Московский государственный технический университет имени Н.Э. Баумана

Кафедра ИУ5
«Системы обработки информации и управления»

Отчет по лабораторной работе №6

«Работа с СУБД»

по дисциплине «Разработка Интернет-приложений»

Выполнил:
студент группы ИУ5-53
Слимов Никита

Москва, 2016

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

Файл lab/models.py

```python
from django.db import models


class Tutor(models.Model):
    lastname = models.CharField(max_length=50)
    firstname = models.CharField(max_length=50)
    middlename = models.CharField(max_length=50)
    birthday = models.DateField(null=True, blank=True)
    sex = models.BooleanField(default=True)


class Course(models.Model):
    name = models.CharField(max_length=100)
    full_name = models.CharField(max_length=255)
    tutor = models.ForeignKey(Tutor)
```

Файл lab/views.py

```python
from django.shortcuts import render
from django.views import View

from lab.models import Tutor, Course
from django.db.models import Count

import random

def main(request):
    tutors = Tutor.objects.all()

    if len(tutors) == 0:
        for i in range(0, 10):
            t = Tutor.objects.create(lastname="Lastname " + str(i),
                                     firstname="Firstname " + str(i),
                                     middlename="Middlename " + str(i),
                                     birthday=str(random.randint(1950, 2000)) +
"-09-01",

                                     sex=random.randint(1,10) > 5)
            t.save()

            for j in range(0, random.randint(1,6)):
                c = Course.objects.create(name="Course # {} of tutor
{}".format(j, t.id),
                                                  full_name="Course fullname",
                                                  tutor=t)
                c.save()

    tutors = Tutor.objects.all()

    return render(request, 'main.html', {
        'tutors': tutors
    })


class TutorView(View):
    def get(self, request, id):
        tutor = Tutor.objects.get(id=int(id))

        courses = Course.objects.filter(tutor=tutor).all()

        return render(request, 'tutor.html', {
            'tutor': tutor,
            'courses': courses
        })
```

Файл lab6/urls.py

```
from django.conf.urls import url
from django.contrib import admin
from lab import views

urlpatterns = [
    url(r'^$', views.main, name='main'),
    url(r'^tutor/(?P<id>\d+)$', views.TutorView.as_view(), name='tutor'),
    url(r'^admin/', admin.site.urls),
]
```

Файл templates/layout.html

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>{% block title %}{% endblock %}</title>

    <link href="{% static 'css/bootstrap.min.css' %}" rel="stylesheet">
    <link href="{% static 'css/my.css' %}" rel="stylesheet">
</head>

<body>

<nav class="navbar navbar-static-top navbar-dark bg-inverse">
    <a class="navbar-brand" href="{% url 'main' %}">Project</a>
</nav>

<div class="jumbotron">
    <div class="container">
        <h1 class="display-3">{% block title_visible %}Hello, world!{%
endblock %}</h1>
    </div>
</div>

<div class="container">
    {% block body %}{% endblock %}

    <hr>

    <footer>
        <p>&copy; BMSTU 2016</p>
    </footer>
</div>

<script src="{% static 'js/jquery-3.1.1.slim.min.js' %}"></script>
<script src="{% static 'js/bootstrap.min.js' %}"></script>
</body>
</html>
```

Файл templates/main.html

```
{% extends 'layout.html' %}

{% block title %}
    Main page
{% endblock %}

{% block title_visible %}
    Main
{% endblock %}


{% block body %}
    {% for t in tutors %}
        {% if forloop.counter|divisibleby:2 %}
            <div class="row">
        {% endif %}
    <div class="col-md-6">
        {% include 'tutor_short.html' with tutor=t %}
    </div>
    {% if forloop.counter|divisibleby:2 or forloop.last %}
        </div>
    {% endif %}

    {% empty %}
        The list is empty.
    {% endfor %}

{% endblock %}
```

Файл templates/tutor.html

```
{% extends 'layout.html' %}

{% block title %}
    Tutor {{ tutor.lastname }} {{ tutor.fistname }} {{ tutor.middlename }}
{% endblock %}

{% block title_visible %}
    Tutor {{ tutor.lastname }} {{ tutor.fistname }} {{ tutor.middlename }}
{% endblock %}

{% block body %}
```

```
    Tutor {{ tutor.lastname }} {{ tutor.fistname }} {{ tutor.middlename }}
    <br/>
    {{ tutor.birthday }}
    {{ tutor.sex }}

    <br/>
    <br/>

    Courses:
    <ul>
        {% for c in courses %}
            <li>{{ c.name }} ({{ c.full_name }})</li>
        {% endfor %}
    </ul>
{% endblock %}
```

Файл templates/tutor_short.html

```
{% with id=tutor.id %}
<h2><a href="{% url 'tutor' id=id %}">{{ tutor.lastname }}</a></h2>
<p>{{ tutor.lastname }} {{ tutor.firstname }} {{ tutor.middlename }}</p>
<p><a class="btn btn-secondary" href="{% url 'tutor' id=tutor.id %}"
role="button">&rarr; </a></p>
{%  endwith %}
```

Файл db-test.py

```
try:
    import MySQLdb
except:
    import pymysql

    pymysql.install_as_MySQLdb()
    import MySQLdb

db = MySQLdb.connect(
    host="127.0.0.1",
    user="lab6",
    passwd="lab6",
    db="lab6",
    charset="utf8"
)

cursor = db.cursor(MySQLdb.cursors.DictCursor)
cursor.execute("""INSERT INTO lab_tutor
               (lastname, firstname, middlename, birthday, sex)
               VALUES
               (%s, %s, %s, %s, %s),
               (%s, %s, %s, %s, %s),
               (%s, %s, %s, %s, %s)""",
               ("Муслимов", "Петр", "Ренатович", "1965-09-09", True,
                "Ананасов", "Федор", "Сергеевич", "1971-08-16", True,
                "Хилякова", "Анна", "Львовна", "1987-04-19", False)
               )

db.commit()

cursor.execute("SELECT * FROM lab_tutor")

tutors = cursor.fetchall()

for tutor in tutors:
    print("{}: {} {} {}, {}, {}".format(tutor['id'],
                                        tutor['firstname'],
                                        tutor['lastname'],
                                        tutor['middlename'],
                                        "М" if tutor['sex'] else "Ж",
                                        tutor['birthday'].strftime("%d %m
%Y")))

cursor.execute("DELETE FROM lab_tutor WHERE 1=1")
db.commit()

cursor.close()
db.close()
```

Файл db-test-classes.py

```
try:
    import MySQLdb
except:
    import pymysql

    pymysql.install_as_MySQLdb()
    import MySQLdb


import random

class Connection:
    def __init__(self, user, password, db, host=''):
        self.user = user
        self.password = password
        self.db = db
        self._connection = None

    @property
    def connection(self):
        self.connect()
        return self._connection

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

    def connect(self):
        if not self._connection:
            self._connection = MySQLdb.connect(
                host="127.0.0.1",
                user=self.user,
                passwd=self.password,
                db=self.db,
                charset="utf8"
            )


class Tutor:
    def __init__(self, db, lastname, firstname, middlename, birthday, sex,
id=None):
        self.db = db
        self.lastname = lastname
        self.firstname = firstname
        self.middlename = middlename
        self.birthday = birthday
        self.sex = sex
        self._id = id

    def save(self):
        cursor = self.db.connection.cursor()
        if self._id is None:
            cursor.execute(
                "INSERT INTO lab_tutor (lastname, firstname, middlename,
birthday, sex) VALUES(%s, %s, %s, %s, %s)",
                (self.lastname, self.firstname, self.middlename,
self.birthday, self.sex))
            self._id = self.db.connection.insert_id()

        else:
            cursor.execute(
                "UPDATE lab_tutor SET lastname = %s, firstname = %s,
middlename = %s, birthday = %s, sex = %s WHERE id = %s",
                (self.lastname, self.firstname, self.middlename,
self.birthday, self.sex, self._id)
            )

        self.db.connection.commit()
        cursor.close()

    @staticmethod
    def select_all(db):
        cursor = db.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute("SELECT * FROM lab_tutor")

        entities = cursor.fetchall()

        entities = map(
            lambda x: Tutor(db, x['lastname'], x['firstname'],
x['middlename'], x['birthday'], x['sex'], x['id']),
            entities)

        cursor.close()

        return entities

    @staticmethod
    def clear_all(db):
        cursor = db.connection.cursor()
        cursor.execute("DELETE FROM lab_tutor WHERE 1=1")
        db.connection.commit()
        cursor.close()

    def __repr__(self):
        return "#{}: {} {} {} {} {}".format(self._id, self.lastname,
self.firstname, self.middlename, self.birthday,
                                            self.sex)


class Course:
    def __init__(self, db, name, full_name, tutor_id, id=None):
        self.db = db
        self.name = name
        self.full_name = full_name
        self.tutor_id = tutor_id._id if isinstance(tutor_id, Tutor) else
tutor_id
        self._id = id
    def save(self):
        cursor = self.db.connection.cursor()
        if self._id is None:
            cursor.execute(
                "INSERT INTO lab_course (name, full_name, tutor_id)
VALUES(%s, %s, %s)",
                (self.name, self.full_name, self.tutor_id))
            self._id = self.db.connection.insert_id()

        else:
            cursor.execute(
                "UPDATE lab_course SET name = %s, full_name = %s, tutor_id =
%s WHERE id = %s",
                (self.name, self.full_name, self.tutor_id, self._id)
            )

        self.db.connection.commit()
        cursor.close()

    @classmethod
    def select_all(self, db, tutor_id):
        cursor = db.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute("SELECT * FROM lab_course WHERE tutor_id = %s",
(tutor_id))

        entities = cursor.fetchall()

        entities = map(lambda x: Course(db, x['name'], x['full_name'],
x['tutor_id'], x['id']), entities)

        cursor.close()

        return entities


    @staticmethod
    def clear_all(db):
        cursor = db.connection.cursor()
        cursor.execute("DELETE FROM lab_course WHERE 1=1")
        db.connection.commit()
        cursor.close()

    def __repr__(self):
        return "#{}: {} {} {}".format(self._id, self.name, self.full_name,
self.tutor_id)


db = Connection("lab6", "lab6", "lab6", "127.0.0.1")

t = Tutor(db, "L", "F", "M", None, True)
t.save()

tutors = list(Tutor.select_all(db))
print(tutors)

t.lastname = "Last"
t.firstname = "First"
t.save()

tutors = list(Tutor.select_all(db))
print(tutors)
```

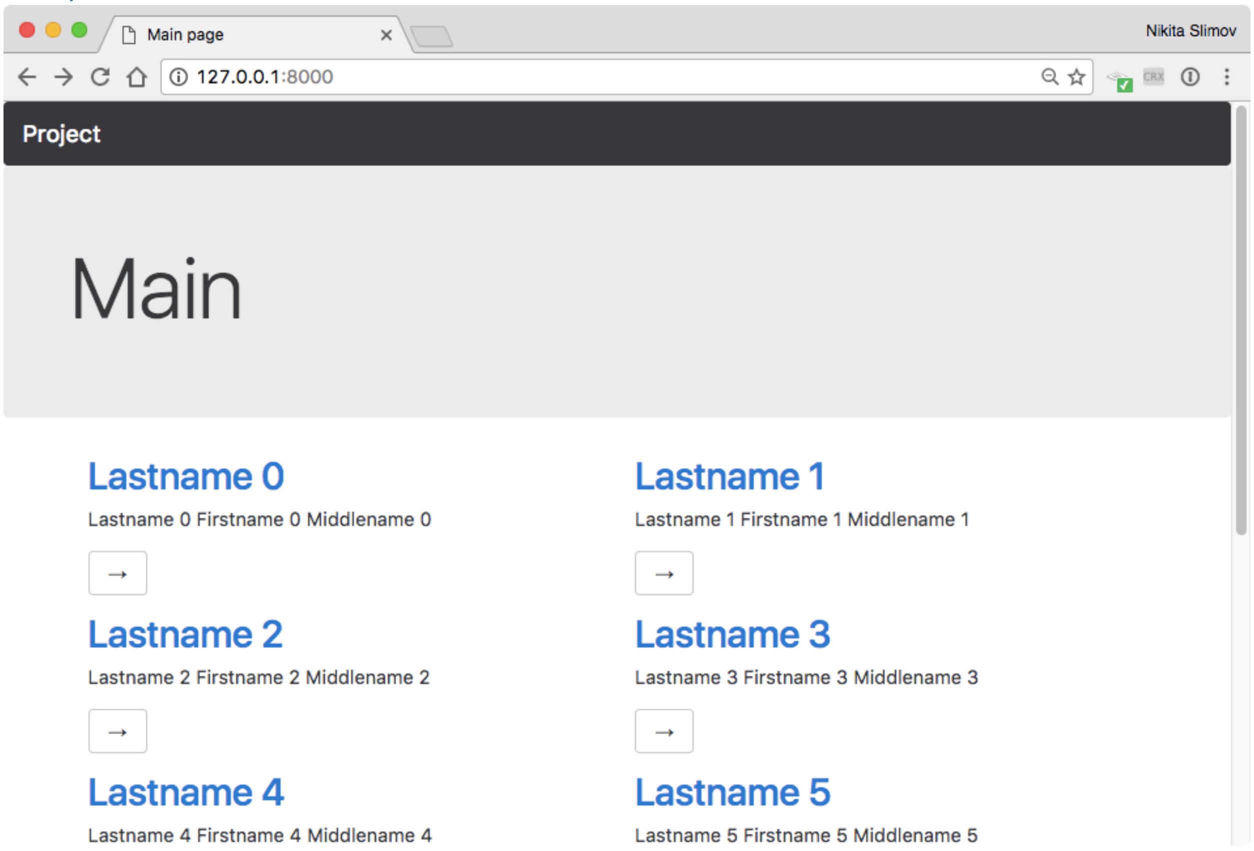
```
course = Course(db, "Name" + str(random.randint(1, 10000)), "Fullname",
tutors[0])
print(course)
course.save()
print(course)

courses = list(Course.select_all(db, tutors[0]._id))
print(courses)

Course.clear_all(db)
Tutor.clear_all(db)

db.close()
```

Результат



Main

Lastname 0

Lastname 0 Firstname 0 Middlename 0

→

Lastname 1

Lastname 1 Firstname 1 Middlename 1

→

Lastname 2

Lastname 2 Firstname 2 Middlename 2

→

Lastname 3

Lastname 3 Firstname 3 Middlename 3

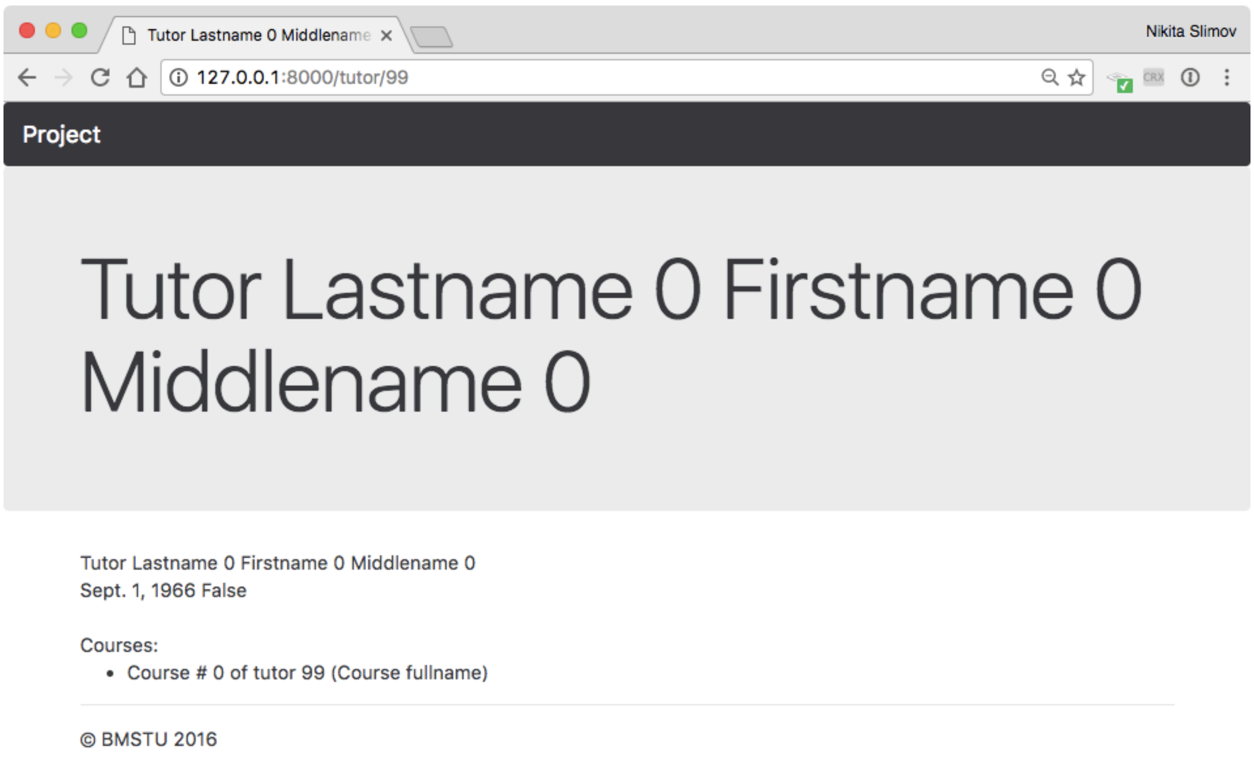
→

Lastname 4

Lastname 4 Firstname 4 Middlename 4

Lastname 5

Lastname 5 Firstname 5 Middlename 5

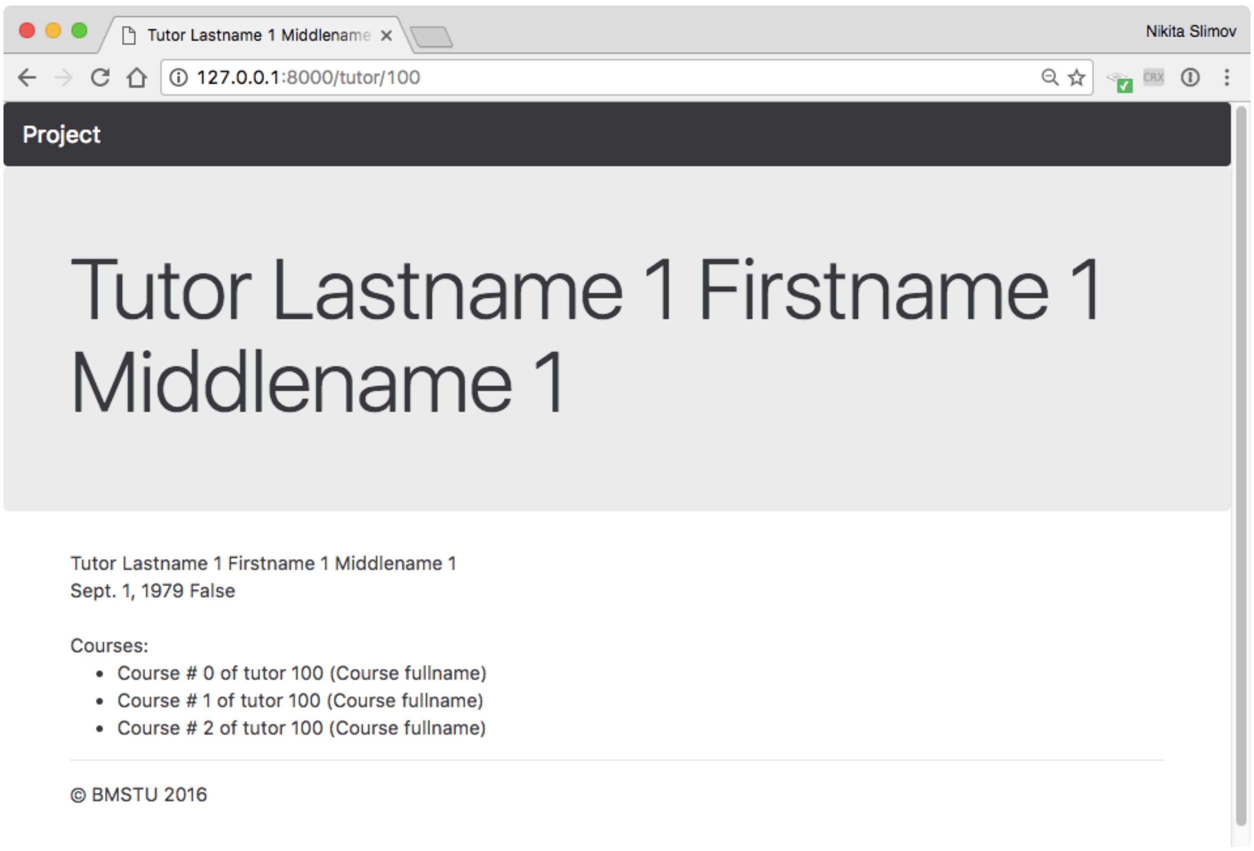


Tutor Lastname 0 Firstname 0 Middlename 0

Tutor Lastname 0 Firstname 0 Middlename 0
Sept. 1, 1966 False

Courses:
- Course # 0 of tutor 99 (Course fullname)

© BMSTU 2016

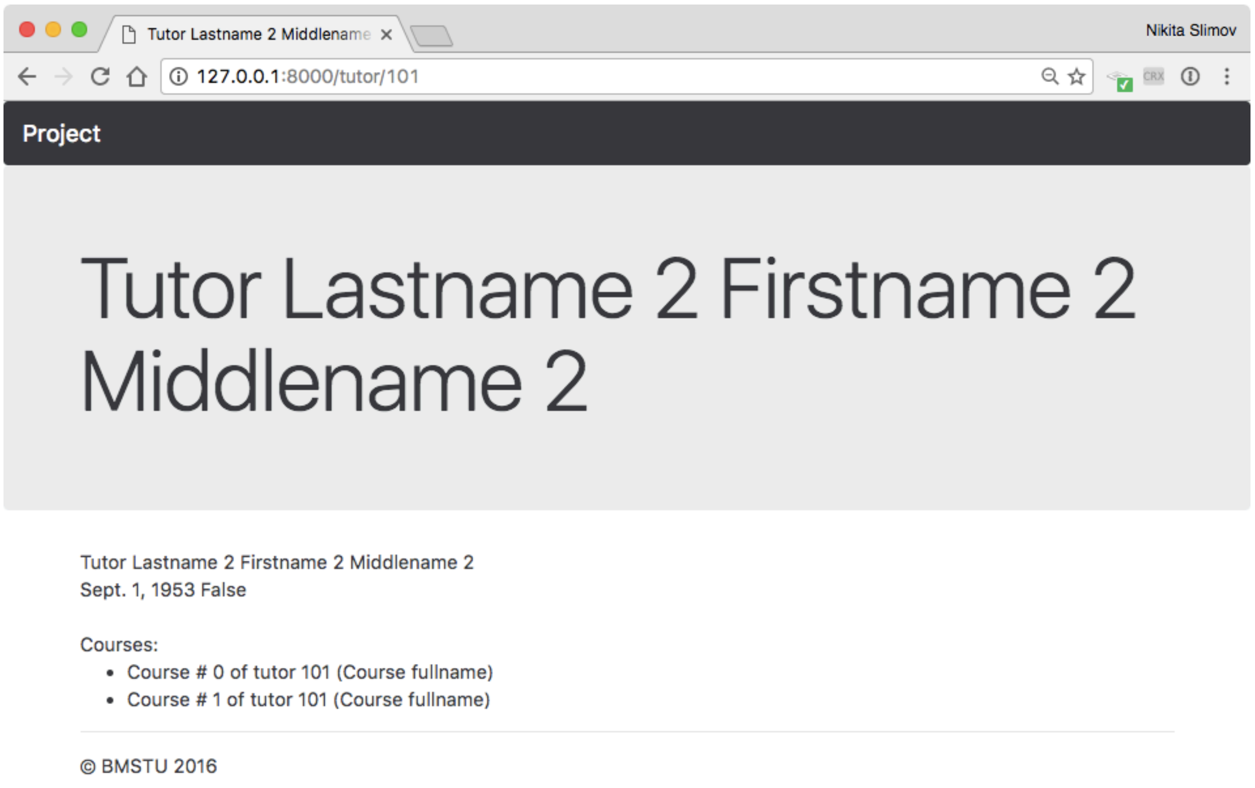


Tutor Lastname 1 Firstname 1 Middlename 1

Tutor Lastname 1 Firstname 1 Middlename 1
Sept. 1, 1979 False

Courses:
- Course # 0 of tutor 100 (Course fullname)
- Course # 1 of tutor 100 (Course fullname)
- Course # 2 of tutor 100 (Course fullname)

© BMSTU 2016



Tutor Lastname 2 Firstname 2 Middlename 2

Tutor Lastname 2 Firstname 2 Middlename 2
Sept. 1, 1953 False

Courses:

- Course # 0 of tutor 101 (Course fullname)
- Course # 1 of tutor 101 (Course fullname)

© BMSTU 2016