使用JDBC搭建miniORM框架

一、什么是ORM框架

- ORM（Object Relational Mapping）对象关系映射
  - 广义上：ORM指的是面向对象的对象模型和关系型数据库的数据结构之间的相互转换。
  - 狭义上：ORM可以被认为是，基于关系型数据库的数据存储，实现一个虚拟的面向对象的数据访问接口。理想情况下，基于这样一个面向对象的接口，持久化一个OO对象应该不需要要了解任何关系型数据库存储数据的实现细节。
- ORM思想：表实体和数据表的相互转化——使得Java程序员不需要面对SQL编程，只需要面对对象OO编程。
  - 操作（1）：把表实体的变化转化到数据库里面（把Object映射为一条记录）
- 操作（2）：把表数据转成表实体（把Table Row数据映射为一个Object）

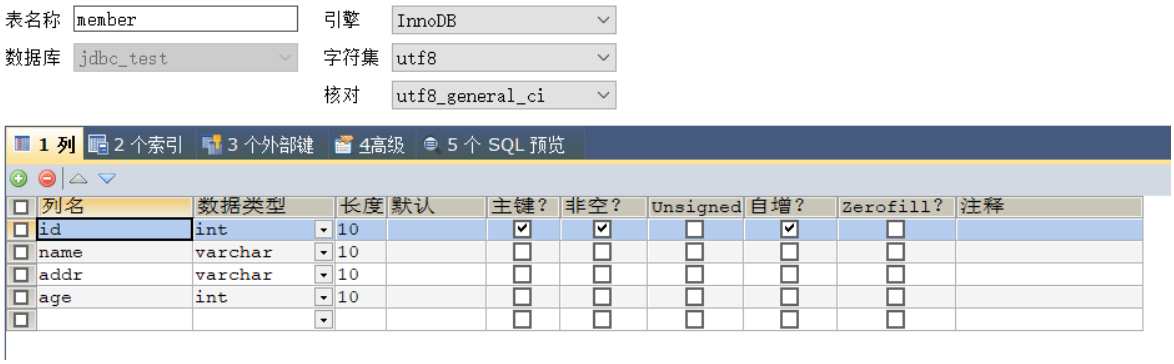
二、ORM框架的优缺点

- 优点：
  1. 提高开发效率，降低开发成本
  2. 使开发更加对象化
  3. 可移植
  4. 可以很方便地引入数据缓存之类的附加功能
- 缺点：
  1. 自动化进行关系数据库的映射需要消耗系统性能。其实这里的性能消耗还好啦，一般来说都可以忽略之。
  2. 在处理多表联查、where条件复杂之类的查询时，ORM的语法会变得复杂。

三、常见的ORM框架

- Hibernate、JPA
  - 全自动框架；如同生活中的全自动汽车，门槛低，易上手。
- 并发量比较低
- Mybatis
  - 半自动框架；如同生活中手自一体的汽车，适中，轻量级
  - 特点：SQL语句写于mapper.xml中
  - 并发量上高了，适合使用Mybatis
- Spring JDBC
  - 全手动框架；如同生活中全手动挡的汽车，门槛高
  - 性能高

四、数据库表结构



五、实体类

```
@Entity
@Table(name="member")
@Data
public class Member {
    @Id private Integer id;
    @Column(name = "name") private String name;
    private String addr;
    private Integer age;
}
```

- **@Table** 注解中的name属性需要与数据库中的表名相对应
- **@Column**注解中的name属性需要与数据库中的字段名相匹配
    - 实体类中的属性名需要与数据库中的字段名相匹配
        - 当数据库字段出现user_name带下划线的字段名时，属性名应将"_"后的第一个字母大写。并使用**@Column注解映射数据库中的字段名**，此时@Column注解中的name属性应亦改为"user_name"。
    - 实体类中的属性类型需要与数据库汇总的字段属性相匹配，匹配规则如下：

Java数据类型	Hibernate数据类型	标准SQL数据类型 (PS:对于不同的DB可能有所差异)
byte、java.lang.Byte	byte	TINYINT
short、java.lang.Short	short	SMALLINT
int、java.lang.Integer	integer	INGEGER
long、java.lang.Long	long	BIGINT
float、java.lang.Float	float	FLOAT
double、java.lang.Double	double	DOUBLE
java.math.BigDecimal	big_decimal	NUMERIC
char、java.lang.Character	character	CHAR(1)
boolean、java.lang.Boolean	boolean	BIT
java.lang.String	string	VARCHAR
boolean、java.lang.Boolean	yes_no	CHAR(1)(‘Y’或‘N’)
boolean、java.lang.Boolean	true_false	CHAR(1)(‘Y’或‘N’)
java.util.Date、java.sql.Date	date	DATE
java.util.Date、java.sql.Time	time	TIME
java.util.Date、java.sql.Timestamp	timestamp	TIMESTAMP
java.util.Calendar	calendar	TIMESTAMP
java.util.Calendar	calendar_date	DATE
byte[]	binary	VARBINARY、BLOB
java.lang.String	text	CLOB
java.io.Serializable	serializable	VARBINARY、BLOB
java.sql.Clob	clob	CLOB
java.sql.Blob	blob	BLOB
java.lang.Class	class	VARCHAR
java.util.Locale	locale	VARCHAR
java.util.TimeZone	timezone	VARCHAR

Java数据类型	Hibernate数据类型	标准SQL数据类型 (PS:对于不同的DB可能有所差异)
java.util.Currency	currency	VARCHAR

六、通过SQL语句进行查询

（一）、查询方法

```
 private static List<Member> select(String sql) {
        List<Member> result = new ArrayList<>();
        Connection con = null;          //连接对象
        PreparedStatement pstm = null;  //语句集
        ResultSet rs = null;            //结果集
        try {
            //1、加载驱动类，千万不要忘记了
            Class.forName("com.mysql.jdbc.Driver");
            //2、建立连接
            con =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/jdbc_test" +
                            "?
useUnicode=true&useJDBCCompliantTimezoneShift=true" +
                            "&useLegacyDatetimeCode=false&serverTimezone=UTC",
                    "root", "root");
            //3、创建语句集
            pstm = con.prepareStatement(sql);
            //4、执行语句集
            rs = pstm.executeQuery();
            while (rs.next()) {
                //纯粹的硬编码
                Member instance = new Member();
                instance.setId(rs.getInt("id"));
                instance.setName(rs.getString("name"));
                instance.setAge(rs.getInt("age"));
                instance.setAddr(rs.getString("addr"));
                result.add(instance);
            }
            //5、获取结果集
        } catch (Exception e) {
            e.printStackTrace();
        }
        //6、关闭结果集、关闭语句集、关闭连接
        finally {
            try {
                rs.close();
                pstm.close();
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return result;
    }
```

- con = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/jdbc_test" +
  "?useUnicode=true&useJDBCCompliantTimezoneShift=true" +
  "&useLegacyDatetimeCode=false&serverTimezone=UTC",
  "root", "root");
  - 该语句中"?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC"为Mysql8版本需要添加。
  - 数据库名以及密码需要改为自己相对应的数据库名称与密码

（二）、调用查询方法

```
//ORM，完成了一部分，只完成了从数据表到对象的映射
//对象到数据库表还没有
//我传的条件是一条SQL语句，我还是在面向SQL编程
List<Member> result = select("select * from member");
System.out.println(result);
```

（三）、查询结果

```
[Member(id=1, name=Tom, addr=gd, age=18), Member(id=2, name=Mike, addr=gd, age=18), Member(id=3, name=Jack, addr=gd, age=18)]
```

七、以上代码存在的问题

- 上述代码中，可以发现以下问题：
  - 存在可重用代码，需提取出
  - 未面向OO编程
  - 无法解决数据库字段名与实体类属性名不相符的情况

八、优化一：提取公共部分

（一）、单独通过一个方法进行赋值

```
private static Member mapperRow(ResultSet rs, int i) throws Exception {
        Member instance = new Member();
        instance.setId(rs.getInt("id"));
        instance.setName(rs.getString("name"));
        instance.setAge(rs.getInt("age"));
        instance.setAddr(rs.getString("addr"));
        return instance;
    }
```

（二）、优化后的代码

```
 private static List<Member> select(String sql) {
        List<Member> result = new ArrayList<>();
        Connection con = null;
        PreparedStatement pstm = null;
        ResultSet rs = null;
        try {
            //1、加载驱动类
```

```
            Class.forName("com.mysql.jdbc.Driver");
            //2、建立连接
            con =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/jdbc_test" +
                                "?
useUnicode=true&useJDBCCompliantTimezoneShift=true" +
                                "&useLegacyDatetimeCode=false&serverTimezone=UTC",
                        "root", "root");
            //3、创建语句集
            pstm = con.prepareStatement(sql);
            //4、执行语句集
            rs = pstm.executeQuery();
            while (rs.next()) {
                Member instance = mapperRow(rs, rs.getRow());
                result.add(instance);
            }
            //5、获取结果集
        } catch (Exception e) {
            e.printStackTrace();
        }
        //6、关闭结果集、关闭语句集、关闭连接
        finally {
            try {
                rs.close();
                pstm.close();
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return result;
    }
```

九、优化二：面向OO编程

（一）、查询方法

```
private static List<?> select(Object condition) {
        List<Object> result = new ArrayList<>();

        Class<?> entityClass = condition.getClass();


        Connection con = null;          //连接对象
        PreparedStatement pstm = null;  //语句集
        ResultSet rs = null;            //结果集


        try {
            //1、加载驱动类，千万不要忘记了
            Class.forName("com.mysql.jdbc.Driver");
            //2、建立连接
            con =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/jdbc_test" +
                                "?
useUnicode=true&useJDBCCompliantTimezoneShift=true" +
```

```
                            "&useLegacyDatetimeCode=false&serverTimezone=UTC",
                    "root", "root");

            Field[] fields = entityClass.getDeclaredFields();

            StringBuffer sql = new StringBuffer();
            //3、创建语句集
            Table table = entityClass.getAnnotation(Table.class);
            sql.append("select * from " + table.name() + " where 1=1 ");
            for (Field field : fields) {
                field.setAccessible(true);
                Object value = field.get(condition);
                if (null != value) {
                    if (String.class == field.getType()) {
                        sql.append(" and " + field.getName() + " = '" + value +
"'");
                    } else {
                        sql.append(" and " + field.getName() + " = " + value);
                    }
                    //其他依次类推
                }
            }

            pstm = con.prepareStatement(sql.toString());

            //4、执行，获取结果集
            rs = pstm.executeQuery();

            int columnCounts = rs.getMetaData().getColumnCount();
            while (rs.next()) {
                Object instance = entityClass.newInstance();
                for (int i = 1; i <= columnCounts; i++) {
                    String columnName = rs.getMetaData().getColumnName(i);
                    Field field = entityClass.getDeclaredField(columnName);
                    field.setAccessible(true);
                    field.set(instance, rs.getObject(columnName));
                }
                result.add(instance);
            }


        } catch (Exception e) {
            e.printStackTrace();
        }
        //6、关闭结果集、关闭语句集、关闭连接
        finally {
            try {
                rs.close();
                pstm.close();
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return result;
    }
```

（二）、调用查询方法

1、查询所有

```
Member condition = new Member();
        List<?> result = select(condition);
        System.out.println(Arrays.toString(result.toArray()));
```

2、设置条件查询

```
Member condition = new Member();
            condition.setName("Tom");
        List<?> result = select(condition);
        System.out.println(Arrays.toString(result.toArray()));
```

（三）、查询结果

1、查询所有结果

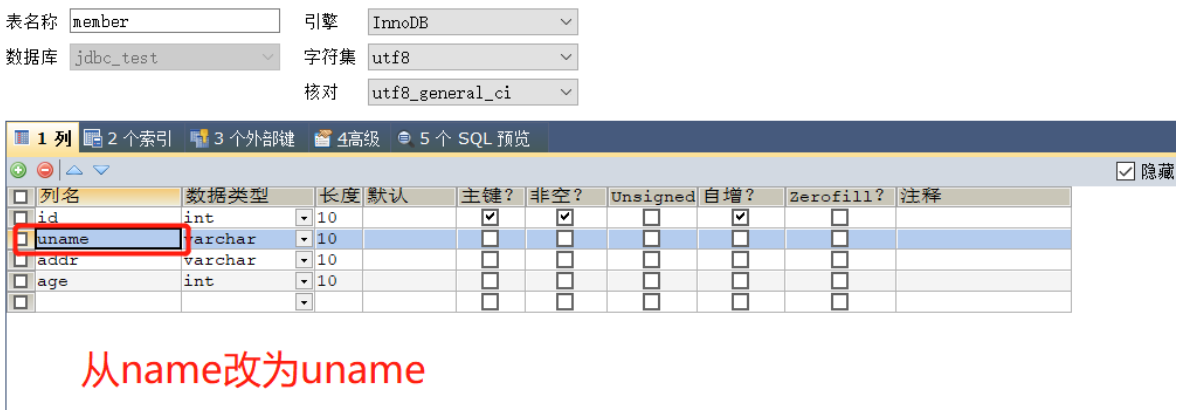
```
[Member(id=1, name=Tom, addr=gd, age=18), Member(id=2, name=Mike, addr=gd, age=18), Member(id=3, name=Jack, addr=gd, age=18)]
```

2、设置条件查询结果

```
[Member(id=1, name=Tom, addr=gd, age=18)]
```

十、优化三：解决数据库字段名与实体类属性名不相符的情况

（一）、新的数据库字段名



（二）、修改实体类注解

```
@Entity
@Table(name="member")
@Data
public class Member {
    @Id private Integer id;
    // 修改@Column注解的name属性为uname
    @Column(name = "uname") private String name;
    private String addr;
    private Integer age;
}
```

(三)、查询方法

```
 private static List<?> select(Object condition) {
        List<Object> result = new ArrayList<>();

        Class<?> entityClass = condition.getClass();


        Connection con = null;          //连接对象
        PreparedStatement pstm = null;  //语句集
        ResultSet rs = null;            //结果集


        try {
            //1、加载驱动类，千万不要忘记了
            Class.forName("com.mysql.jdbc.Driver");
            //2、建立连接
            con =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/jdbc_test" +
                            "?
useUnicode=true&useJDBCCompliantTimezoneShift=true" +
                            "&useLegacyDatetimeCode=false&serverTimezone=UTC",
                    "root", "root");

            Map<String, String> getFieldNameByColumn = new HashMap<String,
String>();
            Map<String, String> getColumnByFieldName = new HashMap<String,
String>();
            Field[] fields = entityClass.getDeclaredFields();
            for (Field field : fields) {
                field.setAccessible(true);
                String fieldName = field.getName();
                if (field.isAnnotationPresent(Column.class)) {
                    Column column = field.getAnnotation(Column.class);
                    String columnName = column.name();
                    getFieldNameByColumn.put(columnName, fieldName);
                    getColumnByFieldName.put(fieldName, columnName);
                } else {
                    //默认属性名就是列名
                    getFieldNameByColumn.put(fieldName, fieldName);
                    getColumnByFieldName.put(fieldName, fieldName);
                }
            }
```

```
            StringBuffer sql = new StringBuffer();
            //3、创建语句集
            Table table = entityClass.getAnnotation(Table.class);
            sql.append("select * from " + table.name() + " where 1=1 ");
            for (Field field : fields) {

                Object value = field.get(condition);
                if (null != value) {
                    if (String.class == field.getType()) {
                        sql.append(" and " +
getColumnByFieldName.get(field.getName()) + " = '" + value + "'");
                    } else {
                        sql.append(" and " +
getColumnByFieldName.get(field.getName()) + " = " + value);
                    }
                    //其他依次类推
                }
            }

            pstm = con.prepareStatement(sql.toString());

            //4、执行，获取结果集
            rs = pstm.executeQuery();

            int columnCounts = rs.getMetaData().getColumnCount();
            while (rs.next()) {
                Object instance = entityClass.newInstance();
                for (int i = 1; i <= columnCounts; i++) {
                    String columnName = rs.getMetaData().getColumnName(i);
                    Field field =
entityClass.getDeclaredField(getFieldNameByColumn.get(columnName));
                    field.setAccessible(true);
                    field.set(instance, rs.getObject(columnName));
                }
                result.add(instance);
            }


        } catch (Exception e) {
            e.printStackTrace();
        }
        //6、关闭结果集、关闭语句集、关闭连接
        finally {
            try {
                rs.close();
                pstm.close();
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return result;
    }
```

（四）、调用查询方法

1、查询所有

```
Member condition = new Member();
        List<?> result = select(condition);
        System.out.println(Arrays.toString(result.toArray()));
```

2、设置条件查询

```
Member condition = new Member();
            condition.setName("Mike");
        List<?> result = select(condition);
        System.out.println(Arrays.toString(result.toArray()));
```

（五）、查询结果

1、查询所有结果

```
[Member(id=1, name=Tom, addr=gd, age=18), Member(id=2, name=Mike, addr=gd, age=18), Member(id=3, name=Jack, addr=gd, age=18)]
```

2、设置条件查询结果

```
[Member(id=2, name=Mike, addr=gd, age=18)]
```