Generujemy projekt

```
$ django-admin startproject przyklad
```

Dostajemy:

```
`-- przyklad
    |-- __init__.py
    |-- manage.py
    |-- settings.py
    `-- urls.py
```

Konfigurujemy bazę

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': 'default.db',
    }
}
```

Listing 1: settings.py

Konfigurujemy katalogi

```
PROJECT_PATH = os.path.abspath(os.path.dirname
    (__file__)) # dodana

MEDIA_ROOT = os.path.join(PROJECT_PATH, "
    site_media")
MEDIA_URL = '/site_media/'

TEMPLATE_DIRS = (
     os.path.join(PROJECT_PATH, 'templates'),
)
```

Listing 2: settings.py

Tworzymy appkę

```
$ ./manage.py startapp blog
```

Dostajemy:

```
`-- blog
    |-- __init__.py
    |-- models.py
    |-- tests.py
    `-- views.py
```

Piszemy pierwszy model

```
class Post(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField()
```

Listing 3: blog/models.py

i synchronizujemy bazę:

```
$ ./manage.py syncdb
```

Więcej o modelach

- (wielo-)dziedziczenie modeli
- modele abstrakcyjne
- asocjacje
- sygnały

Operacje na modelach

- __init__
- self.save()
- self.delete()
- cls.objects

Manager

- all()
- get()
- filter()
- exclude()
- order_by()

Ładny model

Zawiera:

- pola
- klasę Meta
- metodę __unicode__ lub __str__
- metodę get_absolute_url

Widok

```
def index(request):
    posts = Post.objects.all()
    return render_to_response("blog/index.html",
        {'posts': posts},
        context_instance=RequestContext(request))
```

Listing 4: blog/views.py

Więcej o widokach

- request
- argumenty widoku
- middleware

URL

```
urlpatterns = patterns('',
    (r'^blog/', include('blog.urls', namespace='
        blog')),
)
```

Listing 5: urls.py

```
urlpatterns = patterns('blog.views',
    url(r'^$', 'index', name='index'),
)
```

Listing 6: blog/urls.py

Nasze dzieło

```
$ ./manage.py runserver
```

I tak dochodzimy do pierwszego błędu :)

Template

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

Listing 7: templates/base.html

Template właściwy

```
{% extends "base.html" %}

{% block content %}
  <h2>Notki</h2>
  {% for post in posts %}
    <h3>{{ post.title }}</h3>
    <div>{{ post.content }}</div>
  {% empty %}
    <h3>Brak.</h3>
  {% endfor %}
{% endblock %}
```

Listing 8: templates/blog/index.html

Template tagi

- dodawanie logiki do widoku
- for, if, url, block, extends
- można pisać własne (load)

Filtry

- przetwarzanie zmiennych kontekstu
- użycie: zmienna—filtr

Podsumowanie templatów

- logika ograniczona do minimum
- dziedziczenie

Panel administracyjny

- dołączamy django.contrib.admin do naszych appek (w settings.py)
- dołączamy panel do naszych urls.py
- definiujemy co chcemy dołączyć do panelu

Co ma być w panelu?

```
class PostAdmin(admin.ModelAdmin):
    list_display = ('title', )

admin.site.register(Post, PostAdmin)
```

Listing 9: blog/admin.py

Wyświetlanie pojedynczego posta - widok

```
def show(request, post_id):
    post = Post.objects.get(pk=post_id)
    return render_to_response("blog/show.html",
        {'post': post},
        context_instance=RequestContext(request))
```

Listing 10: blog/views.py

Wyświetlanie pojedynczego posta - template i url

Drobna zmiana w blog/index.html:

```
<h3><a href="{% url blog:show post.pk %}">{{
    post.title }}</a></h3>
```

Listing 11: templates/blog/index.html

Zmiana w blog/urls.py:

```
url(r'^(?P<post_id>\d+)$', 'show', name='show'
    ),
```

Listing 12: blog/urls.py

Co warto jeszcze poznać?

- forms
- generic views
- cache
- i18n
- sesje

Pomoce i inspiracje

- DjangoProject.com
- DjangoBook.com
- DjangoAdvent.com
- b-list.org
- DjangoPony.com