Django - Models

A model is a class that represents table or collection in our DB, and where every attribute of the class is a field of the table or collection. Models are defined in the app/models.py (in our example: myapp/models.py)

Creating a Model

Following is a Dreamreal model created as an example −

```
from django.db import models

class Dreamreal(models.Model):

   website = models.CharField(max_length = 50)
   mail = models.CharField(max_length = 50)
   name = models.CharField(max_length = 50)
   phonenumber = models.IntegerField()

   class Meta:
      db_table = "dreamreal"
```

Every model inherits from django.db.models.Model.

Our class has 4 attributes (3 CharField and 1 Integer), those will be the table fields.

The Meta class with the db_table attribute lets us define the actual table or collection name. Django names the table or collection automatically: myapp_modelName. This class will let you force the name of the table to what you like.

There is more field's type in django.db.models, you can learn more about them on

https://docs.djangoproject.com/en/1.5/ref/models/fields/#field-types

After creating your model, you will need Django to generate the actual database −

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

Manipulating Data (CRUD)

Let's create a "crudops" view to see how we can do CRUD operations on models. Our myapp/views.py will then look like −

myapp/views.py

```
from myapp.models import Dreamreal
from django.http import HttpResponse

def crudops(request):
   #Creating an entry

   dreamreal = Dreamreal(
      website = "www.polo.com", mail = "sorex@polo.com",
      name = "sorex", phonenumber = "002376970"
   )

   dreamreal.save()

   #Read ALL entries
   objects = Dreamreal.objects.all()
   res ='Printing all Dreamreal entries in the DB : <br>'

   for elt in objects:
      res += elt.name+"<br>"

   #Read a specific entry:
   sorex = Dreamreal.objects.get(name = "sorex")
   res += 'Printing One entry <br>'
   res += sorex.name

   #Delete an entry
   res += '<br> Deleting an entry <br>'
   sorex.delete()

   #Update
   dreamreal = Dreamreal(
      website = "www.polo.com", mail = "sorex@polo.com",
      name = "sorex", phonenumber = "002376970"
   )

   dreamreal.save()
   res += 'Updating entry<br>'

   dreamreal = Dreamreal.objects.get(name = 'sorex')
   dreamreal.name = 'thierry'
   dreamreal.save()

   return HttpResponse(res)
```

Other Data Manipulation

Let's explore other manipulations we can do on Models. Note that the CRUD operations were done on instances of our model, now we will be working directly with the class representing our model.

Let's create a 'datamanipulation' view in **myapp/views.py**

```
from myapp.models import Dreamreal
from django.http import HttpResponse

def datamanipulation(request):
   res = ''

   #Filtering data:
   qs = Dreamreal.objects.filter(name = "paul")
   res += "Found :  %s results<br>"%len(qs)

   #Ordering results
   qs = Dreamreal.objects.order_by("name")

   for elt in qs:
      res += elt.name + '<br>'

   return HttpResponse(res)
```

Linking Models

Django ORM offers 3 ways to link models −

One of the first case we will see here is the one-to-many relationships. As you can see in the above example, Dreamreal company can have multiple online websites. Defining that relation is done by using django.db.models.ForeignKey −

myapp/models.py

```
from django.db import models

class Dreamreal(models.Model):
   website = models.CharField(max_length = 50)
   mail = models.CharField(max_length = 50)
   name = models.CharField(max_length = 50)
   phonenumber = models.IntegerField()
```

```
   online = models.ForeignKey('Online', default = 1)

   class Meta:
      db_table = "dreamreal"

class Online(models.Model):
      domain = models.CharField(max_length = 30)

   class Meta:
      db_table = "online"
```

As you can see in our updated myapp/models.py, we added the online model and linked it to our Dreamreal model.

Let's check how all of this is working via manage.py shell −

First let’s create some companies (Dreamreal entries) for testing in our Django shell −

```
$python manage.py shell

>>> from myapp.models import Dreamreal, Online
>>> dr1 = Dreamreal()
>>> dr1.website = 'company1.com'
>>> dr1.name = 'company1'
>>> dr1.mail = 'contact@company1'
>>> dr1.phonenumber = '12345'
>>> dr1.save()
>>> dr2 = Dreamreal()
>>> dr1.website = 'company2.com'
>>> dr2.website = 'company2.com'
>>> dr2.name = 'company2'
>>> dr2.mail = 'contact@company2'
>>> dr2.phonenumber = '56789'
>>> dr2.save()
```

Now some hosted domains −

```
>>> on1 = Online()
>>> on1.company = dr1
>>> on1.domain = "site1.com"
>>> on2 = Online()
>>> on2.company = dr1
>>> on2.domain = "site2.com"
>>> on3 = Online()
>>> on3.domain = "site3.com"
>>> dr2 = Dreamreal.objects.all()[2]
>>> on3.company = dr2
>>> on1.save()
>>> on2.save()
```

```
>>> on3.save()
```

Accessing attribute of the hosting company (Dreamreal entry) from an online domain is simple −

```
>>> on1.company.name
```

And if we want to know all the online domain hosted by a Company in Dreamreal we will use the code −

```
>>> dr1.online_set.all()
```

To get a QuerySet, note that all manipulating method we have seen before (filter, all, exclude, order_by....)

You can also access the linked model attributes for filtering operations, let's say you want to get all online domains where the Dreamreal name contains 'company' −

```
>>> Online.objects.filter(company__name__contains = 'company'
```

Note − That kind of query is just supported for SQL DB. It won’t work for non-relational DB where joins doesn’t exist and there are two '_'.

But that's not the only way to link models, you also have OneToOneField, a link that guarantees that the relation between two objects is unique. If we used the OneToOneField in our example above, that would mean for every Dreamreal entry only one Online entry is possible and in the other way to.

And the last one, the ManyToManyField for (n-n) relation between tables. Note, those are relevant for SQL based DB.

Revision #1
Created 14 December 2022 11:07:49 by Admin
Updated 14 December 2022 11:08:20 by Admin