Django Rest Framework

Notes By

Naveen

Mixin's

It is a type of multiple inheritance which allows classes in Python to share methods between any class that inherits from that mixin.

It is used when we want to implement a specific functionality in different classes.

Mixins can be implemented by creating a class.

Client classes then inherit the Mixin class, often with other mixin classes and possibly a concrete base class.

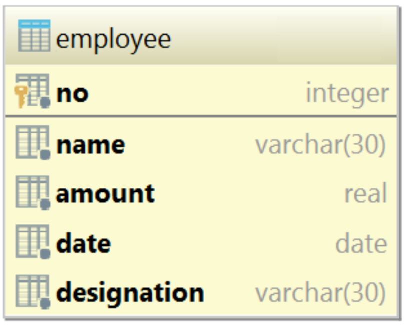
Model Class Inheritance

```
from django.db import models

class CommonModel(models.Model):
    no = models.IntegerField(primary_key=True)
    name = models.CharField(max_length=30)
    amount = models.FloatField()
    date = models.DateField(auto_now_add=True)
    class Meta:
        abstract = True


class Product(CommonModel):
    quantity = models.IntegerField()
```

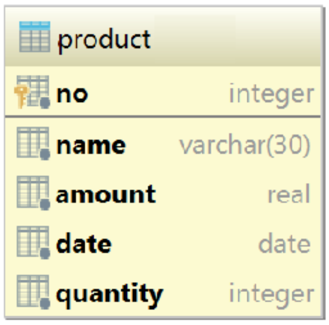
```
class Employee(CommonModel):
    designation = models.CharField(max_length=30)
```

In the above example CommonModel is abstract class.

Models in Visualization



employee	
no	integer
name	varchar(30)
amount	real
date	date
designation	varchar(30)



product	
no	integer
name	varchar(30)
amount	real
date	date
quantity	integer

In the above 2 models no,name,amount,date are common

In employee model amount means Salary,date means Joining Date.

In product model amount means price,date means current date.

Forms Mixin's

Create a new python file in app and name it as **"mixins.py".**

```
from django import forms
import re
class CommonValidationsMixin(forms.Form):
    # no validation
    def clean_no(self):
        no = self.cleaned_data["no"]
        if no>=1:
            return no
        else:
            raise forms.ValidationError("Invalid No")
    # name validation
    def clean_name(self):
        name = self.cleaned_data["name"]
        result = re.match("^[A-Za-z]*$",name)
        if result == None:
            raise forms.ValidationError("Invalid Name")
        else:
            return name
    # amount validation
    def clean_amount(self):
        amount = self.cleaned_data["amount"]
        if amount>=1:
            return amount
        else:
            raise forms.ValidationError("Invalid Amount")
```

Create a new python file in app and name it as **"forms.py".**

```
from django import forms
from app2.models import Employee
from app2.models import Product
from app2.mixins import CommonValidationsMixin
import re

class EmployeeForm(CommonValidationsMixin,forms.ModelForm):
    class Meta:
        model = Employee
        fields = "__all__"

    def clean_designation(self):
        designation = self.cleaned_data["designation"]
        result = re.match("^[A-Za-z]*$", designation)
        if result == None:
            raise forms.ValidationError("Invalid Designation")
        else:
            return designation

class ProductForm(CommonValidationsMixin,forms.ModelForm):
    class Meta:
        model = Product
        fields = "__all__"

    def clean_quantity(self):
        quantity = self.cleaned_data["quantity"]
        if quantity >= 1:
            return quantity
        else:
            raise forms.ValidationError("Invalid Quantity")
```

urls.py

```
path('add_employee/',csrf_exempt(views.AddNewEmployee.as_view())),

path('add_product/',csrf_exempt(views.AddNewProduct.as_view())),
```

views.py

```
from django.http import HttpResponse
from django.views.generic import View
from app2.forms import EmployeeForm,ProductForm
import json
```

```
class AddNewEmployee(View):
    def post(self,request):
        emp = EmployeeForm(json.loads(request.body))
        if emp.is_valid():
            emp.save()
            message = {"message":"Employee Details are saved"}
        else:
            message = {"error":emp.errors}

        json_data = json.dumps(message)
        return
HttpResponse(json_data,content_type="application/json")

class AddNewProduct(View):
    def post(self,request):
        product = ProductForm(json.loads(request.body))
        if product.is_valid():
            product.save()
            message = {"message":"Product Details are saved"}
        else:
            message = {"error":product.errors}

        json_data = json.dumps(message)
        return
HttpResponse(json_data,content_type="application/json")
```