**محمد هادي اميد**

**تمرين 15**

**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 بري در جنگو تقريبا مشابه ارث بري در کلاس هاي پايتون مي باشد. از کلاس هاي اوليه بايد ارث بري گردد. تنها تصميمي که بايد گرفته شود اين است که آيا مدل ارث بري شده با جداولش باشد و يا اين که مدل والد تنها اطلاعات عادي را که توسط مدل فرزند قابل نمايش است نگه دارد.

سه روش ارث بري در پايتون وجود دارد:

1. **Abstract base classes:**

**وقتي ميخواهيم از کلاس والد براي چند کلاس فرزند اطلاعاتي را ارث بري کنيم به طوري که نخواهيم آن اطلاعات را براي هر کلاس فرزند دوباره تکرار و تايپ کنيم. در واقع اين روش ارث بري زماني استفاده مي شود که بخواهيم بکسري اطلاعات مشترک را براي تعدادي از مدل هاي ديگر ارث بري کنيم. ابتدا کلاس پايه را نوشته و در کلاس متا abstract را برابر True قرار مي دهيم. با استفاده از اين کلاس، جدول جديد در ديتا بيس ايجاد نمي شود بلکه پس از استفاده در کلاس فرزند فيلدهاي آن در کلاس فرزند ايجاد مي 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)

1. **Multi-table inheritance:**

**اگر از مدلي ارث بري کنيم و بخواهيم که مدل شامل جدول هاي خودش باشد. در اين حالت هر مدل خودش به تنهايي يک مدل مستقل است. به عبارتي هر مدل جدول خودش را در ديتابيس دارد. به عبارتي جدولي جداگانه ايجاد شده و قابليت کوئري زدن دارد. با ارث بري از والد، يک رابطه يک به يک به طور اتوماتيک بين کلاس فرزند و کلاس والد ايجاد مي 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 فيلد هاي مدل place در کلاس Restaurant وجود دارد، گرچه داده ها در جداولي جداگانه ذخيره ميشوند. البته براي کوئري زدن داده هاي Restaurant، از place هم مي توان استفاده کرد.**

1. **Proxy Model:**

**اگر فقط بخواهيم که در سطح پايتون مدلي را تغيير دهيم، بدون تغيير فيلد هاي مدل، ازproxy model استفاده ميکنيم. وقتي از multi-table inheritance استفاده مي کنيم، يک جدول جديد براي هر کلاس فرزند مدل در ديتابيس ايجاد ميشود. معمولا اين چيزي هست که ميخواهيم چون کلاس هاي فرزند نياز به يک فضايي جهت ذخيره اطلاعات اضافه شده دارند که در کلاس والد نيست. ولي بعضي اوقات فقط مي خواهيم که رفتار پايتوني مدل را عوض کنيم. منيجر پيش فرض را عوض کنيم يا يک متد جديد اضافه کنيم. اين کاري است که proxy model inheritance انجام مي دهد. ذخيره داده ها همانند مدل هاي غير proxy است و تفاوت در اين است که ميتوانيم ترتيب مدل يا منيجر مدل را تغيير دهيم. بدون آنکه نيازي به تغيير در مدل اصلي باشد.**

**در جنگو با فرار دادن proxy برابر True در زيرکلاس متا؛ کلاس فرزندمان به عنوان proxy model شناسايي ميشود.**

**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 خانوادگي داده ها را مرتب کرده ايم. همچنين کلاس OrderedPerson بر روي همان جدول person در ديتابيس ذخيره ميشود و تمام داده هاي هر دو جدول از هر يک از مدل ها قابل دسترسي است.**