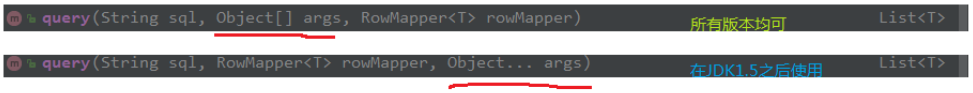
JdbcTemplate实现jdbc的基本操作以及几个重要的包

数据库连接　url=　jdbc:mysql://localhost:3306/mybatis?characterEncoding=utf8

一、查询操作(重点)

我们有什么：sql语句，语句的参数
我们要什么 返回一个List集合

针对不同的JDK版本



1、BeanPropertyRowMapper

实现了 RowMapper 接口；

```
List<Account> accounts = jt.query("select * from account where money > ?",new BeanPropertyRowMapper<Account>(Account.class),1000f);
```

可以代替以下代码：

```
//查询所有
//需要定义AccountRowMapper实现AccountRowMapper
List<Account> accounts = jt.query("select * from account where money > ?",new AccountRowMapper(),1000f);

//查询一个
List<Account> accounts = jt.query("select * from account where id = ?",new BeanPropertyRowMapper<Account>(Account.class),1);
System.out.println(accounts.isEmpty()?"没有内容":accounts.get(0));

//查询返回一行一列（使用聚合函数，但不加group by子句）
Long count = jt.queryForObject("select count(*) from account where money > ?",Long.class,1000f);
```

定义AccountRowMapper 实现RowMapper，由spring**自动**把每个Account加到集合中

```
/**
 * 定义Account的封装策略
 */
class AccountRowMapper implements RowMapper<Account>{
    /**
     * 把结果集中的数据封装到Account中，然后由spring把每个Account加到集合中
     * @param rs
     * @param rowNum
     * @return
     * @throws SQLException
     */
    @Override
    public Account mapRow(ResultSet rs, int rowNum) throws SQLException {
        Account account = new Account();
        account.setId(rs.getInt("id"));
        account.setName(rs.getString("name"));
        account.setMoney(rs.getFloat("money"));
        return account;
    }
}
```

二、增删改

```
public static void main(String[] args) {
    //准备数据源：spring的内置数据源
    DriverManagerDataSource ds = new DriverManagerDataSource();
```

```
    ds.setDriverClassName("com.mysql.jdbc.Driver");
    ds.setUrl("jdbc:mysql://localhost:3306/eesy");
    ds.setUsername("root");
    ds.setPassword("1234");

    //1.创建JdbcTemplate对象
    JdbcTemplate jt = new JdbcTemplate();
    //给jt设置数据源
    jt.setDataSource(ds);
    //2.执行操作
    jt.execute("insert into account(name,money)values('ccc',1000)");

      //保存
    jt.update("insert into account(name,money)values(?,?)","eee",3333f);


    //更新
    jt.update("update account set name=?,money=? where id=?","test",4567,7);


    //删除
   jt.update("delete from account where id=?",8);
```

三、JdbcDaoSupport

1、JdbcDaoSupport 可以直接由Spring支持

```
import org.springframework.jdbc.core.support.JdbcDaoSupport;

public class AccountDaoImpl extends JdbcDaoSupport implements IAccountDao {
    }


//注入数据<!-- 配置账户的持久层-->
<bean id="accountDao" class="com.itheima.dao.impl.AccountDaoImpl">
    <property name="dataSource"  ref="dataSource"></property>
</bean>

<!-- 配置数据源-->
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
    <property name="url" value="jdbc:mysql://localhost:3306/eesy"></property>
    <property name="username" value="root"></property>
    <property name="password" value="1234"></property>
</bean>
```

2、它的底层实现：

```
@Repository
public class AccountDaoImpl2 extends  JdbcDaoSupport  implements IAccountDao {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    @Override
    public Account findAccountById(Integer accountId) {
        List<Account> accounts = jdbcTemplate.query("select * from account where id = ?",
        new BeanPropertyRowMapper<Account>(Account.class),accountId);
        return accounts.isEmpty()?null:accounts.get(0);
    }
}
```

为了将 @Autowired　　private JdbcTemplate jdbcTemplate;

这部分代码提取出来，建立并继承JdbcDaoSupport实现

```
public class JdbcDaoSupport {

    private JdbcTemplate jdbcTemplate;

    public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    public JdbcTemplate getJdbcTemplate() {
        return jdbcTemplate;
    }


    public void setDataSource(DataSource dataSource) {
        if(jdbcTemplate == null){
            jdbcTemplate = createJdbcTemplate(dataSource);
        }
    }

    private JdbcTemplate createJdbcTemplate(DataSource dataSource){
        return new JdbcTemplate(dataSource);
    }
}
```

然后便可以通过注入dataSource实现

```
<!-- 配置账户的持久层-->
<bean id="accountDao" class="com.itheima.dao.impl.AccountDaoImpl">
    <property name="dataSource"  ref="dataSource"></property>
</bean>

<!-- 配置数据源-->
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
    <property name="url" value="jdbc:mysql://localhost:3306/eesy"></property>
    <property name="username" value="root"></property>
    <property name="password" value="1234"></property>
</bean>
```

来代替原本的

```
<!-- 配置账户的持久层-->
<bean id="accountDao" class="com.itheima.dao.impl.AccountDaoImpl">
    <property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>

<!-- 配置数据源-->
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
    <property name="url" value="jdbc:mysql://localhost:3306/eesy"></property>
    <property name="username" value="root"></property>
    <property name="password" value="1234"></property>
</bean>
```