

**本科毕业论文**

**基于Scrapy框架的计算机专业学习资源网站**

**吴伟杰**

**201525050420**

|  |  |
| --- | --- |
| 指导教师 | **司国东 副教授** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学院名称 | **数学与信息学院** |  | 专业名称 | **网络工程** |
| 论文提交日期 | 2019年4月 日 |  | 论文答辩日期 | 2019年5月 日 |

摘　　　　要

本文主要介绍“基于Scrapy框架的计算机专业学习资源网站”的开发目的、设计思想和具体实现方法等内容。本系统是一个提供计算机专业理论知识学习与计算机语言编程实践的平台，目标用户是有学习计算机专业知识的需求的人群。面向用户的主要功能有：计算机专业知识学习资料的检索、计算机语言编程题目在线评判。计算机相关学习资料来源主要有：网络爬虫抓取各大博客文章并使用数据库索引正文等内容、网站内用户整理并发布学习资料。计算机语言编程题目由网站内具有题目管理权限的用户编写题目描述、测试用例，解答题目的用户编写解题源码并上传到后端，处于系统后端独立运行的代码评判器编译、运行代码后将运行结果呈现给用户。该系统的学习资料与编程题目拥有分类标签，学习资料与题目能够通过标签直接关联。与常见的通用搜索引擎、在线评判网站相比，本系统对计算机专业更具有针对性，理论与实践相结合，对用户学习效率的提高有所帮助。

本系统采用MVVM设计模式，前端是以Vue.js为主的一系列框架与插件开发的SPA。后端采用微服务架构，划分为4个模块：服务注册中心、配置服务器、资源服务器、代码评判器，支持分布式部署、集群。系统主要基于Spring体系的Spring Cloud、Spring Data等系列的框架开发。其中，数据持久层JPA的具体实现选用Hibernate框架；对外提供的API采用REST风格，具体实现基于Spring MVC框架。本系统的学习资料等资源存储、检索功能基于Elasticsearch，系统用户信息、编程题目相关数据、运行时产生的数据存储在关系型数据库MySQL中，并在应用层引入基于内存的数据结构存储数据库Redis实现缓存等功能，用户提供的图片、文档资料等文件采用了第三方服务——阿里云对象存储OSS。代码评判器通过消息中间件RabbitMQ并采用AMQP协议与资源服务器传输数据。网络爬虫是本系统外部资源的重要来源，能够提供大量站外资源供本系统用户检索，模块基于Python语言Scrapy框架开发，使用Redis实现任务队列与URL去重。

关键词：Scrapy；Spring；在线评判；轻量级；

**Tourist Base Support System Based on Data Validator Architecture Model Group**

XXX

(College of Software Engineering, South China Agricultural University, Guangzhou 510642, China)

**Abstract:**

**Key words:** Scrapy; Spring; Online Judgement;

目 录

[1 前言 1](#_Toc5055617)

[1.1 研究背景 1](#_Toc5055618)

[1.2 研究意义 2](#_Toc5055619)

[1.3 系统设计思路 2](#_Toc5055620)

[1.4 技术选型思路 3](#_Toc5055621)

[2 关键技术介绍 4](#_Toc5055622)

[2.1 Scrapy 4](#_Toc5055623)

[2.2 Redis 4](#_Toc5055624)

[2.3 Vue.js 4](#_Toc5055625)

[2.3.1 Vue Router 4](#_Toc5055626)

[2.3.2 Vuex 4](#_Toc5055627)

[2.4 Spring Framework 5](#_Toc5055628)

[2.4.1 Spring Boot 5](#_Toc5055629)

[2.4.2 Spring Cloud 5](#_Toc5055630)

[2.4.3 Spring Data 5](#_Toc5055631)

[2.5 RabbitMQ 6](#_Toc5055632)

[2.6 Elasticsearch 6](#_Toc5055633)

[2.7 Hibernate 6](#_Toc5055634)

[2.8 MySQL 6](#_Toc5055635)

[2.9 Nginx 6](#_Toc5055636)

[2.10 Docker 6](#_Toc5055637)

[3 系统架构与关键部分实现细节 7](#_Toc5055638)

[3.1 系统总体架构 7](#_Toc5055639)

[3.2 数据库结构设计 8](#_Toc5055640)

[3.2.1 Elasticsearch索引设计 8](#_Toc5055641)

[3.2.2 MySQL表设计 10](#_Toc5055642)

[3.3 服务启动与注册 15](#_Toc5055643)

[3.4 爬虫实现细节 16](#_Toc5055644)

[3.4.1 网站URL跟踪方式 16](#_Toc5055645)

[3.4.2 页面内容的解析 17](#_Toc5055646)

[3.4.3 URL去重与任务队列 18](#_Toc5055647)

[3.5 资源服务器实现细节 18](#_Toc5055648)

[3.5.1 搜索功能实现 18](#_Toc5055649)

[3.6 评判器实现细节 18](#_Toc5055650)

[3.6.1 提交源码的编译 18](#_Toc5055651)

[3.6.2 测试用例的解析 18](#_Toc5055652)

[3.6.3 被测代码运行信息 19](#_Toc5055653)

[3.6.4 运行结果评判 19](#_Toc5055654)

[3.6.5 评判器健壮性 19](#_Toc5055655)

[4 系统测试 20](#_Toc5055656)

[4.1 功能测试 20](#_Toc5055657)

[4.1.1 资源检索功能测试 20](#_Toc5055658)

[4.1.2 代码评判功能测试 20](#_Toc5055659)

[4.2 水平扩展与性能测试 20](#_Toc5055660)

[4.2.1 测试环境 20](#_Toc5055661)

[4.2.2 资源服务器性能测试 20](#_Toc5055662)

[4.2.3 评判器性能测试 20](#_Toc5055663)

[4.3 评判器健壮性测试 20](#_Toc5055664)

[5 总结与展望 21](#_Toc5055665)

# 前言

* 1. 研究背景

根据中华人民共和国工业和信息化部发布的《2017年1-12月全国软件和信息技术服务业主要指标快报表》，我国软件和信息技术服务业从业人员平均人数同比增速3.4%；《2018年全国软件和信息技术服务业主要指标快报表》显示，我国软件和信息技术服务业从业人员平均人数同比增速达到4.2%。我国的软件业一直存在较大人才缺口，吸引了部分高校学生选择或转入软件、计算机相关专业，以及部分社会人士转行从事计算机软件相关行业。

计算机行业的专业性很强，尤其是开发类的工作，涉及大量专业知识，对从业者的基础如数据结构、常见算法、编程语言基础等有一定要求。基础知识的学习方式较多样化，校内有教师课堂教学，校外有培训机构开设课程，互联网上也有大量资料。由于课堂开班教学局限性较大，教师难以在有限时间内传授大量专业知识，通过互联网自学是计算机专业最重要的学习手段。

目前互联网上有较多计算机理论知识学习网站、数据结构与算法的可视化学习网站、计算机相关专业人员的博客，也有例如UVa等用于编程练习、比赛的网站，这些网站目前能够满足大部分用户的需求，但仍存在一些问题。

从使用者的角度，这些计算机专业学习网站的功能相对单一，理论知识学习网站只能基本满足理论学习的需求，实践还需要用户在自己的电脑或者其他在线评判系统进行，各个不同网站之间联系不大，导致所学的理论知识难以及时关联实践所需知识。

从系统架构的角度，目前大多数网站为单体架构，即所有应用模块集中在同一个项目中，开发流程较简单，但不易修改、扩展，系统复杂难以维护。以Java为例，在早期开发项目主要基于MVC分层模型开发Web应用，系统中不同模块都集中在同一项目的Model和Controller层，View层后端渲染Web页面模板文件。在扩展方面，单体系统只能水平扩展，无法只对系统中某一模块进行扩展，缺乏灵活性。

随着互联网的发展以及用户的需求不断增加，各个不同的功能将会集成到一个大的平台中，例如数据结构、算法理论知识与编程实践题目对接，紧密关联，节省用户用于检索资料、寻找实践方法的时间。除了功能方面，由于互联网开放性，系统使用人数不确定，系统在可用性和并发量方面都有一定的要求。因此，模块高度耦合、难以扩展的传统MVC单体架构已经难以满足用户需求，越来越多的新项目基于微服务架构开发，也有少数较早的系统被重构为微服务架构。微服务架构解决了复杂性问题，它将单体架构分解为一组服务，把系统各个不同的模块解耦。服务之间定义了一系列明确的接口，只要遵循接口等规则，每个服务的具体实现对于其他服务完全透明。对于压力较大的服务，可以有针对性地进行水平扩展，而无需像单体架构只能对整个系统水平扩展。微服务架构的优势非常明显，但也增加了开发过程的复杂度，开发基于微服务架构的系统需要解决服务发现、服务间调用、服务容错、服务部署等问题。

* 1. 研究意义

本系统面向的用户群体是需要学习计算机专业知识的人群，能够提供一个计算机专业理论学习与编程实践平台给计算机专业的学生用于课后学习，或需要自学计算机专业知识的用户用于自习。本系统结合理论与实践，能够提供一种更便利地学习计算机专业知识的方式，提高使用者的学习效率。

其次，互联网上现有的开源在线评判系统数量较少，国内常见的有基于Python的青岛大学qduoj、基于PHP的hustoj与uoj，基于Java以Spring体系框架开发的OJ数量更为稀少。开发基于Java的计算机专业资源学习网站，能够为目前为数不多的开源的在线评判系统做出一些贡献。

* 1. 系统设计思路

本系统是一个在互联网上开放的系统，面向的用户群体是计算机专业相关人员。与一些组织或个人内部使用的系统相比，开放网络的用户人数难以预计，因此系统的可用性与并发量是系统架构设计所需考虑的因素。

互联网上存在海量信息，可以作为本系统资料的主要来源，结合系统内用户人工整理、发布的资料，能够实现本站的资源检索功能。

通用搜索引擎仅仅具备检索功能，且搜索结果不具有针对性。本系统开发的检索功能必须与通用搜索引擎相比更具有针对性，否则将失去使用的意义。

在编程实践方面，常见的在线评判系统的评判器在评判代码时能够精确统计时间、内存等信息，评判器也具有一定的健壮性，不会轻易被用户提交的恶意代码攻击系统。但这些评判器对操作系统的依赖性较强、评判代码时涉及进程创建、切换等操作，运行时开销较大。本系统评判器作为架构中的一个服务，如果能在保证健壮性与运行性能的前提下减少评判器对操作系统的依赖，在扩展、集群部署方面都会有一定的优势。

综上，本系统设计思路可以概括为：

1. 采用微服务架构，支持分布式以保证可用性和并发性能；
2. 学习资源来源以互联网各大计算机相关博客为主，站内用户整理发布为辅；
3. 对文章资源等数据进行索引时，需要索引更多详细的信息；
4. 理论学习资源与编程题目需要通过一定的关系进行关联；
5. 系统架构的各个服务实现时需要降低对运行环境的依赖性，降低部署难度，保证系统运行与维护具有一定的灵活性。
   1. 技术选型思路

爬虫模块：网络爬虫在爬取大型网站时需要部署较多实例，因此要求每个实例拥有较低的部署难度。基于Python语言的Scrapy是一款易用、成熟的爬虫框架，由于框架自身封装了大量与业务逻辑无关的代码，上手简单，开箱即用，只需编写少量逻辑即可启动。Scrapy的轻量级、开箱即用的特点恰好能够满足低部署难度。

系统前端：作为与用户直接交互的部分，前端的体验非常重要。近年来，前端出现了一种使用MVVM设计模式的应用类型——SPA（Single Page Web Application，单页面应用程序）。与传统的多页应用程序相比，由于SPA首次启动便完成了应用所需的静态资源的加载，切换页面没有了请求应用自身静态资源的开销，具有非常快的响应速度，流畅度也高于传统的多页应用程序。本系统开发SPA应用使用JavaScript框架Vue.js及其生态圈下的Vue Router、Vuex插件，该框架上手容易，性能也相对良好。

系统后端：系统后端采用微服务架构，需要解决的问题包括服务发现、应用配置等。结合各方面考虑，系统后端开发选用Java作为主要编程语言。Java拥有庞大的生态圈，是很多企业级应用开发的合适选择。Spring Framework是以Java语言为主的一个庞大的框架体系，所提供的工具几乎涉及了应用开发的所有层次，包括了本系统所需的微服务架构、数据持久层、消息中间件、RESTful API开发，因此本系统开发技术首选以Spring Boot、Spring Cloud、Spring Data的一系列框架。

数据存储：本系统的资源服务器需要实现包括全文搜索、数据聚合等功能，且由于数据量会随爬虫的运行而增加，数据库性能、存储空间都具有一定的压力，数据库需要具有分布式部署能力。Elasticsearch是个性能强大、技术成熟的搜索引擎，天生支持分布式部署，使系统的灵活性和可用性都有一定的保证，故选用Elasticsearch存储、索引数据。

# 关键技术介绍

* 1. Scrapy

Scrapy是基于Python的开源爬虫框架，能够爬取网页并解析结构化的数据。Scrapy框架封装了较多繁琐的底层操作，使用简单，对于比较简单的网站，只需编写少量逻辑与修改一些配置项即可开始抓取数据。除了封装基本的流程，Scrapy框架可扩展性非常高，拥有大量强大的插件，根据各种应用场景使用对应的插件，能够极大提高使用者的开发效率。

* 1. Redis

Redis是一个基于内存的NoSQL数据库，支持5种常用数据结构：String、List、Hash、Set、Sorted Set。由于数据存储在内存中，Redis拥有极高的吞吐量，应用场景非常广泛，例如缓存、计数器、分布式锁、队列、适用于大数据的位操作、排行榜等。为了应对内存的易失性，Redis提供了两种持久化的方式：一是保存当前内存快照，这种方式占用磁盘空间较小，但存在丢失少量数据的风险；二是通过追加命令的方式记录对Redis的所有操作，这种方式不易丢失数据，但日志文件会随着时间的推移一直扩大，对磁盘空间占用较大。Redis支持主从模式、集群，在可用性和并发性能方面有所保障。

* 1. Vue.js

Vue.js是一款目前比较流行的JavaScript框架，结合ECMAScript6模块化后，能够以编写组件的方式前端页面。与直接使用HTML、CSS、JavaScript编写HTML等文件的传统方式相比，用组件构建前端页面的方式能够在一定程度减少代码冗余，有效提高代码复用率。Vue.js框架还有许多插件，扩展了框架能够实现的功能，能够满足绝大多数项目的需求。目前还有许多基于Vue.js框架开发的UI库，例如Element，提供了很多常用UI，能够快速搭建逻辑清晰、结构合理且高效易用的前端界面。

* + 1. Vue Router

Vue Router是Vue.js官方提供的路由管理器，该插件能够让构建SPA易如反掌。该插件包含的功能较多，包括常用的嵌套路由或视图、基于组件的模块化的路由配置、支持通配符的路由参数查询、视图过渡动画效果等。

* + 1. Vuex

Vuex是专门为Vue.js应用开发的状态管理模式，可以简单地解为全局变量，但Vuex的功能远远强大于一个简单的全局变量。Vuex采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化。可以用Vuex保存的信息有很多，例如用户登录后，服务端返回基于JWT的令牌可以存储在Vuex的数据部分，以便于后续操作调用服务端API时携带令牌。

* 1. Spring Framework

Spring Framework是一个很庞大的体系，它实现了很多经典设计模式，封装大量繁琐与业务无关的底层代码，提供很多高级API，极大提高项目开发效率。Spring系列框架覆盖了绝大多数项目的各个层级，例如Web、数据库、中间件、批处理等，也提供了搭建分布式、微服务架构所需的工具。而且，Spring能够做到开箱即用，引入所需依赖，只需少量的配置，即可开始专注于实现业务逻辑代码。

* + 1. Spring Boot

Spring Boot能够实现Spring Framework开箱即用。在Spring Boot出现之前，引入Spring框架的项目搭建与配置是一项非常繁琐的事情。举个例子，搭建一个简单的基于Spring MVC的Web项目时，需要先手动或用Maven之类的项目依赖管理工具引入Spring框架的依赖，然后需要配置XML文件或者使用注解配置引入Spring MVC的DispacherServlet或其他相关Servlet与Filter，最后发布到Tomcat之类的Servlet容器才能使基本项目运行。如果要连接数据库，还需要配置数据源、事务管理器等，过程非常繁琐。但Spring Boot的出现使得这一切发生变化。

Spring Boot的理念是“约定优于配置”，底层隐藏了大量的自动配置逻辑，最后只需少量配置甚至实现零配置。除此，Spring可以内置Tomcat之类的Servlet容器，使得运行项目时无需再单独配置Servlet容器，实现开箱即用。

* + 1. Spring Cloud

分布式系统由于自身的特点，增加了开发过程的复杂性。Spring Cloud为开发人员提供了快速构建分布式系统中常见模式的实现，例如服务发现、配置管理、断路器、路由、分布式会话、集群监控等，是快速搭建微服务架构系统的利器。

* + 1. Spring Data

Spring Data封装了操作各类数据源的底层代码，例如Spring Data JPA能够实现声明式编程，只需定义接口，无须实现代码甚至无须SQL语句即可实现数据库的增删查改。需要实现复杂查询的时候，也只需少量的代码，能够保持低耦合。

* 1. RabbitMQ

RabbitMQ是基于Erlang语言编写的消息中间件。得益于Erlang语言本身并发性能较高的特性，RabbitMQ的性能较好；支持AMQP等多种协议，且大多数编程语言都提供了AMQP协议客户端，语言无关使其兼容性良好。RabbitMQ支持消息持久化以应对各种紧急情况，保证消息不会丢失，且提供了控制台，能够实时监控消息队列的各种状态或对队列进行管理配置。

* 1. Elasticsearch

Elasticsearch是一个基于Lucene的分布式、RESTful风格的搜索和数据分析引擎。Elasticsearch具有很高的灵活性，提供了非常多的数据类型，能够对数字、文本、地理位置、结构化数据、非结构化数据进行存储、分析。

在性能方面，Elasticsearch通过有限状态转换器实现了用于全文检索的倒排索引，实现了用于存储数值数据和地理位置数据的BKD树，以及用于分析的列存储。而且由于每个数据都被编入了索引，不会因为某些数据没有被索引而遗漏。

Elasticsearch的可扩展性也非常强，既能在性能不高的机器运行单个实例，也可以在数百个节点的集群运行，能够自动管理索引和查询在集群中的分布方式，使得在各种不同环境下能够以相同的操作方式实现查询与分析，实现开发环境和生产环境无缝切换。

* 1. Hibernate

ORM

* 1. MySQL

数据库

* 1. Nginx

高性能

* 1. Docker

容器

# 系统架构与关键部分实现细节

* 1. 系统总体架构

系统架构见图1。



图1 系统架构

本系统基于微服务架构，支持集群、分布式部署。

前端使用Nginx作为HTTP服务器与反向代理服务器，负责用户获取静态页面以及AJAX请求的反向代理。

后端微服务主要分为两个部分：用于维护微服务架构的服务注册中心与配置服务、用于面向用户提供功能的资源服务器与评判器集群。其中，资源服务器与评判服务器依赖消息中间件RabbitMQ，资源服务器需要连接MySQL、Redis、Elasticsearch。

用于采集数据的爬虫为独立模块，可以任意部署在可以连接Elasticsearch服务器的机器，分布部署爬虫需要依赖同一个Redis实现任务队列与去重。

* 1. 数据库结构设计
     1. Elasticsearch索引设计

本系统学习资源存储与索引基于Elasticsearch实现。本索引用于实现资源检索功能，详细功能包括：精确搜索、模糊搜索、基于前缀的搜索建议、拼音搜索、搜索关键词纠错等。学习资源索引结构见图2。



图2 学习资源索引结构

在Elasticsearch中，字符串对应的数据类型拥有两种：keyword、text。在Elasticsearch对文档进行索引时，使用keyword数据类型存储的字符串会不会被分词器分析，整个字符串直接存储到Elasticsearch中；而使用text数据类型的字符串，在索引时会被默认或指定的分词器分析，并形成倒排索引。因此，使用text类型存储的字符串可以被用于搜索功能，但不可被作为term进行数据聚合；使用keyword类型存储的字符串查找时只能够使用term或者prefix，但在进行聚合时可以作为被聚合的字段。

* + 1. MySQL表设计

用户表主要存储了用户名和加密后的用户密码。每个用户拥有唯一的用户名，登录时使用用户名、密码登录。为了提高系统的安全性，登录密码存储经过不可逆加密后的值。用户表结构如表1。

表1 用户表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 列名 | 列含义 | | | 数据类型 | | 长度 | 备注/默认值 | | |
| id | | 用户编号 | bigint | | 20 | | | 主键 |
| username | | 用户名 | varchar | | 255 | | | 值唯一 |
| password | | 登录密码 | varchar | | 255 | | | BCrypt算法加密 |
| sign\_up\_date | | 注册时间 | timestamp | |  | | | CURRENT\_TIMESTAMP |

由于本系统使用RBAC（基于角色的权限访问控制），每个用户拥有一定数量的角色，不同的角色能够使用不同的功能，例如发布新的编程题目需要具有题目管理员权限的用户执行，删除其他用户上传的资源需要用户具有资源管理员权限。用户角色表见表2。

表2 用户角色表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 列名 | 列含义 | | | 数据类型 | 长度 | | | 备注/默认值 | |
| id | | 编号 | bigint | | | 20 | 主键 | |
| user\_id | | 用户编号 | bigint | | | 20 | 外键 | |
| role\_name | | 角色名称 | varchar | | | 255 |  | |
| grant\_time | | 角色授予时间 | timestamp | | |  | CURRENT\_TIMESTAMP | |

本系统会在为每位用户记录历史浏览记录，以便于用户回顾过去浏览的文章。由于本站资源来源分为站外和站内，站外资源可以直接通过url访问，站内资源需要经过资源服务器获取资源，于是使用external字区分站内与站外资源，以便于在前端采用不同的处理方式。

表3 用户浏览历史

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 列名 | 列含义 | | | 数据类型 | 长度 | | 备注/默认值 |
| id | | 历史记录编号 | bigint | | | 20 | 主键 | |
| user\_id | | 用户编号 | bigint | | | 20 | 外键 | |
| title | | 资源标题 | varchar | | | 255 |  | |
| url | | 资源链接 | text | | |  |  | |
| external | | 是否外部资源 | tinyint | | | 1 |  | |
| view\_time | | 浏览时间 | timestamp | | |  | CURRENT\_TIMESTAMP | |

编程题目表用于存储本系统中拥有的编程题目以及该题目的发布者。编程题目表结构如表4。

表4 编程题目表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 列名 | 列含义 | | | 数据类型 | 长度 | | 备注 |
| id | | 题目编号 | bigint | | | 20 | 主键 | |
| title | | 题目标题 | varchar | | | 255 |  | |
| description | | 题目描述 | text | | |  |  | |
| contributor\_id | | 贡献者编号 | bigint | | | 20 | 外键 | |

由于不同编程语言的语法不同，每个编程题目需要存储对应各编程语言的代码模板，如果代码中包含多个方法/函数，则需要显式指定入口方法/函数。代码模板表结构见表5。

表5 代码模板表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 列名 | 列含义 | | | 数据类型 | 长度 | | 备注 |
| id | | 代码模板编号 | bigint | | | 20 | 主键 | |
| problem\_id | | 题目编号 | bigint | | | 20 | 外键 | |
| template | | 代码模板 | text | | |  |  | |
| entry\_method | | 默认入口方法 | varchar | | | 255 |  | |
| language | | 编程语言 | varchar | | | 255 |  | |

测试用例表用来存储验证提交代码所用的测试用例，每一行记录都对应一个问题的一个测试用例。输入值与期望值以JSON格式的文本存储，评判器在获取到JSON格式的字符串后，会把字符串解析成输入值、输出值对应类型的实例，以便于输入被测函数/方法。测试用例表结构见表6。

表6 测试用例表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 列名 | 列含义 | | | 数据类型 | 长度 | | 备注 |
| id | |  | bigint | | | 20 | 主键 | |
| problem\_id | | 题目编号 | bigint | | | 20 | 外键 | |
| input | | 输入值 | text | | |  |  | |
| expect | | 期望值 | text | | |  |  | |
| timeout | | 时间限制 | bigint | | | 20 |  | |

标签表用于存储编程题目相应的标签，通过该标签，编程题目可以和拥有相同标签的学习资源关联。标签表需要通过关联表与题目关联。标签表结构见表7。

表7 标签表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 列名 | 列含义 | | | 数据类型 | 长度 | | 备注 |
| id | |  | bigint | | | 20 | 主键 | |
| tag\_name | | 标签名 | varchar | | | 255 |  | |

题目与标签的关联，每一行记录包含了题目编号与标签编号，题目标签关联表结构如表8所示。

表8 题目标签关联表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 列名 | 列含义 | | | 数据类型 | 长度 | | 备注 |
| id | |  | bigint | | | 20 | 主键 | |
| problem\_id | | 题目编号 | bigint | | | 20 | 外键 | |
| tag\_id | | 标签编号 | bigint | | | 20 | 外键 | |

用户资源关联表用于记录系统内用户与其发布的资源的关联关系，其中，resource\_id字段与Elasticsearch中每个文档自动生成的\_id对应。用户资源关联表见表9。

表9 用户资源关联表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 列含义 | 数据类型 | 长度 | | 备注 |
| id |  | bigint | | 20 | 主键 | |
| user\_id | 用户编号 | bigint | | 20 | 外键 | |
| resource\_id | 资源编号 | varchar | | 32 | 与Elasticsearch资源索引对应 | |

编程提交表用于记录用户的编程题解提交记录，包括了提交的题目与源码等数据。diagnosis字段用于记录编译出错时的提示消息或者运行抛出异常时的错误堆栈，以便于用户定位出错原因与相关代码行数。本表一般在用户提交代码时创建一条记录，等评判器评判结果返回时通过id对相应的记录进行更新。编程提交表结构如表10所示。

表10 编程提交表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 列名 | 列含义 | | 数据类型 | | 长度 | 备注/默认值 |
| id | | 提交编号 | | bigint | 20 | 主键 |
| problem\_id | | 题目编号 | | bigint | 20 | 外键 |
| code | | 提交代码 | | text |  |  |
| entry\_method | | 入口方法 | | varchar | 255 |  |
| language | | 编程语言 | | varchar | 255 |  |
| compile\_success | | 是否编译成功 | | tinyint | 1 |  |
| message | | 错误消息 | | text |  |  |
| diagnosis | | 错误诊断 | | text |  |  |
| time | | 提交时间 | | timestamp |  | CURRENT\_TIMESTAMP |
| judged | | 已评判 | | tinyint | 1 |  |
| judge\_time | | 评判时间 | | timestamp |  |  |
| user\_id | | 提交用户编号 | | bigint | 20 | 外键 |
| accepted | | 是否正确 | | tinyint | 1 |  |
| used\_testcase\_count | | 运行次数 | | int | 11 |  |
| accepted\_testcase\_count | | 正确次数 | | int | 11 |  |
| error\_case | | 错误输出 | | text |  |  |

* 1. 服务启动与注册流程

服务注册中心启动后，服务的注册与启动流程如图2所示。



图2 服务启动与注册流程

一般传统单体架构系统中各配置文件与代码集成，采用这种方式的系统在启动时流程比较简单，直接运行就可以连接数据库等其他服务器。由于微服务架构的系统中有较多的服务节点需要部署，如果与数据库等相关的配置文件与代码集成，一旦需要更新配置内容，节点越多需要做的操作就越多，需要耗费较多时间、资源。因此，单体架构常用的配置方式不太适用于微服务架构系统。

本系统的配置方式是基于配置服务器。当服务注册中心启动完成后，接着启动配置服务，配置服务向服务注册中心注册并使用Git或SVN等工具或从本地拉取所有配置文件后，此时就可以启动系统中的其他服务了。

以资源服务器为例，资源服务启动时，会先向服务注册中心注册，并且从服务注册中心获取到配置服务的相关信息，例如Socket等信息。得到Socket后，资源服务就向配置服务拉取相关配置。配置文件按照环境可以分为Development（开发环境）、Test（测试环境）、Production（生产环境）等，以便于在部署服务时无缝切换到不同环境。获取到配置文件后，资源服务开始与数据库等服务建立连接，完成服务启动流程。

* 1. 爬虫实现细节
     1. 任务队列与去重

本系统采用Scrapy框架，在默认情况下，任务队列与去重都是在爬虫进程内存中实现的，支持持久化到本地。如果只有一个爬虫实例，默认配置能够满足需求。但在需要多个爬虫实例的情况下，默认配置的爬虫之间无法协作，因此需要将任务队列与去重数据转移到能够共用的区域。



图3 URL去重流程

URL去重流程如图3所示。

URL参数顺序是无关的。例如“a=1&b=2”等价于“b=2&a=1”，因此，去重逻辑第一步是对URL参数按照参数的键重新排序，避免因URL参数顺序问题被识别为不同URL。

RFC3986文档规定，URL中只允许包含英文字母、数字、"-\_.~"4个特殊字符以及所有保留字符。因此，非ASCII编码的字符需要被编码成URL允许的字符。

URL中百分号编码大小写不敏感。例如百分号编码“%2f”等价于“%2F”，因此，需要对URL中的百分号编码进行大小写规整化，确保能够正确去重。

URL长度不固定，如果直接在去重集存储URL及相关的HTTP方法、请求体将会占用较大空间，故先拼接URL相关信息并做SHA1，将每个URL转换为占用20字节空间的哈希值，能够在有效去重的前提下节省大量空间。

Redis的SADD命令是将值放入指定的集合中，并返回成功放入的元素个数。如果被放入的值在集合中已存在，元素放入失败，返回成功放入的元素个数为0。

最后，根据成功放入去重集的元素个数，决定是否将目标URL放入爬虫任务队列中。

* + 1. 网站URL跟踪方式

网络爬虫跟踪URL的方式有许多种，常见的有如下：

1. 分析URL规则并循环，适用于URL中有连续编号的ID或其他数据；
2. 爬取当前页面后解析页面中所有链接，将符合规则的链接放入待爬取队列，这种方式；
3. 解析网站sitemap，适用于提供了sitemap的网站。

本系统的爬虫模块采用了以上2、3两种方式。

* + 1. 配置正则表达式规则跟踪URL

爬取当前网页并解析当前页面中所有URL进行跟踪是一种比较通用的URL跟踪方式，这种方式需要选择合适的爬虫起点，例如首页或者一些专题页面。

本系统的爬虫，主要采用正则表达式配置URL跟踪规则，根据不同的匹配结果，决定该URL的是否需要被继续跟踪，是否需要解析页面内容。

* + 1. 读取sitemap跟踪URL

通过网站提供的sitemap爬取网站是一种比较高效且较容易覆盖全站的方式。sitemap是一种SEO（Search Engine Optimization，搜索引擎优化）手段，sitemap文件中记录了网站大量URL，包括主页或其他重要页面，其目的是让搜索引擎能够尽量索引更多的网站内容。本系统的检索功能就是一个搜索引擎，爬虫通过sitemap爬取目标网站，能够使本系统的搜索功能覆盖更多学习资源。

以CSDN为例。CSDN博客在其robots.txt文件中（https://blog.csdn.net/robots.txt，该文件存储了对爬虫的规定，例如允许/禁止爬取的URL、允许/禁止的User-Agent等）提供了sitemap的URL（https://blog.csdn.net/s/sitemap/pcsitemapindex.xml）。

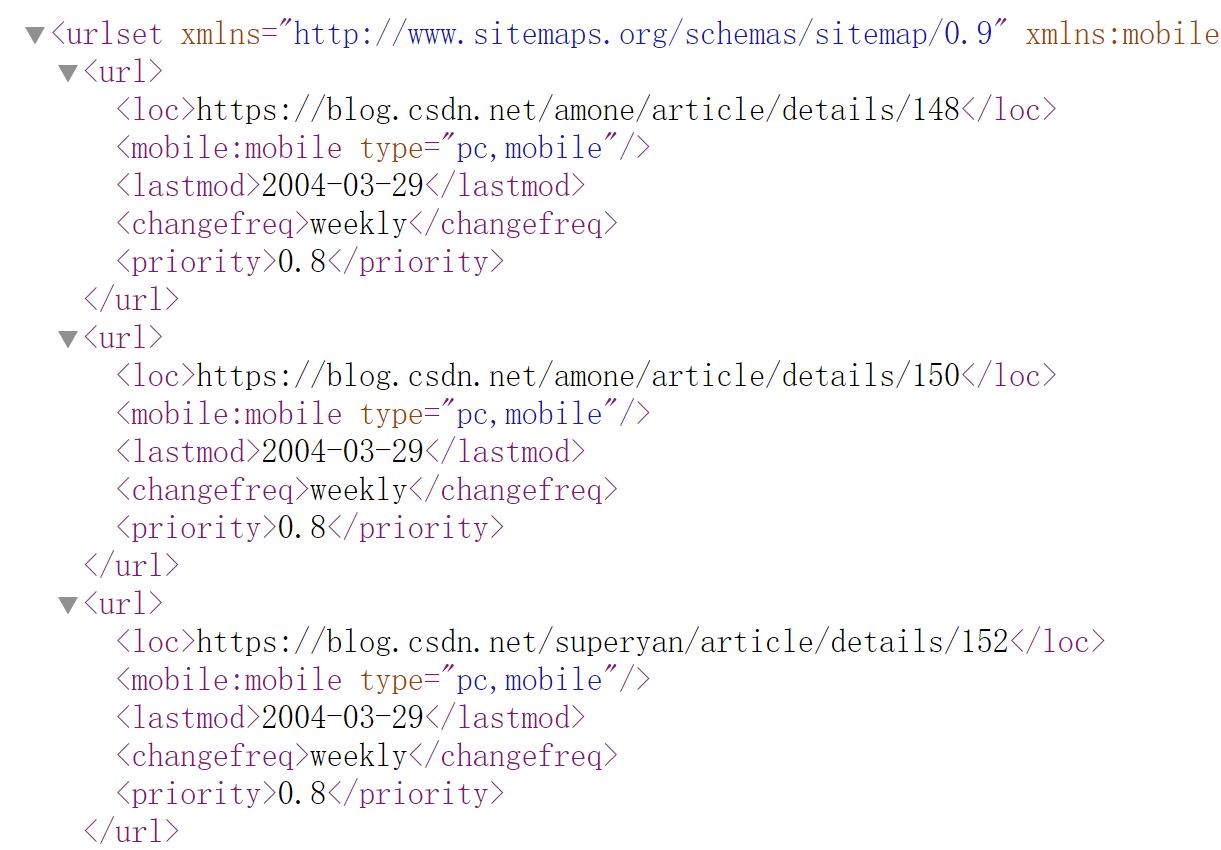


图4 sitemap文件节选

sitemap文件节选如图4所示。sitemap文件包含了URL、最后更新时间、修改频率、优先级等信息，爬虫根据sitemap爬取网站时可以更加高效、全面。

* + 1. 页面内容的解析

爬虫发送HTTP请求并获取响应HTML后，需要从HTML页面中提取出所需信息。本系统的爬虫通过XPath的方式解析HTML。

XPath即为XML路径语言（XML Path Language），它是一种用来确定XML文档中某部分位置的语言。XPath基于XML的树状结构，提供在数据结构树中找寻节点的能力。

对于常见的博客，大多数博文都有相似的HTML页面结构，因此，对于一个网站，一般只需配置一次XPath即可用于解析该网站大部分页面。

以CSDN博客的博文为例：

文章标题的XPath："//\*[@id="mainBox"]/main/div[1]/div/div/div[1]/h1/text()"；

正文内容的XPath："//\*[@id="article\_content"]//text()"。

使用XPath提取出页面各项所需内容后，即可通过Elasticsearch对资源进行索引。

* 1. 资源服务器实现细节
     1. 搜索功能实现
  2. 评判器实现细节
     1. 提交源码的编译
     2. 测试用例的解析
     3. 被测代码运行信息
     4. 运行结果评判
     5. 评判器健壮性

# 系统测试

* 1. 功能测试
     1. 资源检索功能测试
     2. 代码评判功能测试
  2. 水平扩展与性能测试
     1. 测试环境
     2. 资源服务器性能测试
     3. 评判器性能测试
  3. 评判器健壮性测试

# 总结与展望

参 考 文 献

致 谢

**华南农业大学**

**本科生毕业论文成绩评定表**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 学号 |  | | | | | 姓名 | |  | 专业 | 软件工程 | | |
| 毕业论文题目 | | | 基于数据校验器架构模式组的旅游基础支撑系统 | | | | | | | | | |
| 指导教师评语  成绩（百分制）：　　　　　　　　　　　　　　　　　　　　指导教师签名：　　　　　　　　　　　　年　　月　　日 | | | | | | | | | | | | |
| 评  阅  人  评  语  及  成  绩  评  定 | | 成绩  评定  标准 | | **评分项目** | | | | | | | **分值** | **得分** |
| 选题  质量  20% | 1 | | 专业培养目标 | | | | 5 |  |
| 2 | | 课题难易度与工作量 | | | | 10 |  |
| 3 | | 理论意义或生产实践意义 | | | | 5 |  |
| 能力  水平40% | 4 | | 查阅文献资料与综合运用知识能力 | | | | 10 |  |
| 5 | | 研究方案的设计能力 | | | | 10 |  |
| 6 | | 研究方法和手段的运用能力 | | | | 10 |  |
| 7 | | 外文应用能力 | | | | 10 |  |
| 成果  质量40% | 8 | | 写作水平与写作规范 | | | | 20 |  |
| 9 | | 研究结果的理论或实际应用价值 | | | | 20 |  |
| 评阅人评语  成绩（百分制）：　　　　　　　　　　　　　评阅人签名：　　　　　　　　　　　　年　　月　　日 | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 答  辩小组  评语及  成  绩  评  定 | 评价项目 | 具体要求（A级标准） | 最高分 | 评分 | | | | |
| A | B | C | D | E |
| 论文质量 | 论文结构严谨，逻辑性强；有一定的学术价值或实用价值；文字表达准确流畅；论文格式规范；图表（或图纸）规范、符合要求。 | 60 | 55-60 | 49-54 | 43-48 | 37-42 | ≤36 |
|  |  |  |  |  |
| 论文报告、讲解 | 思路清晰；概念清楚，重点（创新点）突出；语言表达准确；报告时间、节奏掌握好。 | 20 | 19-20 | 17-18 | 15-16 | 13-14 | ≤12 |
|  |  |  |  |  |
| 答辩情况 | 答辩态度认真，能准确回答问题 | 20 | 19-20 | 17-18 | 15-16 | 13-14 | ≤12 |
|  |  |  |  |  |
| 答辩小组评语  是否同意通过论文答辩（打√）   1. 同意 2. 不同意   成绩（百分制）：　　　　　　　　 答辩小组成员（签名）：  　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　 年 月 日 | | | | | | | |
| 成绩总评 | 论文总评分数：  教学院长签名：  学院盖章：  年 月 日 | | | | | | | |

续上表：