**软件设计说明书**

题 目： 背包问题知识社区系统

学 院： 计算机科学与工程学院

专 业： 计算机科学与技术

班 级： 2019 级卓越工程师班

团队成员： 王玉慧、汤可意、王晨阳

指导教师： 代祖华

目录

[第一部分 软件系统总体架构 1](#_Toc16949)

[1. 系统总体架构 1](#_Toc24632)

[2. 系统功能总览 2](#_Toc22313)

[3. 系统数据处理 3](#_Toc29789)

[4. 系统异常处理 3](#_Toc30685)

[第二部分 数据库设计 5](#_Toc30839)

[1. 数据实体 5](#_Toc16392)

[2. 数据结构 6](#_Toc3367)

[第三部分 接口设计 9](#_Toc31365)

[1. 内部接口 9](#_Toc13231)

[2.外部接口 10](#_Toc17939)

[第四部分 软件重用方案 11](#_Toc20527)

[1. 知识重用方案 11](#_Toc15662)

[2. 方法和标准的重用方案 11](#_Toc9899)

[3. 软件成分的重用方案 12](#_Toc8565)

[4. 类构件实现软件重用方案设计 13](#_Toc28998)

[第五部分 设计关键类的重点服务 15](#_Toc20346)

# 软件系统总体架构

## 系统总体架构

Spring Boot是Spring框架的模块。它用于轻松创建独立的, 生产级的基于Spring的应用程序。它是在核心Spring Framework之上开发的。Spring Boot遵循一个分层的体系结构, 其中每一层都与它下面或它上面的层(层次结构)进行通信.他共有四层:

表示层

业务层

持久层

数据库层

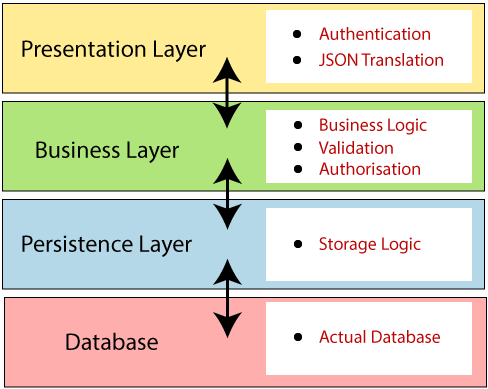


图1.1系统结构

表示层：表示层处理HTTP请求, 将JSON参数转换为对象, 并对请求进行身份验证并将其传输到业务层。简而言之, 它包括视图, 即前端部分

业务层：业务层处理所有业务逻辑。它由服务类组成, 并使用数据访问层提供的服务。它还执行授权和验证。

持久层：持久层包含所有存储逻辑, 并将业务对象与数据库行进行相互转换。

数据库层：在数据库层中, 执行CRUD(创建, 检索, 更新, 删除)操作。

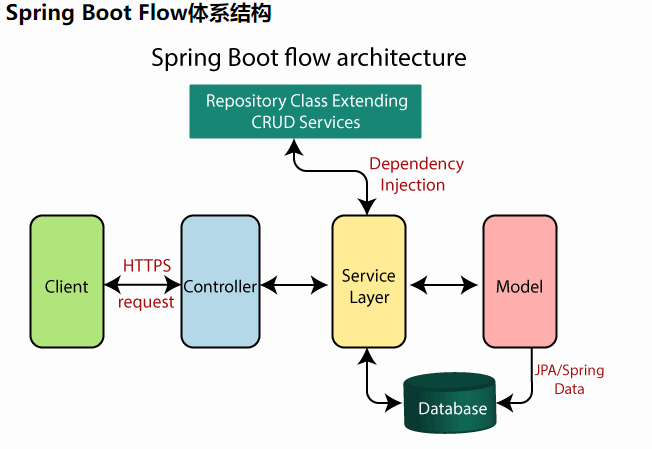


图1.1Spring Boot Flow体系结构

对于前端，我们计划采用layui作为前端框架进行开发，与后端连接起来。layui更多是面向于后端开发者，所以在组织形式上毅然采用了几年前的以浏览器为宿主的类 AMD 模块管理方式，却又并非受限于 CommonJS 的那些条条框框，它拥有自己的模式，更加轻量和简单。layui 定义为“经典模块化”，并非是刻意强调“模块”理念本身，而是有意避开当下 JS 社区的主流方案，试图以尽可能简单的方式去诠释高效。它的所谓经典，是在于对返璞归真的执念，它以当前浏览器普通认可的方式去组织模块。 layui 认为这种轻量的组织方式，仍然可以填补 WebPack 以外的许多场景。所以它坚持采用经典模块化，也正是能让人避开工具的复杂配置，重新回归到原生态的 HTML/CSS/JavaScript本身。

## 系统功能总览

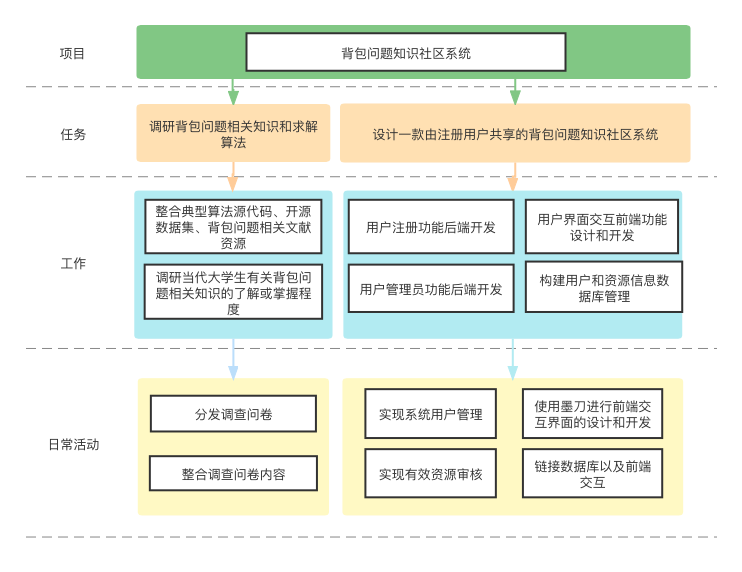


图1.3 系统项目总览

## 系统数据处理

输入输出（input/output，I/O）描述的是在计算机上输入输出数据的操作系统、程序或设备。一般的输入输出设备有打印机、硬盘、键盘和鼠标。实际上，有些设备只有输入功能，如键盘和鼠标；有些设备只有输出功能，如打印机；还有些设备具有输入输出2种功能，如硬盘、磁碟和可写性只读光盘（CD-ROM）。

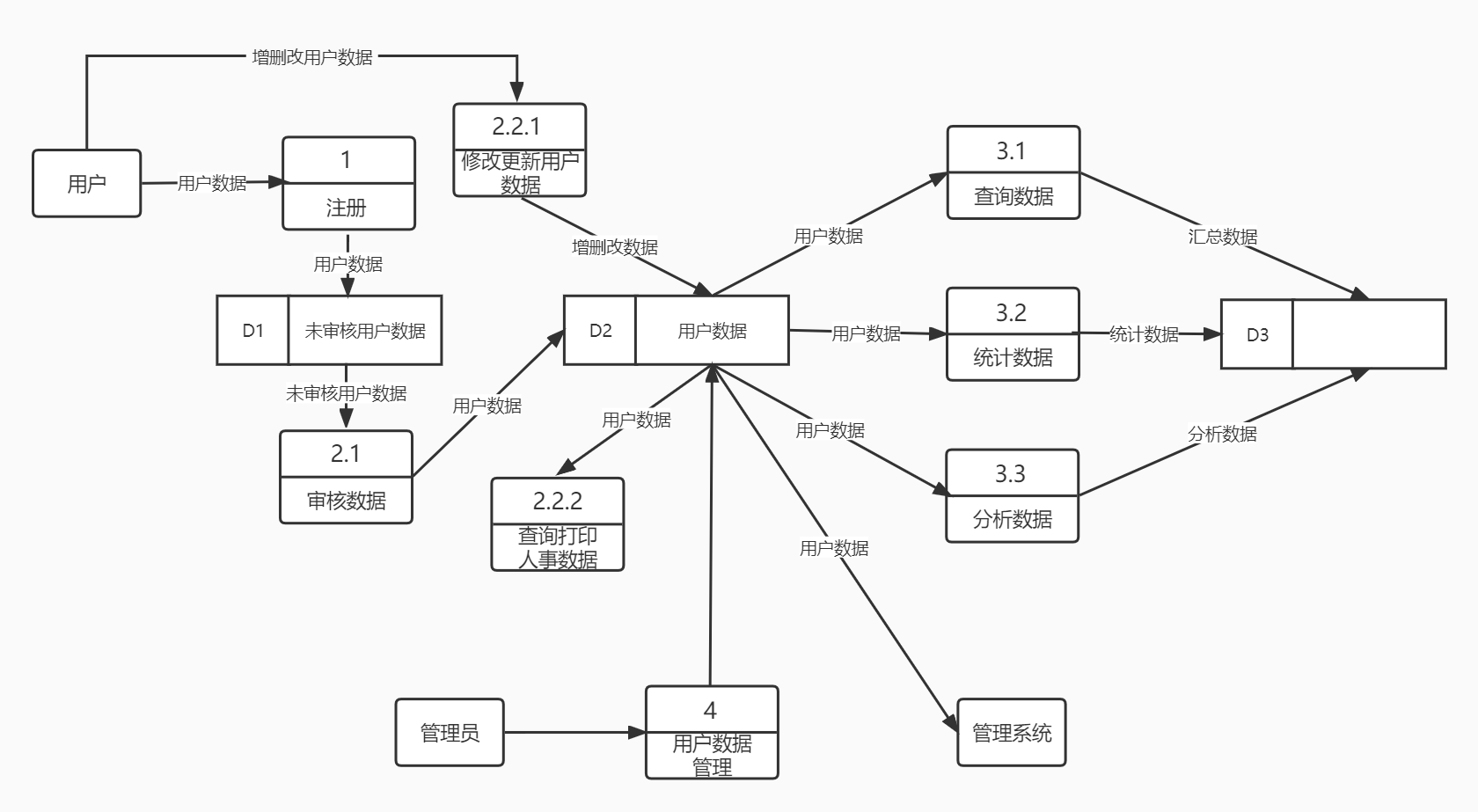


图1.4 数据流图

## 系统异常处理

异常处理（又称为错误处理）是编程语言或计算机硬件里的一种机制，用于处理软件或信息系统中出现的异常状况（即超出程序正常执行流程的某些特殊条件）。其功能提供了处理程序运行时出现的任何意外或异常情况的方法。本系统的异常处理机制如下：

（1）在应用程序遇到异常情况（如被零除情况或内存不足警告）时，就会产生异常。发生异常时，控制流立即跳转到关联的异常处理程序（如果存在）。

（2）如果给定异常没有异常处理程序，则程序将停止执行，并显示一条错误信息。

（3）可能导致异常的操作通过try关键字来执行。

（4）程序可以使用throw关键字显式地引发异常。

（5）异常对象包含有关错误的详细信息，其中包括调用堆栈的状态以及有关错误的文本说明。

（6）即使引发了异常，finally块中的代码也会执行，从而使程序可以释放资源。

（7）异常处理使用try、catch和finally关键字来尝试可能未成功的操作，处理失败，以及在事后清理资源。异常处理通常是防止未知错误产生所采取的处理措施。

采用异常处理机制的好处是开发者不用再绞尽脑汁去考虑各种错误，这为处理某一类错误提供了一个很有效的方法，使项目软件开发效率大大提高。

异常由公共语言运行库(CLR)、第三方库或使用throw关键字的应用程序代码生成。用户也可以用自定义的异常处理类来扩展php内置的异常处理类。

# 数据库设计

概念模型用于信息世界的建模，与具体的DBMS无关。为了把现实世界中的具体事物抽象、组织为某一DBMS支持的数据模型。人们常常首先将现实世界抽象为信息世界，然后再将信息世界转换为机器世界。也就是说，首先把现实世界中的客观对象抽象为某一种信息结构，这种信息结构并不依赖于具体的计算机系统和具体的DBMS，而是概念级的模型，然后再把模型转换为计算机上某一个DBMS支持的数据模型。实际上，概念模型是现实世界到机器世界的一个中间层次。

信息世界中包含的基本概念有实体和联系。

（1）实体

客观存在并可相互区别的事物称为实体。实体可以是具体的人、事、物，也可以是抽象的概念或联系。例如，一个学生、一门课、一个供应商、一个部门、一本书、一位读者等都是实体。

（2）联系

在现实世界中，事物内部以及事物之间是有联系的，这些联系在信息世界中反映为实体内部的联系和实体之间的联系。实体内部的联系通常是组成实体的各属性之间的联系。两个实体型之间的联系可分为3类，一对一联系，(1:1)；一对多联系(1:n)；多对多联系(m:n)。

概念模型是对信息世界建模，所以概念模型应该能够方便、准确地表示信息世界中的常用概念。概念模型的表示方法很多，其中最为常用的是P.P.S.Chen于1976年提出的实体，联系方法(Entity-Relationship Approach)简记为E-R表示法)。该方法用E-R图来描述现实世界的概念模型，称为实体-联系模型，简称E-R模型。

## 数据实体

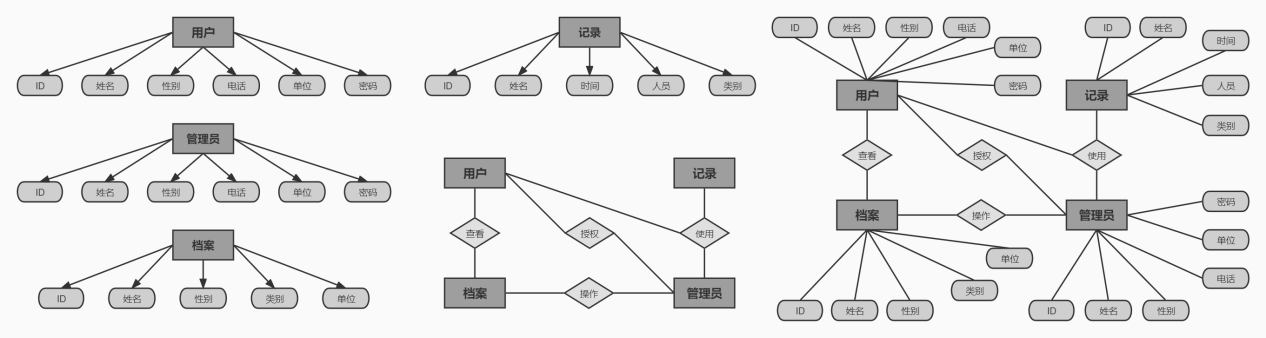


图2.1 E-R图

## 数据结构

数据项条目，用于标识实体。数据字典是数据库的重要部分，它存放有数据库所用的有关信息，对用户来说是一组只读的表。它是关于数据信息的集合。它是数据流图中所有要素严格定义的场所，这些要素包括数据流、数据流的组成、文件、加工小说明及其他应进入字典的一切数据，其中每个要素对应数据字典中的一项条目。其中，对引用的一些关键字进行说明 : PK（主键 ），FK（外键 ）， Check（检查的范围约束），Not null（不为空值）。

数据表项

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表名 | Academy | | | | |
| 字段名 | 数据类型 | 默认值 | 允许非空 | 自动递增 | 备注 |
| academy | varchar(20) |  |  |  | 学院名称 |
| academy\_id | smallint(6) unsinged |  | NO |  | 学院编号，唯一 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表名 | Student | | | | |
| 字段名 | 数据类型 | 默认值 | 允许非空 | 自动递增 | 备注 |
| student\_id | smallint(12) unsinged |  | NO |  | 学生学号，有唯一性 |
| s\_name | varchar(4) |  | NO |  |  |
| s\_sex | tinyint(1) | 1 |  |  | 学生性别 1:女 0:男 |
| s\_keywords | smallint(20) unsinged |  | NO |  | 学生密码 |
| s\_phonenumber | smallint(20) unsinged |  | NO |  | 学生电话，有唯一性 |
| s\_birthplace | varchar(20) |  |  |  | 学生籍贯 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表名 | Administrator | | | | |
| 字段名 | 数据类型 | 默认值 | 允许非空 | 自动递增 | 备注 |
| Administrator\_id | smallint(20) unsinged |  | NO |  | 管理员ID，有唯一性 |
| ad\_name | varchar(4) |  | NO |  | 管理员姓名 |
| ad\_sex | tinyint(1) | 1 |  |  | 管理员性别 1:女 0:男 |
| ad\_keywords | smallint(20) unsinged |  | NO |  | 管理员密码 |
| ad\_ phonenumber | smallint(20) unsinged |  | NO |  | 管理员电话，有唯一性 |
| ad\_position | varchar(20) |  |  |  | 管理员职位 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表名 | Document | | | | |
| 字段名 | 数据类型 | 默认值 | 允许非空 | 自动递增 | 备注 |
| Documet\_id | smallint(12) unsinged |  | NO |  | 档案ID，有唯一性 |
| do\_classes | varchar(4) |  |  |  | 档案类别 |
| do\_years | smallint(4) unsinged |  |  |  | 档案年代 |
| do\_level | varchar(20) |  |  |  | 档案级别 |
| do\_name | varchar(4) |  | NO |  | 档案名称，有唯一性 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表名 | Recoded Information | | | | |
| 字段名 | 数据类型 | 默认值 | 允许非空 | 自动递增 | 备注 |
| Recoded\_time | varchar(20) |  |  |  | 记录时间 |
| Recoded\_name | varchar(4) |  |  |  | 记录人姓名 |

数据结构

|  |  |  |
| --- | --- | --- |
| 1 | Academy | academy、academy\_id |
| 2 | Student | student\_id、student\_id、s\_sex、s\_keywords、s\_phonenumber、s\_birthplace |
| 3 | Administrator | Administrator\_id、ad\_name、ad\_keywords、ad\_ phonenumber、ad\_position |
| 4 | Document | Documet\_id、do\_classes、do\_years、do\_level、do\_name |
| 5 | Recoded Information | Recoded\_time、Recoded\_name |

# 接口设计

本系统软件的接口设计遵循六大基本原则：分别是：单一职责原则、依赖倒置原则、接口隔离原则、迪米特法则、里氏替换原则、开闭原则。这6个原则是建立稳定的，灵活和健壮设计的基础，但设计的时候还要根据实际情况考虑。

（1）单一职责原则(SRP)：该原则要求我们在设计类或者接口的时候。尤其在设计接口的时候把职责分清楚，通常一个职责不是单一的方法，是一类方法的组合。

（2）依赖倒置原则：为高层模块不应该依赖底层模块，他们都应该依赖抽象。抽象不能依赖细节。细节应该依赖抽象，这个原则要求设计的时候尽量用抽象（抽象类或者接口）把各个模块独立开来，实现解耦，使各模块相对独立。简单来说就是要用面向接口设计。

（3）接口隔离原则(ISP)：接口分两种，分别为类实例接口和类接口。这个原则依赖建立在最小的接口之上，依赖自己需要的接口。

（4）迪米特法则(LOD)：也称为最少知识原则。就是一个对象应该对其他对象有最少的了解。这个法则的本质就是解耦，解耦是有限度的不能为了解耦而解耦。

（5）里氏替换原则：（所有引用基类的地方，都能透明地使用其子类的对象。）简单来说就是父类出现的地方，替换为子类不会产出异常。里氏替换原则的好处就是增强程序的健壮性，保持程序的兼容性。

（6）开闭原则：软件的实体如类，模块和函数应该对扩展开发，对修改关闭。这个原则要求我们设计的系统扩展性好，因为需要不会一直不变的，我们需要应对的永远是变化。

## 1. 内部接口

内部接口也被称为嵌套接口，这意味着在另一个接口内声明一个接口。例如，Entry接口声明在Map接口中。由于各模块之间相互独立又彼此关联，系统主要通过函数调用实现各部分连接。

## 2.外部接口

（1）用户界面：在界面设计上，应做到简单明了，易于操作，并且要注意到界面的布局，应突出的显示重要以及出错信息。外观上也要做到合理化，考虑到用户多对Windows风格较熟悉，所以该系统尽量向这一方向靠拢。

（2）软件与硬件接口：本系统设有人机操作界面，考虑到操作简单，易于管理方面，主要硬件与接口设备为pc、鼠标、键盘。而软件接口主要以Windows平台为基本平台。

# 软件重用方案

## 知识重用方案

知识如此珍贵，最好能重复使用：不仅自己使用，也让别人使用。让自己人不用花代价，就可以获得知识。于是，知识传播也变得重要起来。人类的祖先“智人”之所以能够战胜“尼人”，一个重要的原因就是“智人”可以用语言传播知识。设备远程维护最好让设备开发商参与，就是因为他们便于知识复用。我们国家制造业的优势在于市场大——市场大的好处就是便于知识的重用。在有些行业中，企业越来越大。其中重要的“粘合剂”就是知识。

一般来说，知识需要在使用的过程中才能不断丰富、完善。这就叫“从实践中来，到实践中去”。实践多了，才能走向成熟。知识成熟了，使用的风险就小。所以，阿波罗计划拒绝采用不成熟的技术。进入现代工业社会以后，企业都要求按照标准化进行生产。而“标准”就是一种成熟以后固化起来的知识。在智能化的时代，知识可以固化在计算机里，自动地使用知识；可以在互联网上传播，极大地促进知识的重用。知识被重用的次数多了，获得知识的成本就可以被摊平，从而进一步促进知识的产生。而大数据的时代，能够帮助人类方便地获得更多的知识——甚至包括图像识别这样不容易描述的“感性知识”。所以，认识智能化时代的一个角度，是知识经济。

对于本系统，学习过的知识可以得到充分利用，比如数据库，Java EE，WEB前端技术等。对于我们软件需求分析以及后期软件开发都有很好的作用。

## 方法和标准的重用方案

对于本系统，采用面向对象的方法，对于标准的重用，我们采用国家规定的软件开发规范。整体遵守代码规范，对每个人的编码都进行规范化，使得后期维护方便快捷，代码整体结构清晰。定义通用的接口以及方法，使用过程中直接调用即可。

## 软件成分的重用方案

软件重用分类比较困难，因为软件重用技术众多，一种重用技术可以包括多种重用形式。比如说：框架即可以包括代码级重用，也可以包括设计级重用。有一种分类方法是按照软件重用所应用的领域范围，把重用划分为两种：横向重用和纵向重用。

（1）横向重用是指重用不同应用领域中的软件元素，例如数据结构、分类算法、人机界面构件等。标准函数库是一种典型的原始的横向重用机制。

（2） 纵向重用是指在一类具有较多公共性的应用领域之间进行软部品重用。因为在两个截然不同的应用领域之间实施软件重用非常困难，潜力不大，所以纵向重用才广受瞩目，并成为软件重用技术的真正所在。纵向重用活动的主要包括以下几个步骤：

a. 首先进行域分析。根据应用领域的特征及相似性预测软部件的可重用性。

b. 然后进行软部品的开发。一旦确认了软部件的重用价值，即可进行软部品的开发并对具有重用价值的软部品进行一般化，以便它们能够适应新的类似的应用领域。

c. 最后，软部件及其文档即可进入软部品库，成为可供后续项目使用的可重用资源。

在本系统中，在后期编码的过程中，使用调用库函数，实现代码重用，可以大大提高代码的效率。对于编写的源代码、用户界面的设计、数据等都可以重用，比如源代码的编写中，所要用到的一些前端框架，我们就可以结合自身本系统的需求进行分析，然后进行框架整合，这样就减少了我们的编码工作量，有助于加快项目进度。在UI界面中，可以重用之前的原型设计中设计好的界面，对应进行设计，因为已经对原型进行了调研和试用，所以重用原型设计来设计界面，可以保证我们最后研发出来的产品是满足用户审美的，是符合用户需求的。对于数据重用，比如数据库中表的设计，我们可以对逻辑分析中的E-R图进行加工，根据E-R图进行建表等工作，根据逻辑分析设计数据库。

## 类构件实现软件重用方案设计

利用面向对象技术,可以更方便更有效地实现软件重用。面向对象技术中的“类”,是比较理想的可重用软构件,不妨称之为类构件。类构件有3种重用方式,分别是实例重用、继承重用和多态重用。下面进一步讲述与类构件有关的内容。

类构件的重用方式

（1）实例重用

由于类的封装性,使用者无须了解实现细节就可以使用适当的构造函数,按照需要创建类的实例,然后向所创建的实例发送适当的消息,启动相应的服务,完成需要完成的工作,这是最基本的重用方式。此外,还可以用几个简单的对象作为类的成员创建出一个更复杂的类,这是实例重用的另一种形式。

虽然实例重用是最基本的重用方式,但是,设计出一个理想的类构件并不是一件容易的事情。例如,决定一个类对外提供多少服务就是一件相当困难的事,提供的服务过多会增加接口复杂度,也会使类构件变得难于理解;提供的服务过少,则会因为过分一般化失去重用价值。每个类构件的合理服务数都与具体应用环境密切相关,因此找到一个合理的折衷值是相当困难的。

（2）继承重用

面向对象方法特有的继承性提供了一种对已有的类构件进行裁剪的机制,当已有的类构件不能通过实例重用完全满足当前系统需求时,继承重用提供了一种安全地修改已有类构件,以便在当前系统中重用的手段。

为提高承重用的效果,关键是设计一个合理的、具有一定深度的类构件继承层次结构。这样做有下述两个好处:

a. 每个子类在继承父类的属性和服务的基础上,只加入少量新属性和新服务,这不仅降低了每个类构件的接口复杂度,表现出一个清晰的进化过程,提高了每个子类的可理解性,面且为软件开发人员提供了更多可重用的类构件。因此,在软件开发过程中,应该时刻注意提取这种潜在的可重用构件,必要时应在领域专家帮助下,建立符合领域知识的继承层次。

b. 为多态重用奠定良好基础

c. 多态重用

利用多态性不仅可以使对象的对外接口更加一般化(基类与派生类的许多对外接口是相同的),从而降低了消息连接的复杂程度,而且还提供了一种简便可靠的软构件组合机制,系统运行时,根据接收消息的对象类型,由多态性机制启动正确的方法,去响应一个一般化的清息,从而简化了消息界面和软构件连接过程。

对于本系统我们使用继承重用，对于面向对象语言来说，必须要保证它的可封装性，就要用到继承机制来达到代码复用的目的。对于接口重用，我们考虑用类指针调用派生类的方法来实现接口重用。比如在后期的编码中，用户信息类可以用户类类。

为了提高重用的效果,关键是设计一个合理的、具有一定深度的类构件继承层次结构，这样每个子类在继承父类的属性和服务的基础上,只需要加入少量新属性和新服务,这不仅降低了每个类构件的接口复杂度,表现出一个清晰的进化过程,提高了每个子类的可理解性,而且为软件开发人员提供了更多可重用的类构件。

# 设计关键类的重点服务

系统根据状态图和数据流图可筛选出本项目的关键类及相应重点服务如下：

用户：登录后可以添加发帖，查询发帖等；

管理员：登录后可修改话题、删除话题，用户话题管理等一系列查看功能。