



# EducaCiência FastCode

Fala Galera,

- Artigo: 45/2021 Data: Fevereiro/2021
- Público-alvo: Desenvolvedores – Iniciantes
- Tecnologia: Java
- Tema: Artigo 45 - SpringBoot JPA MySql Delete
- Link: <https://github.com/perucello/DevFP>

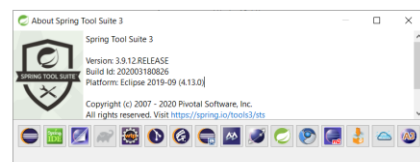
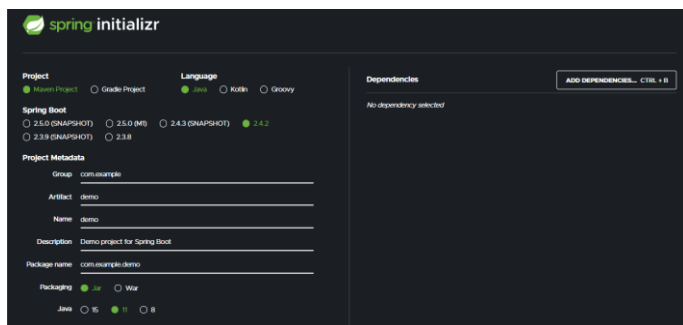
Neste artigo, abordaremos Spring Boot e iremos mapear o Método Delete com repositório JPA Repository e Banco de Dados MySql.

Traremos uma série de artigos onde exploraremos os métodos de maneira individual até chegarmos em uma API com todos os métodos.

Lembrando que os fins são didáticos !

Para este ambiente , utilizaremos do Banco de Dados MySql , deixaremos um script de criação do Banco anexado ao nosso artigo.

Criaremos nosso projeto utilizando do link - <https://start.spring.io/> e abriremos na IDE Spring Tool Suíte 3 “STS”.





Com nosso Projeto já aberto no STS , iremos preparar nosso arquivo “pom”, ou seja, nossas dependências que iremos trabalhar no projeto.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.2</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.project.jpa.mysql</groupId>
  <artifactId>SpringBoot-JPA-Mysql</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>SpringBoot-JPA-Mysql</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>1.8</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <scope>runtime</scope>
    </dependency>

    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
    </dependency>

    <dependency>
      <groupId>javax.validation</groupId>
      <artifactId>validation-api</artifactId>
      <version>1.1.0.Final</version>
    </dependency>

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

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

    <dependency>
      <groupId>org.springframework.boot</groupId>
```

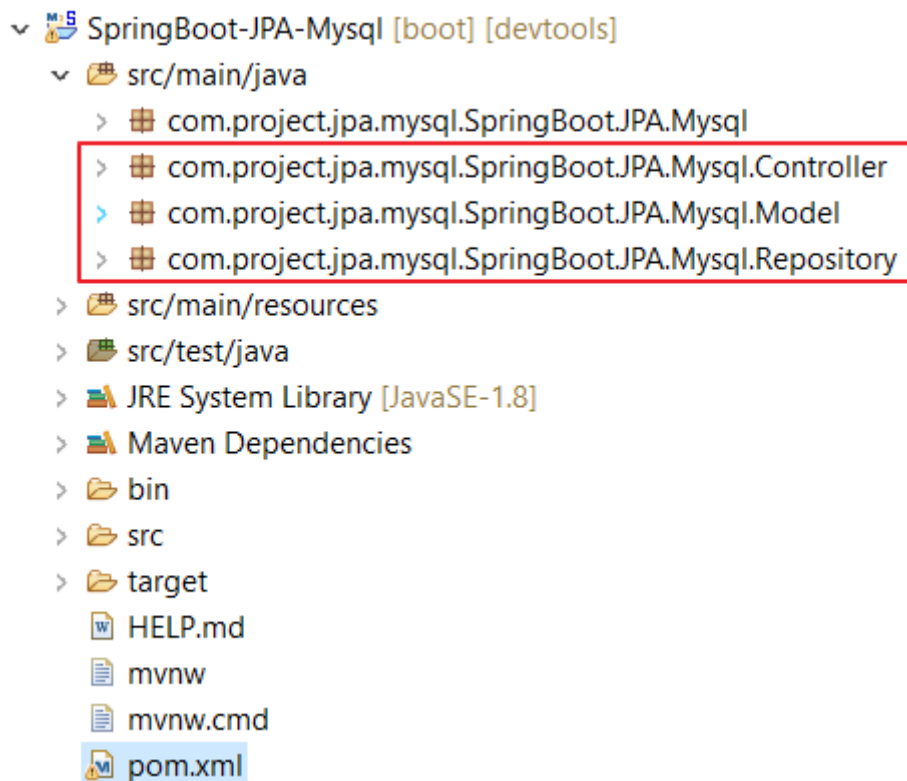


```
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>
```

Feito isso, vamos atualizar nosso Maven – para isso, basta executar “Maven Install”.

Vamos preparar agora nossos “pacotes” como demonstrados abaixo:



Feito isso, vamos criar nossas classes sendo:



Vamos detalhar:

- Controller** – neste pacote teremos nossa classe “ClientesController” onde será escrito nosso código que manipularemos os métodos CRUD e nosso endpoint;
- Model** – neste pacote teremos nossa classe “Cliente” onde criaremos a estrutura da nossa tabela Cliente no Banco de Dados
- Repository** – teremos nossa “interface” que se estenderá da classe “Cliente” e receberá nosso Repository JPA.

Primeiramente, vamos criar a estrutura da nossa tabela, como a seguir:

```
package com.project.jpa.mysql.SpringBoot.JPA.Mysql.Model;
```

```
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.validation.constraints.NotNull;
```

```
@Entity  
public class Cliente {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
  
    private String nome;  
  
    @NotNull  
    private String email;  
  
    public Cliente() {  
        super();  
    }  
  
    public Cliente(Long id, String nome, String email) {  
        super();  
        this.id = id;  
        this.nome = nome;  
    }  
}
```



```
        this.email = email;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((id == null) ? 0 : id.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Cliente other = (Cliente) obj;
        if (id == null) {
            if (other.id != null)
                return false;
        } else if (!id.equals(other.id))
            return false;
        return true;
    }
}
```

Agora, podemos estender nossa interface, para isso , abra o pacote Repositório e clique em Clientes.java e insira o seguinte código:

```
package com.project.jpa.mysql.SpringBoot.JPA.MySql.Repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.project.jpa.mysql.SpringBoot.JPA.MySql.Model.Cliente;

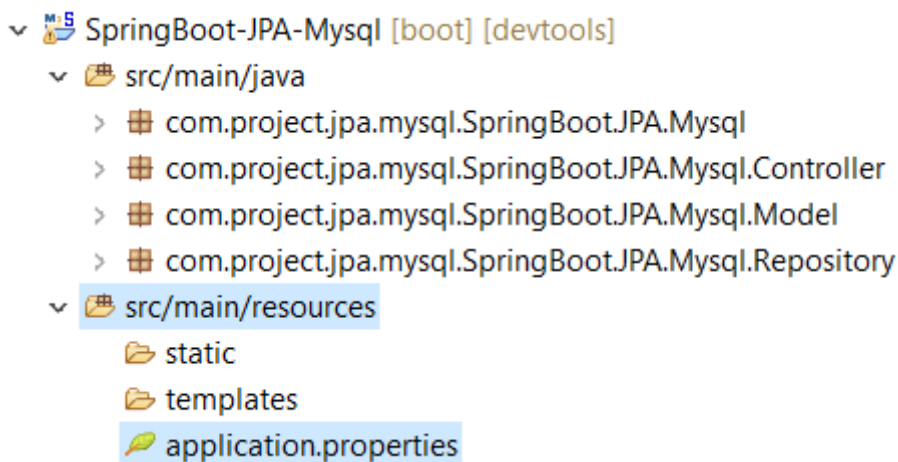
public interface Clientes extends JpaRepository<Cliente, Long> {
}
```



Feito isso, temos a estrutura do nosso banco de dados ( tabela ) pronta e agora , agora precisamos criar nosso Banco de Dados no Mysql, para isso, abra seu Workbench e insira os seguintes códigos e execute.

```
1 • create database bdJpaMysql;
2 • use bdJpaMysql;
```

Com o Banco de Dados criado, vamos preparar a conexão do nosso Projeto, sendo assim, acesse application.properties pela rota src/main/resources e insira os seguintes códigos:



```
spring.datasource.url=jdbc:mysql://localhost:3306/bdJpaMysql?useTimezone=true&serverTimezone=UTC
&useSSL=false
spring.datasource.username=root
spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
```

Com a nossa estrutura do Banco de Dados criado, ( entidade e repositório ) agora, podemos focar em nosso CRUD.

Para este trabalho, abra o Script **ClientesController** que está no pacote Controller e insira os seguintes códigos:

Com isso , temos nosso Post do CRUD pronto onde os métodos que manipularemos será o seguinte:

```
package com.project.jpa.mysql.SpringBoot.JPA.Mysql.Controller;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
```



```
import com.project.jpa.mysql.SpringBoot.JPA.MySql.Model.Cliente;
import com.project.jpa.mysql.SpringBoot.JPA.MySql.Repository.Clientes;
```

```
@RestController
@RequestMapping("/api/JPA/MySql/clientes")
public class ClientesController {

    @Autowired
    private Clientes clientes;

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deletar(@PathVariable Long id){
        Optional<Cliente> cliente = clientes.findById(id);
        if(cliente != null) {
            clientes.deleteById(id);
        }
        return ResponseEntity.noContent().build();
    }
}
```

⇒ **Delete** – este método iremos deletar um registro passando o ID como parâmetro.

Agora, basta iniciarmos nossa “Aplicação”

⇒ Run As \ Spring Boot App

```
2021-02-21 13:18:55.361 INFO 18300 --- [ restartedMain] .j.m.s.j.m.SpringBootJpaMySqlApplication : Starting SpringBootJpaMySqlApplication using Java 1.8.0_212-3-redhat on DESKTOP-F3A10P2 with PID 18300
2021-02-21 13:18:55.367 INFO 18300 --- [ restartedMain] .j.m.s.j.m.SpringBootJpaMySqlApplication : No active profile set, falling back to default profiles: default
2021-02-21 13:18:55.510 INFO 18300 --- [ restartedMain] e.DevToolsPropertyDefaultsPostProcessor : DevTools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable
2021-02-21 13:18:55.510 INFO 18300 --- [ restartedMain] e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' property to 'DEBUG'
2021-02-21 13:18:56.768 INFO 18300 --- [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2021-02-21 13:18:56.886 INFO 18300 --- [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 102 ms. Found 1 JPA repository interfaces.
2021-02-21 13:18:57.909 INFO 18300 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-02-21 13:18:57.923 INFO 18300 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-02-21 13:18:57.923 INFO 18300 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.41]
2021-02-21 13:18:58.143 INFO 18300 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2021-02-21 13:18:58.144 INFO 18300 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2632 ms
2021-02-21 13:18:58.523 INFO 18300 --- [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2021-02-21 13:18:58.626 INFO 18300 --- [ restartedMain] org.hibernate.Version : HHH0000412: Hibernate ORM core version 5.4.27.Final
2021-02-21 13:18:58.886 INFO 18300 --- [ restartedMain] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations (5.1.2.Final)
2021-02-21 13:18:59.100 INFO 18300 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2021-02-21 13:18:59.502 INFO 18300 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2021-02-21 13:18:59.525 INFO 18300 --- [ restartedMain] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL5InnoDBDialect
2021-02-21 13:19:00.612 INFO 18300 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.J
2021-02-21 13:19:00.628 INFO 18300 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2021-02-21 13:19:00.694 INFO 18300 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2021-02-21 13:19:01.293 INFO 18300 --- [ restartedMain] jpaBaseConfigurationJpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during vie
2021-02-21 13:19:01.517 INFO 18300 --- [ restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2021-02-21 13:19:01.926 INFO 18300 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-02-21 13:19:01.943 INFO 18300 --- [ restartedMain] .j.m.s.j.m.SpringBootJpaMySqlApplication : Started SpringBootJpaMySqlApplication in 7.346 seconds (JVM running for 9.669)
```

Nossa aplicação se iniciou como esperado.

Agora podemos testar , manipulando via Postman, para isso abra seu aplicativo e vamos testar o método que criamos em nossa API.

⇒ Lembrando que no artigo 42/2021 nós realizamos já o Post e no artigo 44/2021 trouxemos novas inserções de registros para que pudéssemos manipular, e iremos acessar nosso banco de Dados para validar os registros que temos, para isso, execute o código:







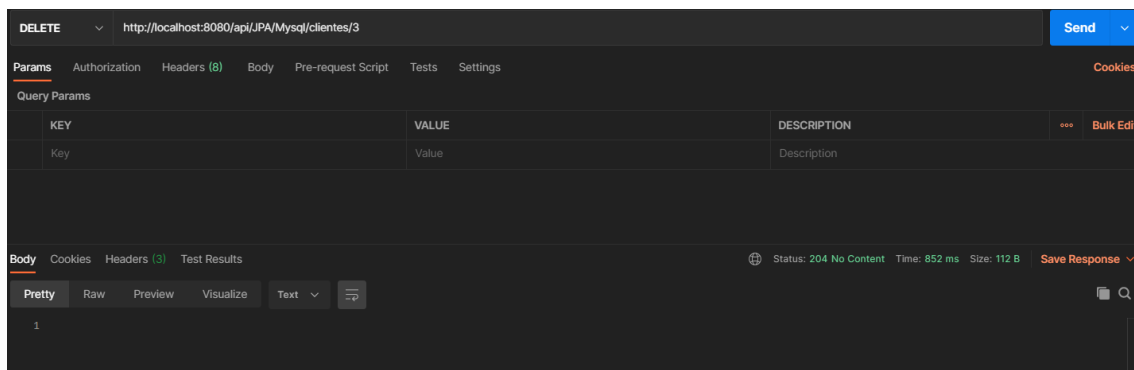
```
4 • select * from cliente;
5
6 • INSERT INTO `bdjpamysql`.`cliente` (`email`, `nome`) VALUES ('fulano@fulano.com.br', 'Fulano de Tal');
7 • INSERT INTO `bdjpamysql`.`cliente` (`email`, `nome`) VALUES ('beltrano@beltrano.com.br', 'Beltrano da Silva');
8
```

	id	nome	email
▶	1	EducaCiencia	educaciencia-fastcode@outlook.com.br
	2	Fulano de Tal	fulano@fulano.com.br
	3	Beltrano da Silva	beltrano@beltrano.com.br
*	NULL	NULL	NULL

Nota-se que temos três registros, e iremos com nossa API excluir o registro numero 3.

Vamos inserir o Endpoint de acesso e manipulá-lo como abaixo:

⇒ **Retorno esperado quando se obtém sucesso é 204 neste caso.**



O retorno foi exatamente o esperado, vamos validar no nosso Banco de Dados, execute novamente o comando Select no seu workbench

	id	email	nome
▶	1	educaciencia-fastcode@outlook.com.br	EducaCiencia
	2	fulano@fulano.com.br	Fulano de Tal
*	NULL	NULL	NULL

Nossa API funcionou como nossa proposta, sendo assim finalizamos este artigo, onde os códigos estão disponíveis no GitHub para consumo.

Até mais !  
Espero ter ajudado !

