Guia Prático:

Criando uma Aplicação Java com Spring Boot, JPA e MVC

Neste guia, vamos construir uma aplicação Java simples utilizando:

- **Spring Boot** para configuração e execução rápida
- **Spring Data JPA** para persistência de dados
- **Banco de dados H2** (em memória)
- **Padrão MVC** para separação de responsabilidades
- **REST Controller** para endpoints web

Etapa 0: Gerando o Projeto com Spring Initializr

Acesse o site: https://start.spring.io

Passos:

1. **Project:** Maven
2. **Language:** Java
3. **Spring Boot:** 3.1.0 ou superior
4. **Project Metadata:**
    - Group: com.example
    - Artifact: demo
    - Name: demo
    - Package name: com.example.demo
5. **Packaging:** Jar
6. **Java:** 17 ou superior
7. **Dependencies:**
    - Spring Web
    - Spring Data JPA
    - H2 Database

Clique em **"Generate"** para baixar o projeto ZIP. Extraia o arquivo em seu ambiente local e abra no IDE de sua preferência (IntelliJ, Eclipse, VS Code, NetBeans).

Estrutura do Projeto

```
com.example.demo
├── Controller
│   └── CursoController.java
├── Entidade
│   └── Customer.java
├── Model
│   └── CursoModel.java
├── Principal
│   ├── AccessingDataJpaApplication.java
│   └── MVCPatternDemo.java
├── Repositorio
│   └── CustomerRepository.java
├── View
│   └── CursoView.java
├── HelloController.java
└── DemoApplication.java
```

Etapa 1: Entidade e Repositório JPA

Customer.java

```
@Entity
public class Customer {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String firstName;
    private String lastName;
    // Construtores, getters, setters e toString()
}
```

CustomerRepository.java

```
public interface CustomerRepository extends CrudRepository<Customer, Integer> {
    List<Customer> findByLastName(String lastName);
}
```

Etapa 2: Inicialização com Spring Boot

AccessingDataJpaApplication.java

```
@SpringBootApplication
public class AccessingDataJpaApplication {
    public static void main(String[] args) {
        SpringApplication.run(AccessingDataJpaApplication.class, args);
    }

    @Bean
    public CommandLineRunner demo(CustomerRepository repo) {
        return args -> {
            repo.save(new Customer("João", "Silva"));
            repo.findAll().forEach(System.out::println);
        };
    }
```

```
}
```

Output Esperado

```
Customer [id=1, firstName=João, lastName=Silva]
```

Etapa 3: Criando um REST Controller

HelloController.java

```
@RestController
public class HelloController {
    @GetMapping("/hello")
    public String hello(@RequestParam(defaultValue = "World") String name) {
        return "Hello, " + name;
    }
}
```

Teste no Navegador

```
http://localhost:8080/hello?name=Fábio
```

Resposta:

```
Hello, Fábio
```

Etapa 4: Aplicando o Padrão MVC

Model: CursoModel.java

```
public class CursoModel {
    private String nome;
    private int duracao;
    // Construtores, getters e setters
}
```

View: CursoView.java

```
public class CursoView {
    public void imprimirDetalhes(String nome, int duracao) {
        System.out.println("Curso: " + nome + ", Duração: " + duracao + " horas");
    }
}
```

Controller: CursoController.java

```
public class CursoController {
    private CursoModel model;
    private CursoView view;

    public CursoController(CursoModel model, CursoView view) {
        this.model = model;
        this.view = view;
    }
```

```
    public void atualizarView() {
        view.imprimirDetalhes(model.getNome(), model.getDuracao());
    }
}
```

Execução: MVCPatternDemo.java

```
public class MVCPatternDemo {
    public static void main(String[] args) {
        CursoModel model = new CursoModel("Java Web", 40);
        CursoView view = new CursoView();
        CursoController controller = new CursoController(model, view);

        controller.atualizarView();
    }
}
```

Output Esperado

Curso: Java Web, Duração: 40 horas

Etapa 5: Configuração com Maven

pom.xml (Trecho)

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

Com esse projeto, você aprendeu como:

- *Construir um back-end com Spring Boot*
- *Aplicar o padrão MVC com classes Java*
- *Persistir dados com JPA e H2*
- *Expor endpoints com REST*

EducaCiência FastCode para a comunidade