

# Отчет по лабораторной работе 6

Студентов группы ПИМ-21 Бубенцова С.А. Носкова И.А.

---

## 1 Постановка задачи

В процессе выполнения лабораторной работы необходимо выполнить следующие задачи:

1. Реализовать три сущности.
  2. Реализовать CRUD операции для каждой сущности.
  3. (\*) Форматировать поле с типом Date с помощью библиотеки Jackson.
- 

## 2 Выполнение

### 2.1 Структура проекта

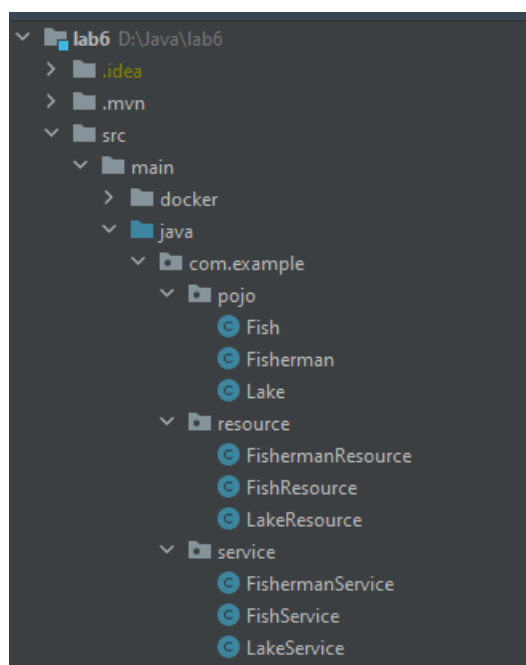


Рисунок 1. Структура проекта

### 2.2 Задание

Были разработаны сущности: Fisherman, Lake, Fish.

Листинг 1. Листинг класса Fisherman.

```

package com.example.pojo;

import com.fasterxml.jackson.annotation.JsonFormat;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
import java.util.Date;

@Entity
@Table(name = "fisherman")
public class Fisherman {
    //    Уникальное имя рыбака
    @Id
    private String login;
    //    Дата рождения рыбака
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd-MM-yyyy")
    private Date birthDate;
    //    Пол рыбака
    private String sex;
    //    ФИО рыбака
    private String fio;
    //    Контактный телефон рыбака
    private String phoneNumber;
    //    Адрес проживания рыбака
    private String address;

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getSex() {
        return sex;
    }

    public void setSex(String sex) {
        this.sex = sex;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public String getFio() {
        return fio;
    }

    public void setFio(String fio) {
        this.fio = fio;
    }

    public Date getBirthDate() {
        return birthDate;
    }

    public void setBirthDate(Date birthDate) {
        this.birthDate = birthDate;
    }

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }
}

```

*Листинг 2. Листинг класса Lake.*

```

package com.example.pojo;

import javax.persistence.*;

@Entity
@Table(name = "lake")
public class Lake {
    //    Уникальный идентификатор озера
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    //    Площадь озера
    private Double area;
    //    Глубина озера
    private Double depth;
    //    Название озера
    private String name;

    public Long getId() {
        return id;
    }

    public Double getArea() {
        return area;
    }

    public void setArea(Double area) {
        this.area = area;
    }

    public Double getDepth() {
        return depth;
    }

    public void setDepth(Double depth) {
        this.depth = depth;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}

```

*Листинг 3. Листинг класса Fish.*

```

package com.example.pojo;

import javax.persistence.*;

@Entity
@Table(name = "fish")
public class Fish {
    //    Уникальный идентификатор рыбы
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    //    Название семейства рыб
    private String kind;
    //    Название рыбы
    private String name;
    //    Средняя глубина обитания рыбы в озере
    private Double depth;
    //    Средний вес рыбы
    private Double weight;

    public Long getId() {
        return id;
    }

    public String getKind() {
        return kind;
    }

    public void setKind(String kind) {
        this.kind = kind;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Double getDepth() {
        return depth;
    }

    public void setDepth(Double depth) {
        this.depth = depth;
    }

    public Double getWeight() {
        return weight;
    }

    public void setWeight(Double weight) {
        this.weight = weight;
    }
}

```

Далее были поддержаны следующие операции для каждой сущности:

1. Создание.
2. Получение.
3. Замена.
4. Удаление.

Класс, содержащий операции для работы с сущностью Fisherman:

*Листинг 4. Листинг класса FishermanResource*

```

package com.example.resource;

import com.example.pojo.Fisherman;
import com.example.service.FishermanService;

import javax.inject.Inject;
import javax.ws.rs.*;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

@Path("/fisherman")
public class FishermanResource {

    @Inject
    FishermanService fishermanService;

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    @Path("/get")
    public Response getFishermen() {
        return Response.ok(fishermanService.getFishermen()).build();
    }

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    @Path("/init")
    public Response getFisherman() {
        return Response.ok(fishermanService.initFisherman()).build();
    }

    @POST
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_JSON)
    @Path("/update")
    public Response updateFisherman(Fisherman fisherman) {
        return Response.ok(fishermanService.updateFisherman(fisherman)).build();
    }

    @POST
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_JSON)
    @Path("/delete")
    public Response deleteFisherman(Fisherman fisherman) {
        return Response.ok(fishermanService.deleteFisherman(fisherman)).build();
    }

    @POST
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_JSON)
    @Path("/insert")
    public Response insertFisherman(Fisherman fisherman) {
        return Response.ok(fishermanService.insertFisherman(fisherman)).build();
    }
}

```

Класс, содержащий операции для работы с сущностью Lake:

*Листинг 5. Листинг класса LakeResource*

```

package com.example.resource;

import com.example.pojo.Lake;
import com.example.service.LakeService;

import javax.inject.Inject;
import javax.ws.rs.*;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

@Path("/lake")
public class LakeResource {
    @Inject
    LakeService lakeService;

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    @Path("/get")
    public Response getLakes() {
        return Response.ok(lakeService.getLakes()).build();
    }

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    @Path("/init")
    public Response getLake() {
        return Response.ok(lakeService.initLake()).build();
    }

    @POST
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_JSON)
    @Path("/update")
    public Response updateLake(Lake lake) {
        return Response.ok(lakeService.updateLake(lake)).build();
    }

    @POST
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_JSON)
    @Path("/delete")
    public Response deleteLake(Lake lake) {
        return Response.ok(lakeService.deleteLake(lake)).build();
    }

    @POST
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_JSON)
    @Path("/insert")
    public Response insertLake(Lake lake) {
        return Response.ok(lakeService.insertLake(lake)).build();
    }
}

```

Класс, содержащий операции для работы с сущностью Fish:

*Листинг 6. Листинг класса FishResource*

```

package com.example.resource;

import com.example.pojo.Fish;
import com.example.service.FishService;

import javax.inject.Inject;
import javax.ws.rs.*;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

@Path("/fish")
public class FishResource {

    @Inject
    FishService fishService;

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    @Path("/get")
    public Response getFishes() {
        return Response.ok(fishService.getFishes()).build();
    }

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    @Path("/init")
    public Response getFish() {
        return Response.ok(fishService.initFish()).build();
    }

    @POST
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_JSON)
    @Path("/update")
    public Response updateFish(Fish fish) {
        return Response.ok(fishService.updateFish(fish)).build();
    }

    @POST
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_JSON)
    @Path("/delete")
    public Response deleteFish(Fish fish) {
        return Response.ok(fishService.deleteFish(fish)).build();
    }

    @POST
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_JSON)
    @Path("/insert")
    public Response insertFish(Fish fish) {
        return Response.ok(fishService.insertFish(fish)).build();
    }
}

```

Также было выполнено форматирование даты, с помощью библиотеки Jackson. Был задан формат вида: dd-MM-yyyy.

#### *Листинг 7. Задание формата*

```

//    Дата рождения рыбака
@JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd-MM-yyyy")
private Date birthDate;

```

---

## 3 Результаты выполнения

Для визуализации ресурсов и взаимодействия с ними, была подключена библиотека swagger-ui.

## Fish Resource ^

|      |              |   |
|------|--------------|---|
| POST | /fish/delete | ▼ |
| GET  | /fish/get    | ▼ |
| GET  | /fish/init   | ▼ |
| POST | /fish/insert | ▼ |
| POST | /fish/update | ▼ |

## Fisherman Resource ^

|      |                   |   |
|------|-------------------|---|
| POST | /fisherman/delete | ▼ |
| GET  | /fisherman/get    | ▼ |
| GET  | /fisherman/init   | ▼ |
| POST | /fisherman/insert | ▼ |
| POST | /fisherman/update | ▼ |

## Lake Resource ^

|      |              |   |
|------|--------------|---|
| POST | /lake/delete | ▼ |
| GET  | /lake/get    | ▼ |
| GET  | /lake/init   | ▼ |
| POST | /lake/insert | ▼ |
| POST | /lake/update | ▼ |

Проверка работы CRUD операций для сущности Fisherman.

POST /fisherman/insert ^

Parameters Cancel Reset

No parameters

Request body application/json ▼

```
{  "login": "test12",  "birthDate": "28-11-2021",  "sex": "male",  "fio": "Ivanov Ivan Ivanovich",  "phoneNumber": "123-1314-32",  "address": "Rybinsk city"}
```

Execute Clear

Responses

Рисунок 2. Создание записи



Code

Details

200

Response body

```
[
  {
    "login": "test12",
    "birthDate": "28-11-2021",
    "sex": "male",
    "fio": "Ivanov Ivan Ivanovich2",
    "phoneNumber": "123-1314-32",
    "address": "Rybinsk city"
  },
  {
    "login": "test123",
    "birthDate": "28-11-2021",
    "sex": "male",
    "fio": "Ivanov Ivan Ivanovich",
    "phoneNumber": "123-1314-32",
    "address": "Rybinsk city"
  }
]
```

 [Download](#)

Response headers

```
content-length: 283
content-type: application/json
```

Responses

| Code | Description | Links    |
|------|-------------|----------|
| 200  | OK          | No links |

Рисунок 3. Вывод всех записей класс Fisherman

Code

Details

200

Response body

```
{  "login": "test12",  "birthDate": "28-11-2021",  "sex": "male",  "fio": "Ivanov Ivan Ivanovich2",  "phoneNumber": "123-1314-32",  "address": "Rybinsk city"}
```

Download

Response headers

```
content-length: 140content-type: application/json
```

Responses

| Code | Description | Links    |
|------|-------------|----------|
| 200  | OK          | No links |

Рисунок 4. Удаление указанной записи


| Code      | Details  |          |
|-----------|--|----------|
| 200       | <div><div>Response body</div><div><pre>{  "login": "test12",  "birthDate": "28-11-2021",  "sex": "male",  "fio": "Ivanov Ivan Ivanovich",  "phoneNumber": "123-1314-32",  "address": "Rybinsk city"}</pre><div> <a href="#">Download</a></div></div><div><div>Response headers</div><pre>content-length: 139 content-type: application/json</pre></div></div> |          |
| Responses |  |          |
| Code      | Description  | Links    |
| 200       | OK   | No links |

Рисунок 5. Изменение указанной записи

## Вывод

В результате выполнения лабораторной работы, мы попрактиковались в работе с базой данных в quarkus. В итоге, были реализованы три сущности и CRUD операции для них, для визуализации ресурсов и взаимодействия с ними использовалась библиотека swagger-ui.