

**Отчет по лабораторной работе № 6 по курсу
“Разработка интернет-приложений”**

«Работа с СУБД»

ИСПОЛНИТЕЛЬ:

студентка группы **ИУ5-53**

(подпись)

Бабин В.Е.

"__" _____ 2017 г.

1. Работа с базой данных без использования ORM, без создания классов сущностей из предметной области.

Файл 1.py:

```
import pymysql
pymysql.install_as_MySQLdb()

db = pymysql.connect(
    host='localhost',
    user='Victor',
    passwd='12345678',
    db='first_db'
)

c = db.cursor()

c.execute('INSERT INTO myapp_film (id, name, description, country, author) VALUES (%s, %s, %s, %s, %s);', (5, 'Book4', 'Des4', 'USA', 'Author4'))

db.commit()

c.execute('SELECT * FROM myapp_film;')

entries = c.fetchall()

for e in entries:
    print(e)

c.close()
db.close()
```

Результат работы:

```
C:\Users\Виктор\AppData\Local\Programs\Python\Python36-32\python.exe C:/Projects/lab_5/first_quest/1.py
(1, 'Book1', 'Descl', 'Rus', 'Author1')
(2, 'Fight Club', 'Was create by Palauhnik', 'USA', 'Chuck Palauhnik')
(3, 'Book3', 'Des3', 'USA', 'Author3')
(4, 'Book1', 'Descl', 'Rus', 'Author1')
(5, 'Book4', 'Des4', 'USA', 'Author4')
(6, 'Book4', 'Des4', 'USA', 'Author4')
(7, 'Book1', 'Descl', 'Rus', 'Author1')
(8, 'Book1', 'Descl', 'Rus', 'Author1')
(9, 'Book4', 'Des4', 'USA', 'Author4')

Process finished with exit code 0
```

2. Работа с базой данных без использования ORM, создали классы сущностей из предметной области.

Сначала создали класс для подключения к БД (файл connection.py):

```

import pymysql
pymysql.install_as_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 = pymysql.connect(
                host=self.host,
                user=self.user,
                passwd=self.password,
                db=self.db
            )

    def disconnect(self):
        if self._connection:
            self._connection.close()

    def disconnect(self):
        if self._connection:
            self._connection.close()

```

После чего создадим классы сущностей предметной области (файл 2.py):

```

from first_quest.Connection import Connection

class Film:
    def __init__(self, db_connection, name, description, country, author):
        self.db_connection = db_connection
        self.name = name
        self.description = description
        self.country = country
        self.author = author

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

class User:
    def __init__(self, db_connection, sex, age, first_name, last_name):
        self.db_connection = db_connection
        self.sex = sex
        self.age = age
        self.first_name = first_name
        self.last_name = last_name

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

class Review:
    def __init__(self, db_connection, film_id, user_id, title, review_text, publication_date):
        self.db_connection = db_connection
        self.film_id = film_id
        self.user_id = user_id
        self.title = title
        self.review_text = review_text
        self.publication_date = publication_date

    def save(self):
        c = self.db_connection.cursor()
        c.execute("INSERT INTO myapp_review (film_id, user_id, title, review_text, publication_date) VALUES (%s, %s, %s, %s, %s);", (self.film_id, self.user_id, self.title, self.review_text, self.publication_date))
        self.db_connection.commit()
        c.close()

con = Connection('Victor', '12345678', 'first_db')

with con:
    film = Film(con, 'Stalker', 'Created by Tarkovskiy', 'Russia', 'Tarkovskiy')
    film.save()
    user1 = User(con, 'M', '27', 'Vasilii', 'Rupkin')
    user2 = User(con, 'F', '33', 'Oksana', 'Rukareku')
    user3 = User(con, 'F', '19', 'Olga', 'Smirnova')
    user1.save()
    user2.save()
    user3.save()

```

3. Работа с базой данных с использованием ORM:

Перед тем, как начать, необходимо подключить БД к нашему проекту в файле settings.py:

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'first_db',
        'USER': 'Victor',
        'PASSWORD': '12345678',
        'HOST': '127.0.0.1',
        'PORT': 3306,
        'OPTIONS': {'charset': 'utf8'},
        'TEST_CHARSET': 'utf8',
    }
}

```

После этого, необходимо создать миграцию для моделей нашей предметной области. Модели представлены в файле models.py:

```
from django.db import models

class Film(models.Model):
    name = models.CharField(max_length=30)
    description = models.CharField(max_length=254)
    country = models.CharField(max_length=50)
    author = models.CharField(max_length=70)

    def __unicode__(self):
        dict = {}
        dict['name'] = self.name
        dict['description'] = self.description
        dict['country'] = self.country
        dict['author'] = self.author
        return dict

class User(models.Model):
    sex = models.CharField(max_length=2)
    age = models.IntegerField()
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=40)

    def __unicode__(self):
        dict = {}
        dict['sex'] = self.sex
        dict['age'] = self.age
        dict['first_name'] = self.first_name
        dict['last_name'] = self.last_name
        return dict

class Review(models.Model):
    film_id = models.ForeignKey(Film)
    user_id = models.ForeignKey(User)
    title = models.CharField(max_length=100)
    review_text = models.CharField(max_length=500)
    publication_date = models.DateField()

    def __unicode__(self):
        dict = {}
        dict['film_id'] = self.film_id
        dict['user_id'] = self.user_id
        dict['title'] = self.title
        dict['review_text'] = self.review_text
        dict['publication_date'] = self.publication_date
        return dict
```

Получившиеся миграции:

```
class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Film',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('name', models.CharField(max_length=30)),
                ('description', models.CharField(max_length=254)),
                ('country', models.CharField(max_length=50)),
                ('author', models.CharField(max_length=70)),
            ],
        ),
        migrations.CreateModel(
            name='Review',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('title', models.CharField(max_length=100)),
                ('review_text', models.CharField(max_length=500)),
                ('publication_date', models.DateField()),
                ('film_id', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='myapp.Film')),
            ],
        ),
        migrations.CreateModel(
            name='User',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('sex', models.CharField(max_length=2)),
                ('age', models.IntegerField()),
                ('first_name', models.CharField(max_length=30)),
                ('last_name', models.CharField(max_length=40)),
            ],
        ),
        migrations.AddField(
            model_name='review',
            name='user_id',
            field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='myapp.User'),
        ),
    ]
```

После чего нужно создать view в файле views.py:

```
from django.shortcuts import render, HttpResponse
from django.views import View
from myapp.models import Film, User, Review
# Create your views here.




class Test(View):
    def get(self, request):
        film = Film(name='Book1', description='Desc1', country='Rus', author='Author1')
        film.save()
        films = Film.objects.all()
        data = {
            'films': films
        }
        return render(request, 'info_file.html', data)
```

Файл urls.py:

```
from django.conf.urls import url
from django.contrib import admin
from myapp.views import Test

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^prob/', Test.as_view()),
]
```

И, наконец, результат работы нашей view на веб-странице:

   127.0.0.1 Some data

- Book1, Desc1, Режиссёр - Author1, Страна - Rus.
- Fight Club, Was create by Palauhnik, Режиссёр - Chuck Palauhnik, Страна - USA.
- Book3, Des3, Режиссёр - Author3, Страна - USA.
- Book1, Desc1, Режиссёр - Author1, Страна - Rus.
- Book4, Des4, Режиссёр - Author4, Страна - USA.
- Book4, Des4, Режиссёр - Author4, Страна - USA.
- Book1, Desc1, Режиссёр - Author1, Страна - Rus.
- Book1, Desc1, Режиссёр - Author1, Страна - Rus.