## 启动nginx和fastDfs来当做图片服务器。

## nginx的启动

cd /usr/local/nginx/sbin

./nginx //启动

,/nginx -s stop //停止

./nginx -s reloadt //刷新配置

## FastDFS的启动

1. 启动tracker服务

/usr/bin/fdfs\_trackerd /etc/fdfs/tracker.conf

重启

/usr/bin/fdfs\_trackerd /etc/fdfs/tracker.conf

1. 启动storage服务。

/usr/bin/fdfs\_storaged /etc/fdfs/storage.conf

重启

/usr/bin/fdfs\_storaged /etc/fdfs/storage.conf restart

# 根据进程名来查看进程是否启动的命令

ps aux|grep name

# spring和springMVC的父子容器的概念

1. 子容器可以访问父容器的对象，父容器无法访问子容器的对象;即spring是父容器，springMvc是子容器。在spring的对象中无法装配spirngmvc所扫描装配的controller对象。
2. 当spring和springmvc所扫描的范围是一样时。Spring和springMvc的容器中都会装载这些对象。但是会各不相同。有可能导致spring配置的事务在springmvc中失效的情况。

Springmvc（前端控制器）（子容器）

Controller对象

Spring容器（父容器）

1. Mapper代理对象
2. Service对象
3. 注意：在spring和springmvc中配置的**属性**不能跨容器调用.

# Spring中获取资源文件中的值

使用@value注解

|  |
| --- |
| @value(“${属性名}”) |

注意:

前提是在spring的配置文件中必须要有加载配置文件的配置。

|  |
| --- |
| <context:property-placeholder location=”classpath:properties/\*.poreties /> |

# Linux 复制文件

|  |
| --- |
| cp 源文件名 目标路径 |

# Linux vim快速查询和取消查询

查询

|  |
| --- |
| :查询的关键字 |

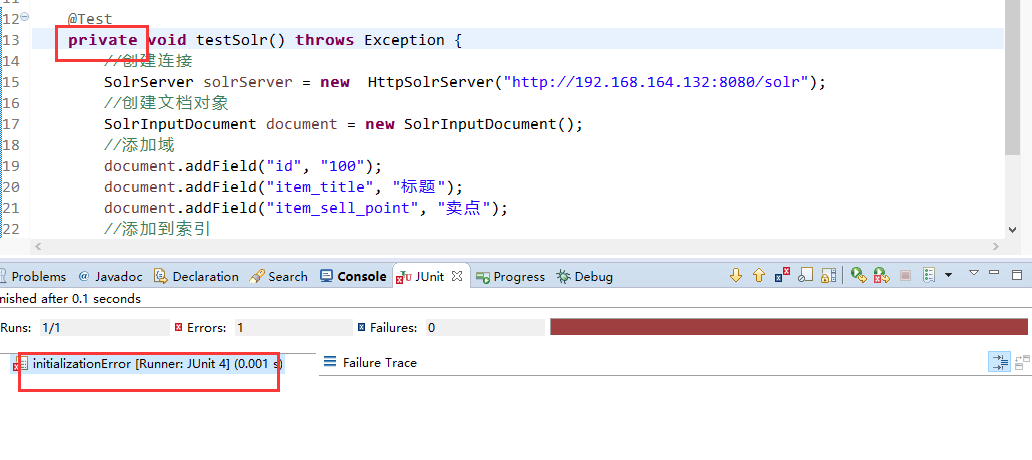
取消查询 即取消查询的高亮

|  |
| --- |
| :noh |

# 第八天笔记

## Junit笔记

如果执行的方法是private类型 将会执行不了，并会报initializationError



## Solr安装

### Solr安装 以centos 6.5 为例

第一步：安装jdk、安装tomcat

第二步：解压solr压缩包。

命令为 tar -zxf solr-4.10.3.tgz.tgz

第三步：把dist/solr-4.10.3.war部署到tomcat下。

|  |
| --- |
| cp solr-4.10.3.war /usr/local/solr/tomcat/webapps/solr.war -r |

第四步：解压缩war包。启动tomcat解压。

即启动Tomcat再关闭

|  |
| --- |
| [root@localhost tomcat]# ./bin/startup.sh |

再关闭tomcat 就解压完成

第五步：需要把/root/solr-4.10.3/example/lib/ext目录下的所有的jar包添加到solr工程中。

|  |
| --- |
| [root@localhost ext]# cp \* /usr/local/solr/tomcat/webapps/solr/WEB-INF/lib/ |

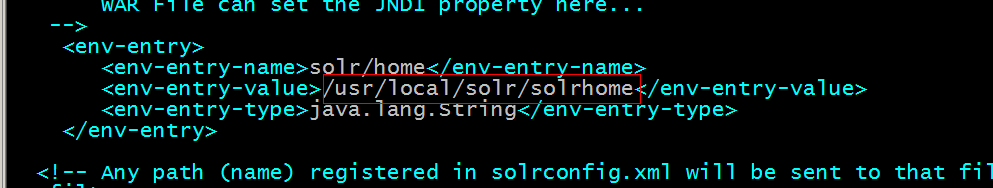
第六步：创建solrhome。把/root/solr-4.10.3/example/solr文件夹复制一份作为solrhome。

|  |
| --- |
| [root@localhost solr]# cp -r solr /usr/local/solr/solrhome |

第七步：告诉solr服务solrhome的位置。需要修改web.xml

web.xml在/usr/local/solr/tomcat/webapps/solr/WEB-INF/web.xml

修改结果为

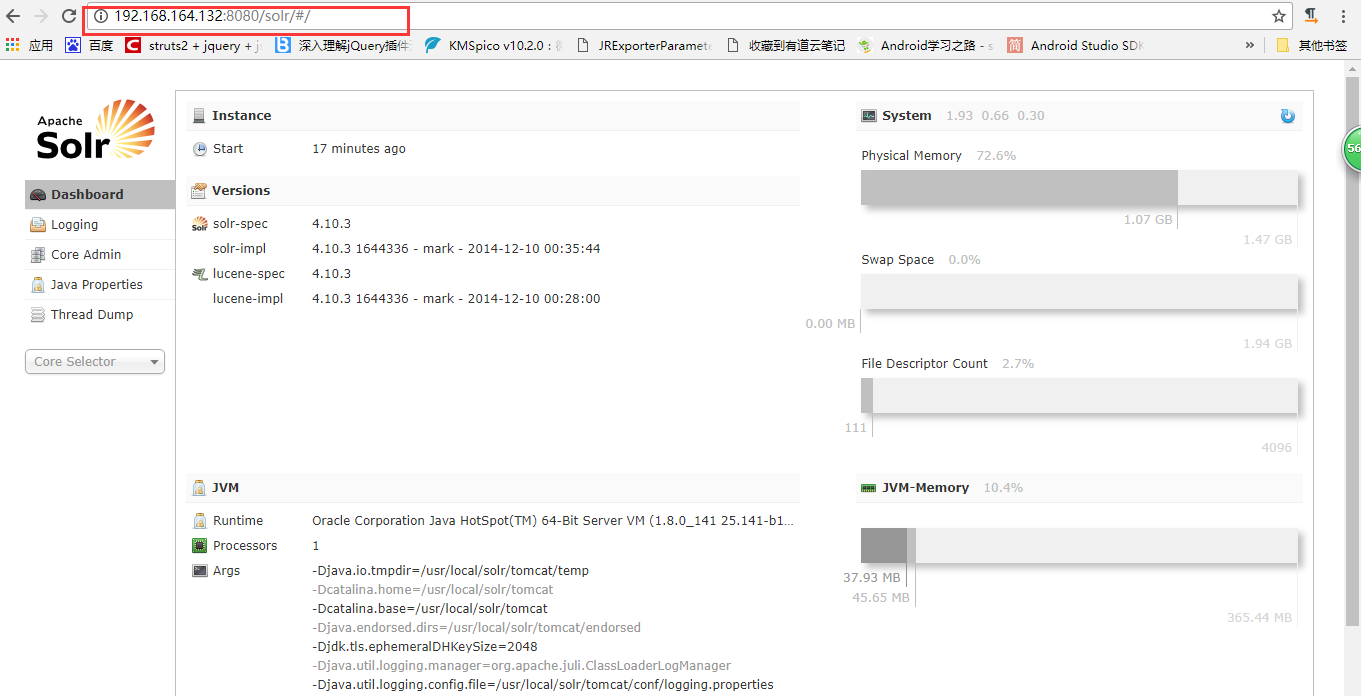


第八步：启动tomcat。

查看tomcat输出日志命令

|  |
| --- |
| tail -f logs/catalina.out |

最终结果：即安装成功



### solr配置中文分析器、自定义业务域

分析器使用IKAnalyzer。

使用方法：

第一步：把IKAnalyzer依赖的jar包添加到solr工程中。把分析器使用的扩展词典添加到classpath中。

|  |
| --- |
| [root@localhost IK Analyzer 2012FF\_hf1]# cp IKAnalyzer2012FF\_u1.jar /usr/local/solr/tomcat/webapps/solr/WEB-INF/lib/ |
| [root@localhost IK Analyzer 2012FF\_hf1]# cp ext\_stopword.dic IKAnalyzer.cfg.xml mydict.dic /usr/local/solr/tomcat/webapps/solr/WEB-INF/classes |

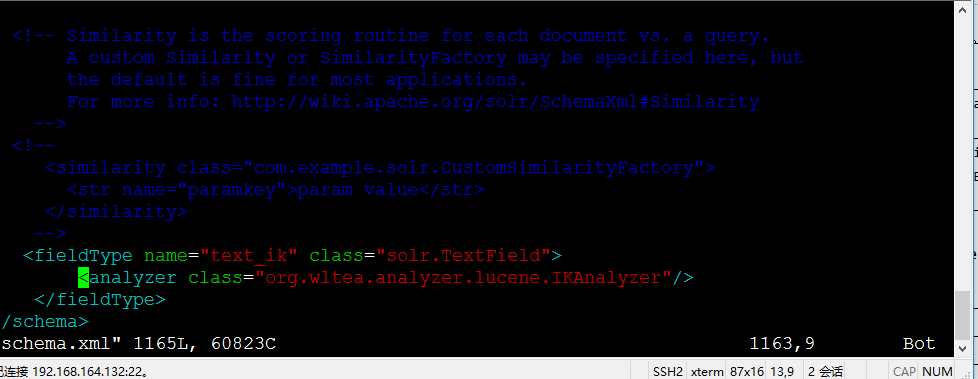
第二步：需要自定义一个FieldType。Schema.xml中定义。可以在FieldType中指定中文分析器。

地址：

|  |
| --- |
| /usr/local/solr/solrhome/collection1/conf/schema.xml |

在schema.xml的底部添加如下配置

|  |
| --- |
| <fieldType name="text\_ik" class="solr.TextField">  <analyzer class="org.wltea.analyzer.lucene.IKAnalyzer"/>  </fieldType> |



即完成中文分析器的配置

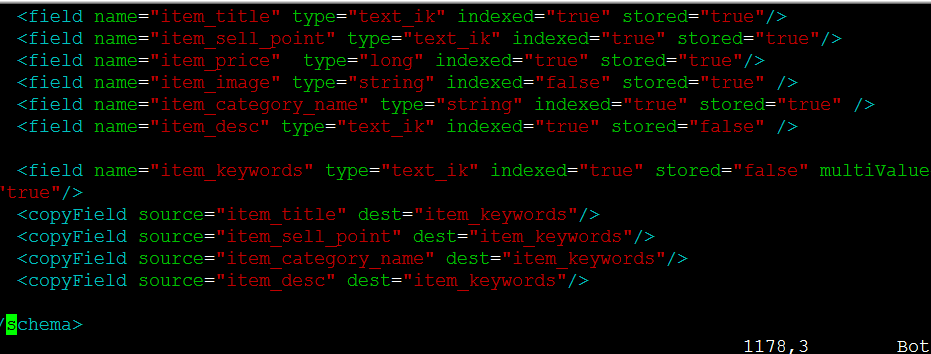
第三步：自定义域。指定域的类型为自定义的FieldType。

Sql语句：

|  |
| --- |
| SELECT  a.id,  a.title,  a.sell\_point,  a.price,  a.image,  b.`name` category\_name,  c.item\_desc  FROM  tb\_item a  LEFT JOIN tb\_item\_cat b ON a.cid = b.id  LEFT JOIN tb\_item\_desc c ON a.id = c.item\_id  WHERE  a.`status` = 1 |

将如下配置配置到schema.xml中完成solr的自定义域的配置

|  |
| --- |
| <field name="item\_title" type="text\_ik" indexed="true" stored="true"/>  <field name="item\_sell\_point" type="text\_ik" indexed="true" stored="true"/>  <field name="item\_price" type="long" indexed="true" stored="true"/>  <field name="item\_image" type="string" indexed="false" stored="true" />  <field name="item\_category\_name" type="string" indexed="true" stored="true" />  <field name="item\_desc" type="text\_ik" indexed="true" stored="false" />  <field name="item\_keywords" type="text\_ik" indexed="true" stored="false" multiValued="true"/>  <copyField source="item\_title" dest="item\_keywords"/>  <copyField source="item\_sell\_point" dest="item\_keywords"/>  <copyField source="item\_category\_name" dest="item\_keywords"/>  <copyField source="item\_desc" dest="item\_keywords"/> |



第四步：重新启动tomcat

## 搜索服务发布

调用服务传递过来一个查询条件，根据查询条件进行查询。返回查询结果。参数中包括分页条件。

参数：

String queryString

Int page

Int rows

返回结果：返回json数据。

包含查询结果的列表。使用商品的pojo来描述。SearchItem

包含查询结果总记录数。

包含查询结果的总页数。

包含当前页码。

包含查询的状态。

包含错误信息。

创建一个SearchResult

包含四个属性：

1、商品列表

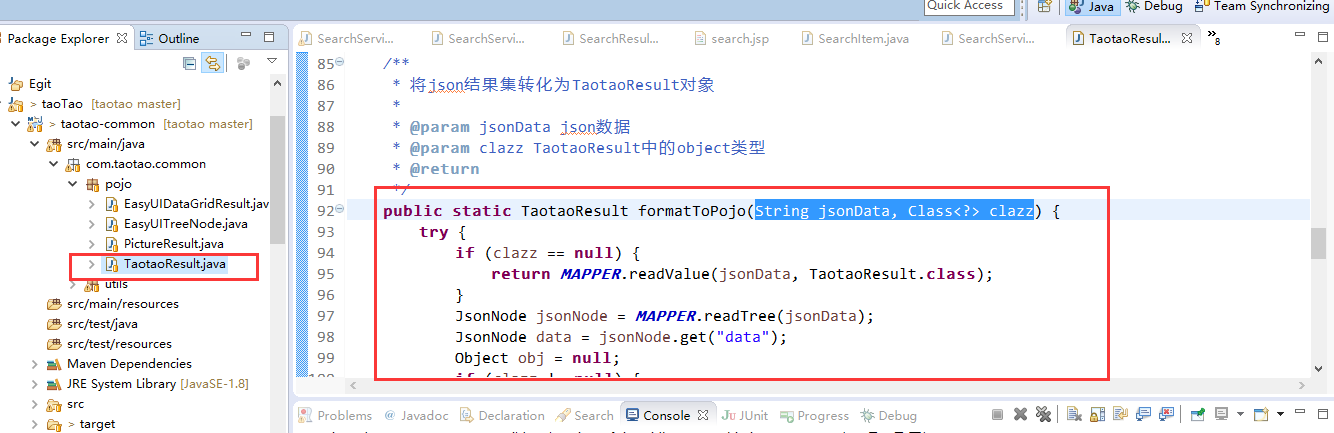
2、查询结果总记录数

3、查询结果的总页数

4、当前页码

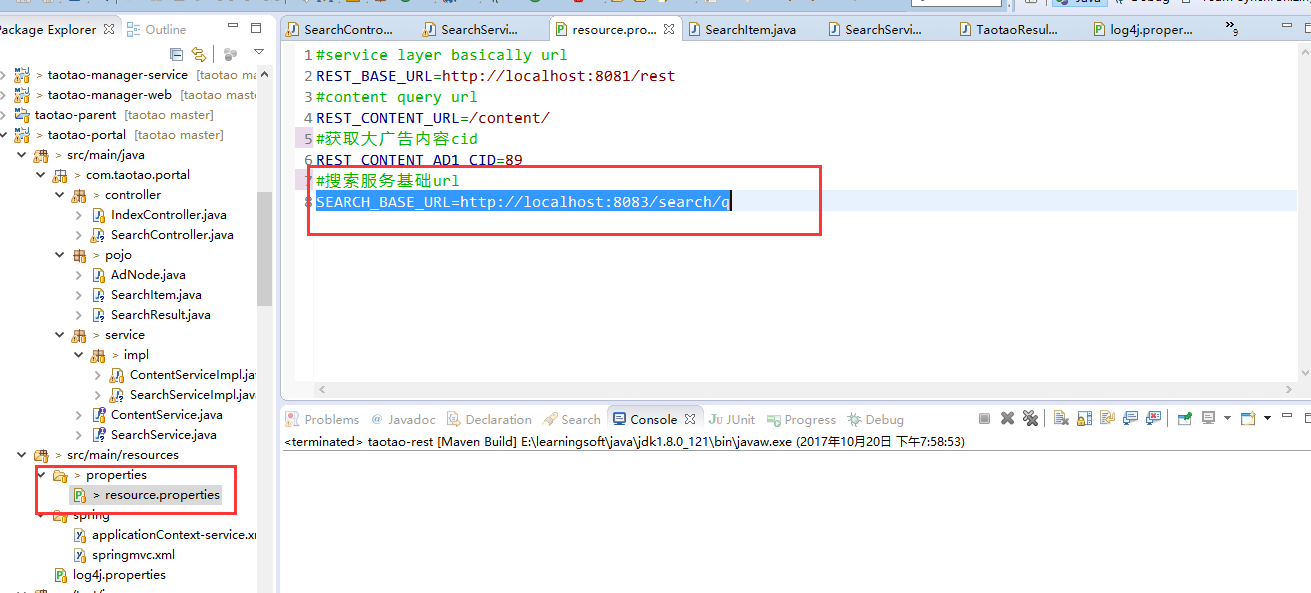
### 注意点

1. json转taotaoResult使用TaotaoResult.formatToPojo(String jsonData, Class<?> clazz;



1. 使用HttpClient向search服务请求时注意url的正确性。

SEARCH\_BASE\_URL=http://localhost:8083/search/q



4.HttpClient请求带参数的方式为

|  |
| --- |
| //使用HttpClientUtil 向search工程请求数据 返回类型为json字符串  Map<String, String> param = **new** HashMap<>();  param.put("keyword", keyword);  param.put("page", page + "");  param.put("rows", rows + "");  String json = HttpClientUtil.*doGet*(SEARCH\_BASE\_URL,param); |