

## 16 | 再回首：JdbcTemplate章节小结

2023-04-17 郭屹 来自北京

《手把手带你写一个MiniSpring》



你好，我是郭屹。

恭喜你学完了 MiniSpring 的第三部分——JdbcTemplate 了。JdbcTemplate 在 Spring 框架里，扮演着非常重要的角色。通过它，我们可以更加便捷地进行数据库操作，缩短了开发周期和开发成本，同时也降低了出错的风险。

它对 Spring 应用程序的稳定性和性能表现有着至关重要的影响，已经成为开发高效、高质量应用程序的不可或缺的一部分。

为了让你更好地掌握这部分内容，下面我们对这一整章做一个重点回顾。

### JdbcTemplate 重点回顾

JdbcTemplate 是 Spring 框架中的一部分，是 Spring 对数据访问的一个实现，在 Spring 应用程序中被广泛采用。它这个实现特别好地体现了 Rod Johnson 对简洁实用的原则的把握。JdbcTemplate 封装了 JDBC 的 API，并提供了更为便捷的访问方式，使得开发人员在不需要编写大量代码的情况下，能够高效、灵活地进行数据库操作。

我们知道，JDBC 的程序都是类似的，所以这个部分我们提取出一个 JDBC 访问的模板，同时引入 DataSource 概念，屏蔽具体的数据库，就便利了上层应用业务程序员。然后，我们再进行 SQL 参数的处理，SQL 请求带有参数，实现把数据转换成 SQL 语句所需要的参数格式，对 SQL 语句执行后的返回结果，又要自动绑定为业务对象。

之后，为了支持大量的数据访问，我们实现了数据库连接池提高性能，并且把连接池构造变成一个 Bean 注入到 IoC 容器里，还可以让用户自行配置连接池的参数。最后，进一步把程序里的 SQL 语句也抽取出来，配置到外部文件中，实现一个简单的 MyBatis。

这就是这一章实现 JdbcTemplate 的过程，你可以再回顾一下。另外我们每一节课后面都给了一道思考题，让你在我们实现的这个极简框架上进行扩展，如果你认真学习了这一章的内容，相信你是可以举一反三的，自己提出解决方案。

方法可能不同，但目标是一样的。我把参考答案写在文稿中了，你可以看一下，如果你有更好的思路 and 想法，也欢迎和我分享。下节课我们马上要进入 AOP 的环节了，一起期待一下吧！

## 13 | JDBC 访问框架：如何抽取 JDBC 模板并隔离数据库？

### 思考题

我们现在只实现了 query，想一想如果想要实现 update 应该如何做呢？

### 参考答案

我们现在 JdbcTemplate 类的结构，对于 query() 和 update() 是并列设计的，只要在类中对应的提供一个方法，形如：int update(String sql, Object[] args, int[] argTypes)。这个方法内部是一个 PreparedStatement，SQL 是要执行的 SQL 语句，args 是 SQL 参数，argTypes 是数据类型，返回值是受影响的行数。

## 14 | 增强模板：如何抽取专门的部件完成专门的任务？

### 思考题

你想一想我们应该怎么改造数据库连接池，保证多线程安全？

### 参考答案


这个问题有不同的方案，下面是一种思路供参考。

提供两个队列，一个用于忙的连接，一个用于空闲连接：

 复制代码

```
1     private BlockingQueue<PooledConnection> busy;
2     private BlockingQueue<PooledConnection> idle;
```


获取数据库连接就从 idle 队列中获取，程序大体如下：

 复制代码

```
1 while (true) {
2     conn = idle.poll();
3 }
```

就是死等一个空闲连接。然后加入忙队列。


当然，进一步考虑，还应当判断连接数是否到了最大，如果没有，则要先创建一个新的连接。创建的时候要小心了，因为是多线程的，所以要再次校验是否超过最大连接数，如使用 CAS 技术：

 复制代码

```
1 if (size.get() < getPoolProperties().getMaxActive()) {
2     if (size.addAndGet(1) > getPoolProperties().getMaxActive()) {
3         size.decrementAndGet();
4     } else {
5         return createConnection(now, con, username, password);
6     }
7 }
```

```
6         }
7     }
```

而且还应当设置一个 timeout，如果在规定的时间内还没有拿到一个连接，就要抛出一个异常。

 复制代码

```
1  if ((System.currentTimeMillis() - now) >= maxWait) {
2      throw new PoolExhaustedException(
3          "Timeout: Unable to fetch a connection in " + (maxWait / 1000
4              " seconds.");
5  } else {
6      continue;
7  }
```

关闭连接，也就是从 busy 队列移除，然后加入到 idle 队列中。

## 15 | mBatis : 如何将 SQL 语句配置化?

### 思考题


我们只是简单地实现了 select 语句的配置，如何扩展到 update 语句？进一步，如何实现读写分离？

### 参考答案

我们可以在 sql 节点类 MapperNode 中增加一个属性 sqltype，表示 sql 语句的类型，比如 0 表示 select，1 表示 update，2 表示 insert，3 表示 delete。这样我们就知道了一个 sql 语句是 read 还是 write。


然后 datasource 变成两个，一个是 readDataSource，一个是 writeDataSource，可以配置在外部文件中。JdbcTemplate 也提供一个 setDataSource() 允许动态设置数据源。

DefaultSqlSession 类中配置两个 data source，形如：

 复制代码

```
1 private DataSource readDataSource;  
2 private DataSource writeDataSource;
```

然后在 `selectOne()` 中这么判断：

 复制代码

```
1 public Object selectOne(String sqlid, Object[] args, PreparedStatementCallback  
2     int sqltype = this.sqlSessionFactory.getMapperNode(sqlid).getSqlType();  
3     if (sqltype==0) { //read  
4         jdbcTemplate.setDataSource(readDataSource);  
5     }  
6     return jdbcTemplate.query(sql, args, pstmtcallback);  
7 }
```

也就是说，每一次用 `SqlSession` 执行 SQL 语句的时候，都判断一下 SQL 类型，如果是 `read`，则设置 `readDataSource`，否则设置 `writeDataSource`。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

## 精选留言 (2)



风轻扬

2023-05-21 来自北京

读写分离的实现，按照老师的思路，实现了一下。

1、增加`sqlType`参数。

首先改造`MapperNode`，增加`sqlType`参数。

2、注入双数据源。

因为我们配置到`applicationContext.xml`中的是`SqlSessionFactory`，所以数据源的注入要在这个类中。

```
public class DefaultSqlSessionFactory implements SqlSessionFactory {
```

.....

```

@Autowired
private DataSource readDataSource;
@Autowired
private DataSource writeDataSource;

@Override
public SqlSession openSession() {
    .....
    defaultSqlSession.setReadDataSource(readDataSource);
    defaultSqlSession.setWriteDataSource(writeDataSource);
    return defaultSqlSession;
    .....
}
.....
}

```

3、根据sqlType将不同的数据源注入jdbcTemplate中

之后在openSession的时候，塞给SqlSession对象。当用户执行操作的时候，根据操作类型的不同，给JdbcTemplate注入不同的数据源

```

public class DefaultSqlSession implements SqlSession{

    .....

    private DataSource readDataSource;
    private DataSource writeDataSource;

    ....省略readDataSource和writeDataSource的set方法....

    @Override
    public Object selectOne(String sqlId, Object[] args, PreparedStatementCallback pstmtcallback) throws Exception {
        MapperNode mapperNode = this.sqlSessionFactory.getMapperNode(sqlId);
        if (mapperNode.getSqlType().equals("3")) {
            jdbcTemplate.setDataSource(readDataSource);
        }
        return jdbcTemplate.queryObject(mapperNode.getSql(), args, pstmtcallback);
    }
}

```

```
@Override
public Integer delete(String sqlId, Object[] args) throws Exception {
    MapperNode mapperNode = this.sqlSessionFactory.getMapperNode(sqlId);
    if (mapperNode.getSqlType().equals("1")) {
        jdbcTemplate.setDataSource(writeDataSource);
    }
    return jdbcTemplate.delete(mapperNode.getSql(), args);
}
.....
}
```

作者回复: 赞一个!



**peter**

2023-04-17 来自北京

请教老师几个问题:

Q1: 会讲解MiniTomcat吗?

很期望针对Tomcat也出一个类似的专栏。

Q2: MVC会导致低并发吗?

MVC用一个单一的 Servlet 拦截所有请求, 这个设计会降低系统的并发吗?

作者回复: 会跟平台商量是不是要出MiniTomcat。

并发问题需要系统级的解决(web server, servlet 容器, 缓存, 连接池等等), 不是单一servlet这一点可以决定的, servlet本身也是单实例多线程的。

