<HoneyComb >

软件架构文档

版本 <1.0>

[

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <日/月/年> | <x.x> | <详细信息> | <姓名> |
| 2018.9.6 | 0.1 | 初步数据整理和文档编写 | 曹金坤 |
| 2018.9.7 | 0.2 | 添加算法设计部分 | 马轲，谢添翼 |
| 2018.9.7 | 0.3 | 添加进程视图部分 | 曹金坤 |
| 2018.9.8 | 0.5 | 添加数据视图和实现视图部分 | 曹金坤 |
| 2018.9.8 | 1.0 | 添加用例视图、逻辑视图、部署视图 | 马轲，谢添翼 |

目录

1. 简介 2

1.1 目的 2

1.2 参考资料 2

2. 用例视图 2

3. 逻辑视图 2

3.1 概述 2

3.2 在构架方面具有重要意义的设计包 2

4. 进程视图 2

5. 部署视图 2

6. 实现视图 2

7. 数据视图（可选） 2

8. 核心算法设计（可选） 2

软件架构文档

## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要设计和决策。对于系统架构的描述，会从不同的角度进行说明。

## 参考资料

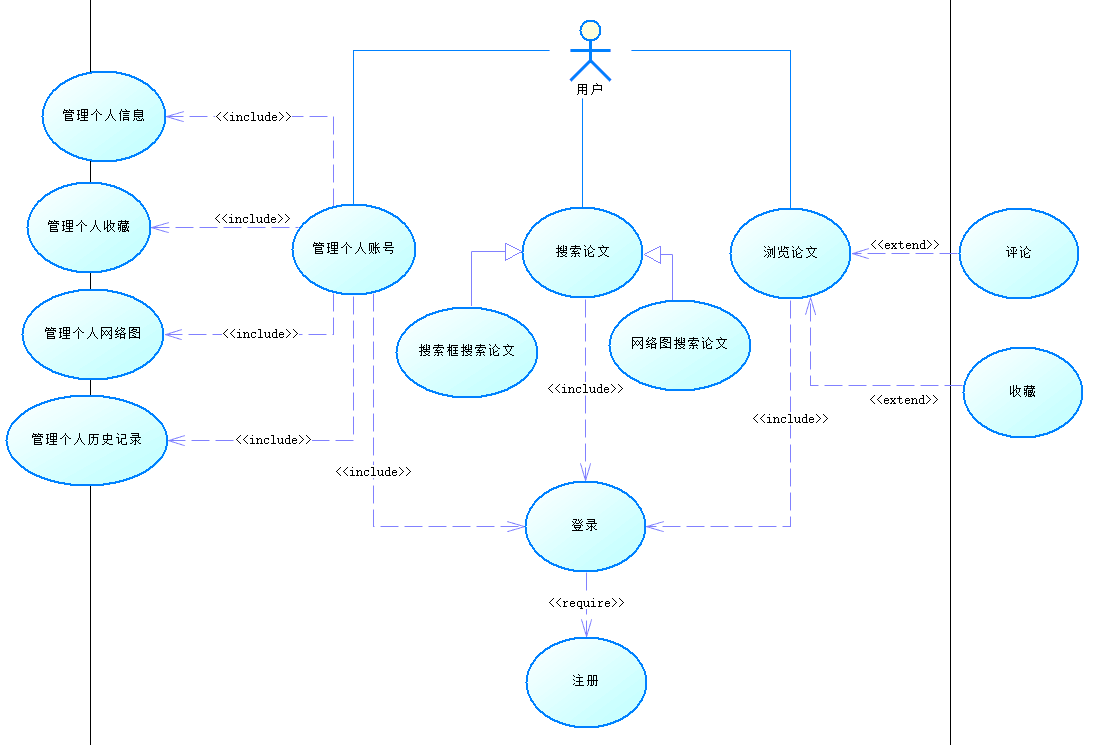
《HoneyComb项目立项建议书》*,2018.6.20,Honeycomb小组*

《HoneyComb需求规约》*,2018.6.20,Honeycomb小组*

# 用例视图

软件系统的用例大概可以分为以下几个部分（依据立项时的需求设计）：

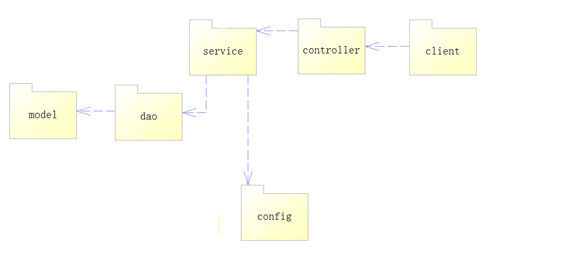
1. 个人信息管理：包括管理个人账号信息、管理个人收藏内容和个人的定义网络等，注册以及登录等；
2. 论文相关用例：包括对论文进行浏览和收藏等；
3. 信息检索用例：包括对论文依照文本信息进行检索以及在可视化的网络图中进行查找等；



# 逻辑视图

## 概述

系统可以分解为客户端与服务端，服务端可进一步分解为接口层、业务层、数据层等。接口层即controller包，服务层即service包，数据层即dao包与model包。 config包作为辅助工具存在。



## 在构架方面具有重要意义的设计包

* + 1. **config包**

config包储存系统配置，包括session相关配置、mongodb相关配置、mysql相关配置等。

* + 1. **model包**

model包由OR映射的entity类组成。

* + 1. **dao包**

dao包数据库访问层，负责与数据库交互。每一个OR映射类，均有相应的repository类进行访问。

* + 1. **service包**

由系统功能抽象而得的service类组成，实现业务逻辑。

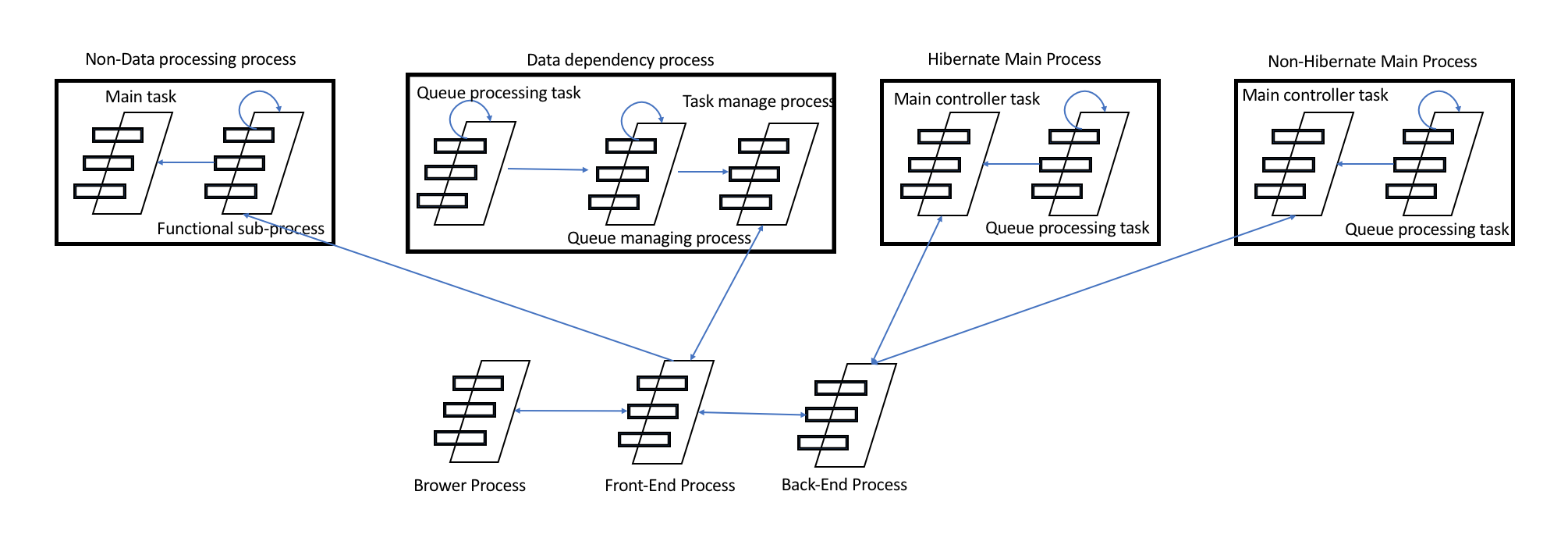
* + 1. **controller包**

controller包由MVC模式中controller类组成，响应用户请求。针对Web端有相应API接口。

* + 1. **client子系统**

client子系统指website的front-end。

# 进程视图



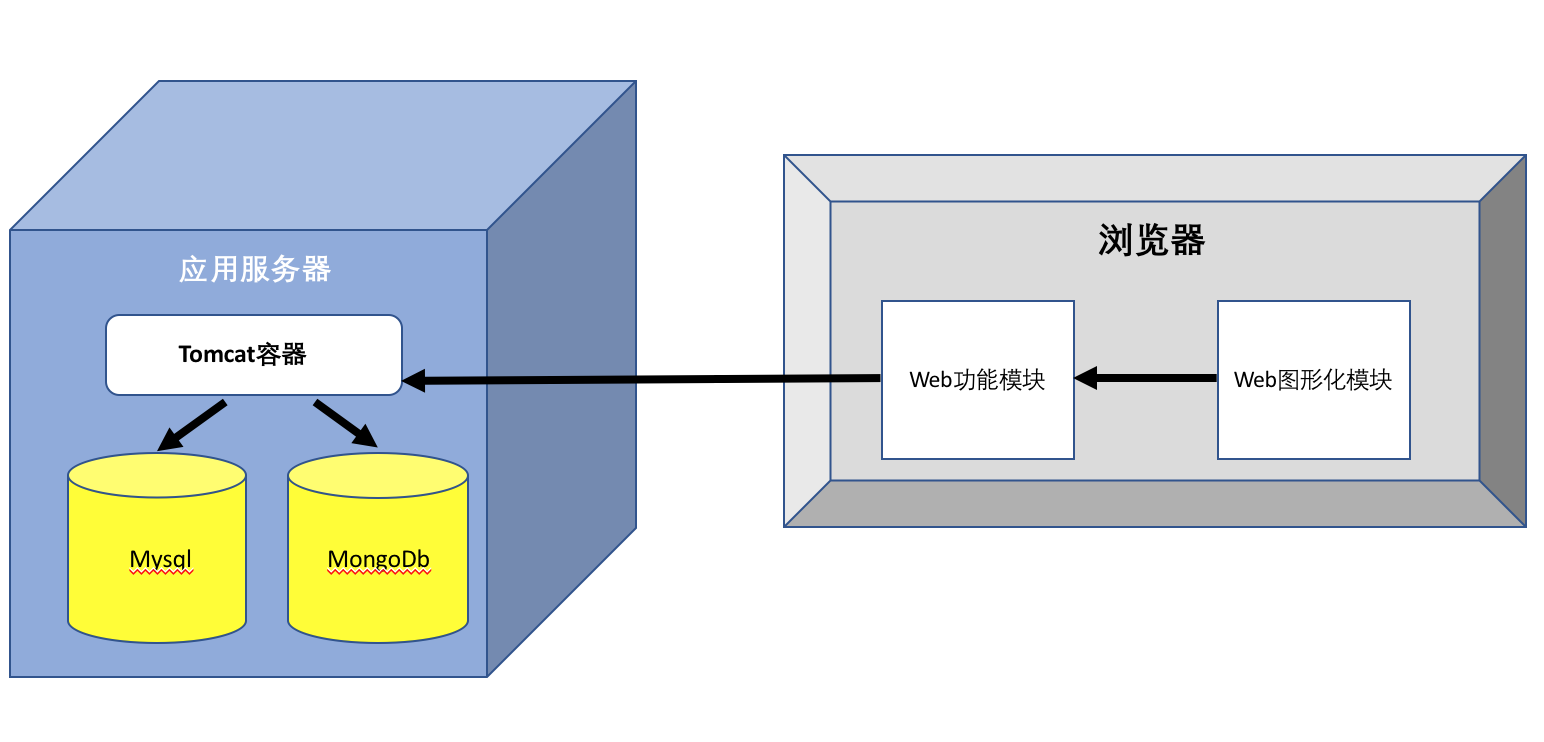
功能进程视图示意图

项目的进程设置以三层逻辑为主，体现在两个分离的前后端工程中，三层逻辑分别是：

1. 面向用户的浏览器层对应的以可视化和交互响应为主要工作的进程；
2. 链接后端和浏览器层的前端工程衍生出来的进程组，主要包括两类重进程组，一类是需要对后端发送请求的进程，这类进程中依赖axios进行前后端的通信，另一类是不对后端有请求的纯前端进程，此类进程只需要进行基于Vue的处理即可
3. 后端工程衍生出来的进程类型也有两类，一类是基于Hibernate的对接关系型数据库Mysql的进程组，基于Struts定义的类controller模式；并在进程组中不断使用子进程进行数据处理；另一类是对接Mongodb的进程组，前端接口的controller定义依旧在struts中完成，但是进行工作时，对子进程的调度和Hibernate不同。

# 部署视图

工程的部署视图如下，在后端的机器上，运行着Tomcat容器，他接受了前端发来的对于后端的请求并且链接了mysql和mongodb两个数据库，在前端的一侧，浏览器上与用户直接交互部分包括功能性模块和单纯的展示部分，前者和后端的应用服务器协作，构成了系统的功能主体。



# 实现视图

软件的实现层次中，主要包括了若干重要的子系统和层结构，包括：

1. 数据层：保存了Section 7中描述的所有永久性数据和在使用软件过程中产生的非永久性的持续性和非持续性数据，对于永久性数据的存储采用Mysql和Mongodb两套解决方案并用的设计方法，满足对于结构化数据和非结构化数据的不同需求；
2. 算法层：在系统功能的实现过程中，此层的算法层面承担了从存储的原始数据中分析和提取信息，该层由后端程序中实现，算法的部分主要设计对于数据层保持的数据的操作，主要是查找、比对和返回操作，其中最重要的自实现算法包括比对文本相似度的TF-IDF算法和支持论文网络图构建的MapReduce算法；
3. Hibertnate数据操作系统：基于Hibernate实现，对Mysql数据库进行操作，通过在struts中定义的接口函数，返回需要的结果，一般返回形式为json；
4. Mongodb数据操作系统：对Mongodb数据库进行操作，返回json类型的结果数据；
5. “社区”模块：系统中用于用户分享和交流的子系统，主要的功能涵盖创建板块、发帖、跟帖、添加评论等。该模块中还集成了其他粒度更小的模块，包括编辑器模块、帖子模块等；
6. 论文网络图模块：系统的主体功能部分，主要成分包括搜索论文后扩散开形成的网络图、通过网络图的节点进行论文简介查看、选择变动网络的规模等等；
7. 其他的前端模块：其他的不独立提供主页入口的小模块，包括用户的登录注册模块、网站使用的指导页面等等。

# 数据视图（可选）

系统架构中，涉及到的主要永久性数据有：

1. 论文数据

作为Honeycomb项目的主要功能，对于学术论文数据的查找、浏览和可视化都依赖于我们小组事先爬取的公开论文数据。数据来自于对IEEE Xplore网站论文数据的爬取，总数量超过30W条，数据的内容包括：

1. 论文标题
2. 论文作者名单
3. 论文发布的会议/期刊/网站
4. 论文发表年份
5. 论文的简述(Abstract)
6. 论文引用的其他论文

论文数据存储在Mysql数据库中，因为要表示论文之间的“引用”关系、论文和作者之间的“著作”关系，还产生了额外的数据表。

1. 用户登录数据

用户在注册之后，即开始产生永久性数据内容。相关数据在用户自己进行修改的时候才会发生改变，具体的数据条目有：

1. 用户名
2. 用户登录密码
3. 用户个人简介(bio)
4. 用户自定义数据

用户在登录之后，可以在自己的个人空间之中，定制自己的论文收藏和关系网络图，对应的数据为：

1. “社区”板块的后台存储数据

为了支持用户交流而设置的“社区”板块中，用户可以发帖(panel)，可以在别人创建的帖子中跟帖(post)，也可以评论别人发布的内容乃至于评论别人的评论内容，由此产生了非形式化的大量数据，由Mongodb数据库存储，包括：

1. 发帖板块(panel)，包括创建panel的用户ID、时间、标题、内容、关联的评论或者帖子等
2. 帖子(post)，包括发帖人的ID、时间、内容、评论内容和关联的panel信息等
3. 评论(comment)，包括评论人信息、时间、内容、关联的评论或者帖子信息等

# 核心算法设计（可选）

在系统中，主要有两个核心算法参与了主要功能部分，分别是实现搜索关联度匹配的TF-IDF算法和实现数据相似度排序的MapReduce算法。

1. TF-IDF算法：

TFIDF是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。TFIDF实际上是：TF \* IDF，TF词频(Term Frequency)，IDF逆向文件频率(Inverse Document Frequency)。TF表示词条在文档d中出现的频率。

https://gss1.bdstatic.com/9vo3dSag_xI4khGkpoWK1HF6hhy/baike/w%3D268/sign=c852fb9b57fbb2fb342b5f1477482043/5ab5c9ea15ce36d3448570f638f33a87e850b177.jpg

IDF的主要思想是：如果包含词条t的文档越少，也就是n越小，IDF越大，则说明词条t具有很好的类别区分能力。

https://gss0.bdstatic.com/94o3dSag_xI4khGkpoWK1HF6hhy/baike/w%3D268/sign=52c075efe9c4b7453494b010f7fe1e78/a8014c086e061d9552eec4fe79f40ad163d9ca48.jpg

其中，|D|为语料库中的文件总数。

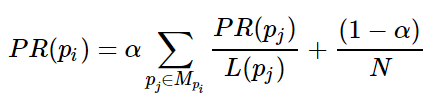
如果某一类文档C中包含词条t的文档数为m，而其它类包含t的文档总数为k，显然所有包含t的文档数n=m+k，当m大的时候，n也大，按照IDF公式得到的IDF的值会小，就说明该词条t类别区分能力不强。

1. MapReduce算法：

PageRank (简称PR)是一种数学排名算法，常常用于评估网页连接的质量，此评估有助于确定页面重要性和权威的相对分数。在本项目中，我们想要去恰当地衡量每个论文的价值借此挑选出那些值得我们去阅读的好论文而避免在水论文上浪费不必要的时间，我们利用上述的PageRank算法来达成我们这一目标，算法原理首先从以下朴素的想法出发：

1. 当一篇论文被许多论文所引用时，这个论文的价值含量越高
2. 一篇价值含量越高的论文，它所引用的论文的价值含量也会越高，同时论文的作者的知名程度也会越高

PageRank可以建立相应的数学模型来解决这一关系，我们可以先建立一个初步的网络图，网络图中的节点是论文与读者，网络图中的连接（含箭头）代表论文的引用关系，以及撰写关系。

初步的网络关系图确定后，我们为每一个节点赋予相同的起始权重，再根据PageRank来调整不同节点的权重。根据前述原理，定义一个节点的权重值（PR值）如下：

其中PR(pi)是节点的权重值，Mpi是所有引用论文pi的论文集合，L(pj)是论文pj的总引用和总撰写数量之和，N是节点总数，α一般取0.85，α的设置是为了防止数据中的一些论文没有引用对象导致权重值（PR值）无法收敛。

采用以上公式进行权重值的迭代，在不断迭代每个节点的权重值平稳的时候，我们即可获得在已有节点基础上的论文网络图，论文的价值量即可在节点的权重值中体现出来