De cero a Django en media hora

Matías Herranz


django

Algunos conceptos básicos

¿Qué es Django?

- Django es un `framework`, pensado, nacido y evolucionado para el desarollo de aplicaciones web.
- Agiliza o resuelve tareas repetitivas
- Soluciona problemas o requerimientos frecuentes
- Rápido, limpio

Algunos conceptos básicos

¿Para qué se usa?

Aplicaciones web de los más variados sabores y colores:

- Blogs
- Sitios de compras(eCommerce)
- dpaste.
- Chicago Crime
- Washington Post
- NASA
- Juegos online como SteamEnd(Por JuanBC)
- Larga lista de etcéteras.

En general en la vida y en particular, por si vienen zombies....


CAUTION!
ZOMBIES!
AHEAD!!!

- ¡Hay que estar preparados!


IN CASE OF
ZOMBIES
BREAK GLASS
Remington

¡Así que!

Veamos un poco de qué se trata Django:

Allá lejos y hace tiempo, por el año 2003, en el "Lawrence Journal World", en Kansas

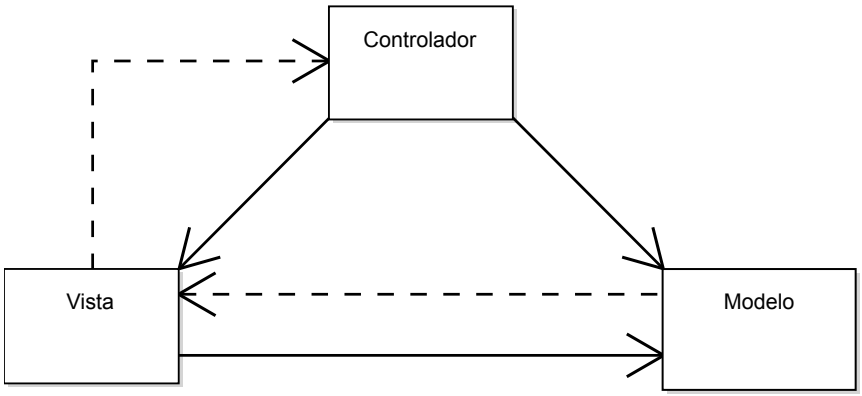
- Repetición de código
- Pedazos de código que podían reutilizar entre proyectos
- Resolver los mismos problemas una y otra vez
- Juntaron todo, enceraron, pulieron, le pusieron nombre y lo liberaron en 2005

Siga contando, nomás!

Patrón de diseño: MVC (Model-View-Controller)

Patrón de arquitectura de software que separa los **datos**, la **interfaz de usuario** y la **lógica de control** de una aplicación.

- **Modelo:** Esta es la ***representación de la información*** con la cual el sistema opera.
- **Vista:** Este presenta el modelo en un formato adecuado para interactuar, usualmente la ***interfaz de usuario***.
- **Controlador:** Este responde a ***eventos***, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista.



MVC módulo Django:

¿isomorfismo?

MVC aplicado a Django

Model:

- Representado en Django por los ***models*** (en el respectivo models.py de cada app)
- Clases de Python
- Modelan los datos

View:

- representado en Django por los ***templates***
- forma de visualizar los datos
- HTML, CSS, JS, Django templates

Controller

- Representado por las ***views***
- Procesa los datos ingresados por el usuario
- Lógica de la aplicación
- Procesamiento de los datos a renderizar mediante los templates

Habiendo hablado suficiente....

- **Un ejemplo, por favor!**

Hagamos un pequeño Twitter

Instalamos Django:

```
1 MacBookPro:PyDay2011 matias$ pip install django
```

Creamos el proyecto:

```
1 MacBookPro:PyDay2011 matias$ mkdir TwitterClone
2 MacBookPro:PyDay2011 matias$ cd TwitterClone/
3 MacBookPro:TwitterClone matias$ django-admin.py startproject twitterclone
```

Creamos una app:

```
1 MacBookPro:TwitterClone matias$ cd twitterclone/
2 MacBookPro:twitterclone matias$ python manage.py startapp tweets
```

Hagamos un pequeño Twitter

Los archivos que nos crea el comando django-admin.py tiene rol función muy bien definido:

- ***init.py:*** este archivo no está relacionado con Django particularmente, sino que es más específico de Python e indica que el directorio actual es un módulo.
- ***manage.py:*** se utiliza para interactuar con el proyecto en particular. Es similar a django-admin.py pero específico de este proyecto. Mediante este programa se puede actualizar la base de datos, ejecutar el servidor de desarrollo, correr tests, etc.
- ***settings.py:*** en este archivo se encuentra toda la configuración del proyecto en general; la conexión a la base de datos, qué tipo de base usar, en qué lenguaje está el sitio, etc. Observar que este archivo de configuración también es código Python.
- ***urls.py:*** en este archivo se encuentran todas las url's disponibles para nuestro sitio. La prolijidad que permite Django en el manejo de urls(con expresiones regulares, nombres, etc) es uno de los puntos fuertes del framework.

Pequeño Twitter: models

TwitterClone/twitterclone/tweets/models.py:

```
 1 # -*- coding: utf-8 -*-
 2 import datetime
 3 from django.db import models
 4 from django.contrib.auth.models import User
 5
 6 class Tweet(models.Model):
 7     """
 8     Model que representa los tweets.
 9     """
10     user = models.ForeignKey(User)
11     message = models.CharField(max_length=140)
12     date_posted = modelst.DateTimeField(default=datetime.datetime.now)
13
14     def __unicode__(self):
15         return "Tweet from: %s, Txt: %s" % (self.user.username, self.message,)
```

Pequeño Twitter: views

TwitterClone/twitterclone/tweets/views.py:

```
1 # -*- coding: utf-8 -*-
2 from models import Tweet
3 from django.shortcuts import render_to_response
4
5 def show_timeline(request):
6     return render_to_response('twitter/timeline.html',
7                       {'tweets': Tweet.objects.order_by('-date_posted')},
8             )
```

Pequeño Twitter: template

```
 1 <html>
 2 <head>
 3     <title>Tweets!</title>
 4 </head>
 5 <body>
 6     <h1>Tweeter Clone Timeline</h1>
 7     <div id="container">
 8         {% for tweet in tweets %}
 9         <p> <b>[Tweet id:{{ tweet.id }} | {{ tweet.user.username }}]</b> --> {{ tweet.messa
10         {% endfor %}
11     </div>
12 </body>
13 </html>
```

Pequeño Twitter: urls

TwitterClone/twitterclone/urls.py:

Agregamos una url para nuestra view:

```
1 url(r'^$', 'tweets.views.show_timeline'),
```

Pequeño Twitter: settings

TwitterClone/twitterclone/settings.py:

Agregamos la app que creamos dentro de INSTALLED_APPS, en settings.py:

```
 1 INSTALLED_APPS = (
 2     'django.contrib.auth',
 3     'django.contrib.contenttypes',
 4     'django.contrib.sessions',
 5     'django.contrib.sites',
 6     'django.contrib.messages',
 7     'django.contrib.staticfiles',
 8     # Uncomment the next line to enable the admin:
 9     'django.contrib.admin',
10     # Uncomment the next line to enable admin documentation:
11     # 'django.contrib.admindocs',
12     'tweets'
13 )
```

Pequeño Twitter: settings

Configuramos la base de datos con sqlite:

```
 1 DATABASES = {
 2     'default': {
 3         'ENGINE': 'django.db.backends.sqlite3',
 4         'NAME': 'bd.sqlite3',
 5         'USER': '',
 6         'PASSWORD': '',
 7         'HOST': '',
 8         'PORT': '',
 9     }
10 }
```

Pequeño Twitter: admin

TwitterClone/twitterclone/tweets/admin.py:

Para que podamos acceder a los tweets desde el panel de administración(para borrarlos, editarlos, crear nuevos, listarlos), debemos registrar el model en el archivo admin.py:

```
1 from django.contrib import admin
2 from models import Tweet
3
4 class TweetAdmin(admin.ModelAdmin):
5     list_display = ('user', 'message', 'date_posted',)
6     search_fields = ('user__username', 'message', 'date_posted',)
7
8 admin.site.register(Tweet, TweetAdmin)
```

Habilitamos el admin:

En settings.py, descomentamos esta línea:

```
1 # 'django.contrib.admin',
```

En urls.py, descomentamos las siguientes líneas:

```
1 # from django.contrib import admin
2 # admin.autodiscover()
```

y esta línea:

```
1 url(r'^admin/', include(admin.site.urls)),
```

Pequeño Twitter

La estructura de archivos queda así:

```
MacBookPro:TwitterClone matias$ tree
.
└── twitterclone
    ├── __init__.py
    ├── bd.sqlite3
    ├── manage.py
    ├── settings.py
    ├── tweets
    |   ├── __init__.py
    |   ├── admin.py
    |   ├── models.py
    |   ├── templates
    |   |   └── tweets
    |   |       └── timeline.html
    |   ├── tests.py
    |   └── views.py
    └── urls.py

4 directories, 11 files
```

Pequeño Twitter

¡Hora de hacerlo andar!

Sincronicemos nuestros models a la base de datos:

```
1 MacBookPro:twitterclone matias$ python manage.py syncdb
```

Creemos unos cuantos tweets:

```
1 usr = User.objects.all()[0]
2
3 for i in range(0, 30):
4     tw = Tweet()
5     tw.user = usr
6     tw.message = unicode(i)
7     tw.save()
```

Shine on you, crazy.... "twitterclone" :-P

```
1 MacBookPro:twitterclone matias$ python manage.py runserver
```

Algunas dulzuras de Django:

Forms:

- Django resuelve de manera simpática y prolija el manejo de forms.
- Form from models

Ejemplo:

```
 1 from django.forms import ModelForm
 2
 3 # Create the form class.
 4 class ArticleForm(ModelForm):
 5     class Meta:
 6     model = Article
 7
 8 # Creating a form to add an article.
 9 form = ArticleForm()
10
11 # Creating a form to change an existing article.
12 article = Article.objects.get(pk=1)
13 form = ArticleForm(instance=article)
```

Algunas dulzuras de Django:

Validación:

- En los models, para cada campo
- Lista de funciones(validators).
- Cada validator hace un raise de ValidationError si el valor que tomó no cumple cierto criterio.

Ejemplo:

```
 1 from django.core.exceptions import ValidationError
 2 def validate_even(value):
 3     if value % 2 != 0:
 4         raise ValidationError(u'%s is not an even number' % value)
 5
 6 # forms.py
 7 from django import forms
 8 class MyForm(forms.Form):
 9     even_field = forms.IntegerField(validators=[validate_even])
10
11 # models.py
12 from django.db import models
13 class MyModel(models.Model):
14     even_field = models.IntegerField(validators=[validate_even])
```

Algunas dulzuras de Django:

ORM: Agregación:

El ORM de Django tiene poderes de super vaca.

- Entre otra cosas súper interesantes, permite usar agregación.
- Funciones de agregación: Avg, Max, Min, Count

Ejemplo:

```
 1 from django.db.models import Avg
 2
 3 class Book(models.Model):
 4    isbn = models.CharField(max_length=9)
 5    name = models.CharField(max_length=300)
 6    price = models.DecimalField(max_digits=10, decimal_places=2)
 7    authors = models.ManyToManyField(Author)
 8
 9 from django.db.models import Avg
10 Book.objects.all().aggregate(Avg('price'))
11 >>> {'price__avg': 34.35}
```

Algunas dulzuras de Django:

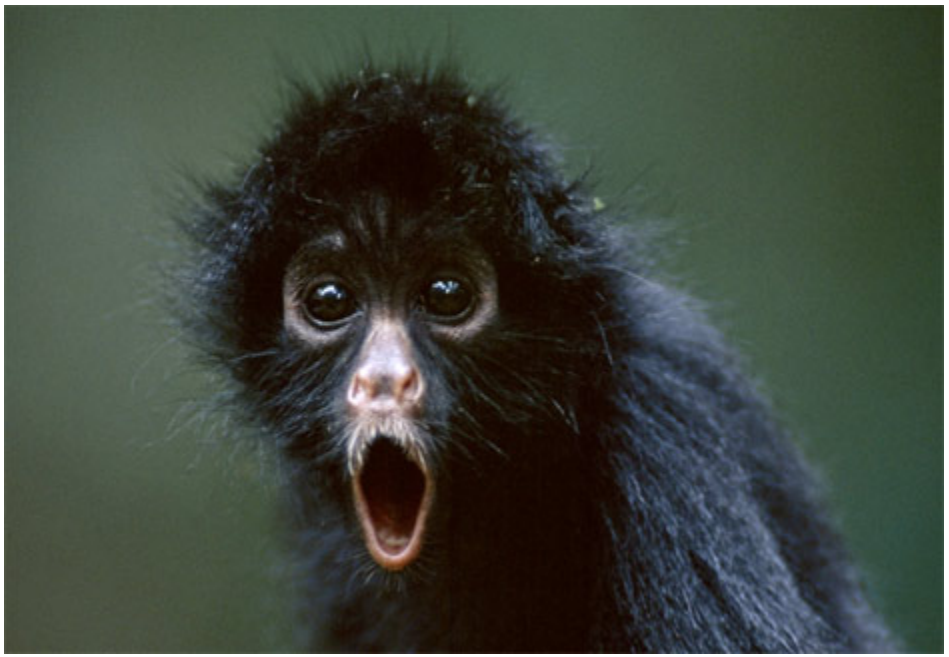
Manejo de fechas:

- Cómo ser feliz con DateFormat o "Todo lo que siempre quizo poder hacer y siempre se quejó de tener que hacer parcialmente a mano con fechas"

Ejemplo:

```
 1 from django.utils.dateformat import DateFormat
 2 import datetime
 3
 4 d = datetime.datetime.now()
 5 df = DateFormat(d)
 6
 7 print df.format('jS F Y H:i')
 8 # 7th October 2003 11:39
 9
10 print df.format('l F jS Y @ H:iA')
11 # Saturday April 30th 2011 @ 07:15AM
```

That's all, folks!



¿Preguntas? ¿Comentarios?

De cero a Django en 30 minutos

Ahora que parecen estar de moda los QR-Codes

Acá les va uno con la url de mi blog:

[http://scoobygalletas.blogspot.com/]

Matías Herranz [scoobygalletas@gmail.com]

El material de la charla, código incluído, lo voy a subir a mi blog/GitHub.

¡Gracias!

De cero a Django en 30 minutos

Referencias útiles:

- Web oficial del proyecto Django: http://www.djangoproject.com/
- http://docs.djangoproject.com/en/dev/topics/db/aggregation/
- http://docs.djangoproject.com/en/dev/ref/validators/
- http://docs.djangoproject.com/en/dev/topics/forms/modelforms/
- http://docs.djangoproject.com/en/dev/topics/db/aggregation/
- Lista de correo: http://groups.google.com/group/django-es
- Libro en español: http://trac.usla.org.ar/django-book
- Canal de IRC: #django-es en irc.freenode.net
- Colaboración de Manuel Kaufmann(a.k.a. "Humitos"), un tipazo y un grosso de Paraná.