申请上海交通大学工程硕士学位论文

基于CBIR的内容管理平台的设计与实现

学校代码： 10248

作者姓名： 马丁

学 号： 1130379193

第一导师： 孙焱

第二导师： 胡海亮

学科专业： 软件工程

答辩日期： 年 月 日

上海交通大学软件学院

2016年4月

Dissertation Submitted to Shanghai Jiao Tong University for the Degree of Master

THE DESIGN AND IMPLEMENTATION OF CONTENT MANAGEMENT SYSTEM BASED ON CBIR

|  |  |
| --- | --- |
| Candidate： | 10248 |
| Student ID: | 1130379193 |
| Supervisor： | Sun Yan |
| Assistant Supervisor: | Hu Hailiang |
| Academic Degree Applied for： | Master of Engineering |
| Speciality： | Circuits and System |
| Affiliation： | School of Microelectronics |
| Date of Defence： |  |
| Degree-Conferring-Institution： | Shanghai Jiao Tong University |

**上海交通大学**

**学位论文原创性声明**

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期： 年 月 日

**上海交通大学**

**学位论文版权使用授权书**

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

**保密**□，在 年解密后适用本授权书。

本学位论文属于

**不保密**□。

（请在以上方框内打“**√**”）

学位论文作者签名： 指导教师签名：

日期： 年 月 日 日期： 年 月 日

基于CBIR的内容管理平台的设计与实现

摘 要

对企业而言，伴随自身的不断发展，总会产生各种各样的内容资源，如文档，web网页，图片素材等等。提高这些内容资源的重复利用率以及检索效率将会对员工的工作效率产生巨大的积极影响。

为解决上述需求，内容管理系统应运而生。传统的内容管理系统已经能够解决企业对于内容管理的大部分需求。但其检索往往是基于文本的，在图片内容地位更加重要的今天则具有一定的局限性。

据此，本文以百度公司内部项目RMP，一款基于CBIR(基于内容的图像检索)的内容管理平台为背景，对如下内容做出了研究：

1. 基于业务需求研究、设计并实现一个目标用户为千人左右的内容管理系统。系统功能涵盖了内容生命周期的大部，从内容的产生到内容的发布。针对内容管理相关的上传、下载、版本控制等功能研究其解决方案并给出最终的实现。

2. 研究基于内容的图片检索技术(CBIR)。针对现有的图像检索算法，如基于颜色特征的图像检索算法，基于轮廓特征的图像检索算法，SIFT算法等进行研究。最终选择了较为适合本系统的感知哈希技术，并对其实现以及相关技术，如离散傅立叶变换进行了研究与分析。

3. 将CBIR技术整合仅本系统之中，在图片内容上传时，以及图片检索时应用CBIR技术，为用户的图像检索提供一种额外的选择。对系统图片检索的准确率进行测试。

目前系统已经投入正式使用，应用于百度公司内部的数个开发中，并在进一步推广。

关键词：内容管理 内容管理系统 基于内容的图片检索

THE DESIGN AND IMPLEMENTATION OF CONTENT MANAGEMENT SYSTEM BASED ON CBIR

**ABSTRACT**

For enterprises, with its continuous development, it will produce a variety of content resources, such as documents, web pages, picture material and so on. To improve the reuse efficiency and retrieval efficiency of these resources will have a great positive impact on the efficiency of the staff.

In order to solve the above requirements, the content management system came into being. Traditional content management systems have been able to address most of the needs of enterprise content management. But its retrieval is often based on the text, with the increasing importance of picture content today, it has some limitations.

According to the above, this paper takes the Baidu Inc internal project RMP, a content management platform based on CBIR (content based image retrieval) as the background, and makes a research on the following content:

Firstly, research, design and implement a content management system based on business requirements. System function covers the life cycle of contents, from the creation of content to the release of content. To study the solution of content management related upload, download, version control and so on.

Secondly, research on content based image retrieval (CBIR). Research on the existing image retrieval algorithms, such as the color feature based image retrieval algorithm, contour feature based image retrieval algorithm, SIFT algorithm and so on. Finally, we choose the perceptual hash technology which is suitable for this system, and the implementation and related technologies, such as Discrete Fourier Transformation, are studied and analyzed.

Lastly, CBIR technology is integrated into the system, in the image content upload, and image retrieval when the application of CBIR technology, to provide users with an additional choice of image retrieval. To test the accuracy of image retrieval system.

At present, the system has been put into formal use, used in the development of a number of Baidu Inc, and in the further promotion.

**KEY WORDS:** Content Management, Content Management System, Content Based Image Retrieval

目 录

[摘 要 I](#_Toc460214911)

[**ABSTRACT** III](#_Toc460214912)

[**1 绪论** 1](#_Toc460214913)

[**1.1** 论文的研究背景及意义 1](#_Toc460214915)

[**1.2** 国内外研究现状 2](#_Toc460214916)

[**1.2.1** 内容管理技术的国内外研究现状 2](#_Toc460214917)

[**1.2.2** 图像检索技术的国内外研究现状 2](#_Toc460214918)

[**1.3** 论文的研究目的和组织结构 3](#_Toc460214919)

[**1.3.1** 论文的研究目的 3](#_Toc460214920)

[**1.3.2** 论文的组织结构 4](#_Toc460214921)

[第二章 相关技术及算法简介 5](#_Toc460214922)

[**2.1** 内容管理关键技术简介 5](#_Toc460214925)

[**2.2** 图像检索技术主流算法简介 6](#_Toc460214926)

[**2.3** 感知哈希算法 7](#_Toc460214927)

[**2.3.1** 基于低频的均值哈希算法 8](#_Toc460214928)

[**2.3.2** 基于DCT的哈希算法 8](#_Toc460214929)

[**2.4** MVC模式及Spring Framework简介 10](#_Toc460214930)

[**2.4.1** MVC 模式 10](#_Toc460214931)

[**2.4.2** Spring Framework 11](#_Toc460214932)

[**2.5** 本章小结 13](#_Toc460214933)

[第三章 系统的分析与设计 14](#_Toc460214934)

[**3.1** 系统需求分析 14](#_Toc460214936)

[**3.1.1** 项目概述 14](#_Toc460214937)

[**3.1.2** 系统的功能性需求： 14](#_Toc460214938)

[**3.1.3** 系统的非功能性需求 18](#_Toc460214939)

[**3.2** 系统总体设计 19](#_Toc460214940)

[**3.2.1** 工作流程设计 19](#_Toc460214941)

[**3.2.2** 系统架构设计 21](#_Toc460214942)

[**3.2.3** 数据库设计 24](#_Toc460214943)

[**3.3** 本章小结 26](#_Toc460214944)

[第四章 内容管理系统的实现 27](#_Toc460214945)

[**4.1** 身份认证模块的实现 27](#_Toc460214948)

[**4.2** 内容资源上传下载模块的实现 29](#_Toc460214949)

[**4.3** 文档服务模块详细设计 32](#_Toc460214950)

[**4.4** 图像检索模块的实现 37](#_Toc460214951)

[**4.4.1** 图像检索的流程 37](#_Toc460214952)

[**4.4.2** 图像检索算法的具体实现 38](#_Toc460214953)

[**4.5** 分类管理模块的设计实现 41](#_Toc460214954)

[**4.6** 本章小结 43](#_Toc460214955)

[第五章 系统测试 44](#_Toc460214956)

[**5.1** 自动测试 44](#_Toc460214959)

[**5.2** 人工测试 45](#_Toc460214960)

[**5.3** 图片检索测试 48](#_Toc460214961)

[**5.4** 本章小结 49](#_Toc460214962)

[第六章 总结与展望 50](#_Toc460214963)

[**6.1** 总结 50](#_Toc460214965)

[**6.2** 未来的工作 50](#_Toc460214966)

[参考文献 51](#_Toc460214967)

图 录

[图 2‑1 DFT与DCT的比较 16](#_Toc460215290)

[图 2‑2 MVC模式 18](#_Toc460215291)

[图 2‑3 Spring Framework 组织结构 19](#_Toc460215292)

[图 3‑1 系统内部用户用例图 22](#_Toc460215293)

[图 3‑2 系统外部用户用例图 25](#_Toc460215294)

[图 3‑3 内容的生命周期 27](#_Toc460215295)

[图 3‑4 系统工作流程 28](#_Toc460215296)

[图 3‑5 系统架构图 29](#_Toc460215297)

[图 4‑1 登录流程图 34](#_Toc460215298)

[图 4‑2系统登录界面展示 35](#_Toc460215299)

[图 4‑3 执行请求的一般流程 36](#_Toc460215300)

[图 4‑4 上传图片资源界面展示（测试环境） 37](#_Toc460215301)

[图 4‑5 文件上传一般流程 38](#_Toc460215302)

[图 4‑6 pdf转换逻辑 41](#_Toc460215303)

[图 4‑7 在线编辑界面展示（测试环境） 42](#_Toc460215304)

[图 4‑8 文档在线编辑流程 43](#_Toc460215305)

[图 4‑9 图片检索流程 44](#_Toc460215306)

[图 4‑10 搜索界面展示 45](#_Toc460215307)

[图 4‑11图像缩小效果 46](#_Toc460215308)

[图 4‑12 图片上传流程图 48](#_Toc460215309)

[图 5‑1 Junit单元测试实例 52](#_Toc460215310)

[图 5‑2 自动测试分支覆盖率 52](#_Toc460215311)

[图 5‑3 检索测试图片 55](#_Toc460215312)

表 录

[表 3‑1 系统数据库表 33](#_Toc460215549)

[表 4‑1 用户表说明 36](#_Toc460215550)

[表 4‑2 上传记录表 40](#_Toc460215551)

[表 4‑3 文档表结构 41](#_Toc460215552)

[表 4‑4 图片哈希表结构 48](#_Toc460215553)

[表 4‑5 分类信息表结构 50](#_Toc460215554)

[表 4‑6 分类详细信息表结构 51](#_Toc460215555)

[表 5‑1用户登录测试用例 54](#_Toc460215556)

[表 5‑2 上传下载测试用例 54](#_Toc460215557)

[表 5‑3 文档相关测试用例 55](#_Toc460215558)

[表 5‑4分类相关测试用例 56](#_Toc460215559)

[表 5‑5 汉明距离测试结果 57](#_Toc460215560)

1. 绪论

# 

* 1. 论文的研究背景及意义

二十一世纪是一个信息爆炸的世纪。随着互联网技术的普及应用，各种各样的内容充斥于网络之上，文档、图片、视频影片、web页面等等，令人目不暇接。海量的信息内容极大的丰富了人们的生活，但随着内容的进一步丰富，我们发现，想要找到自己需要的内容信息却变得越来越困难。对于企业而言亦是如此, 调查表明,大中型企业平均会有至少一个主要的内容库每天都在运行中[1]。一个互联网企业，在自身发展过程中，往往会产生大量的图片，文档等素材，如不能对其实现有效的管理，则会造成资源的浪费，重复开发等问题。

中国的互联网企业，往往缺乏内容管理的传统。开发者本身往往缺乏编写文档意识，而即使形成了简单的接口文档，也往往是保留于开发者手中，在有人需要的时候在通过近似“口耳相传”的方式传播，这对系统的后期维护造成了极大的困难，人员变更时的工作交接更是成为了一个噩梦。同时，开发时实用的美工素材等往往也得不到有效保存，导致不同部门甚至同一部门下的不同产品风格各异，重复的开发工作也加重了美工，UI设计人员的工作量。而内容管理系统则是解决企业这一困难的有效工具。

内容管理，即Content Management(CM)，是指一系列用以支持对内容的收集，管理，发布的过程与技术的集合，其关注的重点放在各种非结构化或半结构化的数字资源的采集、管理、使用方案、传递和增值上，并可以集成到结构化数据的环境中[2]。这里的内容广义上讲，可以是任何形式的信息；狭义上讲，则是指通过电脑存储的数字信息。具体而言，内容可以是文本，图片，音频、视频等多媒体信息，或任何需要被管理的文件类型。

内容管理技术已经被广泛应用于各种领域之中，例如新闻媒体，电商网站等，都对内容管理有着天然的需求。同时，用户对于内容管理系统的需求也是多种多样的。有些使用者可能更关心海量内容的存储，而另一些则关心如何快速准确的搜索到需要的内容信息。因此，对很多企业而言，通用化的内容管理系统可能并非不能很好的适应其需求，其所需要的是功能更加灵活并且订制化的内容管理系统。

普通的内容管理系统的实现，在存储，版本控制，文本搜索等基本功能上已经日趋完善，然而基于文本内容检索图片则往往不能取得很好的效果。在目前多数企业应用的内容管理系统之中，起搜索功能往往是基于关键字，或是TAG（标签）的。对于文本类内容而言，这样的搜索形式已经足够满足大部分使用需求，而对于图片等难以结构化存储的内容数据而言，其效果往往较差。图片类资源的关键字，TAG等，往往需要用户在上传资源时主动添加。在系统中资源数量积累到一定程度后，此种搜索方式只能模糊定位，用户还需要在搜索结果中进行二次过滤，才能找到自己需要的资源。同时，也容易造成资源重复上传的问题，浪费存储空间。在这种背景下，让用户可以通过“以图搜图”的形式，例如使用分辨率较低的缩略图获取高清的图片资源，或是通过手绘的简略内容，获取类似的图片素材文件，就具有十分现实的意义。如何将图像识别技术应用于内容管理系统之中，为用户带来更好的使用体验，将是本文的重点研究内容之一。本文将在实现内容管理基本功能的同时，致力于将图片检索技术应用于内容管理平台之中，实现“以图搜图”的功能，让美工人员，UI开发人员可以更加方便的寻找到自己需要的图片素材，提高工作效率，具有良好的现实意义与应用前景。

* 1. 国内外研究现状
     1. 内容管理技术的国内外研究现状

从内容产生的时刻起，内容管理也就伴随着应运而生了。而“内容”本身的历史也可以说是人类认识的历史[3] 。例如图书馆书籍的管理，家谱族谱等等，都可以认为是内容管理的范畴。而数字化的内容管理技术被广泛认知则是在互联网应用流行以后[4]。IBM曾把信息管理定义为数据管理(DM Data Management)和内容管理(CM Content Management)两部分集成，显示出内容管理的重要性[5]。

用于内容管理的数字化软件应用被称为内容管理系统(Content Management System)[6][7][8]。内容管理系统的概念从被提出至今，其内涵随着数字技术、互联网技术的发展不断丰富。在当前的技术条件下，基于Web技术的内容管理系统已经成为了主流的发展方向。目前国外从事内容管理系统研发的主要厂商包括Mambo、Vignette、Interwoven、BroadVision、Openmarket、ATG、Allaire、Documentum、Hummingbird 等[9]，这些公司的内容管理产品与解决方案大多面向企业级用户，具有很强的专业性。TRS与TurboCMS则是国内影响力较大的两个内容管理产品[10]。

* + 1. 图像检索技术的国内外研究现状

“图像检索”，顾名思义，即指对图像进行搜索的过程。在过去数十年所产生的信息爆炸中，内容信息的数量呈几何倍数增长，而其中很大一部分是非文本类的数据。这同时也意味着这些非文本类的数据的搜索需求也有了惊人的增长。

传统的图像检索技术是基于描述的。即用户人为地对图片进行描述，或给图片打上关键字标签（TAG），本质上仍是文本的检索。这种方式有如下几个严重的缺陷：

1. 必须对图片进行描述才能搜索。这个过程可能需要人为进行，自动化的效果不高，并没有实际意义。
2. 检索精度不够。例如用户输入“动物”，“鸟类”这样较为模糊的关键词时，很有可能还需要花费大量精力对结果进行二次筛选。
3. 对于复杂的图片描述困难。对于内容很丰富的图片，只能使用简单的关键词描述，会使检索的结果大打折扣。

因为存在着上述困难，基于内容的图像检索(Content Based Image Retrieval, CBIR)[11]就应运而生了。CBIR并不需要对图片进行描述，而是依据图片本身的内容，根据用户提供的相似图片进行检索，这就大大提高了检索的精度。CBIR的实现依赖于两个关键步骤：图像的特征提取和匹配。

图像特征提取：图像的特征是多方面的。上文所述的“关键字”本质上也是对图像特征的一种描述。限于目前的技术发展水平，计算机只能自动化的识别图像的一些低阶特征，如纹理，颜色，形状等。而在图像检索算法中，检索的基础则正是这些低阶视觉特征。提取这些视觉特征，并将其转化为数字特征以便被计算机识别，就成为了图像检索算法的重中之重，也是一大难点。

图像特征的匹配：在正确提取了图像特征之后，就可以对数据库中的图像进行索引以供检索。图像特征的匹配指的是在用户给出了目标图像之后，如何快速准确地通过索引定位到相似图片的过程。

CBIR从1992年被提出至今，已经经过了二十多个年头的发展，并取得了一定的成果。目前已经有一些较为成熟的算法被提出。选择合适的图像检索方法，并将其应用于内容管理系统之中，为用户检索内容素材提供更加便捷的途径，是本文的重点研究内容之一。

* 1. 论文的研究目的和组织结构
     1. 论文的研究目的

研究分析和研究网站内容管理系统的软件架构、工作原理和功能应用。将内容管理系统与企业的具体需求相结合，针对企业文档、图片素材多，人员权限需要细致管理的特点，对系统以及各个模块进行详细设计。

研究图像检索的相关技术。查阅，学习相关研究论文，了解掌握目前主流的几种图像检索算法的原理，并根据系统的需求以及企业软硬件条件的限制，从其中选取几种合适的算法。分析，比较其作为内容管理平台图像搜索工具的优劣，从中选取最合适的算法，将其集成至内容管理系统之中。

编码实现内容管理系统。选取合适的开发框架，实现内容管理系统，并对其进行测试验证，最终达到可交付的水平。

* + 1. 论文的组织结构

本文的章节安排如下：

第一章为绪论，描述了课题产生的背景，对内容管理系统和图像检索技术作了简单的介绍，并阐释了本文的研究内容和研究意义。

第二章为相关技术及算法简介，前半部分针对内容管理技术，Java EE相关的框架开发技术作了简介。后半部分重点针对目前主流的图像检索算法作了简介，同时详细介绍了本文所选择的算法。

第三章为内容管理系统的分析与设计，主要进行了系统的需求分析，并在此基础上对系统进行了整体设计，将系统划分为各功能模块，简述了每个模块所起的作用。

第四章为内容管理系统的实现，在第三章的基础上，对系统的各个模块进行详细设计，对系统的关键模块，如文档管理，分类管理，尤其是图片内容检索模块进行了详细的阐述并给出了实现方式。

第五章为系统测试，从人工测试以及自动测试两个方面对系统测试过程进行了阐述，重点介绍了图像检索，内容分类管理等模块的测试过程。

第六章为总结与展望，首先总结了本文的工作及意义，然后对未来可进行的进一步研究工作做了展望。

1. 相关技术及算法简介

# 

本文的研究目的是构建一个基于CBIR的内容管理系统，而其中基于内容的图像检索技术(CBIR)是本文的研究重点。本章将从三个方面对本文所使用到的关键技术进行介绍：首先是内容管理相关的技术，其后对图像检索技术，特别是本文所使用的感知哈希算法进行了重点阐述。

1. 1. 内容管理关键技术简介

内容管理的含义十分广泛，对于内容管理系统应该如何构建，也缺乏统一的标准，各个公司实施的内容管理系统均具有各自的特点，但仍不失一些共性。一般认为，一个完整的内容管理系统至少需要包含以下几方面的内容：

1. 收集系统：即负责进行内容的收集、获取、分发、编辑、整合及转换的工作，同时应该能够满足用户对内容的查询需求。
2. 管理系统：负责对内容的存储进行管理，确保内容存取过程的正确性。同时应该具有对内容版本更替、分支的控制、记录功能。针对系统内不同人员的角色控制也可归入管理系统之中。
3. 工作流系统：即保证内容管理流程可以正确运行的系统。内容管理的流程包括了内容从收集起，经历存储，展现，一直到内容的发布。

内容管理始终是围绕着内容进行的，而以下几个方面则是尤为重要的：内容存储，内容查询。

内容存储：对任何一个内容管理平台而言，其所存储的内容都是相当广泛的。从简单的文本内容，到图片、音频、视频等多媒体内容，以至于压缩归档，可执行程序等等，都可以作为需要被管理的内容。而对内容的存储则是内容管理系统需要解决的首要问题。随着云技术的发展，传统的存储方式已经逐渐被新的分布式存储解决方案所替代。内容管理平台往往需要管理海量的内容信息，因此分布式存储解决方案将是最好的选择。本文所选择的存储方案是百度公司内部开发的NFS存储系统。

内容查询：内容管理系统需要将内容正确的呈现给用户，同时也需要给用户提供快速准确定位内容资源的手段，因此需要合适的内容搜索方法。目前内容管理平台的搜索方法主要有以下几种：

1. 基于标题的搜索：任何内容文件都具有标题，可以根据标题建立索引以供内容检索。
2. 基于标签的检索：用户在发布，编辑内容时，可以根据内容的特征为内容打上标签(TAG)，并据此建立索引以供内容检索。
3. 基于内容的检索：目前主要是针对文本内容，可以直接根据文本的内容信息进行内容检索。

对于图片内容信息，也可以根据内容进行检索。依据内容的图像检索(CBIR)可以满足用户某些特殊的检索需求，如根据缩略图搜索高清原图，根据手绘简图进行相似图像检索等。有关CBIR技术将在下节详细介绍。

* 1. 图像检索技术主流算法简介

CBIR从二十世纪八十年代被提出以后，一直是计算机领域一个热门的研究内容。很多学者，互联网公司纷纷投入其中，美国的谷歌公司，微软公司，国内的百度公司等都纷纷推出了“以图搜图”的引擎。

CBIR的两个核心研究内容是图像特征的提取和图像特征的匹配，两者都是针对“图像特征”而言的。

图像特征，即可以被用来描述图像的一系列内容，可分为图像的全局特征与局部特征，而全局特征又可进一步划分为颜色特征，纹理特征，形状特征等。

颜色特征：颜色特征是图像最显著的特征，具有稳定，旋转、评议、尺度变化无关性，且颜色特征计算简单，表现出很强的鲁棒性[12]。颜色特征的提取一般和颜色空间相关，常见的颜色空间包括RGB空间，HSV，YCrCb[13]等。可以通过颜色直方图[14]，颜色矩[15]，颜色协方差矩阵[16]，颜色布局等来表征图像的颜色特征用以进行图像检索。

纹理特征­[17 高仅仅]：纹理特征主要描述的图像的相随灰度集火颜色的某种规律性变化。各种物体都有各自的纹理特征。比较著名的纹理特征是T.Ojala于2002年提出LBP方法[16]等。

形状特征：相比图像特征和纹理特征，形状特征承载了较多的语义信息，因而在图像检索中可提供比较强的描述能力。形状特征的表示方法包括了傅立叶描述子[17]，多边形逼近，不变矩，B样条，形变模板和曲率尺度空间等。

除了上述颜色、纹理、形状等全局特征，还可以使用图像的局部特征表征图像用以进行检索。相比选举特征，局部特征对于很多尺度变换，摄影变换都是鲁棒的。

在基于图像局部特征的检索算法中，SIFT算法应用比较广泛。SIFT，即尺度不变特征变换（Scale-invariant feature transform，SIFT），是一种针对图像局部特征对图像进行特征提取的算法，由David Lowe于1999年提出[18]。

SIFT具有位置、尺度、旋转不变性，即无论图片进行缩放，旋转，改变亮度或拍摄视角，都不影响算法对图片的特征识别，用于图像的检索和匹配时具有一定的优势。

SIFT算法对图像的尺度，位置均具有不变的特性，图像的旋转亦不能对算法结果造成影响；同时，改变图像视角，对图像进行仿射变换也不会大规模改变算法结果。SIFT的结果蕴含十分丰富的信息内涵，碰撞的可能性亦较小。基于这些特征，SIFT作为图像检索算法，具有检索快速，结果准确的特点。

* 1. 感知哈希算法

哈希函数(Hash Functions)，是一类能够将变长数据映射为定长数据的函数，被广泛应用于计算机的各个领域中，哈希表就是哈希函数的一种最典型应用。除了数据查询外，因为哈希函数具有不可逆的特点，也被应用于信息安全领域，数字签名技术就是哈希函数在信息安全领域的一个典型应用。

传统的哈希方法往往是直接针对数字内容进行运算的。例如一幅图片，可以直接根据其二进制表示通过一系列复杂的运算计算出哈希值。这样的方法运算简单，但由于其与图像本身的内容无关，因而产生了信息的丢失。同一幅图片，经过放缩，裁剪等操作，或者改变压缩方式，都会导致其二进制表示产生巨大的变化，从而彻底的改变哈希值。而从人类认知的角度而言，这些操作并没有改变图片的本质信息，他们是“相似”的图片，也因此应该具有相近的哈希值。而传统的哈希方法并不能满足这一需求，因此针对多媒体文件的感知哈希方法被适时提出。

感知哈希函数，是基于认知心理学的信息加工理论，由多媒体数据集到多媒体感知摘要集的一类单向映射，将具有相同感知内容的多媒体数字表示唯一地映射为一段数字摘要满足感知安全性要求[19]。所谓感知特征，即基于人类认知所总结出的特征信息，如图片的颜色，布局等信息。对于图像的放缩，压缩等操作几乎不会改变这些感知特征。感知哈希函数正是基于对媒体内容的感知特征计算，其显著特点是具有相似感知特征的对象也具有相似的感知哈希值。

感知哈希方法计算迅速，占用存储空间小，在检索时，可以通过欧式距离或汉明距离确定图像是否相似，非常适合用于系统资源有限的系统中作为图像检索算法。

* + 1. 基于低频的均值哈希算法

任何一张图片都可以被视作一个两维信号，包含了不同的频率成分。频率的高低由亮度变化的剧烈程度描述：亮度变化剧烈的区域是高频成分，如图片中不同物体的边缘，往往亮度变化十分剧烈，这样的区域表征了图像的具体细节；反之，亮度变化平缓的区域则是低频成分，直观上讲，就是代表了图像中“连绵不绝”的部分，描述了大范围的信息。因此可以说，图像的低频成分表征了图像的框架结构，而高频成分则表征了图像的具体信息。图像越大（分辨率高）则细节越详细，越小则越粗糙。均值哈希算法就是一种基于图像低频信号的哈希方法，其具体步骤如下：

1. 缩小图片：将图像的像素尺度缩小为8\*8大小。缩小图像实际上是对图像进行降采样的过程，在降采样中，图像的细节信息被过滤掉，即图像的高频信息丢失了，只保留了图像的低频信息。缩小图像的最重要意义在于降低图像哈希的维度，便于特征的存储与检索。
2. 简化色彩，计算均值：将图像转化为灰度图像。这样操作进一步对图像信息进行过滤。将图像转化为灰度图像后，计算8\*8共64像素的图像的灰度均值。
3. 计算图像哈希：对每个像素，将它的灰度与图像灰度均值进行比较，灰度小于均值的记为0，反之记为1，将64位的比较结果连成一个完整的整数，即生成了一个图像的哈希值。组合时必须要保证每幅图像的组合次序相同。

均值哈希算法生成的哈希值对图像的尺度大小，颜色变化，形变等并不敏感，即无论如何改变图像的颜色，亮度，或是缩放图像，都不会对最终产生的哈希值造成太大影响。

由于均值哈希每一位是二值的，因此在检索时直接计算目标图像与数据库中图像的汉明距离即可，因此均值哈希算法应用于图像检索时计算十分迅速。

但均值哈希算法受均值影响很大，对图像进行伽马矫正或直方图均衡就会影响均值，从而影响最终的hash值，基于离散余弦变换(DCT)方法则在均值hash方法的基础上很好的解决了这个问题。

* + 1. 基于DCT的哈希算法

离散余弦变换(Discrete Cosine Transform，DCT)，是与傅里叶变换相关的一种变换，类似于离散傅里叶变换，但是仅使用实数。DCT可以将二维图像映射到频域。DCT有8种不同的标准形式，其中第二类(type-II)为最常用的形式，也通常被直接等价于离散余弦变换，可由公式2-1表示：

|  |  |  |
| --- | --- | --- |
|  |  | ( 2‑1 ) |

其中x[m]代表一个长度为N的实数序列。也可以用公式2-2的简化形式表示：

|  |  |  |
| --- | --- | --- |
|  |  | ( 2‑2 ) |

其中c[n, m]代表了第n行m列的DCT矩阵系数。

DCT本身是一种离散的线性变换，针对二维图像，可以使用两个DCT系数矩阵对其进行变换，可由公式2-3表示如下：

|  |  |  |
| --- | --- | --- |
|  |  | ( 2‑3 ) |

其中I代表一幅正方形图像（宽高相等），M代表DCT系数矩阵。DFT与DCT的比较可用图2-1表示：

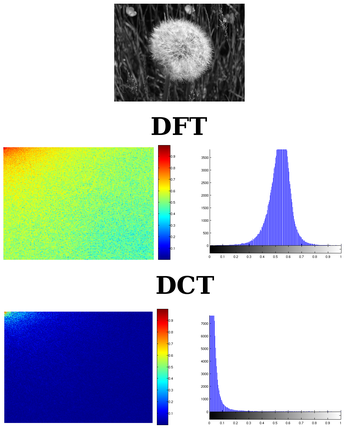


图 2‑1 DFT与DCT的比较

Fig 2‑1 Comparison between DCT and DFT

与离散傅里叶变换(Discrete Fourier Transform)相比，在离散余弦变换所得到的系数矩阵中，从左上角到右下角，频率越来越高。

DCT的一个显著特征是，在原图进行一定程度的改变后，变换所得到的DCT系数矩阵中的低频分量是基本稳定的。也就是说在图像的感知特征没有发生严重变形时，对图像进行DCT所得到的系数矩阵的低频分量也不会有巨大的变化。这是因为图像的明显感知特征具有集中在DCT低频分量的趋势。DCT的这一特征使得它很适合用来计算图像的感知哈希值。

基于DCT的感知哈希算法的基本运算过程如下：

1. 缩小图片尺寸：为了简化DCT的计算量，需要对图片进行缩小为大于8\*8的分辨率。一般会缩小为32\*32。
2. 简化色彩：与均值哈希函数相同，在缩小图片之后仍然需要将其转化为灰度图像，这一步可以进一步简化DCT的计算量。
3. 计算DCT系数矩阵：将图片通过DCT计算，得到系数矩阵。由于原图为32\*32，得到的系数矩阵也是32\*32的，为了降低哈希值维度，只保留系数矩阵左上角8\*8的区域，这部分呈现了图片的最低频率。
4. 计算DCT均值：将64位DCT系数求平均，得到DCT平均值。
5. 计算哈希值：最后一步与均值哈希相同，根据8\*8的DCT系数矩阵，对于高于DCT均值的位，设置为1，反之则设置为0，将64位组合，就得到了图像的哈希值。

如此得到的哈希值并不能完整的反应图像的低频信息，只能粗略的展现各频率分量相对于平均值的比例。与均值哈希相同，基于DCT的哈希方法也对图片的色彩，形变不敏感，只要图片的整体结构不变，哈希值就不会改变。同时它也克服了均值哈希的一些缺点，能够避免伽马矫正和颜色直方图被调整对图片带来的影响。

* 1. MVC模式及Spring Framework简介
     1. MVC 模式

MVC，即Model-View-Controller的简写（模型-视图-控制器），是一种软件设计的体系结构[20]。MVC模式将软件应用分为三个互相联系的部分，三部分之间具有较低的耦合度，可以血协同工作实现系统业务功能。这种三层结构的划分使得系统的开发得到简化，同时也提高了系统的灵活性与可扩展性。

MVC的三层结构之间的关系可用图2-2描述：



图 2‑2 MVC模式

Fig 2‑2 MVC Architecture

模型（model）：模型是指直接对业务数据进行控制的部分，同时也包含了底层的业务处理逻辑。

视图（view）：视图提供了系统与用户的交互功能，将数据展现给用户。一幅图表，一个表格都可以看作一个视图。视图应具有良好的灵活性，可以用多种类型的视图描述同一份数据。

控制器（Controller）：控制器的主要作用是接受用户的输入信息，将其传递给模型层，接受模型的返回，选择合适的视图返回给用户，或者通知视图进行改变。

MVC模型已经在Web应用中得到了广泛的使用，很多流行的web开发框架都有自己的MVC实现。例如Ruby on Rails，Django（Python），ASP.NET MVC等。

本系统作为一款Web应用，十分适合使用MVC模式进行开发。其中模型层负责管理数据源，包括内容资源文件，用户信息等。同时模型层还负责底层数据之间的互相通信交互，即底层的业务逻辑。控制器层接受用户的输入信息并选择合适的视图返回给用户。视图层负责将内容资源合理地展现给用户。

* + 1. Spring Framework

随着Java技术的流行，市面上涌现了很多基于Java EE的开发框架。早期的开发框架如EJB等，在为企业级应用开发提出一套解决方案的同时，仍具有较高的复杂性。为降低这种开发的复杂性，Spring[21]，一款开源框架应运而生。自从[Rod Johnson](https://en.wikipedia.org/wiki/Rod_Johnson_(programmer))于2003年发布了第一个版本以后，Spring已成为了基于Java EE的最流行的开发框架之一。

Spring框架采用了分层的设计方式，其最终要的组件是Spring核心容器(Spring core)，核心容器贯穿Spring中bean的生命周期始终。Spring通过一系列配置文件，或是注解的方式定义bean。在Spring核心之上，还有6个其他的模块，这些模块相互协作，构成了完整的Spring Framework。Spring的体系结构图2-3所示：

****

图 2‑3 Spring Framework 组织结构

Fig 2‑3 Architecture of Spring Framework

Spring的核心特点之一是控制反转模式[22]（Inversion of control，IOC），或称依赖注入。依赖注入使用一种全新的方式来创建对象。当使用IOC时，程序并不在代码中直接创建对象，而是通过配置文件等方式描述了对象的初始化方式。同时，对象与其依赖的组件也并非在代码中显式的连接，而是以配置的方式注明了对象间的依赖关系，在由容器将组件注入，完成依赖关系的构建。

与传统的编程方式不同，控制反转模式并不需要再主程序中主动创建对象，而是实用配置文件定义好对象，然后由Spring容器创建这些对象，然后将他们“注入”到程序之中。控制反转降低了编程中的耦合性，可以在不修改原程序语义的条件下修改程序组件（Component），甚至实现对象的热插拔。

AOP是Spring的又一核心组件。AOP，即面向切面的编程（Aspect-oriented programming），它允许程序员对横切关注点或横切典型的职责分界线的行为（例如日志和事务管理）进行模块化。AOP 的核心构造是切面，它将那些影响多个类的行为封装到可重用的模块中。AOP进一步降低了应用程序的耦合度，为程序的开发与维护提供了方便。

Spring从发布至今已经历了十几个年头，Spring Framework 4.0已于2013年发布。Spring可以极大的简化基于Java EE的企业级应用的开发。本系统将基于Spring MVC进行开发。

* 1. 本章小结

本章的重点是对论文研究所使用到的关键技术进行简要介绍。第一小节主要介绍了内容管理技术的研究重点，分析了内容管理系统的基本结构，并对内容存储，主要是论文所使用到的分布式存储系统做了简单介绍。第二小节对CBIR技术进行了简单的介绍，并介绍了几种常见的CBIR算法。第三小节主要阐述感知哈希算法，详细介绍了三种感知哈希算法：均值哈希算法，pHash算法，差异哈希算法。最后对本文设计系统所使用的MVC模式以及Spring Framework进行了简单的介绍。

1. 系统的分析与设计

# 

本章主要进行内容管理平台的系统需求分析以及总体设计。需求分析阶段的主要任务是从用户的角度出发，总结，整理出详细的系统用例，描述出待开发的软件的功能、性能、数据、界面等要求。总体设计阶段的主要任务是根据系统的需求，选取合适的开发模型，并对系统进行模块划分，从总体上对系统功能等进行描述。

* 1. 系统需求分析
     1. 项目概述

本论文的研究题目取自百度(中国)有限公司所开发的基于图片检索的内容管理平台项目。随着公司业务的拓展，在开发过程中所产生的文档、图片素材不断积累，需要一个统一的平台对其进行管理。本项目具有以下特点：

1. 主要针对文档和图片素材进行管理，用户可以进行对内容资源的检索、上传、编辑、发布等操作。
2. 公司内部人员可以将公司内网资源分享给外部人员，使得外部人员在不具有内网访问权限的条件下仍能获取需要的内容资源。
3. 针对图片素材资源，可以进行基于图片内容的检索，即以图搜图，提高对图片内容的检索准确率。
4. 平台界面简洁，用户没有使用困难。
   * 1. 系统的功能性需求：

功系统的功能性需求，即侧重于“功能”，也就是定义了系统的行为方式。对于一般的软件产品而言，数据都是贯穿于其功能定义的核心之一。因此，从数据的角度而言，系统的功能性需求定义了软件系统对数据生命周期的管理，包括了对数据的记录，传输，显示，转换等处理过程。

需求分析是针对用户而言的，在进行需求分析时必须要首先把握系统的目标用户是那些，具有什么特点。在系统功能较为复杂时，需要对用户群根据特点进行分类，以便简化需求分析过程。

就本系统而言，系统主要负责对开发过程中产生的文档、图片素材进行管理，因此主要面向的是公司内部的开发人员，如网站开发者，美工人员，产品经理等。系统也将部署于公司内网。同时，在某些场景下，系统的使用者可能需要将平台资源分享给外网用户，这就是说外网用户也是系统的潜在用户群。而内网用户和外网用户在权限上将会有明显的区别。

内网用户指的是公司内部用户，主要是网站开发人员，美工人员，用户体验设计人员，产品经理等。内部用户可以任意浏览系统绝大部分内容资源，并可以对其进行检索，编辑，下载等，内部用户也可以上传、发布新的资源。下面将详述内部用户的功能需求。内部用户的用例见图3-1：



图 3‑1 系统内部用户用例图

Fig 3‑1 Use Case of Inner User

身份认证：

用户必须进行登录才能认证身份为内部用户，因此登录、登出是内部用户的天然需求。本系统需要与公司其他系统进行联动，即需要进行单点登录、登出。

内容浏览与检索：

用户对内容资源的最基础需求即是浏览与检索。系统应该为用户提供简洁明了的浏览界面。内容浏览与检索具体包含以下几方面的内容：

1. 资源浏览：用户可以查看系统中的内容资源。查看时的界面应该保持简洁，对于图片资源需要展示缩略图。查看时应该对内容资源进行分类，让用户可以更方便的查看资源。同时用户可以对资源进行排序等操作。
2. 查看内容详细信息：用户点击缩略图之后，可以查看内容的详细信息，如果是图片类型素材，则展示大图，如果是文本类型素材，则展示文本内容。
3. 资源检索：用户可以根据需要对资源进行检索。检索分为三种方式：基于关键词，基于内容，基于特征。基于关键词的检索指的是根据内容资源的标题、人工添加的标签(TAG)进行检索。基于内容的检索指的是根据文本的内容，或是根据图片的内容（以图搜图）进行检索。基于特征的检索指的是，根据资源的发布者，发布时间，修改时间等内容的固有特征进行检索。资源检索是内容管理系统的一个重要功能，实现良好的检索功能可以让内容管理平台的实用性大为提高。

分类管理：

用户可以对系统内的资源进行分类管理。分类即是对系统的内容资源进行分类，例如文档、超文本文档、图片等分类。系统将默认创建全部文件的分类，并且此分类不可修改。分类的管理应该具有严格的权限控制。

1. 分类浏览：在资源浏览界面中，分类类似于一个导航栏的作用。用户可以点击相应的分类浏览器中内容。在本系统中，分类是针对用户的，即不同用户可以拥有不同的分类。分类只是一种方便用户浏览的手段，而非内容文件本身的属性，换句话说，一个分类即是一个“过滤器”，只筛选出特定类型的文件。同一文件可以属于多个分类。
2. 创建分类：用户可以手动创建新的分类。创建分类时将会需要用户输入分类关联的文件格式。创建分类后，所有相关的文件格式的内容资源将会被展示在此分类下。
3. 修改分类：对于已经创建的分类，用户可以对其进行修改。可修改的内容有：分类的名称、分类包含的文件格式。分类被修改后系统视图应该即时响应发生改变。
4. 删除分类：用户可以删除一个已有的分类。删除分类的操作不会对文件内容文件本身产生任何影响。

内容的上传与编辑：

内容上传到系统之重后不是一成不变的，用户可以对其进行编辑与修改。内容的编辑包括内容本身的编辑，对内容特征的编辑，内容的删除等。

1. 上传内容：用户可以上传新的资源文件。为了存储方便，在用户上传某些特殊类型的资源后平台需对其进行转码。用户上传资源后应该有一个初始化的编辑操作。
2. 内容编辑：即用户对内容资源进行修改的行为，这里指修改内容本身。修改以后内容资源将会具有新的版本。在本系统中，内容分为文档类型和素材类型。只有文档类型的内容资源才可以进行内容修改。修改操作应该具有严格的权限检查，只有内容的拥有着（即上传者）才有修改的权限。
3. 内容特征的编辑：内容特征指内容的简介和标签。内容简介是一段限定长度的文字，对内容进行简单的描述。标签是指对内容特点进行描述的短字符串，可以为内容的分类、检索提供方便。用户可以为内容资源添加新的标签，或是删除已有的标签。内容特征的编辑操作不需要进行严格的权限检查。
4. 内容删除：用户可以删除自己有权限的内容资源。删除操作是不可逆的，因此在删除时需要用户进行确认。被删除的资源在整个平台都不再可见。

内容的分享与下载：用户可以使用平台对内容资源进行下载与分享。

1. 资源下载：用户可以下载需要的内容资源。下载可以进行批量操作。资源的下载应该具有权限控制。
2. 分享：在本系统中，分享是指内部用户为平台中的内容资源生成一个可供外部人员访问链接的过程。分享可以看作一个将内网资源发布到外网的过程。

文档类资源的团队管理：

在本系统中，一份文档的所属者可以是一个具体的用户，也可以是一个团队。当一份文档属于一个团队时，此团队中的所有用户都具有文档的使用权，可以对其进行编辑操作。团队具有团队管理员的角色，可以管理此团队。

1. 创建团队：任何系统的内部用户都可以创建一个团队。创建团队时需要为团队命名，系统不允许重复的团队名。团队的创建者将自动成为团队的超级管理员。
2. 添加成员：团队管理员可以向团队添加新的成员，添加的成员必须是系统内部用户。添加的成员默认具有普通成员的身份。
3. 剔除成员：可以将某个成员从团队中剔除，此操作只有超级管理员即团队的创建者可以执行。
4. 修改权限：团队超级管理员，即团队的创建者，可以修改团队普通成员的身份为团队管理员，团队管理员则可以向团队中添加成员。

文档类资源的版本控制：

由于本系统中的文档类资源可以进行内容编辑，同时可能会出现多个用户同时编辑一份文档的情况，因此需要对文档类的内容资源进行版本控制。

1. 创建新版本：这一操作将在文档被修改提交后自动执行。文档的内容一旦被修改成功，则系统会自动为文档创建一个新的版本。
2. 版本浏览：用户可以针对某一分文档浏览其所有的版本信息。版本内容将以树状图的形式展现给用户，用户可以点击任意版本对其内容进行查看。
3. 版本合并：对于两个不同的版本，可以将其进行合并。合并操作将会产生一个新的版本，此版本的父版本为合并前两个版本其中之一。
4. 删除版本：只能删除文档的最新版本，即无后续版本的版本。

外部用户指没有系统部署网络访问权限的用户，由于平台上的内容资源具有不同的密级，需要严格控制外部用户对系统的访问权限。在本系统之中，外部用户只能浏览指定的资源（通过内部用户的分享链接查看），或对其进行下载操作。外部用户的用例见图3-2：



图 3‑2 系统外部用户用例图

Fig 3‑2 Use Case of Outside User

* + 1. 系统的非功能性需求

除了功能性需求以外，根据用户的业务需要，软件系统还必须要满足的特性则称之为非公能性需求。典型的系统非功能性需求有如下几类：系统性能，可扩展性，可用性及可靠性等等。后文将针对一些比较重要的非功能性需求做简单的说明。

性能需求：性能需求反映了用户对于系统一些性能指标的要求。常见的性能指标包括了响应时间，数据吞吐量，并发量等。对本系统而言，内容资源的检索可能会成为性能瓶颈，需要谨慎考察。

安全性需求：一般而言，系统的安全性设计应遵循最小特权原则，即对所有身份的用户应该仅赋予其需求所需的最小特权。对本系统而言，用户区分为内部用户和外部用户，安全性主要是针对内部用户的身份认证，外部用户统一作为“游客”身份登入即可。

易用性需求：易用性反映了用户使用软件系统时需要的学习成本的高低。软件系统应被设计的简单明了，使用户能够迅速的上手使用。对本系统而言，易用性主要体现在系统的人机交互界面设计之中，简单简洁的用户界面可以有效的降低用户的学习成本，使用户可以更方便的上手系统的使用。

可靠性需求：系统的可靠性指系统可以稳定地为用户提供服务，不受或尽量减少意外情况的对系统的影响，也可成为可用性。云部署技术可以大大提高系统的可用性，同时也能使系统的部署上线更加方便。

可扩展性需求：考虑到系统的升级功能，必须在设计时保持系统的良好的可扩展性。同时提供与其他系统的接口，以便系统能够集成其他的功能。

* 1. 系统总体设计

本小节将遵循自顶向下的设计原则，根据前面所述的需求分析对系统进行总体设计。

* + 1. 工作流程设计

内容管理系统是围绕着“内容”这个中心运作的。在对系统的工作流程进行描述之前，需要先了解清楚内容的生存周期。而内容的生存周期，可以简单地概括为“创建”→“发布”的过程。

内容的创建：内容的创建是指内容资源在平台系统中从“无”到“有”的过程。内容管理系统专注于对内容进行管理，而非创作，因此对内容管理系统而言，内容创建的过程往往等价于将内容资源上传至系统并被系统存储的过程。其中上传的过程是用户可控的，而存储的过程则是对用户透明的，即用户无需关心系统以什么方式在何处存储用户上传的内容资源。

“内容”的含义是十分广泛的，对本系统而言，“内容”特指文件资源，具体可以包含如下的类型：（）。对某些特殊的资源类型，可能需要再上传后对其进行一定的处理（例如转码、切分等）。

内容的编辑：内容资源被创建之后并非一成不变的，用户可以对其进行编辑修改。用户对内容的修改具有两大类型：对内容本身的修改，对内容特征的修改。

对内容本身的修改将导致内容的版本变更，这会影响到所有收藏此内容的用户。内容管理系统本身不提供对内容的编辑功能，因此对内容本身的修改实际上是一个重新上传的过程。在上传新的版本后，系统应保留过去的版本。

对内容特征的修改则是修改内容的简介，标签等特征信息。内容的特征信息主要用来进行内容的检索。用户同时也可以对内容进行点赞，这一行为将影响内容的战术顺序。对内容特征的编辑并不会影响内容本身，也就不会产生新的版本，其影响的仅仅是内容的检索结果。

内容的发布：对本系统而言，内容的发布特指将内容从内网向外网发布的过程。主要的流程就是生成一个可供外部用户访问内容资源的链接，发布者可将此链接提供给内容资源的需求方。

内容的删除：内容的拥有者（或系统管理员）可以对内容进行删除操作。这一行为的原因主要是上传者上传了不符合规定的资源、系统中有了更好的替代资源等。内容的删除可以采取物理删除或逻辑删除的方式。物理删除是不可逆的，而逻辑删除则是简单的将内容“隐藏”起来，使其不再对前端用户可见。在本系统中，将采用物理删除的方式，也就意味着内容资源一经删除，将不再能被恢复。删除操作是内容生命周期的终点。内容的生命周期如图3-3所示：



图 3‑3 内容的生命周期

Fig 3‑3 Life Cycle of Contents

根据上述对内容生存周期的分析，可以设计出内容管理系统的工作流程。本系统的工作流程可以用图3-4概括：

图 3‑4 系统工作流程

Fig 3‑4 System Workflow

内容产生：即内容从上传到存储的一系列过程的集合。在本系统中，内容的存储系统与其他系统是分隔的，存储过程由存储模块进行统一管理。内容资源的存储将会以云存储的方式进行，过程中可能会涉及到对内容资源的切分重组，而这一过程对上层是透明的。内容在上传完毕到存储之间可能还会经过转码、压缩等一系列操作，视内容文件的格式而定。

内容管理：这一部分是内容管理系统的重中之重。包括内容的检索、浏览、编辑，用户收藏的管理，内容分类的管理等都包含在其中。内容资源的生命周期的大部分过程都处于内容管理的流程中。需要注意的是，无论内容管理采取怎样的操作，都不影响系统中的内容资源数量。内容管理操作不应产生新的内容、或是消去原有内容。

内容删除：在有些内容不再适合展现于系统中时，需要删除这些内容。内容的删除操作应该进行严格的权限管理。

* + 1. 系统架构设计

逻辑上，系统被分为了三层结构：存储，逻辑模块，前端展现。系统的架构如下图所示。下面详细对各个模块进行说明。

本系统的架构可用图3-5表示：

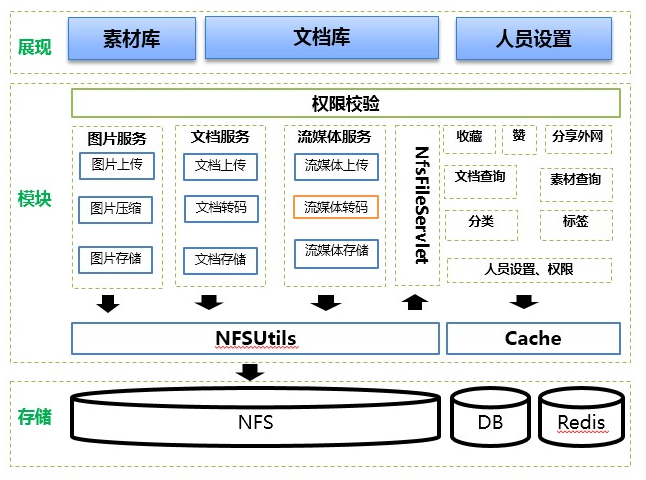


图 3‑5 系统架构图

Fig 3‑5 System Architecture

存储：

存储是系统一切数据持久化模块的总和。存储功能在任何一个Web应用中都是必不可少的，对内容管理系统则尤为重要。本系统的存储层由三个无耦合的部分组成：NFS，DB，Redis。

1. NFS：

CCDB Network File System是百度新存储体系的一部分，简称为CCDB-NFS，或NFS。其具有如下特点：

强一致：即文件的一次写入操作成功后，在所有客户端均可以实时读取到该次操作写入的内容。这是对上层透明的要求之一。

高可靠：文件的多个副本存在不同的File Server上，Master通过心跳检测File Server的存活，一旦File Server宕机，Master将迅速并发发起该File Server所包含文件副本的修复。

高可用：File Server的宕机会影响该File Server上所有文件的写入，以及所有主副本的读取。NFS Client对失败的操作会自动重试，因此后端的故障对上层应用是透明的。除此以外，NFS集群后端的升级可以做到不停服务。

低延时、高并发：NFS主要从两个方面降低延时、提高并发：(1)系统设计上针对大量小文件的情况进行优化；(2) 优化Master，一方面通过元信息的分布化降低对Master的访问压力；另一方面针对读多写少的情况，优化Master的查询性能，在打开内核网卡多通道的情况下，Master能够处理50w/s的并发查询请求。

大规模：NFS的一期实现上，单机群能支持5000台File Server，支持4亿文件（Master 128G内存）。

自运维：NFS内节点的状态监控、故障检测、修复、Master的主从切换都自动进行，无需人工干预。

透明：NFS的Client对外接口通过Linux内核的VFS实现，兼容绝大多数POSIX系统调用。对于绝大多数应用可以做到无缝迁移。后端节点故障的容错在Client内部会主动进行重试，不影响上层应用。

NFS主要用来对内容资源本身进行存储。

1. DB：

即Database，数据库。在本系统中，数据库主要用于存储除了内容资源的结构性数据，这些数据用于对内容资源进行描述或是对系统的其他部分其辅助性作用。例如内容资源名称，标签，存储地址等。内容管理的流程将建立在这些结构化数据之上。

1. Redis：

Redis是一个基于内存缓存的存储系统。具有简单的key-value存储模式，因此存取十分高效。Redis提供了对String，List，Set，SortedSet，HashMap等数据结构的添加，删除，压入，弹出等操作，并能够保证这些操作的原子性。同时，还能够支持有序的存储方式，并可以多种方式对数据进行排序。由于大部分关系型数据库要存储大量数据，都是基于磁盘的存储结构，Redis可以作为一个对他们的有效补充。

在本系统中，Redis主要作为数据库的补充，对一些不经常变动的关键数据进行缓存，或是存储一些没有强的持久化需求的数据。由于Redis采用内存对数据进行存储，因此读写速度远高于利用磁盘存储的MySQL数据库。

逻辑模块

逻辑模块指实现系统业务逻辑的部分。将系统的业务进行拆分，对每个业务功能将编写专用的模块，以实现业务的解耦。

NFSUtils：存储部分已经从系统的业务模块中剥离出去，这个部分则主要负责系统与NFS之间的交互工作。NFSUtils封装了本系统使用到的所有NFS接口，并且会对文件的读写做一定的预处理。

Cache：即缓存模块。本系统的缓存模块主要基于Spring的缓存接口实现，对某些关键数据进行缓存，以提高系统的响应速度。

图片服务模块：即主要负责对图像进行处理的模块。对图片上传，图片压缩，图片存储等功能提供接口。这里的存储主要指在对图片本身存储之外的一些工作，如记录图片存储位置等。

文档服务模块：即主要负责对文档进行处理的模块，对文档上传，文档转码，文档存储等功能提供接口。这里的存储主要指在对文档本身存储之外的一些工作，如记录文档存储位置等。

流媒体服务模块：在本系统中，流媒体主要指Flash文件。流媒体模块主要负责对流媒体进行处理，对流媒体上传，流媒体转码，流媒体存储提供结构。这里的存储主要指在对流媒体本身存储之外的一些工作，如记录流媒体存储位置等。

管理功能模块组：包含了一组模块，实现了内容管理过程中的各项需求，如内容检索，分类管理，标签管理等。

权限校验：权限校验的作用是认证当前用户是否为系统合法用户，并在用户实施操作时确认用户是否有相应的权限。

展现：即视图部分，负责将系统后台数据展现给用户。本系统的展现部分分为两个入口，素材库主要为图片、流媒体，展现时采用缩略图的方式，而文档库则主要包含了文本文档等数据类型，展现时仅展现文档名称、上传者等信息。人员设置则是对人员权限进行管理的界面。

* + 1. 数据库设计

对于任何一款Web应用而言，都具有需要持久化的数据内容。在对系统进行设计时，数据库设计也是重点内容之一。数据库设计的原则是在保证数据的完整性、安全性和可靠性的前提下，尽可能地提高数据库访问效率，同时兼顾可扩展性和伸缩性等。

数据库的选择是数据库设计的一个重要方面。本系统使用了本系统选择使用的数据库为MySQL。作为一款关系型数据库，MySQL由于其开源特性，已经得到了十分广泛的使用。在各种WEB应用中都能找到MySQL的身影，其中不乏一些知名的软件系统。

数据库表设计是数据库设计的另一个重要方面。从具体的业务内容中抽象出数据模型，合理地设计表结构，往往可以对应用的运行效率改善起到良性作用。

表3-1总结了本系统所使用到的所有数据库表，部分重要的表将在第四章详细设计时进行详解。

表 3‑1 系统数据库表

Table 3‑1 Infomation about Databases

|  |  |  |
| --- | --- | --- |
| 表名 | 中文表名 | 功能介绍 |
| tb\_material\_category | 素材分类表 | 存储了素材的分类，如图片类，文字类等。 |
| tb\_material | 素材表 | 存储了所有的素材文件，包含名称，文件类型，存储位置等 |
| tb\_material\_tag | 标签表 | 存储了所有素材的标签，包含了标签名称等 |
| tb\_material\_tag\_relation | 标签素材关联表 | 此表存储了标签与素材的关联关系 |
| tb\_upload\_record | 上传记录表 | 记录了用户的上传文件记录 |
| tb\_type\_info | 分类信息表 | 记录了分类信息 |
| tb\_type\_detail | 分类详细信息表 | 记录了分类的详细信息 |
| tb\_material\_like\_store\_info | 收藏记录表 | 此表记录了用户对素材的收藏记录 |
| tb\_material\_folder | 素材文件夹表 | 用户收藏的素材可以以文件夹的形式进行分类，此表存储了这些分类信息 |
| tb\_material\_folder\_relation | 素材文件夹关联表 | 此表存储了素材和文件夹之间的关联关系，即哪些素材属于哪些素材文件夹 |
| tb\_material\_hash | 素材文件哈希表 | 此表存储了图片素材的哈希值，用以进行图片检索，只有图片类型的素材会进行运算 |
| tb\_share\_record | 分享记录表 | 此表存储了内部用户对素材的分享记录，分享的素材将生成一个短链接，也会存储在这张表中 |
| tb\_document | 文档表 | 此表存储了所有文档类型的素材信息 |
| 表 3-1 系统数据库表(续) | | |
| 表名 | 中文表名 | 功能介绍 |
| tb\_document\_team | 文档团队表 | 文档隶属的团队信息表 |
| tb\_document\_category | 文档分类表 | 此表存储了文档类型内容资源的分类信息 |
| tb\_like\_record | 赞记录表 | 此表存储了用户对素材的点赞记录 |
| tb\_document\_team\_user | 文档团队成员表 | 此表存储了文档团的成员的具体信息 |
| tb\_document\_tag | 文档标签表 | 此表存储了文档类型内容资源的标签信息 |
| tb\_document\_tag\_relation | 文档标签关联表 | 此表存储了文档标签与文档之间的关联关系 |
| tb\_inner\_user | 内部人员信息表 | 此表存了所有系统内部用户的具体信息 |
| tb\_document\_outer\_record | 文档分享信息表 | 此表记录了所有文档的外部分享信息 |
| tb\_document\_version | 文档版本信息表 | 此表存储了所有文档素材的版本信息，在文档进行协同编辑而产生版本变更时，将会记录在此表中 |
| tb\_share\_record\_relation | 分享关系表 | 此表存储了分享操作的关系记录 |

* 1. 本章小结

本章对系统进行了需求分析和总体设计。第一小节对系统需求进行了拆分，在此基础上详细的对每一部分的需求进行了分析。第二小节是系统的总体设计，遵循自顶向下的设计原则，从三个角度对系统进行总体设计。首先是系统工作流程的角度，其次是系统架构的角度，最后阐述了本系统的数据库设计思想，并对系统中出现的数据库表进行了简单的介绍。

1. 内容管理系统的实现

# 

1. 1. 身份认证模块的实现

本系统作为一个内容管理系统，主要供企业内部员工进行内容资源的分享，共享编辑等，因此任何企业内部的员工都是系统的合法用户。作为企业内部应用集合的一部分，系统需要实现单点登录、单点登出，这个部分主要是依赖其他应用实现的。

登录的流程如图4-1所示：



图 4‑1 登录流程图

Fig 4‑1 Flow Chart of Login

1. 用户输入身份信息：包含用户名，密码和验证码。验证码主要用以防止恶意登录。登录界面设计如图4-2所示：

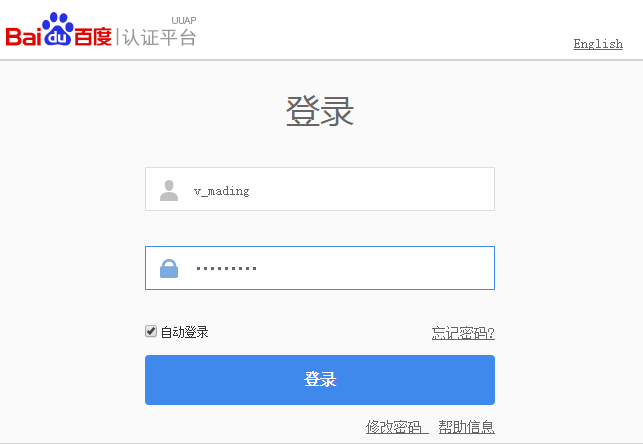


图 4‑2系统登录界面展示

Fig 4‑2 Login Interface

1. 认证用户：这个过程不由本系统负责，本系统会根据用户的身份信息调用外部接口对用户身份进行认证，如果认证通过，外部系统将会回调本系统执行下一步过程。系统之间的互相调用通过https的方式实现。
2. 创建用户：任何公司内部员工都是本系统的合法用户，但仍有身份的区别（即普通用户与管理员，同时一些系统功能，例如收藏，是基于用户身份的，因此系统还需要对认证成功的用户信息进行保存。如果某用户是第一次登录本系统，则需要为他在本系统中创建一个身份角色。在本系统中用户的属性至少包括用户名和角色类型，认证过程部署于本系统的职权，因此不需要（本系统也没有权力）保存用户的密码。可设计用户存储表tb\_user如表4-1所示：

表 4‑1 用户表说明

Table 4‑1 User Table

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 说明 | 特征 |
| id | BIGINT(20) | 主键id | 非空，自增 |
| user\_name | VARCHAR(64) | 用户名称 | 非空 |
| role\_type | TINYINT(4) | 用户类型，0为普通用户，1为管理员，2为超级管理员 | 非空，默认0 |
| add\_time | DATETIME | 用户的添加时间 | 非空，默认当前时间 |
| add\_user | VARCHAR(64) | 用户的添加人 | 非空 |
| update\_time | DATETIME | 用户的修改时间 | 非空，默认当前时间 |
| update\_user | VARCHAR(64) | 用户的修改人 | 非空 |

1. 存储用户身份信息：对于已经登录的用户，需要将其存储于系统之中。具体而言，对登录成功的用户，系统将会返回一个唯一的session\_id给前端浏览器，同时后台将在redis中保存此session\_id作为用户已登录的凭证。每次前端浏览器向系统后台发出逻辑请求（非静态资源类的请求）时，将此session\_id通过cookie的方式回传给后台进行身份认证，只有确认是已登录的用户才能正确的执行请求。session\_id的失效时间为24小时，即用户登录过后24小时，必须重新进行登录。请求的一般流程如图4-3所示：



图 4‑3 执行请求的一般流程

Fig 4‑3 Common Process of a Request

* 1. 内容资源上传下载模块的实现

对本系统而言，绝大多数内容资源都是通过用户的上传行为产生的。而用户直接获取内容资源的途径则是通过内容下载。在本系统中，虽然内容具有文本类型与素材类型之分，但上传与下载的逻辑基本是相同的，因此将上传与下载作为一个通用的模块来实现。内容资源的上传过程可概括如下：

1. 上传文件：用户上传的行为是通过前端浏览器执行的，前端浏览器收到用户上传的文件并将其传送给后台服务器。

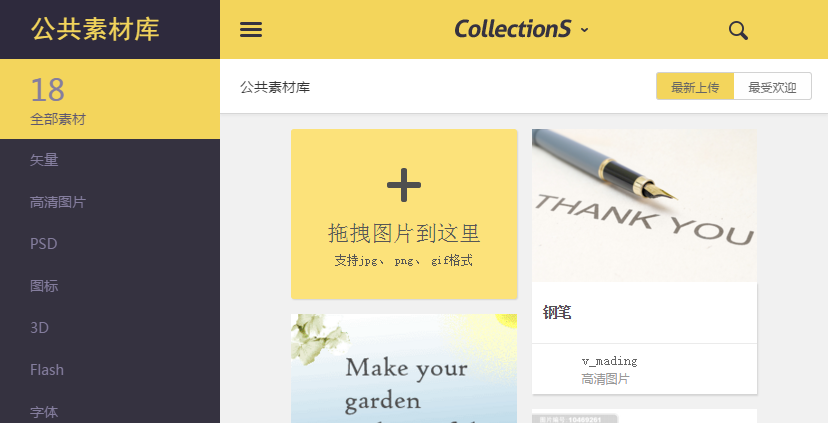


图 4‑4 上传图片资源界面展示（测试环境）

Fig 4‑4 Upload Image Interface (Test Environment)

1. 类型检测：对内容资源的文件类型进行检测，判断其是否是系统可接受的文件类型，如文件类型不能接受则向用户返回一个错误信息。需要注意的是，文件上传具有两个入口，即文档类型的入口和素材类型的入口，两者的检测内容是不一样的。
2. 生成id：在系统中，每个内容资源都需要具有独一无二的id，而无论其是何种类型。具有不重复的id便于系统对内容资源进行统一管理，而由于不同类型的内容资源会被存储在不同的数据库表中，如果使用自增id将会导致重复id的出现，因此使用一个id生成器保证id的独特性。Id生成的策略使用组合的方式，每个id共64位，混合文件类型，机器编码（由于使用云部署的方式，服务具有多台实例运行，需要对其加以区分，区分的方式可以是Mac地址等），时间戳，以及自增部分组成一个完整的id。



图 4‑5 文件上传一般流程

Fig 4‑5 Common Process of Upload

1. 进行预处理：在对文件进行存储之前，需要对文件进行一个预处理的过程。针对不同的文件类型，这个预处理的过程也是不同的。对文档类型的内容资源而言，需要将其转码为pdf格式，并生成封面的预览图；对于图片类型的内容资源而言，需要将其进行压缩。具体的过程将在文档服务模块与素材服务模块中详细讲解。
2. 内容文件存储：内容文件的存储由存储模块NFS实现。
3. 特征持久化：这里的特征指的是文件的名称、文件类型、上传者、存储位置等对内容资源进行表征的属性，这些属性需要被持久化在数据库中。不同的内容资源类型将被持久化在不同的表中。
4. 生成上传记录：对所有文件的上传过程，需要生成上传记录。商专记录主要用于对用户的行为进行分析。上传记录表的结构如下：

表 4‑2 上传记录表

Table 4‑2 Upload Record Table

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 说明 | 特征 |
| id | BIGINT(20) | 主键id | 非空，自增 |
| uploader | VARCHAR(64) | 上传人名称 | 非空 |
| source\_file\_path | VARCHAR(256) | 源文件存储地址 | 非空 |
| preview\_file\_path | VARCHAR(256) | 预览文件存储地址，并非所有文件都一定具有预览文件 | 非空，如果没有预览文件则存储空字符串 |
| upload\_time | DATETIME | 上传时间 | 非空，默认当前时间 |

内容资源下载：除了在线对内容资源进行浏览之外，用户还可以选择将内容资源下载至本地对其进行操作。内容资源下载的流程是，根据用户选择下载的文件定位文件的存储地址，然后通过NFS获取相应文件回传给用户。

* 1. 文档服务模块详细设计

本系统从内容的大分类上分成了文档库与素材库，二者在功能上有所区别。

文档库主要存储了文本类型的内容资源，包括pdf，txt，html等格式的内容资源文件。文本类型文件其内容以字符形式存储，用户直接可读。在本系统中，文本类型的文件主要是一些开发文档，说明等。用户需要对其进行在线编辑。本系统的文本类型文件的所属是基于用户群组的，即一个用户组内的复数数量用户都可以对文件进行创建、编辑与删除操作。用户群组内也有权限的划分。多个用户对文件进行编辑后会产生文件不同的版本，对不同版本的文件可以进行合并，回溯等操作。文本类型文件的检索主要是基于文件名称，或是标签、简介等文件特征。

素材库主要包含了图片类型，如jpg，bmp，png等格式的文件，以及流媒体文件，主要为flv格式。与文本类型的文件不同，这些文件不需要对其提供在线编辑操作，因此也不会产生版本的变迁，也就不需要提供版本操作。图片类型的文件，尤其是分辨率较高的图片文件，大小较大，不利于存储，因此需要对其进行压缩。同时，为了对图片进行格式化的展现，需要在上传时生成图片的缩略图。素材类型文件的检索除了文本文件相同的文件名、标签等检索方式之外，还提供了基于图片内容的检索，即“以图搜图”的检索方式。

文档服务模块包含两个子模块：文档处理模块，人员团队管理模块。在本系统中，由于文档的所属权是属于某个开发团队的，因此需要人员团队模块对文档处理进行辅助服务。

文档处理模块主要负责处理文档内容资源的特有需求，主要包含以下几方面的内容：文档内容上传时预处理，文档特征编辑，文档内容变在线编辑，文档版本控制。

在本系统中，文档是指一个文本类型的文件，文档具有名称、大小、描述、存储位置、版本等属性。文档将使用文件夹的形式进行组织，文件夹可以嵌套。基于文档的特征属性，文档存储表tb\_document结构如表4-3所示：

表 4‑3 文档表结构

Table 4‑3 Document Info Table

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 说明 | 特征 |
| id | BIGINT(20) | 主键id | 非空 |
| document\_id | BIGINT(20) | 文档id，具体代表了一个文档。同一文件的不同版本将具有相同的文档id | 非空 |
| version | INT(11) | 文档版本，随着文档的变更将会逐渐增长 | 非空，默认0 |
| father\_version | INT(11) | 文档的父版本，即此版本的文档是从哪个版本演变而来 | 非空 |
| file\_name | VARCHAR(255) | 文件名称，为可变更的属性，即用户编辑文档时可以更改文件名称 | 非空 |
| description | VARCHAR(255) | 文档描述，即文档的简介，可变更属性，即用户编辑文档时可修改文档的简介 | 非空，若描述为空，也需要存储为空字符 |
| file\_type | TINYINT(4) | 文档类型，枚举类型 | 非空 |
| file\_size | BIGINT(20) | 文档大小，单位是字节 | 非空 |
| 表 4-3文档表结构(续) | | | |
| 属性名 | 类型 | 说明 | 特征 |
| folder\_id | BIGINT(20) | 文档所属的文件夹的id | 非空 |
| source\_path | VARCHAR(255) | 文档的原文件的存储地址 | 非空 |
| pdf\_path | VARCHAR(255) | 文档转化的pdf文件的存储地址，所有文档上传后都需要转化为pdf格式，方便前端显示 | 非空 |
| like\_num | INT(11) | 文件的被赞次数 | 非空，默认0 |
| share\_num | INT(11) | 文件的被分享次数 | 非空，默认0 |
| add\_time | DATETIME | 文件的添加时间 | 非空，默认当前时间 |
| add\_user | INT(11) | 文档的添加人 | 非空 |
| convert\_status | TINYINT(4) | 文档的转化状态 | 非空 |
| update\_time | DATETIME | 文档的修改时间，文档被赞，被分享时此项均会更新 | 非空，默认当前时间 |

文档内容上传预处理：在文档资源进行上传的过程中，需要将其转化为pdf格式，pdf格式具有平台无关的特性，即无论用户在windows环境下还是linux环境下打开都可以得到相同的视图。由于转化过程可能会消耗一定的时间，因此做异步转换。在将文件正确地转化为pdf格式后，将会保存其第一页内容为图片格式作为文档的预览图。Pdf转换逻辑如图4-6所示：



图 4‑6 pdf转换逻辑

Fig 4‑6 PDF Conversion Process

文档内容在线编辑：对于文档类型的内容资源而言，本系统提供了在线编辑系统，即用户可以在线修改文档的内容，并保存更改。保存之后文档的内容变更将会实时的反映在系统中。对文档内容进行修改将会导致文档产生一个新的版本，系统中将总是显示文件的最新版本，即文档最后修改的版本。

用户编辑文档的过程基本可以分为两个部分：在线编辑与保存。

在线编辑：这个部分主要通过前端浏览器执行。用户在内容管理系统中查询到需要的文档资源之后，可以通过点击的方式查看文档详情，并开启编辑。编辑的过程中，前端浏览器将会缓存用户的操作，但并不会实时将其反映到系统后台，只有在用户执行了保存操作后，虽有更改才会返回后台并生效。

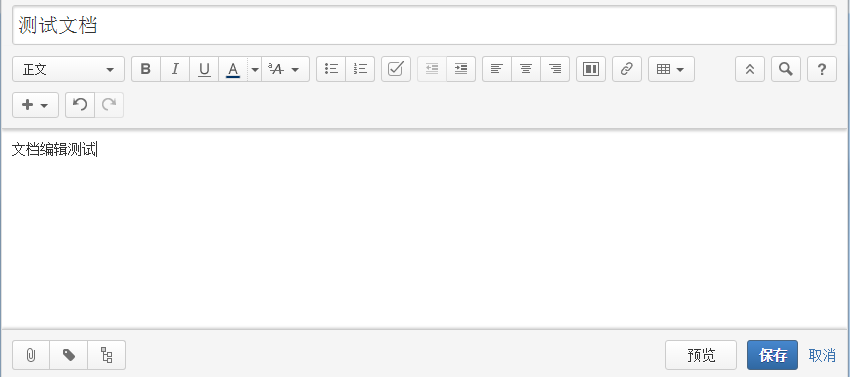


图 4‑7 在线编辑界面展示（测试环境）

Fig 4‑7 Document Edit Interface (Test Enviroment)

保存：用户对文档编辑完成后，可以进行保存，保存时将会提示用户是否确认。用户一旦确认保存则会对文档生成一个新的版本并保存这个版本。反之，则会放弃当前的修改。文档在线编辑流程如图4-8所示：



图 4‑8 文档在线编辑流程

Fig 4‑8 Process of Document Edit

版本控制：用户对文档内容进行编辑后，将会产生文件新的版本。对于同一份文件，同时可存在复数个不同的版本，需要对这些不同的版本进行管理。在本系统中，文档资源的版本控制主要包含了以下内容：

1. 查看版本树：用户指定某个具体的文件，可以查看此文件所有的历史版本信息。不同的版本之间具有承袭关系，除了原始版本外，每个版本都具有父版本，根据这一特点，可以形成一个树状结构，这就是文件的版本树。
2. 删除（回溯）：用户可以删除文件的某个版本，这一操作无法对版本树中的一个父节点执行。即用户只能删除版本树中的叶子节点。由于删除操作将会导致新的版本消失，而此新版本的一个父节点将会成为版本树中的一个叶子节点，因此删除操作相当于对版本进行回溯。
3. 对比：对于两个不同的版本，用户可以对比其差异性，文档的不同将会高亮显示。
4. 合并：对于文档的版本A，用户1和用户2同时进行编辑，即会产生文档的两个新版本B和C。这样文档的版本就形成了一个分支。用户可以选择两个版本，将其进行合并。合并操作涉及的两个版本有主次之分，合并操作是将次版本合并到主版本上。合并操作的结果是产生一个新的文档版本，此版本的父版本为合并操作中的主版本。
   1. 图像检索模块的实现

内容检索是内容管理系统的重要功能，内容管理系统往往会存储数量庞大的内容资源，如果仅支持对内容的罗列展示而没有实现检索功能，则会对用户的正常使用造成巨大的困难。因此，内容检索功能应该是内容管理系统的重要关注点之一。

* + 1. 图像检索的流程

在本系统中，对于图像内容资源，除了使用标签系统和文件特征（文件名，文件简介）进行检索之外，还可以根据图片本身的内容进行检索，即实现“以图搜图”的检索方式。图像检索的流程如图4-9所示：



图 4‑9 图片检索流程

Fig 4‑9 Process of Image Retrieval

1. 用户上传图像：为了进行基于内容给的图片检索，用户首先需要上传一份目标图片文件以进行检索。
2. 计算目标图像特征值：在收到用户上传的文件之后，需要计算此图像文件的特征值以用于检索。如第二章所述，本文所采用的表征图片的特征值为图像的差异哈希值，使用差异哈希算法以及pHash算法生成两个哈希值，比较时将与两者同时比较。



图 4‑10 搜索界面展示

**Fig 4‑10 Image Retrieval Interface**

1. 利用汉明距离进行相似图像检索：汉明距离，具体而言就是两个变量之中差异位的个数，二者差异个位个数越多，则汉明距离越远，反之越近。为了提高检索的效率，使用一张单独的表存储图像的哈希值。当目标图像与检索图像的汉明距离小于等于5时认为命中。
   * 1. 图像检索算法的具体实现

**缩小图片**

感知哈希算法的最终目标是为图片生成一个哈希值用以表征图像。对图片进行缩小操作首先可以降低哈希值的维度，也就减少了搜索的复杂度。同时，缩小图片是一个去除图片高频分量的过程，也就消除了图片的细节特征，只保留了图片的概貌特征，也就扩大了检索范围，即细节不同而整体结构大致相同的两张图片将具有极相似的哈希值，这也就是基于内容图片检索的基石。

对于使用像素表示的图像（非矢量图）而言，图像的放缩根本上讲是一种坐标变换，即将对于变换后图像（目标图像）中的每一个像素点，计算其来源（从原图像中获得），然后填充即可。这是一种逆向映射的思想。

在本论文中，图像缩小的目的是计算图像哈希值用以进行图像检索。无论原图的尺寸如何，在计算哈希值时都需要将图像缩小为固定尺寸。为了计算出的哈希值更为精确，图像的缩小应该尽量的平滑以提高质量。

插值法是一种简单的图像缩小算法，常用的插值法有最邻近插值法，双线性插值法，双三次插值法等。他们的计算难度依次上升，而缩小效果也是由弱至强。



图 4‑11图像缩小效果

**Fig 4‑11 Effect of Image Reduction**

如图，上图是将同一幅图像缩小为40\*40的结果。可以看到，使用最邻近插值法时图像具有明显的锯齿，而双线性插值法的缩小效果则平滑很多。结合缩放效果与计算代价总和考虑，本系统采用双线性插值法对图像进行缩小处理。

**转化灰度图像**

在对图像做DCT之前，需要对其进行量化处理。灰度就是一个很好的量化方法。

点阵图使用一个整数值来代表一个像素，整数是32位的。具体而言，高8为代表透明度，其后24位则代表图像的三个颜色通道值，即RGB值。RGB是一种简单方便的图像表示方式，所谓RGB，即指red，green，blue色光三原色。使用这三种颜色可以表示各种不同的颜色。而图像的RGB表示就是使用了这种方式，通过给三个颜色分量赋予不同的数值来表示像素点的真实颜色。

使用图像的RGB表示很难直接计算图像的差异，这就需要将图像转化为灰度图像再进行比较。所谓灰度图像，就是将图像的颜色差异屏蔽，只保留图像的亮度信息。从视觉上看，就是一幅黑白图像。在RGB表示下，三个通道具有相同的数值，这就将24位的颜色信息降低到了8位的灰度信息。

将RGB图像转化为灰度图像的具体公式如下：

|  |  |  |
| --- | --- | --- |
|  |  | ( 4‑1 ) |

由于浮点数运算与除法运算速度远低于整数运算和移位运算，因此在具体的计算时，可以对此公式进行一定的优化。可以将系数转化为整数，然后通过移位代替除法来进行计算。在使用16位精度的情况下，可以得到系数分别为：

|  |  |  |
| --- | --- | --- |
|  |  | ( 4‑2 ) |
|  |  | ( 4‑3 ) |
|  |  | ( 4‑4 ) |

于是可以得到如下公式：

|  |  |  |
| --- | --- | --- |
|  | Gray = (R\*19595 + G\*38469 + B\*7472) >> 16 | ( 4‑5 ) |

将图像转化为灰度图像后就可以计算差异了。

**生成哈希值**

使用离散余弦变换生成图像的哈希值。这一步在第二章进行了详细的介绍。

**根据汉明距离进行检索**

计算出图片的哈希值以后，就可以根据汉明距离在数据库中进行检索。目前使用两种方式同时检索，将结果整合后展现给用户。展现的顺序为汉明距离的大小，检索图与目标图像的哈希值的汉明距离越小，则展现的优先级越高，反之越低。汉明距离有2个界限，5与10，汉明距离小于5的会被优先展示。当用户需要展现更多的时候，再展示汉明距离在5到10之间的图像。汉明距离超过10的认为是不相关图像，不予展示。

为了提高检索的效率，本系统使用了一张单独的表对图像的哈希值进行存储，图像的其他信息将在另外的表中存储。用于图像检索的表结构如表4-4所示：

表 4‑4 图片哈希表结构

Table 4‑4 Image Hash Table

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 说明 |
| id | BIGINT(20) | 主键id |
| name | VARCHAR(256) | 图片名称 |
| d\_hash | BIGINT(20) | 图片d哈希 |
| p\_hash | BIGINT(20) | 图片p哈希 |
| add\_time | DATETIME | 图片的添加时间 |

**图片上传的预处理工作**

对于图片类型的资源，需要在上传时对其进行一定的预处理。具体而言，在上传时，需要对用户上传的图片资源计算其差异哈希值并进行检索，为用户展现相似的图片，询问用户系统中是否已经存在用户需要上传的图片，用户确认后可选择继续上传或取消上传。这么做的原因是防止重复的图片资源被上传到系统之中，占用系统资源并对检索造成一定干扰。同时，在图片上传时，会将计算出的hash值直接存储于表tb\_material\_hash中。图片上传的具体流程如图4-12所示：



图 4‑12 图片上传流程图

**Fig 4‑12 Image Upload Process**

* 1. 分类管理模块的设计实现

为了使用户检索内容资源变得更加方便，本系统设置了分类功能，用户可以对系统内的内容资源进行分门别类的管理。同时用户可以对分类进行管理，执行包括新建，编辑，删除等一系列操作。

在本系统中，“分类”实质上是一个过滤操作，一个分类相当于一个过滤器，将符合此分类要求的内容资源过滤出来单独进行展示。因此分类与内容资源的关系是多对多的，即同一内容资源可以同时属于不同的分类，同一个分类也能够包含不同的内容资源。

通过基于特征的内容检索也可以实现等同于分类（过滤）的效果，之所以设置分类，是为了方便用户的操作。分类是与用户相关的，即不同的用户可以拥有不同的分类，完全根据用户自己的喜好与需求进行设置。本系统中用于分类的特征有以下几种：

1. 文件名：可以使用文件名对内容资源进行分类，只要文件名符合某种给定的规范的内容资源就属于这个分类。
2. 文件类型：只要是给定的文件类型就属于此分类。
3. 上传者：根据文件的上传者进行分类。

对同一个分类而言，三种要素可以并存，相互之间是与的关系。根据上述对分类进行的分析，可以设计出分类的信息表tb\_type\_info如表4-5所示：

表 4‑5 分类信息表结构

Table 4‑5 Type Info Table

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 说明 |
| id | BIGINT(20) | 主键id |
| name | VARCHAR(64) | 此分类的名称 |
| user\_name | VARCHAR(64) | 此分类所属的用户 |
| add\_time | DATETIME | 分类的添加时间 |
| update\_time | DATETIME | 分类的修改时间 |

除了分类基本信息，在每个分类下还有详细信息，即此分类的具体分类依据。其中包括了分类的依据类型，分类的依据内容，添加时间等等。将分类基本信息与分类的详细信息分开存储的目的是，在修改分类时提供一定的便捷。分类详细信息表tb\_type\_detail\_info结构如下：

表 4‑6 分类详细信息表结构

Table 4‑6 Type Detail Table

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 说明 |
| id | BIGINT(20) | 主键id |
| type\_id | BIGINT(20) | 此分类详细信息所属的分类的id |
| detail\_type | TINYINT(4) | 此分类详细信息的类型，包含文件名，文件类型，上传者 |
| detail\_info | VARCHAR(256) | 此分类详细信息的具体内容，与detail\_type功能起作用 |
| add\_time | DATETIME | 此分类详细信息的添加时间 |

**创建分类**

用户可以创建新的分类。创建分类后用户即可使用此分类的对文件进行筛选操作。用户创建的分类只有自己可见，对其他用户无影响。创建分类的具体流程如下：

1. 创建分类：此时用户只需输入分类名，系统会进行检查，如果没有重复，则分类创建成功，此时的分类没有任何具体信息。
2. 添加具体信息：分类创建成功后，用户可以逐条添加分类的具体信息。首先选择具体信息类型，然后进行输入。用户可以通过点击完成来结束这一操作。

**编辑分类**

用户可以对已经创建好的分类进行编辑。分类的名称不允许修改，可编辑的内容是分类的详细信息。用户可以对此分类下的详细信息进行添加，删除，编辑等操作。

**删除分类**

用户可以删除已有的分类，分类被删除后，分类的详细信息将一并被删除。这将会直接反应在用户的界面上。

* 1. 本章小结

本章为内容管理系统的实现。根据第三章的概要设计，对其中几个重点模块的设计实现进行了简要的介绍。主要选取了身份认证，内容资源上传下载，文档服务，图片检索和分类管理5个部分进行介绍。其中图片内容的检索是本章的重点，除了对流程进行设计之外，还对具体的算法给出了实现。

1. 系统测试

# 

系统测是软件工程实践中的一个重要组成部分，应用在投入正式的生产环境进行使用之前，必须要经过严格的质量控制流程。本系统采用敏捷的开发模式，以3周为一个开发周期进行迭代开发，每个迭代开发或完善一部分系统功能。每个迭代都要经过开发-验证的周期，保证迭代开发功能的可用性。在迭代内的测试主要针对本期迭代开发的功能展开，具有收敛性。当有对系统改动较大的功能更新时，则需要进行回归测试，根据改动的大小，有时甚至需要对整个系统进行回归。因此可说测试贯穿了本系统开发始终。

本系统的测试包含了自动测试与手工测试两个方面内容。自动测试由开发人员自主进行，为白盒测试。开发人员在明了代码逻辑的基础上编写测试用例，保证系统各功能模块逻辑的正确性。手工测试部分则由专门的QA（质量保证）人员进行。与自动测试部分相对，手工测试为黑盒测试。

1. 1. 自动测试

本系统的自动测试使用JUnit框架，其是一个基于Java语言开发的单元测试框架。Junit提供了多种手段减轻了测试代码开发的工作量，并且易用性较高。

本系统自动测试验收指标主要有两个，即正确率与覆盖率。正确率即测试用例的通过率。在Junit中，一个测试用例的运行结果有三种状态：pass fail 和 error。本系统要求每个迭代上线时单元测试通过率为100%。

覆盖率表征了测试用例的完备程度。有行覆盖率，分支覆盖率等。本系统主要关注用例的分支覆盖率，并要求系统上线时自动测试用例的分支覆盖率达到60%以上。



图 5‑1 Junit单元测试实例

**Fig 5‑1 Junit Test Case**

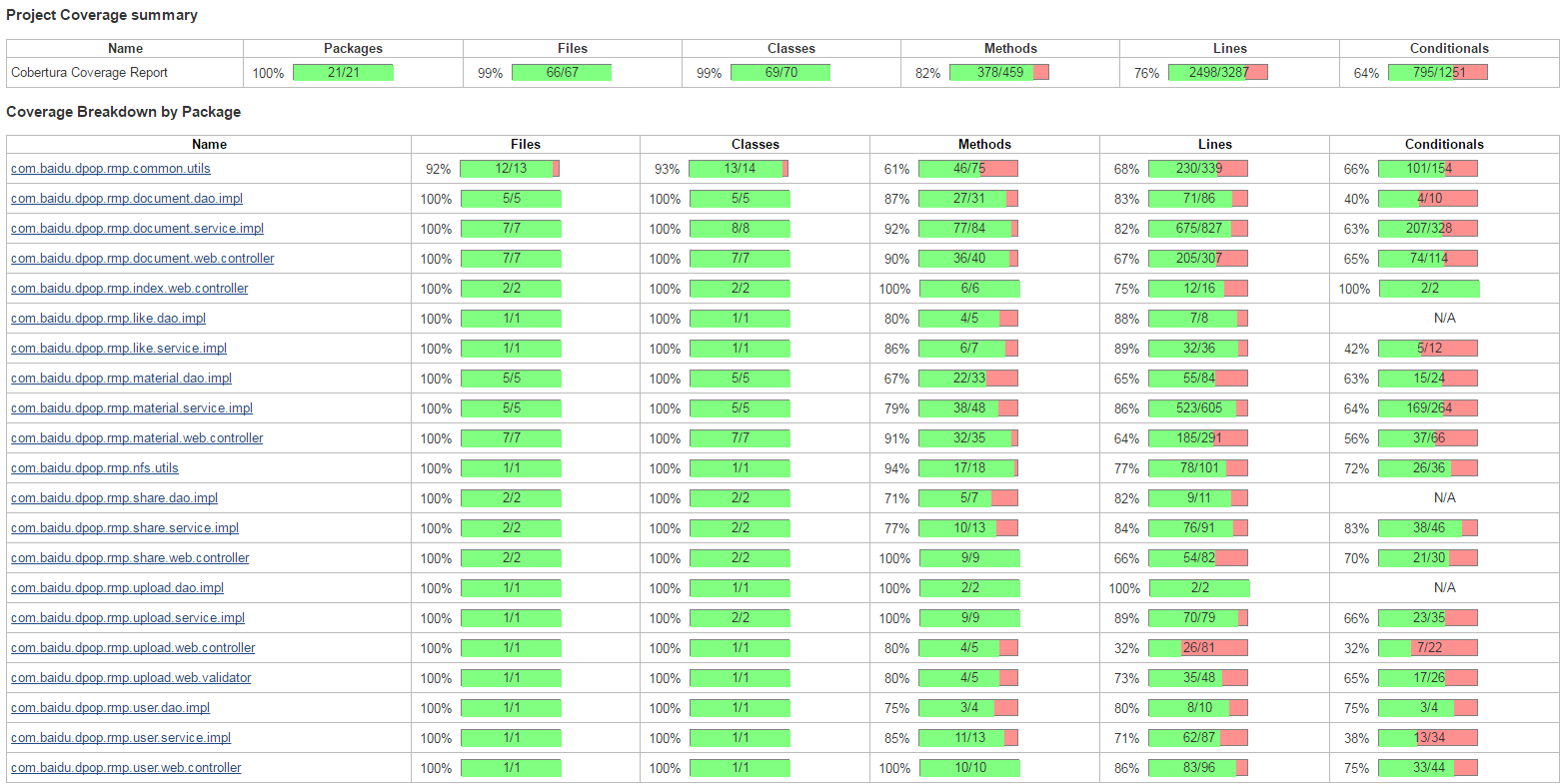


图 5‑2 自动测试分支覆盖率

**Fig 5‑2 Branch Coverage**

可以看到，本系统单元测试分支总体覆盖率最终达到64%，符合预定的要求。

* 1. 人工测试

自动测试是白盒测试，由开发人员自主进行，但其仅仅能作为系统开发质量的第一重保障。开发人员针对自己编写的代码进行测试时，思路往往会受到限制。同时单元测试缺乏模块间的联系，也不能很好的保证一个完整的功能执行是否正确。因此还需要对系统功能进行人工测试。

手工测试的测试人员预先不知道代码的逻辑，仅仅从用户的角度对系统的各个功能点进行测试。在保证正常的操作流程无错误的前提下，测试人员还会针对各种极端情况，便捷情况，或用户输入错误，操作错误的情况进行测试，保证系统在这些情形下仍可以正确运行。后文将详细介绍部分系统重要功能模块的测试用例。

用户登录部分的测试用例如表5-1所示：

表 5‑1用户登录测试用例

Table 5‑1 Test Cases of Login

|  |  |  |  |
| --- | --- | --- | --- |
| 业务流程 | 说明 | 预期结果 | 测试结果 |
| 用户登录 | 未登录系统的用户初次访问系统主页 | 自动跳转至登录系统页面，确认身份信息正确后成功登录跳转系统主页 | 执行成功，达到预期 |
| 错误登录 | 未登录系统的用户访问系统主页 | 同上，用户在登录页面输入非法的身份信息，提示身份认证失败 | 执行成功，达到预期 |
| 用户退出 | 已经登录用户退出 | 用户点击退出，执行成功后跳转至登录系统页面 | 执行成功，达到预期 |

这一部分的测试用例主要对用户的登录退出功能进行验证，保证系统可识别具有登录权限的用户。

文件上传下载部分的测试用例如表5-2所示：

表 5‑2 上传下载测试用例

Table 5‑2 Test Cases of Upload and Download

|  |  |  |  |
| --- | --- | --- | --- |
| 业务流程 | 说明 | 预期结果 | 测试结果 |
| 文档上传 | 用户上传文档类型的资源 | 正确上传，资源出现在系统中，可正常查看 | 执行成功，达到预期 |
| 图片上传 | 用户上传图片类型的资源 | 上传时自动进行检索，如有相似结果则询问用户是否有想要的结果。 | 执行成功，达到预期 |
| 表 5-2 上传下载测试用例（续） | | | |
| 业务流程 | 说明 | 预期结果 | 测试结果 |
| 文件下载 | 用户下载系统中的资源文件 | 用户针对某个文件资源点击下载，文件通过浏览器正确下载 | 执行成功，达到预期 |

这一部分的测试用例针对文件上传下载模块，保证上传下载过程顺利执行。图片与文档的上传流程有所不同，中间有一步检索的过程。

文档相关的测试用例如表5-3所示：

表 5‑3 文档相关测试用例

Table 5‑3 Test Cases of Document Operation

|  |  |  |  |
| --- | --- | --- | --- |
| 业务流程 | 说明 | 预期结果 | 测试结果 |
| 查看文档 | 用户查看文档的内容 | 文件能正确打开，用户可通过浏览器查看 | 执行成功，达到预期 |
| 文档编辑 | 用户在线编辑文档 | 用户通过浏览器对文档在线编辑，完成后保存成为新的版本 | 执行成功，达到预期 |
| 版本树查看 | 用户针对文档查看编辑版本树 | 以动态图的形式展现文档的版本变迁，每个节点具有编辑时间与编辑者信息 | 执行成功，达到预期 |
| 删除版本 | 用户删除文档的某个版本 | 用户只能删除无子版本的节点，同时此节点应不处于被编辑的状态 | 执行成功，达到预期 |
| 合并版本 | 用户合并两个不同的版本 | 合并之后产生一个新的版本，其中内容是二者的合并结果。如果有冲突内容则操作失败。 | 执行成功，达到预期 |

这一部分测试用例针对文档类型资源的一些独有操作。在此阶段的系统中，文档的合并功能还不完善，出现冲突时将会导致操作失败。

分类相关的测试用例如表5-4所示：

表 5‑4分类相关测试用例

Table 5‑4Test Cases of Classification

|  |  |  |  |
| --- | --- | --- | --- |
| 业务流程 | 说明 | 预期结果 | 测试结果 |
| 新建分类 | 用户创建一个新的分类 | 创建分类时填名称，选择分类规则。如名称重复则操作失败，否则成功。创建成功后出现新的分类，同时分类规则可正确生效。 | 执行成功，达到预期 |
| 编辑分类 | 用户编辑已有的分类信息，修改名称或规则 | 名称重复则修改失败，否则成功。规则修改后立即生效 | 执行成功，达到预期 |
| 删除分类 | 用户删除一个分类 | 分类被移除 | 执行成功，达到预期 |

这一部分测试用例主要针对分类的相关操作，保证功能的正确性。

以上给出了系统的几个关键模块的人工测试用例，每个用例下还有若干子用例，这里没有一一列出。

* 1. 图片检索测试

图片检索是本系统的重要功能之一，需要经过严格地验证。本节给出一个简单的测试过程。

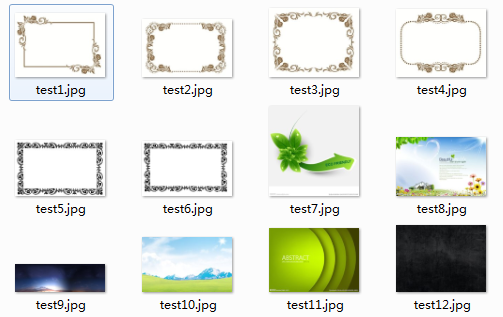


图 5‑3 检索测试图片

**Fig 5‑3 Test Images**

如图，本次测试选取了12张图片作为备选图片。问了取得更直观的结果，测试将直接调用系统模块计算汉明距离得出结果，跳过前端的处理。

表 5‑5 汉明距离测试结果

Table 5‑5 Haming Distance Result

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | 0 | 20 | 19 | 35 | 37 | 28 | 32 | 35 | 32 | 32 | 28 | 32 |
| 2 | 20 | 0 | 15 | 33 | 37 | 30 | 36 | 39 | 28 | 38 | 30 | 28 |
| 3 | 19 | 15 | 0 | 28 | 36 | 27 | 37 | 34 | 27 | 29 | 31 | 29 |
| 4 | 35 | 33 | 28 | 0 | 34 | 35 | 35 | 34 | 27 | 31 | 35 | 33 |
| 5 | 37 | 37 | 36 | 34 | 0 | 26 | 33 | 34 | 33 | 31 | 31 | 33 |
| 6 | 28 | 30 | 27 | 35 | 26 | 0 | 32 | 29 | 36 | 28 | 34 | 34 |
| 7 | 32 | 36 | 37 | 35 | 33 | 32 | 0 | 28 | 27 | 30 | 32 | 30 |
| 8 | 35 | 39 | 34 | 34 | 34 | 29 | 28 | 0 | 37 | 29 | 35 | 39 |
| 9 | 32 | 28 | 27 | 27 | 33 | 36 | 27 | 0 | \_\_ | 34 | 32 | 30 |
| 10 | 32 | 38 | 29 | 31 | 31 | 28 | 30 | 29 | 34 | 0 | 28 | 36 |
| 11 | 28 | 30 | 31 | 35 | 31 | 34 | 32 | 35 | 32 | 28 | 0 | 40 |
| 12 | 32 | 28 | 29 | 33 | 33 | 34 | 30 | 30 | 30 | 36 | 40 | 0 |

本系统选择的阈值是26，即汉明距离小于等于26时即认为两张图片为相似图片。可以看出，对于较为接近的图1，图2，图3以及图5与图6，系统可以正确的识别出。而图4则差别较大。图9则相当容易被误识别为相似。

总体而言，系统已经具有了初步的识别功能，但识别效果仍有改进的空间。

* 1. 本章小结

本章主要对系统测试的情况进行了介绍。第一节主要介绍了本系统自动测试的方法以及结果。第二节针对系统的几个关键模块给出了测试用例以及测试结果。第三节则对系统的图像检索功能进行了简单的测试，给出了具体的实验数字。

1. 总结与展望
2. 1. 总结

对互联网企业而言，在自身业务的增长过程中，往往会产生大量的内容资源，如开发文档，开发的图片素材等。内容管理系统可以让企业更好的利用这些内容资源。基于内容的图片搜索可以让使用者用更灵活的方式检索系统中的图片类内容资源，提高系统的易用性。

本文旨在设计实现出一个基于CBIR的内容管理系统，首先结合企业实际使用的情况，对系统的功能性需求，非功能性需求进行分析；在此基础从系统架构，功能模块划分的角度介绍了系统的总体设计；其后阐述了系统的几个重点功能模块的实现；最后对系统的测试情况进行了简单介绍。系统目前已投入实际使用。

本文的创新点是将基于内容的图片检索(CBIR)技术应用于内容管理平台之上，这对于一些侧重于图片类内容资源管理的系统而言有着很强的实用意义。

* 1. 未来的工作

由于项目本身可获取的资源限制，系统只是实现了内容管理的一些初级功能，图片检索的效果也还没有达到最理想的情况。在未来的工作中，本系统还有以下可改进的方向：

1. 提高图片检索的效果：目前系统使用的感知哈希方式计算速度快，可以完成基础的图像检索，但对于图形旋转等不具有鲁棒性。企业内容管理系统的特点是，内容资源数量不会过多（万到十万级别），但检索灵活性要求较高。在未来的改进中可以使用其他更复杂的算法，牺牲一定的效率而提高检索能力，例如可以让用户使用简笔画的形式进行检索。
2. 进行更完善的测试：系统上线一段时间后，仍会偶现一些问题。单元测试的条件覆盖率仅仅达标，还有提高的余地。图片检索的测试也仅仅停留在检索效果的阶段，缺乏压力测试。

参考文献

**[1]** Dukart J R. Content repositories: One or many?[J]. E-doc, 2004(3).

**[2]** 石雪松. 内容管理的真正内涵[J]. 中国经济和信息化, 2003(4):24-24.

**[3]** Winters J. What Is a Content Management System?[J]. Aall Spectrum, 2009.

**[4]** 陈亚南. 网站内容管理系统的设计与实现[D]. 山东大学, 2007.

**[5]** IBM Content Manager. IBM - Content Manager[J]. Ibm Corporation.

**[6]** Managing Enterprise Content: A Unified Content Strategy. Ann Rockley, Pamela Kostur, Steve Manning. New Riders, 2003

**[7]** The content management handbook. Martin White. Facet Publishing, 2005.

**[8]** Content Management Bible, Bob Boiko. John Wiley & Sons, 2005.

**[9]** 孔佳, 李昀. 内容管理系统的产生与发展[J]. 农业网络信息, 2008(3).

**[10]** 陈洪波. 基于门户技术的内容管理系统研究[D]. 中国科学院计算技术研究所, 2006.

**[11]** 夏德深. 现代图像处理技术与应用[M]. 东南大学出版社, 1997.

**[12]** 付海燕. 基于图像哈希的大规模图像检索方法研究[D]. 大连理工大学, 2014.

高金金. 基于双线性映射与哈希方法的图像检索研究[D]. 中北大学, 2014.

**[13]** Bensaali F, Amira A. Design and Efficient FPGA Implementation of an RGB to YCrCb Color Space Converter Using Distributed Arithmetic[J]. Lecture Notes in Computer Science, 2004, 3203:991-995.

**[14]** 王向阳, 杨红颖, 郑宏亮,等. 基于视觉权值的分块颜色直方图图像检索算法[J]. 自动化学报, 2010, 36(10):1489-1492.

**[15]** 杨红菊, 张艳, 曹付元. 一种基于颜色矩和多尺度纹理特征的彩色图像检索方法[J]. 计算机科学, 2009, 36(09):274-277.

**[16]** 钟智彦, 文志强, 叶德刚. 基于改进协方差矩阵的半色调图像分类研究[J]. 微型机与应用, 2015(19):54-57.

**[17]** Ojala T, Pietik&#, Inen M, et al. Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2002, 24(7):971-987.

**[18]** 刘梅, 刘伟东, 许荣庆,等. 基于小波变换及傅里叶描述子的图像检索[J]. 系统工程与电子技术, 2003, 25(8):1000-1002.

**[19]** Object Recognition from Local Scale Invariant Features[C]// Proceedings of the International Conference on Computer Vision, Greece. 1999.

**[20]** 牛夏牧, 焦玉华. 感知哈希综述[J]. 电子学报, 2008, 36(7):1405-1411.

**[21]** 袁梅冷, 黄烟波, 黄家林,等. J2EE应用模型中MVC软件体系结构的研究与应用[J]. 计算机应用研究, 2003, 20(3):147-149.

**[22]** Johnson R, Hoeller J, Arendsen A, et al. Professional Java Development with the Spring Framework[J]. Apc, 2007:195-237.

**[23]** 薄奇, 许林英. Spring框架中IoC的实现[J]. 微处理机, 2008, 29(1):147-149.