Московский государственный технический

университет им. Н.Э. Баумана.

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Разработка интернет приложений»

Отчет по лабораторной работе №5

«Работа с СУБД»

Выполнил:

студент группы ИУ5

Гунькин М. А.

Подпись и дата:

Проверил:

доцент каф. ИУ5

Гапанюк Ю. Е.

Подпись и дата:

Москва, 2017 г.

Оглавление

Задание

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

Исходный код

First_script.py

```
import MySQLdb


def insert(name,desc):
    c.execute("INSERT INTO books (name, description) VALUES (%s, %s);", (name, desc))
    db.commit()


def get_entries():
    c.execute("SELECT * FROM books;")
    return c.fetchall()


def truncate():
    c.execute("TRUNCATE books;")
    db.commit()

# Открываем соединение
db = MySQLdb.connect(
    host="localhost",
    user="dbuser",
    passwd="123",
    db="first_db"
)
db.set_character_set('utf8')
# Получаем курсор для работы с БД
c = db.cursor()


insert('Книга','Описание книги')
entries = get_entries()
for e in entries: print(e)
truncate()


c.close()   # Закрываем курсор
db.close()  # Закрываем соединение
```

RipLib.py

```
class User:
```

```
    def __init__(self, db_connection, first_name, last_name, phone=None):
        self.db_connection = db_connection.connection
        self.first_name = first_name
        self.last_name = last_name
        self.phone = phone

    def save(self):
        c = self.db_connection.cursor()
        c.execute("INSERT INTO users (first_name, last_name, phone) VALUES (%s, %s, %s);",
                  (self.first_name, self.last_name, self.phone))
        self.db_connection.commit()
        c.close()

    @staticmethod
    def load_users(db_connection):
        c = db_connection.connection.cursor()
        c.execute("SELECT * FROM users")
        lines = c.fetchall()
        c.close()
        users = [User(db_connection, l[0], l[1], l[2]) for l in lines]
        return users


class Hotel:
    def __init__(self, db_connection, name, phone,adress,description=None):
        self.db_connection = db_connection.connection
        self.name = name
        self.phone = phone
        self.adress = adress
        self.description = description

    def save(self):
        c = self.db_connection.cursor()
        c.execute("INSERT INTO hotels (name, phone, adress, description) VALUES (%s, %s, %s, %s);",
                  (self.name, self.phone, self.adress, self.description))
        self.db_connection.commit()
        c.close()


class Booking:
    def __init__(self, db_connection, user_id,hotel_id,price,start_date,end_date):
        self.db_connection = db_connection.connection
        self.user_id = user_id
        self.hotel_id = hotel_id
        self.price = price
        self.start_date = start_date
        self.end_date = end_date

    def save(self):
        c = self.db_connection.cursor()
        c.execute("INSERT INTO bookings (user_id, hotel_id, price, start_date, end_date) VALUES (%d, %d, %d,
%s, %s);",
                  (self.user_id, self.hotel_id, self.price, self.start_date, self.end_date))
        self.db_connection.commit()
        c.close()
```

second_script.py

```
from Connection import Connection
from RipLib import User, Hotel, Booking

con = Connection("dbuser", "123", "rip_course_db")
```

```
with con:
    users = User.load_users(con)
    for user in users:
        print(user.first_name, user.last_name, user.phone)

#with con:
    #user = User(con, 'Иван', 'Иванов')
    #user.save()
```

Models.py

```
from django.db import models
from django.contrib.auth.models import User
from django.contrib import admin

# Create your models here.
class Traveler(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    age = models.IntegerField(null=True)


class Hotel(models.Model):
    owner = models.OneToOneField(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=30)
    adress = models.CharField(max_length=30)
    description = models.CharField(max_length=255,null=True)

    objects = models.Manager()

class Booking(models.Model):
    user = models.ForeignKey(Traveler, on_delete=models.CASCADE)
    hotel = models.ForeignKey(Hotel, on_delete=models.CASCADE)
    price = models.IntegerField()
    start_date = models.DateField()
    end_date = models.DateField()
```

Views.py

```
from django.views.generic import ListView
from django import forms
from django.contrib.auth import authenticate,login, logout
from django.contrib.auth.decorators import login_required
from .models import  *

# Create your views here.
class TravelerList(ListView):
    model = Traveler
    template_name = 'traveler_list.html'


class HotelList(ListView):
    model = Hotel
    template_name = 'hotel_list.html'


class BookingList(ListView):
    model = Booking
    template_name = 'booking_list.html'
```

Результаты работы