vi urls.py
```

Decomentare le linee opportune, come scritto nei commenti:

```
# Uncomment the next two lines to enable the admin:
from django.contrib import admin
admin.autodiscover()

# Uncomment the next line to enable the admin:
(r'^admin/(.*)', admin.site.root),
```

Provare il server di sviluppo:

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

Aprire l'url `http://127.0.0.1:8000/admin/` e entrare come `admin/ciao`

Non ci sono tabelle su articoli e reporter, vanno abilitate nel file (da creare) `admin.py`:

```
vi giornale/admin.py
```

```
import models
from django.contrib import admin

admin.site.register(models.Article)
admin.site.register(models.Reporter)
```

Il server va riavviato: si ferma con CTRL+C e si riavvia col comando di cui sopra.

Rientrare nell'interfaccia di amministrazione e aggiungere una notizia con relativo reporter.

PASSO 8: Aggiungere gli URL necessari

Aggiungere a `urls.py` l'URL per gestire l'archivio annuale degli articoli:

```
$ vi urls.py
```

Aggiungere il seguente URL a `url_patterns`:

```
(r'^articles/(\d{4})/$', 'giornale.views.year_archive'),
```

PASSO 9: Scrivere una view di esempio

```
$ vi giornale/views.py
```

Aggiungere il seguente codice:

```
from DemoDjango.giornale.models import Article
from django.shortcuts import render_to_response

def year_archive(request, year):
    a_list = Article.objects.filter(pub_date__year=year)
    return render_to_response('news/year_archive.html', {'year':
year, 'article_list': a_list})
```

PASSO 10: Scrivere un template di esempio

Creare le directory per i template:

```
$ mkdir templates
$ mkdir templates/news
```

Abilitare la directory dei template:

```
$ vi settings.py
```

Aggiungere la directory templates a TEMPLATE_DIRS, come percorso assoluto.

```
TEMPLATE_DIRS = (  
    '/home/pippo/DemoDjango/templates'  
)
```

Creare un template base:

```
$ vi templates/base.html
```

Aggiungere il seguente codice:

```
<html>  
<head>  
    <title>{% block title %}{% endblock %}</title>  
</head>  
<body>  
      
    {% block content %}{% endblock %}  
</body>  
</html>
```

Creare il template per l'archivio annuale:

```
$ vi templates/news/year_archive.html
```

Aggiungere il seguente codice:

```
{% extends "base.html" %}  
  
{% block title %}Articoli dell'anno {{ year }}{% endblock %}  
  
{% block content %}  
<h1>Articoli dell'anno {{ year }}</h1>  
  
{% for article in article_list %}  
<hr />  
<h2>{{ article.headline }}</h2>  
<h3>Di {{ article.reporter.full_name }}</h3>  
<h4>Pubblicato il {{ article.pub_date|date:" j/m/Y" }}</h4>  
<p>{{ article.content }}</p>  
{% endfor %}  
{% endblock %}
```