|  |
| --- |
| **学士学位论文** |
| **软件测试用例在线评判** |
| |  |  | | --- | --- | | 学 号： | **20131002723** | | 姓 名： | **王肖辉** | | 学 科 专 业： | **软件工程** | | 指 导 教 师： | **张剑波 副教授** | | 培 养 单 位： | **信息工程学院** | |
| 二○一七年六月 |

**学士学位论文原创性声明**

本人郑重声明：本人所呈交的学士学位论文《软件测试用例在线评判》，是本人在指导老师的指导下，在中国地质大学（武汉）攻读学士学位期间独立进行研究工作所取得的成果。论文中除已注明部分外不包含他人已发表或撰写过的研究成果，本人所呈交的学士学位论文没有违反学术道德和学术规范，没有侵权行为，并愿意承担由此而产生的法律责任和法律后果。

学位论文作者签名：

日 期： 年 月 日

**摘 要**

在现今的软件测试教学中，依然偏重于理论的教学，实践教学的缺乏是影响学生软件测试能力的主要因素。在软件工程中，软件测试的地位越来越重要，设计软件测试用例的优劣，对学生以后项目开发至关重要。目前软件测试实践还限制于设备环境，要完成测试，学生需要安装针对不同语言和不同测试类型的测试工具，且对于测试的结果的评判也限制于人工评判，由于测试结果分散在各自的环境中也不利于汇总分析。所以本论文希望设计一款系统，该系统能够接收用户提交的测试用例脚本，然后由测评机对测试用例进行分析并给出评价结果。

目前有许多的在线测评系统，但是大部分是用来判断代码程序的正确性，及代码执行过程中的信息参数。目前市场上有许多优秀的软件测试框架和工具，但是需要安装环境，且测试结果也不便于持久保存。所以本系统拟结合两者来实现一款软件测试用例在线评判系统，参照目前流行的OJ系统的原理，使用Web开发技术完成开发。利用合适的软件测试工具作为测评机的开发基础，将软件测试工具整合到测评机中，实现对软件测试用例的测评，目前对测试用例的测评以覆盖率为标准，二者通过ApacheMQ发送异步消息进行通信。

在本系统为实现扩展的方便，采用Web 端和测评机相互分离的策略。测评机对测评实现进行了抽象封装，扩展只需要实现测评接口，配置对应的测试环境。

本文的主要工作分三个部分：

（1）确定需求，明确架构。根据目前软件测试教学中存在问题以及测试工具和测试方法的不足，确定系统的需求，系统的核心需求是实现在线测评测试用例；根据需求明确系统的架构，一方面，本系统要实现一个在线测评的功能，所以系统采用Web开发满足在线提交管理；另一方面测评是系统的核心，所以将其与Web端分离开发。因此系统大体架构类似B/S结构：B为Web端，S为测评机。

（2）系统设计与实现。详细设计系统各功能模块的结构及实现流程、用户操作界面、数据库以及主要程序模块的实现方式。

（3）系统测试。根据需求设计测试方案完成测试并对测试结果进行分析，验证系统的实现情况。

**关键词**：在线测评；测评机；ApacheMQ；扩展

**Abstract**

In today's software testing teaching, teacher still place extra emphasis on the theory of teaching.For students , the ability to design test cases is not effectively exercised. Software testing, as an important subject area in software engineering, is critical in practical project development. At present,the practice of software testing is also limited to the device environment.If students want to complete the test, students need to install a variety of test tools. The test results are also limited with the evaluation of artificial, due to scattered test results and their respective environment is also not conducive to statistics. Therefore, this paper hopes to design a system and to achieve the test case online evaluation.

There are many online evaluation systems, but most are used to determine the correctness of the code program and the parameters in the time of code execution. There are many frameworks and tools for software testing, but the installation environment is required and the test results are not easy to keep lasting. Therefore, this system intends to combine OJ and Web development to achieve an online software test case judgments.With referenceing to the current popular OJ system principle and the application of Web development technology to complete the development of the Web. The appropriate software testing tools integrated into the evaluation machine to achieve the evaluation of software test cases.The current evaluation of the task is mainly on the coverage of the statistics. Both send an asynchronous message via ApacheMQ for communication each other.

In this system, the Web and the evaluation machine are separated from each other, so the expansion is more convenient. The web almost is no need to modify, and only need to modify the evaluation machine.The evaluation of the evaluation of the evaluation on the line to achieve an abstract package, the expansion only need to achieve evaluation interface and configure the corresponding test environment.

The main work of this paper is divided into three parts.

Identifying requirements and clarifying the structure. Determining the needs of the system according to the current problems in software testing and the lack of test tools and test methods.In general, the core needs of the system is to achieve online test and test cases; To design the system architecture according to the needs of the system.On the one hand, the system need to achieve an online evaluation of the function, so the system uses Web development to meet the needs of the submission and management online; the other hand, the evaluation is the core of the system, so it will be separated from the Web side development. So the system architecture is similar to the B / S structure: the B is the Web side and S is the evaluation machine.

In the aspect of the design and implementation, I designed the structure and implementation of each function module of the system, the user interface, the database and the realization of the main program module in detail.

At the test stage,in order to verify whether the system to meet the needs,I design the test program to test the system according to the needs.After finishing the test,I analyze the results of the test and determine the completion of the system.

**Keywords:** the evaluation machine; online evaluation; ApacheMQ; expansion

**目 录**

[图清单 8](#_Toc483422522)

[表清单 10](#_Toc483422523)

[第一章 绪论 1](#_Toc483422524)

[1.1 研究背景和意义 1](#_Toc483422525)

[1.2 国内外研究现状 1](#_Toc483422526)

[1.3 研究目标和内容 2](#_Toc483422527)

[1.4 论文的组织结构 3](#_Toc483422528)

[第二章 技术支持 5](#_Toc483422529)

[2.1 开发平台及框架 5](#_Toc483422530)

[2.2 消息通信ActiveMQ 6](#_Toc483422531)

[2.3 覆盖率统计 6](#_Toc483422532)

[2.3.1 Java覆盖率统计 6](#_Toc483422533)

[2.3.2 Python覆盖率统计 7](#_Toc483422534)

[2.4 其他技术 7](#_Toc483422535)

[2.4.1 Ant 7](#_Toc483422536)

[2.4.2 Jsoup 8](#_Toc483422537)

[2.4.3 Beetl 8](#_Toc483422538)

[2.5 本章小结 8](#_Toc483422539)

[第三章 系统需求分析和结构设计 9](#_Toc483422540)

[3.1 需求分析 9](#_Toc483422541)

[3.1.1 系统目标 9](#_Toc483422542)

[3.1.2 系统需求 9](#_Toc483422543)

[3.2 系统结构设计 10](#_Toc483422544)

[3.2.1 逻辑架构 10](#_Toc483422545)

[3.2.2 物理架构 11](#_Toc483422546)

[3.2.3 运行架构 12](#_Toc483422547)

[3.2.4 开发架构 13](#_Toc483422548)

[3.2.5 功能结构 15](#_Toc483422549)

[3.3 本章小结 15](#_Toc483422550)

[第四章 系统详细设计 16](#_Toc483422551)

[4.1 系统功能模块设计 16](#_Toc483422552)

[4.1.1 系统管理模块 16](#_Toc483422553)

[4.1.2 系统维护模块 18](#_Toc483422554)

[4.1.3 单元测试模块 19](#_Toc483422555)

[4.2 用户界面设计 22](#_Toc483422556)

[4.2.1 设计原则 22](#_Toc483422557)

[4.2.2 界面设计 22](#_Toc483422558)

[4.3 数据库设计 26](#_Toc483422559)

[4.3.1 界面设计 26](#_Toc483422560)

[4.4 程序模块设计 27](#_Toc483422561)

[4.4.1 测评模块 27](#_Toc483422562)

[4.5 安全设计 28](#_Toc483422563)

[4.5.1 用户权限划分 28](#_Toc483422564)

[4.5.2 数据的一致性 29](#_Toc483422565)

[4.6 本章小结 29](#_Toc483422566)

[第五章 系统实现和测试 30](#_Toc483422567)

[5.1 系统整体实现 30](#_Toc483422568)

[5.2 关键技术实现 31](#_Toc483422569)

[5.2.1 代码生成 31](#_Toc483422570)

[5.2.2 消息通信 31](#_Toc483422571)

[5.2.3 测评机 32](#_Toc483422572)

[5.3 系统测试 33](#_Toc483422573)

[5.3.1 测试环境与测试方案 33](#_Toc483422574)

[5.3.2 系统测试数据与过程 34](#_Toc483422575)

[5.3.3 系统测试结果与分析 39](#_Toc483422576)

[5.4 本章小结 40](#_Toc483422577)

[第六章 总结和展望 41](#_Toc483422578)

[6.1 总结 41](#_Toc483422579)

[6.2 展望 42](#_Toc483422580)

[致谢 43](#_Toc483422581)

[参考文献 44](#_Toc483422582)

# 图清单

[图 1-1在线测评系统工作流程图表格 3](#_Toc483337698)

[图 2-1 Spring Web MVC 处理请求流程 5](#_Toc483337712)

[图 2-2JaCOCO覆盖率统计 6](#_Toc483337713)

[图 2-3 JaCOCO 覆盖率统计报告 6](#_Toc483337714)

[图 2-4 nose-coverage覆盖率统计结果 7](#_Toc483337715)

[图 3-1系统逻辑架构图 11](#_Toc483337723)

[图 3-2 系统物理架构图 12](#_Toc483337724)

[图 3-3 系统运行架构图 13](#_Toc483337725)

[图 3-4 Web端相关的框架与技术 13](#_Toc483337726)

[图 3-5测评机相关技术 14](#_Toc483337727)

[图 3-6系统功能结构 15](#_Toc483337728)

[图 4-1系统管理模块结构 16](#_Toc483337732)

[图 4-2系统管理模块类图 17](#_Toc483337733)

[图 4-3用户登录、注册流程图 18](#_Toc483337734)

[图 4-4系统维护模块结构图 19](#_Toc483337735)

[图 4-5系统维护模块类图 19](#_Toc483337736)

[图 4-6单元测试模块结构图 20](#_Toc483337737)

[图 4-7单元测试模块类图 20](#_Toc483337738)

[图 4-8单元测试模块类图 21](#_Toc483337739)

[图 4-9主界面 23](#_Toc483337740)

[图 4-10登录界面 23](#_Toc483337741)

[图 4-11注册界面 23](#_Toc483337742)

[图 4-12测试案例列表 24](#_Toc483337743)

[图 4-13提交测试用例 24](#_Toc483337744)

[图 4-14测评结果反馈界面 25](#_Toc483337745)

[图 4-15测试案例管理 26](#_Toc483337746)

[图 4-16系统数据库结构 26](#_Toc483337747)

[图 4-17测评程序模块类图 28](#_Toc483337748)

[图 5-1 系统整体实现 30](#_Toc483337749)

[图 5-2测评机消息通信实现 32](#_Toc483337750)

[图 5-3测评机执行 33](#_Toc483337751)

[图 5-4 提交测试用例 35](#_Toc483337752)

[图 5-5 测评结果 36](#_Toc483337753)

[图 5-6添加测试案例 37](#_Toc483337754)

[图 5-7添加成功 37](#_Toc483337755)

[图 5-8用户接口测试 38](#_Toc483337756)

[图 5-9测试案例接口测试 38](#_Toc483337757)

[图 5-10登录和注册错误验证 39](#_Toc483337758)

# 表清单

[表 3-1 项目开发包结构 14](#_Toc483338014)

[表 4-1 表功能 27](#_Toc483338019)

[表 5-2测试需要初始化的表 34](#_Toc483338022)

[表 5-3 功能测试结果 39](#_Toc483338023)

[表 5-4 数据接口测试接口 40](#_Toc483338024)

# 绪论

## 研究背景和意义

软件测试是软件工程范畴中一个重要学科[1]，为了强化培养学生在软件开发领域的实践、设计以及创新能力，必须加强软件测试的实践教学。

目前，在软件测试教学中实践教学依然是薄弱的环节，其一般流程为教师布置实践内容，学生要完成测试需要在各自的计算机上安装相应的测试软件以及搭建测试环境，对于测试的结果老师也不方便统计与分析。一方面，需要为每一台机器配置一套测试框架，造成了资源的浪费；另一方面，测试的结果分散在各自的测试环境中，很难统一管理，不利于软件测试实践教学。

如今，在线考试或者测试模式被广泛应用于职业选拔和教学中[2-4]，相比于传统的测试，在线测试具有低成本、高效率等优势。考虑到软件测试的重要性和目前存在的问题，本文拟采用在线测试的模式，研究实现一款软件测试案例在线测评系统，在线评判将有效的提高软件测试实践教学效率，降低教学成本，同时也为学生提供了方便，有利于他们软件测试的实践学习。

## 国内外研究现状

软件测试作为软件开发进程的关键一个环节，其目标是验证软件是否正确、功能是否完成、软件是否安全可用以及软件的质量是否合格，所以软件测试至关重要。目前，软件测试大多采用自动化测试或人工测试的方式，软件测试用例的在线测评尚未兴起。

在线判题系统（Online Judge，OJ）起初是用于国际的程序设计大赛，使用（OJ）来进行判题和排名。现在各地高校纷纷建立自己的OJ系统，用于程序设计的训练和比赛。现在比较有名气的OJ如：北京大学（POJ）、西班牙Valladolid大学（UVA）等，该类系统能够编译并执行代码，并用预先准备好的数据去验证代码的正确性，及执行过程中的参数（内存占用等），并返回验证结果。但尚未有针对软件测试的在线测评系统。

从软件测试国内使用现状上来看，目前主要的测试工具用来进行单元测试、集成测试、回归测试和性能测试。对于单元测试和集成测试而言，目前市场上有许多的测试工具，如java的Junit和TestNG，C++的Google Test等，通过这些工具包，搭建测试执行环境，完成测试工作。回归测试是对修改代码的再次验证，热门的回归测试工具有stingWhiz 和Sahi等。主流的性能测试工具如负载测试的LoadRunner和Web测试的Jmeter，通过模拟实际用户的操作行为进行性能监控。要使用这些测试工具完成测试不仅需要耗费安装机器资源，也需要花费成本学习如何搭建测试环境，如何运用测试工具。上述这些工具虽然能满足软件测试的需求，但用在教学实验平台方面会受到机房单台机器的硬件资源限制，不能模拟复杂的测试环境。从软件测试发展的趋势上，近些年软件测试从单机模式向云测试模式转变 [5-6]。云测试是一种基于云计算技术的新型测试方案，它将软件测试或者自动化软件测试工具作为一种服务提供给使用者。云测试平台可以提供多种浏览器作为用户交互的客户端，通过网络接收用户编写的测试脚本，在测试环境中运行用户的测试脚本进行测试，最后再通过网络将测试结果传输给用户。

## 研究目标和内容

软件测试教学环节中，测试用例的设计是学生对软件测试技术掌握程度的直观反映。长期以来，对测试用例的执行和评判都是由教师和学生根据案例特点人工评判，容易出现漏判、误判、逻辑覆盖率低的情况。所以本文主要研究目标为：开发一套软件测试在线测评系统，该系统以OJ系统为基础，并结合软件测试工具，借鉴在线评测系统的工作原理，采用JAVA WEB开发技术，实现软件测试用例在线评判。该系统要求设计一种计算机所能接受的测试用例格式，制定评分标准，具备常规Web系统的用户管理、系统配置等基本功能；能够自动解析并执行学生上传的测试用例，计算测试数据的覆盖率，以此作为测试用例的评价标准[7]，实时统计并给出评判分数，从而实现计算机自动判别测试用例的功能。

本论文的主要研究内容是线测评系统的工作原理[8-11]，测评机的测评机制及系统的扩展。具体来说，包含三个部分：测评机与测评系统通信，测评机的实现和系统的扩展。测评机和测评系统的通信使用消息总线服务Apache ActiveMQ来实现。测评机是系统的核心，它实现了对测评任务的接收、测试用例的评判的执行、测评结果的统计处理反馈。测评机的实现需要结合软件测试工具以及覆盖率统计工具。系统的扩展性包含两个方面，一方面是对测试语言支持的扩展，另一方面是对测评机的扩展，当用户量过多时可以增加测评机的数量来减轻负载。图 1-1 展示了系统的大体工作流程，教师可以上传测试案例，学生可以在线编写测试用例并提交，测评机分析提交，反馈测试用例分析结果，教师和学生可以查看结果。

图 1-1在线测评系统工作流程图表格

## 论文的组织结构

本论文共分为六章，其组织结构与主要内容如下：

第一章 介绍本论文研究的主要目标、内容、现状和组织结构。

第二章 这一章介绍本系统实现过程中应用到的主要技术和框架，其中Web开发框架采用Spring+SpingMVC+Mybatis，Web端和测评机间的通信采用消息总线 Apache ActiveMQ，测试覆盖率统计使用JaCOCO（Java覆盖率统计）和Nose（Python覆盖率统计），测试代码的执行使用自动化工具Ant，使用jsoup生成生成测试用例代码。

第三章 本章主要分析系统，明确需求，设计系统的架构包括逻辑架构涉及系统功能、物理架构涉及部署、运行架构涉及运行时概念、开发架构涉及程序包以及系统的功能结构。

第四章 本章主要是系统的详细设计，各功能模块的设计，主要模块包括：系统的管理模块、单元测试模块和系统维护模块，数据库设计，包含数据库E-R图及数据库表字典，操作界面设计、程序模块设计和安全性设计。

第五章 本章主要涉及系统的实现和系统的测试。介绍了系统实现的关键技术包含测试用例代码生成、Web和测评机间的通信实现和测评机的实现，测试方面确定测试的环境、方案以及对测试结果的分析。

第六章 本章主要阐明本系统存在的一些不足，及改进设想，同时对本系统做一个小结。

# 技术支持

## 开发平台及框架

本系统以eclipse MARS.2作为开发工具，以Spring+SpringMVC+Mybatis作为Web开发框架[12]。

Spring是一个轻量级的Java开发框架，它起初主要是为了简化Java EE的企业级应用开发，与过去重量级的企业级应用开发相比，Spring框架更加简单灵活。它所提供的面向切面编程和控制反转是Spring框架的核心功能。面向切面编程（AOP）实现了公共功能的模块化处理，提高了代码的可重用性。控制反转（IOC）

是对象构建的一种策略，不直接进行对象的构建，而是通过注入依赖关系，框架通过依赖关系构建所需要的对象。

SpringMVC作为一个请求驱动的Web框架，其处理控制器的实现策略，与其他的请求驱动的Web框架在总体思路上是相似的，通过分离流程控制逻辑与具体的Web请求处理逻辑。DispatcherServlet它负责接收并处理所有的Web请求，只不过针对具体的处理逻辑，它会委托给它的下一级控制器去实现即Controller。SpringMVC处理请求流程如下图 2-1：

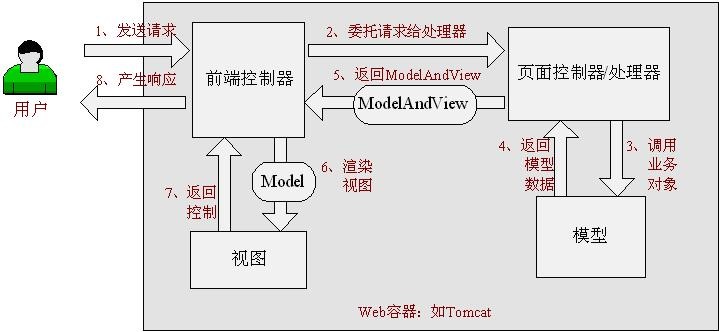


图 2-1 Spring Web MVC 处理请求流程

MyBatis是一个持久化层框架，它隐藏了繁琐的配置和JDBC的代码，可以通过xml 或注解来实现数据与Mapper接口之间的映射。

## 消息通信ActiveMQ

ActiveMQ是一种支持JMS规范的中间件。它提供了一种松散耦合的灵活的应用程序集成机制，程序直接彼此之间不需要直接通信，而是通过消息中间件来作为中介者进行通信。本系统应用Spring集成的ActiveMQ，使用Spring JMS实现异步发收消息。

## 覆盖率统计

### Java覆盖率统计

JaCOCO是一款针对java的单元测试覆盖率统计工具[13]，它的使用方式比较多样，既可以通过Eclipse插件的形式使用，也可以通过嵌入到maven或ant中使用，目前JaCOCO支持不同标准的覆盖率统计,其中有常用的分支覆盖，还有一些像指令级别的覆盖、行覆盖、方法类覆盖。

JaCOCO 作为Java代理运行，它负责在运行测试时对字节码进行检测。在执行测试时，JaCOCO会渗透到每一行指令并且显示哪些行被执行。JaCOCO接收来自JVM工具接口中的事件，使用ASM（一个通用的Java字节码操作和分析框架）进行代码检测，收集覆盖率数据。流程如下图2-2 所示：

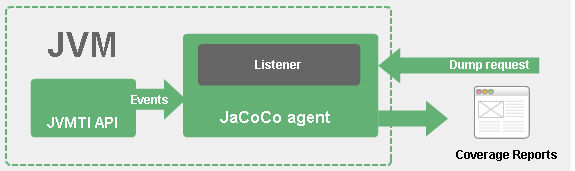


图 2-2JaCOCO覆盖率统计

本系统对于Java语言的测试用例覆盖率统计使用JaCOCO。使用JaCoCo统计Java单元测试覆盖率结果如下图2-2：

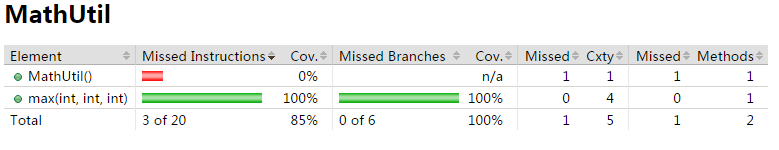


图 2-3 JaCOCO 覆盖率统计报告

### Python覆盖率统计

Nose 作为一个知名的Python单元测试框架[14]，优于unittest 。Nose 可以自动识别继承于unittest.TestCase的单元测试，并且也可以测试非继承于unittest.TestCase的单元测试。Nose为编写单元测试提供了全面的API。Nose也可以通过内置插件实现代码覆盖、文档测试、错误查找等等，同时也可以自定义开发插件来满足测试需求。本系统对于Python语言的测试用例覆盖率统计使用Nose。使用Nose的cover插件统计Python单元测试覆盖率结果如下图2-4：

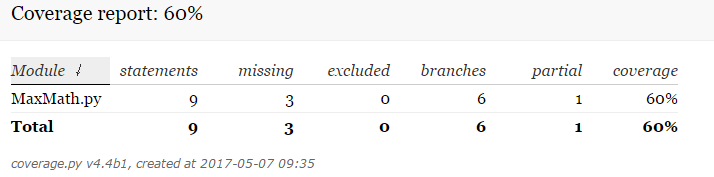


图 2-4 nose-coverage覆盖率统计结果

## 其他技术

### Ant

Ant作为一个跨平台的构件工具，它能够完成工程的自动化构建和部署等功能。Ant的构件文件使用XML格式。构件文件有两个主要元素，project是构件文件的基本元素；基本任务执行单元写在target中。在本系统中Java测试的编译、执行都是通过构建文件来执行，java测试覆盖率统计JaCOCO也是通过集成到构建文件来执行的。

### Jsoup

Jsoup作为一个Java 版本的Html解析工具包[15]，可以以URL、文件以及字符串作为输入解析出DOM。可以通过DOM或选择器进行标签或元素的操作。本系统中对于HTML形式覆盖率结果的解析由Jsoup来完成，使用Jsoup将覆盖率结果以List对象的形式存储。

### Beetl

Beetle作为新一代Java模板引擎的典范[16]，与其他Java模板引擎相比，它使用更加的方便，而且功能也比较多，在性能上更是由于其他模板引擎，学习起来也比较简单，它也可以和各种Web框架进行整合，在扩展方面Beetle支持自定义方法，同时也支持自定义占位符。本系统测中测试用例的代码生成由Beetle来完成，根据设计的用例、题目案例的信息作为输入，参照对应语言的模板生成相应的测试用例代码。

## 本章小结

这一章节主要介绍系统开发过程中所应用到的主要技术，如Web端开发过程中使用IOC生成所需要的对象，使用AOP获取当前对象。使用nose的cover插件统计Python测试用例的覆盖率等等。这些技术是支撑整个系统不可或缺的一部分。

# 系统需求分析和结构设计

## 需求分析

### 系统目标

软件测试教学环节中，测试用例的设计是学生对软件测试技术掌握程度的直观反映。长期以来，对测试用例的执行和评判都是由教师和学生根据案例特点人工评判，容易出现漏判、误判、覆盖率低的情况。如果有一款能借助计算机实现测试用例自动评判的软件系统，将能显著改善教学效果。鉴于此，本系统将借鉴在线测评系统的工作原理，实现一款软件测试用例在线评判系统。此系统能够接收来自Web端学生提交的测试用例脚本，然后结合测试案例对测试用例进行覆盖率分析。

### 系统需求

***系统功能性需求***

**单元测试：**

学生可以根据条件筛选测试案例，提交测试用例，查看提交记录，查看所有提交记录，修改个人资料等。获取学生提交的测试用例，生成测试用例代码。执行测试生成测试覆盖率报告，解析测试报告，持久化测试结果信息。向用户反馈测试消息及结果。

**系统管理**

用户信息管理，涉及到用户的登录及注册，用户的增删改查，用户权限的管理。系统本身的一些属性的管理，如系统的版本，是否允许注册，支持语言等等。

**系统维护**

本部分功能涉及测试相关的元素的维护，包含测试案例、用户提交的添加、删除、修改等操作。还有对不同语言测试工具的管理，对测评机的维护等。

***系统非功能性需求***

**高可用性**

在系统运行过程中可能出现一些特殊情况如服务器宕机，导致测评机死机，为了保持系统的正常运行，系统提供多个测评机，当一个测评机停止工作后不至于影响系统的正常运行。

**可扩展性**

对于一个在线测评系统需要支持多种语言，语言扩展的过程中应当尽量少的修改源程序代码，以模块的形式添加新语言的支持。

## 系统结构设计

### 逻辑架构

从逻辑结构上来看分为四个层次：

**用户展现层**

本层主要为用户提供操作界面，用户目前有三种：一为学生，学生视图提供学生操作的界面，包括案例的选择页面，测试用例的提交页面，提交的查看页面等等；二为教师，教师视图提供了教师操作的界面，教师所提交的测试案例的管理，优秀测试用例的查看等；三为管理员，管理员视图提供管理员的操作界面，包括学生所拥有操作页面，测试案例管理页面，用户的管理页面，提交用例管理页面，系统的基本设置页面和测试工具配置页面等等。

**服务层**

本层次主要为用户展现层提供服务，响应用户提交的各种请求。服务主要分三大模块：系统管理模块，本模块主要包含用户登录注册功能、用户的管理（添加、删除、修改、查询）和用户权限管理等等；系统维护模块，本模块主要包含测试案例的管理、用例提交的管理、测试工具管理和系统设置等等；单元测试模块，本模块主要包含测试用例提交请求处理，测试结果反馈等。

**数据层**

数据层提供对系统相关数据的持久化存储和访问，主要包含三个方面的数据：测试案例数据，存储教师提交的测试案例，包含了测试案例相关的属性；用例提交，存储学生提交的测试用例信息，包含了测试案例的信息、用例、测评结果等相关信息。用户数据，存储学生和教师用户相关信息。

**测评机**

测评机为执行学生提交测试用例的核心，当测评机的消息接受收到web端传来的测试任务创建消息后，创建测试任务并执行，为避免同步线程安全问题，此处采用synchronized来解决，保证一次只执行一个测评任务。待任务执行完毕后，使用解析工具将执行结果解析成java的List对象并将结果持久化到用例提交数据库中，然后发送测试任务执行完毕的消息给Web端。同时对提交的测试用例做筛选排序处理，将优秀的测试用例持久化到数据库。系统的整体逻辑架构如图3-1所示：



图 3-1系统逻辑架构图

### 物理架构

本系统分两部分，一为Web，此部分提供了用户操作界面的构建以及用户提交请求的处理，是整个项目的前台；二为测评机，此是测评系统的核心部分，负责完成测评任务的处理及反馈工作。部署上二者相互独立，通过ActiveMQ来进行通信，二者共同完成测评工作。物理架构图如下图3-2：



图 3-2 系统物理架构图

### 运行架构

学生提交测试用例，Web端将提交的测试用例及相关信息存入数据库，同时发送异步消息给测评机，测评机收到消息创建测试任务，此过程为同步线程安全的，完成测试任务后持久用例测试结果数据到用例提交数据库，同时发送消息给Web，向用户展示测评结果。运行架构图如下图3-3：



图 3-3 系统运行架构图

### 开发架构

图3-4展示了Web端开发中要使用的技术、图3-5展示了测评机开发所使用的技术。本系统的静态包组织结构如下表3-1；开发视图从技术及组织结构的角度介绍了系统的开发结构。



图 3-4 Web端相关的框架与技术



图 3-5测评机相关技术

表 3-1 项目开发包结构

|  |  |  |
| --- | --- | --- |
| 项目名 | 包名 | 说明 |
| caseonline.engine |  | 测评机 |
|  | caseonline.judger.engine.application | 入口 |
|  | caseonline.judger.engine.core | 测评核心实现 |
|  | caseonline.judger.engine.exception | 异常 |
|  | caseonline.judger.engine.mapper | 数据接口 |
|  | caseonline.judger.engine.messenger | 消息处理 |
|  | caseonline.judger.engine.model | 数据实体 |
|  | caseonline.judger.engine.util | 工具 |
| caseonline.web |  |  |
|  | caseonline.judger.web.aspect | 切面 |
|  | caseonline.judger.web.controller | 控制 |
|  | caseonline.judger.web.exception | 异常 |
|  | caseonline.judger.web.mapper | 数据接口 |
|  | caseonline.judger.web.messenger | 消息处理 |
|  | caseonline.judger.web.model | 数据接口 |
|  | caseonline.judger.web.service | 数据服务接口 |
|  | caseonline.judger.web.util | 工具 |

### 功能结构

大体介绍系统的功能结构。系统管理主要是用户管理和系统设置，单元测试是系统的主要功能，实现测试用例的测评分析，系统维护是测试相关数据的管理。系统功能结构如下图所示：



图 3-6系统功能结构

## 本章小结

本章主要介绍了系统的需求和设计。确定了实现在线测评测试用例的核心目标，也由此明确了系统的需求；采用架构设计的五视图法实现系统的设计，从逻辑视图、物理视图、运行视图、开发视图的角度分析设计系统的架构，功能结构图展示了系统设计的功能目标。总的来说这一部分是从需求和设计角度来为系统的开发提供全方位的指导。

# 系统详细设计

## 系统功能模块设计

### 系统管理模块

**模块结构**



图 4-1系统管理模块结构

**模块类图**



图 4-2系统管理模块类图

AccountsController控制器：用来处理用户的登录、注册以及修改个人资料等请求。

UserManagerController控制器：用来处理用户管理的请求，用户的添加、删除、修改、更新。

AdminController控制器：用来处理系统设置的请求，支持语言的修改，系统是否允许注册等。

UserService：提供用户相关的数据操作服务。

GeneralService：提供系统一般设置的数据服务。

**模块功能**

主要实现用户信息管理和系统的设置。

用户管理

1.用户的注册、登录

用户输入用户验证信息，通过验证通过则注册或登录成功，未通过则显示错误信息，用户重新操作。其流程图如下图 4-3 所示：



图 4-3用户登录、注册流程图

2.用户管理

实现对用户信息的添加、删除、修改、查询操作

用户权限管理

根据角色的不同将请求提交到不同的控制器，由控制器显示不同的操作界面或是返回不同用户角色属性到页面上。对于测试提交有哪些人能够查看设计的测试用例实现通过角色属性进行判断，只有教师、管理员和提交者本身能够看到。

系统设置

是否允许注册，支持语言管理等

### 系统维护模块

**模块结构**



图 4-4系统维护模块结构图

**模块类图**



图 4-5系统维护模块类图

SubmissionManagerController控制器：测试用例提交管理请求处理器。

TestcaseManagerController控制器：测试案例管理请求处理器。

SubmissionService：提供测试用例提交数据管理服务。

TestcaseService：提供测试案例数据管理服务。

模块功能

本模块主要用于测试案例、测试用例提交以及测试工具的管理等工作。

### 单元测试模块

**模块结构**



图 4-6单元测试模块结构图

**类图**



图 4-7单元测试模块类图

TestcaseController控制器：接收用户的测试用例提交请求。

SubmissionService：提供提交信息服务。

TestcaseService：提供测试案例信息服务。

Dispatcher: 测评调度器，完成测评任务的创建与执行。

ApplicationDispatcher:消息处理器，处理测评过程中消息,如测试相关文件创建完成等。

**单元测试模块的整体流程**



图 4-8单元测试模块流程图

**模块功能**

单元测试模块主要负责测评的工作，从用户选择测试案例，提交测试用例，交由测评机进行测试、覆盖率统计、及测试结果处理（优秀测试用例的筛选，结果消息的反馈），用户查询所有测试用例提交信息等等。

## 用户界面设计

### 设计原则

一个成功的应用系统，不仅拥有健全的功能还要有良好的操作体验。系统最终的使用者是用户，所以一个好的系统还应当有一个友好的操作界面。在设计是应当遵循一定的界面设计原则：

用户原则：界面设计首先要确定用户类型，针对用户的特点来设计。例如一个节日营销网站，页面应当是色彩斑斓的，而一个信息管理系统，应当简洁朴素的。

简单原则：尽量减少用户的记忆负担，一个完整的任务，仅仅需要两三步操作就可以完成，而不需要用户去记忆繁琐的操作步骤。

直接操作：在任何一个界面，用户都能以最少的点击到达另一个想要进入的界面。或许添加连接可以到达这一目的，但过多的连接、按钮会影响用户对界面功能把握，所以要把握好这个度。

### 界面设计

**主页面**

主页面为系统的首页，此页面是用户操作的起点，在此可以登录、注册、查看问题以及查看提交。界面如下图4-9所示：

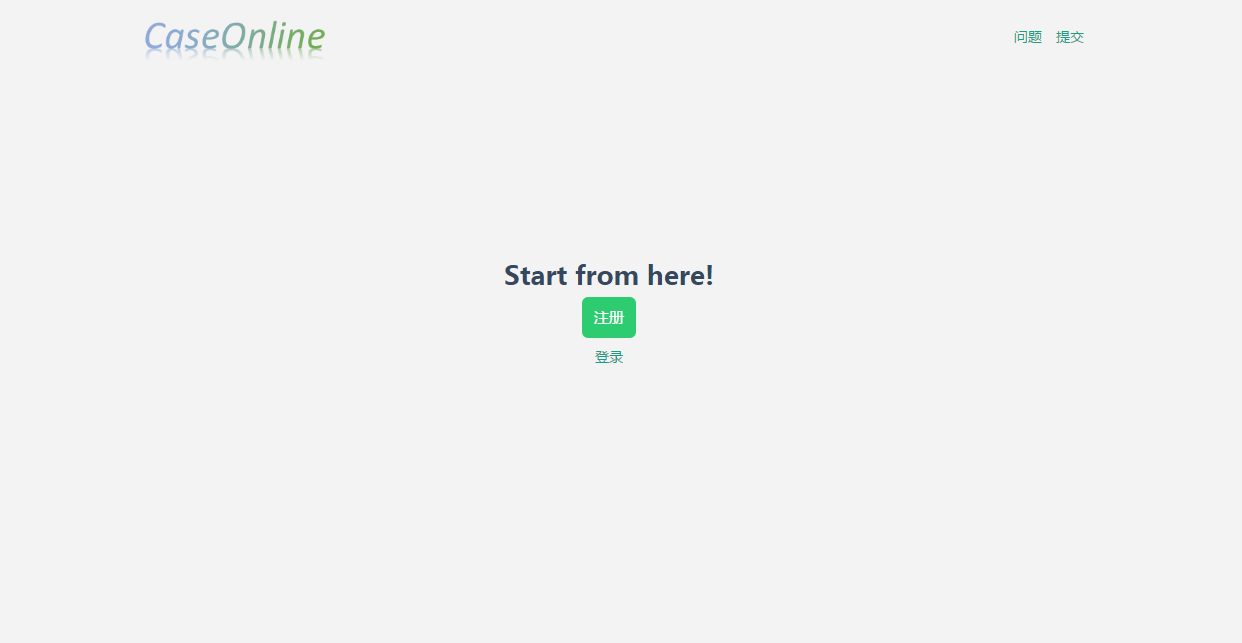


图 4-9主界面

登录及注册界面

用户用来完成登录和注册，且二者之间可以通过一个连接进行相互跳转，方便已有账号用户直接到达登录页面，没有账号用户快速到达注册页面。界面如下图4-10和4-11所示：

|  |  |
| --- | --- |
| 图 4-10登录界面 | 图 4-11注册界面 |

**测试案例界面**

测试案例展示界面，主体部分展示测试案例的条目，用户可以通过搜索或直接点击分类类型快速定位到所希望的测试案例。界面如下图4-12所示：



图 4-12测试案例列表

**提交测试用例界面**

通过一个遮盖层向用户提供测试用例的编写和提交，这样可以避免用户进行其他不必要的操作，简化用户的提交操作。

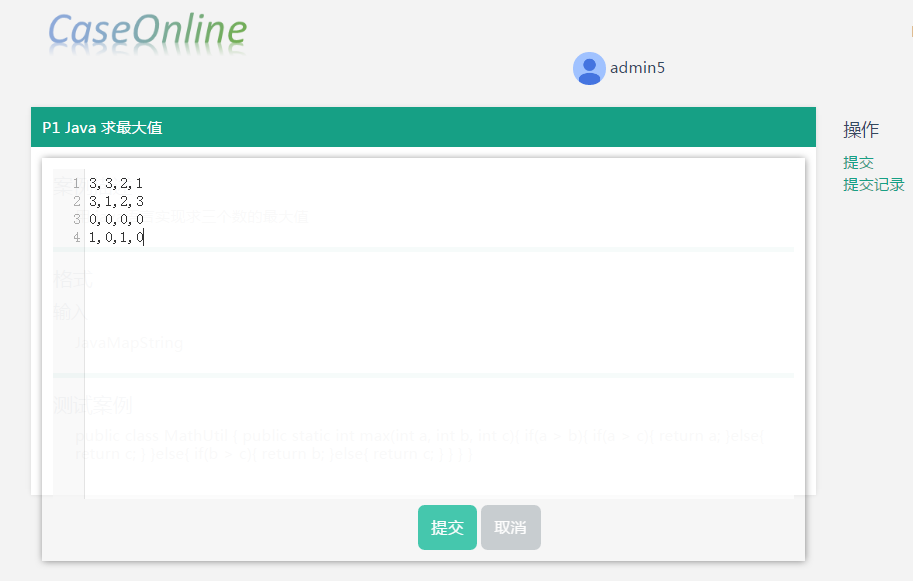


图 4-13提交测试用例

**结果反馈界面**

结果展示界面向用户反馈了测评结果，分三部分：最上端向用户展示了测试用例测评的总体信息；中间向用户展示了测评过程的日志记录，便于用户查看详细的测评过程。最下端向用户展示了自己编写的测试用例，便于用户回忆自己提交的测试用例。



图 4-14测评结果反馈界面

**系统管理界面首页**

管理首页向管理员展示了系统的大体概况，用户情况、测试案例情况、提交状况以及测评机信息。

**测试案例管理页面**

此界面向管理员展示了测试案例列表，提供了批量删除，按条件查询，翻页，详情等操作按钮或连接。



图 4-15测试案例管理

**其他界面设计**

其他界面设计风格与上述界面类似，受篇幅有限，不再列出。

## 数据库设计

### 界面设计



图 4-16系统数据库结构

本系统总共涉及14张表，每张表的作用如下表所示：

表 4-1 表功能

|  |  |  |
| --- | --- | --- |
| Name | Code | Comment |
| 测试案例表 | | |
| coj\_testcase | coj\_testcase | 测试案例表 |
| coj\_testcase\_category | coj\_testcase\_category | 测试案例类型 |
| coj\_testcase\_rela\_cate | coj\_testcase\_rela\_cate | 测试案例和类型关系表 |
| 用户表 | | |
| coj\_user\_role | coj\_user\_role | 用户角色类型分类 |
| coj\_user | coj\_user | 记录用户信息 |
| 提交用例表 | | |
| coj\_submission | coj\_submission | 提交测试案例 |
| coj\_excellence\_usecase | coj\_excellence\_usecase | 优秀的测试用例设计 |
| 数据字典 | | |
| coj\_usecase\_format | coj\_usecase\_format | 向用户展示测试用例编写格式 |
| coj\_language | coj\_language | 所支持的语言，及与语言相关联的信息 |
| coj\_judge\_result | coj\_judge\_result | 测评结果 |
| coj\_option | coj\_option | 系统属性 |

## 程序模块设计

### 测评模块

**测评模块类图**



图 4-17测评程序模块类图

测评程序模块在设计上，将整个测评过程抽象为三步：测评前的准备、测评和测评后的结果分析。为语言扩展的方便，定义统一接口ICoverage，其中定义三个函数，来完成对应工作。JavaCoverage和PythonCoverage是对java和Python语言覆盖率分析过程的实现，测试前期准备工作工具类Preprocessor实现测试案例，测试用例文件的创建工作，UsecaseGenerator类实现根据测试用例脚本生成测试用例代码，Bean和Param是UsecaseGenerator的辅助类。 覆盖率结果解析工具HtmlParse实现对测试用例覆盖率报告的解析。要实现某一语言的测试用例覆盖率统计扩展，需要实现ICoverage接口，实现接口中测评的三个方法。

## 安全设计

### 用户权限划分

目前系统提供了三种用户类型，管理员、教师、学生。管理员拥有最高权限，可以管理及维护测试案例，测试用例提交等等；教师拥有二级权限，可以管理维护自己提交的测试案例查看优秀的测试用例等；学生拥有最低权限，查看测试案例、测试用例提交和自己提交的用例脚本等。目前的设计方案为根据用户角色类型的不同，其可见操作界面不同，处理请求的控制器Controller映射不同，返回的不同的页面且控制器会把角色类型参数传递到页面上，页面在浏览器中解释的时候会根据角色参数的不同解释不同的页面元素。

### 数据的一致性

本系统数据库设计中并未使用外键，在相关联的数据表进行添加和删除操作时，往往会出现不一致导致出错情况。所以为了避免这种情况，在进行删除操作时，同时考虑将其相关联的表项进行删除操作。

## 本章小结

这一章中详细介绍了系统的设计，从各个方面展示了系统的实现的方案。功能模块设计展示了系统的模块划分、各模块的功能和模块的出入参数和处理流程；数据库设计展示了系统的数据库基础，确定了系统的数据存储结构；用户界面设计展示了系统的门面，确定系统界面设计的简介明了的整体风格。程序模块设计主要涉及到一些程序代码的设计，以提高代码的可重用性；安全性设计主要考虑两个方面，一是用户权限，保障系统数据的安全性，避免一般用户对系统造成破坏。二是数据库，由与相关表之间没有设计外键，没有自动级联删除，所以需要通过代码来实现级联删除，保证数据的一致性，避免系统获取数据异常。

# 系统实现和测试

## 系统整体实现

系统整体上采用B/S结构实现，使用Spring+SpringMVC+Mybatis开发框架来实现，MySQL作为数据库，Tomcat8.0作为服务器，MQ作为消息服务。使用bootstrap作为页面开发框架。整体实现结构如下图5-1所示：



图 5-1 系统整体实现

## 关键技术实现

### 代码生成

测试用例提交时，由于编辑框没有代码提示，所以学生直接写代码不方便，另外直接编写测试用例代码也容易出错，考虑上述两个方面的原因，实际上提交的是测试脚本而不是测试代码，所以在执行测试时，需要根据脚本数据生成测试代码，即以测试用例脚本数据以及对应测试案例作为输入参数，以测试用例代码文件为输出结果。在代码生成工具的选择上，通过对比目前主流模板引擎Freemarker和Velocity，Beetle性能以及操作上都要优于它们，所以代码生成采用Beetle模板引擎，具体实现方式如下：

第一步：编写模板文件（\*.template）。

第二步：加载模板文件目录，获取模板。

第三步：封装Bean，将输入参数数据封装成Bean，作为下一步的绑定参数。

第四步：绑定Bean执行。

### 消息通信

为了提高系统的可用性，防止测评时出现问题导致系统死机的情况，本系统将测评机作为一个独立的整体。测评机与其他部分的通信通过MQ消息服务器来实现，具体实现过程为：学生提交测试用例后，首先将测试用例相关属性信息存入数据库中，然后向测评机发送提交创建消息，测评机收到消息后，首先根据提交ID获取提交用例信息，然后创建测评任务（此部分为了避免线程同步问题，采用synchronized来保障一次只有一个测评任务在执行），执行测试用例的测评，完成测评后，将测评日志信息和结果存入数据库，同时将测评进度和结果信息打包成消息反馈给Web端，向用户展示测评结果。相关实现点如下图5-2所示：



图 5-2测评机消息通信实现

### 测评机

测评机是在线测评系统的核心，其主要功能就是接收测评请求、创建测评任务（执行前准备、执行测评任务、覆盖率解析）、结果处理及反馈。接收测评请求的实现是通过MQ消息服务器来实现；测评任务的创建，测评请求消息体包含由提交ID，创建测评任务时可以根据提交ID来获取提交的测试用例以及对应的测试案例信息，根据获取的信息完成相应的文件创建等前期准备工作，执行测评任务通过Java的Running.exec()来执行相应语言的测评指令，执行完成之后会生成对应的覆盖率报告（针对目前支持的语言Java和Python，来说报告是html形式），通过html解析工具（Jsoup）完成覆盖率报告的解析；测评完成后将测评的过程日志、结果存入数据库，根据测评的覆盖率选出有优秀的测试用例存入数据库中，

在测评的过程中每完成一个阶段都会给Web端发送相应的消息，告知测评的进度。由于测评机是独立的，所以可以同时部署多个测评机，以此来提高系统的效率，减轻测评机的负载。其实现流程如下图5-3所示：



图 5-3测评机执行

## 系统测试

### 测试环境与测试方案

**系统测试环境：**

系统：Window7

Java：jdk1.8

Python：3.6

数据库：MySQL

服务器：tomcat 8.0

Ant：apache-ant-1.9.9

Apache ActiveMQ: 5.14.3

**测试方案：**

系统测试大致分三块进行测试：

***系统的核心功能测试***

描述：测试用例的测评，主要测试系统的测评功能，能否正确分析测试用例，获取测评结果。

流程：

第一步：选择测试案例，根据类型或名称筛选测试案例。

第二步：编写测试用例并提交，查看测评结果及测评日志。

第三步：查看提交列表。

***系统相关实体的管理功能测试***

描述：测试数据库实体的添加、修改、删除，查询。

流程：

第一步：输入操作实体参数。

第二步：执行操作。

第三步：结果对照。

***数据服务接口测试***

描述：测试数据层的数据操作接口。

流程：

第一步：测试参数环境配置。

第二步：编写测试类。

第三步：执行分析执行结果。

***错误处理测试***

描述：测试程序对错误输入参数，错误操作的处理。

流程：

第一步：分析可能出错的操作。

第二步：执行错误操作。

第三步：查看错误处理是否与设想的一致。

### 系统测试数据与过程

**数据准备**

通过sql脚本完成数据库的初始化，涉及的数据表如下：

表 5-2测试需要初始化的表

|  |  |
| --- | --- |
| 名称 | 说明 |
| coj\_user\_role | 角色 |
| coj\_user | 用户 |
| coj\_usecase\_format | 用例格式 |
| coj\_testcase\_category | 案例类型 |
| coj\_option | 系统属性 |
| coj\_language | 语言 |
| coj\_judge\_result | 测评结果 |

测试使用测试数据库testcaseonline，使用设计好的脚本完成数据初始化。

**测试过程展示**

受篇幅限制，本文中仅展示部分测试过程。

***提交测试用例测试***

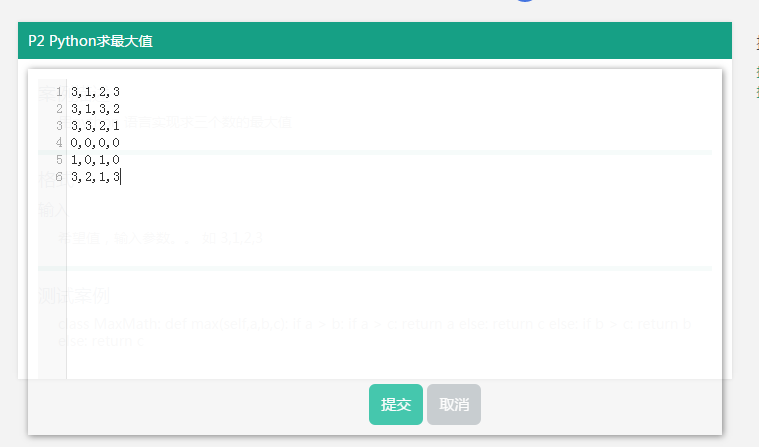


图 5-4 提交测试用例



图 5-5 测评结果

***系统元素信息管理***

测试案例：

添加：

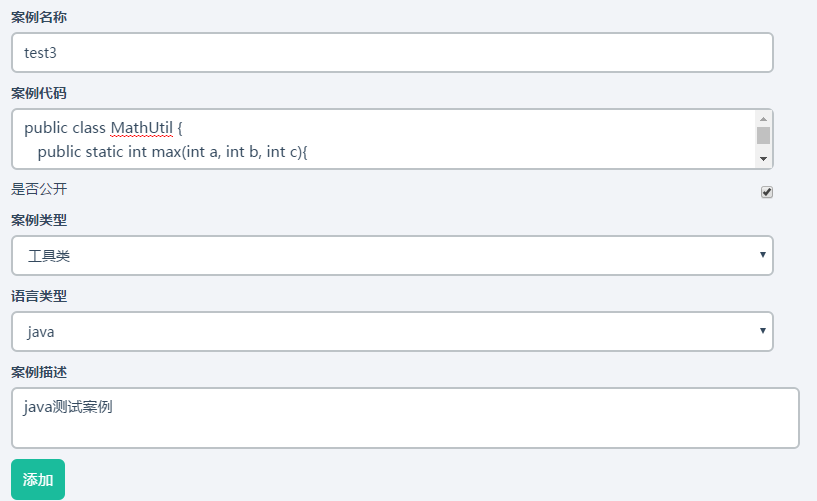


图 5-6添加测试案例

添加成功

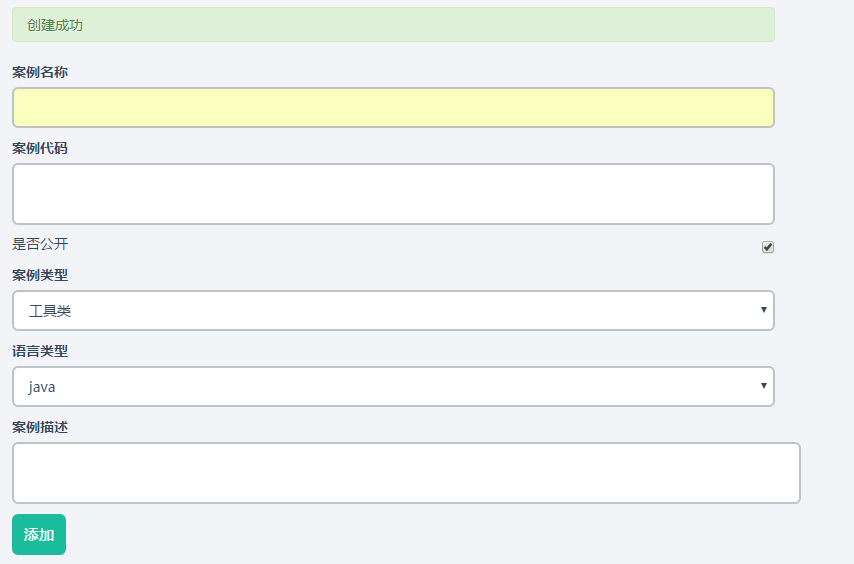


图 5-7添加成功

***接口测试***

用户角色接口：

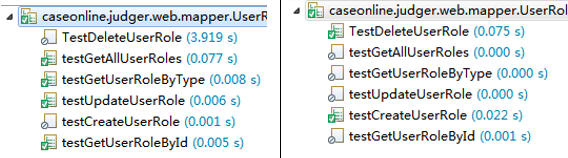
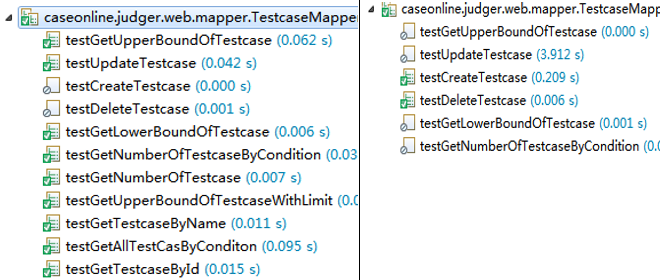


图 5-8用户接口测试

测试案例接口：



|  |
| --- |
| 图 5-9测试案例接口测试 |

***错误测试***

登录和注册



图 5-10登录和注册错误验证

### 系统测试结果与分析

**功能测试：**

表 5-3 功能测试结果

|  |  |
| --- | --- |
| 测试功能点 | 测试结果 |
| 测试案例的查询筛选 | 正常 |
| 测试用例的提交获取测评结果 | 正常 |
| 测试用例提交的查询筛选 | 正常 |
| 用户的管理（添加、删除、修改、查询） | 正常 |
| 测试案例的管理（添加、删除、修改、查询） | 正常 |
| 测试用例提交（查询、删除） | 正常 |
| 系统设置 | 正常 |
| 用户操作限制 | 正常 |
| 验证（测试案例代码、用户注册信息等） | 正常 |
| 提交用例排名 | 正常 |
| 用户的登录注册 | 正常 |
| 测试用例检查验证 | 正常 |

**数据接口测试：**

表 5-4 数据接口测试接口

|  |  |  |
| --- | --- | --- |
| 接口 | 说明 | 测试结果 |
| ExcellenceUsecaseMapper | 优秀测试用例提交接口 | 正常 |
| JudgeResultMapper | 测评结果接口 | 正常 |
| LanguageMapper | 支持语言接口 | 正常 |
| OptionMapper | 系统属性接口 | 正常 |
| SubmissionMapper | 测试用例提交接口 | 正常 |
| TestcaseCategoryMapper | 测试案例类型接口 | 正常 |
| TestcaseMapper | 测试案例接口 | 正常 |
| TestcaseRelaCateMapper | 测试案例类型关系接口 | 正常 |
| UsecaseFormatMapper | 测试用例格式接口 | 正常 |
| UserMapper | 用户接口 | 正常 |
| UserRoleMapper | 用户角色接口 | 正常 |

就当前测试，系统基本上可以正常工作，实现所设计的功能。但是测试上难免有所疏漏，或许系统还有一些没有测试出的Bug，这就有待以后使用过程中发现了。

## 本章小结

在这一章中主要讲解了系统的整体实现以及系统中关键技术点的实现方式，包括如何根据测试用例数据代码的生成、Web是如何与测评机进行消息通信的以及测评机实现细节。然后介绍了系统的测试，包含了系统测试环境与测试方案，测试需要的数据环境，简要的测试过程展示，最后是对测试过程中的结果进行统计与分析。

# 总结和展望

## 总结

本文主要根据在线判题系统的原理、结合软件测试工具和Web开发技术，开发出一款能够进行软件测试用例的在线测评系统。在线判题系统基本原理就是通过Web页面选择试题提交，具体对题提交答案的判定由判题引擎来完成。所以本文主涉及两个方面的开发探究，一是Web端的开发，一是测评机的开发。

测评机：目前确定的评判标准是单元测试的覆盖率，单元测试覆盖率统计工具有java语言的JaCOCO，python 语言的nose-cover等等，这些工具以源码及单元测试类作为输入参数，以覆盖率报告作为输出结果。所以测评机需要做的就是：根据测试任务请求创建文件，然后使用指令执行覆盖率统计工具，得出覆盖率结果。同时考虑到测评机对支持语言的扩展性，将测评的工作抽象封装成对应的三步，文件创建、执行、结果解析。添加语言支持，需要继承测评接口实现这三个函数。

Web：Web的开发采用SSM框架开发，包含四个层次：控制层XXXController用来处理具体的用户请求，本系统的请求大致分两类，一类是XXXManagerController这一类型的控制器用来处理管理员的请求管理，另一类就是用来处理用户请求及公用请求等等。服务层XXXService为控制层提供调用方法，实现对下层数据处理的封装。数据层XXXMapper为服务层提供了数据操作接口，实现了数据的基本操作。用户展现层即页面提供用户的可操作界面，前后端通过Ajax进行数据通信。

Web和测评机通过ActiveMQ服务实现数据通信，测评任务的创建通知以及测评结果的反馈都是以MQ为通信通道。二者操作同一个数据库，通过MQ发送的仅仅是一个类似于指令或通告的消息，具体如何执行后续工作都有二者来独自根据消息来执行。

目前的开发成果，可以使用于针对单元测试案例设计的测试用例的测评工作，且支持Java和Python语言。

## 展望

目前系统可以完成单元测试测试用例的覆盖率的测评，受覆盖率统计工具的限制，支持的覆盖率标准较少，如Python的nose-cover 仅仅支持分支覆盖率的统计，所以可以考虑自己设计实现一种单元测试覆盖率统计工具[17]。另一方面在支持语言扩展上，目前需要手动去配置新语言单元测试覆盖率统计环境，且需要改动测评机代码，一种理想的扩展方案是，将新语言的测评实现进行封装做成一个插件，直接在系统管理页面上配置实现扩展，或者是将测评实现打包，通过页面添加扩展时，向测评机发送下载包的指令，将新语言的测评包下载到服务器，并关联到测评机，以实现对新语言的支持。另外在前端功能上可以添加班级管理的功能，教师可以按班级布置测试任务，班级间的结果数据对比等等。此外此系统使用于单元测试，可以在此基础上进行开发，实现对性能测试的支持。

# 致谢

本系统设计过程中，得到了许多的帮助。很是感谢张剑波老师的指导，从系统设计到论文的编写，张老师都给出了不少的建议。另外张老师严格的教学方式和耐心的指导以及做事的严谨性对我的提高有不少帮助。还要感谢研究生学长在系统设计上的帮助。

另外还要感谢我身边的朋友和同学们，感谢他们为我提供了一个良好的学习环境，当然还要感谢自己的坚持和努力。

# 参考文献

[1] 王秀.谈软件工程中软件测试的重要性及方法[J]. 天津成人高等学校联合学报,2004,(02):76-78+81.

[2] 张盈. 在线考试评判系统的设计与实现[D].四川大学,2005.

[3] 卢依宁. 基于静态分析的Java单元测试教学反馈系统[D].南京大学,2016.

[4] 王迎. 自动化测试服务平台的设计与实现[D].北京交通大学,2015.

[5] 荀江萍. 浅谈基于云计算的软件测试[J]. 无线互联科技,2014,(05):45.

[6] 熊娇. 新型软件测试技术研究与实现[D].电子科技大学,2008.

[7] 焦安涛. 测试用例综合评价模型的研究[D].昆明理工大学,2009.

[8] 黄晓华,沈健,常晋义,张明新. 基于Online Judge与HTML批注技术的实验教学平台设计[J]. 计算机与现代化,2014,(11):117-121.

[9] 王腾,姚丹霖.Online Judge系统的设计开发[J]. 计算机应用与软件,2006,(12):129-130+137.

[10] 何迎生,罗强. Online Judge评判内核的设计与实现[J]. 吉首大学学报(自然科学版),2010,(06):37-39.

[11] Huanda Lu,Liangzhong Fan,Zhenyu Shu Laboratory of Information and Optimization Technologies Ningbo Institute of Technology, Zhejiang University Ningbo, Zhejiang Province, ChinaXin Yu,Qiyu Liu,Xinjie Yu Laboratory of Information and Optimization Technologies Ningbo Institute of Technology, Zhejiang University Ningbo, Zhejiang Province, China. Online Judge System under a Cloud Computing Environment[A]. Hubei University of Technology, China.Proceedings of 2010 Third International Conference on Education Technology and Training(Volume 2)[C].Hubei University of Technology, China:,2010:3.

[12] SSM框架笔记http://www.kancloud.cn/digest/lunaticssm/122056[OL]

[13] JaCoCo：分析单元测试覆盖率的利器https://www.ibm.com/developerworks/cn/java/j-lo-jacoco/[OL]

[14] nose introduction http://pythontesting.net/framework/nose/nose-introduction/[OL]

[15] Jsoup中文文档 http://www.open-open.com/jsoup/parsing-a-document.htm[OL]

[16] Beetl中文文档 http://ibeetl.com/guide/#beetl[OL]

[17] 魏光新,苏丽. 逻辑覆盖测试工具的设计与实现[J]. 计算机工程与应用,2000,(05):106-109.