Working with historical versions of Django models in migrations using apps.get_model()

parijat purohit

March 10, 2023

When you run migrations in Django, it compares the current state of your models with the historical versions of your models stored in all migration files that have not yet been applied to the database. Each migration file represents a historical version of the models, and Django applies these migrations in the order specified by the 'dependencies' attribute in each migration file. When you create a new migration, Django generates a new migration file that includes the changes to the models since the last migration.

Sometimes, you may need to work with the historical versions of your models in migrations. For example, when you use the **RunPython** operation or have **allow_migrate** methods on your database routers, you may need to refer to the historical model versions rather than importing them directly. This is because when you use 'RunPython', you are essentially running Python code in the context of the migrations framework, which means that the current state of your models may not be available.

To work with the historical versions of your models, you can use the **apps.get_model()** method. This method provides access to the historical version of the model that is defined in the migration file that is currently being applied to the database.

Here is an example of using **apps.get_model()** in a migration:

```
from django.db import migrations

def set_default_body(apps, schema_editor):
    Post = apps.get_model('blog', 'Post')
    Post.objects.update(body='Default body text')

class Migration(migrations.Migration):

    dependencies = [
        ('blog', '0001_initial'),
    ]
```

```
operations = [
    migrations.RunPython(set_default_body),
]
```

In this migration, we define a function called **set_default_body** that takes two arguments: **apps** and **schema_editor**. The **apps** argument is an object that provides access to the historical versions of your models, while the **schema_editor** argument is an object that allows you to make changes to the database schema. Inside the **set_default_body** function, we use **apps.get_model()** to get the historical version of the **Post** model, and then update all existing **Post** objects to have a default body text.

After the migration has been applied, you can assume that importing the 'Post' model directly will import the latest version of the model that is defined in your code:

```
from blog.models import Post

post = Post.objects.create(title='My_Title', body='My_Body')
```

In this code, we are importing the 'Post' model directly and creating a new 'Post' object. This code assumes that the latest version of the 'Post' model, which includes the 'body' field with the default value, is defined in your code.

In summary, **apps.get_model()** is a method that provides access to the historical versions of your models, and is used when you need to work with these historical versions, rather than the current version of the model. The **RunPython** operation is executed in the context of the migrations framework, so it is necessary to use **apps.get_model()** to ensure that you are working with the historical version of the model that is stored in the migration file. Once a migration has been applied, you can assume that importing the model directly will import the latest version of the model that is defined in your code.

Special Credit: ChatGPT(https://chat.openai.com/chat)