**Московский государственный технический университет им. Н.Э.Баумана**

|  |  |  |
| --- | --- | --- |
|  |  |  |

**Отчет по лабораторной работе №5 по курсу**

**РИП**

«Работа с СУБД»

7

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| студент группы **ИУ5-52** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (подпись) |
| **Писарчук Н.** | "22" октября 2018 г. |

Москва, МГТУ - 2018

Задание и порядок выполнения

В этой лабораторной работе вы познакомитесь с популярной СУБД MySQL, создадите свою базу данных. Также вам нужно будет дополнить свои классы предметной области, связав их с созданной базой. После этого вы создадите свои модели с помощью Django ORM, отобразите объекты из БД с помощью этих моделей и ClassBasedViews.

Для сдачи вы должны иметь:

1. Скрипт с подключением к БД и несколькими запросами.
2. Набор классов вашей предметной области с привязкой к СУБД (класс должен уметь хотя бы получать нужные записи из БД и преобразовывать их в объекты этого класса)
3. Модели вашей предметной области
4. View для отображения списка ваших сущностей

1. Подключение к БД

import MySQLdb  
class Connection:  
 def \_\_init\_\_(self, user, password, db, host='localhost'):  
 self.user = user  
 self.password = password  
 self.db = db  
 self.host = host  
 self.\_connection = None  
 @property  
 def connection(self):  
 return self.\_connection  
 def \_\_enter\_\_(self):  
 self.connect()  
 def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):  
 self.disconnect()  
 def connect(self):  
 if not self.\_connection:  
 self.\_connection = MySQLdb.connect(  
 host=self.host,  
 user=self.user,  
 passwd=self.password,  
 db=self.db,  
 charset='utf8'  
 )  
 def disconnect(self):  
 if self.\_connection:  
 self.\_connection.close()

2. Классы с привязкой к СУБД

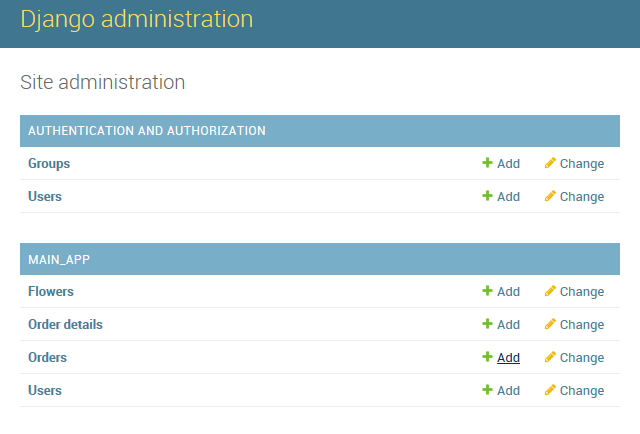
import MySQLdb  
db = MySQLdb.connect(  
 host="localhost",  
 user="dbuser",  
 passwd="1234",  
 db="lab5",  
 charset='utf8'  
)  
cursor = db.cursor()  
cursor.execute("SELECT \* FROM books;")  
entries = cursor.fetchall()  
for e in entries:  
 print(e)  
cursor.close()  
db.close()

from lab5.connect import Connection  
class Book:  
 def \_\_init\_\_(self, db\_connection, name, description):  
 self.db\_connection = db\_connection.connection  
 self.name = name  
 self.description = description  
 def save(self):  
 cur = self.db\_connection.cursor()  
 cur.execute("INSERT INTO books (bookname, description) VALUES (%s, %s);", (self.name, self.description))  
 self.db\_connection.commit()  
 cur.close()  
con = Connection("dbuser", "1234", "lab5")  
with con:  
 book = Book(con, 'electronic', 'especially for electroded')  
 book.save()

3. Модель

Предметная область – магазин цветов. Будут определены 3 сущности – цветок, пользователь, заказ. Между пользователем и заказом связь 1:М, цветком и заказом связь М:М, которая будет реализована через дополнительную таблицу – детали заказа. В базе данных этой ПО будет 4 таблицы: цветок, пользователь, заказ, деталь заказа.

from django.db import models  
  
# Create your models here.  
class User(models.Model):  
 first\_name = models.CharField(max\_length=100)  
 last\_name = models.CharField(max\_length=100)  
 mail\_address = models.EmailField(max\_length=255, unique=True)  
 phone\_number = models.CharField(max\_length=12, unique=True)  
 birth\_date = models.DateField(null=True)  
class Flower(models.Model):  
 flower\_name = models.CharField(max\_length=90, unique=True)  
 price = models.IntegerField()  
 color = models.CharField(max\_length=255)  
 country = models.CharField(max\_length=50)  
 img = models.ImageField()  
class Order(models.Model):  
 order\_date = models.DateField(auto\_now\_add=True)  
 flowers = models.ManyToManyField(Flower, through='OrderDetail')  
 user = models.ForeignKey(User, on\_delete=models.SET\_NULL, null=True)  
 total\_price = models.IntegerField(null=True)  
class OrderDetail(models.Model):  
 order = models.ForeignKey(Order, on\_delete=models.PROTECT)  
 flower = models.ForeignKey(Flower, models.SET\_NULL, null=True)  
 quantity = models.IntegerField()



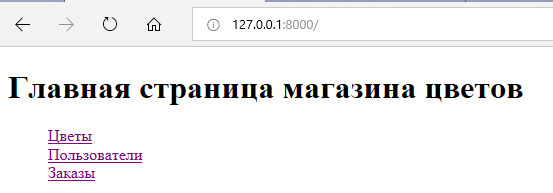
4. View

Views.py

from django.shortcuts import render  
from django.views.generic import DetailView, FormView, ListView  
from main\_app.models import \*  
class FlowerView(ListView):  
 model = Flower  
class OrderView(ListView):  
 model = Order  
class UserView(ListView):  
 model = User  
def static\_main(request):  
 return render(request, 'main.html')

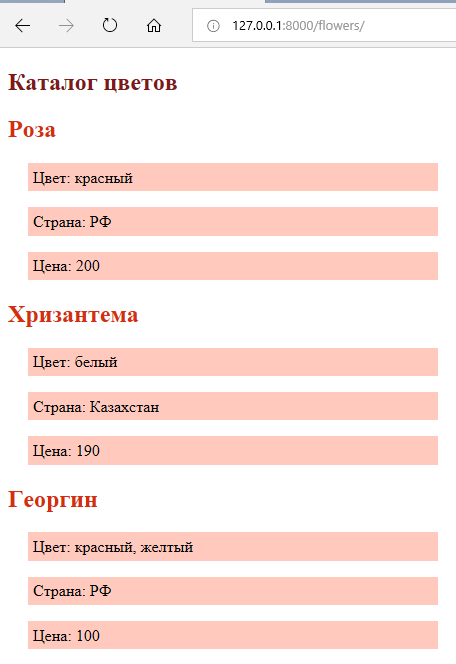
main.html

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Flower Store</title>  
</head>  
<body>  
 <h1>Главная страница магазина цветов</h1>  
 <ul>  
 <a href="/flowers">Цветы</a>  
 <br>  
 <a href="/users">Пользователи</a>  
 <br>  
 <a href="/orders">Заказы</a>  
 </ul>  
</body>  
</html>



Flower\_list.html

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Flowers</title>  
 <style>  
 body{  
 background-color: #fff;  
 }  
 p{  
 color: #000000;  
 margin-left: 20px;  
 background-color: #ffcabd;  
 padding: 5px;  
 width: 400px;  
 }  
 strong{  
 color: #d42e0d;  
 }  
 h2{  
 color: #801818;  
 }  
 </style>  
  
</head>  
<body>  
 <h2>Каталог цветов</h2>  
 {% for Flower in object\_list %}  
 <h2><strong> {{ Flower.flower\_name }}</strong></h2>  
 <p>Цвет: {{ Flower.color }}</p>  
 <p>Страна: {{ Flower.country }}</p>  
 <p>Цена: {{ Flower.price }}</p>  
 {% endfor %}  
</body>  
</html>



Order\_list.html

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Orders</title>  
</head>  
<body>  
 <h2>Заказы</h2>  
 <a>Пока нет заказов</a>  
</body>  
</html>



User\_list.html

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Users</title>  
</head>  
<body>  
 <h2>Пользователи, зарегистрированные на сайте</h2>  
 {% for User in object\_list %}  
 <li>Имя:{{ User.first\_name }}</li>  
 <a>Фамилия:{{ User.last\_name }}</a>  
 {% endfor %}  
</body>  
</html>

