# typeHandler使用

数据表：

|  |
| --- |
| CREATE TABLE `admin` (  `id` bigint NOT NULL AUTO\_INCREMENT,  `username` varchar(255) NOT NULL,  `password` varchar(255) NOT NULL,  `sex` int DEFAULT NULL,  PRIMARY KEY (`id`)  ) |

枚举定义

|  |
| --- |
| */\*\*  \* 性别枚举  \*/* public enum EnumSex {  *FEMALE*(0, "女"),  *MALE*(1, "男");  */\*\*  \* 编码  \*/* private int code;  */\*\*  \* 显示的名称  \*/* private String name;  EnumSex(int code, String name) {  this.code = code;  this.name = name;  }  public int getCode() {  return code;  }  public String getName() {  return name;  }  public static EnumSex getByCode(int code) {  *//获取所有的枚举值（枚举数组）* EnumSex[] values = EnumSex.*values*();  *//遍历数组，查找每一个枚举的code是否与传进来的参数code相等，如果相等就返回对应的枚举* for (EnumSex value : values) {  if (value.getCode() == code) {  return value;  }  }  return null;  } } |

自定义typeHandler处理器

|  |
| --- |
| */\*\*  \* 自定义性别枚举处理器  \*/* public class EnumSexTypeHandler implements TypeHandler<EnumSex> {  */\*\*  \* javaType 转换 jdbcType 参数设置调用  \*  \* @param preparedStatement  \* @param i  \* @param enumSex  \* @param jdbcType  \* @throws SQLException  \*/* @Override  public void setParameter(PreparedStatement preparedStatement, int i, EnumSex enumSex, JdbcType jdbcType) throws SQLException {  *//增加和修改 sex字段 用到jdbc参数设置* preparedStatement.setObject(i, enumSex.getCode());  }   */\*\*  \* jdbctye 转 javatype 用户查询  \*  \* @param resultSet  \* @param s  \* @return  \* @throws SQLException  \*/* @Override  public EnumSex getResult(ResultSet resultSet, String s) throws SQLException {  *//根据columnLabel值 获取对应结果集的值 sex 对应code* int sexCode = resultSet.getInt(s);  *//把sexCode转换成枚举值 返回出去* return EnumSex.*getByCode*(sexCode);  }   @Override  public EnumSex getResult(ResultSet resultSet, int i) throws SQLException {  *//根据columnIndex值 获取对应结果集的值 sex 对应code* int sexCode = resultSet.getInt(i);  *//把sexCode转换成枚举值 返回出去* return EnumSex.*getByCode*(sexCode);  }   */\*\*  \* 存储过程调用  \*  \* @param callableStatement  \* @param i  \* @return  \* @throws SQLException  \*/* @Override  public EnumSex getResult(CallableStatement callableStatement, int i) throws SQLException {  int sexCode = callableStatement.getInt(i);  *//把sexCode转换成枚举值 返回出去* return EnumSex.*getByCode*(sexCode);  } } |

Admin实体类

|  |
| --- |
| public class Admin {  private Long id;  private String username;  private String password;  private EnumSex sex;   public Admin() {  }   public Admin(String username, String password, EnumSex sex) {  this.username = username;  this.password = password;  this.sex = sex;  }   public Long getId() {  return id;  }   public void setId(Long id) {  this.id = id;  }   public String getUsername() {  return username;  }   public void setUsername(String username) {  this.username = username;  }   public String getPassword() {  return password;  }   public void setPassword(String password) {  this.password = password;  }   public EnumSex getSex() {  return sex;  }   public void setSex(EnumSex sex) {  this.sex = sex;  }   public Admin(Long id, String username, String password, EnumSex sex) {  this.id = id;  this.username = username;  this.password = password;  this.sex = sex;  }   @Override  public String toString() {  return "Admin{" +  "id=" + id +  ", username='" + username + '\'' +  ", password='" + password + '\'' +  ", sex=" + sex +  '}';  } } |

AdminMapper接口

|  |
| --- |
| public interface AdminMapper {  void insert(Admin admin);   Admin selectById(Long id); } |

AdminMapper.xml映射文件

|  |
| --- |
| *<?*xml version="1.0" encoding="utf-8" *?>* <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" *>* <mapper namespace="com.tjetc.dao.AdminMapper">  <insert id="insert" parameterType="admin" useGeneratedKeys="true" keyProperty="id">  insert into admin(username,password,sex) values(#{username},#{password},#{sex,typeHandler=com.tjetc.handler.EnumSexTypeHandler})  </insert>  <select id="selectById" parameterType="long" resultMap="adminMap">  select id,username,password,sex from admin where id=#{id}  </select>  <resultMap id="adminMap" type="admin">  <id column="id" property="id"></id>  <result column="username" property="username"></result>  <result column="password" property="password"></result>  <result column="sex" property="sex" typeHandler="com.tjetc.handler.EnumSexTypeHandler"></result>  </resultMap> </mapper> |

Mybatis.xml增加配置

|  |
| --- |
| *<!--配置自定义typeHandler-->* <typeHandlers>  *<!--<package name=""/>-->* <typeHandler handler="com.tjetc.handler.EnumSexTypeHandler"/> </typeHandlers> |

测试：

|  |
| --- |
| public class TestAdmin {  @Test  public void testInsert() throws IOException {  SqlSession sqlSession = SqlSessionUtil.*getSqlSession*();  AdminMapper mapper = sqlSession.getMapper(AdminMapper.class);  Admin admin = new Admin("wangwu", "5555", EnumSex.*FEMALE*);  mapper.insert(admin);  sqlSession.commit();  System.*out*.println(admin);  sqlSession.close();  }   @Test  public void testSelectById() throws IOException {  SqlSession sqlSession = SqlSessionUtil.*getSqlSession*();  AdminMapper mapper = sqlSession.getMapper(AdminMapper.class);  Admin admin = mapper.selectById(4L);  System.*out*.println(admin);  sqlSession.close();  } } |

# MyBatis分页插件PageHelper

|  |
| --- |
| *<!--mybatis的分页插件-->* <**dependency**>  <**groupId**>com.github.pagehelper</**groupId**>  <**artifactId**>pagehelper</**artifactId**>  <**version**>5.1.9</**version**> </**dependency**> |
| <**plugins**>   *<!-- com.github.pagehelper为PageHelper类所在包名 -->* <**plugin interceptor="com.github.pagehelper.PageInterceptor"**>   *<!-- 什么都不配，使用默认的配置 -->* </**plugin**>  </**plugins**> |
| **public void** testFindAll2() {  **try** { *// 创建SqlSessionFactory* SqlSessionFactory sqlSessionFactory = **new** SqlSessionFactoryBuilder().build(Resources.*getResourceAsReader*(**"mybatis.xml"**)); *// 得到session对象* SqlSession session = sqlSessionFactory.openSession();  *//得到mapper接口的实现对象* UserMapper mapper = session.getMapper(UserMapper.**class**);  PageHelper.*startPage*(2, 2); *// 通过session对象操作数据库* List<User> list = mapper.findAll();  PageInfo<User> pageInfo = **new** PageInfo<>(list); *// 控制台打印list对象* **for** (User user : list) {  System.***out***.println(user);  } *// 关闭session* session.close();  } **catch** (IOException e) {  e.printStackTrace();  }  } |