**今天完成的事情：**

**第一部分：JAVA基础**

**学习预览：**

**第二部分：修真任务**

**任务预览：**

* JDBC template

**第二部分**

1. **JDBC template**

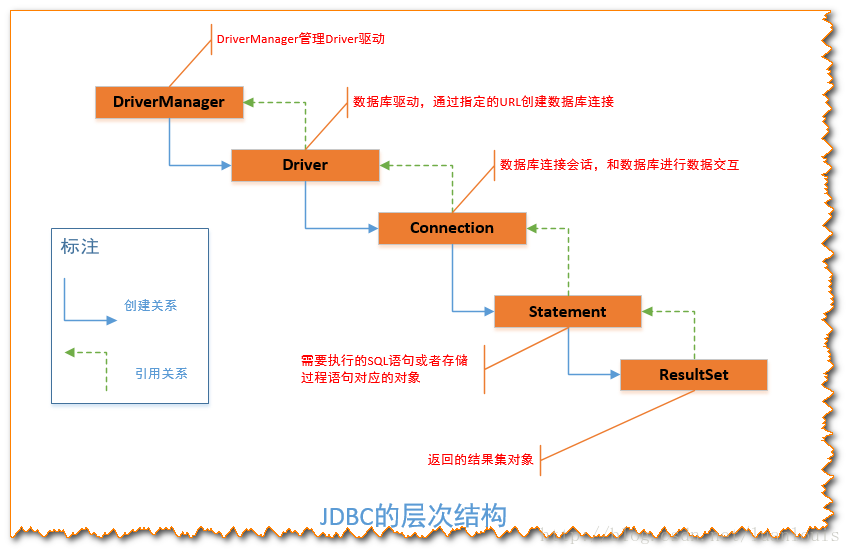
今天开始写JDBC template的代码构件，首先熟悉一下JDBC的概念

* **JDBC（Java DataBase Connectivity）是一种用于执行SQL语句的Java API，由一组用Java语言编写的类和接口组成，可以为多种关系数据库提供统一访问**
* **JDBC Template是Spring框架的构件，对数据库的操作在 jdbc上做了深层次的封装，使用Spring的注入功能，可以把DataSource注册到JDBC Template中**

1. **JDBC传统结构**

**JDBC（Java DataBase Connectivity）是一种用于执行SQL语句的Java API，提供连接各种数据库的能力，其提供一套标准接口，实现对各数据库的增删改查等操作**

**JDBC构件包含了几个执行层次，具体关系图如下所示：**



**JDBC包含了从连接数据库到操作数据库的整个流程**

**以下对传统JDBC基本构成进行分析：**

1. **加载驱动程序：**



1. **获得数据连接：**



1. **创建Statement对象：**



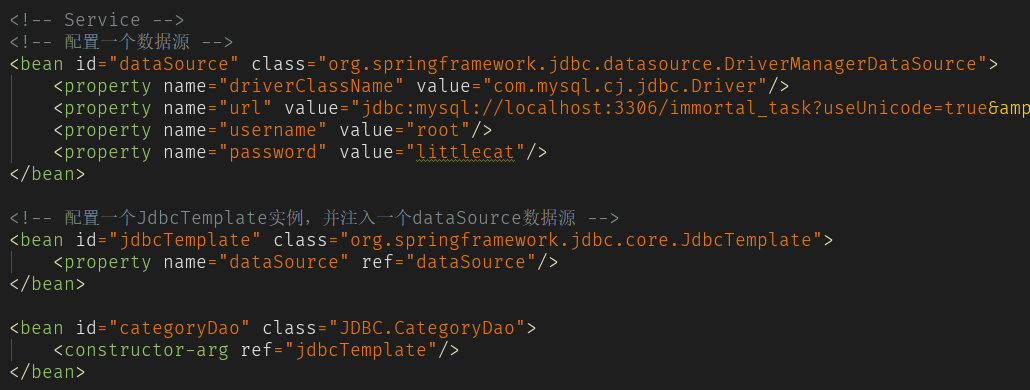
1. **完整实例**



1. **JDBC Template**

**Spring对数据库的操作在JDBC上面做了深层次的封装，使用Spring的注入功能，可以把DataSource注册到JdbcTemplate之中**

1. **JDBC配置文件**



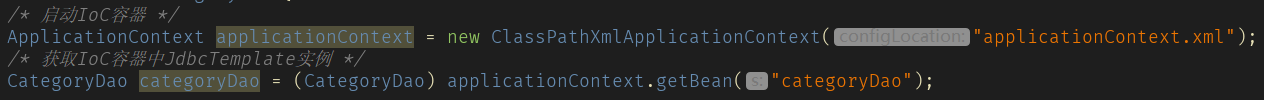
**第一行：用来配置一个数据源，赋予一个id，并标注路径Class**

**第二行：数据源的相关属性，接口**

**第八行：配置JdbcTemplate实例，property中ref为引用类**

1. **初始化容器**

**写在执行文件头部**



**1# 启动IoC容器**

**ApplicationContext applicationContext = new ClassPathXmlApplicationContext（“配置文件名”）**

**2# 获取IoC容器中JdbcTemplate实例**

**CategoryDao categoryDao = (强制转换类型名称) applicationContext.getBean("实现类名称");**

1. **DAO 模式**

在写JdbcTemplate功能的时候，需要明确的一个概念就是DAO模式

**DAO (DataAccessobjects 数据存取对象)是指位于业务逻辑和持久化数据之间实现对持久化数据的访问。通俗来讲，就是将数据库操作都封装起来**

**DAO处于JDBC底层中，包含JDBC对数据库的Statement和Result两个操作，负责对数据库操作的实现与调用的进行封装。**

**DAO模式对外提供相应的接口**

* **DAO 模式提供了访问关系型数据库系统所需操作的接口**
* **将数据访问和业务逻辑分离，对上层提供面向对象的数据访问接口**

**一个典型的DAO 模式主要由以下几部分组成：**

* DAO接口
* DAO 实现类
* 实体类
* 数据库连接和关闭工具类

**从DAO 模式使用可以看出，DAO 模式的优势就在于它实现了两次隔离：**

* **隔离了数据访问代码和业务逻辑代码**

业务逻辑代码直接调用DAO方法即可，完全感觉不到数据库表的存在。分工明确，数据访问层代码变化不影响业务逻辑代码,这符合单一职能原则，降低了藕合性，提高了可复用性

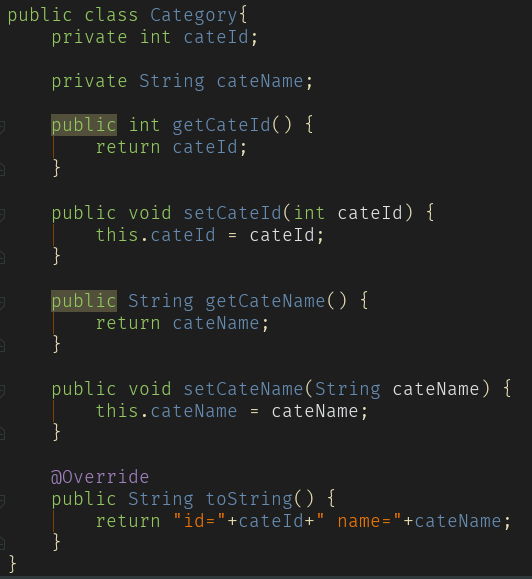
* **隔离了不同数据库实现**

采用面向接口编程，如果底层数据库变化，如由 MySQL 变成 Oracle 只要增加 DAO 接口的新实现类即可，原有 MySQ 实现不用修改。

这符合 "开-闭" 原则。该原则降低了代码的藕合性，提高了代码扩展性和系统的可移植性

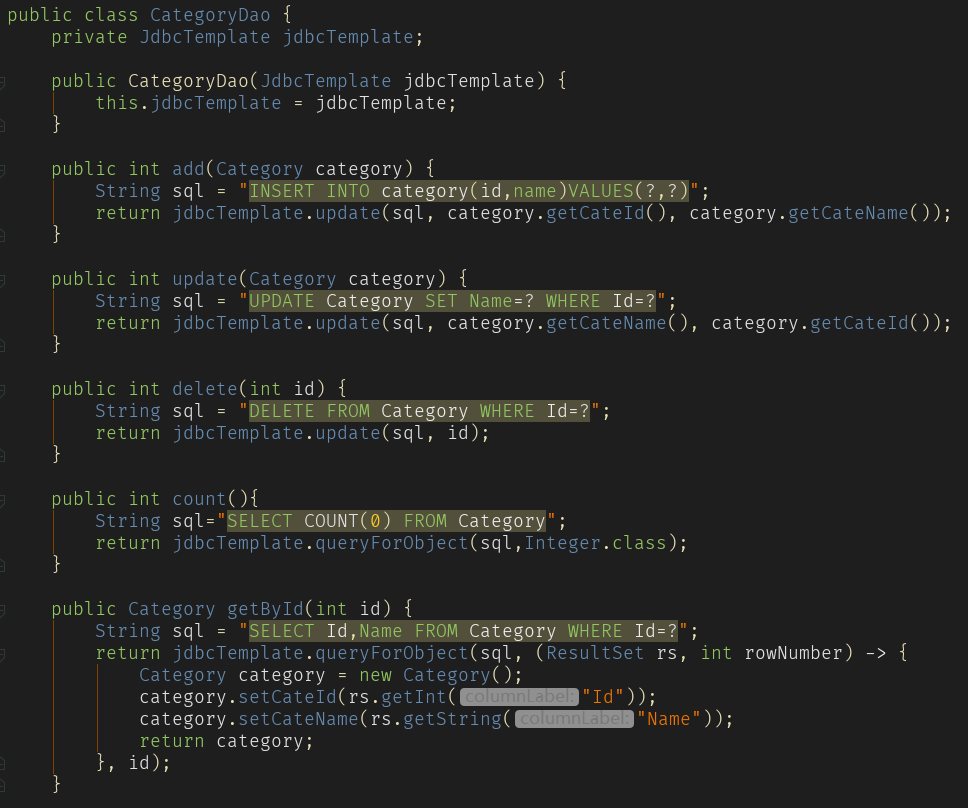
1. **实例代码部分**
2. **创建实体类**

实现getter()和setter()方法



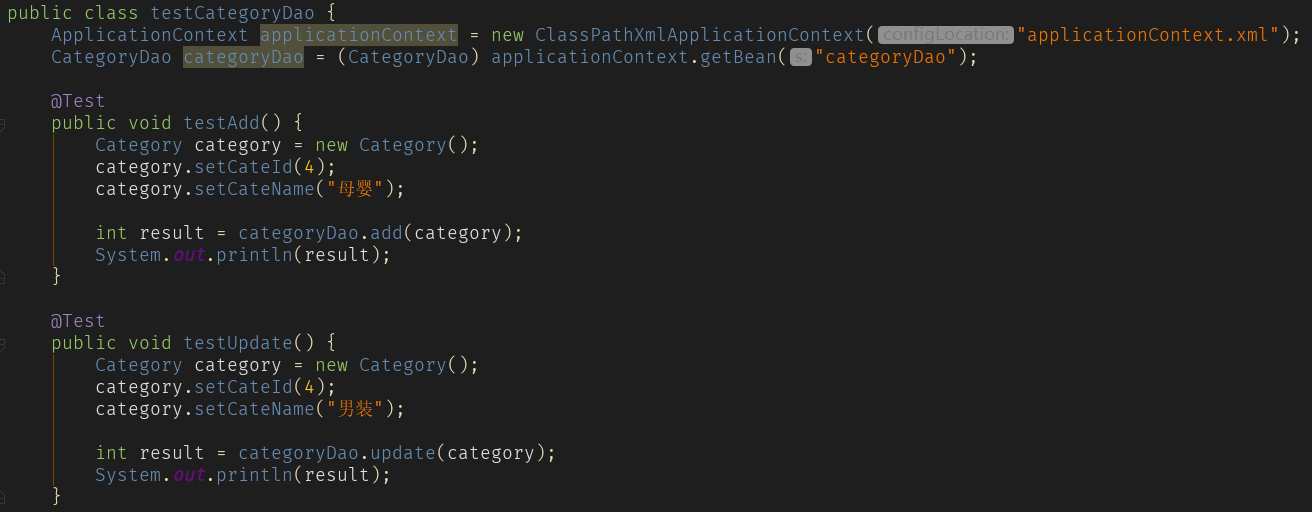
1. **创建数据访问实现类（DAO）**

执行对数据库的增删改查操作



1. **创建测试类**

执行业务操作



1. **遇到的问题（报错）**
2. **第一个问题：XML配置文件的装载问题**

因为在测试类中使用了注释的方法加载XML文件，导致莫名其妙的无法使用的问题，所以最后改为自动加载方法

**1# 注释方法加载（问题）**

@ContextConfiguration(locations = "classpath:applicationContext.xml")

（整合JUnit测试时，使用注解引入配置文件）

@RunWith(SpringJUnit4ClassRunner.class)

（让测试运行于Spring测试环境）

类内代码：

@Autowired

private CategoryDao categoryDao;

**2# 自动加载方法（解决方案）**

类内代码：

ApplicationContext applicationContext = new ClassPathXmlApplicationContext("applicationContext.xml");

CategoryDao categoryDao = (CategoryDao) applicationContext.getBean("categoryDao");

1. **第二个问题：创建项目时，使用了Spring boot框架的不兼容问题**

Spring boot框架的配置原则是 约定大于配置，所以在配置XML文件时，会发生无法装载等问题

重新建立项目，使用Maven管理，导入基本Spring核心包

1. **第三个问题：数据库版本与MySQL支持包版本不兼容**

数据库版本为8.x MySQL支持包版本为5.x

所以有不兼容问题

换支持包版本为8.x解决问题

1. **第四个问题：Junit监听器报错**

**问题原因不清楚，但是换了稳定的JUnit4.12版本后解决问题**

1. **第五个问题：客户端不支持服务器的验证协议**

**Client does not support authentication protocol requested by server**

**查询资料后发现是因为MySQL版本的问题：**

MySQL8.0版本的加密方式和MySQL5.0的不一样，连接会报错

更改MySQL加密方式即可

1. **第六个问题：测试方法testAdd()一直指向第三个字段**

* 原因是数据库表单字段比修改的字段多，由于字段属性设置为非null，所以每次增加删减数据时，空的字段会导致报错
* 如果字段与修改字段相等时还报错，就是因为设置属性时，没有使用包装类，导致变量初始化时赋值为空，从而引起报错
* **包装类：**赋值时初始化为NULL
* **基本数据类型：**赋值时初始化为0

取消字段的非null属性既解决问题

1. **第七个问题：其中DAO层实现类总是出现返回值错误**

Incorrect result size: expected 1, actual 0

执行代码为：



牵扯到方法需要再搜索学习一下

：

1. jdbcTemplate.queryForObject(String sql, Class<T> requiredType, Object... args)
2. jdbcTemplate.query()

**明天计划完成的事情：**

今天基本把JdbcTemplate整体操作流程完成了，并且使用SQL语句操作数据成功

明天计划学习使用MyBatis，单元测试等知识

**遇到的问题：**

各种BUG，各种不兼容，多亏有Junit插件帮忙报错提示，能解决一部分

**收获：**

成功运行JdbcTemplate构件，完成了整个过程的代码编写和运行

遇到几个BUG，在解决过程中也体验了下莫名其妙的各种问题，边学习边修改

对JdbcTemplate整体结构有了基本的概念，希望能进一步加深印象，继续学习