Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА № 5

ПО ДИСЦИПЛИНЕ «ТЕХНОЛОГИИ ВЕБ-СЕРВИСОВ»

Студент: Калинин Даниил Леонидович

Группа: P41141

Преподаватель: Дергачев Андрей Михайлович

\

Санкт-Петербург

2021

Задание:

Необходимо выполнить задание из второй работы, но с использованием REST-сервиса. Таблицу базы данных, а также код для работы с ней можно оставить без изменений

Этапы выполнения:

Для выполнения лабораторной работы необходимо создать обработчики методов для HTTP-запроса согласно CRUD-операциям.

Create	POST
Read	GET
Update	PUT
Delete	DELETE

Обработчик GET-запросов был создан в рамках предыдущей лабораторной работы.

1. POST

```
@POST
@Consumes(MediaType.APPLICATION_JSON)
public Response addPerson(Person person) throws ServerException,
EmptyRequestException {
        mdb.executeUpdateQuery(sqlQueryBuilder.buildInsertQuery(person));
        List<Person> added =
mdb.getPersonsBySqlQuery(sqlQueryBuilder.buildSelectQuery(person));
        return Response.status(200).entity(
                "Person added. id : " + added.get(added.size()-
1).getId()).build();
}
```

Метод принимает на вход объект Person, который автоматически создается из тела POST-запроса, затем добавляет его в базу и, в случае успеха, возвращает ответ со статусом 200 – OK и ID созданного пользователя.

2. PUT

```
@PUT
@Produces({MediaType.APPLICATION_JSON})
public Response updatePersons(Person newPerson, @QueryParam("id") int id)
throws PersonDoesNotExistException, ServerException, EmptyRequestException {
    if(!mdb.checkIfPersonExists(sqlQueryBuilder.buildSelectQuery(new
Person(id))))
        throw new PersonDoesNotExistException("Person you trying to change
was not found!");
    String result =
mdb.executeUpdateQuery(sqlQueryBuilder.buildUpdateQuery(id, newPerson));
    return Response.status(200).entity(result).build();
}
```

Метод принимает объект Person в теле запроса и ID в параметрах, затем возвращает информацию об успехе или неуспехе операции обновления данных.

3. DELTE

```
@DELETE
@Produces({MediaType.APPLICATION_JSON})
public Response deletePersons(@QueryParam("id") int id) throws
ServerException, EmptyRequestException, PersonDoesNotExistException {
        if(!mdb.checkIfPersonExists(sqlQueryBuilder.buildSelectQuery(new
Person(id))))
            throw new PersonDoesNotExistException("Person you trying to
change was not found!");
        String result =
mdb.executeUpdateQuery(sqlQueryBuilder.buildDeleteQuery(id));
        return Response.status(200).entity(result).build();
}
```

Метод принимает ID в параметрах запроса, затем возвращает информацию об успехе или неуспехе операции удаления данных.

4. В клиент были добавлены соответствующие методы для отправки запросов

```
public boolean addPerson(Person person){
    WebResource webResource = client.resource(URL);
    ClientResponse response =

webResource.type(MediaType.APPLICATION_JSON).post(ClientResponse.class,
person);
    response.bufferEntity();
    System.out.println(response.getEntity(String.class));
    return response.getStatus() == 200;
}

public boolean updatePerson(int id, Person newPerson){
    WebResource webResource = client.resource(URL);
    webResource.queryParam("id", String.valueOf(id));
    ClientResponse response =

webResource.type(MediaType.APPLICATION_JSON).put(ClientResponse.class,
newPerson);
    response.bufferEntity();
    System.out.println(response.getEntity(String.class));
    return response.getStatus() == 200;
}

public boolean deletePerson(int id){
    WebResource webResource = client.resource(URL);
    webResource = webResource.queryParam("id", String.valueOf(id));
    ClientResponse response =
            webResource.delete(ClientResponse.class);
    response.bufferEntity();
    System.out.println(response.getEntity(String.class));
```

```
    return response.getStatus() == 200;
}
```

Выводы:

В ходе выполнения лабораторной работы были добавлены обработчики PUT, POST и DELETE – методов. На клиенте были добавлены методы обработки соответствующих запросов.

Вопросы:

Ссылка на GitHub:

https://github.com/KalininDL/web_services_spring