محمد هادی امید

تمرین ۱۵

سوال دوم:

در بحث مدل های جانگو Abstract base classes, proxy Model, Multi-table inheritance چیست و برای هرکدام یک توضیح کوتاه دهید که چه کاربردی دارد و یک مثالی برای پیاده سازی آن بنویسید.

ارث بری در جنگو تقریبا مشابه ارث بری در کلاس های پایتون می باشد. از کلاس های اولیه باید ارث بری گردد. تنها تصمیمی که باید گرفته شود این است که آیا مدل ارث بری شده با جداولش باشد و یا این که مدل والد تنها اطلاعات عادی را که توسط مدل فرزند قابل نمایش است نگه دارد.

سه روش ارث بری در پایتون وجود دارد:

۱- **Abstract base classes**:

وقتی میخواهیم از کلاس والد برای چند کلاس فرزند اطلاعاتی را ارث بری کنیم به طوری که نخواهیم آن اطلاعات را برای هر کلاس فرزند دوباره تکرار و تایپ کنیم. در واقع این روش ارث بری زمانی استفاده می شود که بخواهیم بکسری اطلاعات مشترک را برای تعدادی از مدل های دیگر ارث بری کنیم. ابتدا کلاس پایه را نوشته و در کلاس متا abstract را برابر True قرار می دهیم. با استفاده از این کلاس، جدول جدید در دیتا بیس ایجاد نمی شود بلکه پس از استفاده در کلاس فرزند فیلدهای آن در کلاس فرزند ایجاد می شود.

```
from django.db import models

class CommonInfo(models.Model):
    name = models.CharField(max_length=100)
    age = models.PositiveIntegerField()
    class Meta:
        abstract = True

class Student(CommonInfo):
    home_group = models.CharField(max_length=5)
```

۲- **Multi-table inheritance**:

اگر از مدلی ارث بری کنیم و بخواهیم که مدل شامل جدول های خودش باشد. در این حالت هر مدل خودش به تنهایی یک مدل مستقل است. به عبارتی هر مدل جدول خودش را در دیتابیس دارد. به عبارتی جدولی جداگانه ایجاد شده و قابلیت کوئری زدن دارد. با ارث بری از والد، یک رابطه یک به یک به طور اتوماتیک بین کلاس فرزند و کلاس والد ایجاد می شود.

```
from django.db import models


class Place(models.Model):
    name = models.CharField(max_length=50)
    address = models.CharField(max_length=80)


class Restaurant(Place):
    serves_hot_dogs = models.BooleanField(default=False)
    serves_pizza = models.BooleanField(default=False)
```

در مثال فوق تمام فیلد های مدل place در کلاس Restaurant وجود دارد، گرچه داده ها در جداولی جداگانه ذخیره میشوند. البته برای کوئری زدن داده های Restaurant، از place هم می توان استفاده کرد.

۳- **Proxy Model**:

اگر فقط بخواهیم که در سطح پایتون مدلی را تغییر دهیم، بدون تغییر فیلد های مدل، از proxy model استفاده میکنیم. وقتی از multi-table inheritance استفاده می کنیم، یک جدول جدید برای هر کلاس فرزند مدل در دیتابیس ایجاد میشود. معمولا این چیزی هست که میخواهیم چون کلاس های فرزند نیاز به یک فضایی جهت ذخیره اطلاعات اضافه شده دارند که در کلاس والد نیست. ولی بعضی اوقات فقط می خواهیم که رفتار پایتونی مدل را عوض کنیم. منیجر پیش فرض را عوض کنیم یا یک متد جدید اضافه کنیم. این کاری است که proxy model inheritance انجام می دهد. ذخیره داده ها همانند مدل های غیر proxy است و تفاوت در این است که میتوانیم ترتیب مدل یا منیجر مدل را تغییر دهیم. بدون آنکه نیازی به تغییر در مدل اصلی باشد.

در جنگو با فرار دادن proxy برابر True در زیرکلاس متا؛ کلاس فرزندمان به عنوان proxy model شناسایی میشود.

```python
from django.db import models

class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)

class OrderedPerson(Person):
    class Meta:
        ordering = ["first_name"]
        proxy = True
```

در مثال فوق در کلاس فرزند OrderedPerson، بر اساس نام خانوادگی داده ها را مرتب کرده ایم.
همچنین کلاس OrderedPerson بر روی همان جدول person در دیتابیس ذخیره میشود و تمام داده های هر دو جدول از هر یک از مدل ها قابل دسترسی است.