

Data Engineering 0: основы баз данных и SQL - Семинар 6

Структура семинара

Семинар 6 разделен на 2 части.

В первой части семинара студенты демонстрируют свои домашние задания, а в аудитории проводится коллективное обсуждение, согласны ли одноклассники с решением и что можно было бы в нем поменять. За демонстрацию своего ДЗ студенты получают балл за активность на семинаре.

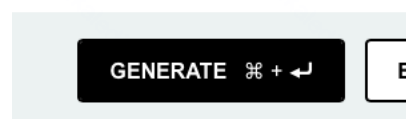
Во второй части семинара семинарист показывает пример простого SpringBoot приложения, который взаимодействует с базой данных.

Создание backend-приложения

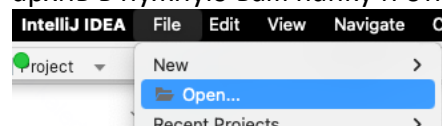
Для начального создания backend-приложения удобно воспользоваться сайтом <https://start.spring.io>. На странице ниже нужно указать название вашего будущего приложения, зависимости, которые понадобятся вам (их можно подключить и позже вручную) и другие опции.

The screenshot shows the Spring Initializr website interface. On the left, there are sections for 'Project' (with radio buttons for Gradle - Groovy, Gradle - Kotlin, and Maven), 'Language' (with radio buttons for Java, Kotlin, and Groovy), 'Spring Boot' (with radio buttons for various versions including 4.0.0, 3.5.0, 3.5.0 (RC1), 3.4.6, 3.4.5, 3.3.12, and 3.3.11), and 'Project Metadata' (with input fields for Group, Artifact, Name, Description, and Package name, and radio buttons for Packaging and Java version). On the right, there is a 'Dependencies' section with a button 'ADD DEPENDENCIES... + B' and three pre-selected dependencies: 'Spring Web' (WEB), 'JDBC API' (SQL), and 'PostgreSQL Driver' (SQL). Each dependency has a brief description.

Когда все опции выбраны, нужно нажать внизу страницы Generate.



После этого на ваш компьютер будет скачан архив с приложением. Распаковываете архив в нужную вам папку и открываете приложение из IDEA.



Основным классом является *ВашеНазваниеПриложенияApplication*. Из него можно запускать ваше приложение

```
3 import ...
5
6 @SpringBootApplication
7 public class DbConnectorApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(DbConnectorApplication.class, args);
11     }
12
13 }
14
```

Как общаться с приложением?

С запущенным локально или в облаке приложением обычно можно взаимодействовать (иначе зачем оно вам). Одним из способов взаимодействия являются REST-запросы. Самый простой способ отправить запрос к вашему локально запущенному приложению – написать адрес приложения и желаемый метод прямо в адресной строке. Если выполнить приведенный ниже запрос, то браузер отправит GET запрос на адрес localhost:8888 по протоколу HTTP и попытается достучаться до функции /test.

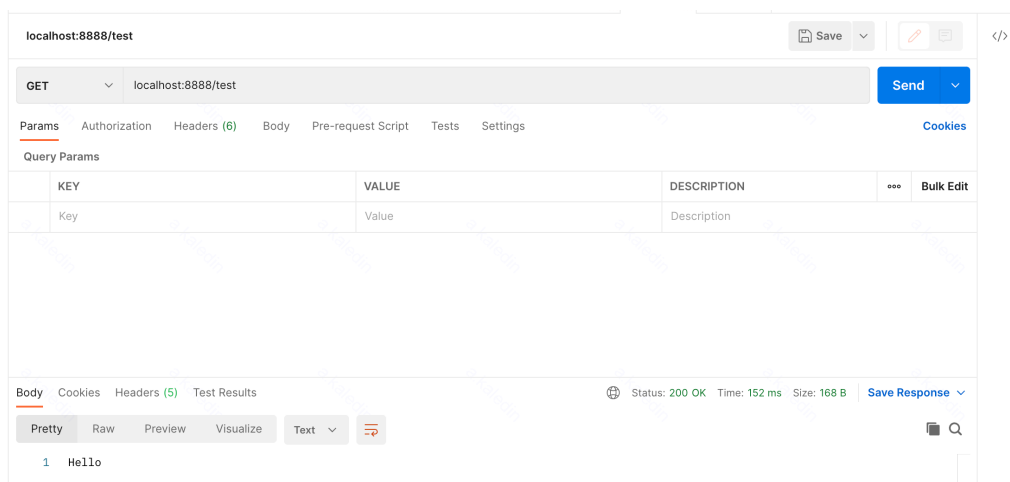
<http://localhost:8888/test>

Если в приложении будет существовать метод как на картинке ниже, то пользователю в браузере будет возвращен текст Hello на пустом фоне.

```
@RestController
public class TestController {

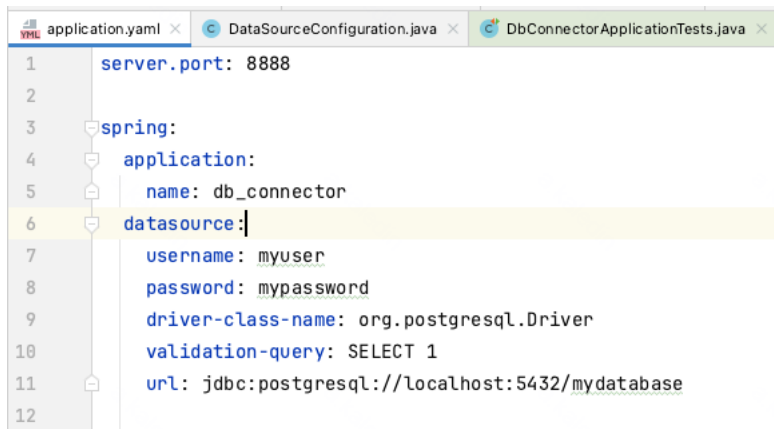
    @GetMapping("/test")
    public String hello() {
        return "Hello";
    }
}
```

Для более полноценного общения с приложением можно использовать программу Postman.



Как настроить подключение приложения к БД?

У приложения конфигурации хранятся в файле `application`. Этот файл может быть в формате `yaml` / `properties` / `xml` / и т.д.



```
1 server.port: 8888
2
3 spring:
4   application:
5     name: db_connector
6   datasource:
7     username: myuser
8     password: mypassword
9     driver-class-name: org.postgresql.Driver
10    validation-query: SELECT 1
11    url: jdbc:postgresql://localhost:5432/mydatabase
12
```

В конфиге выше предполагается, что база данных PostgreSQL работает у вас локально на порту 5432, логин для подключения `myuser`, а пароль – `mypassword`.

Примеры запросов и ответов от сервиса

Следующие методы позволят выполнять `sql` запросы из приложения к базе данных. Метод `/count` возвращает общее количество записей в таблице `students`, а метод `/add` позволяет добавлять новых студентов.



```
@RestController
@AllArgsConstructor
public class StudentController {

    @Autowired
    public NamedParameterJdbcTemplate myNamedParameterJdbcTemplate;

    @GetMapping("/count")
    public Integer getStudentsCount() {
        var sqlQuery = "select count(*) from students";
        var parameters = new MapSqlParameterSource();
        return myNamedParameterJdbcTemplate.queryForObject(sqlQuery, parameters, Integer.class);
    }

    @PostMapping("/add")
    public int addStudent(@RequestParam("fullName") String fullName, @RequestParam("enrollmentYear") Integer enrollmentYear) {
        var sqlQuery = "insert into students (full_name, enrollment_year) values (:full_name, :enrollment_year)";
        var parameters = new MapSqlParameterSource();
        parameters.addValue( paramName: "full_name", fullName);
        parameters.addValue( paramName: "enrollment_year", enrollmentYear);
        return myNamedParameterJdbcTemplate.update(sqlQuery, parameters);
    }
}
```

Обращаться к этим методам можно, к примеру, вот так:

localhost:8888/count

Save

GET

localhost:8888/count

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time: 39 ms

Size: 166 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

12

localhost:8888/add?fullName=Eric Cartman&enrollmentYear=2015

Save

POST

localhost:8888/add?fullName=Eric Cartman&enrollmentYear=2015

Send

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	fullName	Eric Cartman			
<input checked="" type="checkbox"/>	enrollmentYear	2015			
	Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time: 340 ms

Size: 165 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

1

students

Enter a SQL expression to filter results (use Ctrl+Space)

	123 id	ABC full_name	123 enrollment_year
1	1	Алексей Смирнов	2,022
2	2	Мария Иванова	2,023
3	3	Сергей Кузнецов	2,021
4	4	Анна Попова	2,022
5	5	Дмитрий Соколов	2,023
6	6	Ольга Лебедева	2,022
7	7	Игорь Морозов	2,021
8	8	Наталья Козлова	2,023
9	9	Евгений Новиков	2,022
10	10	Татьяна Михайлова	2,021
11	11	Lol Kek	2,015
12	12	Eric Cartman	2,015

ДЗ

В качестве дз можно выбрать одно из двух заданий на ваш личный выбор:

Задание 1

В рамках предыдущих ДЗ вам нужно было работать над схемой данных для интернет-магазина.

В рамках ДЗ6 вам необходимо создать свое backend-приложение, которое будет работать с моделью данных интернет-магазина, которую вы создали ранее.

Приложение должно уметь обрабатывать следующие запросы:

1. Создание товара
2. Редактирование каких-то полей товара
3. Создание новой категории
4. Привязка товара к категории
5. Получение всех товаров вместе с названиями их категорий (намек на join-запрос)

Сдать ДЗ нужно в ваш приватный гитхаб-репозиторий:

1. В репозитории должны содержаться файлы springboot приложения (весь проект, чтобы проверяющий мог его скачать и запустить)
2. PDF файл со скринами запросов и результатов их выполнения + ваше описание, что это за запросы и на какие вопросы отвечают / зачем вы их выполнили

Задание 2

Вам дана часть DDL-схемы интернет-магазина. Нужно проанализировать, как улучшить производительность запросов из пункта 1, а также написать SQL-запросы для пункта 2.

Схема данных:

```
CREATE TABLE users (  
    id SERIAL PRIMARY KEY,  
    name TEXT NOT NULL,  
    email TEXT UNIQUE,  
    registration_date DATE DEFAULT NOW()  
);  
  
CREATE TABLE products (  
    id SERIAL PRIMARY KEY,  
    title TEXT NOT NULL,  
    price DECIMAL(10, 2),  
    category TEXT  
);  
  
CREATE TABLE orders (  
    id SERIAL PRIMARY KEY,  
    user_id INT NOT NULL,  
    product_id INT NOT NULL,  
    quantity INT DEFAULT 1,  
    order_date DATE NOT NULL,  
    FOREIGN KEY (user_id) REFERENCES users(id),  
    FOREIGN KEY (product_id) REFERENCES products(id)  
);
```

Пример данных:

```
INSERT INTO users (name, email) VALUES  
( 'Эрик Картман', 'e.cartman@hse.ru'),  
( 'Стэн Марш', 's.marsh@lol.com'),  
( 'Баттерс Сточ', 'b.stotch@sp.uk');
```

```
INSERT INTO products (title, price, category) VALUES  
( 'Футболка хлопковая', 1499.50, 'Одежда'),  
( 'Ноутбук Pro Extra 15', 120000.00, 'Электроника');
```

```
INSERT INTO orders (user_id, product_id, quantity, order_date) VALUES  
(1, 1, 3, '2024-03-15'),  
(1, 1, -2, '2024-03-20'), -- отрицательные quantity обозначают возвраты  
(2, 2, 1, '2023-12-01');
```

Пункт 1. Предложите изменения схемы данных (и напишите запросы, изменяющие схему), чтобы улучшить производительность следующих запросов:

1. Поиск пользователей по email
2. Фильтрация товаров по category и price

Пункт 2. Напишите запросы, отвечающие на следующие вопросы:

1. Выведите email пользователей и названия товаров, которые они заказывали в 2024 году. Включите только заказы с количеством товаров > 2.
Поля результата: user_email, product_title, order_date, quantity
2. Для каждого пользователя выведите общее количество заказанных и возвращенных им товаров
Поля результата: user_id, received_products, returned_products
3. Выведите товары, которые ЛИБО никогда не заказывались, ЛИБО были заказаны более 3 раз
Поля результата: product_id, title, status ("Не заказан" или "Много заказов")
4. Для каждого пользователя выведите общее количество его заказов и самый дорогой товар, который он когда-либо заказывал.
Поля результата: user_name, total_orders, most_expensive_product_name, most_expensive_product_price

Сдать ДЗ нужно в ваш приватный гитхаб-репозиторий в виде PDF файла со скринами запросов и результатов их выполнения + ваше описание, что это за запросы и на какие вопросы отвечают / зачем вы их выполнили