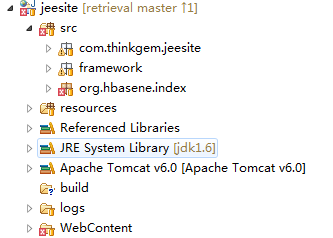
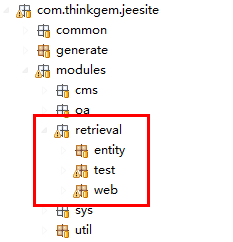
**全文检索项目**

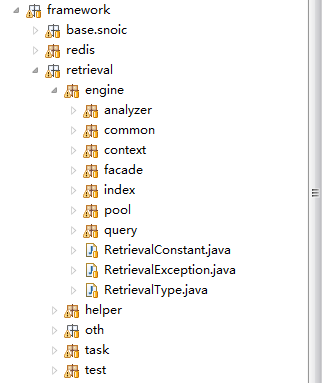
|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 作者 | 时间 | 备注 |
| V1.0 | 沈晓军 | 2013-12-23 |  |
|  |  |  |  |
|  |  |  |  |

# 开发指南

1. **项目结构**



1. **其中****中包括jeesite项目和文件检索页面展示，如下，所有全文检索相关的页面操作都放在这个retrieval下。**
2. **其中****中包括了全文检索，redis等所有内容，包括分词，建索引，搜索操作，都在这个下面**



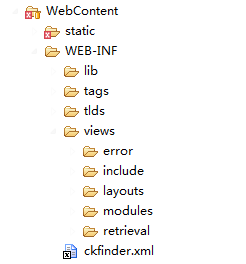
其中base.snoic包含了最基本的操作。

Redis包含了reids操作（暂时没有用，我集成了spring的redis）。

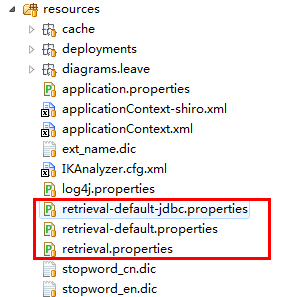
其中retrieval最为重要：

|  |  |  |
| --- | --- | --- |
| 主包 | 子包 | 作用 |
| Engine | analyzer | 分词包，例如将“中国人”分词分成“中国”和“中国人”，默认使用IK分词 |
| Common | Common中定义了lucene的版本等信息 |
| Context | 上下文，在检索和搜索时都通过这个上下文去获取基本信息 |
| Facade | 门面类，用于建索引和更新索引的入口 |
| Index | 建索引 |
| Pool | IndexRead和IndexWriter池，用来保证并发搜索的高效性 |
| query | 搜索 |
| Helper |  | 帮助类，用来分页查询 |
| Oth |  | 自己定义的类，包括，对象转json,对象转xml等等。今后会加入quartz |
| task |  | 任务模块，主要是建索引时的并发控制 |
| Test |  | 测试类（测试检索，建索引，删除、更新） |

1. **全文检索的所有页面都在retrieval下面：**



1. **配置文件**



全文检索用到的配置文件主要有这3个，

第一个jdbc方式的配置是用来配置建索引数据库的地址的，此文件今后要删掉的，数据库信息准备配置在redis中。

第二个是通用配置，配置了索引文件夹路径、redis等信息，今后所有需要配置的信息都存放到这个文件中。

第三个是全文检索建索引用到的配置文件，用来性能调优、配置合并因子和配置分词等等，这个可以不用管，应该用不到的。

1. **获取数据源的方式**

现在通过类DefaultDBGetMethodImpl这个实现类去完成从配置文件中加载数据源信息。如设置：

database.indexall.jdbc.driver=com.mysql.jdbc.Driver

database.indexall.jdbc.url=jdbc:mysql://127.0.0.1:3306/jeecg

database.indexall.jdbc.user=root

database.indexall.jdbc.password=11111

后续会将数据源存放到redis数据库中，通过redis方式去加载需要：

1.实现接口IDBGetMethod，例如  
**public** **class** RedisDBGetMethodImpl **implements** IDBGetMethod{

@Override

**public** DatabaseLink loadDatabaseLink() {

//doing some thing

**return** **null**;

}

}

2.在配置文件中retrieval-default.properties中配置

#配置采用获取数据源的方式，默认为framework.retrieval.engine.index.all.database.impl.DefaultDBGetMethodImpl:loadDatabaseLink

#其中冒号名前为类，冒号后为方法名，默认方法名为：loadDatabaseLink

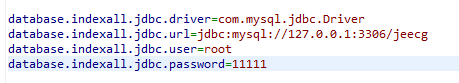
DEFAULT\_RETRIEVAL\_DATABASE\_CHOOSE\_CLASS=framework.retrieval.engine.index.all.database.impl.RedisDBGetMethodImpl:loadDatabaseLink

1. **测试类使用方法**

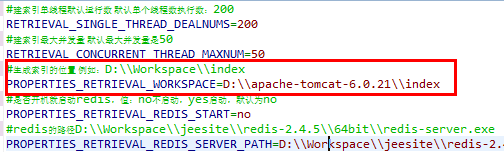
由于时间原因，我只测试了mysql数据库，使用方法简绍如下：

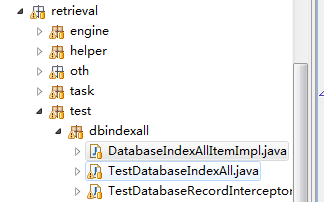
(1).配置retrieval-default-jdbc.properties中的数据库连接。（暂时只支持mysql）

配置参数如下：



(2)配置索引文件夹路径，在retrieval-default.properties中配置



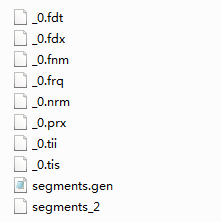
(3).测试建索引  


DatabaseIndexAllItemImpl用来设置数据库表，设置搜索出来的标题、正文等字段。

TestDatabaseIndexAll用来模拟索引的，配置好之后，右击run。

TestDatabaseRecordInterceptor拦截类，用来拦截数据库中查出的数据，进行调整，现在主要用来去除内容字段中的html标签。

当建完索引后，可以在文件夹中查看到索引：

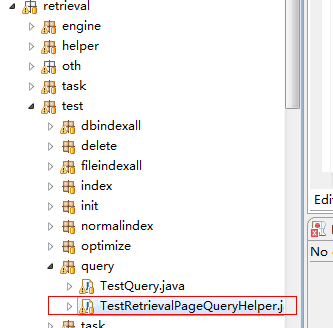


(3).测试搜索

有两种方式测试搜索，

第一种是直接页面访问：http://localhost:8080/jeesite/f/retrieval/search/

第二种是用测试类:TestRetrievalPageQueryHelper, 右击run.



具体使用可以参考下：  
双击打开这个pdf。

1. **安装ImageMagick**

在windows上安装ImageMagick：

download 地址：<http://ftp.icm.edu.pl/pub/unix/graphics/GraphicsMagick/windows/>

配置 环境变量加入PATH：D:\Files\ImageMagick-6.8.8-Q16（个人习惯放在D盘）

# 需求

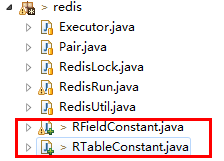
1. **数据源维护**

原数据源配置在retrieval-default-jdbc.properties这个文件中，现要将数据源配置到redis中。

修改入口DefaultRDatabaseIndexAllImpl类

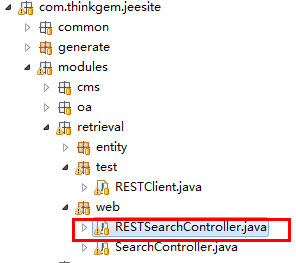


Reids数据库设计

数据源表用redis的hash表来存。键值对存放到这两个java文件中。  


Database表字段设计

|  |  |  |
| --- | --- | --- |
| 主键 | 字段 | 值 |
| Database: UUID:数据源名称 | databaseName | 数据源名称 |
| databaseType | 数据源类型： Oracle/Mysql/SqlServer |
| ip | 127.0.0.1:3306 |
| port | 3306 |
| user | root |
| password | 11111 |

页面可以做的简单点，但一定要能想rest一样能调用。参照下面这个类做：  


1. **基本字段管理**

InitFiled表字段设计，设计全文检索中用到的字段。不包括主键、标题、内容字段。

|  |  |  |
| --- | --- | --- |
| 主键 | 字段 | 值 |
| InitFiled/field名称 | Field | Pageurl,imgepath等基本字段 |
| fieldType | 设置分词或者不分词等类型 |
| DefaultValue | 字段为空时设置一个初始值 |

1. **字段映射管理**

FiledMapper字段映射器表。设计不同表与基本字段的映射关系。

|  |  |  |
| --- | --- | --- |
| 主键 | 字段 | 值 |
| FiledMapper/RDatabaseIndex的主键/UUID | sqlField | 数据库字段，比如page\_url |
| indexField | 索引字段，比如：Pageurl |

1. **特殊字段映射管理**

FiledSpecialMapper表字段设计，设计将数据库字段字段是否要进行大字段的处理，是否要去除html标签等。

|  |  |  |
| --- | --- | --- |
| 主键 | 字段 | 值 |
| FiledSpecialMapper/RDatabaseIndex的主键/UUID | sqlField | 数据库字段，比如title |
| SpecialType | 特殊处理类型，blob/clob/去除html标签。 |

1. **分类管理**

IndexCagetory表字段设计，设计全文检索的分类。

|  |  |  |
| --- | --- | --- |
| 主键 | 字段 | 值 |
| IndexCagetory/RDatabaseIndex的主键 | indexInfoType | 如：知识库、信息发布、停电信息 |
| IndexPathType | DB、FILE、IMAGE |
| IndexPath | TEST/TEST\_WEB |

1. **定时任务管理**

JustSchedule表字段设计，设计全文检索的分类。

|  |  |  |
| --- | --- | --- |
| 主键 | 字段 | 值 |
| JustSchedule/RDatabaseIndex的主键 | scheduleType | 是simple还是cron方式 |
| scheduleName | 调度名称 |
| frequencyUnits | 间隔单位 |
| startDate | 开始执行时间 |
| endDate | 结束时间 |
| execCount | 执行次数 |
| expression | 表达式 |
| frequency | 调度间隔 |

1. **索引管理**

RDatabaseIndex表字段设计，管理建索引

|  |  |  |
| --- | --- | --- |
| 主键 | 字段 | 值 |
| RDatabaseIndex/UUID | tableName | 数据源表名 |
| keyField | 主键字段 |
| Sql | Sql查询语句 |
| IndexOperatorType | 1.如果无论记录是否存在，都新增一条索引内容  2. 如果索引中记录已经存在，则只更新索引中的对应的记录，否则新增记录 |
| DefaultTitleFieldName | 设置数据库字段对应的标题 |
| DefaultResumeFieldName | 设置数据库字段对应的摘要字段 |
| RmDuplicate | 是否要去除重复标题正文字段；默认为true |
| DatabaseRecordInterceptor | 设置拦截器。 |
| isOn | 是否启用 |
| createTimeField | 创建时间 |
| updateTimeField | 修改时间 |
| fieldMapperList | 字段映射集合list |
| filedSpecialMapperrLsit | 特殊字段处理list |
| error | 错误信息 |
| isError | 是否存在error |
| Style | 风格 |
| condtion | sql条件 |
|  | indexTriggerRecord | 触发器表 |
|  | isInit | 是否初始化 |

1. **框架（想法：源于jfinal）**

采用jfinal

1. **数据库（想法：源于hadoop）**

不采用mysql数据库，而选择redis数据库，redis作为内存数据库，读写速度快，而且利于读写和数据分析。Java接连redis需要采用jedis实现。

1. **智能分析（想法：源于hadoop）**

当搜索时，将搜索的内容存到redis数据库lpush，然后起一个线池，监控redis中的数据，将完整关键字传过去，根据关键字分析。（分析名字、分析地名等）

例如：关键字只有2个或3个字，那么有可能是名字。比较第一个字，是否在百家姓中。如果在则（1）写入日志，（2）加入到reids库中。

后台一个任务，定时检查redis库中的数据，按照一定的策略：

H/D/W/M/Y,小时、日、周、月、年达到一定的搜索次数，就将词加入到ik词库中。配置格式：2 H 64 M 2小时超过64M

如果在时间段内日志量超过量（64M）则清空reids数据库，采用hadoop的MapReduce分析日志。然后将分析的结果加入到词典中。

1. **索引界面设计**
2. 增加一个风格选择“rest”，“复合”，rest方式是全由其他系统来控制索引的增删改，不需要配置新增、修改时间以及不需要调度。否则需要调度。
3. 增加一个启用、关闭按钮（字段），如果启用，则会对索引进行处理，否则不许处理调度等。
4. 前台去掉sql语句，后台增加字段验证是否通过检测。主要检测数据的完整性，只有通过了才能运行。
5. 初始字段对应的sql字段用用逗号分隔开。
6. **搜索排序**

原来建索引按表中的rowguid排序，现在改为按时间排序，方便搜索时时间排序,使用整型的唯一标识字段排序的处理时间是3016ms，使用索引ID（INDEXORDER）排序的时间是303ms, 为了能够使索引ID倒序等同于时间倒序：在建立索引时，就要按照数据的时间顺序建立，老的数据先索引，新的数据后索引

1. **搜索自定义打分（想法：源于黄浦项目需求）**

增加一个ScoreQuery类,对搜索的内容进行自定义打分，如：按年打分，2014年的内容分数打高一点，2013年的分数打底一点，参考如下：

<http://grantbb.iteye.com/blog/181802>

1. **检索采用触发器方式（想法：源于数据交换）**

由于目前检索只能对新增的数据库索引进行检索，而不能数据进行更新，删除索引，所以建议采用将数据交换一样的触发器方式对数据库进行索引，这样是最安全的。

1. **检索采用rest方式（想法：源于对rest的理解）**

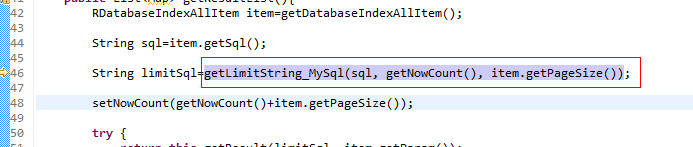
提供有效的rest API，可以增加、更新、删除索引，同时支持搜索，支持返回json格式的搜索结果也能支持返回xml的搜索结果。

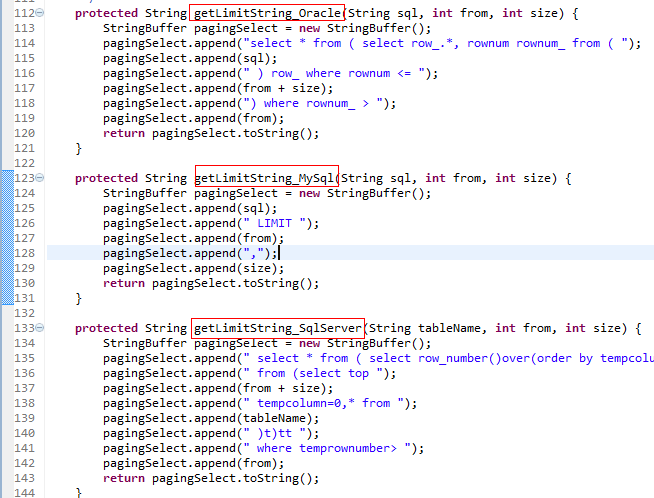
1. **定时任务（想法：源于数据交换）**

增加定时任务，可以个性化不同索引分类的调度时间。

1. **索引文件夹优化（想法：源于OA）**
2. **字段模糊搜索（想法：源于OA邮件）**
3. **针对不同数据库取数据(已修改——2013-12-24 沈晓军)**

现在只测了mysql方式，需要增加oracle和sqlserver方式读数据：

代码在在AbstractDefaultRDatabaseIndexAll类的第46行，我这边只做了mysql的分页：如下图所示：  


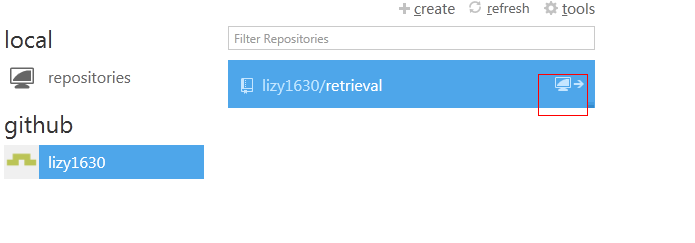
实际上应该包括Oracle和SqlServer的分页，分页代码我已经写了，在AbstractRDatabaseIndexAll类的第112行到144行，如下图：  


需要做一个配置项，确定取数据的数据源，然后测试下我上面写的分页是否正确。

# GitHub更新代码。

**Github上更新自己fork的代码**

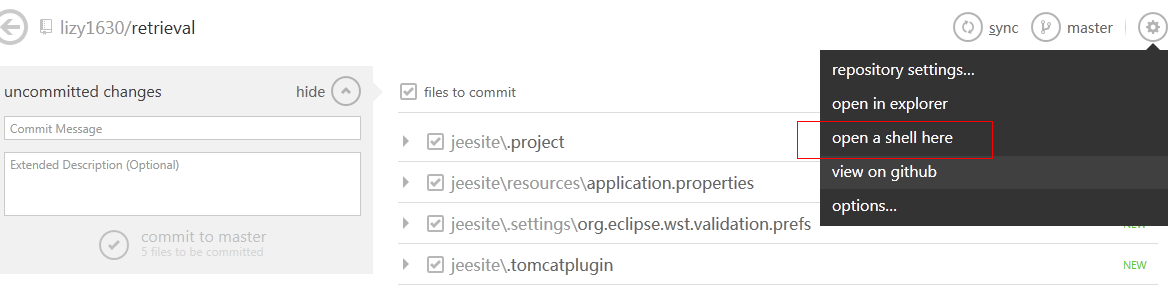
检出自己在github上fork别人的分支到rrest目录下，其中rrest目录之前是不存在的。



git clone git@github.com:yss/rrestjs.git rrest

或者如上图，直接clone这个分支到本地路径下；

然后打开shell增加远程分支（也就是你fork那个人的分支）名为bob（这个名字任意）到你本地。



git remote add bob <https://github.com/DoubleSpout/rrestjs.git>

如果你运行命令：git remote -v你会发现多出来了一个Bob的远程分支。如下：

bob https://github.com/DoubleSpout/rrestjs.git (fetch)

bob https://github.com/DoubleSpout/rrestjs.git (push)

origin git@github.com:yss/rrestjs.git (fetch)

origin git@github.com:yss/rrestjs.git (push)

然后，把对方的代码拉到你本地。

git fetch bob

最后，合并对方的代码。

git merge bob/master

最最后，把最新的代码推送到你的github上。

git push origin master

这样就完成了自己的代码更新。