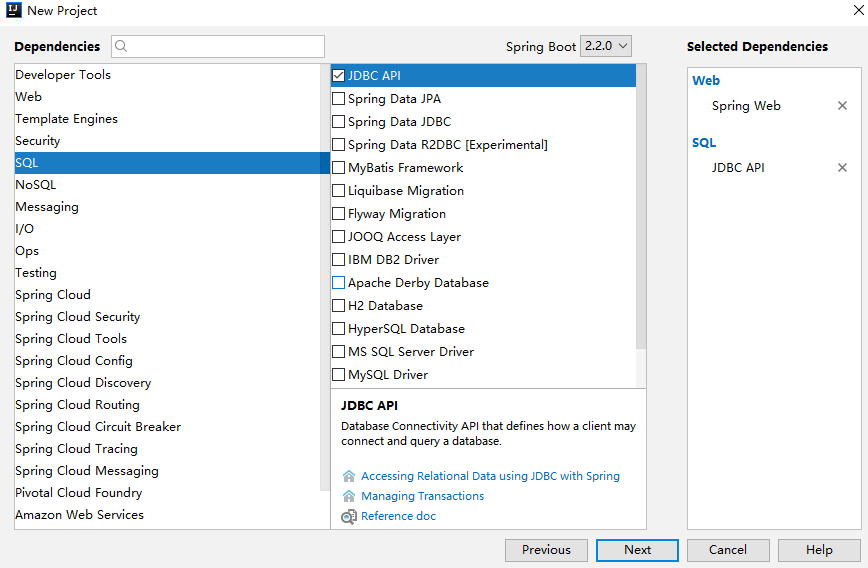
https://mp.weixin.qq.com/s/kF8ihKc2a0HkAiWQ33KclA

# Spring Boot 整合 JdbcTemplate

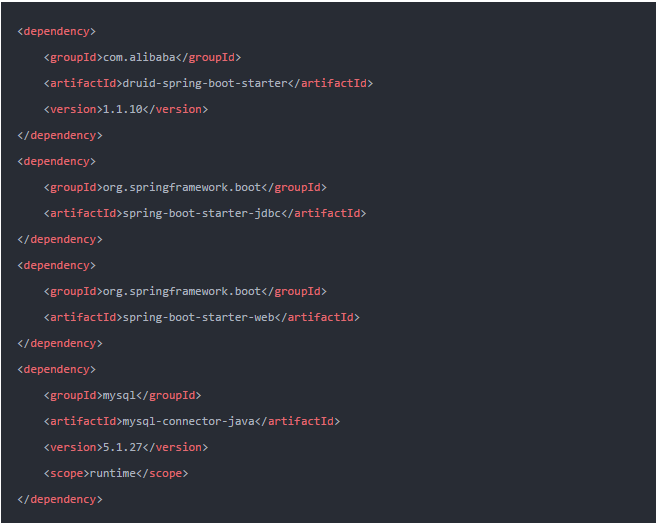
在 Java 领域，数据持久化有几个常见的方案，有 Spring 自带的 JdbcTemplate 、有 MyBatis，还有 JPA，在这些方案中，最简单的就是 Spring 自带的 JdbcTemplate 了，这个东西虽然没有 MyBatis 那么方便，但是比起最开始的 Jdbc 已经强了很多了，它没有 MyBatis 功能那么强大，当然也意味着它的使用比较简单，事实上，JdbcTemplate 算是最简单的数据持久化方案了，本文就和大伙来说说这个东西的使用。

## 1. 基本配置

JdbcTemplate 基本用法实际上很简单，开发者在创建一个 SpringBoot 项目时，除了选择基本的 Web 依赖，再记得选上 Jdbc 依赖，以及数据库驱动依赖即可，如下：



项目创建成功之后，记得添加 Druid 数据库连接池依赖（注意这里可以添加专门为 Spring Boot 打造的 druid-spring-boot-starter，而不是我们一般在 SSM 中添加的 Druid），所有添加的依赖如下：



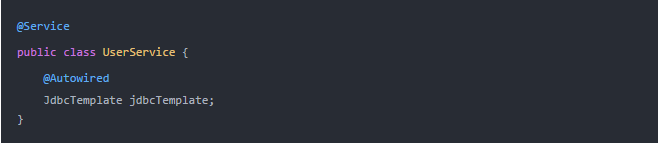
如此之后，所有的配置就算完成了，接下来就可以直接使用 JdbcTemplate 了？咋这么方便呢？其实这就是 SpringBoot 的自动化配置带来的好处，我们先说用法，一会来说原理。

## 2. 基本用法

首先我们来创建一个 User Bean，如下：



然后来创建一个 UserService 类，在 UserService 类中注入 JdbcTemplate ，如下：

好了，如此之后，准备工作就算完成了。

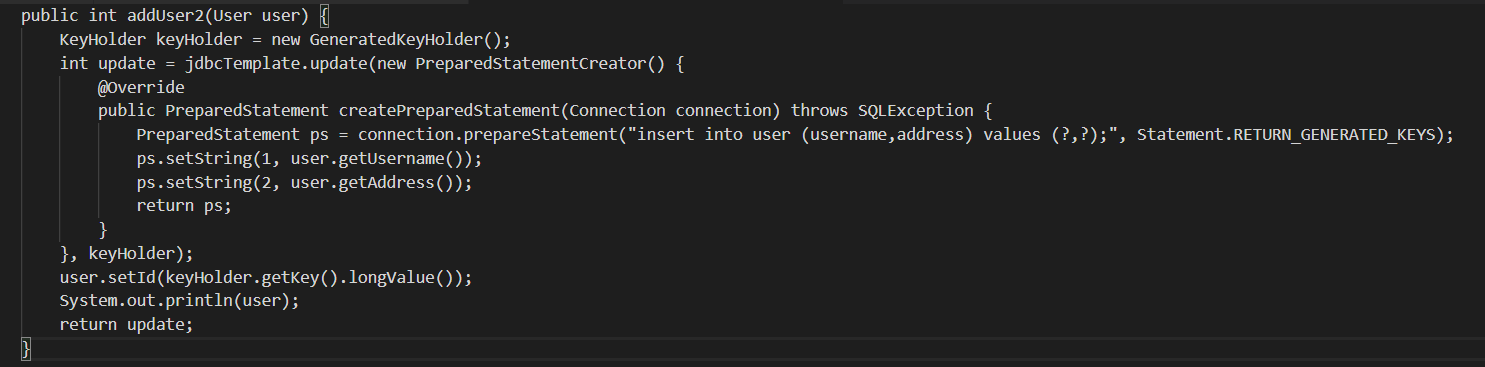
### 2.1 增

JdbcTemplate 中，除了查询有几个 API 之外，增删改统一都使用 update 来操作，自己来传入 SQL 即可。例如添加数据，方法如下：

public int addUser(User user) {  
 return jdbcTemplate.update("insert into user (username,address) values (?,?);", user.getUsername(), user.getAddress());  
}

update 方法的返回值就是 SQL 执行受影响的行数。

这里只需要传入 SQL 即可，如果你的需求比较复杂，例如在数据插入的过程中希望实现主键回填，那么可以使用 PreparedStatementCreator，如下：

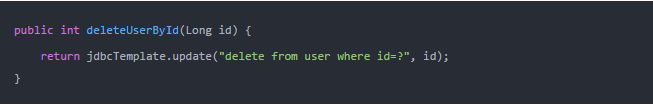


实际上这里就相当于完全使用了 JDBC 中的解决方案了，首先在构建 PreparedStatement 时传入 Statement.RETURN\_GENERATED\_KEYS，然后传入 KeyHolder，最终从 KeyHolder 中获取刚刚插入数据的 id 保存到 user 对象的 id 属性中去。

你能想到的 JDBC 的用法，在这里都能实现，Spring 提供的 JdbcTemplate 虽然不如 MyBatis，但是比起 Jdbc 还是要方便很多的。

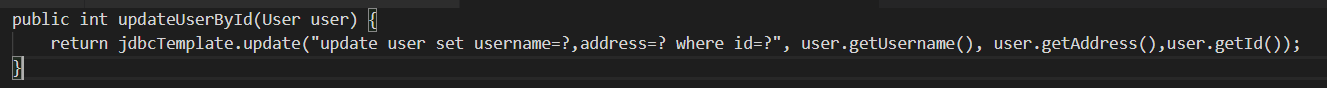
### 2.2 删

删除也是使用 update API，传入你的 SQL 即可：



当然你也可以使用 PreparedStatementCreator。

### 2.3 改



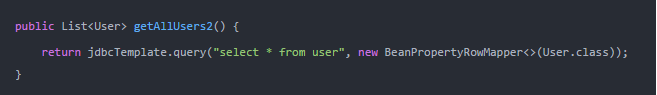
当然你也可以使用 PreparedStatementCreator。

### 2.4 查

查询的话，稍微有点变化，这里主要向大伙介绍 query 方法，例如查询所有用户：



查询的时候需要提供一个 RowMapper，就是需要自己手动映射，将数据库中的字段和对象的属性一一对应起来，这样。。。。嗯看起来有点麻烦，实际上，如果数据库中的字段和对象属性的名字一模一样的话，有另外一个简单的方案，如下：

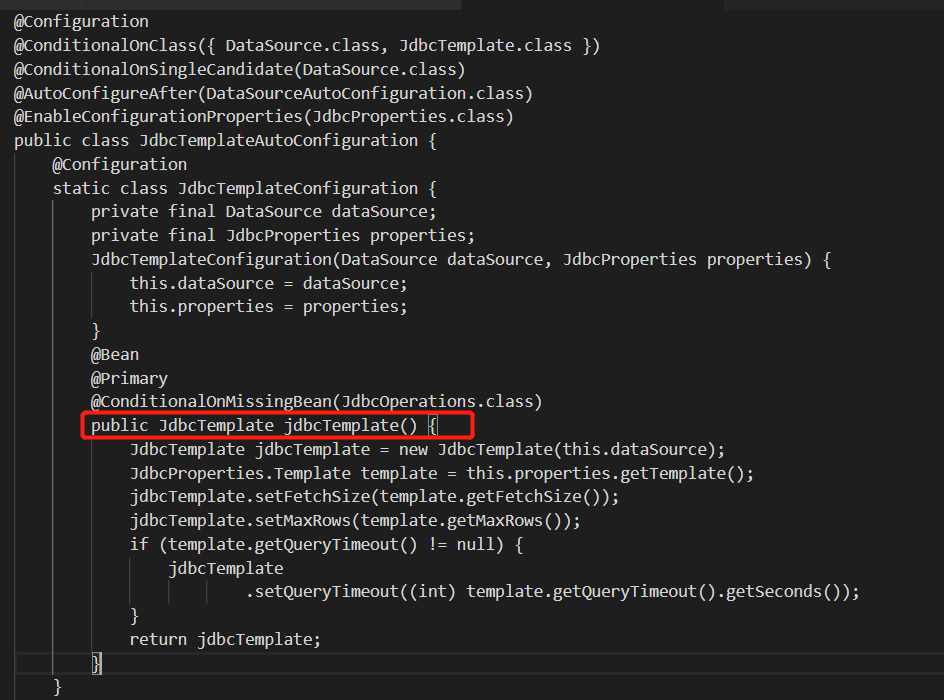


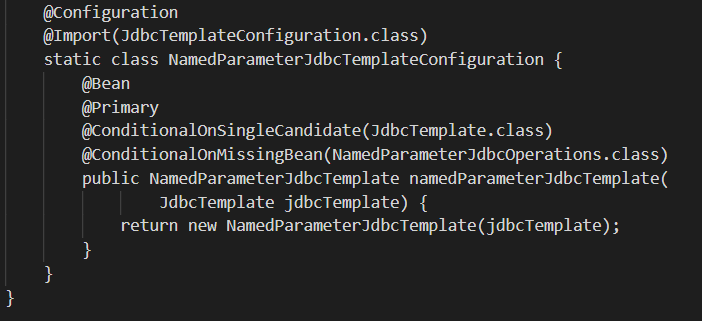
### 2.5 其他

除了这些基本用法之外，JdbcTemplate 也支持其他用法，例如调用存储过程等，这些都比较容易，而且和 Jdbc 本身都比较相似，这里也就不做介绍了，有兴趣可以留言讨论。

## 3. 原理分析

那么在 SpringBoot 中，配置完数据库基本信息之后，就有了一个 JdbcTemplate 了，这个东西是从哪里来的呢？源码在 org.springframework.boot.autoconfigure.jdbc.JdbcTemplateAutoConfiguration 类中，该类源码如下：





从这个类中，大致可以看出，当当前类路径下存在 DataSource 和 JdbcTemplate 时，该类就会被自动配置，jdbcTemplate 方法则表示，如果开发者没有自己提供一个 JdbcOperations 的实例的话，系统就自动配置一个 JdbcTemplate Bean（JdbcTemplate 是 JdbcOperations 接口的一个实现）。好了，不知道大伙有没有收获呢？