

Disciplina Regular 3

# Desenvolvimento Web com Java EE

Graduação em Engenharia de Software - 2020

# Etapa 7 Aula 1

JDBCTemplate

# JDBCTemplate

# JDBCTemplate

- Simplifica o uso do JDBC e ajuda a evitar erros comuns.
- Executa o fluxo de trabalho JDBC principal, deixando o código do aplicativo para fornecer SQL e extrair resultados.
- Essa classe executa consultas ou atualizações SQL, iniciando a iteração sobre ResultSets e capturando exceções JDBC e convertendo-as na hierarquia de exceções genérica.
- O código que usa essa classe precisa implementar apenas interfaces de retorno de chamada, dando a eles um contrato claramente definido.

# JdbcTemplate

- Pode ser usado em uma implementação de serviço via instanciação direta com uma referência DataSource ou ser preparado em um contexto de aplicativo e fornecido aos serviços como referência de bean.
- Todas as operações SQL executadas por essa classe são registradas no nível de depuração, usando "org.springframework.jdbc.core.JdbcTemplate" como categoria de log.

```
applicationContext.xml x
Source History
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xmlns:p="http://www.springframework.org/schema/p"
6       xmlns:aop="http://www.springframework.org/schema/aop"
7       xmlns:tx="http://www.springframework.org/schema/tx"
8       xsi:schemaLocation="http://www.springframework.org/schema/beans
9                           http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
10                          http://www.springframework.org/schema/context
11                          http://www.springframework.org/schema/context/spring-context.xsd
12                          http://www.springframework.org/schema/aop
13                          http://www.springframework.org/schema/aop/spring-aop-3.1.xsd
14                          http://www.springframework.org/schema/tx
15                          http://www.springframework.org/schema/tx/spring-tx-3.1.xsd">
16
17     <!-- ===== -->
18     <!-- Configurações Gerais -->
19     <!-- ===== -->
20     <context:annotation-config/>
21     <context:component-scan base-package="br.edu.infnet" />
22
23     <!-- ===== -->
24     <!-- Configurações do JDBC DataSource -->
25     <!-- ===== -->
26
27     <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManage
28
29         <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
30         <property name="url" value="jdbc:mysql://localhost:3306/sakila" />
31         <property name="username" value="root" />
32         <property name="password" value="root" />
```

```
1 package br.edu.infnet.contatos;
2
3 import javax.validation.constraints.NotEmpty;
4 import javax.validation.constraints.Size;
5
6 public class Contato {
7
8     private Integer id;
9     @NotEmpty(message = "O campo Nome é obrigatório")
10    @Size(min = 1, max = 100, message = "O campo Nome pode ter no máximo 100 car
11    private String nome;
12    @NotEmpty(message = "O campo Email é obrigatório")
13    @Size(min = 1, max = 150, message = "O campo Email pode ter no máximo 50 car
14    private String email;
15    @NotEmpty(message = "O campo Telefone é obrigatório")
16    @Size(min = 1, max = 150, message = "O campo Telefone pode ter no máximo 50
17    private String fone;
18
19    public Integer getId() {
20        return id;
21    }
```

- Source Packages
  - br.edu.infnet.contatos
    - Contato.java
    - ContatoController.java
    - ContatoRepository.java
  - br.edu.infnet.countries
- Test Packages
- Other Sources
  - src/main/resources
    - META-INF
      - br.edu.infnet.contatos
        - ContatoRepository.xml
      - br.edu.infnet.countries
        - CountryRepository.xml

ContatoRepository.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
3
4 <properties>
5   <entry key="ContatoRepository.listar">
6     <![CDATA[
7       SELECT
8         id,
9         nome,
10        email,
11        fone
12      FROM contatos
13    ]]>
14  </entry>
15  <entry key="ContatoRepository.obter">
16    <![CDATA[
17      SELECT
18        id,
19        nome,
20        email,
21        fone
22      FROM contatos
23      WHERE id = ?
24    ]]>
25  </entry>
```



ContatoRepository.java x

Source

History



```
1 package br.edu.infnet.contatos;
2
3 import ...10 lines
13
14 @Repository
15 @PropertySource("classpath:/br/edu/infnet/contatos/ContatoRepository.xml")
16 public class ContatoRepository extends JdbcDaoSupport {
17
18     @Autowired
19     private Environment queries;
20
21     @Qualifier("dataSource")
22     @Autowired
23     private DataSource dataSource;
24
25     @PostConstruct
26     private void initialize() {
27
28         this.setDataSource(this.dataSource);
29     }
```

```
31 public List<Contato> listar() {  
32  
33     String sql = queries.getProperty("ContatoRepository.listar");  
34     BeanPropertyRowMapper rowMapper = new BeanPropertyRowMapper(Contato.class);  
35     return this.getJdbcTemplate().query(sql, rowMapper);  
36 }  
37  
38 public Contato obter(Integer id) {  
39  
40     String sql = queries.getProperty("ContatoRepository.obter");  
41     BeanPropertyRowMapper rowMapper = new BeanPropertyRowMapper(Contato.class);  
42     Object[] params = new Object[]{  
43         id  
44     };  
45     return (Contato) this.getJdbcTemplate().queryForObject(sql, params, rowMapper);  
46 }
```

```
48 public void inserir(Contato contato) {  
49  
50     String sql = queries.getProperty("ContatoRepository.inserir");  
51     Object[] params = new Object[]{  
52         contato.getNome(),  
53         contato.getEmail(),  
54         contato.getFone()  
55     };  
56     this.getJdbcTemplate().update(sql, params);  
57 }  
58  
59 public void alterar(Contato contato) {  
60  
61     String sql = queries.getProperty("ContatoRepository.alterar");  
62     Object[] params = new Object[]{  
63         contato.getNome(),  
64         contato.getEmail(),  
65         contato.getFone(),  
66         contato.getId()  
67     };  
68     this.getJdbcTemplate().update(sql, params);  
69 }
```

```
17 @RequestMapping("/contatos/listar")
18 public ModelAndView listar() {
19
20     ModelAndView retorno = new ModelAndView("contatos/index");
21     List<Contato> lista = contatoRepository.listar();
22     if (lista != null && !lista.isEmpty()) {
23
24         retorno.addObject("contatos", lista);
25     } else {
26
27         retorno.addObject("mensagem", "Não existem dados para exibir");
28     }
29     return retorno;
30 }
```

```
32 @RequestMapping("/contatos/obter")
33 public ModelAndView obter(Integer id) {
34
35     ModelAndView retorno = new ModelAndView("contatos/index");
36     retorno.addObject("contato", contatoRepository.obter(id));
37     return retorno;
38 }
```



```
40 @RequestMapping("/contatos/salvar")
41 public ModelAndView salvar(@Valid Contato contato, BindingResult br) {
42
43     ModelAndView retorno = new ModelAndView("contatos/index");
44     if (!br.hasErrors()) {
45         if (contato != null && contato.getId() == null) {
46
47             contatoRepository.inserir(contato);
48         } else {
49
50             contatoRepository.alterar(contato);
51         }
52         retorno.addObject("contato", null);
53         retorno.addObject("contatos", contatoRepository.listar());
54     } else {
55
56         retorno.addObject("erros", br.getFieldErrors());
57     }
58     return retorno;
59 }
```

# Exercício

- Criar o cadastro de contatos visto na demonstração.

The screenshot shows a web browser window with the title 'Sakila' and the URL 'localhost:8080/Dr3E7A1\_JdbcTemplate/contatos/index.jsp'. The page is titled 'Sakila - Contatos' and contains a form for adding a new contact. The form has four input fields: 'ID', 'Nome', 'Email', and 'Telefone', followed by a 'Salvar' button. Below the form is a table displaying a list of contacts. The table has columns for 'ID', 'Nome', 'Email', and 'Telefone'. There are three rows of data. At the bottom of the page, there is a pagination control showing 'Mostrando de 1 até 3 de 3 registros' and buttons for 'Anterior', '1', and 'Próximo'.

**Sakila - Contatos**

ID   
Nome   
Email   
Telefone

10 resultados por página Pesquisar

ID	Nome	Email	Telefone
1	Machado de Assis	machado@abl.org.br	21228811
3	Rachel de Queiroz	rachel@abl.org.br	33224411
4	Rubem Fonseca	rubem.fonseca@gmail.com	123456

Mostrando de 1 até 3 de 3 registros Anterior 1 Próximo