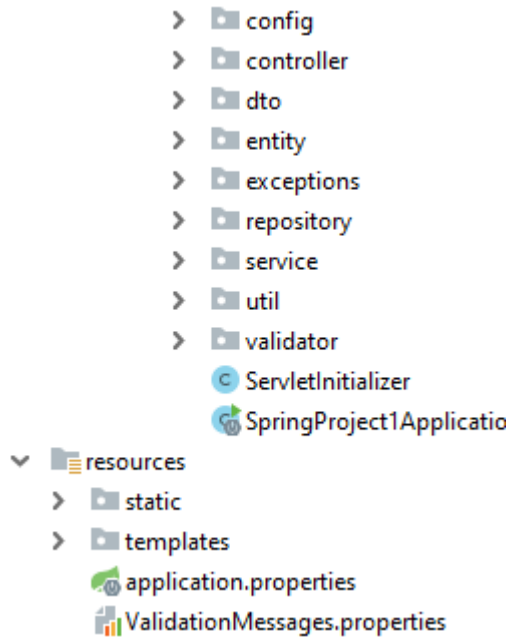
Задание №4 Spring Data. JPA, Hibernate. Service, Repository

1. Поменяйте структуру проекта на *Controller*, *Service*, *Repository*.
   Repository - отвечает за работу с данными, **Service** - определяет логику и **Controller** будет только обрабатывать запросы и вызывать нужные методы сервиса.
   У вас должна быть примерно следующая структура пакетов.



Пакет **dto** - data transfer object - для передачи объектов в базу данных и обратно.

Пакет **repository** – специальная прослойка, которая будет отвечать исключительно за доступ к данным (работа с базой данных). Там будут интерфейсы унаследованные от CrudRepository или JpaRepository.

Сделайте пакет со своими **исключениями**.

Пакет **entity** будет хранить сущности.

В **service** будут сервисы, которые вызывают методы репозитория.

Пакет **controller** должен содержать контроллеры, откуда будут вызываться методы сервисов.

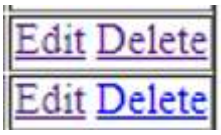
2. Добавьте поддержку базы данных (MySQL, Oracle, MongoDB и др).

3. Определите Entity и сгенерируйте базу данных на их основе.

4. Добавьте дополнительные сущности связанные с вашими «объектами» на основе @OneToMany, @ManyToOne, @ManyToMany.

Например:

1	*Студент, Предметы*
2	*Работник (компания), выполненные проекты*
3	*Товар, Магазины*
4	*Фильм , Актеры*
5	*Альбом (музыкальный), исполнитель*
6	*Транспорт, Маршруты*
7	*Игра, Игроки*
8	*Приложение, Издатель*
9	*Книга, Авторы*
10	*Задача, Исполнители*
11	*Недвижимость, Агенты*
12	*Документ, Владелец*
13	*Компьютер, Пользователи*
14	*Проект, Исполнители (работники)*

5. Создайте таблицы с логином и паролем (используйте шифрование для пароля) и ролями пользователей (админы и юзеры).
6. Создайте страницу входа в систему.
7. Добавьте операции удаления и редактирования «**объектов**» вашей системы. (можно сделать в таблице ссылки на операции 

)

.

Указания к выполнению

Рассмотрим процесс создания приложения Spring, связанного с базой данных MySQL. Будем использовать Spring Data JPA для доступа к базе данных (есть другие технологии, например вы можете использовать обычный Spring JDBC).

1) Добавляем Maven зависимости

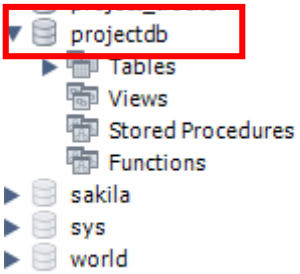
В файле pom.xml у вас уже была зависимость

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Добавим еще

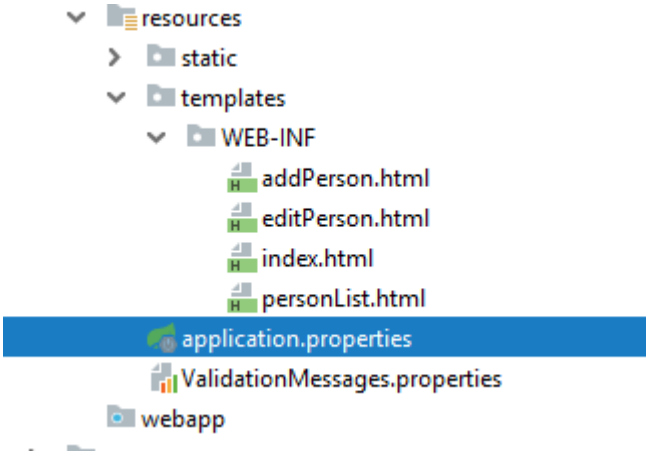
```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.15</version>
</dependency>
```

2) Откройте MySQL клиента и создайте базу данных. Например, через WorkBench



3) Настройка соединения.

Spring Boot предоставляет настройки по умолчанию. И еще по умолчанию используется H2 база данных, поэтому, если вы хотите изменить тип и использовать любую другую базу данных, вы должны определить атрибуты соединения в файле application.properties.



Определите параметры:

```
spring.jpa.hibernate.ddl-auto=create
spring.datasource.url=jdbc:mysql://localhost:3306/projectdb?useSSL=true
spring.datasource.username=root
spring.datasource.password=root
```

Здесь *spring.jpa.hibernate.ddl-auto* может иметь значения:

- **none** - это значение по умолчанию для MySQL, говорит что надо оставить без изменений структуру базы данных.
- **update** - Hibernate изменяет базу данных в соответствии с заданными структурами сущностей.
- **create** - создает базу данных каждый раз, но не удаляет ее при закрытии.
- **create-drop** -создает базу данных, а затем удаляет ее, когда SessionFactory закрывается.

Мы поставим **create**, потому что у нас еще нет структуры базы данных, у нас есть entity, которую мы хотели бы отобразить. После первого запуска можете изменить это значение на **update** или **none** в соответствии с требованиями программы. Используйте **update**, если хотите внести некоторые изменения в структуру базы данных.

По умолчанию для H2 и других встроенных баз данных является **create-drop**, но для других, таких как MySQL по умолчанию **none**.

Хорошей практикой безопасности является изменение значения на **none**, после того, как ваша база данных находится в рабочем состоянии. После чего вы отнимаете все привилегии у пользователя MySQL, подключенного к приложению Spring, а затем даете ему только SELECT, UPDATE, INSERT, DELETE.

4) Создание Entity

Определите все сущности в системе

Например,

```
@Data
@Entity
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "activity")
public class Activity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;

    @ManyToOne
    @JoinColumn(name = "creator")
    private User owner;


    @Column(name = "information")
    private String info;
}
```

5) Создание репозитория

Создайте репозиторий с автоматической реализацией. Если потребуется напишите запросы.

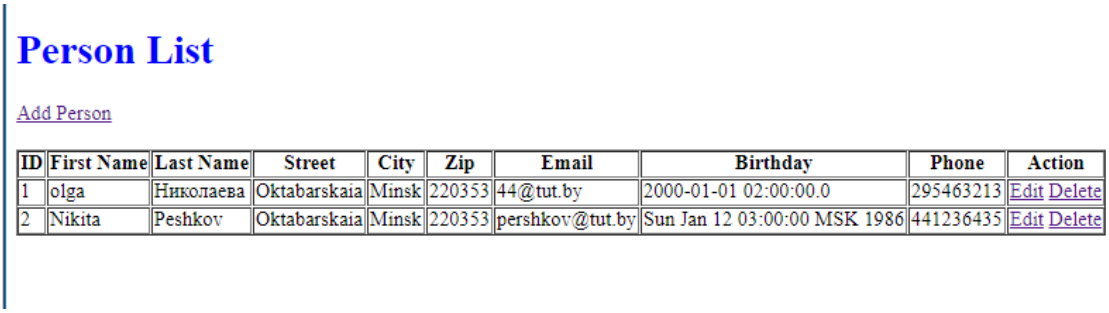
```
@Repository
public interface PersonRepository extends CrudRepository<Person, Long> {



}
```

Вместо СкгвRepository можно использовать JpaRepository.

6) Контроллеры

Котроллер был в предыдущем проекте и остается без изменения.

Все можно запускать приложение и тестировать. Создадим пару записей

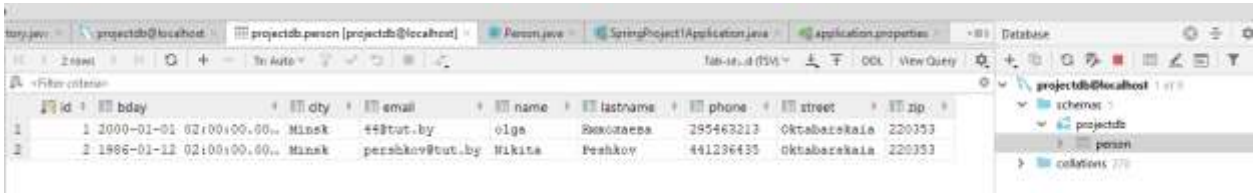


Person List

Add Person

ID	First Name	Last Name	Street	City	Zip	Email	Birthday	Phone	Action
1	olga	Николаева	Oktabarskaia	Minsk	220353	44@tut.by	2000-01-01 02:00:00.0	295463213	Edit Delete
2	Nikita	Peshkov	Oktabarskaia	Minsk	220353	pershkov@tut.by	Sun Jan 12 03:00:00 MSK 1986	441236435	Edit Delete

Зайдите и проверьте создание таблицы person и ее заполнение



id	bday	city	email	name	lastname	phone	street	zip
1	2000-01-01 02:00:00.00…	Minsk	44@tut.by	olga	Николаева	295463213	Oktabarskaia	220353
2	1986-01-12 02:00:00.00…	Minsk	pershkov@tut.by	Nikita	Peshkov	441236435	Oktabarskaia	220353

Выполните несколько операций модификации и удаления.

В Spring Boot 1 был пул соединений по умолчанию Tomcat, но в Spring Boot 2 он был изменен на HikariCP. Вы могли видеть лог запуска приложения. Для настройки HikariCP надо будет внести в файл свойств настроек

```
# spring.datasource.hikari.*
spring.datasource.hikari.connection-timeout=60000
spring.datasource.hikari.maximum-pool-size=5
spring.datasource.type=com.zaxxer.hikari.HikariDataSource
```