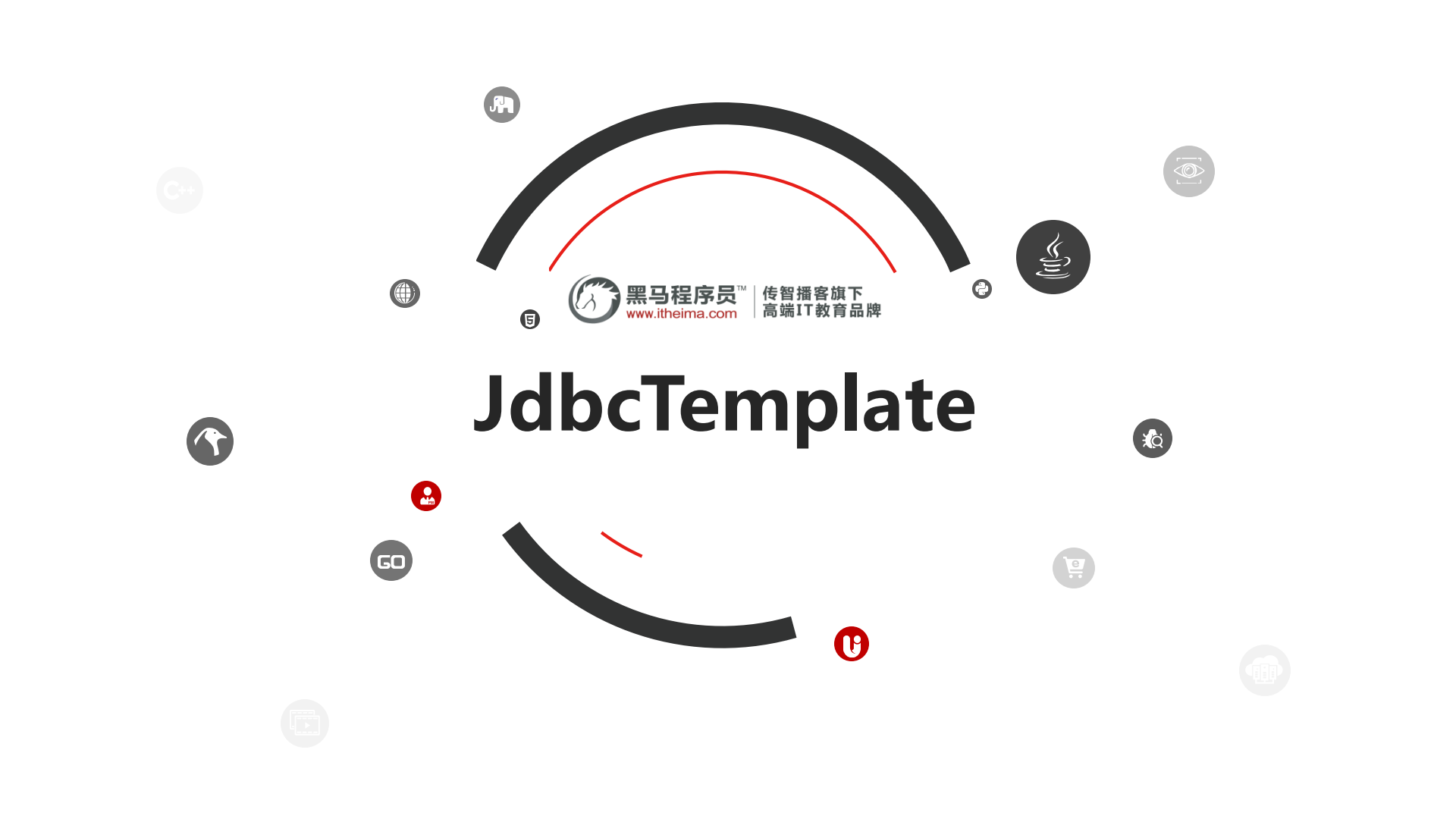
黑马程序员™
www.itheima.com
传智播客旗下
高端IT教育品牌
JdbcTemplate

目录 Contents

1. Spring JdbcTemplate基本使用

1.1 JdbcTemplate概述

它是spring框架中提供的一个对象，是对原始繁琐的Jdbc API对象的简单封装。spring框架为我们提供了很多的操作模板类。例如：操作关系型数据的JdbcTemplate和HibernateTemplate，操作nosql数据库的RedisTemplate，操作消息队列的JmsTemplate等等。

1.2 JdbcTemplate开发步骤

① 导入spring-jdbc和spring-tx坐标
② 创建数据库表和实体
③ 创建JdbcTemplate对象
④ 执行数据库操作

1. Spring JdbcTemplate基本使用

1.3 JdbcTemplate快速入门

① 导入坐标

```
<!--导入spring的jdbc坐标-->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jdbc</artifactId>
  <version>5.0.5.RELEASE</version>
</dependency>
<!--导入spring的tx坐标-->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-tx</artifactId>
  <version>5.0.5.RELEASE</version>
</dependency>
```

1. Spring JdbcTemplate基本使用

1.3 JdbcTemplate快速入门

② 创建accout表和Accout实体

栏位 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

名	类型	长度	小数点	允许空值 (	
name	varchar	50	0	☐	1
money	double	0	0	☑	

```
public class Account {
    private String name;
    private double money;
    //省略get和set方法
}
```

1. Spring JdbcTemplate基本使用

1.3 JdbcTemplate快速入门

③ 创建JdbcTemplate对象

④ 执行数据库操作

```
//1、创建数据源对象
ComboPooledDataSource dataSource = new ComboPooledDataSource();
dataSource.setDriverClass("com.mysql.jdbc.Driver");
dataSource.setJdbcUrl("jdbc:mysql://localhost:3306/test");
dataSource.setUser("root");
dataSource.setPassword("root");
//2、创建JdbcTemplate对象
JdbcTemplate jdbcTemplate = new JdbcTemplate();
//3、设置数据源给JdbcTemplate
jdbcTemplate.setDataSource(dataSource);
//4、执行操作
jdbcTemplate.update("insert into account values(?,?)","tom",5000);
```

1. Spring JdbcTemplate基本使用

1.4 Spring产生JdbcTemplate对象

我们可以将JdbcTemplate的创建权交给Spring，将数据源DataSource的创建权也交给Spring，在Spring容器内部将数据源DataSource注入到JdbcTemplate模版对象中，配置如下：

```
<!--数据源DataSource-->
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
    <property name="driverClass" value="com.mysql.jdbc.Driver"></property>
    <property name="jdbcUrl" value="jdbc:mysql:///test"></property>
    <property name="user" value="root"></property>
    <property name="password" value="root"></property>
</bean>
<!--JdbcTemplate-->
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="dataSource"></property>
</bean>
```

1. Spring JdbcTemplate基本使用

1.4 Spring产生JdbcTemplate对象

从容器中获得JdbcTemplate进行添加操作

```
@Test
public void testSpringJdbcTemplate() throws PropertyVetoException {
    ApplicationContext applicationContext = new
ClassPathXmlApplicationContext("applicationContext.xml");
    JdbcTemplate jdbcTemplate = applicationContext.getBean(JdbcTemplate.class);
    jdbcTemplate.update("insert into account values(?,?)","lucy",5000);
}
```

1. Spring JdbcTemplate基本使用

1.5 JdbcTemplate的常用操作

修改操作

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("classpath:applicationContext.xml")
public class JdbcTemplateCRUDTest {
    @Autowired
    private JdbcTemplate jdbcTemplate;
    @Test
    //测试修改操作
    public void testUpdate(){
        jdbcTemplate.update("update account set money=? where
name=?",1000,"tom");
    }
}
```

1. Spring JdbcTemplate基本使用

1.5 JdbcTemplate的常用操作

删除和查询全部操作

```
@Test
public void testDelete(){
    jdbcTemplate.update("delete from account where name=?","tom");
}
@Test
public void testQueryAll(){
    List<Account> accounts = jdbcTemplate.query("select * from account", new
BeanPropertyRowMapper<Account>(Account.class));
    for (Account account : accounts) {
        System.out.println(account.getName());
    }
}
```

1. Spring JdbcTemplate基本使用

1.5 JdbcTemplate的常用操作

查询单个数据操作操作

```
@Test
//测试查询单个对象操作
public void testQueryOne(){
    Account account = jdbcTemplate.queryForObject("select * from account where
name=?", new BeanPropertyRowMapper<Account>(Account.class), "tom");
    System.out.println(account.getName());
}
@Test
//测试查询单个简单数据操作(聚合查询)
public void testQueryCount(){
    Long aLong = jdbcTemplate.queryForObject("select count(*) from account",
Long.class);
    System.out.println(aLong);
}
```

1. Spring JdbcTemplate基本使用

1.6 知识要点

① 导入spring-jdbc和spring-tx坐标

② 创建数据库表和实体

③ 创建JdbcTemplate对象

```
JdbcTemplate jdbcTemplate = new JdbcTemplate();
jdbcTemplate.setDataSource(dataSource);
```

④ 执行数据库操作

更新操作:

```
jdbcTemplate.update  (sql,params)
```

查询操作:

```
jdbcTemplate.query  (sql,Mapper,params)
jdbcTemplate.queryForObject(sql,Mapper,params)
```

传智播客旗下高端IT教育品牌