From App to DB

Ser517 Project Technique Study

I v been planning to write this study for a long time and now I realize it shall be done. Since this is the first time I actually use Python to do a WEB-based PROJECT(not just an assignment), there are a lot things for me to learn. I firmly believe the connection or interaction between APP to DB is the most important one among those techniques.

At first I thought it easy and unnecessary because 2 years ago, in Shanghai, we used SpringMVC+MyBatis to write Back-end code, which means we wrote SQL in a proper way required by Mybatis to make the interaction between App and DB happen. But now we don't necessarily to do the same work because we have Django, which is even easier than SpringMVC for us to use.

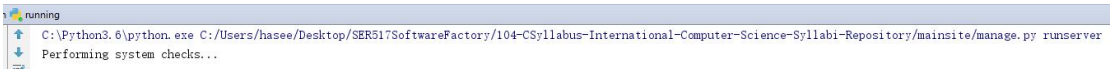
Django can do the interaction and data modeling work once u successfully configure the DB engine for it.

```
# Database
# https://docs.djangoproject.com/en/1.11/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'SER517',
        'USER': 'npiedy',
        'PASSWORD': 'root',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

We have some options for 'Engine'. Indicated by Django Documentation, we can use SQLite, PostGresQL, MySQL and Oracle. Though it supports some other DB engines, we choose PostGresQL for no reason..LOL. So we input 'django.db.backends.postgresql_psycopg2'
Remember one thing: we only use PostGresQL here as an engine, nothing more. We don't even write any SQL literally.

OK, now we look at Django. Once we run the server, we can access our admin page of this project



We can see 2 modules.

AUTHENTICATION AND AUTHORIZATION has a table named Users, which comes from:

```
from django.contrib.auth.models import User
```

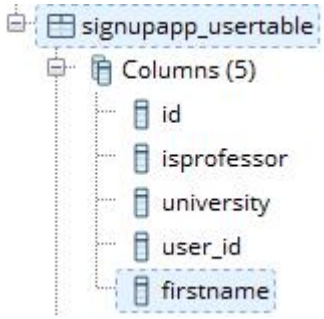
This table is not created by me but it can encrypt password using PBKDF2 algorithm with a SHA256 hash so that it can authenticate the password and keep it safe.

In order to make use of Django, I wanna create my own data-model(table).

```
from django.db import models
from django.contrib.auth.models import User
from django.utils import timezone


class UserTable(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    isprofessor = models.BooleanField()
    university = models.CharField(max_length=200)
```

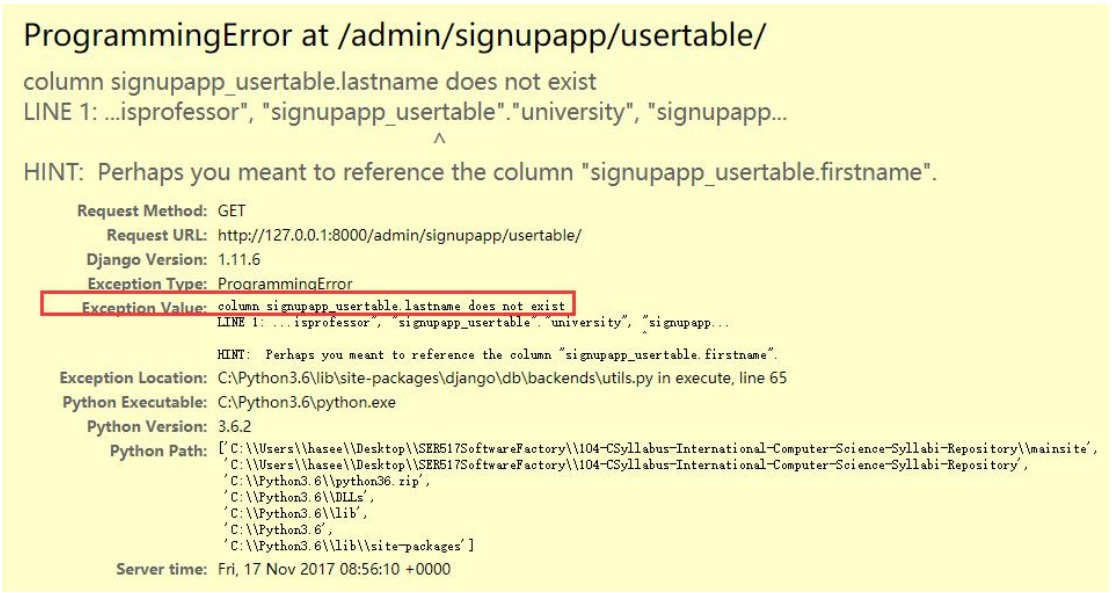
The name of this table is UserTable and the name of this class is UserTable. It has 3 attributes which means the table has at least 3 columns. Let's get into pgAdmin4:



However, if I try to add an attribute that table doesn't have, it will report an error:

```
from django.db import models
from django.contrib.auth.models import User
from django.utils import timezone


class UserTable(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    isprofessor = models.BooleanField()
    university = models.CharField(max_length=200)
    lastname=models.CharField(max_length=200)
```



That's because we don't have lastname column.

After defining the model, we can use this object in our application:

```
from .models import UserTable

university = request.POST['university']
is_professor = request.POST['is_professor']

usertable=UserTable()
usertable.university = university
usertable.isprofessor = isprofessor
usertable.user = user
usertable.save()
```

But we should keep in mind that User and UserTable are 2 different models.

Great! Since we have already known how to store the data, it is time to learn how to get the data from DB!

```
from .models import UserTable

nob = UserTable.objects.all()
print(nob)
```

And the results are:

```
<QuerySet [<UserTable: asd>, <UserTable: ad>]>
```

In fact we have several columns in this table, the reason it only shows the username is that:

```
class UserTable(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    isprofessor = models.BooleanField()
    university = models.CharField(max_length=200)
    lastname=models.CharField(max_length=200)
    def publish(self):
        self.published_date = timezone.now()
        self.save()

    def __str__(self):
        return self.user.username
```

We override the function.
So if we change this __str__ function body, it will give us more information.

Challenges:

Now we know how to save and get the data by doing data_updating and data_query. But I don't know how to access DB in different APP, which does bother me a lot because I need those data from my previous tables.

Also, now we are developing Profile

Dashboard Module. A user can get this page only if he/she successfully login, which means we should access his/her information including username. Then we can create a ForeignKey as username in Document Table in order to specifically query the documents uploaded by a user.

However we don’t know how to extract the information from a user who has already logined in our website.

So we still need to spend a lot of time on learning how to transfer the data from DB to App.