湖南科技大学计算机科学与工程学院

数据库系统课程设计报告

专业班级： _____

姓　　名： _____

学　　号： _____

指导教师： _____

时　　间： 2024.11.11-2024.11.22

地　　点： 逸夫楼 535

指导教师评语： **成绩：　　等级：** **签名：** _____ 年　月　日

一、课程设计题目

员工培训管理系统

二、课程设计目的

企业的不断学习能够帮助企业更快地适应市场环境的飞速变化，通过有效地培训企业员工，赋予员工学习专业技能的机会与能力，企业可以迅速根据市场需求变化，调整分配企业组织的人力资源分布，形成高效的企业组织单元，更好地完成企业运作任务。培训已经成为企业现代化的重要标志。

本课程设计项目的目的就是基于以上的现状，开发一款员工培训管理系统软件，实现员工培训的管理。

三、总体设计（含背景知识或基本原理与算法、或模块介绍、设计步骤等）

基本技术说明：

本项目使用 Java 开发，使用 Swing 实现用户界面，需要使用 Java 17 或更新版本运行。
本项目的数据库共有 9 张表，项目有约 3600 行 Java 代码，49 个 Java 源文件。
本项目采用 Gradle 构建系统，使用 MyBatis 来访问数据库，使用国产 Excel 库 EasyExcel 来实现报表导出功能。
本项目前端的一些关键功能使用到了反射和注解技术（位于 EntityUtils 类中），使用了 Apache Commons Lang 库来简化部分反射代码。
本项目的密码验证功能使用了 Apache Commons Codec 来提供加密算法支持。
本项目使用的数据库软件为 MySQL 8.4。

基本功能特性说明：

1. 简洁

本项目在整体上呈现了比较简洁美观的界面，各种关键功能都配有快捷键来提高效率，界面导航逻辑清晰。

2. 易用

软件的设计充分考虑用户体验，例如管理员界面中每个页面的数据表都支持快捷搜索功能，且部分页面的搜索功能支持同时搜索多个属性；数据的增删改功能都经过精心设计，其中所有增添/修改数据项的功能都会单独显示数据录入窗口，所有外键选项都不会让用户输入 ID 或其他标识符，而是弹出选择窗口让用户选择所需选项（也可以选择“未设置”，即设为 NULL），既符合用户需求，又避免了输入错误带来的麻烦。
在部分仅用于为用户提供选择功能的数据表视图中，不重要的数据属性列会被隐藏

3. 健壮

在数据录入界面中，各个必要的位置都配备数据范取值检验功能，阻止用户输入非法数据。此外，软件经过了相当充分的测试，修复了出现过的所有 bug，可以保证可靠运行。

4. 安全

本软件可以避免用户非法输入导致的 SQL 注入攻击。本软件的所有 SQL 调用都使用占位符来填充用户输入，可以有效阻止 SQL 注入攻击。此外，软件中所有用户的登录密码都使用了 SHA-512 摘要算法进行单向加密，数据库中不会明文存储用户的密码，这有效防止了用户密码的泄露。

模块介绍和功能演示：

1. （共用模块）登录、密码验证相关功能

应用启动后，显示登录对话框（图 1），用户输入账号和密码后，按 Enter 键或点击登录按钮来登录。
若输入账号不存在或密码错误，则弹出错误提示（图 2）。
若输入账号密码正确，则根据登录用户身份启动对应的系统界面（管理员用户启动管理员界面，普通用户启动普通用户个人管理界面）。

图 1 登录对话框

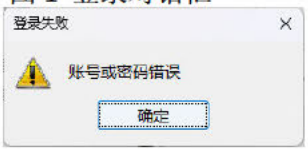


图 2 登录失败的提示

2. （管理员）课程管理界面
课程管理界面用于管理课程，顶部按钮用于切换页面，底部搜索框可以用键盘快捷键快速激活和进行搜索，双击列表中的一项可进入数据编辑界面。
所有的外键选项（此处为任课教师和课程状态）的编辑方式都是在列表中选择，十分方便。

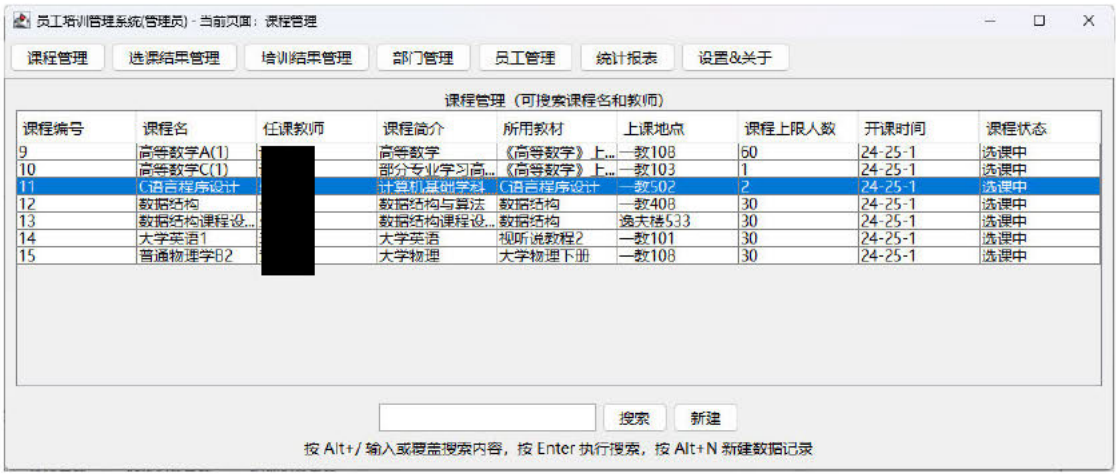
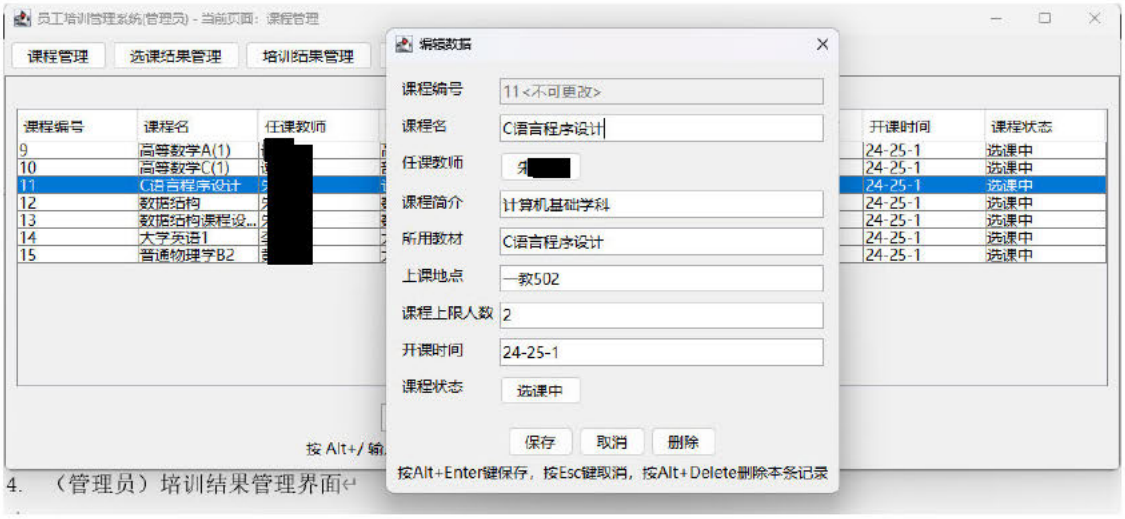


图 3 课程管理界面



4. （管理员）培训结果管理界面↵

图 4 课程编辑界面

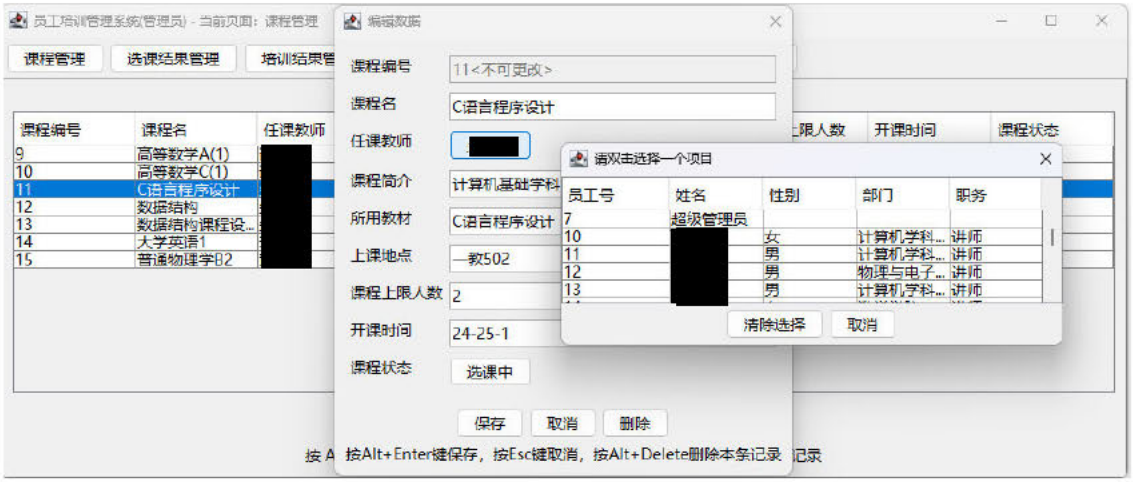


图 5 任课教师选择界面

3. （管理员）选课结果管理界面

选课结果管理界面中，列出了所有非管理员普通用户的列表，双击其中一个用户可以查看和修改其选课信息。该界面会对用户输入进行检验，若课程选课人数已满，或者课程不是选课中状态，则不允许选课。

图 6 选课结果管理界面

图 7 查看选课信息

图 8 添加选课

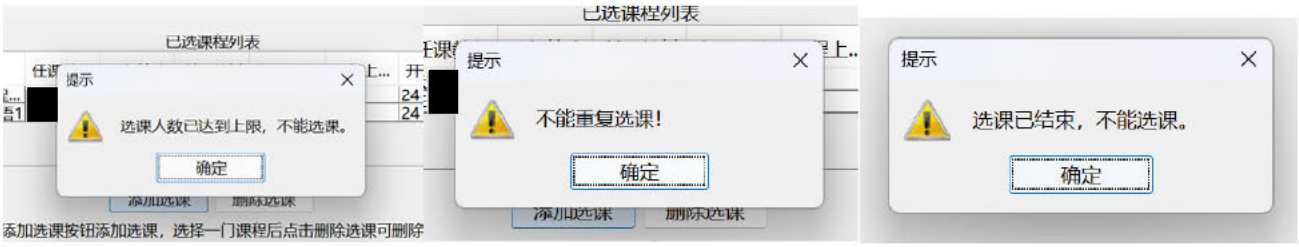


图 9 用户输入校验提示

4. （管理员）培训结果管理界面

培训结果管理界面中，可以查看和管理所有员工的课程成绩。该界面支持按照员工名或课程名搜索。录入成绩时，需要选择课程，此处的列表中只会列出员工已经选择的课程，不允许为员工未选择的课程录入成绩。

图 10 培训结果管理界面

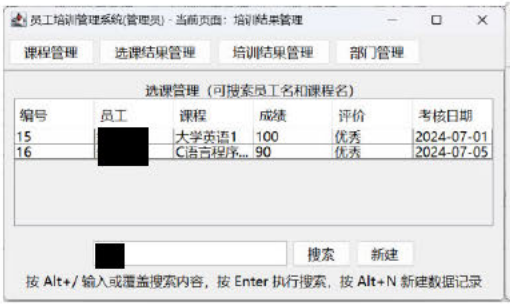


图 11 搜索员工名

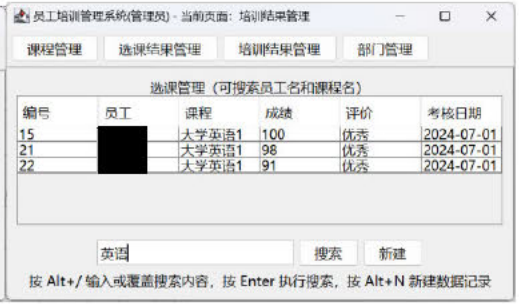


图 12 搜索课程名

图 13 成绩录入

5. （管理员）部门管理界面

部门管理界面可以查找、删除、更改部门信息。

图 14 部门管理界面

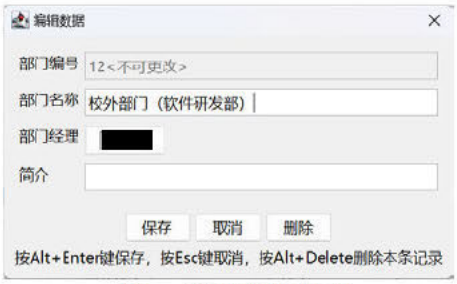


图 15 编辑部门信息

6. （管理员）员工管理界面

员工管理界面用于对员工信息进行增删改查。

图 16 员工管理界面

图 17 员工信息编辑界面

7. （管理员）统计报表导出功能

统计报表导出功能用于导出统计报表，点击导出按钮后，程序会弹出窗口让用户选择报表保存位置，然后保存成绩报表。

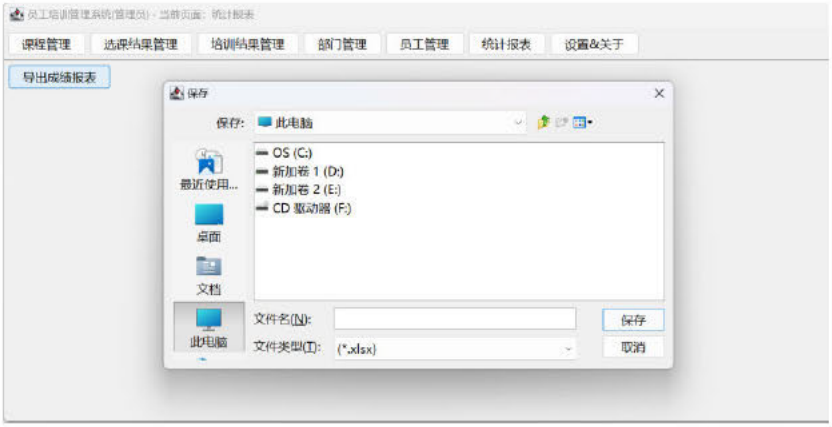


图 18 导出报表功能

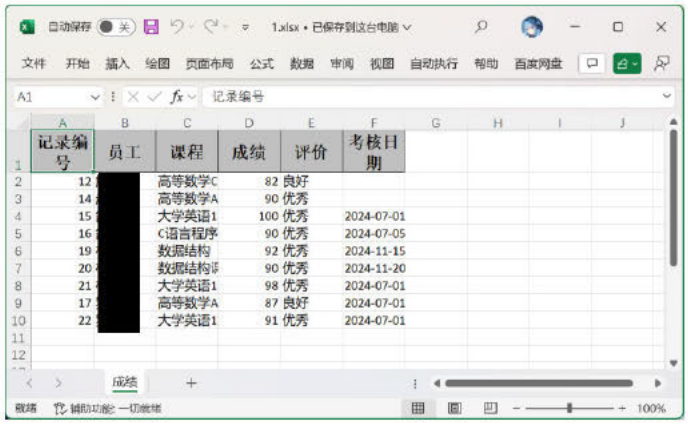


图 19 导出后的成绩报表

8. （普通用户）个人信息查看和修改界面

以登录普通员工账号后，进入普通员工管理界面，第一页是个人信息查看和修改页面。该界面中，用户可以修改自己的个人信息，也可以点击“重置”撤销更改。

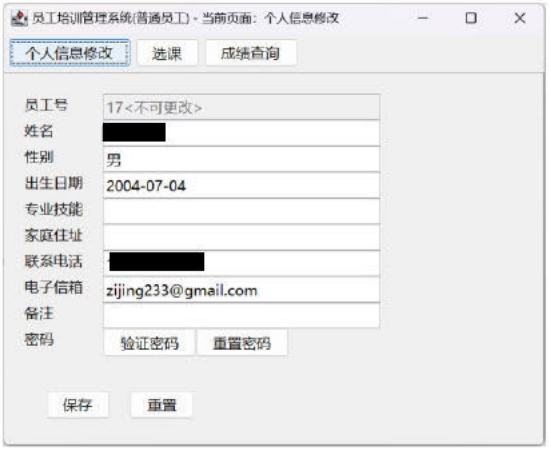


图 20 个人信息修改页面

9. （普通用户）个人选课页面

个人选课页面中，用户可以查看和修改自己的选课信息。和管理员操作选课一样，系统会对选课人数、课程状态进行检查，在不符合选课条件时拒绝操作。

图 21 个人选课页面

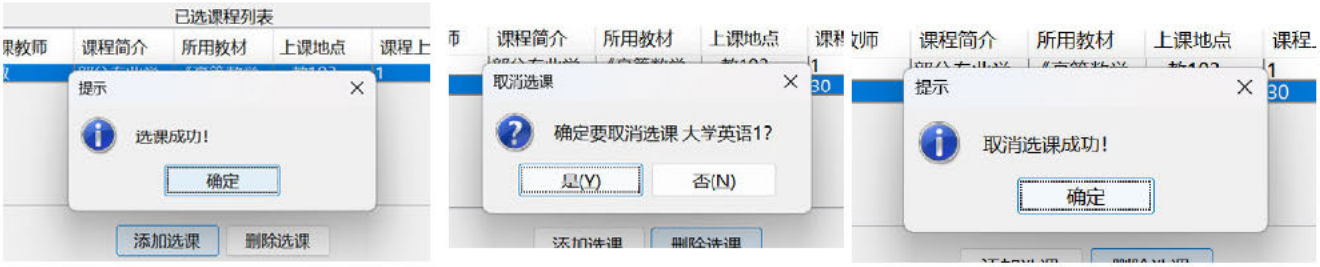


图 22-24 选课功能的一些提示

10. （普通用户）个人成绩查询界面

在个人成绩查询界面，用户可以查询自己各个科目的成绩。若用户选的某门课程没有录入成绩，则不会显示。

图 25 成绩查询界面

11. （共用模块）通用的数据记录编辑对话框

该系统中所有的数据编辑界面使用的是同一个界面，在构造方法中，程序用反射技术自动解析相应实体类的字段及其类型，并动态生成界面结构。

这种对话框的示例可以在图 4、图 13、图 15、图 17 看到。

12. （共用模块）通用的对象选择对话框

该系统的数据记录编辑对话框使用的是同一个界面，程序使用反射技术解析实体类的字段，并生成列表供用户选择，窗口会将用户选择的选项传递给父窗口。

由于这个对话框中的列表仅用于选择，表格中部分不重要的列会隐藏，实体类中带有@KeyColumn 的字段为在这里需要展示的字段。

这种对话框的示例可以在图 5 看到。

设计步骤：

1. 阅读课程设计题目，了解功能需求，划分数据表。
2. 建立数据库的模式，编写基本程序，测试数据库连通性。
3. 建立所有数据表，设置各个数据表的主键约束和外键约束；设计管理员角色的基本界面，实现基本查询功能；实现搜索功能。
4. 设计数据修改界面，实现数据修改功能，然后在此基础上实现数据的插入功能。
5. 在之前编写的基本界面基础上，快速实现管理员角色其他功能的基本界面。
6. 实现密码加密和验证功能，并且设计登录界面，实现登录功能，准备开始设计员工角色界面。
7. 实现员工界面的基本功能。
8. 实现选课和成绩管理两个功能。
9. 实现管理员角色的成绩报表导出功能。
10. 向数据库中填充足够多的测试数据，对软件进行测试，解决出现的所有问题。

四、详细设计（含主要的数据结构、程序流程图、关键代码等）

本项目数据库的 E-R 图（简图，省略了成绩评价、用户权限、课程状态、人员状态四个代码表）：

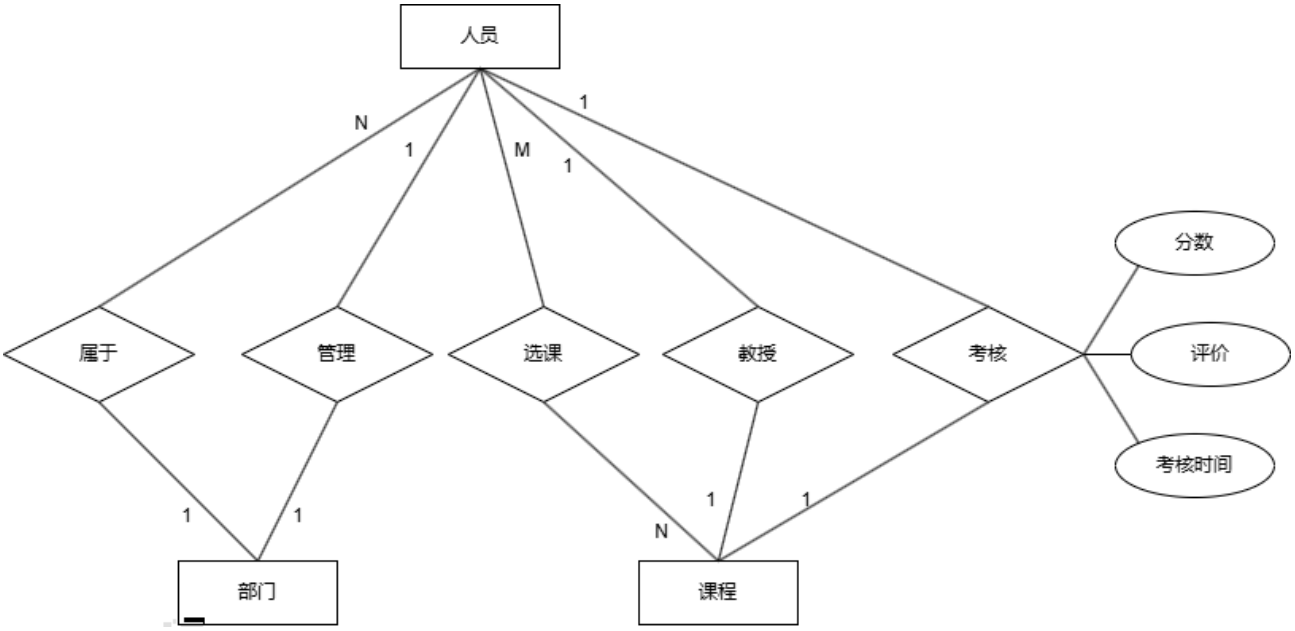


图 26 本系统的 E-R 图

程序运行关键流程和原理说明：

一、程序运行基本流程说明：

1. 程序启动，对界面样式进行初始化设置，包括更换界面主题为 Windows 主题、设置全局字体（此时还未显示任何界面）。
2. 启动登录界面，要求用户输入用户名和密码。
3. 用户点击“登录”后，对登录名和密码进行验证，若输入正确，则根据登录用户权限启动对应的管理界面，登录窗口自动关闭；若输入错误，弹窗提示用户登录失败，然后要求用户重新输入。
4. 进入主界面。

二、数据库访问技术说明：

本项目使用 MyBatis 来对 JDBC 功能进行封装，通过 MyBatis 来连接和访问数据库。

项目中使用 MyBatisUtils 的 getSqlSession()静态方法来获取 SqlSession，关键代码如下：

```
public class MyBatisUtils {
    private static final SqlSessionFactory sqlSessionFactory;

    static {
        try {
            String resource = "kitra/employeetraining/mybatis_config.xml";
            InputStream inputStream = Resources.getResourceAsStream(resource);
            sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);
        } catch (IOException e) { throw new RuntimeException(e); }
    }

    public static SqlSession getSqlSession() {
        return sqlSessionFactory.openSession();
    }
}
```

在这里，sqlSessionFactory 遵循**单例模式**，它由类的静态块初始化，不能对其进行修改，这保证了 sqlSessionFactory 只会被实例化一次，且在整个程序中是唯一的。

MyBatis 配置文件如下（简略版）：

kitra/employeetraining/mybatis_config.xml

```
...
<configuration>
    <!--引入外部配置文件-->
    <properties resource="db.properties"/>
    <!—使用控制台输出记录日志-->
    <settings>
        <setting name="logImpl" value="STDOUT_LOGGING"/>
    </settings>
    <!—实体类所在的包-->
    <typeAliases>
        <package name="kitra.employeetraining.common.datamodel"/>
    </typeAliases>
    ...
    <!—DAO 类所在的包-->
    <mappers>
        <package name="kitra.employeetraining.common.dao"/>
    </mappers>
</configuration>
```

本项目使用注解来定义 MyBatis 增删改查语句。

访问数据库的实际代码示例，位于“登录功能说明”一节中。

二、登录功能说明：

本软件的身份验证相关代码位于 kitra.employeetraining.common.auth 包中的 PasswordVerify 类中，下面解释的功能都对应这个类中的代码。

这个类是静态工具类，供外部调用的方法主要有两个：boolean verifyPassword(String, char[])，Authority getAuthority(int)和 void setPassword(int, String)。verifyPassword(String, char[])方法用于验证给定的用户名字符串和密码 char 数组是否与数据库中的账号密码信息对应;getAuthority(int)用于获取指定用户的权限信息，供登录程序判断登录的用户角色；setPassword(int, String)用于修改一个用户的密码。

前面提到，本软件不会明文存储密码。本软件的数据库中存储的密码为明文密码的 SHA-512 散列值。在验证密码时，也是计算密码的 SHA-512 散列值，并将其与数据库中的信息比较，返回验证成功或失败的结果。
相关关键代码：

```
public static boolean verifyPassword(String username, char[] password) {
    if(!hasAccount(username)) return false;
    try(SqlSession session = MyBatisUtils.getSqlSession()) {
        PersonDao mapper = session.getMapper(PersonDao.class);
        int userId = mapper.getByUserName(username).getId();
        String hash = mapper.getPasswordHash(userId);
        if(hash == null) return false;
        if(hash.isEmpty()) return true;
        return verifyHash(new String(password), hash);
    }
}
private static boolean verifyHash(String password, String hash) {
    return hash(password).equals(hash);
}
private static String hash(String data) {
    return DigestUtils.sha512Hex(data);
}
```

其中，DigestUtils 工具类由 Apache Commons Codec 库提供。

三、报表导出功能说明：

本项目的报表导出功能使用 EasyExcel 库实现。相关代码位于 kitra.employeetraining.manager.tools 包的 GradeExportTool 类中。程序的基本思路为，首先打开 Swing 提供的文件保存路径选择框，然后对文件路径进行处理，最后创建文件写入数据。
关键代码：

```
JFileChooser fileChooser = new JFileChooser();
FileNameExtensionFilter filter = new FileNameExtensionFilter("(*.xlsx)", "xlsx");
fileChooser.setFileFilter(filter);
int option = fileChooser.showSaveDialog(parent);
// 如果用户确认保存文件
if(option == JFileChooser.APPROVE_OPTION) {
    File file = fileChooser.getSelectedFile();
    ... // 处理文件扩展名的逻辑
    EasyExcel.write(file, GradeData.class).sheet("成绩").doWrite(getAllGradeData());
}
```

GradeData.class 是一个用 private static 修饰的内部类，用 EasyExcel 要求的格式定义了待输出表格的数据结构信息，其中，表头每一列的名称都通过在字段上添加的注解@ExcelProperty 来指定。
getAllGradeData()方法返回 List<GradeData>，用于从数据库中读取所有成绩信息，构造每条记录的 GradeData 实例并加到 List 中。

四、实体类结构解析功能说明：

前面提到，整个项目中所有的数据编辑窗口都采用了同一个界面，界面结构由传入的参数自动生成。此外，所有表格的列名也均为自动获取，没有在界面中用一个数组来手动指定。这得益于位于 kitra.employeetraining.common.util 包中的 EntityUtils 工具类。
EntityUtils 是一个用反射来获取数据实体各项信息的工具类，其主要功能包括从一个实体类获取到所有字段的显示名称（用于生成 JTable 表头）、从实体类获取到外键列表（用于生成数据编辑窗口结构）、从实体

类获取到哪些列是重要的（用于在外键选择窗口中显示精简后的表格）、将实体对象转换为字符串列表（用于在表格中显示数据）、将实体对象展开为 Object 数组（用于将数据编辑窗口中数据写回数据库）、通用的插入/更新/删除方法（传入实体类、实体对象和参数数组）。

为了降低反射带来的性能开销，EntityUtils 类中定义了多个 HashMap 静态字段，将一些方法的结果或中间结果存入 Map，在下次再次调用方法时直接从 Map 取出结果，避免重复调用。由于 Java 的类被加载后结构就不可改变，将和类的结构相关的信息缓存下来是提高反射性能的简单可靠的方法。

为了提高代码复用性，本项目将所有这些信息的定义放到了实体类中。**这种信息定义依靠注解来实现。**

在 kitra.employeetraining.common.annotation 包中定义了四个注解，分别为 ColumnName（加在字段上，用于指定字段显示名称）、ImportantColumn（加在字段上，用于标记这个列是重要的）、KeyColumn（加在字段上，表明这是主属性）、EntityClass（加在实体类上，用于指定它对应的 DAO 类，建立实体类→DAO 类的联系，用于调用通用增删改查方法）。

为了方便理解，下面给出一个简单的实体类及其对应的 DAO 类，以演示这个系统如何使用。

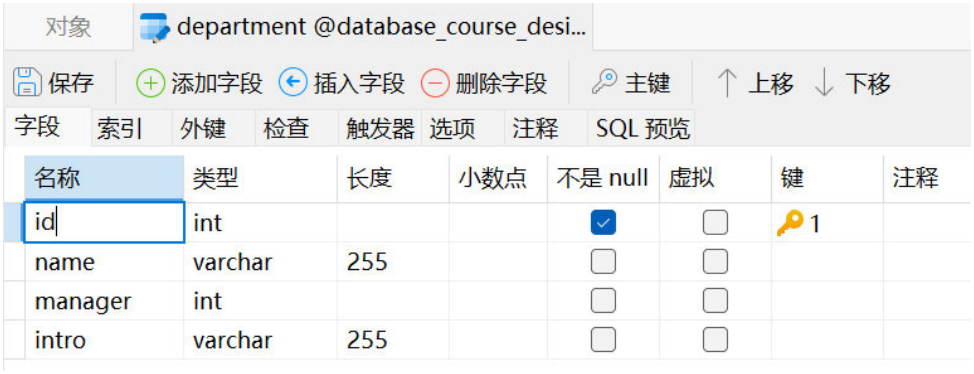


图 27 department 表的结构

Department 实体类：

```
@EntityClass(daoClass = DepartmentDao.class)
@Alias("department")
public class Department implements Entity {
    @KeyColumn
    @ImportantColumn
    @ColumnName("部门编号")
    private int id;
    @ImportantColumn
    @ColumnName("部门名称")
    private String name;
    @ImportantColumn
    @ColumnName("部门经理")
    private Person manager;
    @ColumnName("简介")
    private String intro;

    ... // 省略空参构造函数，getter，setter

    @Override
    public String toString() {
        return this.name;
    }
}
```

这个类中，@EntityClass 注解指定了它的 DAO 类；@Alias 注解由 MyBatis 提供，用来指定它对应的数据表的名称。每个字段都使用了我定义的注解，用来指定它的一些必要信息。

这个类继承了 Entity 接口，Entity 接口是我编写的一个空接口，用来标记这个类是实体类。**这么做的目的是方便 EntityUtils 在解析实体类字段的时候判断一个属性是否为外键**，只要字段的类型是 Entity 的实例，就说明它是外键，否则它就不是外键。

这个类还实现了 toString()方法，其返回值为 this.name。toString()方法是实体类必须单独实现的，因为 EntityUtils 在将实体对象转换为字符串数组时，是通过调用每个字段的 toString()方法来获取其字符串值的。toString()的实现决定了这个实体的值如何显示在界面上。

PersonDao 类：

```
public interface PersonDao {
    @Select("SELECT * FROM person WHERE id = #{id}")
    @Results({
            @Result(property = "password", column = "passwd"),
            @Result(property = "eduLevel", column = "edu_level"),
            @Result(property = "department", column = "department", one = @One(select =
"kitra.employeetraining.common.dao.DepartmentDao.getByIdWithoutManager")),
            @Result(property = "state", column = "state", one = @One(select =
"kitra.employeetraining.common.dao.PersonStateDao.getById")),
            @Result(property = "authority", column = "authority", one = @One(select =
"kitra.employeetraining.common.dao.AuthorityDao.getById"))
    })
    Person getById(Integer id);

    @Results(...) // 省略
    List<Person> getAll();

    @Results(...) // 省略
    @Select("SELECT * FROM person WHERE name LIKE CONCAT('%', #{search}, '%') OR username LIKE
CONCAT('%', #{search}, '%') OR EXISTS(SELECT * FROM authority_code WHERE description LIKE
CONCAT('%', #{search}, '%') AND person.authority = authority_code.code)")
    List<Person> searchByName(String search);

    @Delete("DELETE FROM person WHERE id = #{id}")
    int delete(Integer id);

    ... // 修改、插入以及非通用查询方法
}
```

为了让 EntityUtils 系统正常工作，DAO 类中至少要有 getById、getAll、insert、update 和 delete 方法。而为了让通用的数据显示界面正常工作，还需要 searchByName 方法，它应该实现模糊搜索，并且有些时候需要支持搜索多个字段。

@Results 是 MyBatis 提供的注解，用于指定读取外键实体的方法，以及处理数据库列名与实体类字段名不一致的情况。

五、结果与分析

我向数据库中填入了足够的数据，对各项功能进行测试，结果表明，软件各项功能均正常工作，无故障发生。

六、小结与心得体会

项目前期的难点及解决措施（开发日志）

1. [11 月 11 日，身份验证和访问控制的实现问题]

准备开始编写功能时，我认为用户权限的管理应该用 MySQL 用户和视图的方式实现，但后来我想到，这样会非常麻烦。我想到，在实际应用的时候，客户端应用都不会直接访问数据库，而是通过向服务端应用发送请求来访问数据库。在这种情况下，用户身份验证和权限控制都由服务端应用这个中间层来完成，服务端应用可以直接访问数据库。

摆在我面前的有两种方案，第一种是创建服务端和客户端两套程序，让服务端可以直接访问数据库，

客户端只能通过服务端间接访问数据库。这种方法可以很好地保证数据的安全性；第二种方案是在程序内部创建中间层来进行访问控制，这是实现访问控制的简单方式，但不能抵御逆向工程攻击。

由于开发时间有限，我使用第二种简单方案。我们假定用户只会正常使用软件，不会通过逆向工程方式获取用户名和密码来直接访问本机的数据库。

2. [11 月 12 日，数据库访问方案问题]

我的用户界面已经有了大致的框架，于是我开始着手于功能的开发，首先是数据库访问功能的实现。

我的规划是将数据库访问功能分为两层，一层直接调用 JDBC 相关方法，一层在上一层的基础上抽象出数据库实体的访问操作，并为其加入鉴权操作。我首先在 datamodel 包中将所有数据实体做成 JavaBean 形式的类，然后准备在此基础上编写 DAO（Data Access Object，数据访问对象，即上面提到的抽象层）。此时我注意到，手动封装 JDBC 未免过于复杂，为什么不用 MyBatis 来帮我实现数据库访问的抽象化呢？

当天晚上我开始学习 MyBatis，并成功让 MyBatis 正常工作。我为“考核评价代码”表 apprisement_code 编写了 DAO（MyBatis 称其为 Mapper）接口 ApprisementCodeDao，编写了 MyBatis 初始化代码和配置文件，并解决了所有问题后，测试以下程序：

```
try(SqlSession session = MyBatisUtils.getSqlSession()) {
    var e = session.getMapper(ApprisementCodeDao.class);
    for(short i = 0; i <= 4; i++) {
        System.out.println(e.getApprisementDesc(i));
    }
}
```

输出结果：

```
未考核
不及格
及格
良好
优秀
```

这证明我成功让 MyBatis 访问到了数据库，我可以用更简易的方法实现健壮且支持并发的数据库访问方法了。这可以让我更加专注于程序功能的开发。

3. [11 月 13 日，数据实体内容展示方法问题]

我已经可以做到从数据库中获取到一系列的数据实体对象，但是要将其内容都显示到表格中，比较麻烦。因为有一些数据实体具有大量的属性，若我一个个获取其值，相关的代码会变得极其复杂。

此时我想到了一种方法：使用反射来获取对象的所有属性，再将其放入字符串数组即可。我很快实现了这个功能（详见第四部分第四小节关于实体类结构解析的内容）。

解决完这个问题后，表头仍然需要用一个相当长的字符串数组表示，且由于它与数据实体无直接关联，很容易出现遗漏或数组元素顺序与数据实体的属性顺序不一致的问题，且以后若需对数据的结构进行更改，该数组也要一并修改，十分麻烦。

和朋友讨论该问题后，我想到了用注解配合反射来为数据实体附上名称的方法。

在阅读相关资料后，我成功实现了在实体类中用注解标记属性名，并在程序中自动获取属性名的功能。

我编写了一个注解@ColumnName，附加到数据实体类的每个字段上，然后在 UI 中用反射方式获取对应注解，即可获取表格中的所有列。我将这个功能提取到了工具类 EntityUtils 中。为了减少反射带来的性能开销，我对实体类的 ColumnName 进行了缓存。相关代码如下：

```
private static final Map<Class<?>, Vector<String>> entityColumnNamesMap = new HashMap<>();
...
public static Vector<String> getColumnNames(Class<?> entityClass) {
    if(entityColumnNamesMap.containsKey(entityClass)) {
        return new Vector<>(entityColumnNamesMap.get(entityClass));
    }
    Vector<String> columnNames = new Vector<>();
    Field[] fields = entityClass.getDeclaredFields();
    for(Field field : fields) {
        ColumnName columnNameAnnotation = field.getAnnotation(ColumnName.class);
        if(columnNameAnnotation != null) columnNames.add(columnNameAnnotation.value());
    }
    entityColumnNamesMap.put(entityClass, columnNames);
    return new Vector<>(columnNames);
}
```

4. [11 月 13 日，界面元素的 padding 问题]

在实现图形界面时，我很早开始思考的一个问题是：如何设置元素外部的填充空间？因为默认情况下，Swing 边缘布局都会让元素填满所有空间，这样并不美观。

查找资料后，我了解到可以用对元素添加空白边框的方式实现该功能，对控件调用 setBorder(new EmptyBorder(top, left, bottom, right))方法可以实现各个方向的 padding。

5. [11 月 14 日，数据实体序列化性能调优]

EntityUtils 中的 getColumnNames 已经使用缓存方式来提高性能，但调用频率最高的序列化方法 getStringVectorFromEntity 方法并没有缓存，我开始思考如何提高它的性能。

我发现该方法每次调用都会先获取实体类的所有字段，然后设置字段值可访问，最后依次获取每个字段的值。其中除了最后一步无法缓存外，前面两个主要操作都可以缓存。最后，我使用一个 Map<Class<?>, Field[]>来缓存这些结果。这样优化后，数据量很大的情况下，性能会有显著提升。

5. [11 月 14 日，数据编辑窗口中外键数据选择]

如图所示，数据编辑窗口中的“任课教师”“课程状态”都是经过转换得到的容易阅读的字符串类型，它们在数据库中的原始存储形式都是外键。但是，在用户输入数据的时候，还需要将这种经过处理的属性再对应回原始的外键，才能存入数据库。由于处理后的字符串类型不一定能唯一对应到 key，需要做一个选择器来供用户选择需要的条目。例如，在编辑“任课教师”属性时，让用户从列表中选择出一个人作为任课教师属性的值。

图 28 一个简单的数据编辑窗口

由于传入的参数只有实体的类和实体的实例，我需要从这两个参数中获取所有信息。我定义了一个注解@EntityClass(daoClass)，加在所有实体类上，这样我就可以从实体类获取到 DAO 类；我又规定获取全部数据记录的方法名统一为 getAll()，那么我就成功做到了利用这两个参数来获取全部数据。

用户在选择教师的时候不需要了解教师实体中所有属性的值，用户只需要看到一部分重要的列。因此，我定义了注解@ImportantColumn，加在实体类中所有需要展示的字段上，创建 boolean[] EntityUtils.getIsImportantColumn(Class<? extends Entity> entityClass)方法获取所有 ImportantColumn 信息，创建 filteredGetStringVectorFromEntity 和 filteredGetColumnNames 两个方法来获取筛选后的列。

6. [11 月 15 日，数据编辑窗口中的数据更新操作]

如前文所述，我已经实现了获取外键信息等等的操作，我只要实现将窗口中的数据写入数据库即可。经过研究，我认为应该在每个 DAO 类中定义统一的插入/修改/删除方法，其名称统一，固定参数统一（即为 id），可变参数用 Object 数组传入。考虑到数据库中的字段数据类型各不相同，我认为不应该靠从界面中获取各个输入框的值来作为参数，而是在数据更新窗口添加一个 entityParams 字段用来存放原始数据的 Object 数组，在点击保存按钮时，先根据实际情况用输入框中的值对 entityParams 进行更新，然后再将 entityParams 和 id 一起传入 insert/update 方法。在一天的努力后，我实现了数据的插入和更新功能。

项目开发心得体会

本次课程设计开发中我获益匪浅。尽管我之前接触过数据库相关的应用设计，但这是我第一次实现表的数量较多、属性多、联系复杂的数据库系统。这个项目的规模之大，也是前所未有的。

这是我第一次完整开发一个功能复杂的 GUI 应用程序。尽管我使用的是不算流行的 Swing，我仍然学到了 GUI 开发中的一些重要的设计思想，以及布局模式的重要性。这对我之后的前端开发非常有指导意义。

这也是我第一个使用了多个依赖库的 Java 项目。在这个项目中，我第一次接触到了 MyBatis，并了解到了它在 Java 数据库编程中带来的便利。我还接触到了注解和反射技术，并真正将它们应用起来，还学会了缓存反射的中间结果以提高性能。我使用了 Apache Commons 和国产 Excel 库 EasyExcel，并和一些大佬进行了技术交流，学到了很多干货。

本次项目开发中我也有不足之处，例如前期我为了保证项目的“优雅”和代码复用性，多次对项目进行重构，让项目的结构变得十分复杂，而且开发缓慢。后来我了解到，让软件功能得到实现比让软件写得更好更加重要，我应该避免本末倒置的开发思维，先从整体下手，再关注细节，这样可以有效保证项目开发思路流畅清晰不受阻碍。在我最近的新项目中，我已经开始采用这种思想。

写在最后

最后，感谢您能够耐心读完我的课设报告，项目的完成离不开老师和朋友们的指导。我希望我能够在未来的项目开发中不断进步，也祝愿您能够身体健康、万事如意。

谢谢。