

## Практическая работа Java

Настроим бд:

```
Java > demo > src > main > resources > ! application.yml
1  spring:
2    application:
3      name: demo
4
5    security:
6      basic:
7        enabled: true
8      user:
9        name: "admin"
10       password: "admin123"
11       roles: "ADMIN"
12
13   datasource:
14     url: "jdbc:h2:mem:taskdb"
15     driver-class-name: "org.h2.Driver"
16     username: "sa"
17     password: ""
18
19   jpa:
20     database-platform: "org.hibernate.dialect.H2Dialect"
21     show-sql: true
22     hibernate:
23       ddl-auto: update
24
25   h2:
26     console:
27       enabled: true
28       path: /h2-console
29       settings:
30         web-allow-others: false
31         trace: false
```

Создание модели задачи:

```
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <optional>true</optional>
</dependency>
```

```

1  package com.example.demo.model;
2
3  import jakarta.persistence.Entity;
4  import jakarta.persistence.GeneratedValue;
5  import jakarta.persistence.GenerationType;
6  import jakarta.persistence.Id;
7  import lombok.Data;
8  import lombok.NoArgsConstructor;
9  import lombok.AllArgsConstructor;
10
11  import java.time.LocalDateTime;
12
13  @Entity
14  @Data
15  @NoArgsConstructor
16  @AllArgsConstructor
17  public class Task {
18      @Id
19      @GeneratedValue(strategy = GenerationType.IDENTITY)
20      private Long id;
21
22      private String title;
23      private String description;
24      private String status;
25      private LocalDateTime createdAt;
26  }

```

Создание репозитория задач:

```

Java > src > main > java > com > example > demo > repository > TaskRepository.java
1  package com.example.demo.repository;
2
3  import com.example.demo.model.Task;
4  import org.springframework.data.jpa.repository.JpaRepository;
5  import org.springframework.stereotype.Repository;
6
7  @Repository
8  public interface TaskRepository extends JpaRepository<Task, Long> {
9  }
10

```

Создание REST контроллера

Класс для обработки ошибок:

```
Java > demo > src > main > java > com > example > demo > exception > J ResourceNotFound
1  package com.example.demo.exception;
2
3  import org.springframework.http.HttpStatus;
4  import org.springframework.web.bind.annotation.ResponseStatus;
5
6  @ResponseStatus(HttpStatus.NOT_FOUND)
7  public class ResourceNotFoundException extends RuntimeException {
8      public ResourceNotFoundException(String message) {
9          super(message);
10     }
11 }
```

Контроллер:

```

ava > demo > src > main > java > com > example > demo > controller > TaskController.java
1  package com.example.demo.controller;
2
3  import com.example.demo.exception.ResourceNotFoundException;
4  import com.example.demo.model.Task;
5  import com.example.demo.repository.TaskRepository;
6  import org.springframework.beans.factory.annotation.Autowired;
7  import org.springframework.web.bind.annotation.*;
8
9  import java.time.LocalDateTime;
10 import java.util.List;
11
12 @RestController
13 @RequestMapping("/api/tasks")
14 public class TaskController {
15
16     @Autowired
17     private TaskRepository taskRepository;
18
19     @GetMapping
20     public List<Task> getAllTasks() {
21         return taskRepository.findAll();
22     }
23
24     @PostMapping
25     public Task createTask(@RequestBody Task task) {
26         task.setCreatedAt(LocalDateTime.now());
27         return taskRepository.save(task);
28     }
29
30     @PutMapping("/{id}")
31     public Task updateTask(@PathVariable Long id, @RequestBody Task updatedTask) {
32         return taskRepository.findById(id)
33             .map(task -> {
34                 task.setTitle(updatedTask.getTitle());
35                 task.setDescription(updatedTask.getDescription());
36                 task.setStatus(updatedTask.getStatus());
37                 return taskRepository.save(task);
38             })
39             .orElseThrow(() -> new ResourceNotFoundException("Задача с id " + id + " не найдена"));
40     }
41
42     @DeleteMapping("/{id}")
43     public void deleteTask(@PathVariable Long id) {
44         taskRepository.deleteById(id);
45     }
46 }

```

Добавление безопасности:

В pom.xml добавим зависимость:

```

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
</dependencies>

<build>

```

Теперь обновляем application.yml:

```

demo > src > main > resources > ! application.yml
spring:
  application:
    name: demo

  security:
    basic:
      enabled: true
    user:
      name: "admin"
      password: "admin123"
      roles: "ADMIN"

  datasource:
    url: "jdbc:h2:mem:taskdb"
    driver-class-name: "org.h2.Driver"
    username: "sa"
    password: ""

  jpa:
    database-platform: "org.hibernate.dialect.H2Dialect"
    show-sql: true
    hibernate:
      ddl-auto: update

  h2:
    console:
      enabled: true
      path: /h2-console
      settings:
        web-allow-others: false
        trace: false

```

Конфигурация безопасности:

```

Java > demo > src > main > java > com > example > demo > config > J SecurityConfig.java
1  package com.example.demo.config;
2
3  import org.springframework.context.annotation.Bean;
4  import org.springframework.context.annotation.Configuration;
5  import org.springframework.security.config.Customizer;
6  import org.springframework.security.config.annotation.web.builders.HttpSecurity;
7  import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
8  import org.springframework.security.web.SecurityFilterChain;
9
10 @Configuration
11 @EnableWebSecurity
12 public class SecurityConfig {
13
14     @Bean
15     public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
16         http
17             .csrf(csrf -> csrf.disable())
18             .headers(headers -> headers
19                 .frameOptions()
20                 .sameOrigin()
21             )
22             .authorizeHttpRequests(auth -> auth
23                 .requestMatchers("/h2-console/**").permitAll()
24                 .requestMatchers("/api/tasks/**").authenticated()
25                 .anyRequest().authenticated()
26             )
27             .httpBasic(Customizer.withDefaults());
28
29         return http.build();
30     }
31 }

```

Тестируем приложение:

POST: создание задачи:

The screenshot displays a REST client interface with the following components:

- Method and URL:** POST `http://localhost:8080/api/tasks`
- Authorization:** Basic Auth. Username: `admin`, Password: `admin123`. A warning message states: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)."
- Body:** JSON format. The request body is:

```
{
  "title": "Первая задача",
  "description": "Описание первой задачи",
  "status": "NEW"
}
```
- Response:** 200 OK, 127 ms, 583 B. The response body is shown in "Pretty" format:

```
{
  "id": 1,
  "title": "Первая задача",
  "description": "Описание первой задачи",
  "status": "NEW",
}
```

Проверка в H2:

← ↻ 🌐 localhost:8080 H2 Console

English ▼ Preferences Tools Help

Login

Saved Settings: Generic H2 (Embedded) ▼

Setting Name: Generic H2 (Embedded) Save Remove

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:mem:taskdb

User Name: sa

Password:

Connect Test Connection

🔍 📄 🔄 Auto commit 🔄 Max rows: 1000 🔄 🔄 Auto complete Off ▼ Auto select On ▼ ?

jdbc:h2:mem:taskdb

TASK

INFORMATION\_SCHEMA

Users

H2 2.2.224 (2023-09-17)

Run Run Selected Auto complete Clear SQL statement:

SELECT \* FROM TASK;

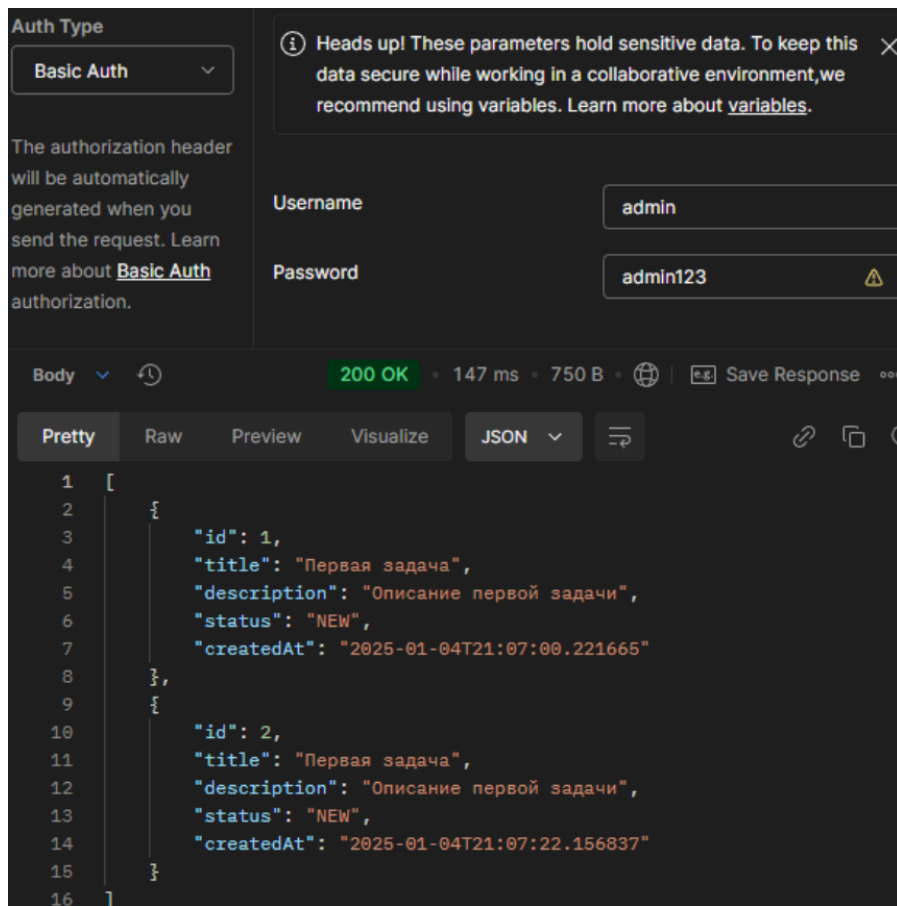
SELECT \* FROM TASK;

ID	CREATED_AT	DESCRIPTION	STATUS	TITLE
1	2025-01-05 21:07:00.221665	Описание первой задачи	NEW	Первая задача

(1 row, 0 ms)

Edit

GET: получения списка задач



POST: создание второй задачи



POST http://localhost:8080/api/tasks Send

Params Auth Headers (9) Body Scripts Settings Cookies

raw JSON Beautify

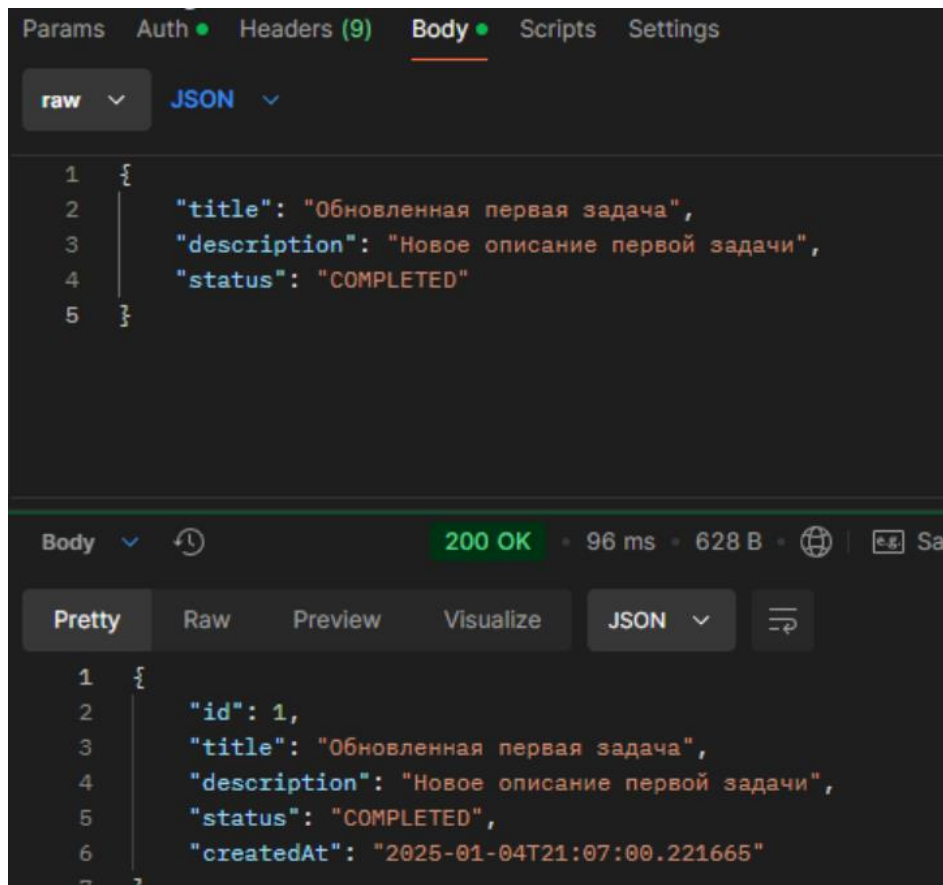
```
1 {  
2   "title": "Вторая задача",  
3   "description": "Описание второй задачи",  
4   "status": "IN_PROGRESS"  
5 }
```

Body 200 OK • 71 ms • 596 B • Save Response

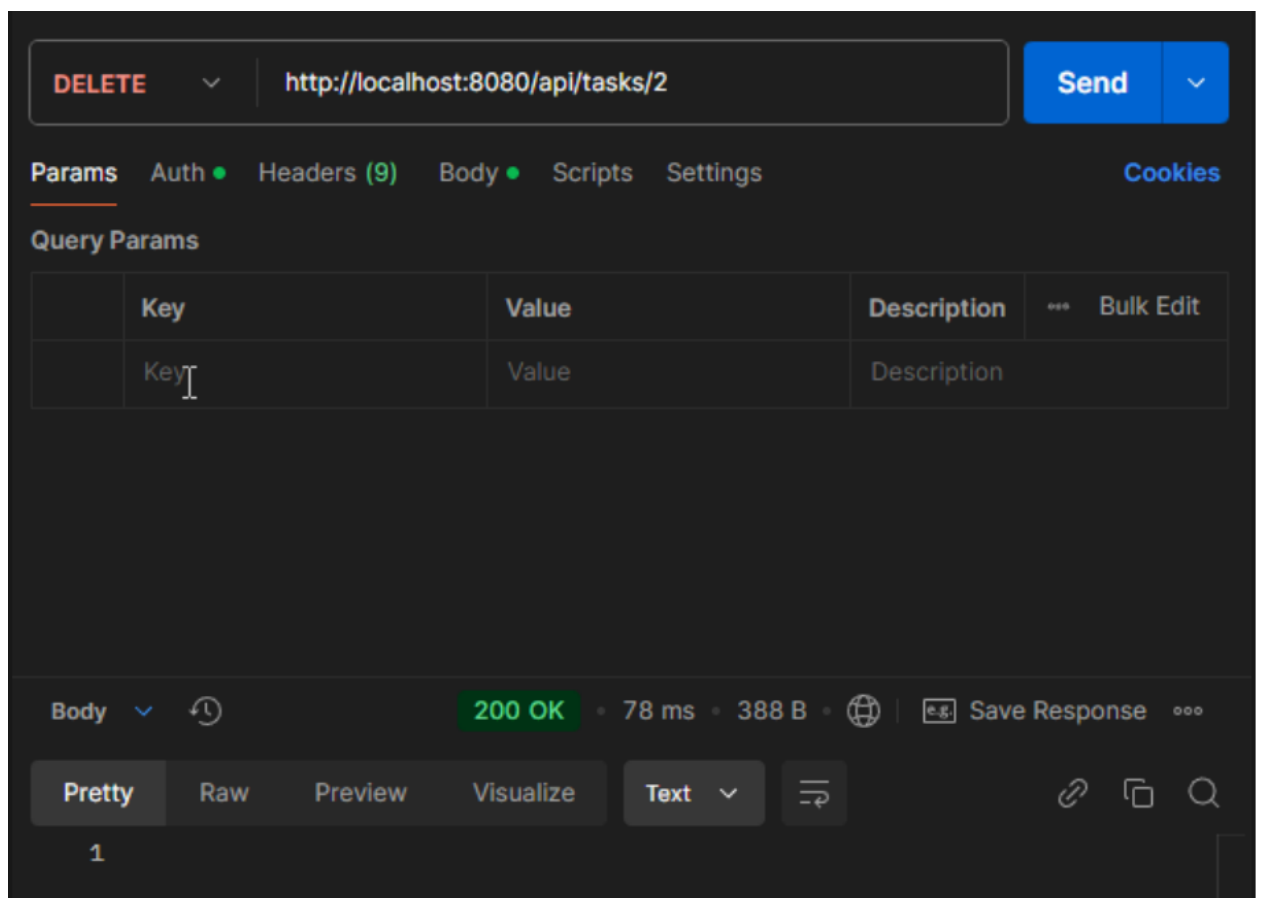
Pretty Raw Preview Visualize JSON

```
1 {  
2   "id": 3,  
3   "title": "Вторая задача",  
4   "description": "Описание второй задачи",  
5   "status": "IN_PROGRESS",  
6   "createdAt": "2025-01-04T21:07:52.557658"  
7 }
```

PUT: обновим задачу



DELETE: удаление задачи:



SELECT \* FROM TASK;

SELECT \* FROM TASK;

ID	CREATED_AT	DESCRIPTION	STATUS	TITLE
1	2025-01-04 21:07:00.221665	Новое описание первой задачи	COMPLETED	Обновленная первая задача
3	2025-01-04 21:07:52.557658	Описание второй задачи	IN_PROGRESS	Вторая задача

(2 rows, 0 ms)

Edit