Universidad Tecnológica de Tijuana

TSU. en Tecnologías de la Información

en Entornos Virtuales y Negocios Digitales

Made By:

“Humaran Beltran Saul Alejandro“

Teacher:

“Ing. Ray Brunett Parra Galaviz”

Assigment:

“Web Aplications“

Actividad:

“Django inheritance“

Cuatrimestre: Third ***Grupo:*** 3-F

Date

“25/10/2023“

Introduction:

In Django, model inheritance allows you to create new models that are based on existing models, inheriting their fields and behavior. This concept helps in reusing and extending existing models, which can be quite useful in various situations.

There are three types of model inheritance in Django

1. **Abstract Base Classes:**

- Abstract base classes are not meant to be instantiated on their own. Instead, they serve as a base for other models. Fields and methods defined in an abstract base class are inherited by the child models.

- To create an abstract base class, you set the abstract attribute to True in the model's Meta class.

- Example:

```
from django.db import models

class BaseModel(models.Model):
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        abstract = True

class MyModel(BaseModel):
    # MyModel will inherit the fields 'created_at' and 'updated_at'
    name = models.CharField(max_length=100)
```

2. Multi-table Inheritance:

- Multi-table inheritance is used when you want to create a model that is a subclass of another model, and each model has its own database table.

- Fields from the parent model are included in the child model's table. Each model can have additional fields and methods.

- Example:

```python
from django.db import models

class Animal(models.Model):
    name = models.CharField(max_length=100)

class Dog(Animal):
    breed = models.CharField(max_length=100)

class Cat(Animal):
    color = models.CharField(max_length=100)
```

3. Proxy Models:

- Proxy models are a way to create a new model that extends the functionality of an existing model. However, both models share the same database table, and no new fields are added.

- Proxy models are often used for adding custom methods to an existing model without modifying the original model.

- Example:

```python
from django.db import models

class MyModel(models.Model):
    name = models.CharField(max_length=100)

class MyModelProxy(MyModel):
    class Meta:
        proxy = True

    def custom_method(self):
        # Add custom methods to the proxy model
        return f"Custom method for {self.name}"
```

Conclusion:

In Django, model inheritance is a powerful tool for code reuse and creating more organized and maintainable code. It allows you to structure your models in a way that best fits your application's requirements while promoting the DRY (Don't Repeat Yourself) principle.