中山大学

硕士学位论文

# 垂直搜索引擎的设计开发

姓名：刘琦

申请学位级别：硕士

专业：软件工程

指导教师：李中华

20100602

论文题目：垂直搜索引擎的设计开发
专　　业：软件工程
硕 士 生：刘琦
指导教师：李中华

# 摘 要

随着计算机技术和互联网技术的快速发展，网络上的信息量也急剧增加。面对如此巨大的信息，搜索引擎的出现在一定程度上给人们带来了方便。通过搜索引擎，人们可以快速地访问一些自己不知道的网页，从中获取到很多有用的信息。然而，很多搜索引擎在搜索信息时往往会产生一些问题，如搜索到的页面过多、网页重复、网页数据更新慢、符合搜索要求的网页比较少等。而垂直搜索技术的产生很好的解决了这些问题，它为人们提供的不再是成千上万的相关网页信息，而是范围很小，且极具针对性的具体信息。

本论文以手机资源为背景，通过运用扩展 Heritrix 和 Lucene，初步构建出了一个检索结果比较精准的垂直搜索引擎。本文主要的研究工作分有以下四个部分：(1)通过定制和扩展 Heritrix 这个由 Java 开发的、开源的 Web 网络爬虫来从互联网上爬取相关的信息资源；(2)利用 HtmlParser 工具对爬取的信息进行分析和抽取；(3)深入研究 Lucene 以及及其相关的技术，将 Lucene 成功地运用到了系统当中，从而为系统提供更好的全文索引/检索服务；(4)设计了 MVC 的查询接口。

关键词：垂直搜索引擎、本体、Heritrix、Lucene、信息抽取、MVC

Title: Design and development of vertical search engines

Major: Software Engineering

Name: Qi Liu

Supervisor: Zhonghua Li

# Abstract

With the rapid development of computer and Internet technology, the information increases dramatically. According with such huge information, the emergence of search engine brings convenience to people in certain degree. People can access to the Web pages quickly that they don't know and obtain a lot of useful information by search engine. However, lots of search engines often cause some problems. For example, many unwanted pages and the repetition of pages will be searched. It takes a lot of time to update the data and there are few pages that are required. But the appearance of Vertical Search technology figures out these problems, which no longer provides thousands of related information on pages but limits the range and focuses on specific information.

This paper discusses about cell phone resource. It initially comes up with a vertical search with fairly precise outcome through expanding the use of Heritrix and Lucene. The major research work of this paper is divided into four parts. Firstly, by customizing and extending the Heritrix, which is a WebCrawler and developed in Java to crawl some information from Internet. Secondly, I'll have the information that we crawled analyzed and cramp out some of that with the tool of HtmlParser. Thirdly, I'll have a deep research about Lucene and the related technology to apply to the system successfully, thus provide a better full-text index or retrieval service for the system. Finally, the system designs a MVC connector.

**Key Words:** Vertical Search Engine, Neumann, Heritrix, Lucene, Information Extraction, MVC

# 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。对本文的研究作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：刘培涛

日　　期：2010.5.22

# 学位论文使用授权声明

本人完全了解中山大学有关保留、使用学位论文的规定，即：学校有权保留学位论文并向国家主管部门或其指定机构送交论文的电子版和纸质版，有权将学位论文用于非赢利目的的少量复制并允许论文进入学校图书馆、院系资料室被查阅，有权将学位论文的内容编入有关数据库进行检索，可以采用复印、缩印或其他方法保存学位论文。

学位论文作者签名：刘培涛　　导师签名：

日期：2010年 5月22日　　日期：2010年 5月22日

# 第 1 章 绪 论

## 1.1 选题背景

搜索引擎的产生是为了让用户能够方便的从网络空间获得其需要的信息。它根据一定的策略、运用特定的计算机程序搜集互联网上的信息，在对信息进行组织和处理后，并将处理后的信息显示给用户，是为用户提供检索服务的系统。但是随着网络的快速发展，各个领域内信息的不断增加，通用搜索引擎已经不能满足用户的一些需求，特别是当用户需要的是搜索某一领域内的信息时。垂直搜索引擎就正好符合了这种需求，它是针对某一个领域的专业搜索引擎，而不是将所有的信息呈现给用户，它将跟该领域相关的网页下载之后，从中提取出用户需要的信息，经过进一步的处理之后在呈现给用户。目前，一些企业的主页中都需要提供对应的搜索引擎，一般都是通过检索数据库直接获取数据，当数据量比较小的时候，响应速度还可以满足用户的需求，可是当数据量较大时，数据库的性能和速度便无法满足要求。为了解决这一问题，很多企业通过购买百度等公司的搜索服务，这种做法不仅需要高昂的成本，还会使得企业的信息造成泄露。Heritrix和Lucene是一个完全由Jave开发，并且开源的软件，它们分别提供了一个网络蜘蛛的框架和全文检索系统的框架，可以方便快速的搭建一个检索系统。因此，通过Heritrix和Lucene构建一个全文检索系统，无疑为中小型企业减少开销提供了极大的便利。

## 1.2 选题意义

随着计算机技术和互联网技术的快速发展，网络上的信息量也急剧增加。面对如此巨大的信息，搜索引擎的出现在一定程度上给人们带来了方便。通过搜索引擎，人们可以快速地访问一些自己不知道的网页，从中获取到很多有用的信息。然而，很多搜索引擎在搜索信息时往往会产生一些问题，如搜索到的页面过多、

网页重复、网页数据更新慢、符合搜索要求的网页比较少等。而垂直搜索技术的产生很好的解决了这些问题，它不是提供跟关键词相关的所有信息，而是提供某一领域内跟该关键词有关的信息。

垂直搜索引擎的开发主要分四个部分完成，第一部分是对领域内的信息进行抓取，通过爬虫将对应领域内的信息抓取下来；第二部分是对领域内的信息进行抽取，由于下载下来的信息量大，所以需要抽取出用户需要的信息以便呈现给用户；第三部分就是对抽取出的信息建立索引，主要是对信息进行违法字过滤，分词等；第四部分是建立检索模块，主要是通过用户输入的关键字进行查询，并将查询的结果以一定结构呈现给用户。本文就是通过 Lucene 和 Heritirx 框架来开发一个手机垂直搜索引擎。

## 1.3 国内外发展现状

在国外，垂直搜索引擎的研究正在蓬勃的发展，并且成为了研究的热点，许多专门领域都有自己的垂直搜索引擎。以下就是国外具有代表性的垂直检索系统。

(1)Berkeley 的 Focused Project[1]

该系统中的蜘蛛爬行器主要有两个控制器：一个是主题分类器 Classifier，该分类器分析计算下载的目标与对应主题的相关度；另一个程序是净化器 Distiner，该净化器用来确定哪些 URL 链接指向相关领域的页面。

(2)NEC 研究院的 CiteSeer[2]

Citeseer 的主要功能是对论文领域进行检索。Citeseer 的核心是 ACI(Automatically Citation Index)，它可以自动地对网上的电子文件(Postscript 和 PDF 等格式)进行索引并分类。

(3)美国国家科学数字图书馆的 Collection Building Program(CBP)[2]

这个系统主要是构建一个图书馆检索系统，同时研究是否可能使某一主题上的资源进行自动建设。以下三点是 CBP 最为显著的优点：第一，在 CBP 的整个项目构建过程中，由于其信息采集主要是面向教育、教学，所以

在整个领域的准备性方面要比全面性方面更为重要；第二，整个项目不提供所有的内容，只是提供相关内容的 URL 链接；第三，整个项目只需要用户输入极少的查询关键词，就可以返回用户感兴趣的相关的 URL 链接。

在国内，垂直搜索引擎正处于一个蓬勃发展的时期，各种专业搜索引擎层出不穷，许多专门领域都有自己的垂直搜索引擎。赛迪 IT 罗盘是中文领域首个真正意义上的垂直搜索引擎，其它垂直搜索引擎主要有以奇虎(http://www.qihoo.com)为代表的论坛搜索；以酷讯(http://www.kooxoo.com)为代表的生活搜索；以去哪儿(http://www.qunar.com)为代表的旅游搜索；以搜职(http://www.globehr.com)为代表的招聘搜索等等。

## 1.4 论文的主要工作

本文就 Lucene 和 Heritrix 以及相关的技术进行了深入的研究和探讨，并在此基础上构建了具有检索功能的垂直搜索引擎，主要工作包括以下几方面。

(1)对垂直搜索引擎的总体架构进行分析，同时介绍其整个的工作流程。

(2)在信息采集模块中分析垂直搜索系统通用的采集策略，并指出其缺点。然后通过对本体的介绍提出了解决该缺点的方法，最后实现了基于本体知识库的信息采集策略。

(3)通过对 Heritrix 网络爬虫框架进行研究学习，尤其针对其特有的可扩展性进行了详细的探讨，并将对应的采集策略扩展进 Heritrix 中。

(4)本文对 Lucene 索引框架进行了研究学习，分析了其主要架构和各个部件的功能，然后将之应用于检索模块当中。

(5)本文在理论分析的基础上，针对实际应用，通过运用 Lucene 检索框架和扩展 Heritrix 爬虫框架，构建了一个手机垂直搜索引擎。并详细论述了整个系统的设计思想以及系统的主要模块，

(6)设计了基于 MVC 架构的查询接口，实现原型系统的检索功能。

## 1.5 论文的章节安排

本文主要分为以下几个部分：

第一章为绪论部分，主要介绍垂直检索系统的研究意义，研究背景，主要工作，章节安排等。

第二章为垂直搜索引擎的系统结构与信息采集部分，分析了目前通用的信息采集策略的核心思想及该思想的不足，同时提出了解决该不足的方法，即构建基于本体知识库的信息采集策略。

第三章为关键技术分析与研究，主要是对 Heritrix 和 Lucene 的相关分析，详细分析了 Heritrix 和 Lucene 的原理、主要模块等。

第四章为面向手机产品信息的垂直搜索引擎的设计与实现，利用已分析的技术尝试实现一个面向手机产品信息的垂直搜索引擎的原型系统。

第五章为通过分析整个系统的创新与不足，为继续完善系统做准备工作。

# 第 2 章　搜索引擎的系统结构与信息采集

## 2.1 垂直搜索引擎的系统结构

通常，垂直搜索引擎主要由信息抓取模块、信息抽取模块、索引模块、检索模块四大模块组成，系统结构图如图 2.1 所示。

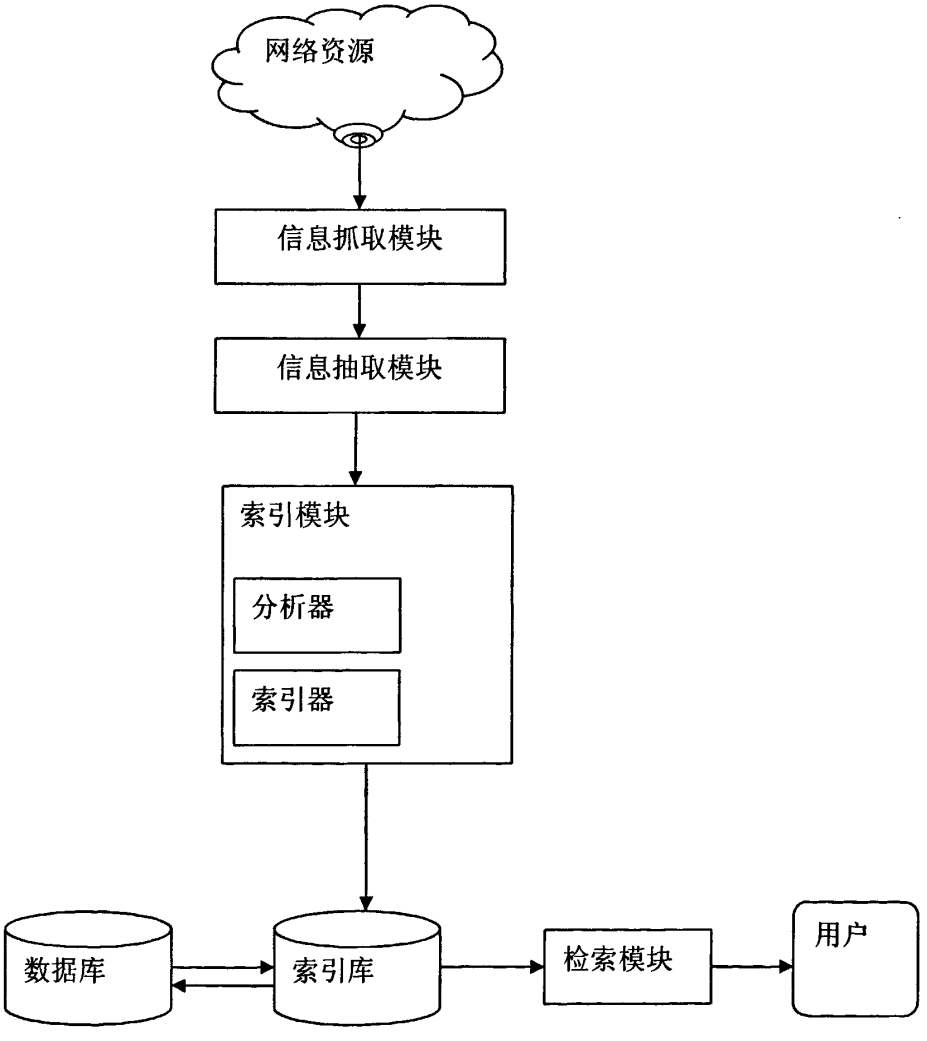


图 2.1 垂直搜索引擎的系统结构图

(1)信息抓取模块

信息抓取模块主要是对网络上的信息进行抓取，这一过程是通过网络爬虫来实现的，网络爬虫通过分析网页URL，并自动采集网页上的相关内容，然后在采集的过程中通过原始分析的网页URL来发现新的URL，之后再对新的URL内容进行采集，一直不断重复这一过程直到不会产生新的URL为止，而在这一采集过程当中，垂直搜索引擎需要进行相应的优化，才能使得整个采集更有效率。

(2)信息抽取模块

信息抽取模块主要是对下载下来的网页URL的信息进行分析和抽取，提取出用户需要的信息，并将其形成结构化的信息进行存储，以方便之后呈现给用户。

(3)索引模块

索引模块主要由两个子模块组成：一个是分析器，分析器的作用是对结构化的文本信息进行分析处理，如过滤违法字，特殊字符等，然后在对该文本信息进行分词，以便建立索引。另一个是索引器。索引器的作用就是用来为已经分析处理完的信息建立索引。当信息量巨大的时候，通过索引来进行检索能提高系统的性能。

(4)检索模块

这一模块是系统与用户交互的模块，系统对用户输入的查询关键字进行分析处理，然后通过索引数据库搜索出相关的数据，过滤掉不相关的数据，然后对结果按照一定的算法排序，并最终呈现给用户。

## 2.2 常用的信息采集策略

在整个垂直检索系统的四大模块中，信息抓取模块是其中的基础和核心，在通用搜索引擎的抓取模块中，由于是要满足所有用户的要求，所以通用搜索引擎在采集信息的时候是遍历所有可以遍历的Internet空间，以便将所有可以采集的信息都采集下来，而为垂直搜索引擎是检索特定领域的内容，在信息采集的时候仅仅需要采集对应领域的内容，如果对信息采集不加限制的话，不仅浪费时间，

也对信息的结构化抽取造成困难，因此垂直搜索引擎的采集策略应该根据所需要采集的领域，采用一定的策略，来控制所采集的内容，通过分析相应的网页URL，使得信息采集尽可能的在对应的领域内进行。这样一来，由于只对对应领域内的网络空间进行遍历，而且不需要经常的更新信息，就使得垂直搜索引擎在信息采集方面可以大量的节省网络资源，并减少硬件开销，从而降低了构建垂直检索系统的难度。但是，如何使得信息采集只在特定的领域内进行，并形成结构化的领域信息就成了实现垂直检索系统的难点。

为了采集到特定领域内的信息，通常的做法是在信息采集的过程中加入对应的主题相关度分析策略，对一个URL网页，在决定是否处理之前需要对该URL的页面进行主题相关度分析，当该URL的主题相关度满足设定的值时才对该URL进行处理，否则就放弃该URL。因为如果该URL和领域的主题相关，它所包含的URL链接也很可能与领域的主题相关。目前主要的做法是通过提取出对应URL网页中的关键词，将之形成一个词库，然后来进行判定。其主要的思路是：首先确定需要采集的领域，然后在通过领域专家来确定一组能够代表该领域的关键词，将之形成一个主题词库，然后将相应的权重分配给对应的关键词，用该词库来表示一个确定的主题；之后从被分析URL页面中提取其中的关键词，采用向量空间模型算法计算网页的主题相关度来决定页面是否被采集。其主要的算法思想是计算出给定词库中的关键词在URL页面中出现的次数，在进行加权平均，如果关键词出现的次数越多，则认为跟主题词库的相关性越高。然后实际情况是，一个概念可以表示多个关键词，而多个概念也可以表示一个关键词，同时，多个子概念表示的内容也可能是属于父概念的内容。这种基于关键字的主题相关度分析策略对这些概念的关系并不能处理的很好。

## 2.3 基于本体知识库的信息采集策略

### 2.3.1 本体及本体知识库的构建

Ontology 最早属于哲学的范畴，在哲学领域中，本体被定义为“客观存在的一个系统的解释和说明，客观现实的一个抽象本质”，后来被应用到计算机领域，在计算机应用领域中，本体被定义为“概念化对象的明确表示和描述”[3]，

从这个定义中可以发现构造本体的整个流程，即：抽象出客体的对象，然后构建出这些对象间的关系和属性。本体主要有以下四层含义[4][5]。

(1)共享(share)：表示本体表达的是所有对象实体的抽象，是共同的概念集合。

(2)形式化(formal)：表示对象概念可以被计算机读取并且可以被计算机处理。

(3)概念模型(conceptualization)：对客观世界抽取出对象的概念，与客观实体有直接的对应关系，但是又独立于客观实体。

(4)明确(explicit)：指对象概念与对象概念之间有明确的关系。

构建本体主要是为了形成一个领域知识的结合，抽象出该领域内所有对象的概念，然后对这些概念进行明确的定义，并给出这些概念之间的相互关系，本体主要有以下5个基本原语组成。

(1)关系(relations)：表示领域内抽象实体的各种关系，一般通过n维笛卡儿积来表示。如R:A1×A2×…×An。

(2)类(classes)：表示对领域内的抽象实体的描述，形成一个表达各种概念之间的关系的集合。

(3)公理(axioms)：表示的是正确的判定，如概念A是概念B的子类，概念B是概念C的子类，则概念A是概念C的子类。

(4)实例(instances)：代表一个实体对象。

(5)函数(functions)：表达一个确定的关系，如一个函数instanceof（），则表示2个抽象实体之间是类与实例的关系。

本体之间相互关系主要有以下4种。

(1)part-of：表示一个抽象实体是属于另外一个抽象实体的组成部分。

(2)kind-of：表示一个抽象实体是属于另一个抽象实体的范畴。

(3)instance-of：表示一个抽象实体是另一个抽象实体的实例，类似于面向对象中的接口和继承类之间的关系。

(4)attribute-of：表示一个抽象实体是另一个抽象实体的属性，类似于面向对象中的变量。

从以上介绍中可以看出，本体可以对客观世界进行描述，进而可以抽象出特

定领域的概念，并形成一个概念之间的关系的集合，由此就可以对领域内关键词的各种概念关系进行很好的处理。

目前，构建本体主要使用的工具是Protégé，该工具可以提供一个交互式的构建界面。构建领域本体的基本思路是：首先需要确定一个特定的领域，然后在列出该领域内抽象出的实体对象即类，实体对象定义完成之后需要对之进行描述，即定义对象之间的关系和属性，之后创建对应类的实例就完成了本体的创建，本体构建好之后，就可以将其存储在OWL文件中形成本体知识库。

### 2.3.2 基于本体知识库的采集策略

由于本体能够很好的表达领域内相互概念之间的关系，所以将之应用于检索系统，就可以克服通用信息采集策略的不足，将基于关键词的相关度分析策略提高到基于本体的关系匹配，当判定一个URL页面是否需要被处理时，可以分析该URL页面的关键词，然后通过查询该关键词在本体知识库中概念以及跟该概念有关的概念就可以判定出该URL页面是否需要被采集，同时由于本体概念之间的关系，就可以很好的解决一次多义跟一义多词的问题。例如，对于一个面向手机领域的垂直搜索引擎，在采集页面的时候需要判定的关键词是“手机”时，可以通过查询本体知识库，假设发现它和“移动电话”之间是属于同义的关系，就表示这两个关键词是属于同一个概念，包含“移动电话”关键词的网页应该属于要采集的内容。同样的，关键词“诺基亚N95”表示一中手机，可以在本体知识库中查询出跟“手机”是属于类和实例的关系，因此包含“诺基亚N95”关键词的网页也是要采集的内容。如果遇到一词多义或者一义多词的情况，可以先分析关键词所在本体知识库中的父类对象和子类对象是否包含该关键词，如果包含则表示他们是属于同一个概念，需要对该URL页面进行采集，否则则放弃该URL页面，并选择一个新的URL页面进行分析，一直不断重复直到没有新的URL页面。可见，通过构建领域本体知识库，可以对页面的关键词进行概念的匹配，一旦匹配则进行采集，准确率将大为提高。

## 2.4 本章小结

本章首先分析了垂直搜索引擎的总体结构，然后对其四大模块进行逐一介

绍，最后分析了基于关键词的主题相关度采集策略，指出了该策略不能很好的处理一词多义和一义多词的情况。最后通过引入本体的概念，提出了改善这种情况的智能信息采集策略的设计思路。

# 第 3 章 关键技术分析与研究

## 3.1 Heritrix 分析与研究

### 3.1.1 Heritrix 分析与研究

Heritrix 是一个网络爬虫框架，通过使用该框架，可以从 Internet 空间抓取想要抓取的 URL 网页信息。它的主页网址是 http://crawler.archive.org/，Heritrix 的可扩展性是其最显著的优点，通过对 Heritrix 进行相应的扩展，可以实现自己特殊的采集策略。

### 3.1.2 Heritrix 工作原理

多线程和链接队列是 Heritrix 体系的基础和核心，由中央控制器 CrawlController 来控制整个体系的运行和工作。而在其运行时所需要的参数都是从配置文件 Order.xml 中获取，通过这下参数可以生成对应的管理器和资源列表。其整个下载过程是这样的：首先下载列表通过范围控制器 CrawlScope，去除掉不符合要求的 URL 页面，把符合要求的 URL 页面传递给下载地址管理器，形成等待下载队列，而多线程作为整个 Heritrix 的基础，每个线程都会从下载队列中获取需要下载的 URL 页面来进行处理。处理器链就会根据当前系统资源情况，不断形成内部工作链；调用 CrawlURI 对象来分析 URL 页面，从而从 URL 页面中分析出新的 URL 连接，并将之加入到下载任务队列中。整个过程不断的重复直到没有新的 URL 连接，这样形成了一个自动采集网页信息的网络爬虫。Heritrix 的系统框架图如图 3.1 所示。

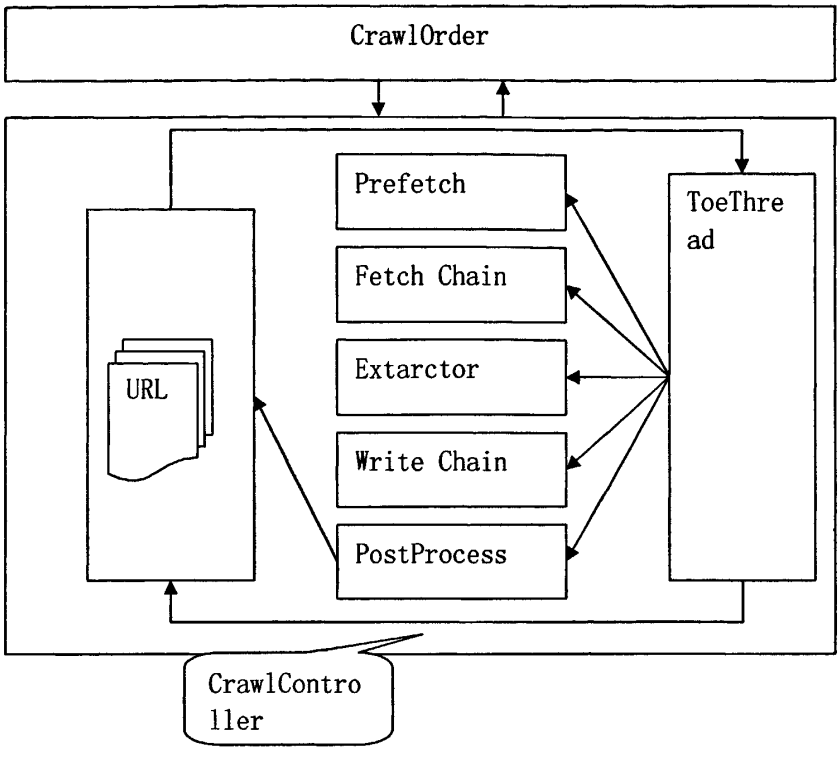


图3.1 Heritrix的系统框架图

### 3.1.3 Heritrix工作原理

Heritrix主要由以下几个组件组成：

(1)CrawlController：中央控制器，它是Heritrix的基础和核心，用来协调和控制整个体系的正常工作。

(2)CrawlOrder：整个抓取工作是从CrawlOrder开始的，通过读取外部配置文件order.xml中所配置好的参数来运行整个组件。

(3)ProcessorChainList：包括一些处理链，在整个抓取任务开始之后，通过ProcessorChainList组件可以对URL页面进行分析筛选。

(4)CrawlScope：当一个抓取任务开始之后，通过CrawlScope组件可以控制整个任务的抓取范围。

(5)ToePool：通过ToePool组件可以不断的产生新的线程，然后新的线程会在Frontier组件中获取一个新的URL链接，之后通过调用其它组件来处理这个链接。

(6)Frontier：Frontier是用协助每个线程来抓取URL页面，通过Frontier

组件可以判定当前线程需要处理的URL连接是否已经被处理过，同时该组件还要负责接受新的连接。

(7)ToeThread：由ToePool组件产生的线程，通过从Frontier组件中获取一个新的URL链接，并对该链接进行一些处理工作。

(8)Web控制台：Heritrix的用户交互界面，通过该界面可以对一些参数进行设置。

(9)Servercache：一个缓冲组件，通过该组件可以查询一些网络爬虫爬行的信息。

## 3.2 Lucene分析与研究

### 3.2.1 Lucene简介

Lucene是一个全文检索框架，通过Lucene框架，可以实现各种搜索引擎的检索功能。Lucene框架没有具体要求需要建立索引的信息的格式，Lucene内部提供了一个实体类 Document，通过从需要建立索引的信息中提取结构化的文本信息，就可实例化Lucene的实体类Document，从而对该信息建立索引。然后就可以通过Lucene提供的各种检索方法来对该信息进行检索了。检索产生的结果会按照Lucene默认的排序算法排好序，然后再将之呈现给用户。

### 3.2.2 Lucene框架构成

Lucene 作为一个开源的全文检索框架，最主要的特点就是其面向对象化。首先其内部的索引文件的建立是与平台无关的，其次其内部的核心类都设置成为抽象类，根据平台的不同对应为不同的具体类来实现该抽象类，Lucene 根据不同的平台将有关的部分封装成为不同的组件，面向对象的思想贯穿了整个Lucene框架的设计，最终形成了一个低耦合高性能，容易再次开发的检索框架。图3.2是Lucene的系统结构图。

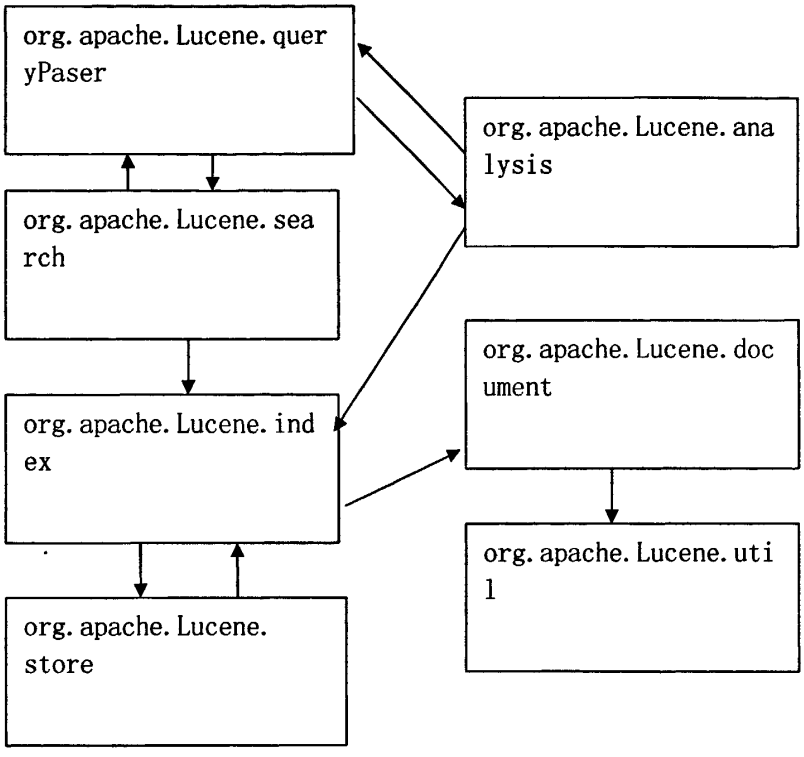


图 3.2 Lucene 系统结构图

从图 3.2 可以看到，Lucene 主要由以下七个部分组成。

(1)org.apache.Lucene.index：该组件是整个框架的索引起点，Lucene 框架主要是通过该组件来建立索引，该组件包含 2 个类：IndexWriter 和 IndexReader，前面的类是用来创建索引文件，后面的类是用于删除索引信息。

(2)org.apache.Lucene.search：该组件是整个框架的搜索起点，当通过 Lucene 框架建立好索引之后，可以通过该组件来对索引文件进行搜索。搜索的结果会通过 Lucene 内部类 Hits 来进行保存。

(3)org.apache.Lucene.queryParser：该组件是一个查询器，当通过 Lucene 框架建立好索引之后，可以通过该组件对关键词进行查询分析。

(4)org.apache.Lucene.analysis：该组件是一个分词器，在通过 Lucene 框架建立索引之前，需要对建立索引的信息进行分词，该组件主要是为建立索引做前期准备。

(5)org.apache.Lucene.util：该组件是一个工具包，内部实现了 Lucene 框架的一些工具类。

(6) org. apache. Lucene. document：该组件是 Lucene 内部提供的存储方式。只要可以从需要建立索引的信息中抽取出结构话的文本信息，就可以建立索引并存储在该组件中。

(7) org. apache. Lucene. store：该组件是一个数据存储包，主要是实现了 Lucene 框架的一些 I/O 操作。

## 3.3 本章小结

本章介绍了 Heritrix 框架的基本概念，分析其框架的体系结构，并对其主要组件进行了介绍，这些都是为后续章节有关本文系统的设计与实现做准备。

# 第 4 章 垂直搜索引擎系统设计与实现

## 4.1 总体系统设计

本项目是通过 Heritrix 框架和 Lucene 框架来构建一个手机垂直检索系统，本系统是基于 J2EE 框架下，设计了一个 MVC 模式的查询接口，可以检索出手机产品的所有相关信息。另外，由于本系统使用 Lucene 来建立索引，同时使用基于本体知识库的采集策略，所以能具有较高的准确率和响应速度。本系统的总体流程如图 4.1 所示。

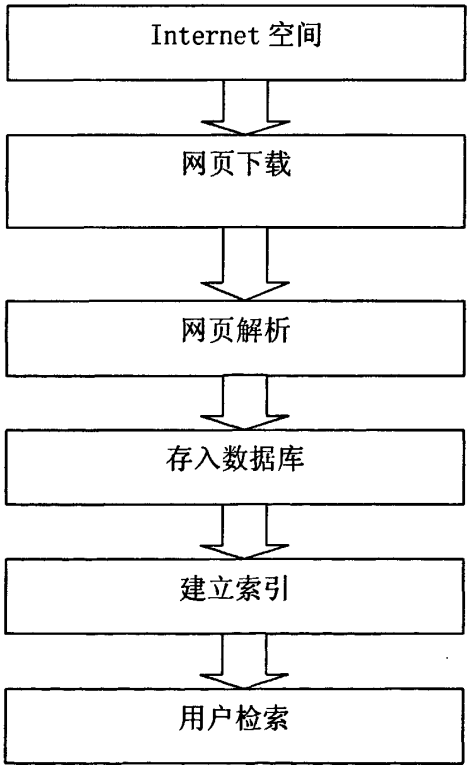


图 4.1 垂直搜索引擎的流程图

(1)信息抓取模块

本项目使用网络爬虫 Heritrix 来进行采集信息，并对其功能进行相应的扩展，将基于本体知识库的信息采集策略扩展进 Heritrix，实现了只抓取相关领

域的信息。

(2)信息抽取模块

本项目通过分析所抓取网页的页面结构，通过使用页面解析工具HtmlParserr来对页面的信息进行抽取，最终提取出结构化的文本信息。

(3)索引模块

常用的信息存储方式是数据库和索引文件方式，数据库方式简单灵活，便于按照产品关键字检索和组织，但是不利于实现全文检索。索引文件方式速度快，能够组织海量的信息，但不利于格式化存储。本项目将结合两种方法，将所有的信息存储在数据库，而在索引文件中只存储需要检索的字段和跟数据库一一对应的字段，其中通过 Lucene 来建立索引，并通过引入 JE 分词工具来取代 Lucene 自带的分词工具。

(4)检索模块

由于本系统采用 J2EE 框架，当用户在查询界面输入查询关键词时，后台配置文件会根据该关键词分发到对应的处理方法，然后通过对应的业务逻辑在索引库中检索关键词，当在索引库中查询到该关键词时，系统会到数据库中查询到对应产品的详细信息，最后以特定的排序方式返回给用户。由于系统三层逻辑的严格分离，使得查询效率和代码的重用性大大提高。

## 4.2 信息采集模块的设计与实现

### 4.2.1 手机本体知识库的构建

本体的基本知识以及构建工具在第二章已经详细介绍，接下来介绍通过Protégé工具构建手机本体。由于篇幅所限，以下只演示构建的主要过程，详细流程如下。

(1)当电脑上已经装好Protégé工具时，启动Protégé工具，如果启动正常，在电脑屏幕上会有如下结果。

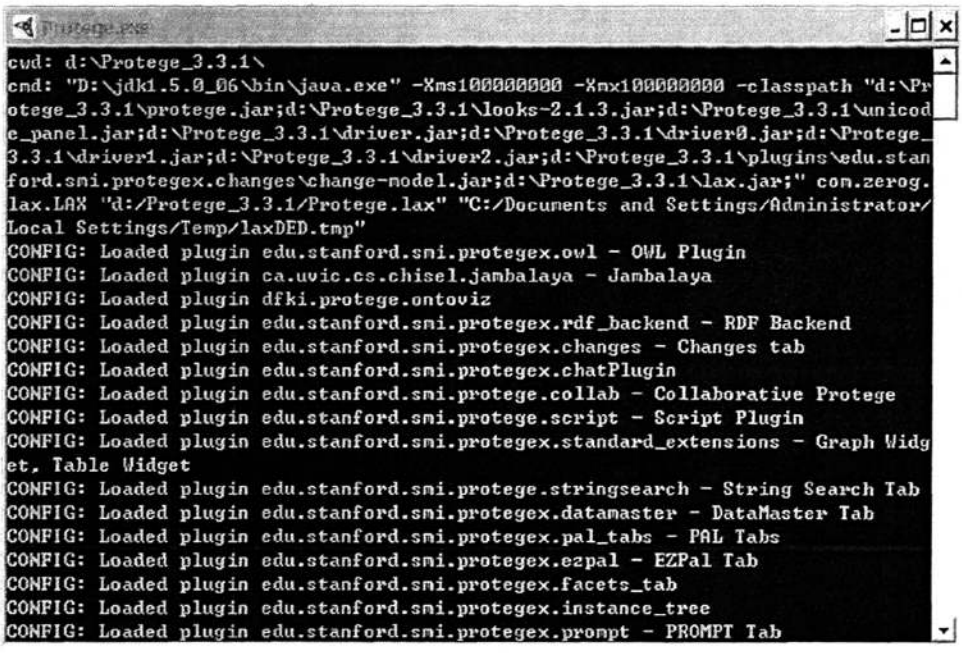


图 4.2 启动 Protégé的屏幕截图

(2) Protégé工具启动之后，需要建立一个新的本体项目，点击最左上角的 File 按钮，然后在点击 NewProject，之后就会出现一个对话框，在此对话框中选择 OWL/RDF Flies 后，一直点击 Next，就可以建立一个新的本体工程，界面如下。

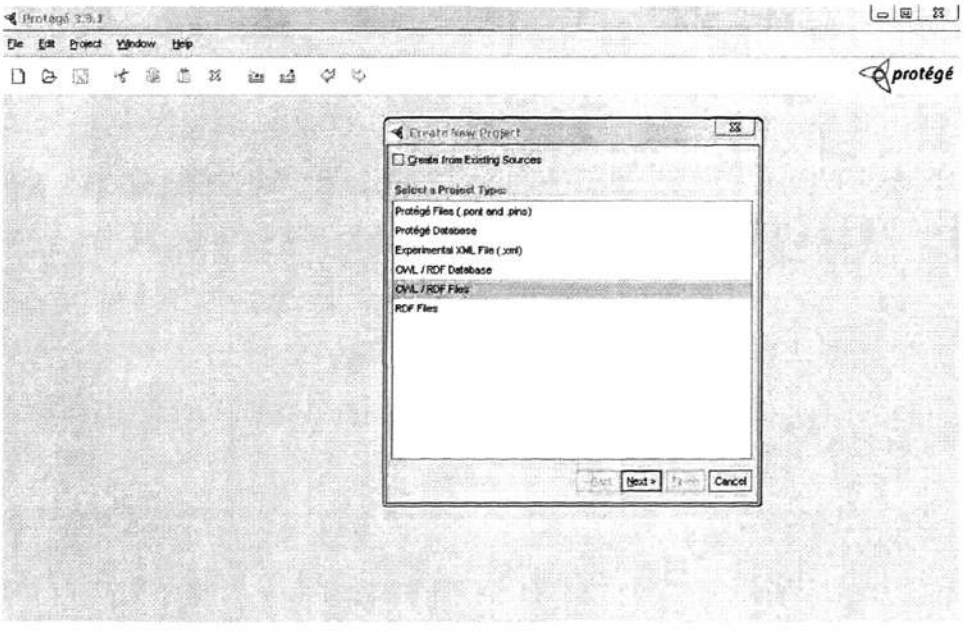


图 4.3 构建本体的屏幕截图

(3)本体项目建立好之后，就需要添加实体对象，也就是类。在 Protégé工具的状态栏下面会有五个标签，分别是 Metadata(元类)，OWL Classes(OWL 类)，Properties(属性)，Individuals(个体)，Forms(表单) 。通过选择 OWL Classes 标签就可以对类进行添加删除。当类建立好之后，可以通过右键点击所建立好的类，然后可以选择其中 subclass 来建立子类。整个类的构建界面如下。

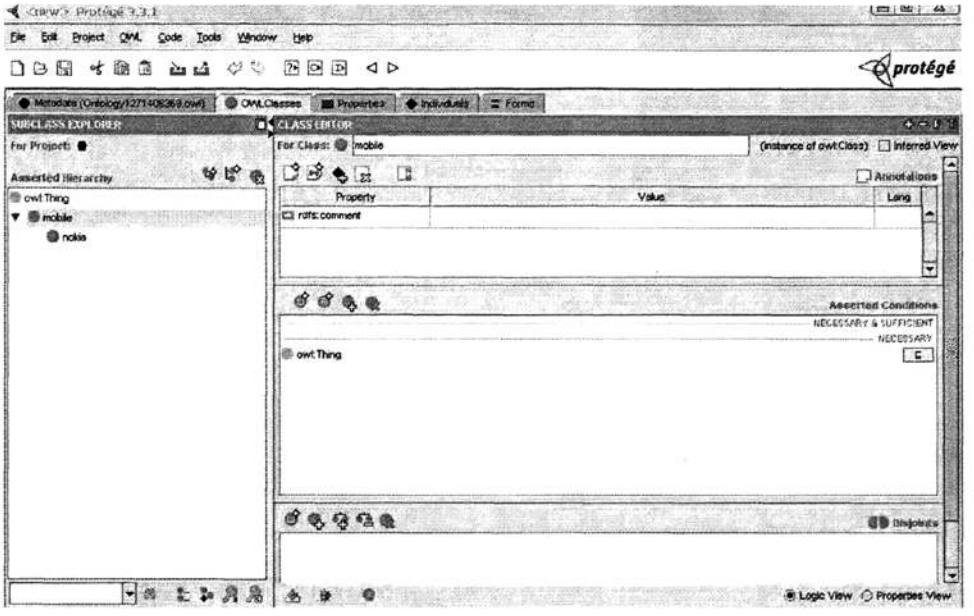


图 4.4 添加实体对象的屏幕截图

(4)当类建立好之后，就需要添加对应的属性，选择对应的 Properties 标签来进行编辑，比如要新建一个 Datatype，选择 Datatype 标签，Name 改为 HasName，然后选择右边的 Range 为 String，说明这是一个 String 类型，其构建界面如下。

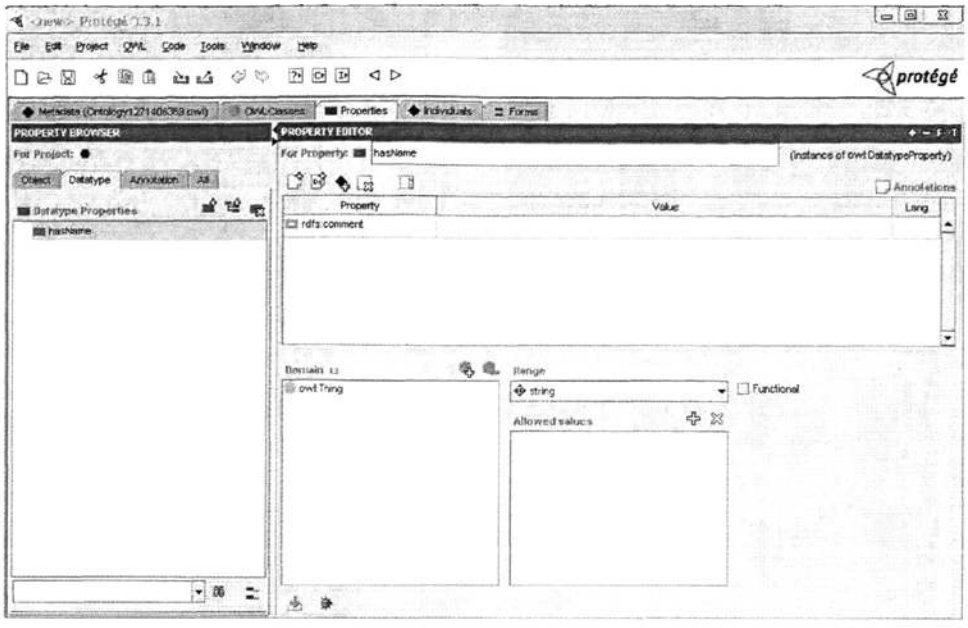


图 4.5 添加属性的屏幕截图

(5)属性建立完之后，就需要对所有的类建立相应的实例，选择 Individuals 标签来进行编辑，在其中 Asserted Instances 中点击 Create Instance 来添加对应的实例，其建立界面如下。

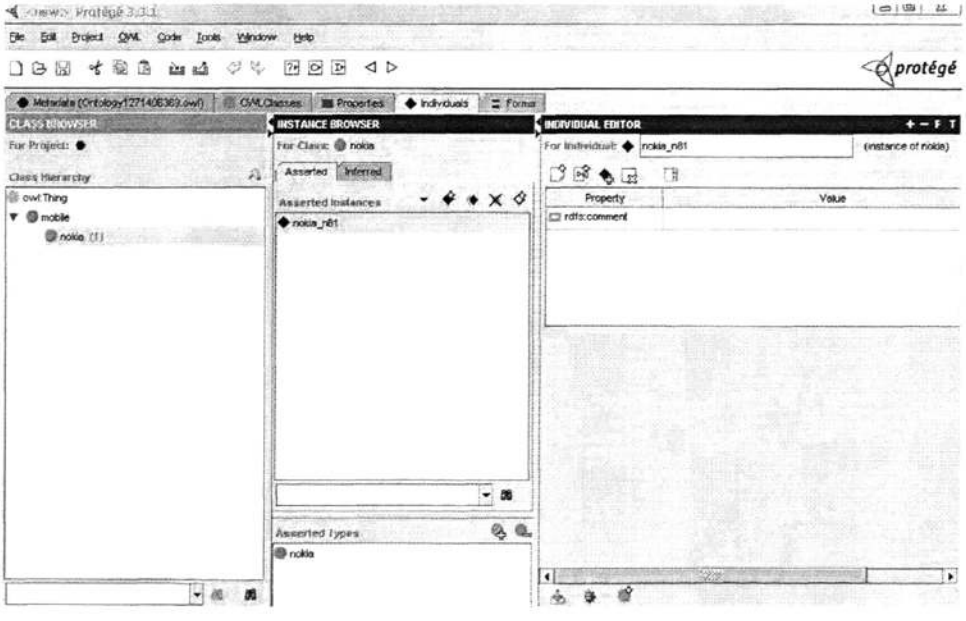


图 4.6 添加实例的屏幕截图

(6)建立完成之后，就可以通过点击 File 中的 Save 来保存整个项目，保存完成之后，整个本体就构建完成。

### 4.2.2 定向网页抓取分析

搜索引擎系统都需要建立好需要被检索的信息库，用来存储所下载下来的网页信息。这整个过程是需要在用户检索之前进行的，而不是在用户发出检索请求之后在去整个网络上去搜索用户的查询信息，因为这样做导致的直接结果是在用户发出检索之后，获得结果之前将会有一个很长的等待过程，这样就会使得用户的体检效果很差，显然是不合理的。所以就需要将用户需要的信息预先通过一些工具下载整理并储存到本地数据库中，如此就可以及时的响应用户的检索请求，因此，在整个检索之前就需要通过网络爬虫来收集对应的网页信息。

在系统抓去对应的网页信息的时候，首先需要考虑的是如何全面的抓取相关领域的所有信息，在此基础上再去考虑如何尽可能的减少抓取跟本领域无关的网页信息。

本系统是通过网络爬虫 Heritrix 来对相关领域的信息进行抓取，而在抓取的过程中，由于 Heritrix 本身默认是对所有能抓取到的网页信息都进行抓取，对网页信息并没有过滤选择功能，这样显然不符合本系统的需求。

因此，在通过 Heritrix 来对网页的信息进行抓取之前，需要对 Heritrix 的相关组件进行相应的扩展，以达到能够有选择的对网页信息进行抓取。网络爬虫框架 Heritrix 的最大特点之一就是其强大的扩展功能，只要对其提供的相应的扩展接口进行实现，就可以实现不同的抓取逻辑，以下具体的介绍如何对 Heritrix 进行扩展。

为了满足开发者自身的抓取逻辑，Heritrix 提供了 Extractor 和 FrontierScheduler 两个接口供开发者自己实现，通过实现这两个接口，可以对网页信息的抓取进行有效的控制。Extractor 扩展接口的功能，在 Heritrix 内部 Extractor 继承自 org.archive.crawler.extractor.Extractor 这个抽象基类，其主要是为对当前的页面进行分析，抽取出当前页面的的新的 URL 链接。其抽取的主要过程如下。

(1)通过当前的页面来获取其 URL 对象，然后在根据 Heritrix 自带的方法将该 URL 对象转换为对应的 URL 网页字符串。

(2)通过获取该 URL 页面的所有新的 URL 链接，然后依次对这些链接进行处理，以判定这些 URL 链接是否合法，并检查这些 URL 链接是否已经被处理，如果

这些URL链接合法并且没有被处理过，则将其加入到对应的待处理队列以等待后续的处理。否则则将其放弃处理。

因此通过实现Extractor扩展接口，就可以改变默认的处理方法，从而引进开发者自身需要的处理方法。

FrontierScheduler扩展接口的功能：FrontierScheduler是一个PostProcessor，它主要是在确定一个URL是否下载之前，对其进行检查是否符合要求，例如，当一些格式的页面不需要被下载时，就可以在该扩展接口中对其进行删除，其处理的主要过程如下。

(1)通过分析所有URL连接在一个队列中的优先级，选择其中高优先级的URL链接进行处理，之后再对整个队列进行遍历处理。

(2)将从当前URL链接中分析的新的URL链接加入到需要处理的队列中等待后续处理；

通过扩展该接口，开发则就可以选择自身需要的URL链接进行下载，可以尽可能的减少无关领域的页面的下载。

### 4.2.3 定向网页抓取机制设计

在使用Heritrix框架进行采集相关信息之前，需要一些前期准备工作，首先需要获得Heritrix框架源码或者其工具包，Heritrix框架的源码跟工具包都可以在其主页http://crawler.archive.org/download.html上面进行下载。将所需要的工具包和源码下载到本地之后，解压开可以看到，在src目录下包含其所有的源码，在lib目录下包含其所有的工具包，可以通过命令行或者导入其源码来启动Heritrix，本项目是将其源码src目录全部导入到MyEclipse中的一个工程项目下来运行。下载完Heritrix框架的工具包之后，解压到一个本地目录下，导入之后可以看到，在工程项目的目录下面有一个conf目录，其中包含了一个配置文件heritrix.properties，通过该配置文件可以配置许多跟Heritrix启动有关的参数。主要的配置如图4.7所示。当通过类org.archive.crawler.Heritrix启动Heritrix时，需要对该文件中的参数进行一些设置，例如设置新的用户名和密码等，其界面如下。

```
heritrix.properties
#############################################################################
# H E R I T R I X  P R O P E R T I E S
#############################################################################

# Properties with 'heritrix.' or 'org.archive.crawler.' prefix get loaded
# into System.properties on startup so available via System.getProperties.

# Version is filled in by the maven.xml pregoal. It copies here the project
# currentVersion property.
heritrix.version = 1.10.0

# Location of the heritrix jobs directory.
heritrix.jobsdir = jobs

# Default commandline startup values.
# Below values are used if unspecified on the command line.
heritrix.cmdline.admin = admin:admin
heritrix.cmdline.port = 8080
heritrix.cmdline.run = false
heritrix.cmdline.nowui = false
heritrix.cmdline.order =
heritrix.cmdline.jmxserver = false
heritrix.cmdline.jmxserver.port = 8081

#############################################################################
# L O G G I N G                                                             #
```

图4.7 启动Heritrix的屏幕截图

有关信息采集策略的思想在第二章已经详细描述，这里对如何扩展Heritrix进行详细描述。

在导入的项目工程文件中创建一个包 com.skally.mypconlineextractor，在该包下创建类MobilepconlineExtractor，它是专门为爬取“太平洋手机网”中各款手机而编写的URL抽取策略类，它是通过对Extractor扩展接口进行实现来产生新的URL抽取功能的，代码片段如下所示。

```
protected void extract(CrawlURI curi) {
    .......
    String content = cs.toString();
        try {
            BufferedReader reader = new BufferedReader(new
StringReader(content));
             // 一行行的读
            String line = reader.readLine();
            while (line != null) {
                .....
                if(!"".equals(titleStr)){

    com.skally.Utils.Util.getURLMap().put(hrefStr,titleStr);
                    }
                }
                line = reader.readLine();
            }
}
```

其主要的功能是通过将CrawlURI对象转变为URL网页字符串，然后通过该字符串获得其URL页面的所有内容，通过分析该内容的每一行来抽取出新的URL链接并将之放进待处理队列。

在将整个项目导入之后，就可以在该项目下创建一个对应的包com.skally.mypconlineFrontier，同时在该包下面创建一个对应的类FrontierSchedulerForPconlineMobile，它可以对URL链接来进行对应的筛选，它是通过实现FrontierScheduler扩展接口来获得筛选功能的，主要代码片段如下所示。

```
protected void schedule(CandidateURI caUri) {
    •••••
    if(!com.skally.Utils.Util.getURLMap().containsKey(URL)){
          return;
    }else{
          String                          URLName                          =
com.skally.Utils.Util.getURLMap().get(URL);
                if(com.skally.Utils.Util.getInsMap().isEmpty()){
                   com.skally.Utils.Util.loadModel(filePath);
                }
    if(com.skally.Utils.Util.getClMap().containsKey(URLName)||com.
skally.Utils.Util.getInsMap().containsKey(URLName)){
       String hlName = com.skally.Utils.Util.extract(URL);
       List subList = null;
       List supList = null;
       if(com.skally.Utils.Util.getClMap().containsKey(URLName)){
                        subList                                          =
com.skally.Utils.Util.instanceFromOntClass(com.skally.Utils.Util.
getClMap().get(URLName));
                        supList                                          =
com.skally.Utils.Util.subClassFromOntClass(com.skally.Utils.Util.
getClMap().get(URLName));
                    }

if(URLName.indexOf(hlName)!=-1||hlName.indexOf(URLName)!=-1){

    getController().getFrontier().schedule(caUri);
                    }else{

    if(subList!=null&&subList.contains(hlName)){

    getController().getFrontier().schedule(caUri);
    •••••
```

其主要思想就是在获得一个URL网页之后，分析其中的关键词，看其是否在已经建立好的本体知识库中，又或者是与本体知识库中的概念有对应的关系，如果有则下载该URL页面，否则则放弃处理该URL页面。

在实现了这两个扩展接口之后，还需要将其配置进Heritrix框架中才能够运行，在工程项目中找到modules包，在该包下面打开对应的Processor.options文件，然后将具体实现类MobilepconlineExtractor和

FrontierSchedulerForPconlineMobile 添加到 Processor.options 文件中，添加方式是把之前实现好的两个实体类直接加进去即可，其界面如图如图 4.8 所示。


图 4.8 添加实体类的屏幕截图

对 Heritrix 扩展完之后，就可以启动 Heritrix 来对网页信息进行抓取，本系统是通过网页界面来启动 Heritrix。

在工程项目的 org.archive.crawler 包中有一个类 Heritrix，运行该类之后，Heritrix 就会被启动起来。在将 Heritrix 成功启动之后就可以打开浏览器同时输入网址 http://127.0.0.1:8080，完成之后就可以看到 Heritrix 的登陆主界面，结果如图 4.9 所示。


图 4.9 Heritrix 的登录界面

输入在 Heritrix.properties 中设置好的用户名和密码，就可以进入 Heritrix 的主界面并开始利用 Heritrix 来创建新的抓取任务来抓取特定网页的信息了。在 Heritrix 页面上可以看到有五个对应的标签，单击菜单栏上的“Jobs”标签就可以创建新的抓取任务，其总共有以下四种抓取方式：

(1)Based on a recovery：如果在之前的任务中暂停过或者设置过一些断点，就可以通过这个方式以之前的断点来开始新的抓取任务。

(2)Based on a profile：Heritrix自带了一些已经配置好的模块，可以通过这个方式以Heritrix自带的模板来生成新的抓取任务。

(3)With defaults：通过这个方式可以创建一个需要手动配置参数的抓取任务。选择With defaults之后会出现新建任务窗口，填入抓取任务的名称，描述信息，起始种子网址信息。这个以抓取太平洋手机网的手机产品信息为例创建抓取任务。

(4)Based on existing job：当之前已经抓取过一些网页的信息的时候，就可以这个方式来以之前抓取过的任务为模板来创建新的抓取任务。

在设置好了抓取方式之后，就会出现如下图4.10所示的界面。

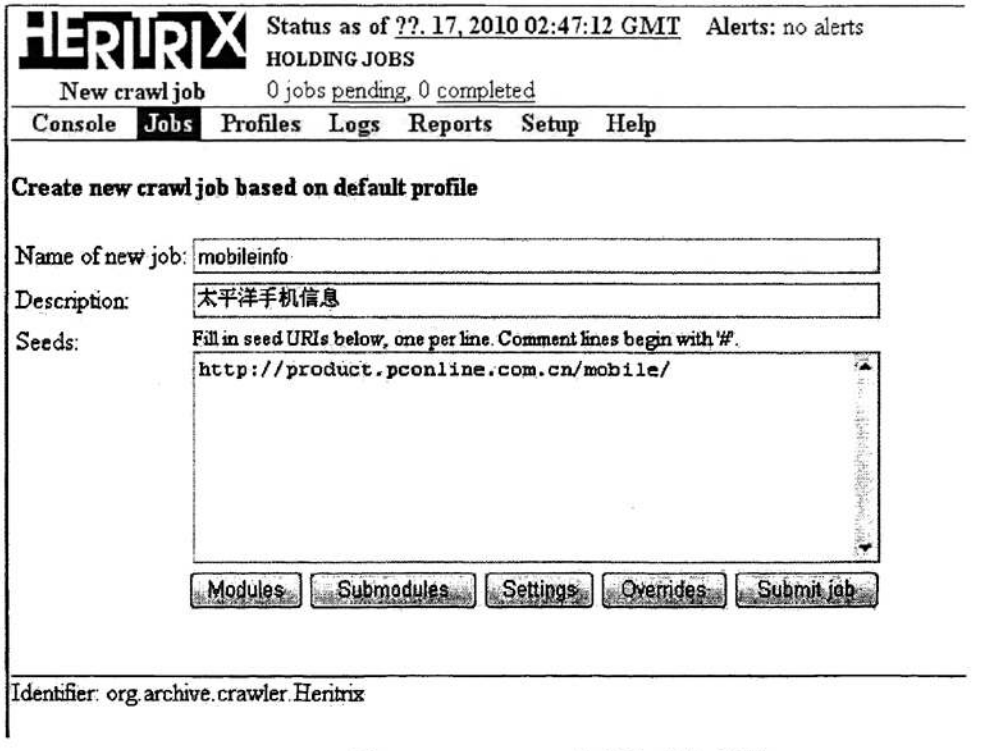


图4.10 Heritrix的抓取任务界面

本系统是采用第三种方式来创建抓取任务，因此，在抓取任务创建好之后，还需要对任务做详细的设置，首先点击图4.10最下面的Modules模块按钮，页面会跳转到有关处理链的设置页面，如图4.11所示，这里需要配置的内容如下。

Select Extractors，首先REMOVE掉Heritrix默认的Extractor，然后添

加扩展类“com.skally.mypconlineextractor.MobilepconlineExtractor”以实现URL的抽取。

Select Writers，首先REMOVE掉Heritrix默认的Writer，然后在选择“org.archive.crawler.writer.MirrorWriterProcessor”。表示以镜像的方式存储下载下来的内容。

Select Post Processors，首先REMOVE掉Heritrix默认的Frontier，然后在添加扩展类以实现URL的筛选。

其他的配置大多数使用默认的设置，额外需要配置一下“user-agent”，将@VERSION@和PROJECT_URL_HERE设置成对应的Heritrix的版本信息和一个完整的URL的链接，然后在“from”对应文本框中填写一个正确Email地址即可。界面如图4.11所示。

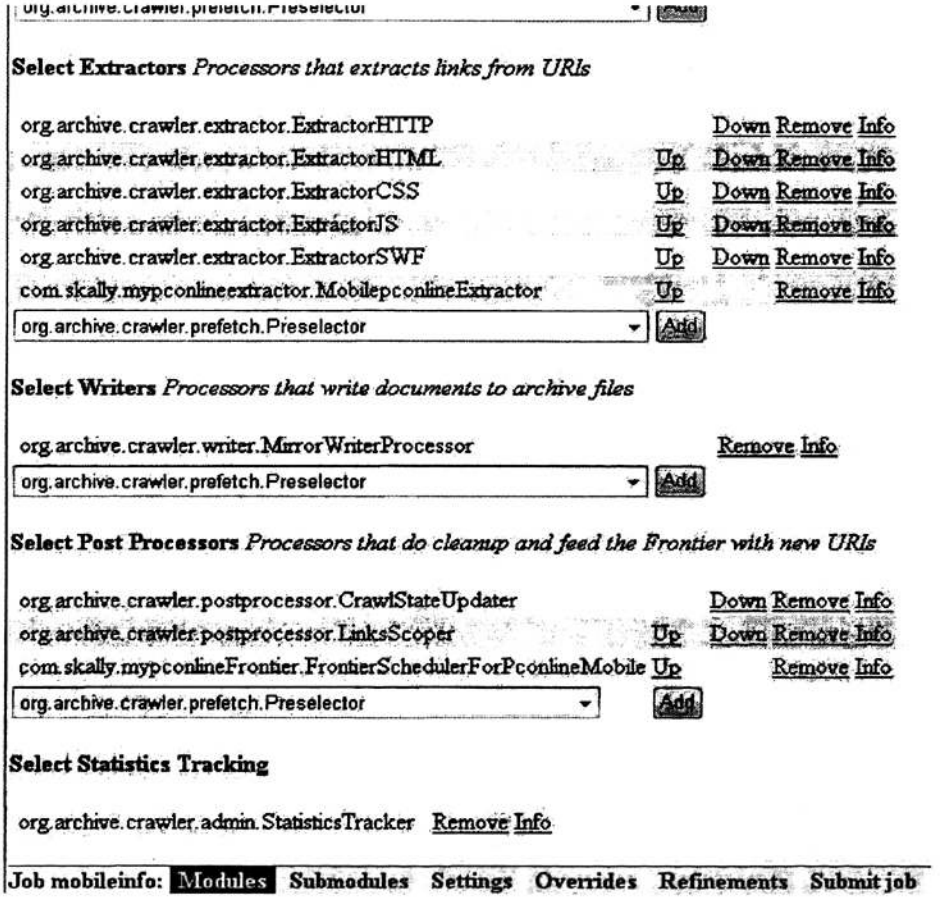


图4.11 Heritrix配置图

当正确设置了各个参数后，需要提交该抓取任务，只要单击submit job标

签即可。然后单击“Console”标签回到主界面，再点击 “Start”按钮，就会将刚才提交的任务激活，然后 Heritrix 就会依据配置好的参数来对网页信息进行抓取。界面如图 4.12 所示。

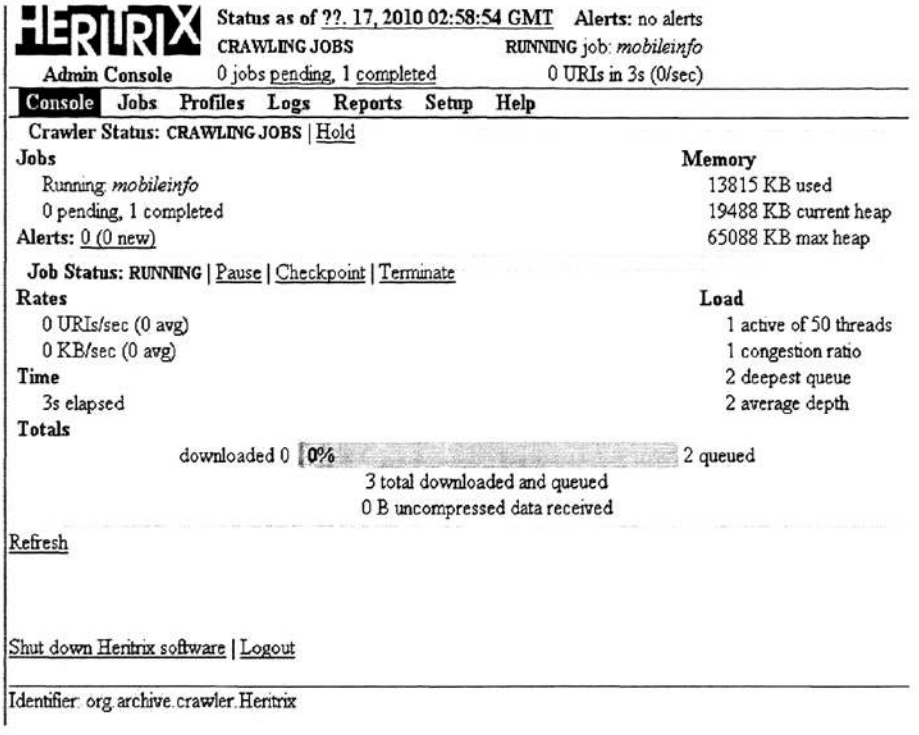


图 4.12 任务激活图

经过 MobilepconlineExtractor 和 FrontierSchedulerForPconlineMobile 这两个具体实现类对 URL 网页进行筛选处理之后，就可以尽可能的抓取相关领域的信息并且剔除掉跟领域无关的信息，当 Heritrix 成功的抓取了 URL 网页信息之后，就会以之前配置好的镜像的方式存储在磁盘上，所谓镜象方式存储，就是将 URL 地址按“/”进行切分，进而按切分出来的层次存储，如当下载的网页地址是 http://product.pconline.com.cn/mobile/asus/1.html 是，就会以 product.pconline.com.cn\mobile\asus\1.html 的格式存储在磁盘中。抓取结果如图 4.13 所示。


图 4.13 下载后的文件结构

## 4.3 信息抽取模块的设计与实现

### 4.3.1 设计思想

由于使用 Lucene 建立索引并没有规定信息的格式，只需要将信息转化为文本信息并且构造一个 Document 对象，就可以建立索引，所以本模块主要是对 HTML 形式的网页信息进行抽取，利用抽取工具从 HTML 网页中提取出相应的文本内容，然后将之建立索引并且存入数据库。

常用的抽取方法是通过正则表达式去提取 HTML 文本，而在使用正则表达式的时候需要考虑很多细节，比如一些大小写和空白符都必须考虑，不然就得不到想要的结果，同时正则表达式的复用性太差，针对每个特定的网页都需要单独的写正则表达式去提取。针对这一情况，本系统使用 HtmlParser 工具来抽取 HTML 文本的信息。

HtmlParser 是一个纯 Java 写的 HTML 解析的库，HtmlParser 不依赖于其它的 Jave 库，HtmlParserr 主要用于改造或提取 HTML。HtmlParser 能超高速解析 HTML，而且不会出错。使得搜索引擎开发者摆脱了繁琐的正则匹配过程，只需要通过 HtmlParser 提供的 API，就可以很方便的提取特定文本，大大提高开发效率，从而方便了开发者对特定网页信息的提取。HtmlParser 包的主要结构如表 4.1 所示。

表 4.1 HtmlParser 的主要包结构

| | |
|---|---|
| org.HtmlParserr | 对 HTML 处理的开发人员提供基本的 API |
| org.HtmlParserr.lexerapplications.tabby<br>org.HtmlParserr.lexerapplications.thumbelina<br>org.HtmlParserr.parserapplications<br>org.HtmlParserr.parserapplications.filterbuilder<br>org.HtmlParserr.parserapplications.filterbuilder.layouts<br>org.HtmlParserr.parserapplications.filterbuilder.wrappers | 调用 API 解析 HTML 的实例程序 |
| org.HtmlParserr.nodes<br>org.HtmlParserr.tags | 所有 Nodes 和 Tags 的定义 |
| org.HtmlParserr.lexer | 底层的 IO 子系统 |
| org.HtmlParserr.scanners | 扫描 Tags |
| org.HtmlParserr.beans | 调用 API 解析 HTML 的 javabean 实例程序 |
| org.HtmlParserr.filters<br>org.HtmlParserr.visitors | 两种访问节点过滤器和 visitor 模式 |
| org.HtmlParserr.http | 负责建立 http 连接 |
| org.HtmlParserr.sax | 为 HTML 实现的一个 |

| | SAX parser |
|---|---|
| org.HtmlParserr.util<br>org.HtmlParserr.util.sort | 通用类，包括查找和排序 |

### 4.3.2 抽取实现

首先从 http://HtmlParserr.sourceforge.net/ 这一网址中下载 HtmlParser，本模块需要 HtmlLexer.jar 和 HtmlParser.jar 两个包。将其加入 ClassPath 中，这样准备工作也就做好了。

通过定义 HtmlParser 的节点过滤器可以很方便的抓取自己需要的节点信息，比如想要抓取标签 游戏 的内容信息，可以定义如下过滤器。

NodeFilter filter = new andFilter(new TagNameFilter( "div" ),new hasAttributeFilter( "class" ," skally" ))

通过该过滤器,HtmlParser就会提取页面上所有的div标签,并且含有class属性的节点。然后就可以获取其对应的文本信息。HtmlParser 共有以下十六种过滤器。

AndFilter：相当于 AND 操作符，接受所有同时满足两个 Filter 的节点。

CssSelectorNodeFilter：接受所有支持 CSS2 选择器的节点。

HasAttributeFilter：接受所有含有某个属性的节点。

HasChildFilter：接受所有符合含有该子节点的节点。

HasParentFilter：接受所有符合含有该父节点的节点。

HasSiblingFilter：接受所有符合含有兄弟节点的节点。

IsEqualFilter：接受所有和某个节点相同的节点。

LinkRegexFilter：接受所有 linkTag 标签的 link 值，匹配出对应的节点。

LinkStringFilter：匹配出给定字符串的节点。

NodeFiltter：接受所有指定的类的节点。

NotFilter：接受所有不含有指定节点的节点。

OrFilter：相当于一个 OR 操作符，接受满足其中一个条件的节点。

RegexFilter：接受满足对应正则表达式的节点。

StringFilter：接受满足指定字符串的节点。

TagNameFilter：接受满足指定 TagName 的节点。

XorFilter：相当于一个 XOR 操作符。

通过这些过滤器的不同组合，可以很方便的获取自己需要的网页信息。针对已经下载下来的手机网页信息，分析其中网页的结构，需要抽取出手机的名称，类型，详细参数，价格等信息，主要的抽取代码如下。

```
//创建属性过滤器
NodeFilter attributeFilter = new AndFilter(new
TagNameFilter("td"),new OrFilter(new HasAttributeFilter("class",
"tdL"),new HasChildFilter(new TagNameFilter("span"))));
//创建标题过滤器
NodeFilter titleFilter = new TagNameFilter("h1");
//创建图片过滤器
NodeFilter imageFilter = new AndFilter( new TagNameFilter("div"),new
HasAttributeFilter("class", "dPic"));
//创建价格过滤器
NodeFilter priceFilter = new AndFilter( new
TagNameFilter("span"),new AndFilter( new HasChildFilter(new
TagNameFilter("em")),new HasAttributeFilter("class", "mark")));
```

抽取结束后，所有生成的文本文件都保存在程序指定的目录下面，如图 4.14 所示。在本系统中，所有文件的文件名都是以手机名称-型号命名，其内部主要包含的手机产品的所有详细信息如价格、参数等。

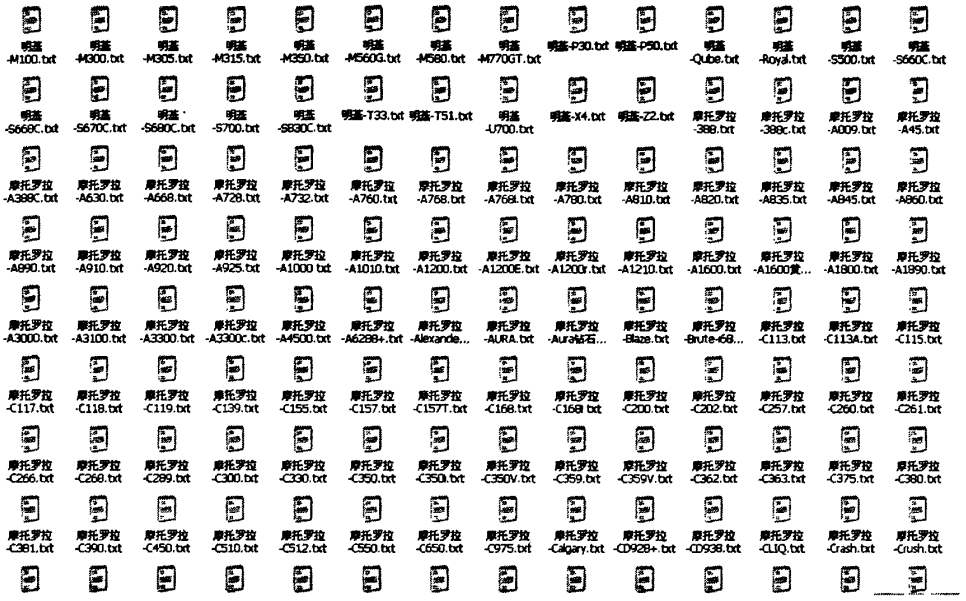

图 4.14 生成的文本文件

## 4.4 索引模块的设计与实现

本节主要实现搜索引擎中数据的存储功能以及索引的建立，由于数据库不利于全文检索而索引不利于全文检索，本系统将两者结合起来，先将结构化数据存入数据库，在通过 Lucene 对数据库中的数据建立索引。

### 4.4.1 数据库的设计

本数据库的功能：主要用来存储产品的相关信息，产品的相关信息存储完之后，数据库会生成产品的唯一标识值 id，在通过 Lucene 建立索引库的时候将该 id 也存入索引库，这样当索引到产品的时候就可以同过标志值 id 在数据库中查询到产品的详细信息。

本数据库的设计与实现：由于本系统数据量不是特别大，所以只需要小型数据库就可以满足所有的要求，因此本系统采用 MySQL 作为后台数据库。同时采用了 Hibernate 框架来将产品的信息写入数据库。Hibernate 是一个开放源代码的对象关系映射框架，它对 JDBC 进行了非常轻量级的对象封装，使得 Java 程序员可以随心所欲的使用对象编程思维来操纵数据库。 Hibernate 可以应用在任何使用 JDBC 的场合，既可以在 Java 的客户端程序使用，也可以在 Servlet/JSP 的

Web应用中使用，最具革命意义的是，Hibernate可以在应用EJB的J2EE架构中取代CMP，完成数据持久化的重任。

本系统中是将手机产品的名称、型号、内容等存入数据库，依照这些数据来确定数据的所有字段，其格式如表4.2所示：

表4.2 格式表

| | |
|---|---|
| name | 表示品牌的名称 |
| type | 表示型号 |
| content | 表示具体内容 |
| summary | 用于显示在搜索结果的信息摘要 |
| productURL | 原始URL |
| imageURL | 图片的文件名 |
| price | 产品的价格 |
| updatedTime | 最后的更新时间 |
| id | 产品的ID |

由于是使用 Hibernate 连接数据库，所以需要根据数据库建立对应的实体类，实体类如下：

```
public class Product {
    private int id;
    private String name;
    private String type;
    private String content;
    private String summary;
    private String productURL;
    private String imageURL;
    private String price;
    private String updatedTime;
    ..........
}
```

同时要生成对应的 get()，set()方法，还需要在 Product.hbm.xml 文件中配置实体对象与关系数据库的映射，主要配置如下：

```
<hibernate-mapping package="com.skally.model">
    <class name="Product">
        <id name="id">
            <generator class="native"/>
        </id>
        <property name="name"/>
        <property name="type"/>
        <property name="content" length="2000"/>
        <property name="summary" length="1000"/>
        <property name="productURL" length="600"/>
        <property name="imageURL" length="600"/>
        <property name="price"/>
        <property name="updatedTime"/>
    </class>
</hibernate-mapping>
```

对象关系映射建立之后，在将该配置加入到Hibernate.cfg.xml文件中，同时在该文件中配置好数据库连接。主要配置如下：

```
<property
name="hibernate.connection.driver_class">com.MySQL.jdbc.Driver
</property>
 <property
name="hibernate.connection.URL">jdbc:MySQL://127.0.0.1/mobiledb
</property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password">root</property>
<property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect
</property>
<property name="hibernate.show_sql">true</property>
 <property name="hibernate.hbm2ddl.auto">update</property>
 <property
name="hibernate.current_session_context_class">thread</property>
<mapping resource="com/skally/model/Product.hbm.xml"/>
```

配置完成之后，就可以通过ProductToDB类来进行数据的插入操作，如果数据量巨大，可以通过批量插入以提高效率。在数据插入完成之后，每条数据都会自动生成一个id，该id可以作为标识，在建立索引的时候，传递给索引，作为索引中的Field将其加入索引，以实现索引与数据库数据的一一对应。

### 4.4.2 索引的设计

主要功能：当将结构化的信息存储在数据库之后，就需要对数据库中的所有信息来建立索引，然后形成一个索引数据库，本系统只将名称加上型号再加上主要参数来作为一个字段建立对应的索引，同时将对应 id 仅仅存储在索引库中。本系统的索引主要分成了两个模块进行实现，首先查询出数据库中的每条记录，然后对每条记录在通过类 ProductIndex 建立索引。Lucene 在建立索引过程中主要是通过 Document 类来完成的，按照之前的数据库设计，需要建立对应的手机实体类，通过类 Product 来封装手机产品的所有信息，然后将从数据库中查询出的每条记录构造成对应的 Product。并将 Product 中需要被索引的字段通过 buildDocument 方法来构建为每个 Document 对象，最后通过类 ProductIndex 中的 main()方法将之建立索引并且形成索引库，最终完成整个索引过程的建立。整个流程图如图 4.15 所示：

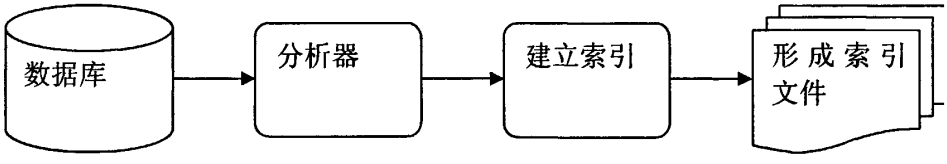


图 4.15 索引过程图

其主要代码如下。

```
public static Document buildDocument(Product product, int id) {
        Document doc = new Document();
        //保存id
        Field idField = new Field("productid", id + "", Field.Store.YES,
                Field.Index.UN_TOKENIZED);
//针对名称构建一个Filed
        Field name = new Field("name", product.getName(),
                Field.Store.YES, Field.Index.TOKENIZED);
        //针对型号构建一个Filed
        Field type = new Field("type", product.getType(),
                Field.Store.YES, Field.Index.TOKENIZED);
        String allName
= product.getName()+product.getType()+product.getContent();
        allName = allName.trim();
        //针对名称加型号构建一个Filed
        Field all = new Field("all", allName, Field.Store.YES,
Field.Index.TOKENIZED);
        doc.add(idField);
        doc.add(name);
        doc.add(type);
        doc.add(all);
        return doc;
    }
    public static void main(String[] args){
        Session session = null;
        try {
            analyzer = new MMAnalyzer();
            writer = new IndexWriter(indexPath, analyzer, true);
            //开启事务
            session = HibernateUtils.getSession();
            session.beginTransaction();
            Query query = session.createQuery("from Product");
            for (Iterator iter=query.iterate();iter.hasNext(); ) {
                Product pro = (Product)iter.next();
                int id = pro.getId();
                Document document = buildDocument(pro,id);
                writer.addDocument(document);
            }
            //索引优化
            writer.optimize();
            //提交事务
            session.getTransaction().commit();
        }
```

(一). buildDocument 方法

buildDocument 方法的功能：对每个 Product 对象，提取出其中需要被索引的字段，然后将该字段构造成一个 Document 对象，并且设置好 Document 中 Field 的属性。

buildDocument 方法的设计和实现：在设置 Document 中 Field 的属性的时候，可以通过两个字段的组合来表示对应的存储方式和索引方式。以构建出所有的存储和索引方式。

其中 Store 用来表示 Field 对象是否需要被存储进索引库

(1)Store.NO 表示构建的 Filed 对象不需要被存储进索引库

(2)Store.YES 表示构建的 Filed 对象需要被存储进索引库

(3)Store.COMPRESS 表示构建的 Filed 对象需要被压缩存储进数据库

其中 Index 用来表示 Field 对象是否需要被索引：

(1) Index.NO 表示构建的 Filed 对象不需要被索引

(2) Index.UN_TOKENIZED 表示构建的 Filed 对象无需被分词

(3) Index.TOKENIZED 表示构建的 Filed 对象需要被分词之后在建立索引

(4) Index.NO_NORMS 表示构建的 Filed 对象需要被索引，但是不需要被分词

本系统是要能搜索出手机的信息，因此只需要对手机的品牌加上型号建立索引即可，索引对这两个字段构造的 Filed 对象需要进行存储，分词之后再建立索引，而对于手机产品的 id 这个字段由于仅仅是需要跟数据库对应，因此只需要对构造出的 Filed 对象进行存储就可以了。根据以上分析，定义 Document 中各类 Field 属性如表 4.3 所示：

表 4.3 属性表

| Field | 属性 |
|---|---|
| idField | Field.Store.YES Field.Index.UN_TOKENIZED |
| name | Field.Store.YES Field.Index.TOKENIZED |
| type | Field.Store.YES Field.Index.TOKENIZED |
| all | Field.Store.YES Field.Index.TOKENIZED |

表 4.3 中前三项是从 Product 类中获取的，分别表示产品的 id 号、产品的名称、产品的型号；由于索引并不利于结构化存储，所以为了减少存储空间，只

在索引库中存储了以上几项，其它的内容可以通过 id 在数据库中查询获得。最后的 all 则是对 name，type 和 content 三个 Field 进行合并构造的。这样做就可以使得用户每次搜索就可以直接在 all 这个 Filed 中进行检索，由于在这个字段中包含了手机的名称，型号和所有的参数，基本上是用户检索时可能用到的所有关键词，因此就可以避免系统对整个索引库进行多域检索，对索引库进行多域搜索的时候对系统整体性能来说是个损失，虽然通过构建一个额外的 Field 会损失一部分磁盘空间，但是相对于系统的整体性能而言，这一部分的损失是值得的。

本系统在对 Field 对象进行设计的时候，预留了以后对字段的扩展。对产品的类型和名称都建立了索引。本系统中，由于没有分为按类型还是按名称来进行检索，所以增加了一个默认检索域 all，在以后的系统功能扩充上，可以增加按相应的方式来进行检索。这就使得以后的扩展非常方便，当需要建立其它 Field 对象的时候，可以直接转化默认检索字段，可以进一步的提高搜索效率。

在 Lucene 框架之中，Field 对象主要有以下五种公有的构造函数：

(1)Public Field(String names,String values,Store stores,Index indexs,TermVetortermVetors)

(2)Public Field(String names,String values,Store stores,Index indexs)

(3)Public Field(String names,Reader readers);

(4)Public Field(String names,Reader readers,Store stores,TermVetor termVetors)

(5)Public Field(String names,byte[]values,Store stores)

本系统中需要建立索引的结构化的文本信息都存储在本地磁盘上，而且所有的文本信息的数量并不是特别巨大，因此在构建 Field 对象时本系统采用第二种构造方法。

buildDocument 方法首先是获得 Product 对象中如表 4.3 的参数，然后通过这些参数来建立对象的 Field 对象，然后 buildDocument 方法在通过调用 Document 提供的 add()方法，将四类 Field 按照定义好的存储方式和索引方式加入 Document，生成每一个 Product 的 Document。

(二).main 方法

main 方法的功能：main 方法将建好的 Document 加入索引。

main 方法的设计与实现：通过 Lucene 中的 IndexWriter 来将索引加入到索引库，IndexWriter 是在 Lucene 的 org.apache.Lucene.index 包中被实例化的。当 IndexWrite 被实例化之后，就可以对需要被建立索引的字段创建索引，添加 Document，并且可以对所建立的索引进行各种相关的操作。

IndexWriter 对象总共有三个公有的构造函数：

(1)public IndexWriter(String paths,Analyzer analyzer,Boolean boo)

(2)public IndexWriter(Directory directory,Analyzer analyzer,Boolean boo)

(3)public IndexWriter(File paths,Analyzer analyzer,Boolean boo)

Paths 指的是形成索引库之后需要被存放的绝对路径。Analyzer 指的是在建立索引的过程中所使用的分词器。Boo 表示是需要重新建立一个新的索引库还是打开一个已经创建好的索引库。其中需要注意的是 Analyzer 类是一个抽象类，在具体使用的时候需要使用一个已经实现其方法的具体类。如 MMAnalyzer，StandardAnalyzer 等。

Lucene 自带的分词器仅仅是通过空格等简单的方法来分词，因此对于中文的分词其显然不能满足要求，因此本系统采用了第三方的中文分词工具，JE 中文分词器，在使用 JE 中文分词之前，需要初始化一个 JE 分词对象，然后在将 IndexWriter 对象实例化的时候将它作为 Analyzer 的参数传入，就可以完成整个索引的初始化。

在将 IndexWriter 对象实例化之后，就可以形成一个索引库，并且像其中加入需要创建的索引，IndexWriter 类中有以下两种方式可以建立索引：

(1)public void addDocument(Document document,Analyzer analyzer)

(2)public void addDocument(Document document)

第一种方式需要传入一个 Analyzer 参数，因此可以使用一个第三方提供的分词器，而不是使用 Lucene 自带的 Analyzer，通过第三方的 Analyzer 来对需要建立索引的信息进行分词、过滤等操作，第二种方式可以让开发者直接将 Document 添加入索引，默认就使用了 Lucene 自带的分词器。一般来说建议是跟检索时使用同一个分词器，因为如果分词器不统一的话，会造成检索的时候同一

个短语被不同的分词器分成不同的词语，就会导致检索出错误的结果。

按照系统的需求，通过Hibernate中的session开启事务，然后查询出所有的Product对象，并通过调用addDocument()方法将索引加入索引库。在索引库形成之后，就可以通过id直接查询到数据库中产品的详细信息，进一步的提高了检索的效率。

### 4.4.3 索引的优化

在建立索引的时候，如果需要被建立索引的信息量巨大，在磁盘上所产生的索引文件就会非常之多，当对索引库进行检索的时候就会涉及到大量的磁盘I/O操作，而对磁盘进行I/O操作是十分耗费资源和时间的，所以通过对索引文件进行优化以提供效率就成了必不可少的步骤。

在Lucene自身的索引组件中，提供了相应的优化索引方式，如果不是性能要求极高，可以通过使用Lucene自带的优化方式，根据自身的机器性能来对其中的参数做出相应的修改，就可以达到优化索引的目的，本系统就是使用Lucene自带的优化方式来对索引进行优化。

当使用Lucene建立和读取索引的时候，为了提供其建立和读取的效率，Lucene会在开辟一块内存作为缓冲区，当需要建立索引的时候，每个Document对象都会先被读入到这个缓冲区，而当这个缓冲区的容量达到设定值的时候，Lucene就会将这些Document对象统一的写入磁盘文件当中，也就是会在磁盘上产生一个segment文件。而当磁盘上的segment文件的个数超过另一个设定值的时候，Lucene就会将磁盘上的所有的segment文件进行一次合并，同时在磁盘上生成一个合并后的segment文件，整个过程一直重复直到整个索引建立完成。

在整个优化的过程中主要有以下三个限定值在起作用：

(1)maxBufferedDocs：当Lucene在内存中开辟一个缓冲区的时候，需要对这个缓冲区设定一个限定值，这个参数就起限定缓冲区大小的作用。如果需要被建立索引的信息量巨大时，而同时开发者的电脑的内存也够大时，就可以加大maxBufferedDocs参数的值，这样就可以减少对磁盘上文件I/O操作。

(2)mergeFactor：该参数限定了内存中Document，同时也限定了在磁盘中segment文件的数量。当mergeFactor的值较大时，就表示在内存中可以一次性存放的Documnt就比较多，如此会占用较多的内存，但因为减少了磁盘的I/O操

作，所以建立索引的速度会较快。反之，如果 mergeFactor 的值较小的时候，就表示内存中可以一次性存放的 Document 较少，占用的内存也就会较小，但是就需要频繁的读取磁盘上的文件，所以建立索引的速度会较慢。因此，在实际建立索引的过程中，mergeFactor 的值应该根据不同的情况做出不同的改变，在 Lucene 中，mergeFactor 的默认值是 10。

(3)maxMergeDocs：该参数限定了在磁盘中一个 segment 文件的最大存放的 Document 的数量，如果 maxMergeDocs 参数设置的过大的时候，会使得最后合并的 segment 文件非常巨大，这显然是不合适的，而当 maxMergeDocs 参数设置过小时，就会频繁的进行合并操作，造成不必要的资源浪费，因此，maxMergeDocs 参数的取值大小，也应该根据实际情况来确定。在 Lucene 中，maxMergeDocs 的默认值为整型的最大值。

对于 mergeFactor，maxBufferedDocs，maxMergeDocs 三个参数各自具体应该取什么值，本系统根据测试电脑的情况，作出如下测试，首先测试电脑的配置如下：

Cpu：P4 2.0GHz

硬盘：250G

内存：2G

系统操作系统：Windows XP

对 6000 个文件来建立索引，通过改变三个参数的值，来计算每次改变所花费的时间，就可以看出对于测试电脑来说，三个参数的最佳取值是多少。测试结果如表 4.4 所示：

表 4.4 建立索引的测试结果

| mergeFactor | maxBufferedDocs | maxMergeDocs | 文档数量 | 时间(秒) |
|---|---|---|---|---|
| 10 | 10 | Integer.MAX_VALUE | 6000 | 186 |
| 50 | 50 | Integer.MAX_VALUE | 6000 | 162 |
| 150 | 150 | Integer.MAX_VALUE | 6000 | 149 |
| 500 | 500 | Integer.MAX_VALUE | 6000 | 141 |

从上表可以看出，随着 mergeFactor，maxMergeDocs 取值的逐渐增加，完成建立索引所花费的时间在逐渐减少，但是根据测试电脑的情况，可以发现其对内

存的需求也是不断增加，而当三个参数的取值达到500时，完成建立索引的时间的相差已经不是很大，但是对测试电脑的内存的需求还是持续的增大。所以本系统三个参数最终都取Lucene的默认值：

maxMergeDocs=Integer.MAX_VALUE

mergeFactor=10

maxBufferedDocs=10

### 4.4.4 中文分词

对于检索系统，需要建立索引的信息都不是一个一个词组，而是以一个一个文本出现的，因此就需要对这些文本进行处理，最后形成一个一个词组的形式存放在索引库中。在Lucene的org.apache.Lucene.analysis包中，有其自带的一些分词工具，主要包括 StandardFileter，SimpleFileter， Stopanalyzer，Whitespaceanalyzer等四种分词器。

Lucene在开发之初，仅仅只有对英文分词的支持，这样就使得Lucene在中文方面无法进行处理，随着Lucene的不断完善，在Lucene2.0之后，也对中文分词进行了支持。但是，其支持方法仅仅是简单的以空格等来对中文进行分词，而中文由于其词义的特殊性，仅仅如此来分词显然是不合理的。因此，虽然Lucene已经支持处理中文文本，但是在很多实际情况下并不能得到很好的结果。

由于Lucene对中文文本的处理并不是很好，而当需要构建的是中文检索系统的时候，显然会造成很大困难。

为了解决这一困难，在一些小型中文检索系统中，这写系统都使用针对自身情况开发的分词系统。这种分词系统也仅仅能满足本系统的需求，同时对不断更新的词汇的识别能力也很差。因此，针对这一情况，本系统通过采用第三方的分词系统JE分词工具来取代Lucene自身的Analyzer对中文文本进行分词。

JE中文分析器是一套由Java编写的中文分词系统，使用该分词器的时候无需安装任何插件，可以直接取代Lucene自身的分词器来使用。JE中文分词器内部主要是采用正向最大匹配算法，支持英文、数字、中文等混合分词，通过使用JE中文分析器可以很好的处理中文分词问题。

Lucene分词器与JE分词器对“诺基亚摩托罗拉三星”进行分词效果比较，结果如表4.5所示：

表4.5 不同分词器的分词结果

| 分词器 | 分词结果 |
| --- | --- |
| Whitespaceanalyzer | 诺基亚摩托罗拉三星 |
| Stopanalyzer | 诺基亚摩托罗拉三星 |
| SimpleFileter | 诺基亚摩托罗拉三星 |
| StandardFileter | 诺\|基\|亚\|摩\|托\|罗\|拉\|三\|星 |
| JE分词器 | 诺基亚\|摩托罗拉\|三星 |

通过表4.5可以看出，相对于Lucene自身的分词系统，JE分词系统在对中文分词中无论是成词效率还是准确率都要高，并且JE分词器还提供了词库的扩展接口，使得开发者可以根据自己的需要来添加自己需要的词语。因此本系统使用JE分词器作为分词系统。

## 4.5 检索模块的设计与实现

### 4.5.1 MVC模式

MVC是Model-View-Controller的缩写，中文翻译为“模型-视图-控制器”。其中，Model就是业务流程/状态的处理以及业务规则的制定，通常以一个实体bean来实现。View代表用户交互界面，对于Web应用来说，通常采用Jsp来输出HTML。Contrller可以理解为从用户接收请求，将模型与视图匹配在一起，共同完成用户的请求，通常由Servlet程序实现。MVC模式架构如图4.16所示。

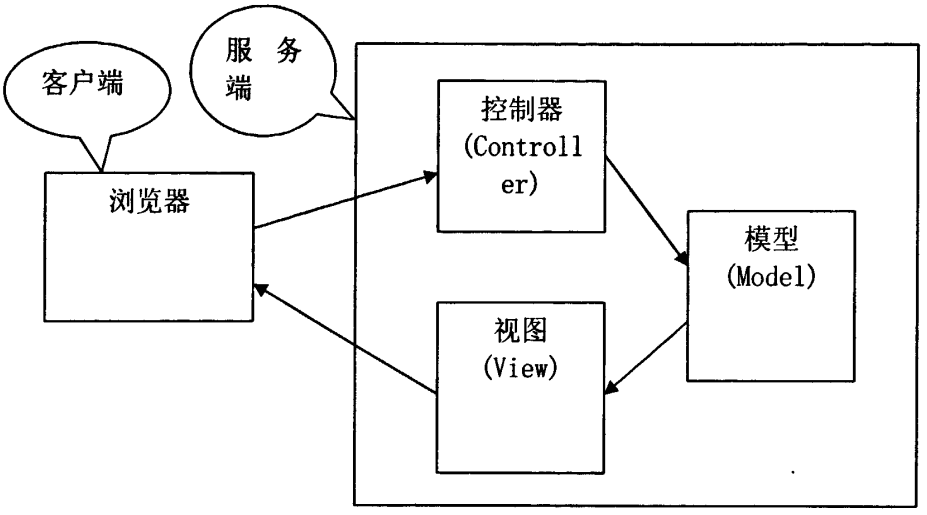


图 4.16 MVC 结构图

通过这种设计模型，业务逻辑、视图逻辑和实体模型被分成三个不同的模块，使得三个模块的功能被严格分离，每个模块的设计和实现都不依赖于其它模块，也不会对其它模块产生影响。通过 MVC 设计模型开发的应用程序有以下三个优点：

(1)开发流程更为明确：通过使用 MVC 的设计模式使得模型层，视图层和业务逻辑层完全分离，可以让美工人员和程序员各自从事自己的模块，有利于系统的维护和开发。

(2)程序管理控制更为方便：由中央控制器控制整个程序的走向，可以将业务逻辑的代码从 Jsp 页面中抽取出来处理，从而使得视图只需要显示页面而不需要处理复杂的业务逻辑。

(3)维护容易：不论是后台的业务逻辑或者是前端的网页呈现，都是通过控制层来处理，如果整个系统的业务逻辑发生变化，只需要修改模型层的代码，而不用去修改视图层的代码。

### 4.5.2 基于 MVC 模式的查询模块的设计与实现

查询模块主要是通过查询用户输入的关键词，然后根据查询的结构返回到视图层，因此本系统的开发是采用 MVC 架构，同时整个系统是置于 J2EE 框架下来实现的。整个设计结构如图 4.17 所示。

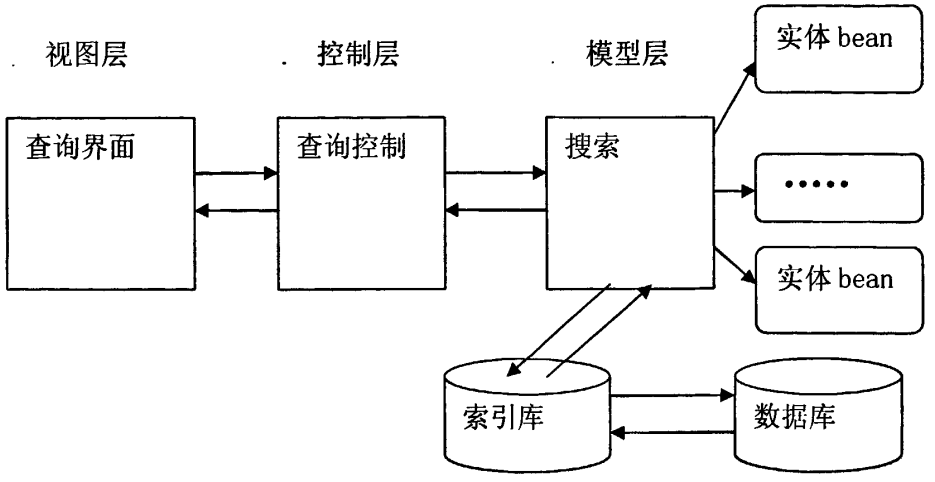


图 4.17 设计结构

(一). 视图层

视图层是通过文件 login.jsp 来实现的，主要功能是提供了用户的查询界面。当用户输入关键词查询的时候，查询出的结果会在该页面直接呈现给用户。程序核心代码如下所示。

```
 .....
<c:if test="${!empty queryStr}">
          <input size="50" type="text" name="query" id="query"
value="${queryStr}" >
          </c:if>
          <c:if test="${empty queryStr}">
          <input size="50" type="text" name="query" id="query"
value="诺基亚" >
          </c:if>
          <input type="button" value=" 搜 索 " id="search"
onClick="javascript:doSearch('')">
 .......
<c:forEach items="${results}" var="result">
<div id="result">
   <table border='0' cellpadding='0' cellspacing='0' width=700>
    ......
   </table>
</div>
</c:forEach>
 .......
```

首先通过El表达式来获取在控制层的数据，然后通过Jstl标签语言将获得的数据通过Jsp页面呈现给用户。

(二).控制层

当控制层接受到用户的请求时，控制层会根据用户的请求到Struts的配置文件Struts-config.xml中找到相应的处理方式，本系统中通过Struts会找到Spring配置好的实体bean来进行处理。相关的主要配置如下。

```
    ......
        <action path="/search"
              name=""
              scope="request"

    type="org.springframework.web.struts.DelegatingActionProxy"
              parameter="method"
        >
        <forward name="search_sucess" path="/login.jsp"/>
        </action>
    ......
<bean    name="/search"    class="com.skally.actions.ProductAction"
scope="prototype">
        <property name="productManager" ref="productManagerImpl"/>
    </bean>
<bean                                          id="productManagerImpl"
class="com.skally.ManagerImpl.ProductManagerImpl">
        <property name="sessionFactory" ref="sessionFactory"/>
    </bean>
 .......
```

跟据以上的配置，控制器会找到类ProductManagerImpl来处理用户的检索请求，主要的检索代码如下。

```
        .....
        //获取查询关键字
           String          queryStr          =          new
           String(request.getParameter("query").getBytes("ISO8859
           -1"),"gb2312");
        //获取页数
            String          startindexStr          =          new
         String(request.getParameter("startindex").getBytes("ISO8
         859-1"),"gb2312");
         //封装页数的类 page
        Page page = new Page();
        //JE 分词器
        MMAnalyzer analyzer = new MMAnalyzer();
        //默认查询 field
        QueryParser qp = new QueryParser("all",analyzer);
        //查询关键字
        Query query = qp.parse(queryStr);
        ......
        IndexSearcher          searcher          =          new
IndexSearcher(INDEX_STORE_PATH);
        Hits hits = searcher.search(query);
        ......
        for (int i = startindex; i <= endindex; i++) {
               Document doc = hits.doc(i-1);
               String id = doc.get("productid");
               Product                    pro                    =
               productManager.getSearchResultById(Integer.parseI
               nt(id));
               list.add(pro);
        }
        .....
```

系统检索功能通过Lucene中提供的检索接口实现，对于用户输入的关键次，首先会将该关键词构造成Query对象，然后将该对象传递给IndexSearcher类，IndexSearcher类会到对应的索引库进行检索。Lucene提供的Query有以下几类：

(1)TermQuery

TermQuery是最简单、也是最常用的Query。TermQuery可以理解为“词条搜索”，它是指通过用户指定的词条，查找索引中所有存在该词条的文档。在Lucene中词条是最基本的搜索单位，从本质上来讲一个词条就代表一个名/值对。其中的“名”表示的是字段名，而“值”则表示的是字段中所包含的某个关键字。

在使用TermQuery之前，需要先生成一个Term对象，代码如下：

```
Term term = new Term("search", "nokia");
```

Term生成之后，再将其作为参数传入TermQuery对象，代码如下：

```
Query query = new TermQuery(term);
```

这样所有在“search”字段中并且包含有“nokia”的文档都会被作为结果返回。

(2)BooleanQuery

BooleanQuery也是实际开发过程中经常使用的一种Query。它其实是一个组合的Query，在使用时可以把各种Query对象添加进去并标明它们之间的逻辑关系。实现方式为：

```
BooleanQuery bquery=new BooleanQuery()
bquery.add(Term t, BooleanClause.Occur)
```

在BooleanQuery的add(Term t, BooleanClause.Occur)方法中，第二个参数是BooleanClause.Occur，该参数主要有以下三种方式来表示：

- BooleanClause.Occur.Must_NOT
- BooleanClause.Occur.Must
- BooleanClause.Occur.SHOULD

它们在实际使用过程中，通过组合使用，可以表示词条之间的关系是“与”或者“或”。

(3) WildcardQuery

WildcardQuery可以实现通配符的检索。如果开发者根据用户的检索关键字需要进行模糊查询时，就可以使用通配符来进行检索。“?”号代表任何单个字符，“*”号可代表文件或文件夹名称中的一个或多个字符。具体的实现方式为：

```
WildcardQuery query=new WildcardQuery(Term t)
```

WildcardQuery 首先需要一个含有通配符的检索关键字，然后将它包装为 Term 对象，最后再通过该term构造出WildcardQuery。

虽然通过Lucene的各种Query子类的构造函数API可以很方便的创建Query对象，但是如果用户所有的查询都使用构造函数，显式地编写出来构建查询显然是不合理的。例如用户输入的是个形如“诺基亚 n97”而不是“诺基亚”的时候。Lucene提供的QueryParser类不仅可以创建前面介绍过的某一个Query子类的

对象，也可以使用自然语言表示的查询表达式。其具体的实现方式为：

```
QueryParser parser=new QueryParser(search,analysis);
Query q=parser.parse(all)
```

QueryParser 中的第一个参数指的是查询的 Field，第二个参数是一个分词器，QueryParser 对象构建好之后，就可以通过指定查询短语来返回一个 Query。

由于本系统中用户使用的多为短语检索，所以使用 QueryParser 来进行查找，返回结果都在类 Hits 中，Hits 类主要是获取查询之后返回的结果，该类位于 Lucene 的 org.apache.Lucene.search 包中。

(三).模型层

本系统中定义了二个实体类，分别为 Product 和 Page，Product 封装了手机产品的全部信息，与数据库形成对象关系映射，Page 则封装了页面的分页信息。主要代码如下。

```
    public class Page {
       //当前页的最开始
       private int startIndex;
       //当前分页栏的开始页
       private int startPage;
       //当前分页栏的结束页
       private int endPage;
       //是否还有下一页
       private int hasNext;
     .....
}
```

同时需要生成对应的 set，get 方法。通过控制层的对页面的分页进行处理，最终显示在 Jsp 页面上。

最后，还需要修改后台服务器 Tomcat 的配置文件 Web.xml，使得使得 Tomcat 能够找到 Struts 的中央控制器，以便分发到对应的 Action 中进行处理，配置完之后整个系统就完成了，Web.xml 的主要配置如下。

```
....
  <welcome-file-list>
    <welcome-file>login.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>action</servlet-name>
<servlet-class>org.apache.struts.action.ActionServlet</servlet-cla
ss>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
  </servlet>
.....
```

此时，我们可以通过MyEclipse来启动Tomcat，当Tomcat正常启动之后，就可以打开浏览器窗口然后输入网址：http://127.0.0.1:8080/MVCLucene/，就可以看到程序的查询界面了。程序执行结果如图4.18。

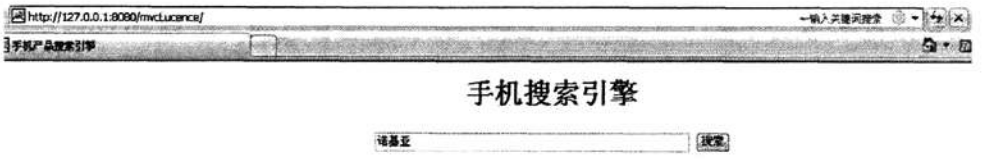


图4.18 手机查询界面

例如，如果想要查询手机“诺基亚n78”的全部信息时候，可以输入全名“诺基亚n78”，也可以输入“n78”进行查询，显示的结果如图4.19所示。

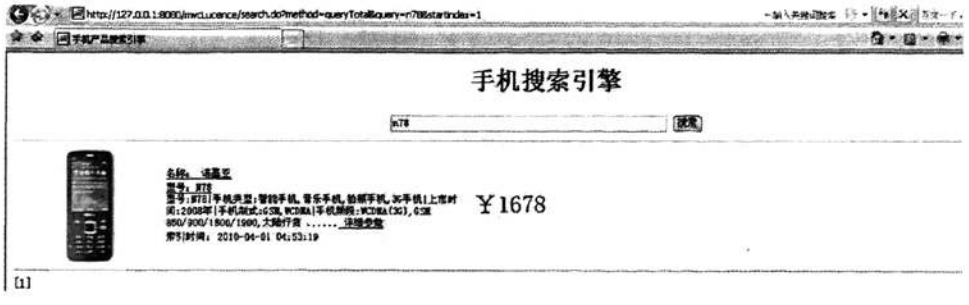


图4.19 手机显示结果界面

查询“诺基亚”的全部信息，从中就可以看到所有的诺基亚手机信息，同时可以看见分页效果如图 4.20 所示。


图 4.20 分页界面

## 4.6 本章小结

本章首先针对 Heritrix 进行了扩展，使得爬虫能够尽可能的在相关领域内进行信息的采集。然后介绍了使用 HtmlParser 工具来对 HTML 网页的信息进行抽取，能够使得开发效率大大提升，在信息抽取完成之后，再通过数据库和 Lucene 工具来对抽取出的信息建立索引，最后再介绍了 MVC 的设计模式，利用 J2EE 框架来实现了整个系统的检索。

# 第 5 章 总结与展望

随着计算机技术和互联网技术的快速发展，网络上的信息量也急剧增加。如今，搜索引擎越来越像专业化和领域化发展。本文通过对 Lucene 框架以及 Heritrix 框架的学习研究，构建了一个能够针对手机产品信息进行检索的垂直搜索系统。具体而言，本文主要完成了下列工作：

(1)对垂直搜索引擎的总体架构进行分析，同时介绍其整个的工作流程。

(2)在信息采集策略方面分析了基于主题相关度的采集策略的核心思想，指出了该采集策略的不足并提出了构建基于本体知识库的采集策略的实现思路。

(3)在对网络爬虫工具 Heritrix 深入分析研究的基础上对其模块的功能进行相应的扩展，以达到对网页信息下载的可控性。

(4) 本文对 Lucene 索引框架进行了研究学习，分析了其主要架构和各个部件的功能，然后将之应用于检索模块当中。

(5)通过运用 Hertrix 开源爬虫和 Lucene 开源框架设计并构建一个面向手机产品信息的垂直搜索引擎，主要实现过程为：

- 通过扩展 Heritrix 相应模块实现了对网页的选择性下载。
- 通过 HtmlParserr 工具从网页信息中抽取出结构化文本信息。
- 利用 Lucene 框架对相关的信息建立了索引。
- 设计并实现了基于 MVC 架构的查询接口。

本人由于涉足搜索领域的时间较短，对于文中所构建的系统还有很多需要改进和完善的地方，下一步将要做的工作如下：

(1)研究并跟踪本体技术的发展，将本体推理技术用于搜索系统，以实现搜索的智能化。

(2)对于本系统而言，由于只对单个网站的信息进行了搜集，所以产品的信息过于单一，此后将对多个网站的信息进行收集并整理。

(3) 由于本文在使用 HtmlParserr 工具对页面信息进行抽取的时候，对所下载的 URL 页面自身的结构有很大的依赖性，所以在对页面信息抽取的方面需要进

一步研究，希望能够找到一个跟页面结构无关的信息提取算法。

# 参考文献

[1]Soumen Chakrabarti, Martin van den Berg, Byron Dom, Focused erawling:a new approach to topic specific web resource discovery[J], Computer Networks, 1999, 31(11-16):1623-1640.

[2]Lawrence S, Giles C L.Digital Libraries and Autonomous Citation Indexing[J], IEEE Computer, 1999, 32(6):71-87.

[3]Neehes R, Fikes R E, Gruber T R, et al.EnablingTechnology for Knowledge Sharing[J].AI Magazine, 1991, 12(56):80-91.

[4]Studer R, Benjamins V R, Fensel D.Knowledge Engineering, Principles and Methods[J].Data and Knowledgeing.1998, 25(122):16-197.

[5]Dejan Milojicic.Mobile agent application from Trend Wars[J].IEEE Concurrency.2000, 36(11):12-16.

[6]肖冬梅，垂直搜索引擎研究[J]，图书馆学研究，2003，20(6):87-128.

[7]余森，杨丹，赵俊芹，垂直搜索引擎的关键技术研究[J]，软件导刊，2007，17(12):18-30.

[8]李向阳，苗壮.自由文本信息抽取技术[J].情报科学,2004,22(7):815—821.

[9]化柏林，搜索引擎面面观[J]，中国计算机用户，2004，12(26):60-93.

[10]吕昊，面向垂直搜索的聚焦爬虫研究及应用[D]，浙江大学硕士学位论文，2008.

[11]刘畅，综合搜索引擎与垂直搜索引擎的比较研究[J]，情报科学，2007，29(23):41-80.

[12]刘迁，贾惠波，中文信息处理中自动分词技术的研究与展望[J]，计算机工程与应用，2006，36(18):35-78.

[13]李晓明，张岩搜索引擎技术及发展趋势[C]，中国计算机科学技术发展报，2006.

[14]CNNIC，中国互联网络发展状况统计报告[R]，北京中国互联网信息中心，2006.

[15]彭哲，陈敬文，基于Lucene/XML全文检索的应用及效率测试研究[J]，情报杂志，2009，15(25):15-58.

[16]梁卓明，陈炬桦，基于专有名次优先的快速中文分词[J]，计算机技术与发展，2008，5(36):58-97.

[17]Gruber T R.A Translation Approach to Portable Ontology Specifications，Knowl-edge Acquisition，1993，25(5):199-280.

[18]Borst W N.Construction of Engineering Ontologies for Knowledge Sharing and Reuse.PhD thesis.University Twente，Enschede，1997:67-72.

[19]Hatcher Erik,Otis Gospodnetic.Lucene in Action[M].Greenwich:Manning PublicationsCo,2005，9(10):7-10.

[20]周像金，综合风险垂直搜索引擎的研究与实现[D]，西北大学硕士学位论文，2007.

[21]彭涛，面向专业搜索引擎的主题爬行技术研究[D]，吉林大学博士学位论文，2007.

[22]孔楠，基于本体的垂直搜索系统的设计与实现[D]，北京交通大学硕士学位论文，2009

[23]刘琳娜，薛建武，汪小梅.领域本体构建方法的研究[J].情报杂志 2007，12(4):14-15.

[24]李刚,宋伟,邱哲.征服Ajax+Lucene构建搜索引擎[M].北京：人民邮电出版社,2006.

[25]朱学昊，王儒敬，余锋林，等.基于Lucene的站内搜索设计与实现[J].计算机应用与软件，2008，25(10)：6-8.

[26]张贤,周娅.基于 Lucene 网页排序算法的改进[J].计算机系统应用，2009(2)：155-158.

[27]李沛环.基于Lucene的搜索引擎的设计和优化[D]：吉林大学硕士学位论文.2008.

[28]郎小伟,王申康.基于 Lucene 的全文检索系统研究与开发[J].计算机工程,2006,32(4)：94-96.

[29]吴青,夏红霞,赵广辉,等.基于Lucene全文检索引擎的应用与改进[J].武汉

理工大学学报，2008，30(7)：145-148.

# 致 谢

在此论文完成之际，我要衷心地感谢所有在我的读研期间关心过我、帮助过我的人们。

在这里首先要感谢我的导师李中华老师。从我的论文选题开始直至结束，老师一直都给予了我耐心的指导，并对我的研究工作提出中肯、有价值的意见和建议，使我能取得现在的进步和成绩。

其次，在本文的写作过程中，也得到了实验室的同学以及班上同学的热情帮助和支持，感谢他们，感谢他们在我进行项目开发的过程中给予了我极大的技术支持。

再次发自肺腑地、由衷地感谢关心和帮助过我的每一个人，祝他们身体健康、天天开心、事事顺心。

# 垂直搜索引擎的设计开发

作者：　　刘琦
学位授予单位：　　中山大学



## 相似文献(2条)

1.学位论文 蓝永健 农业垂直搜索引擎的研究与设计 2009

随着互联网技术的飞速发展，互联网络上的信息量正在以几何级数的增长速度增长，因此，对网络上信息的高效检索成为互联网发展必须要解决的问题，搜索引擎技术得到了特别的重视并且正在飞速的发展。但是，当使用综合型搜索引擎来检索专业内容时，由于其结果存在范围性广、搜索目的不准确的缺点，往往找不到用户们需要的内容。随着电子商务和农业信息化的发展，网络中的农业数据开始高速增长，农业用户使用网络获取相关数据和资源也日益普及，增强了农业用户对信息搜索的依赖性。大部份农业信息网站的“网站内”数据库搜索模式不能满足人们对农业数据搜索越来越多的要求。<br>

本文以用户对农业信息搜索需求为研究背景，在我国农业信息化日益发展之际，结合现今流行的垂直搜索理念和技术，以国内互联网上的农业信息电子商务网站和政府农业信息网站的农业信息为特定的抓取对象，进行自动采集，来实现农业信息的搜索功能。为了实现农业信息的全文检索，引入Lucene全文搜索引擎来实现系统的全文搜索功能。<br>

本文首先阐述了搜索引擎的基本原理以及垂直搜索的关键技术，然后对基于JAVA的Lucene全文搜索引擎技术进行必要的说明，最后在对农业信息搜索业务分析的基础上，实现系统的基本功能需求和最终目标。<br>

在实现系统功能需求和目标的基础上，运用UML分析设计技术和系统三层架构思想，对系统的功能设计和数据库设计进行了详细论述。<br>

本系统在Windows+ Microsoft SQL Server2005+IIS+ASPX开发环境下，设计开发了农业信息搜索网站系统，在Windows+ Eclipse+JAVA+Microsoft soLServer2005开发环境下，设计开发了搜索系统的后台管理系统。因此，本文还对Ajax，eUML建模工具和.NET Framework框架进行了必要的说明。在此基础上，对系统各部分的功能实现进行了简要说明。

2.学位论文 朱春雷 基于Ajax技术的搜索前关键词提示机制的设计与实现 2007

本文围绕着搜索前关键词提示功能的核心-Aiax技术展开论述，集中说明了我就此项技术展开的研究、设计开发并最终实现了该功能的一系列工作。

Google中国在2007年1月23日正式推出了Google Suggest的中文版，它提供了一种搜索前关键词提示的服务。当用户在搜索框中输入要搜索的内容时，将实时的在搜索框下显示相关的搜索关键词提示。随着用户输入的不同给出的提示也不同，同时还会显示这些关键词将会得到的搜索结果数目。笔者在参与开发和慧垂直搜索引擎中，为了改善用户的使用体验和使用效率，决定采用Ajax技术来开发类似Google Suggest的搜索前关键词提示功能。

搜索前关键词提示功能是指在用户输入要搜索的关键词时，系统通过提供关键词的多个提示选项来帮助用户节省时间。用户只需要输入少量文字，对自己中意的关键词就可能从推荐列表中获得。此功能是使用Ajax技术使客户端与服务器之间进行交互，其间不需要打断用户与Web页面间的交互，使用户可以获得良好的使用体验和效率。

本文在介绍了Ajax的四个技术基石之后，详细介绍了搜索前关键词提示功能的设计与实现过程。在设计部分主要包括基本原理的分析、客户端的架构和服务器端的架构，在实现部分主要包括Ajax基础功能的开发、客户端以及服务器程序的开发。

由于和慧企业网站受资金、硬件等各种因素的制约，没有理想的负载均衡设备和多台服务器用来处理请求，因此在应用搜索前关键词提示功能时要尽量限制其对于服务器的影响。至于改善响应能力的策略，则是将数据缓存在客户端，仅仅在需要新的数据时才会访问服务器。通常当输入更多的字符时，搜索前关键词提示系统返回的结果是以前结果的子集，因此可采用的解决方案是保留初始请求的结果，根据用户逐渐输入的内容来去除不匹配的数据项，而不是每次都通过访问服务器去获取数据，同时为每次的键入刷新用户界面。这种解决方