Django Framework

José Ortiz Bejar

job@correo.fie.umich.mx

Universidad Michoacana de San Nicolás de Hidalgo

12 de septiembre de 2014

Configuración de la base de datos

Editar *nombre_del_proyecto/settings.py. Modificar los valores de las siguientes claves en* DATABASE 'defaults':

ENGINE Puede ser '*django.db.backends.postgresql_psycopg1*', 'django.db.backends.mysql','*django.db.backends.oracle' o como en nuestro caso* 'django.db.backends.sqlite3

NAME EL nombre de la base de datos. Si se utiliza sqlite la ruta del archivo (debe ser una ruta absulota). Sino existe el archivo es creado.

Configuración de la base de datos

USER	Usuario de la DB (No es necesario para SQLite)
PASSWORD	Password del usuario de la DB (No es necesario para SQLite)
HOST	Host donde se encuetra la base de datos(se deja en blanco si es el mismo de la aplicación)

Aplicaciones por defecto

django.contrib.auth Sistema de autenticación.

django.contrib.contenttypes Tipos de contenidos del framework

django.contrib.sessions Framework para manejo de sesiones

django.contrib.sites Framework para el manejo de múltiples sitios con instaliónde Django única.

django.contrib.messages Framework para manejo de mensajes.

django.contrib.staticfiles Framework para manejo de archivos estáticos

Aplicaciones por defecto

Cada aplicación hace uso de al menos una tabla de la BD, por ello es necesario crear las tablas antes de poder utilizarlas. Se ejecuta el siguiente comando

```
python manage.py syncdb
```

El comando syncdb revisa el parámetro INSTALLED_APPS y crea las tablas necearías para que cada una de las aplicaciones opere correctamente.

Proyecto y apps

Proyecto Es un conjunto de aplicaciones y configuraciones para un sitio web.

App Es una aplicación web que se encarga de alguna tarea (e.j. un weblog, una encuesta, etc.)

Crear una Aplicación

```
python manage.py startapp encuestas
```

Lo anterior creará un directorio encuestas con la siguiente estructura:

```
encuestas/
        __init__.py
        models.py
        tests.py
        views.py
```

Crear un modelo

El primer paso para crea una aplicación web que incluye BD es crear los modelos. Creamos dos modelos editando *encuestas/models.py y agregando lo siguiente:*

```
class Encuesta(models.Model):
    pregunta = models.CharField(max_length=200)
    fecha_pub = models.DateTimeField('Fecha de publicacion')

class Opcion(models.Model):
    encuesta = models.ForeignKey(Encuesta)
    opcion = models.CharField(max_length=200)
    votos = models.IntegerField()
```

Modelos

- Cada modelo es representado por una clase que hereda a *django.db.models.Model. Cada modelo tiene un grupo de variables de clase, de los cuales cada una representa un campo en la DB.*
- *Cada campo es representado por una instancia de la clase* Field (e.g. *CharField para campos de caracteres y* datetimes para fechas).
- El nombre de cada instancia de *Field es el nombre del campo en la BD.*

Modelos

- Algunos *Field requieren parámetros, por ejemplo* CharField, requiere que se le proporcione *max_length (utilizado en la DB y para validación).*
- *Note que se define una relación, usando* ForeignKey. Esto le dice a Django que cada opción está ligada a una sola encuesta.

Activar Modelos

Con la descripción de los modelos podemos realizar las siguientes acciones:

- Crear el esquema de la base de datos para la aplicación
- Crear un API para el acceso a la base de datos mediante los objetos Encuesta y Opcion.

Pero primero es necesario instalar la *App Encuestas*

Activar Modelos

Editamos *settings.py, y agregamos a INSTALLED_APPS la cadena 'encuestas'.*

```
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Uncomment the next line to enable the admin:
    # 'django.contrib.admin',
    # Uncomment the next line to enable admin documentation:
    # 'django.contrib.admindocs',
    'encuestas',
```

Activar Modelos

Ahora Django sabe de la existencia de *encuestas. Ejecutar el siguiente comando*

```
python manage.py sql encuestas

BEGIN;
CREATE TABLE "encuestas_encuesta" (
    "id" integer NOT NULL PRIMARY KEY,
    "pregunta" varchar(200) NOT NULL,
    "fecha_pub" datetime NOT NULL
)
;
CREATE TABLE "encuestas_opcion" (
    "id" integer NOT NULL PRIMARY KEY,
    "encuesta_id" integer NOT NULL REFERENCES "encuestas_encuesta" ("id"),
    "opcion" varchar(200) NOT NULL,
    "votos" integer NOT NULL
)
;
COMMIT;
```

Activar Modelos

- La salida exacta depende del motor de base de datos utilizado
- Los nombres de las tablas combinan el nombre de la aplicación con el nombre del modelo (aplicacion_modelo).
- Claves primarias se generar automáticamente
- Por convención Django agrega _id a los nombres de las claves foráneas.

Activar Modelos

- Las calves foráneas están ligadas con la sentencia REFERNCES.
- Están ligadas con el motor de base de datos (auto_increment para MySQL, serial en posgres , etc)
- Claves primarias se generar automáticamente
- El comando *sql no ejecuta el código SQL en la BD.*

Activar Modelos

python manage.py validate Verifica si hay errores en los modelos.

python manage.py sqlcustom encuestas Muestra las sentencias SQL definidas para la aplicación (tales como modificaciones de tablas o constraints).

python manage.py sqlclear encuestas Genera las sentencias DROP TABLE para el app enuestas

python manage.py sqlindexes encuestas Genera las sentencias CREATE INDEX para ésta aplicación.

python manage.py sqlall encuestas Una combinación de sql, sqlcustom, y sqlindexes.

Activar Modelos

Ejecutar

```
python manage.py syncdb
```

El comando *syncdb ejecuta el código SQL generado por* sqlall en la BD para todas la apps en INSTALLED_APPS que aún no han sido creadas.

Interacción con el API

Para interactuar con el API es posible hacerlo desde el interprete de python. Ejecute el siguiente comando:

```
python manage.py shell
```

Una vez en el shell para explorar el API

```
# Importar las clase para los modelos que acabamos de crear
>>> from encuestas.models import Encuesta, Opcion

# Verificar que aun no hay encuestas
>>> Encuesta.objects.all()
[]

# Crear una nueva encuesta.
# La configuración default incluye soporte para zonas horarioas,
# Django espera un datetime tzinfo para fecha_pub. Utilizar timezone.now()
# en lugar de datetime.datetime.now().
>>> from django.utils import timezone
>>> p = Encuesta(pregunta="Que hay de nuevo?", fecha_pub=timezone.now())

# Para salvar el objeto en la BD. Se tiene que hacer un save()
#explicito.
>>> p.save()
```

Interacción con el API

```
# Verificar el id asignado
>>> p.id
1
# Acceder a las columnas de la BD mediante atributos Python
>>> p.pregunta
"Que hay de nuevo?"
>>> p.fecha_pub
datetime.datetime(2012, 2, 26, 13, 0, 0, 775217, tzinfo=<UTC>)

# Actualizar valores cambiando atributos, después ejecutar save().
>>> p.pregunta = "Que onda?"
>>> p.save()

# objects.all() muestra todas las encuestas en la BD
>>> Encuesta.objects.all()
[<Encuesta: Encuesta object>]
```

Interacción con el API

```
[<Encuesta: Encuesta object>]
```

es una representación poco entendible para un objeto encuesta. Para mejorar es necesario agregar el método __unicode__ a cada modelo.

```
class Encuesta(models.Model):
    # ...
    def __unicode__(self):
        return self.pregunta

class Opcion(models.Model):
    # ...
    def __unicode__(self):
        return self.opcion
```

Interacción con el API

En el shell probar los siguiente:

```
>>> from encuesta.models import Encuesta, Opcion

# Probar que __unicode__() funciona.
>>> Encuesta.objects.all()
[<Encuesta: Qui onda?>]

# Django provee una gran variedad de facilidades de búsqueda
# guiadas por argumentos keyword
>>> Encuesta.objects.filter(id=1)
[<Encuesta: What's up?>]
>>> Encuesta.objects.filter(pregunta__startswith='Qui')
[<Encuesta: Qui onda?>]

# Obtener todas las encuestas publicadas en 2012
>>> Encuesta.objects.get(fecha_pub__year=2012)
<Encuesta: Qui onda?>

>>> Encuesta.objects.get(id=2)
Traceback (most recent call last):
    ...
DoesNotExist: Encuesta matching query does not exist.

# Buscar por clave primaria es una tarea común
# Lo siguente es equivalente a  Encuesta.objects.get(id=1).
>>> Encuesta.objects.get(pk=1)
<Encuesta: Qui onda?>
```

Interacción con el API

Es posible agregar métodos personalizados.

```
import datetime
from django.utils import timezone
# ...
class Encuesta(models.Model):
    # ...
    def publicada_recientemente(self):
        return self.fecha_pub >= timezone.now() - datetime.timedelta(days=1)
```

Interacción con el API

En el shell probar los siguiente:

```
>>> from encuesta.models import Encuesta, Opcion
# Probar el método personalizado
>>> p = Encuesta.objects.get(pk=1)
>>> p.publicada_recientemente()
True

# Dada una Encuesta y un par de Opciones. Una llamada a create
# construye un objeto Opcion, la sentencia INSERT, agrega una opción
# al conjunto de opciones disponibles y regresa un objeto opcion.
# Django crea un conjunto para mantener la relación de la claves
# foráneas (e.g. encuestas-opciones) el cual puede ser accedido por el API.
>>> p = Encuestas.objects.get(pk=1)

# Muestra cualquier opción del conjunto relacional.
>>> p.opcion_set.all()
[]

# Crear tres opciones.
>>> p.opcion_set.create(opcion='Aqui nomas', votos=0)
<Opcion: Aqui nomas>
>>> p.opcion_set.create(opcion='Sin novedad', votos=0)
<Opcion: Sin novedad>
>>> c = p.opcion_set.create(opcion='Pasando el rato', votos=0)
```

Interacción con el API

```
# Mediante el  API los objetos Opcion pueden acceder a sus
# objetos Encuesta relacionados.
>>> c.encuesta
<Encuesta: Qui onda?>

#  Y vice versa: Los objetos Encuesta tienen accesos a los objetos Opcion.
>>> p.opcion_set.all()
[<Opcion: Aqui nomas>, <Opcion: Sin novedad>, <Opcion: Pasando el rato>]
>>> p.opcion_set.count()
3

# El API sigue automáticamente las relaciones.
# Usar doble guión bajo para  separar relaciones.
# Lo anterior funciona tantos niveles como sea necesario
# Encontrar  todas las opciones de cualquier encuesta de las cuales su
# fecha_pub sea ne 2012.
>>> Opcion.objects.filter(Encuesta__fecha_pub__year=2012)
[<Opcion: Aqui nomas>, <Opcion: Sin novedad>, <Opcion: Pasando el rato>]

# Para borrar use delete()
>>> c = p.opcion_set.filter(opcion__startswith='Pasando el rato')
>>> c.delete()
```