

课程主题

手写mybatis框架V1和V2版本

课程目标

1. 了解mybatis录播课程中有哪些知识点（课下看预习录播课程）
2. 通过手写mybatis框架的V1版本（[JDBC+properties配置文件](#)），去了解[参数处理](#)的思路和[结果映射](#)的思路
3. 不断去改造V1版本，将V1版本的代码，逐渐改造成通用代码（多个业务复用）
4. 通过手写mybatis框架的V2版本（JDBC+XML配置文件+其他，以面向过程思维去编码）

课程内容

1. OOAD（面向对象分析及设计）
2. 程序设计能力----手写公共组件、程序设计（类或者接口的设计、设计模式）
3. 源码阅读能力----

有问题的JDBC

```
1  @Test
2  public void test() {
3      Connection connection = null;
4      PreparedStatement preparedStatement = null;
5      ResultSet rs = null;
6
7      try {
8          // 加载数据库驱动(硬编码、连接的频繁创建)
9          Class.forName("com.mysql.jdbc.Driver");
10
11          // 通过驱动管理类获取数据库链接connection = DriverManager
12          connection =
13          DriverManager.getConnection("jdbc:mysql://localhost:3306/mybatis?
14          characterEncoding=utf-8","root", "root");
15
16          // 定义sql语句 ?表示占位符
17          String sql = "select * from user where username = ?";
18
19          // 获取预处理 statement
20          preparedStatement = connection.prepareStatement(sql);
21
22          // 设置参数，第一个参数为 sql 语句中参数的序号（从 1 开始），第二个参数为
23          设置的
24          preparedStatement.setString(1, "王五");
25
26          // 向数据库发出 sql 执行查询，查询出结果集
```

```

24         rs = preparedStatement.executeQuery();
25
26         // 遍历查询结果集
27         while (rs.next()) {
28             System.out.println(rs.getString("id") + " " +
rs.getString("username"));
29         }
30     } catch (Exception e) {
31         e.printStackTrace();
32     } finally {
33         // 释放资源
34         if (rs != null) {
35             try {
36                 rs.close();
37             } catch (SQLException e) {
38                 e.printStackTrace();
39             }
40         }
41         if (preparedStatement != null) {
42             try {
43                 preparedStatement.close();
44             } catch (SQLException e) {
45                 e.printStackTrace();
46             }
47         }
48         if (connection != null) {
49             try {
50                 connection.close();
51             } catch (SQLException e) {
52                 // TODO Auto-generated catch block e.printStackTrace();
53             }
54         }
55     }
56 }

```

升级改造

XML配置文件 ([参考Mybatis的XML](#))

- 全局配置文件 (配置的是和业务无关的全局信息，比如数据源信息)

```

1  <configuration>
2      <properties resource="phase01/db.properties"></properties>
3      <environments default="development">
4          <environment id="development">
5              <transactionManager type="JDBC" />
6              <dataSource type="POOLED">
7                  <property name="driver" value="${db.driver}" />
8                  <property name="url" value="${db.url}" />
9                  <property name="username" value="${db.username}" />
10                 <property name="password" value="${db.password}" />
11             </dataSource>
12         </environment>
13     </environments>
14     <mappers>
15         <mapper resource="phase01/UserMapper.xml" />
16     </mappers>

```

- 映射文件（和业务有关的配置信息，比如SQL语句、参数信息、映射结果信息。

```

1  <mapper namespace="test">
2
3      <select id="findUserById" parameterType="int"
4          resultType="com.kkb.mybatis.phase01.po.User"
5          statementType="STATEMENT">
6          SELECT * FROM user WHERE id = #{id} AND usernaem = #{username}
7
8      </select>
9      <select id="findUserByName" parameterType="string"
10         resultType="com.kkb.mybatis.phase01.po.User">
11         SELECT * FROM user WHERE username = ${value}
12     </select>
13 </mapper>

```

手写框架思路分析

整体思路

首先写V1.0版本：直接改造JDBC代码去完成。

其次写V2.0版本：是以面向过程的思维方式去实现的，（一个类去完成）。

其次写V3.0版本：是以面向对象的思维方式去实现的，（就是模仿Mybatis的类去实现手写）。

V1.0版本思路

- 需求

根据用户信息，查询用户列表

- 将JDBC中的硬编码，写入properties文件中

```

1  db.driver=com.mysql.jdbc.Driver
2  db.url=jdbc:mysql://111.231.106.221:3306/kkb
3  db.username=kkb
4  db.password=kkb111111
5  db.sql.queryUserById=select * from user where username = ?
6  db.sql.queryUserById.parametertype=com.kkb.mybatis.po.User
7  db.sql.queryUserById.params=username
8  db.sql.queryUserById.resultclassname=com.kkb.mybatis.po.User
9  db.sql.queryOrderById=select * from user where username = ?

```

- 加载properties配置文件

```

1  private Properties properties = new Properties();
2
3  private final static String DRIVER = "db.driver";

```

```

4
5      /**
6       * 加载properties配置文件
7       */
8       public void loadProperties() {
9           try {
10               InputStream inputStream =
11               this.getClass().getClassLoader().getResourceAsStream("jdbc.properties");
12               properties.load(inputStream);
13           } catch (IOException e) {
14               e.printStackTrace();
15           }
16       }

```

- 执行JDBC查询

```

1      /**
2       * 执行JDBC代码，完成查询用户操作
3       *
4       * @return
5       */
6       public List<Object> selectList(String statementId, Object paramObject)
7       {
8           Connection connection = null;
9           PreparedStatement preparedStatement = null;
10          ResultSet rs = null;
11          // 要返回的结果集合
12          List<Object> results = new ArrayList<Object>();
13          try {
14              // 加载数据库驱动
15              Class.forName(properties.getProperty(DRIVER));
16
17              // 通过驱动管理类获取数据库链接
18              connection = DriverManager.getConnection(properties.getProperty("db.url"),
19              properties.getProperty("db.username"),
20              properties.getProperty("db.password"));
21
22              // 定义sql语句 ?表示占位符
23              String sql = properties.getProperty("db.sql." + statementId);
24
25              // 获取预处理 statement
26              preparedStatement = connection.prepareStatement(sql);
27
28              // 设置参数，第一个参数为 sql 语句中参数的序号（从 1 开始），第二个参数为
29              设置的
30
31              // 先获取入参类型
32              String parametertype = properties.getProperty("db.sql." +
33              statementId + ".parametertype");
34              Class<?> parameterTypeClass = Class.forName(parametertype);
35
36              // 先判断入参类型(8种基本类型、String类型、各种集合类型、自定义的Java类
37              型)
38
39              if (SimpleTypeRegistry.isSimpleType(parameterTypeClass)) {

```

```

34         preparedStatement.setObject(1, paramObject);
35     } else {
36         // 自定义的Java类型
37         // 此处需要遍历SQL语句中的参数列表
38         String params = properties.getProperty("db.sql." +
statementId + ".params");
39         String[] paramArray = params.split(",");
40         for (int i = 0; i < paramArray.length; i++) {
41             // 列名
42             String param = paramArray[i];
43
44             // 根据列名获取入参对象的属性值，前提：列名和属性名称要一致
45             Field field =
parameterTypeClass.getDeclaredField(param);
46             field.setAccessible(true);
47             // 获取属性值
48             Object value = field.get(paramObject);
49             preparedStatement.setObject(i + 1, value);
50         }
51     }
52
53     // 向数据库发出 sql 执行查询，查询出结果集
54     rs = preparedStatement.executeQuery();
55
56     // 获取要映射的结果类型
57     String resultclassname = properties.getProperty("db.sql." +
statementId + ".resultclassname");
58     Class<?> resultTypeClass = Class.forName(resultclassname);
59
60     Object result = null;
61     while (rs.next()) {
62         // 每一行对应一个对象
63         result = resultTypeClass.newInstance();
64
65         ResultSetMetaData metaData = rs.getMetaData();
66         int columnCount = metaData.getColumnCount();
67
68         for (int i = 1; i <= columnCount; i++) {
69             // 获取结果集中列的名称
70             String columnName = metaData.getColumnName(i);
71
72             Field field =
resultTypeClass.getDeclaredField(columnName);
73             field.setAccessible(true);
74             field.set(result, rs.getObject(columnName));
75
76         }
77
78         results.add(result);
79     }
80
81     } catch (Exception e) {
82         e.printStackTrace();
83     } finally {
84         // 释放资源
85         if (rs != null) {
86             try {
87                 rs.close();

```

```
88         } catch (SQLException e) {
89             e.printStackTrace();
90         }
91     }
92     if (preparedStatement != null) {
93         try {
94             preparedStatement.close();
95         } catch (SQLException e) {
96             e.printStackTrace();
97         }
98     }
99     if (connection != null) {
100         try {
101             connection.close();
102         } catch (SQLException e) {
103             // TODO Auto-generated catch block
104             e.printStackTrace();
105         }
106     }
107     return results;
108 }
```

- 测试

```
1  @Test
2  public void test() {
3      loadProperties();
4
5      User user = new User();
6      user.setUsername("王五");
7      List<Object> list = selectList("queryUserById", user);
8      System.out.println(list);
9  }
```