

**《Web课程设计》项目设计报告**



**团队编号**  101

**团队名称**  海报突击队

**院 系**  机械与电气工程学院

**专 业**  计算机科学与技术

**团队成员** 陈俊龙、顾煜、侯志宁

**团队成员** 桂婷、邓棉茵、洪志炫

**填写日期** 2024-07-12

# 大学生海报设计竞赛系统项目设计

## **1.1** 需求分析

### **1.1.1 项目背景**

为了展示大学生的艺术才华，激发其创新意识和实践能力，我们决定开发一套大学生海报设计竞赛系统。大学生可以创建或加入队伍进行竞赛、查看赛事、查看作品、分享作品、点赞作品等。管理员可以发布竞赛、设置获奖、发布新闻、数据统计等。评委可以查看作品，设置分数，设置评语等。

为了方便，我们给这个管理系统取一个英文缩写名称：PDCS，也就是Poster Design Competition System系统。开设的海报设计竞赛系统希望达到如表1-1所示目的。

表1-1 PDCS业务前景表

|  |  |
| --- | --- |
| **接口** | **目标** |
| P01 | 大学生可以创建或加入队伍进行竞赛、查看赛事、查看作品、分享作品、点赞作品等。 |
| P02 | 管理员可以发布竞赛、设置获奖、发布新闻、数据统计等。 |
| P03 | 评委可以查看作品，设置分数，设置评语等。 |

### **1.1.2 业务流程分析**

首先通过和项目相关人员访谈、收集整理资料来掌握现有业务流程，然后结合业务前景对流程进行优化。下面分析PDCS项目核心业务流程，注意业务流程分析只聚焦做什么，而不是如何做。图1-1是PDCS项目的核心业务流程图，重点在于对业务情况的整体梳理。

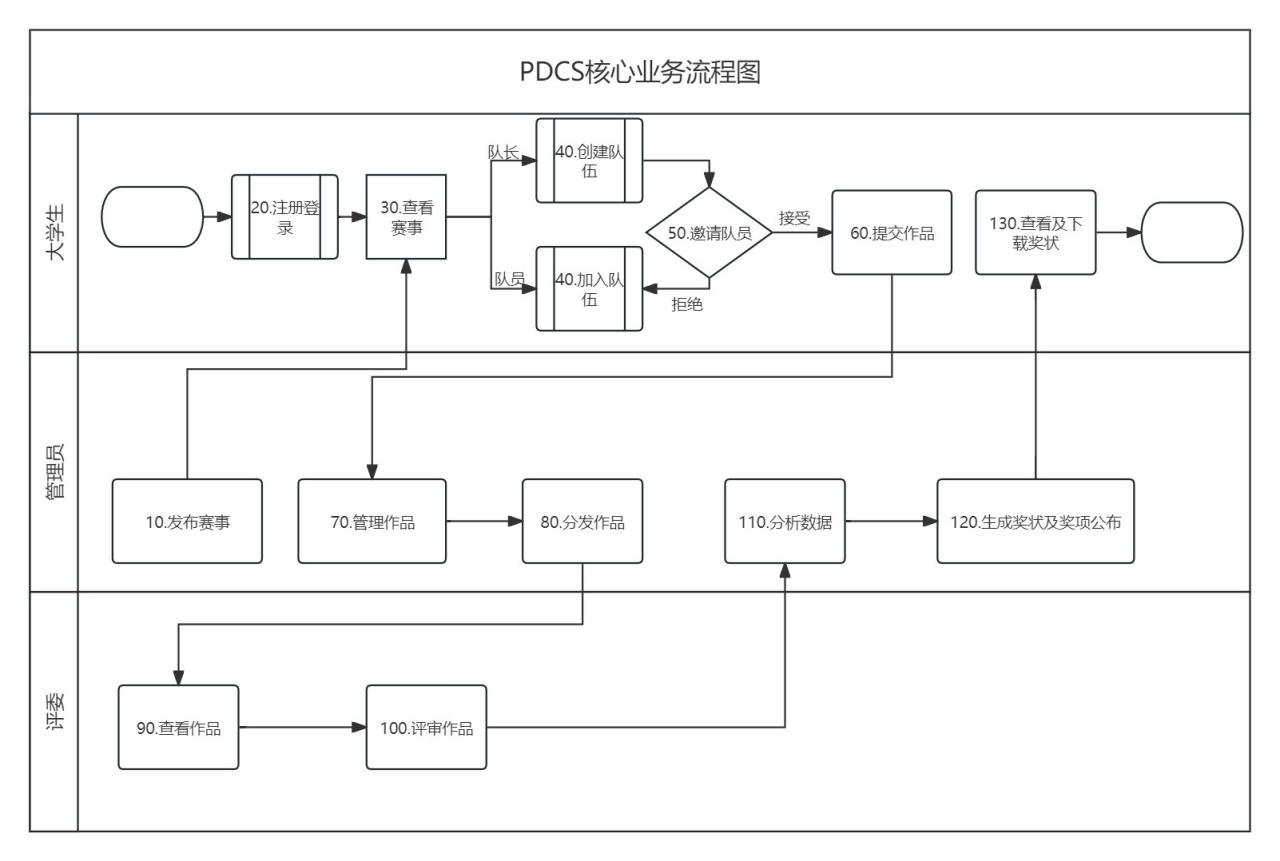


图1-1 核心业务流程图

项目中大量细节的内容，用流程图来表示会显得非常繁琐，此时可以用表1-2的方式对流程图进行描述。为了方便将描述和流程图对照起来，图1-1中给图标编了号。

表1-2 业务流程描述

|  |  |  |
| --- | --- | --- |
| **编号** | **责任人** | **说明** |
| 10 | 管理员 | 管理员发布赛事 |
| 20 | 大学生 | 大学生注册并登录竞赛系统 |
| 30 | 大学生 | 大学生可以查看赛事 |
| 40 | 大学生 | 大学生分为队长和队员，队长创建队伍，队员加入队伍 |
| 50 | 大学生 | 大学生中队长身份可以邀请队员加入队伍 |
| 60 | 大学生 | 大学生邀请成功组建队伍可以提交作品 |
| 70 | 管理员 | 管理员对提交上来的作品进行管理 |
| 80 | 管理员 | 管理员将作品分发给评委 |
| 90 | 评委 | 评委可以查看分发的作品 |
| 100 | 评委 | 评委对作品可以进行评审 |
| 110 | 管理员 | 管理员对评委评审完的作品进行分析 |
| 120 | 管理员 | 管理员给作品生成奖状及奖项公布 |
| 130 | 大学生 | 大学生可以查看奖状及下载 |

### **1.1.3 用例分析**

根据业务流程，结合前景分析得到PDCS项目的用例。由于用例较多，为了避免单个用例图过于复杂，根据业务逻辑将用例划分到多个用例图。

大学生用例图给出针对大学生这个角色的用例图，如图1-2所示。

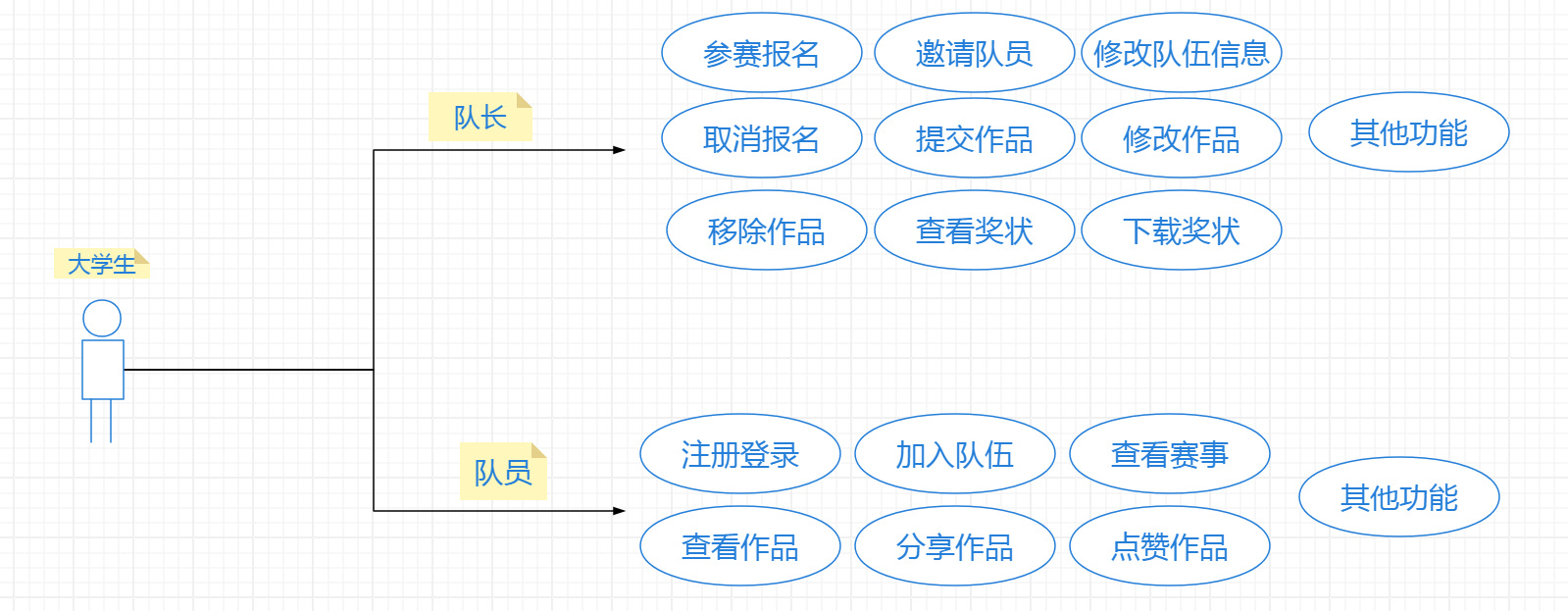


图1-2 大学生用例图

图1-3给出了后台管理用例图，用例名称通常用动词。

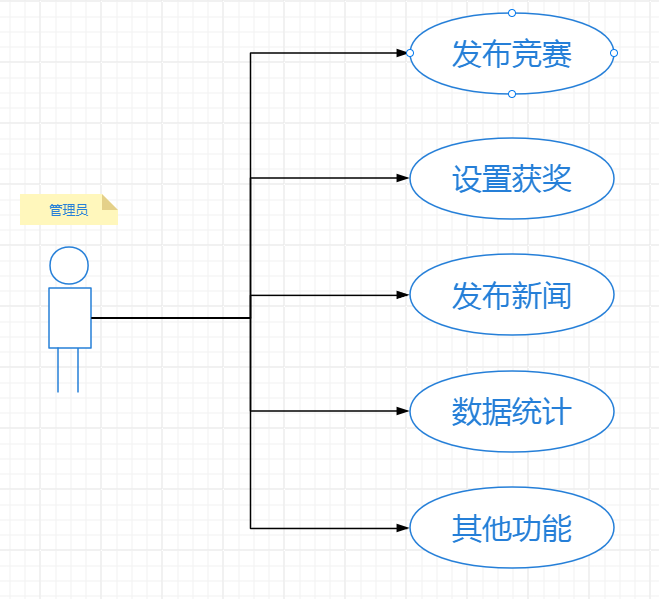


图1-3 后台管理用例图

图1-4给出了评委用例图，用例名称通常用动词。

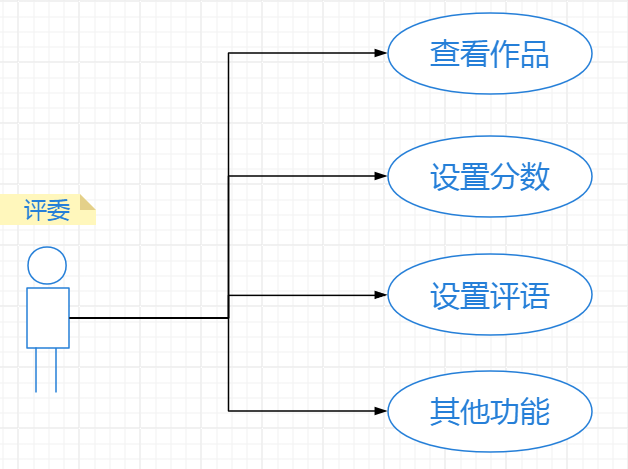


图1-4 评委用例图

## **1.2** 系统设计

### **1.2.1 功能模块设计**

图1-5是PDCS项目功能模块图，由于功能模块比较多，所以采用分层设计，将所有功能分为“基础功能”、“学生功能”、“管理员功能”、“评委功能”和“竞赛中心”五大部分。

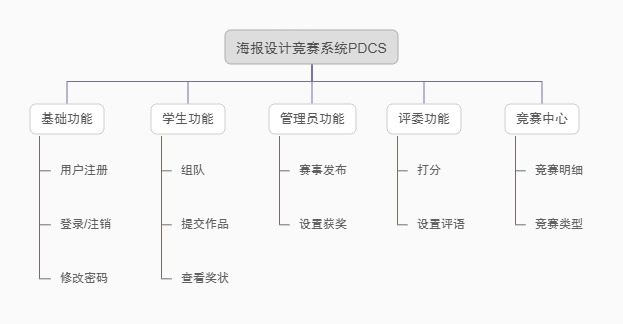


图1-5功能模块图

表1-3给出了PDCS项目功能模块简要说明。

表1-3 功能模块说明

|  |  |  |
| --- | --- | --- |
| **模块** | **功能** | **功能描述** |
| 基本功能 | 用户注册 | 所有人都可以注册成为参与人员、管理员、评委 |
| 登录/注销 | 参与人员登录海报设计竞赛系统和退出海报设计竞赛系统 |
| 密码修改 | 所有人都可以修改登录的密码 |
| 学生功能 | 组队 | 学生可以自由组队，参加竞赛 |
| 提交作品 | 每个队长都可以提交参加比赛的作品 |
| 查看奖状 | 每个队长都可以查看自己队伍是否获得奖状 |
| 管理员功能 | 赛事发布 | 管理员能够发布竞赛、发布获奖情况的新闻 |
| 设置获奖 | 管理员可以根据评委的打分设置获奖 |
| 评委功能 | 打分 | 评委可以给作品打分 |
| 设置评语 | 评委可以给作品设置评语 |
| 竞赛中心 | 竞赛明细 | 包含了竞赛的详细介绍 |
| 竞赛类型 | 包含了竞赛的类型 |

### **1.2.2 MVC开发模式**

MVC是一种软件的开发模式，目前主流的软件系统开发都受到这种思想的指导。

M即模型（Model）是表示系统业务处理相关代码组件的集合。在MVC的三个部件中，模型拥有最多的处理任务。同一个模型能为多个视图提供数据，由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。

V即视图（View）是指用户看到并与之交互的界面。比如由HTML元素组成的网页界面，或者软件的客户端界面。MVC的好处之一在于它能为应用程序处理很多不同的视图。在视图中其实没有真正的处理发生，它只是作为一种输出数据并允许用户操作的方式。

C即控制器（Controller），控制器接受用户的输入并调用模型和视图去完成用户的需求，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型去处理请求，然后再确定用哪个视图来显示返回的数据。

MVC开发模式有如下四个优点：

1. 耦合性低

视图和模型分离，这样就允许更改视图层代码而不用重新编译模型和控制器代码，同样，一个应用的业务流程或者业务规则的改变只需要改动模型层即可。因为模型与控制器和视图相分离，所以很容易改变应用程序的模型层和业务规则。

1. 重用性高

MVC模式允许使用各种不同样式的视图来访问同一个服务器端的代码，因为多个视图能共享一个模型，它包括任何Web浏览器或者无线浏览器（wap）。由于模型返回的数据没有与界面代码混合，所以同样数据能被不同的界面使用。

1. 开发效率高，生命周期成本低

MVC模式下，开发和维护接口代码的技术含量降低。使用MVC模式使开发时间得到相当大的缩减，它使开发人员（Java开发人员）聚焦于业务逻辑，界面程序员（HTML和JSP开发人员）集中精力于表现形式上。

1. 可维护性高

MVC模式下各个层次耦合性低，有利于项目的后期维护和修改。

PDCS项目，采用Java Web开发技术，结合MVC开发模式，项目中各个组件职责划分如图1-6所示。

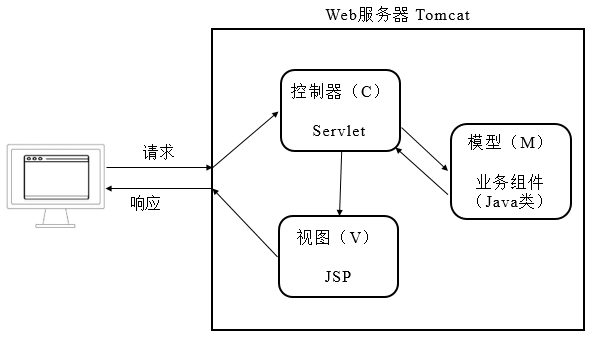


图1-6 MVC开发模式

Servlet组件承担了控制器（Controller）的角色，它们的常规工作流程如下：

* 接收客户端的请求信息。
* 调用执行业务模型，获取其执行后返回的数据。
* 把数据传递给视图，将视图展示给客户端。

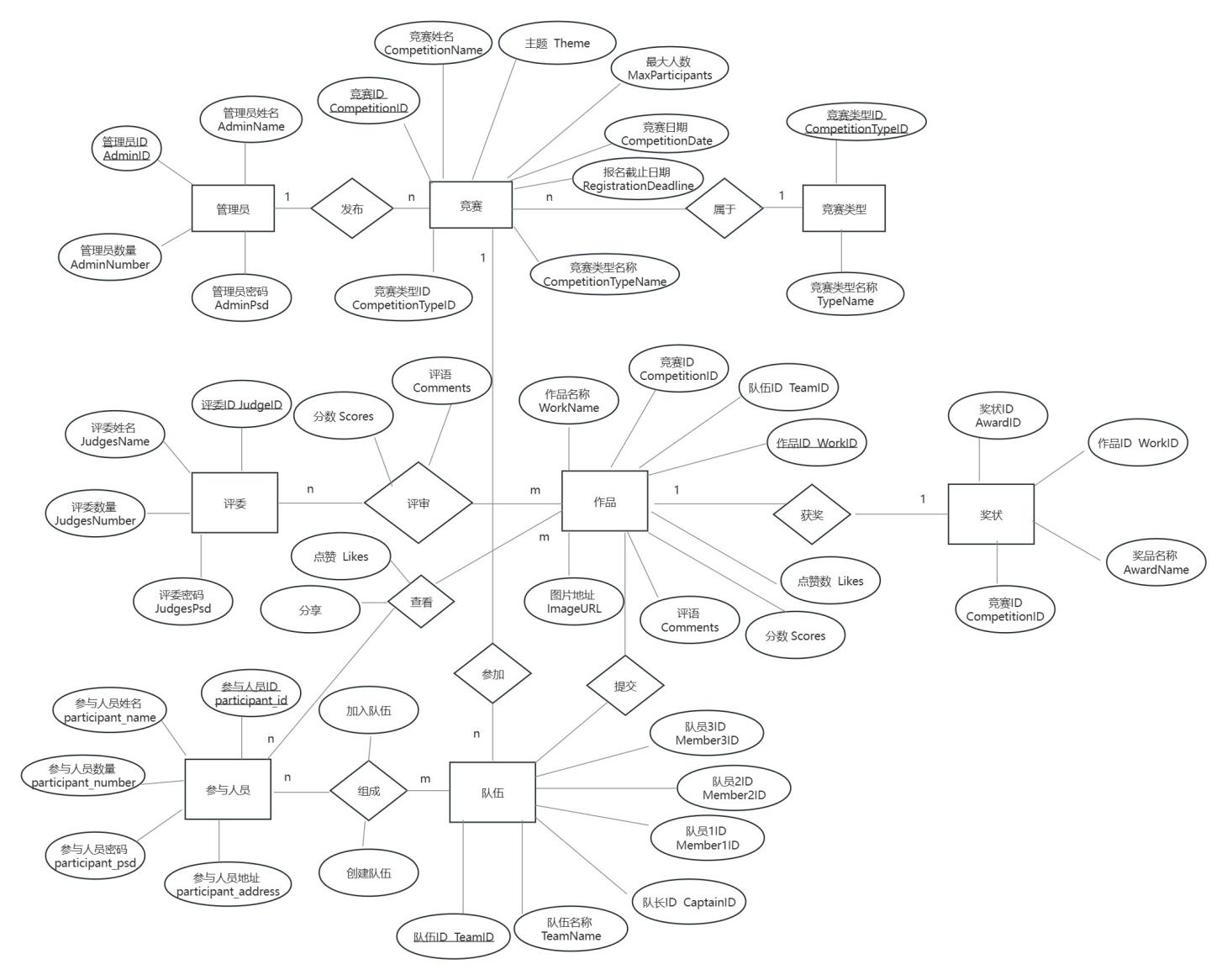
JSP组件承担了视图（View）的角色，它们的主要工作是使用EL或者JSTL在页面中展示数据。项目在后续加入JQuery框架的Ajax请求时，会使用JavaScript在页面上展示数据。

项目中其他的组件，例如数据库存取类、业务数据类、业务逻辑类、工具类等内容，承担了模型（Model）的角色，它们的主要工作是处理项目的业务逻辑，返回结果数据。

## **1.3 数据库设计**

### **1.3.1 概念模型**

根据前面的需求分析和系统设计，可以得到如图1-7所示的ER图和如图1-8所示的实体类图。



101

图1-7 ER图

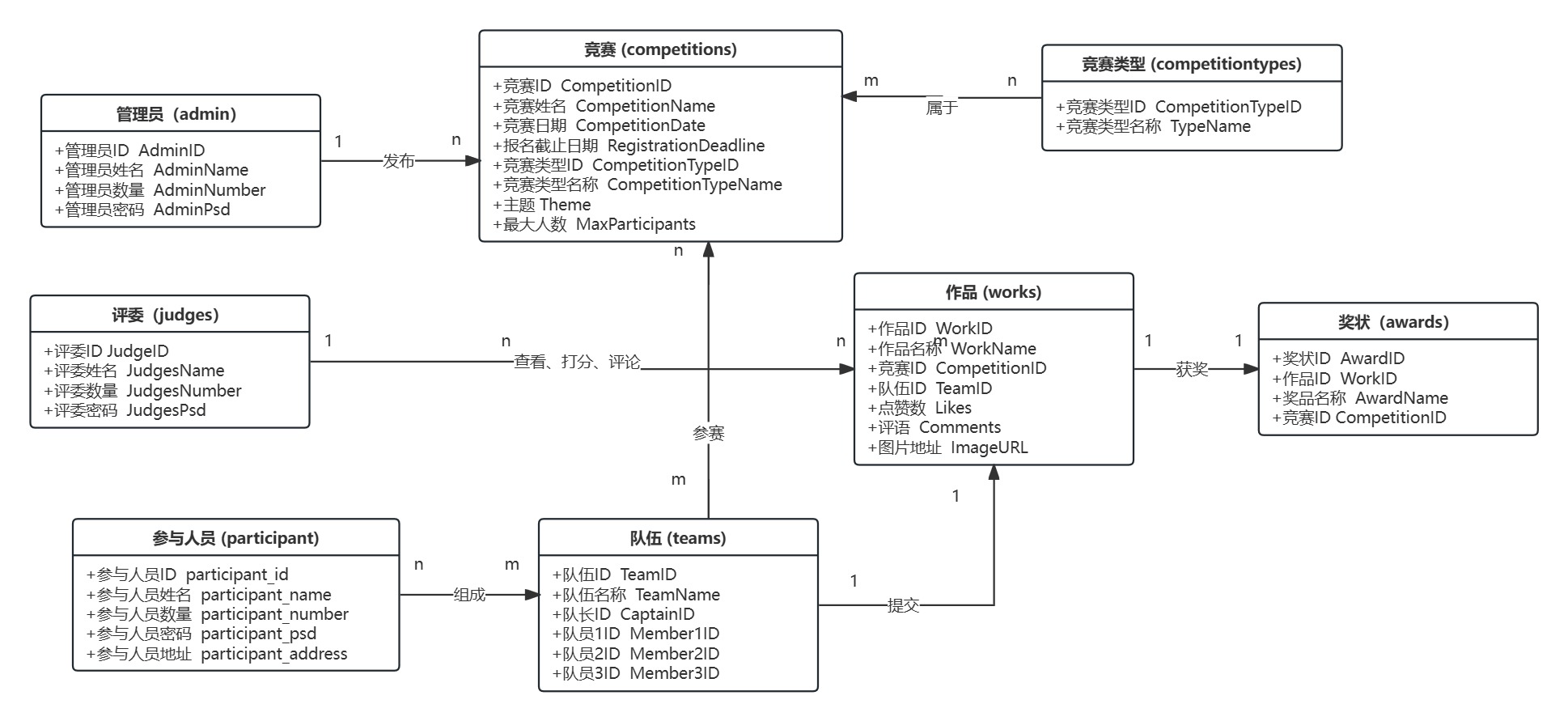


图1-8实体类图

概念模型设计过程中的实体类图需要确保满足具体的业务需求：

* 管理员可以发布多个竞赛
* 评委可以给查看多个作品并进行打分和评论
* 参与人员可以组成多个队伍参加多个竞赛
* 一支队伍在一个竞赛中只能提交一份作品
* 一份作品最多获得一张奖状

### **1.3.2 关系模型**

PDCS项目关系模型采用表格方式描述，关系模型的设计基于需求分析和实体类图，特别注意关系模型如何体现实体类之间的关系，也就是表中外键字段的说明。

如表1-4所示，管理员表（admin）保存系统中所有管理员的信息。

表1‑4管理员表admin

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 属性 | 说明 |
| AdminID | Int | PK，IDENTITY | 管理员Id。(自增型) |
| AdminName | Varchar(50) | NOT NULL | 管理员姓名 |
| AdminNumber | Varchar(50) | NOT NULL | 管理员数量 |
| AdminPsd | Varchar(50) | NOT NULL | 管理员密码 |

如表1-5所示，奖状表（awards）保存奖状信息。

表1‑5奖状表awards

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 属性 | 说明 |
| AwardID | Int | PK，IDENTITY | 奖状ID。(自增型) |
| WorkID | Int | FK,NOT NULL | 作品ID。 |
| AwardName | Varchar (100) | NOT NULL | 奖品名称。 |
| CompetitionID | Int | FK,NOT NULL | 竞赛ID。 |

如表1-6所示，竞赛表（competitions）保存系统中所有竞赛的基本信息。

表1‑6竞赛表competitions

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 属性 | 说明 |
| CompetitionID | Int | PK，IDENTITY | 竞赛ID。(自增型) |
| CompetitionName | Varchar(255) | NOT NULL | 竞赛姓名 |
| CompetitionDate | DateTime | NOT NULL | 竞赛日期 |
| RegistrationDeadline | DateTime | NOT NULL | 报名截止日期 |
| CompetitionTypeID | Int | DEFAULT NULL, | 竞赛类型ID |
| CompetitionTypeName | Varchar(255) | DEFAULT NULL, | 竞赛类型名称 |
| CompetitionDescription | Text | NOT NULL | 竞赛描述 |
| MaxParticipants | Int | DEFAULT NULL, | 最大人数 |
| Theme | Varchar(255) | DEFAULT NULL, | 主题 |

如表1-7所示，竞赛类型表（competitiontypes）保存竞赛类型信息，用于竞赛的管理。

表1‑7 竞赛类型表competitiontypes

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 属性 | 说明 |
| CompetitionTypeID | Int | PK，IDENTITY | 竞赛类型ID。(自增型) |
| TypeName | Varchar(255) | NOT NULL | 竞赛类型名称 |

如表1-8所示，评委表（judges）保存评委信息。

表1‑8评委表judges

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 属性 | 说明 |
| JudgeID | Int | PK，IDENTITY | 评委ID。(自增型) |
| JudgesName | Varchar(255) | NOT NULL | 评委姓名 |
| JudgesNumber | Varchar(255) | NOT NULL | 评委数量 |
| JudgesPsd | Varchar(255) | NOT NULL | 评委密码 |

如表1-9所示，参与人员表（participant）保存参与人员信息。

表1‑9参与人员表participant

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 属性 | 说明 |
| participant\_id | Int | PK，IDENTITY | 参与人员ID。(自增型) |
| participant\_name | Varchar(255) | NOT NULL | 参与人员姓名 |
| participant\_number | Varchar(50) | NOT NULL | 参与人员数量 |
| participant\_psd | Varchar(50) | NOT NULL | 参与人员密码 |
| participant\_address | Varchar(255) | NOT NULL | 参与人员地址 |

如表1-10所示，队伍表（teams）保存队伍信息。

表1‑10队伍表teams

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 属性 | 说明 |
| TeamID | Int | PK，IDENTITY | 队伍ID。(自增型) |
| TeamName | Varchar(255) | NOT NULL | 队伍名称 |
| CaptainID | Int | PK,NOT NULL | 队长ID |
| Member1ID | Int | PK,NOT NULL | 队员1ID |
| Member2ID | Int | PK,NOT NULL | 队员2ID |
| Member3ID | Int | PK,NOT NULL | 队员3ID |
| CompetitionID | Int | NOT NULL | 竞赛ID |

如表1-11所示，作品表（works）保存作品信息。

表1‑11作品表works

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 属性 | 说明 |
| WorkID | Int | PK，IDENTITY | 作品ID。(自增型) |
| WorkName | Varchar(255) | NOT NULL | 作品名称 |
| CompetitionID | Int | NOT NULL | 竞赛ID |
| TeamID | Int | NOT NULL | 队伍ID |
| Likes | Int | NOT NULL | 点赞数 |
| Comments | Text | PK,NOT NULL | 评语 |
| ImageURL | Varchar(255) | NOT NULL | 图片地址 |
| Score | Int | NOT NULL | 作品分数 |

### **1.3.2 物理设计**

完成关系模型的设计后，针对所采用的数据库系统完成物理设计，也就是确定数据库的存储结构、确定表结构等内容。

首先确定数据库系统采用MySQL，数据库名为pdcs。下面给出具体的数据库创建脚本代码，开发人员在数据库客户端中执行pdcs.sql文件中的代码即可完成数据库物理实现。

源程序：pdcs.sql数据库脚本文件

--创建数据库

CREATE DATABASE PDCS charset = utf8;

USE PDCS;

--创建管理员表

CREATE TABLE admin(

AdminID INT(11) NOT NULL AUTO\_INCREMENT,

AdminName VARCHAR(255) NOT NULL,

AdminNumber VARCHAR(255) NOT NULL,

AdminPsd VARCHAR(255) NOT NULL,

PRIMARY KEY (AdminID),

UNIQUE KEY AdminNumber (AdminNumber)

) ENGINE=INNODB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8;

--创建奖状表

CREATE TABLE awards(

AwardID INT(11) NOT NULL AUTO\_INCREMENT,

WorkID INT(11) NOT NULL,

AwardName VARCHAR(255) NOT NULL,

CompetitionID INT(11) NOT NULL,

Winner1ID INT(11) NOT NULL,

Winner2ID INT(11) NOT NULL,

Winner3ID INT(11) NOT NULL,

Winner4ID INT(11) NOT NULL,

PRIMARY KEY (AwardID)

) ENGINE=INNODB DEFAULT CHARSET=utf8;

--创建竞赛表

CREATE TABLE competitions(

CompetitionID INT(11) NOT NULL AUTO\_INCREMENT,

CompetitionName VARCHAR(255) NOT NULL,

CompetitionDate DATE NOT NULL,

RegistrationDeadline DATE NOT NULL,

CompetitionTypeID INT(11) DEFAULT NULL,

CompetitionTypeName VARCHAR(255) DEFAULT NULL,

CompetitionDescription TEXT NOT NULL,

MaxParticipants INT(11) DEFAULT NULL,

Theme VARCHAR(255) DEFAULT NULL,

PRIMARY KEY (CompetitionID),

KEY fk\_competition\_type (CompetitionTypeID),

CONSTRAINT fk\_competition\_type FOREIGN KEY (CompetitionTypeID) REFERENCES competitiontypes (CompetitionTypeID)

) ENGINE=INNODB DEFAULT CHARSET=utf8;

--创建竞赛类型表

CREATE TABLE competitiontypes(

CompetitionTypeID INT(11) NOT NULL AUTO\_INCREMENT,

TypeName VARCHAR(255) NOT NULL,

PRIMARY KEY (CompetitionTypeID)

) ENGINE=INNODB DEFAULT CHARSET=utf8;

--创建评委表

CREATE TABLE judges(

JudgeID INT(11) NOT NULL AUTO\_INCREMENT,

JudgesName VARCHAR(255) NOT NULL,

JudgesNumber VARCHAR(255) NOT NULL,

JudgesPsd VARCHAR(255) NOT NULL,

PRIMARY KEY (JudgeID),

UNIQUE KEY JudgesNumber (JudgesNumber)

) ENGINE=INNODB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8;

--创建参赛人员表

CREATE TABLE participant(

participant\_id INT(11) NOT NULL AUTO\_INCREMENT,

participant\_name VARCHAR(255) NOT NULL,

participant\_number VARCHAR(50) NOT NULL,

participant\_psd VARCHAR(50) NOT NULL,

participant\_address VARCHAR(255) NOT NULL,

enter INT(11) DEFAULT NULL,

PRIMARY KEY (participant\_id)

) ENGINE=INNODB AUTO\_INCREMENT=6 DEFAULT CHARSET=utf8;

--创建队伍表

CREATE TABLE teams(

TeamID INT(11) NOT NULL AUTO\_INCREMENT,

TeamName VARCHAR(255) NOT NULL,

CaptainID INT(11) NOT NULL,

Member1ID INT(11) DEFAULT NULL,

Member2ID INT(11) DEFAULT NULL,

Member3ID INT(11) DEFAULT NULL,

CompetitionID INT(11) DEFAULT NULL,

PRIMARY KEY (TeamID),

KEY CaptainID (CaptainID),

KEY Member1ID (Member1ID),

KEY Member2ID (Member2ID),

KEY Member3ID (Member3ID),

CONSTRAINT teams\_ibfk\_1 FOREIGN KEY (CaptainID) REFERENCES participant (participant\_id),

CONSTRAINT teams\_ibfk\_2 FOREIGN KEY (Member1ID) REFERENCES participant (participant\_id),

CONSTRAINT teams\_ibfk\_3 FOREIGN KEY (Member2ID) REFERENCES participant (participant\_id),

CONSTRAINT teams\_ibfk\_4 FOREIGN KEY (Member3ID) REFERENCES participant (participant\_id)

) ENGINE=INNODB DEFAULT CHARSET=utf8;

--创建作品表

CREATE TABLE works(

WorkID INT(11) NOT NULL AUTO\_INCREMENT,

WorkName VARCHAR(255) NOT NULL,

CompetitionID INT(11) NOT NULL,

TeamID INT(11) NOT NULL,

Likes INT(11) DEFAULT '' 0 '',

Comments TEXT,

ImageURL VARCHAR(255) DEFAULT NULL,

Score INT(11) DEFAULT NULL,

PRIMARY KEY (WorkID),

KEY CompetitionID (CompetitionID),

KEY TeamID (TeamID),

CONSTRAINT works\_ibfk\_1 FOREIGN KEY (CompetitionID) REFERENCES competitions (CompetitionID),

CONSTRAINT works\_ibfk\_2 FOREIGN KEY (TeamID) REFERENCES teams (TeamID)

) ENGINE=INNODB DEFAULT CHARSET=utf8;

为了在系统开发过程中帮助开发人员进行代码调试，需在数据库中初始化一批测试数据，具体数据在测试数据脚本文件pdcs\_data.sql中给出。

源程序：pdcs\_data.sql测试数据脚本文件

USE PDCS;

--插入管理员数据

INSERT INTO admin VALUES ('1', 'Alice', 'admin@1234.com', '1234');

INSERT INTO admin VALUES ('2', 'Bob', 'admin@1213.com', '1211');

--插入竞赛类型数据

INSERT INTO competitiontypes (CompetitionTypeID, TypeName) VALUES (1, '公益类');

INSERT INTO competitiontypes (CompetitionTypeID, TypeName) VALUES (2, '国防类');

INSERT INTO competitiontypes (CompetitionTypeID, TypeName) VALUES (3, '商业类');

--插入评委数据

INSERT INTO judges VALUES ('1 ', 'Jhon','1224.com','1234');

INSERT INTO judges VALUES ('2 ', 'Mike','1432.com','9879');

--插入参与人员数据

INSERT INTO participant VALUES('1','Jim','3415.com','1234','二仙桥','0');

INSERT INTO participant VALUES('2','Kiko','2456.com','1111','成华大道','0');

INSERT INTO participant VALUES('4','朱杨华','1267.com','7890','绍兴','0');

INSERT INTO participant VALUES('5 ','Lili','9868.com','6521','天海市','0');

## **1.4** 项目架构搭建

### **1.4.1 创建项目**

在IDEA菜单栏中，选择File →New →Project命令，在呈现的对话框选择Java Enterprise项目类型，输入项目名称为PDCS、位置D:\IDEA\program\PDCS，选择模块结构为Web application，选择应用服务器Tomcat Server，如图1-7所示。

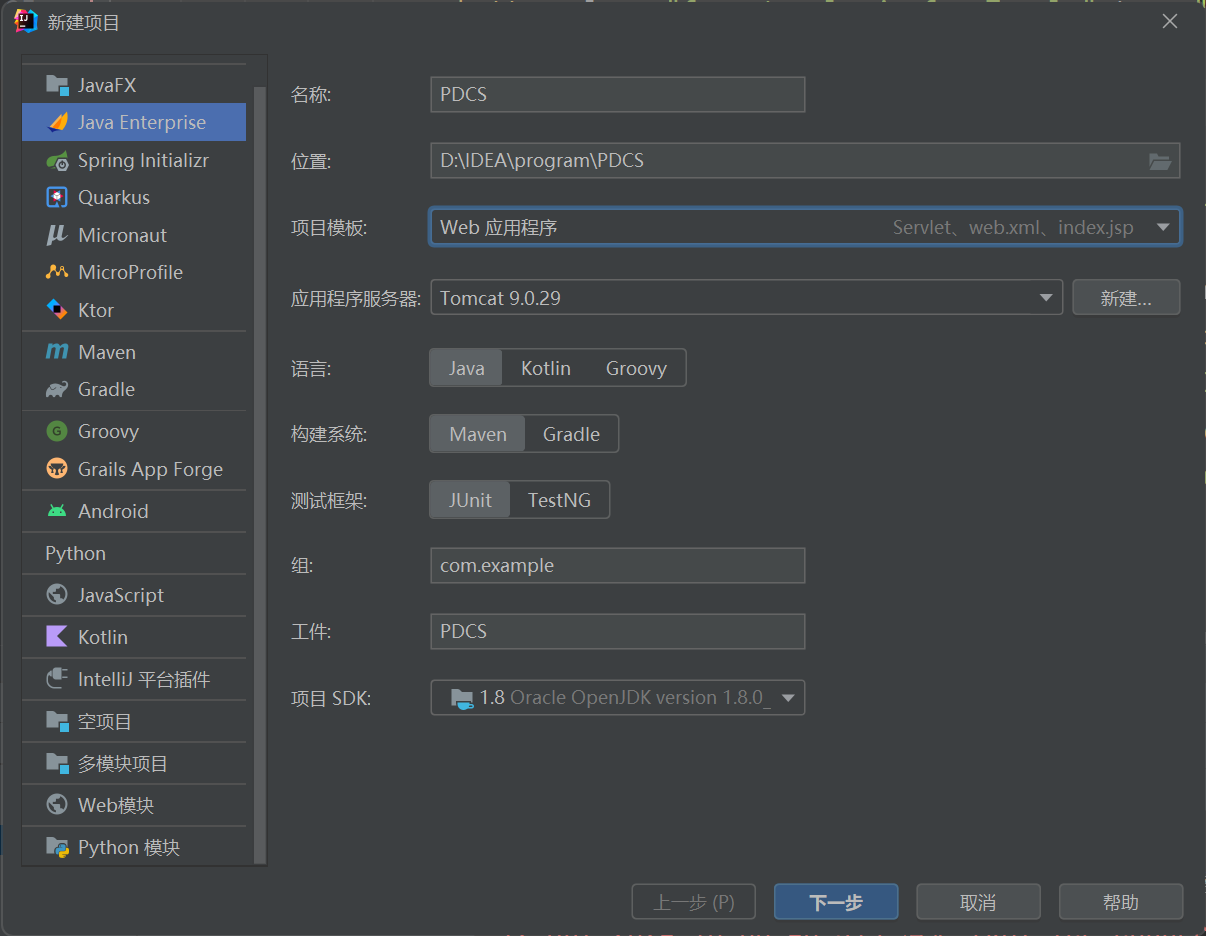


图1-7 新建PDCS项目界面1

然后单击Next按钮，进入项目配置界面，如图1-8所示。

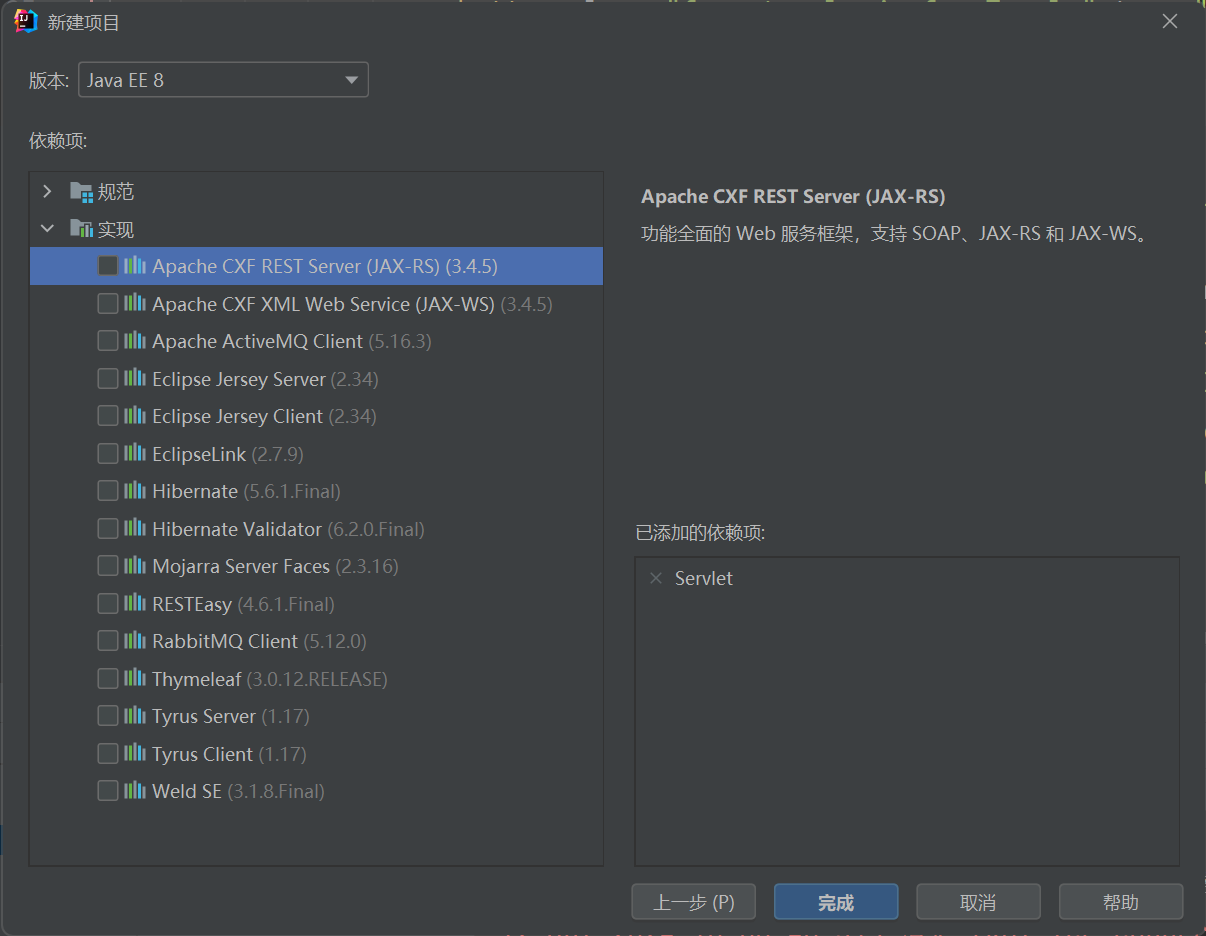


图1-8 新建PDCS项目界面2

最后单击Finish按钮，完成项目创建。项目PDCS初始的文件结构，如图1-9所示。

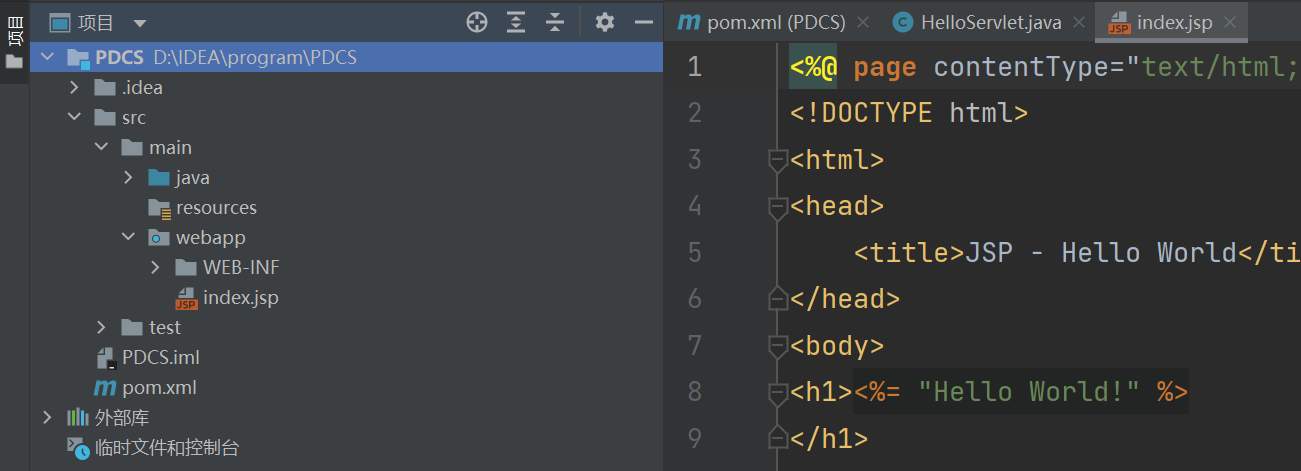


图1-9 PDCS项目初始文件结构

### **1.4.2 项目架构**

依据MVC开发模式，完善项目架构。在PDCS项目的文件夹src\main\java上右击，选择New→Package，在弹出的窗口中输入com.example.servlet，回车创建Package。同理创建com.example.domain、com.example.dao、com.example.utils，如图1-10所示。

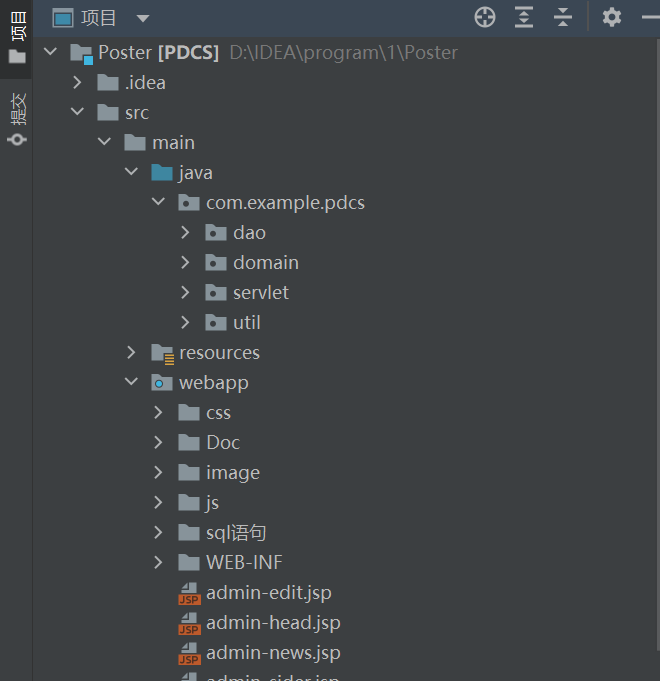


图1-10 PDCS项目架构

PDCS项目中MVC模式说明，com.example.servlet包中编写Servlet代码，代表控制层（C）；webapp站点中编写JSP代码，代表视图层（V）；com.example.domain包中编写业务数据类、业务逻辑类代码，com.example.dao包中编写数据库存取类代码，它们代表模型层（M）。详细说明如表1-9所示。

表1-9 项目架构说明表

|  |  |  |
| --- | --- | --- |
| **包Package/文件夹** | **包含内容** | **备注** |
| com.example.domain | 大学生海报设计竞赛系统业务数据实体类、业务逻辑处理类 | MVC开发模式中模型层（M） |
| com.example.dao | 数据库存储代码类 | MVC开发模式中模型层（M） |
| webapp | 大学生海报设计竞赛系统网站JSP页面 | MVC开发模式中视图层（V） |
| com.example.servlet | 大学生海报设计竞赛系统网站Servlet类 | MVC开发模式中控制层（C） |
| com.example.utils | 通用工具类 |  |

基于MVC开发模式的项目架构，没有固定的结构和命名规范，开发人员可以根据项目的复杂程度自定义结构，符合MVC思想即可。

### **1.4.3 配置jar包**

在PDCS项目开发过程中，使用了多个第三方jar包，具体清单如表1-10所示。

表1-10 第三方jar包清单

|  |  |  |
| --- | --- | --- |
| **jar名称** | **说明** | **版本** |
| mysql-connector-java | MySQL数据库驱动 | 8.0.28 |
| druid | JDBC数据源 | 1.2.8 |
| commons-logging | 日志操作 | 1.2 |
| spring-jdbc | JDBCTemplate数据库工具 | 5.3.17 |
| fastjson2 | JSON解析和生成器 | 2.0.18 |
| taglibs-standard-spec | JSTL接口 | 1.2.1 |
| taglibs-standard-impl | JSTL实现 | 1.2.1 |
| commons-io | 文件上传基础 | 2.2 |
| commons-fileupload | 文件上传 | 1.4 |
| javax.servlet-api | Servlet API | 4.0.1 |
| junit-jupiter-api | JUnit Jupiter API | ${junit.version} |
| junit-jupiter-engine | JUnit Jupiter Engine | ${junit.version} |

第三方jar包在PDCS项目中由maven工具管理，开发人员配置项目中的pom.xml文件，maven工具自动从网络中下载相应jar包到项目中。

源程序：pom.xml文件中依赖配置代码

<dependencies>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-api</artifactId>

<version>${junit.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>${junit.version}</version>

<scope>test</scope>

</dependency>

</dependencies>

编写pom.xml文件中的第三方jar包依赖配置代码，一般情况开发人员不是直接编写，而是从网络中复制配置代码。站点MVNrepository就是提供这类服务的网站，其网址为：https://mvnrepository.com/。在这个网站上，开发人员搜索jar包的关键字即可获取相应的配置代码。

通常第三方jar包配置完成后，在IDEA的项目中可以查看相应的jar包文件，如图1-11所示。

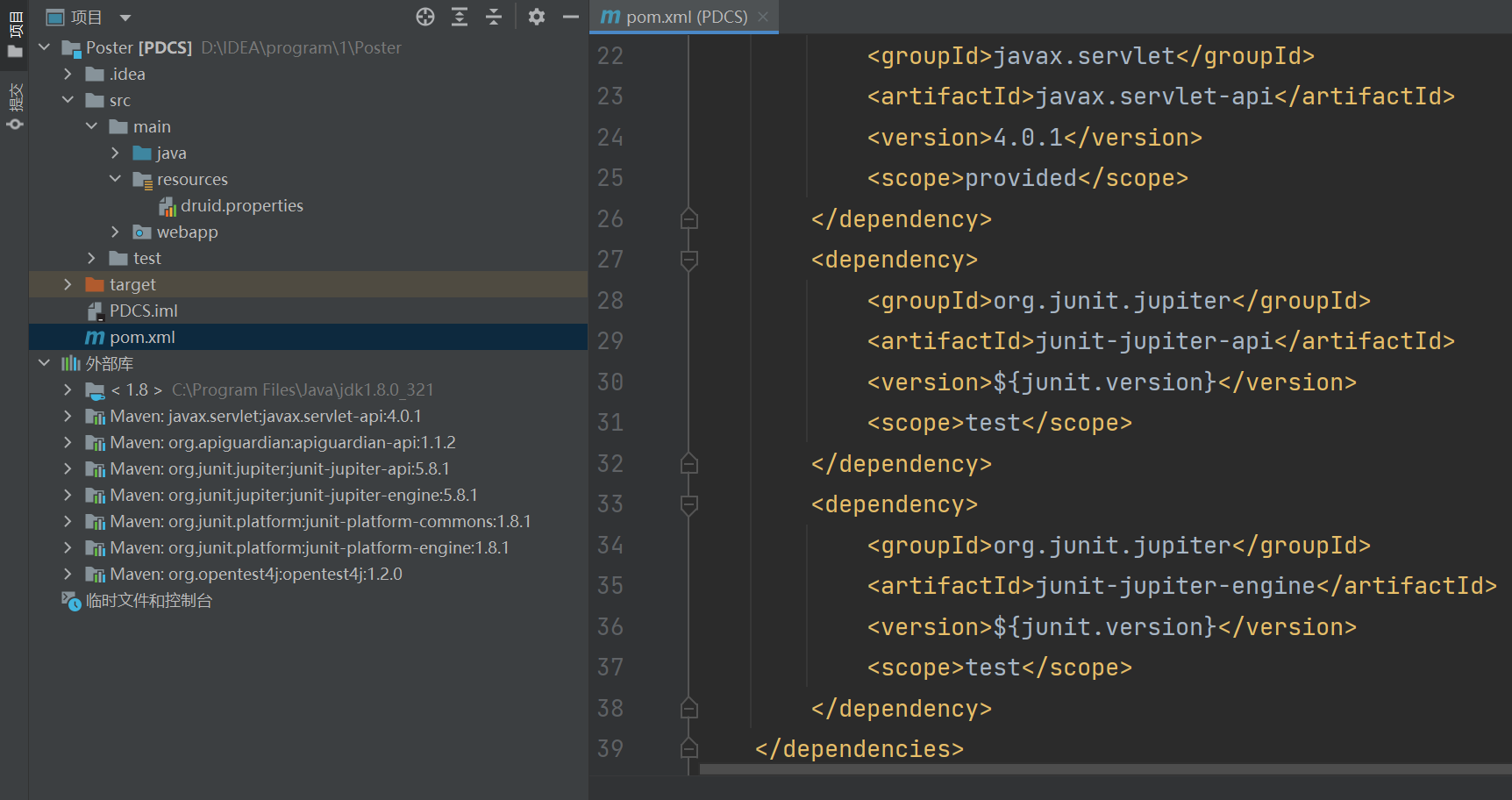


图1-11 PDCS项目依赖的第三方jar包

### **1.4.4 网站结构**

PDCS项目中默认已有webapp站点，需要进一步初始化站点的文件夹，文件夹清单如表1-11所示。

表1-11 webapp站点文件夹清单

|  |  |
| --- | --- |
| **文件夹名称** | **说明** |
| css | 存放样式文件，\*.css |
| image | 存放图片文件，\*.png \*.ico |
| js | 存放JavaScript脚本文件, \*.js |
| doc | 存放文档文件，\*.docx |
| sql | 存放数据库文件，\*.sql |
| WEB-INF | Web站点的安全文件夹, 存放web.xml等文件 |

在PDCS项目的文件夹webapp上右击，选择New→Directory，在弹出的窗口中输入css，回车创建文件夹，同理创建文件夹img、js、doc。创建完成后webapp文件夹结构如图1-12所示。

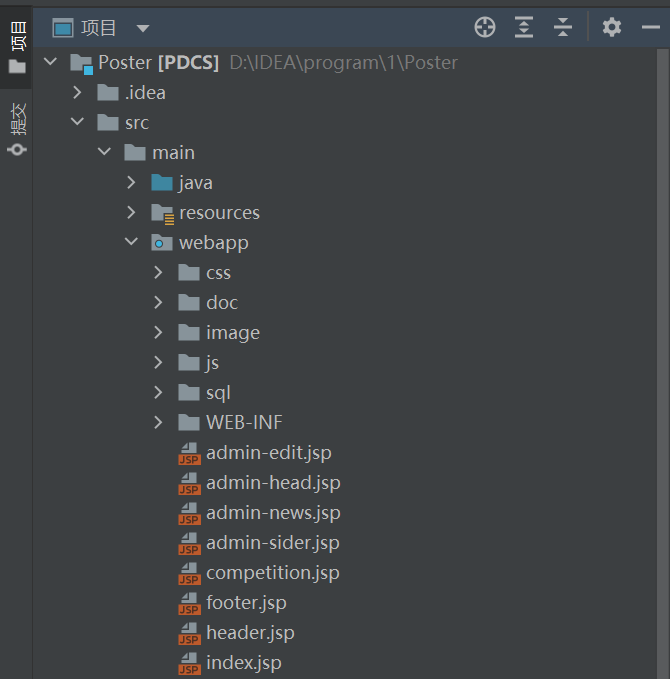


图1-12 webapp文件夹结构

PDCS项目中，各种身份信息存放在MySQL数据库中，图片等在数据库中只保存了文件名。

### **1.4.5 数据库连接配置**

PDCS项目中使用druid数据源连接池以及JDBCTemplate工具，需要在项目中配置数据库连接信息文件。

在PDCS项目的文件夹src\main\resources上右击，选择New→File，在弹出的窗口中输入druid.properties，回车创建文件。同理在文件夹src\test\resources创建文件druid.properties。在这两个新文件中编写配置信息，配置代码如图1-13所示。

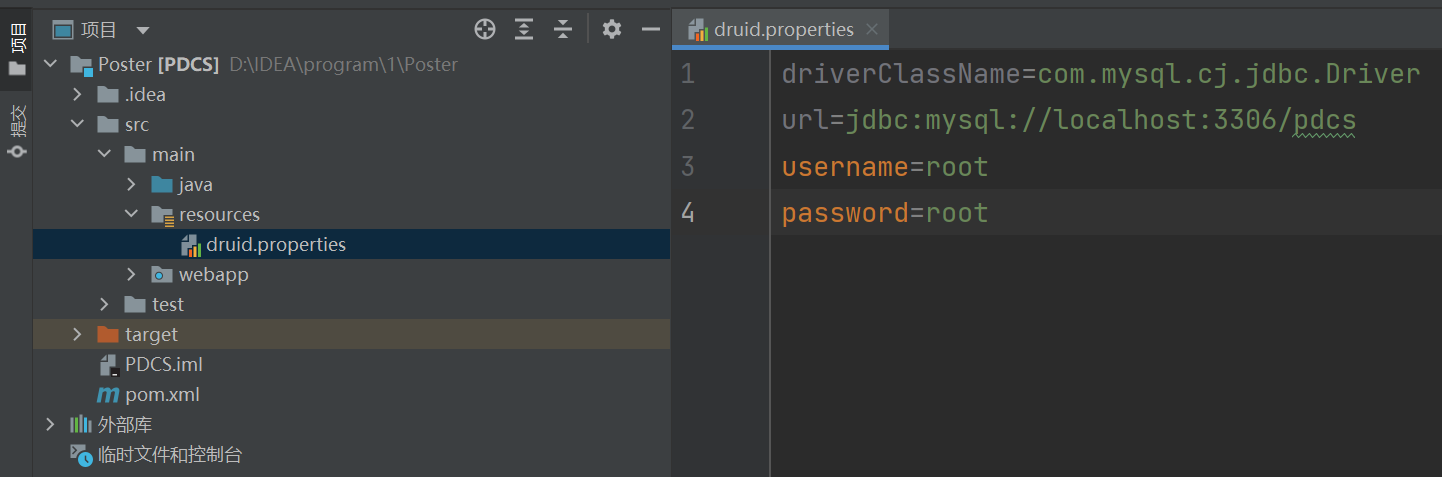


图1-13 数据库连接配置文件

PDCS项目中使用druid数据源连接池，需要设计一个工具类供项目中的数据存取类使用，该工具类命名为JDBCUtils。

在PDCS项目的com.example.utils包上右击，选择New→Java Class，在弹出的窗口中输入JDBCUtils，回车创建文件并输入源代码。

源程序：JDBCUtils.java文件

package com.example.utils;

import javax.sql.DataSource;

import java.io.IOException;

import java.io.InputStream;

import java.sql.Connection;

import java.sql.SQLException;

import java.util.Properties;

import com.alibaba.druid.pool.DruidDataSourceFactory;

public class JDBCUtils {

private static DataSource ds ;

static {

try {

//1.加载配置文件

Properties pro = new Properties();

//使用ClassLoader加载配置文件，获取字节输入流

InputStream is = JDBCUtils.class.getClassLoader()

.getResourceAsStream("druid.properties");

pro.load(is);

//2.初始化连接池对象

ds = DruidDataSourceFactory.createDataSource(pro);

} catch (IOException e) {

throw new RuntimeException("找不到druid.properties文件，"

+"请在resources文件夹中创建druid.properties文件", e);

} catch (Exception e) {

throw new RuntimeException("数据库初始化异常", e);

}

}

public static DataSource getDataSource(){

return ds;

}

public static Connection getConnection() throws SQLException {

return ds.getConnection();

}

}

通过上述步骤的实施，PDCS项目的架构已经搭建完毕。