JdbcTemplate继承了基类JdbcAccessor和接口类JdbcOperation。

* 在基类JdbcAccessor的设计中，对DataSource数据源进行管理和配置。

在JdbcTemplate接口中，定义了通过Jdbc操作数据库的基本方法，而JdbcTemplate提供这些接口方法的实现，比如execute方法、query方法、update方法等。

### JdbcTemplate常用方法

查询：

1）queryForObject：spring 3.2.2版本之后jdbcTemplate.queryForInt()和jdbcTemplate.queryForLong()就取消了，全部用queryForObject代替。

T queryForObject(String sql, Class<T> requiredType)

本质上和queryForInt相同，只是可以返回不同的对象，例如返回一个String对象。

String sql ="select count(\*) from user";

Integer num =jdbcTemplate.queryForObject(sql, Integer.class);

T queryForObject(String sql, Class<T> requiredType, Object... args)

第三个参数是可变参数，用来放动态绑定的参数。

String sql = "select count(\*) fromuser where ID<? AND ID>?";

jdbcTemplate.queryForObject(sql,Integer.class,4,2 );

T queryForObject(String sql, Object[] args, Class<T> requiredType)

第二个参数是参数数组。

String sql = "select count(\*) fromuser where ID>? AND USER\_NAME LIKE ?";

jdbcTemplate.queryForObject(sql, newObject[]{1,"%哈%"},Integer.class);

T queryForObject(String sql, RowMapper<T> rowMapper)

这里查询必须保证只能查询一条数据，否则会报错。

String sql = "select \* from user WHERE ID = 1";

User user = jdbcTemplate.queryForObject(sql,new RowMapper<User>(){

@Override

publicUser mapRow(ResultSet rs, int rowNum) throws SQLException {

    Useruser = new User();

     user.setId(rs.getInt("ID"));

     user.setUserName(rs.getString("USER\_NAME"));

     return user;

    }

});

2）queryForList：返回一个装有map的list，每一个map是一条记录，map里面的key是字段名。

List<Map<String, Object>> rows=

jdbcTemplate.queryForList("SELECT \* FROM user");

//得到装有map的list

for(int i=0;i<rows.size();i++){    //遍历

Map userMap=rows.get(i);

System.out.println(userMap.get("id"));

System.out.println(userMap.get("name"));

System.out.println(userMap.get("age"));

}

3）queryForMap：这个插叙只能是查询一条记录，返回一个map，key的值是column的值。

Map<String, Object> map =

jdbcTemplate.queryForMap("select \* from user");

map.get("name ");

4）queryForRowSet：返回一个结果集然后调用.getString或者getInt等去取值。

5）query：

更新：

1）execute：执行sql语句，无返回值，用于更新操作（增、删、改）

2）update：更新操作，返回受影响的行数

3）batchUpdate：执行批量更新，参数为String数组

注：jdbcTemplate尽量只执行查询操作，莫要进行更新，否则会破坏Hibernate的二级缓存体系

调用存储过程：

JdbcTemplate提供了两个调用存储过程的接口方法：

1）<T> T execute(String callString, CallableStatementCallback<T> action)：

用户通过callString参数指定调用存储过程的SQL语句。

第二个参数CallableStatementCallback<T>是一个回调接口，该接口只有一个方法T doInCallableStatement(CallableStatement cs)，用户可以在该方法中进行输入参数绑定，输出参数注册一级返回数据处理等操作。

2）<T> T execute(CallableStatementCreator csc, CallableStatementCallback<T> action)

使用CallableStatementCreator创建CallableStatement，CallableStatementCreator接口定义了一个方法：CallableStatement createCallableStatement(Connection con)，它使用Connection实例创建CallableStatement对象。CallableStatementCallable<T>负责处理存储过程的返回结果。

Spring与Dao部分，是Spring的两大核心技术IoC与AOP的典型应用体现。

* JDBC模板的使用是IoC的应用，将Jdbc模板对象注入给了Dao层的实现类。
* Spring的事务处理是AOP的应用，将事务作为切面织入到Service层的业务方法中。

Spring提供了一个强有力的模板类——JdbcTemplate来简化JDBC操作。并且，数据源DataSource对象与模板JdbcTemplate对象均可通过Bean的形式定义在配置文件中，充分发挥了依赖注入的作用。

### 1、导入jar包

除了Spring的基本Jar包，数据库驱动Jar包，还需要spring-jdbc包和spring-tx包。

### 2、搭建测试环境

1）定义实体类User

2）定义数据库及表

3）定义UserDao接口

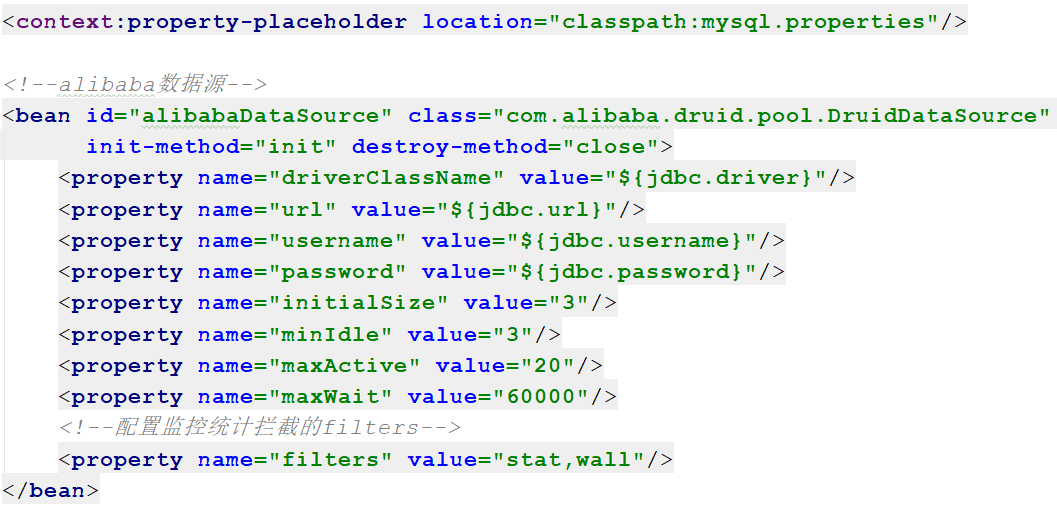
4）初步定义UserDaoImpl

5）定义UserService接口

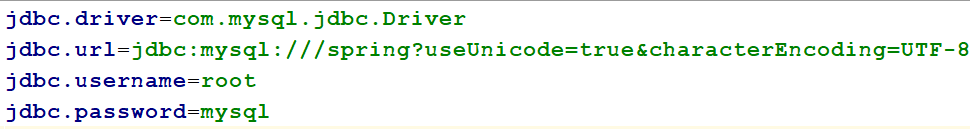
6）定义UserServiceImpl

7）定义测试类

### 3、数据源的配置

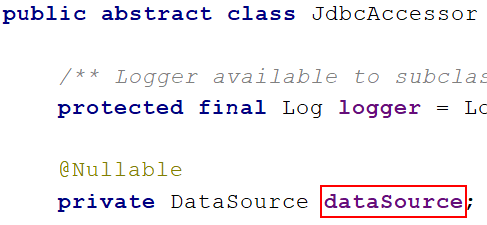


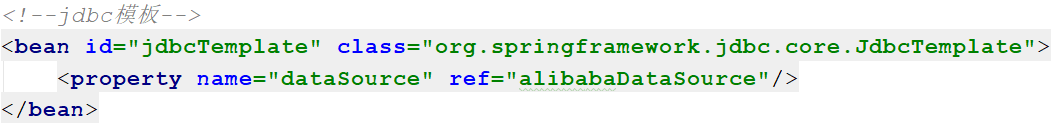
### 4、从属性文件读取数据库连接



### 5、配置JDBC模板

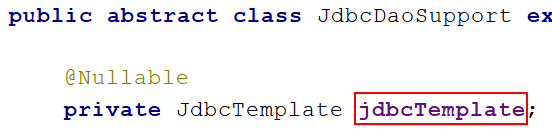
JDBC模板类J的保存Template从其父类JdbcAccessor继承了一个属性dataSource，用于接收数据源。



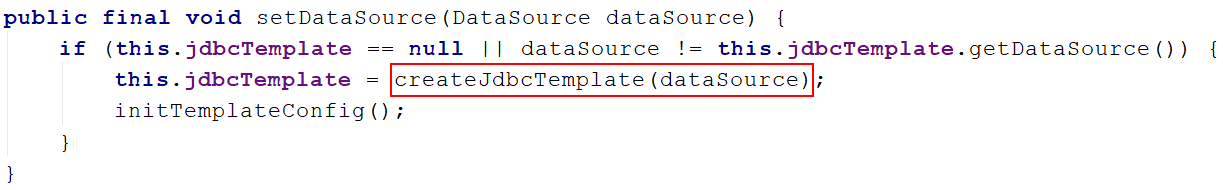


### 6、Dao实现类继承JdbcDaoSupport类

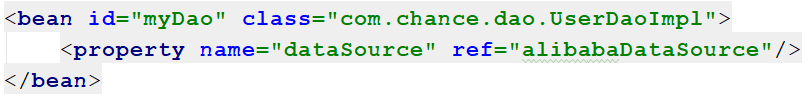
JdbcDaoSupport类个有一个属性JdbcTemplate，用于接收JDBC模板。所以Dao实现类继承JdbcDaoSupport类后，也就具有了JDBC模板属性。配置文件中，只要将模板对象注入即可。



再看JdbcDaoSupport类，发现其有一个dataSource属性，查看setDataSource()方法体可知，若JDBC模板为null，则会自动创建一个模板对象。



因此，spring配置文件中对JDBC模板对象的配置完全可以省去，而是在Dao实现类中直接注入数据源对象。这样会让系统自动创建JDBC模板对象。



### 7、对DB的增、删、改操作

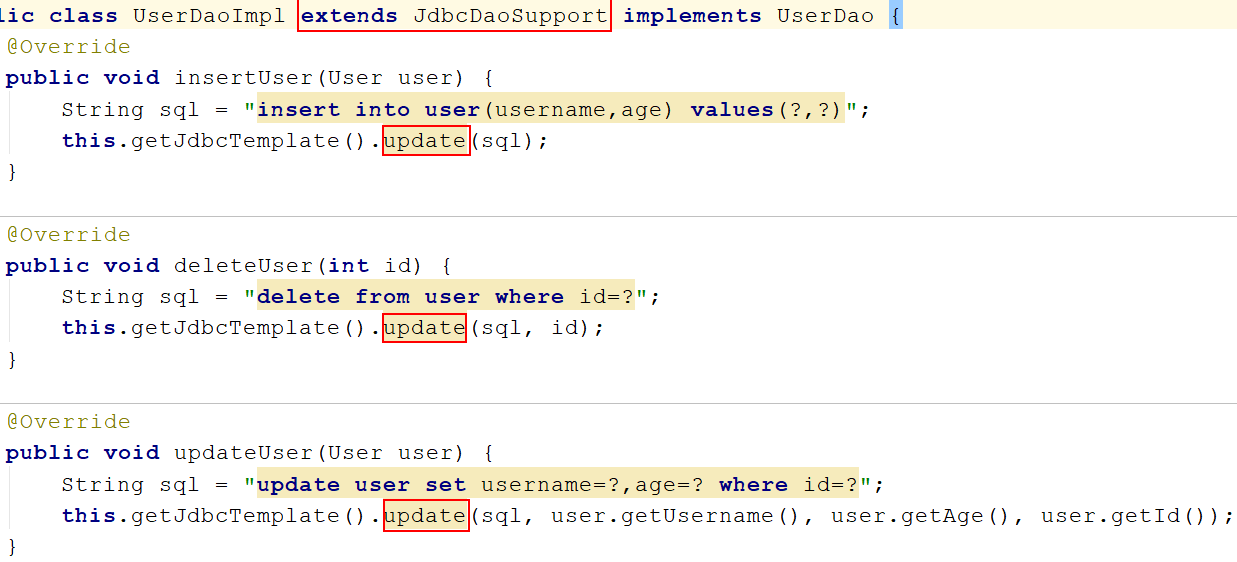
JdbcTemplate类中提供了对DB进行修改、查询的方法。Dao实现类使用继承自JdbcDaoSupport的getTemplate()方法，可以获取到JDBC模板对象。

对DB的增删改都是通过update()方法实现的。该方法常用的重载方法有两个：

int update(String sql)

int update(String sql, Object... args)

第一个参数为要执行的sql语句，第二个参数为sql语句中所包含的动态参数。



### 8、对DB的查询操作

JDBC模板的查询结果均是以对象的形式返回。根据返回对象类型的不同，可以将查询分为两类：简单对象查询、自定义对象查询。

1）简单对象查询

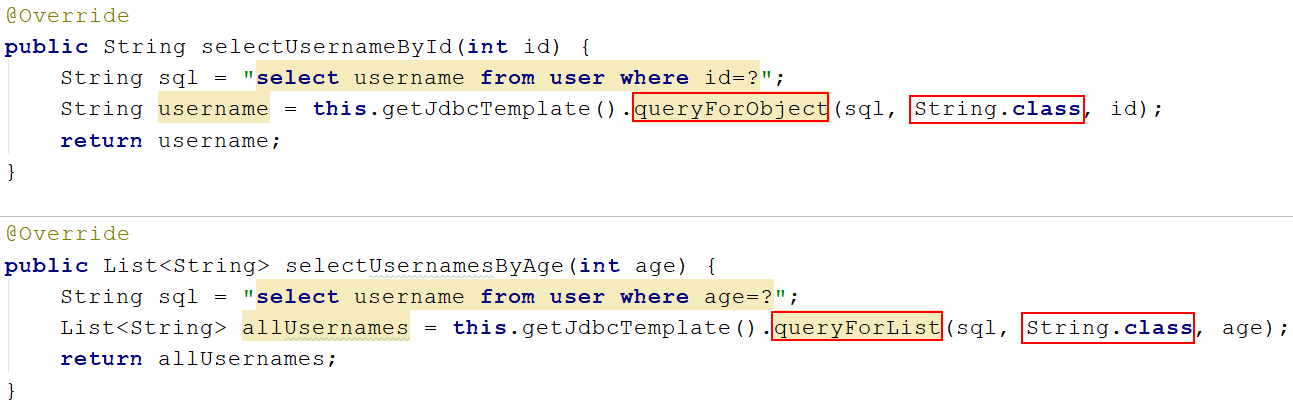
查询结果为String、Integer等简单对象类型，或该类型作为元素的集合类型，如List<User>等。

//查询结果为单个对象

pubic T queryForObject (String sql, Class<T> type, Object... args)

//查询结果为List

pubic List<T> queryForList (String sql, Class<T> type, Object... args)



2）自定义对象查询

常用的自定义对象查询方法有：

//查询结果为单个对象

pubic T queryForObject (String sql, RowMapper<T> m , Object... args)

//查询结果为List

pubic List<T> query (String sql, RowMapper<T > m, Object... args)

注意：

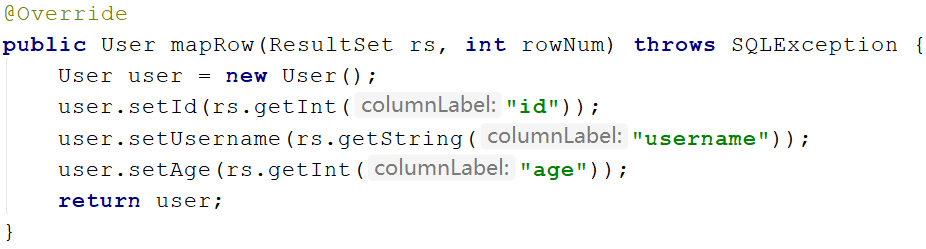
RowMapper为记录映射接口，用于将查询结果集中每一条记录包装为指定对象。该接口中有一个方法需要实现：

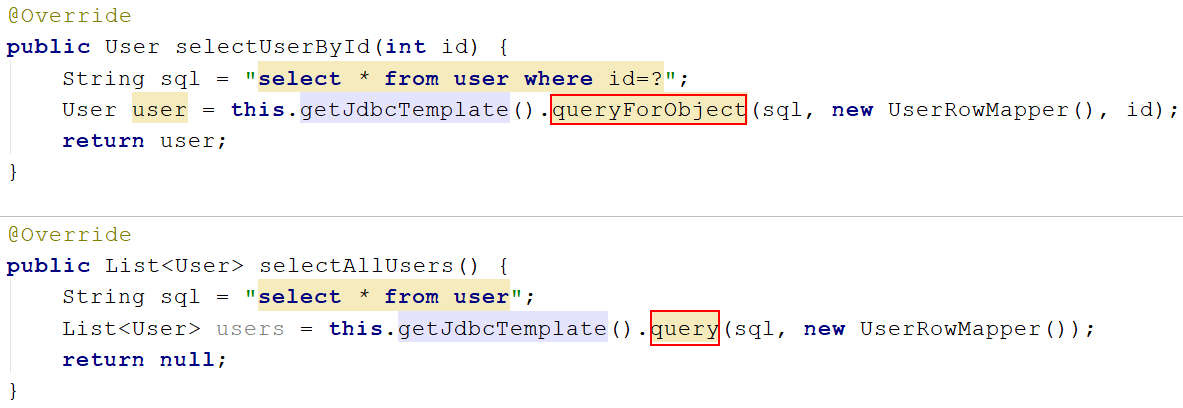
//实现将查询结果中当前行的数据包装为一个指定对象

public Object mapRow(ResultSet rs, int rowNum)

//参数rowNum表示总的结果集中当前行的行号

//参数rs表示rowNum所代表的当前行的记录所定义的结果集，仅仅是当前行的结果。





### 9、JDBC模板是多例的

JdbcTemplate对象是多例的，即系统会为每一个使用模板对象的线程（方法）创建一个JdbcTemplate实例，并且在该线程（方法）结束时，自动释放JdbcTemplate实例。所以在每次使用JdbcTemplate对象时，都需要通过getJdbcTemplate()方法获取。