Django Framework

José Ortiz Bejar

job@correo.fie.umich.mx

Universidad Michoacana de San Nicolás de Hidalgo

10 de septiembre de 2014

Programando al viejo estilo

```
#!/usr/bin/env python

import MySQLdb

print "Content-Type: text/html\n"
print "<html><head><title>Books</title></head>"
print "<body>"
print "<h1>Books</h1>"
print "<ul>"

connection = MySQLdb.connect(user='me', passwd='letmein', db='my_db')
cursor = connection.cursor()
cursor.execute("SELECT name FROM books ORDER BY pub_date DESC LIMIT 10")

for row in cursor.fetchall():
    print "<li>%s</li>" % row[0]

print "</ul>"
print "</body></html>"

connection.close()
```

¿Porque es necesario un framework?

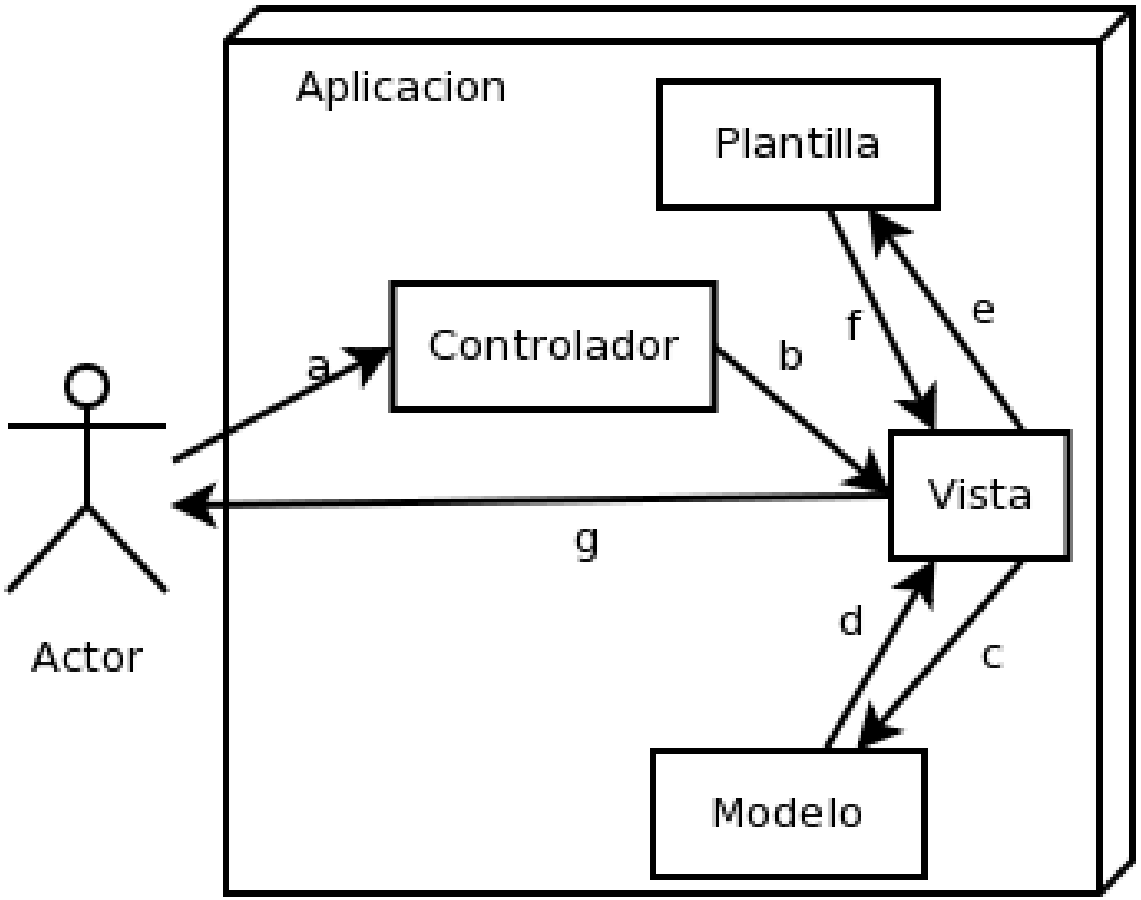
- Se tiene la necesidad de que múltiples módulos de una aplicación se conecten a la base de datos
- Es necesario mezclar lenguajes
- Reusar módulos para diferentes bases de datos
- Separar diseño y desarrollo

¿Que es Django?

Un conjunto de librerías que provee una infraestructura para el desarrollo de aplicaciones

- Permiten escribir programas limpios
- Facilitan el mantenimiento

MVC en Django



Modelo

models.py contiene la descripción de la base de datos, representada como una clase de python.

```
# models.py (descripción de la base de datos)

from django.db import models

class Book(models.Model):
    name = models.CharField(max_length=50)
    pub_date = models.DateField()
```

Modelo

- Definir los objetos de la base de datos
- Establecer las relaciones entre los objetos
- Permite acceder a los datos (leer, crear, modificar y borrar)

Vista

view.py contiene la lógica del sistema de información

```
# views.py (Lógica del sistema)

from django.shortcuts import render_to_response
from models import Book

def latest_books(request):
    book_list = Book.objects.order_by('-pub_date')[:10]
    return render_to_response('latest_books.html', {'book_list': book_list})
```

Vista

view.py contiene la lógica del sistema de información

- Ejecuta las operaciones necesarias para devolver el contenido html
- Implementa la lógica del sistema.
- Interactúa con el modelo para extraer información de la BD

Controlador

urls.py Define como se relacionan las urls con las vistas.

```
# urls.py (the URL configuration)

from django.conf.urls.defaults import *
import views

urlpatterns = patterns('',
    (r'^latest/$', views.latestbooks),
)
```

Plantilla

urls.py Define como se relacionan las urls con las vistas.

```
# latest_books.html (the template)

<html><head><title>Books</title></head>
<body>
<h1>Books</h1>
<ul>
{% for book in book_list %}
<li>{{ book.name }}</li>
{% endfor %}
</ul>
</body></html>
```

Enfoque MVC

Django se apega al enfoque MVC.

- **M** El acceso a los datos, es manejado por la capa de base de datos de Django (Modelos).
- **V** La parte en que se selecciona cuales y como se muestran los datos. Vistas y plantillas.
- **C** Selecciona que vista utilizar de acuerdo con la entrada del usuario. La configuración de urls mapea a métodos Python.

Enfoque MTV

Debido a que la 'C' ya es manejada por el framework. En Django sería mas apropiado hablar de un desarrollo MTV.

- **M** Se sigue utilizando para la capa de datos.
- **T** Se refiere a las plantillas (Template), la capa de presentación de los datos.
- **V** La capa de lógica del sistema.

Instalación

- Descargar y extraer Django-x.x.gz
- cd Django-x-x
- python setup.py install

Probar la instalación

En ms dos/terminal

```
python
>>> import django
>>> django.VERSION
(1, 4, 0, 'final', 0)
```

Habilitando el motor de bases de datos

Django soporta cuatro motores de bases de datos

- PostgreSQL (http://www.postgresql.org/)
- SQLite 3 (http://www.sqlite.org/)
- MySQL (http://www.mysql.com/)
- Oracle (http://www.oracle.com/)

SQlite

En nuestro caso usaremos SQlite, para las versiones 2.5 y mayores Python integra soporte para SQlite.

- La base de datos es un solo archivo
- No se requiere un servidor
- No es necesario hacer instalaciones adicionales

Agregar comandos django al path (solo usuarios windows)

- click derecho en Equipo(o y Mi PC)
- Pestaña de opciones avanzadas
- Variables de entorno
- editar la variable *path y agregar al final* `;path_to_python\lib\site-packages\django\bin` *donde path_to_python es la ruta donde quedo instalado python (e.j.* `c:\python`*).*

Iniciar un proyecto

En ms dos/terminal

```
django-admin.py startproject misitio
```

Esto crea un directorio con la siguiente estructura

```
misitio/
    manage.py
    misitio/
        __init__.py
        settings.py
        urls.py
```

Estructura del proyecto

__init__.py Es un archivo vacío el cual se utiliza para que python trate la aplicación como un paquete. Generalmente no es necesario editarlo.

manage.py Utilidad de línea de comandos que nos permite interactuar con el proyecto.

settings.py La configuración de proyecto Django. Se encuentra pre-editado con valores por defecto.

urls.py Se utiliza para manejar las URLs del proyecto. Como una especie de tabla de contenido. Inicialmente vacío.

Servidor de desarrollo

Django incluye un servidor web ligero, lo cual permite desarrollar sin necesidad de configurar un servidor web.

```
python manage.py runserver
```

```
Validating models...
0 errors found.

Django version 1.0, using settings 'mysite.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

La primera Vista

La primera meta es escribir una página web que diga: Hola Mundo.
La función debe escribirse en *views.py*

```
from django.http import HttpResponse
def hello(request):
    return HttpResponse('Hola Mundo')
```

- *Es necesario importar la clase* HttpResponse que es parte del modulo *django.http*
- *Se define una función* hello (función de vista)
- Cada función toma al menos un argumento llamado *request por convención. Contiene información de la petición web.*
- *La función regresa un objeto* django.http.HttpRequest con el texto 'Hola Mundo'

URLconf

```
from django.conf.urls.defaults import *

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

urlpatterns = patterns('',
    # Example:
    # (r'^mysite/', include('mysite.foo.urls')),

    # Uncomment the admin/doc line below and add 'django.contrib.admindocs'
    # to INSTALLED_APPS to enable admin documentation:
    # (r'^admin/doc/', include('django.contrib.admindocs.urls')),

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

La primera URLconf

Para agregar una vista a URLconf, es necesario poner un tuple que mapea una patrón de URL a una función de vista.

```
from django.conf.urls.defaults import *
from mysite.views import hello

urlpatterns = patterns('',
    ('^hello/$', hello),
)
```

- Primero se importa la función *hello de la vista – misitio/views.py se traduce como misitio.views en la sintaxis de import de python.*
- *Se agrega la línea (*`'^hello/$'`*,hello), a* urlpatterns. El primer elemento es una expresión regular y el segundo es una función de la vista.