

# Java Development for API

---

Mastering the Art of  
API Creation with Java



# Por Que Usar Spring no Desenvolvimento de APIs

O Spring Framework, especialmente com o módulo Spring Boot, simplifica drasticamente o desenvolvimento de APIs REST em Java. Abaixo, listamos suas principais vantagens com exemplos práticos.



# 01

Início Rápido com  
Spring Boot

# Início Rápido com Spring Boot

O Spring Boot permite criar APIs com mínima configuração. Ele elimina a necessidade de configurar servidores e arquivos XML complexos.

 Exemplo de aplicação REST básica:

```
Spring Framework

1  @RestController
2  public class HelloController {
3
4      @GetMapping("/hello")
5      public String hello() {
6          return "Olá, API!";
7      }
8  }
9
```

Com apenas essas anotações, a aplicação já responde a requisições HTTP.

02

# Injeção de Dependência Simplificada

# Injeção de Dependência Simplificada

Com Spring, a injeção de dependência é nativa, promovendo código limpo e testável.

 Exemplo com @Autowired:

```
Spring Framework

1  @Service
2  public class ProdutoService {
3      public List<String> listar() {
4          return List.of("Teclado", "Mouse");
5      }
6  }
7
8  @RestController
9  public class ProdutoController {
10
11      @Autowired
12      private ProdutoService produtoService;
13
14      @GetMapping("/produtos")
15      public List<String> listarProdutos() {
16          return produtoService.listar();
17      }
18  }
19
```

Com Spring, a injeção de dependência é nativa, promovendo código limpo e testável.

# 03

## Criação Facilitada de Endpoints REST

# Criação Facilitada de Endpoints REST

Spring simplifica o mapeamento de URLs para métodos Java, usando anotações como `@GetMapping`, `@PostMapping`, etc.



## Exemplo POST:

```
Spring Framework

1  @PostMapping("/produtos")
2  public String salvarProduto(@RequestBody String nome) {
3      return "Produto salvo: " + nome;
4  }
5
```



# 04

## Integração com Banco de Dados (Spring Data JPA)

# Integração com Banco de Dados (Spring Data JPA)

Você cria repositórios com poucas linhas de código e sem SQL explícito.

 Exemplo com JPA:

```
Spring Framework
1 @Entity
2 public class Produto {
3     @Id @GeneratedValue
4     private Long id;
5     private String nome;
6 }
7
8 public interface ProdutoRepository extends JpaRepository<Produto, Long> {}
9
```

Você cria repositórios com poucas linhas de código e sem SQL explícito.

# 05

Tratamento de Erros  
com `@ExceptionHandler`

# Tratamento de Erros com @ExceptionHandler

Você pode personalizar respostas de erro de forma elegante.

 Exemplo de tratamento de erro:

```
Spring Framework
1 @RestControllerAdvice
2 public class ErrorHandler {
3
4     @ExceptionHandler(Exception.class)
5     public ResponseEntity<String> handleErro(Exception e) {
6         return ResponseEntity.status(500).body("Erro: " + e.getMessage());
7     }
8 }
9
```

O Spring Framework economiza tempo, reduz código repetitivo e oferece uma arquitetura sólida para APIs REST. Se você busca produtividade com robustez, Spring é o caminho certo.