

---

# solr 开发指南

## 1.1 solr 简介

### 1.1.1 官网介绍

Solr是一个基于Lucene的Java搜索引擎服务器。Solr 提供了层面搜索、命中醒目显示并且支持多种输出格式（包括 XML/XSLT 和 JSON 格式）。它易于安装和配置，而且附带了一个基于 HTTP 的管理界面。Solr已经在众多大型的网站中使用，较为成熟和稳定。Solr 包装并扩展了 Lucene，所以Solr的基本上沿用了Lucene的相关术语。更重要的是，Solr 创建的索引与 Lucene 搜索引擎库完全兼容。通过对 Solr 进行适当的配置，某些情况下可能需要进行编码，Solr 可以阅读和使用构建到其他 Lucene 应用程序中的索引。此外，很多 Lucene 工具（如 Nutch、Luke）也可以使用 Solr 创建的索引。

总结一下：solr是一个java搜索引擎服务器（是一套war程序），内部集成了Lucene(apache提供的一些对搜索引擎做支持的jar包)。

### 1.2.2 solr 功能

- 保存数据
- 建立索引，维护索引
- 数据检索(全文搜索，高亮显示，精确搜索等)

### 1.2.3 solr 依赖环境

- Jdk 1.7+
- TOMCAT 7+
- 课程选用版本：solr4.9.1

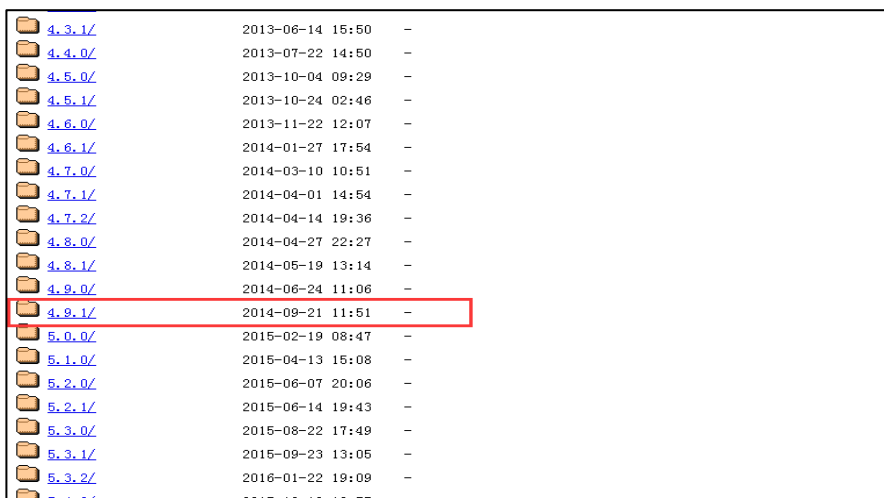
## 1.2 solr 服务器搭建

### 1.2.1 初始配置 solr

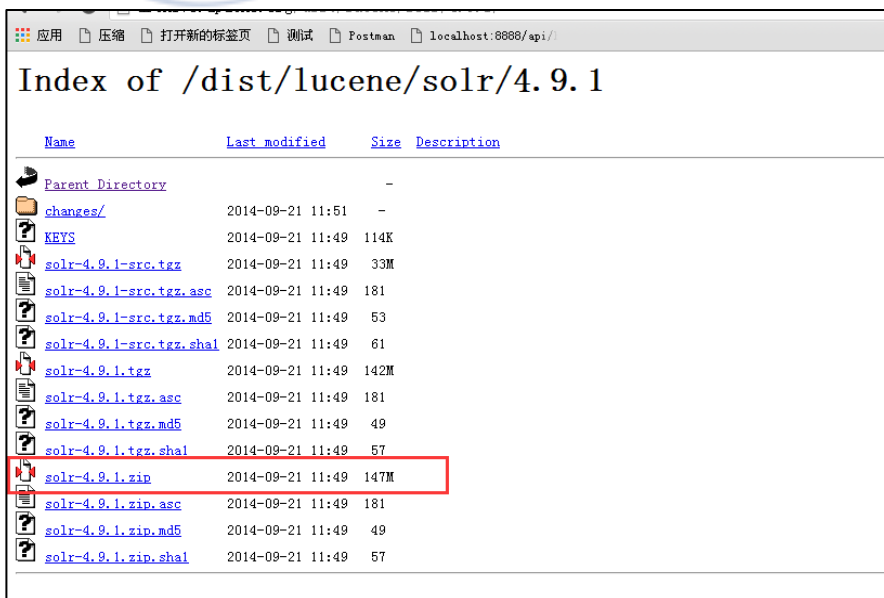
截止到文档编写前，solr目前的最新版本为6.5.1，在本次项目开发中，我们选用solr比较成熟稳定的版本solr 4.9.1。

1. 官网下载solr4.9.1的程序安装包。

下载地址：<http://archive.apache.org/dist/lucene/solr/>



<a href="#">4.3.1/</a>	2013-06-14 15:50	-
<a href="#">4.4.0/</a>	2013-07-22 14:50	-
<a href="#">4.5.0/</a>	2013-10-04 09:29	-
<a href="#">4.5.1/</a>	2013-10-24 02:46	-
<a href="#">4.6.0/</a>	2013-11-22 12:07	-
<a href="#">4.6.1/</a>	2014-01-27 17:54	-
<a href="#">4.7.0/</a>	2014-03-10 10:51	-
<a href="#">4.7.1/</a>	2014-04-01 14:54	-
<a href="#">4.7.2/</a>	2014-04-14 19:36	-
<a href="#">4.8.0/</a>	2014-04-27 22:27	-
<a href="#">4.8.1/</a>	2014-05-19 13:14	-
<a href="#">4.9.0/</a>	2014-06-24 11:06	-
<a href="#">4.9.1/</a>	2014-09-21 11:51	-
<a href="#">5.0.0/</a>	2015-02-19 08:47	-
<a href="#">5.1.0/</a>	2015-04-13 15:08	-
<a href="#">5.2.0/</a>	2015-06-07 20:06	-
<a href="#">5.2.1/</a>	2015-06-14 19:43	-
<a href="#">5.3.0/</a>	2015-08-22 17:49	-
<a href="#">5.3.1/</a>	2015-09-23 13:05	-
<a href="#">5.3.2/</a>	2016-01-22 19:09	-



Index of /dist/lucene/solr/4.9.1			
Name	Last modified	Size	Description
<a href="#">Parent Directory</a>	-	-	
<a href="#">changes/</a>	2014-09-21 11:51	-	
<a href="#">KEYS</a>	2014-09-21 11:49	114K	
<a href="#">solr-4.9.1-src.tgz</a>	2014-09-21 11:49	33M	
<a href="#">solr-4.9.1-src.tgz.asc</a>	2014-09-21 11:49	181	
<a href="#">solr-4.9.1-src.tgz.md5</a>	2014-09-21 11:49	53	
<a href="#">solr-4.9.1-src.tgz.shal</a>	2014-09-21 11:49	61	
<a href="#">solr-4.9.1.tgz</a>	2014-09-21 11:49	142M	
<a href="#">solr-4.9.1.tgz.asc</a>	2014-09-21 11:49	181	
<a href="#">solr-4.9.1.tgz.md5</a>	2014-09-21 11:49	49	
<a href="#">solr-4.9.1.tgz.shal</a>	2014-09-21 11:49	57	
<a href="#">solr-4.9.1.zip</a>	2014-09-21 11:49	147M	
<a href="#">solr-4.9.1.zip.asc</a>	2014-09-21 11:49	181	
<a href="#">solr-4.9.1.zip.md5</a>	2014-09-21 11:49	49	
<a href="#">solr-4.9.1.zip.shal</a>	2014-09-21 11:49	57	

2. 解压solr的zip包，目录如下

名称	大小	类型	修改日期
contrib		文件夹	2014-9-18 4:10
dist		文件夹	2014-9-18 4:10
docs		文件夹	2014-9-18 4:10
example		文件夹	2014-9-18 4:10
licenses		文件夹	2014-9-18 4:10
CHANGES.txt	386 KB	文本文档	2014-9-18 4:10
LICENSE.txt	13 KB	文本文档	2014-9-18 4:10
NOTICE.txt	27 KB	文本文档	2014-9-18 4:10
README.txt	6 KB	文本文档	2014-9-18 4:10
SYSTEM_REQUIREMENTS.txt	1 KB	文本文档	2014-9-18 4:10

3. 将dist\solr-4.9.1.war文件复制到tomcat的webapps目录下，并将文件命名为solr.war

名称	大小	类型	修改日期
solrj-lib		文件夹	2014-9-18 4:10
test-framework		文件夹	2014-9-18 4:10
<b>solr-4.9.1.war</b>	<b>28,746 KB</b>	<b>WAR 文件</b>	<b>2014-9-18 4:09</b>
solr-analysis-extras-4.9.1.jar	18 KB	Executable Jar ...	2014-9-18 4:08
solr-cell-4.9.1.jar	30 KB	Executable Jar ...	2014-9-18 4:08
solr-clustering-4.9.1.jar	51 KB	Executable Jar ...	2014-9-18 4:08
solr-core-4.9.1.jar	2,631 KB	Executable Jar ...	2014-9-18 4:09
solr-dataimporthandler-4.9.1.jar	214 KB	Executable Jar ...	2014-9-18 4:08
solr-dataimporthandler-4.9.1.jar	32 KB	Executable Jar ...	2014-9-18 4:08
solr-langid-4.9.1.jar	750 KB	Executable Jar ...	2014-9-18 4:08
solr-map-reduce-4.9.1.jar	127 KB	Executable Jar ...	2014-9-18 4:08
solr-morphlines-cell-4.9.1.jar	25 KB	Executable Jar ...	2014-9-18 4:08
solr-morphlines-core-4.9.1.jar	42 KB	Executable Jar ...	2014-9-18 4:09
solr-solrj-4.9.1.jar	420 KB	Executable Jar ...	2014-9-18 4:09
solr-test-framework-4.9.1.jar	182 KB	Executable Jar ...	2014-9-18 4:08
solr-uima-4.9.1.jar	40 KB	Executable Jar ...	2014-9-18 4:09
solr-velocity-4.9.1.jar	21 KB	Executable Jar ...	2014-9-18 4:09

名称	大小	类型	修改日期
docs		文件夹	2017-5-3 13:35
examples		文件夹	2017-5-3 13:35
host-manager		文件夹	2017-5-3 13:35
manager		文件夹	2017-5-3 13:35
ROOT		文件夹	2017-5-4 15:11
temp		文件夹	2017-5-5 14:01
<b>solr.war</b>	<b>43,992 KB</b>	<b>WAR 文件</b>	<b>2017-5-11 17:23</b>

4. 复制 solr解压包下example\lib\ext 下所有的jar 到tomcat 的lib目录下

I:\solr-4.9.1\example\lib\ext				
名称	大小	类型	修改日期	
jcl-over-slf4j-1.7.6.jar	17 KB	Executable Jar ...	2014-2-5 17:39	
jul-to-slf4j-1.7.6.jar	5 KB	Executable Jar ...	2014-2-5 17:40	
log4j-1.2.17.jar	479 KB	Executable Jar ...	2012-5-26 5:43	
slf4j-api-1.7.6.jar	29 KB	Executable Jar ...	2014-2-5 17:37	
slf4j-log4j12-1.7.6.jar	9 KB	Executable Jar ...	2014-2-5 17:38	

5. 在计算机本地新建一个文件夹solr\_home（当然你可以随便起名字），然后复制solr-4.9.1\example\solr 下的所有文件到 solr\_home下

I:\solr-4.9.1\example\solr				
名称	大小	类型	修改日期	
bin		文件夹	2014-8-16 6:30	
collection1		文件夹	2014-9-18 4:10	
README.txt	3 KB	文本文档	2014-9-18 4:10	
solr.xml	2 KB	XML 文件	2014-9-18 4:10	
zoo.cfg	1 KB	CFG 文件	2014-9-18 4:10	

地址 (D:) F:\solr_home				
名称	大小	类型	修改日期	
bin		文件夹	2017-5-25 10:13	
collection1		文件夹	2017-5-25 10:13	
README.txt	3 KB	文本文档	2014-9-18 4:10	
solr.xml	2 KB	XML 文件	2014-9-18 4:10	
zoo.cfg	1 KB	CFG 文件	2014-9-18 4:10	

6. 启动tomcat，待tomcat启动成功后，关闭tomcat。打开tomcat的webapps目录。注意，此时solr的war包以及被解压成solr文件夹。删除tomcat 的webapps目录下的solr的war包，保留solr文件夹。
7. 修改配置文件 apache-tomcat-7.0.67\webapps\solr\WEB-INF\web.xml

```

<env-entry>

<env-entry-name>solr/home</env-entry-name>

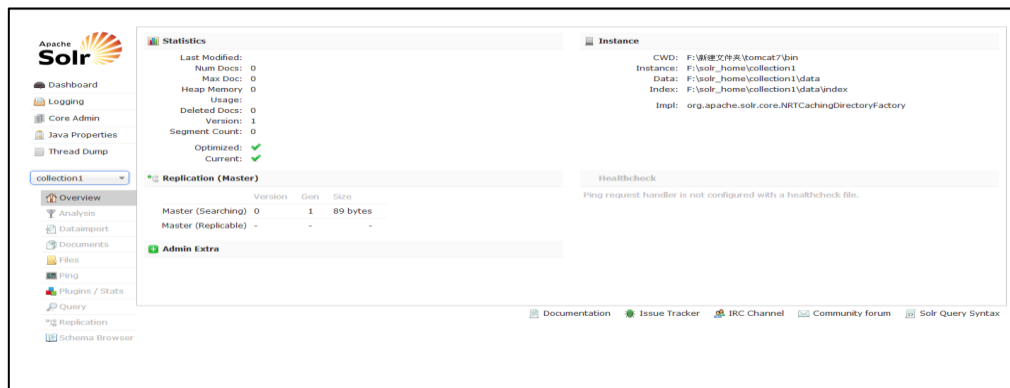
    <env-entry-value> F:/solr_home</env-entry-value>

<env-entry-type>java.lang.String</env-entry-type>

</env-entry>

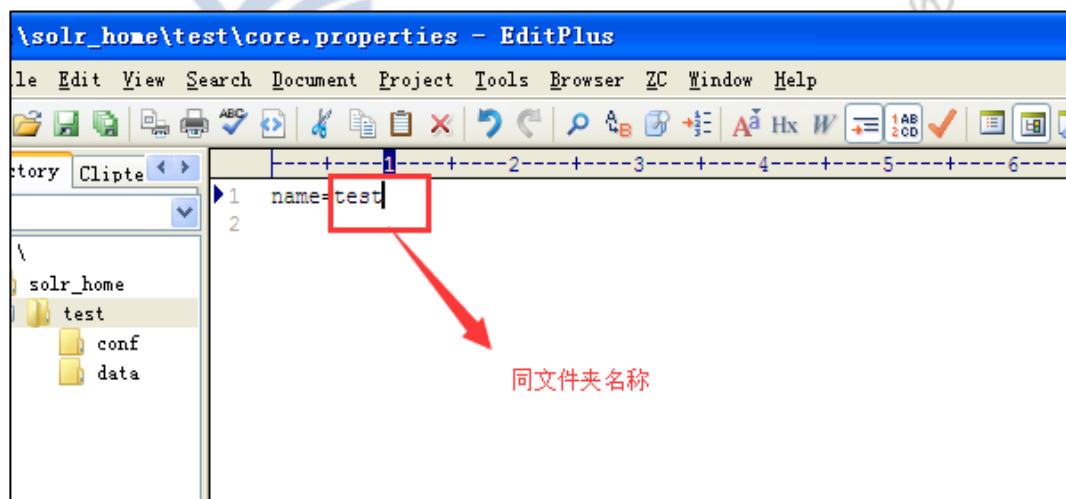
```

8. 访问solr(<http://localhost:端口号/solr/>),如出现以下界面则solr部署成功。

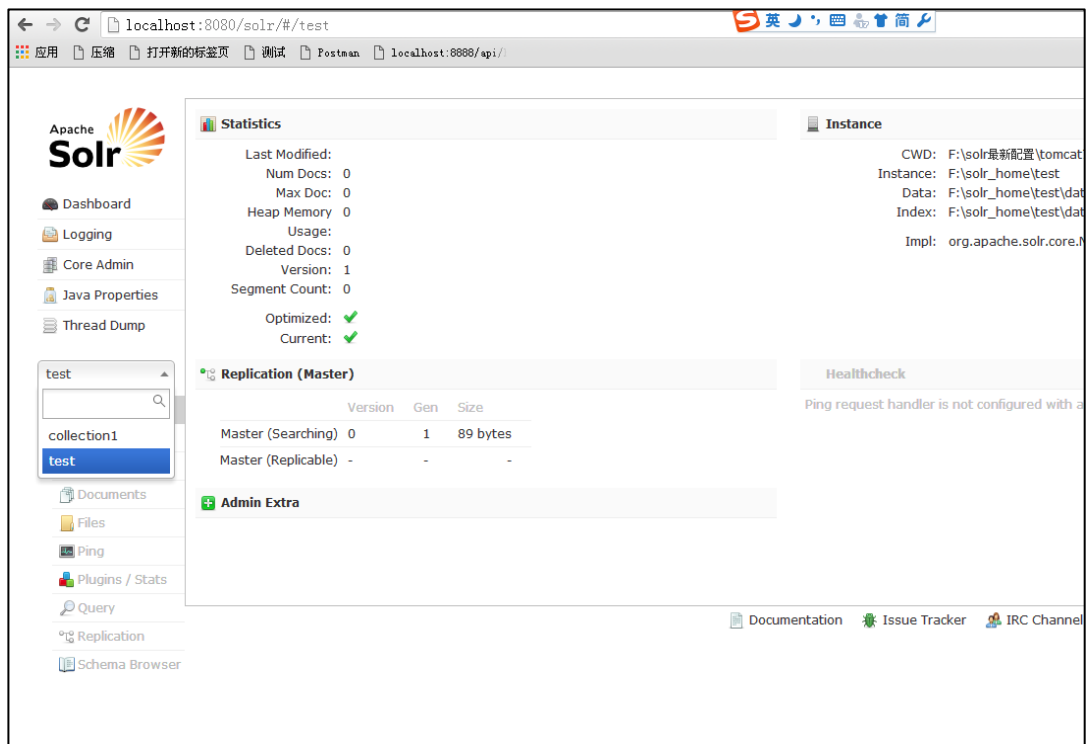


## 1.2.2 新建数据配置 core

1. 新建core(solr中把配置的每一个模块都叫core),在solr\_home目录下, 拷贝collection1文件夹, 并起名为test。打开test文件夹, 修改core.properties文件,将name修改为test



2. 重新启动tomcat, 并访问solr,如出现以下界面,则表示新建test core成功。



3. 重新启动tomcat, 并访问solr,如出现以下界面,则表示新建test core成功。

### 1.2.3 新增数据库配置

到目前为止, 我们已经完成了solr的基础配置, 并且创建了test core, 接下来我们需要把数据的数据和搜索引擎连接起来, 让搜索引擎可以读取数据库的数据。

1. 拷贝数据库连接jar包(mysql-connector-java-5.1.18.jar)到tomcat的lib目录
2. 以创建test core的方式新建hotel core
3. 打开hotel的conf文件夹中的solrconfig.xml文件, 在requestHandler name="/select" class="solr.SearchHandler">前面上加上一个dataimport的处理的Handler

```
<requestHandler name="/ dataimport"
class="org.apache.solr.handler.dataimport.DataImportHandler">

  <lst name="defaults">

    <str name="config">data-config.xml</str>

  </lst>
```

```
</requestHandler>
```

4. 在hotel的的conf文件夹下并新建data-config.xml文件，配置如下

```
<?xml version="1.0" encoding="UTF-8"?>

<dataConfig>

<dataSource type="JdbcDataSource" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://127.0.0.1:3306/itripdb" user="root" password="root" />

<document name="hotel_doc">

    <entity name="hotel" pk="id" query="select id,hotelName,address from
itrip_hotel">

        <field column="id" name="id"/>

        <field column="hotelName" name="hotelName"/>

        <field column="address" name="address"/>

    </entity>

</document>

</dataConfig>
```

- dataSource是数据库数据源。
- Entity就是一张表对应的实体，pk是主键，query是查询语句。
- Field对应一个字段，column是数据库里的column名，后面的name属性对应着Solr的Filed的名字。

5. 打开hotel的conf目录下的schema.xml文件

(1) 保留\_version\_ 这个field

(2) 添加索引字段：这里每个field的name要和data-config.xml里的entity的field的name一样，一一对应。

```

<?xml version="1.0" encoding="UTF-8"?>

<dataConfig>

<dataSource type="JdbcDataSource" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://127.0.0.1:3306/itripdb" user="root" password="root" />

<document name="hotel_doc">

    <entity name="hotel" pk="id" query=" select id,hotelName,address from
itrip_hotel">

        <field column="id" name="id"/>

        <field column="hotelName" name=" hotelName"/>

        <field column="address" name=" address"/>

    </entity>
</document>
</dataConfig>

```

修改同目录下的schema.xml(schema.xml 是solr对数据库里的数据进行索引管理和数据字段展示管理的配置文件)

- 删除多余的field，保留\_version\_ 和test这两个field（注意不要删除fieldType）
- 添加索引字段：这里每个field的name要和data-config.xml里的entity的field的name一样，一一对应。红色加粗部分为新增内容。

```

<field name="_version_" type="long" indexed="true" stored="true"/>

<field name="id" type="string" indexed="true" stored="true"/>

<field name="hotelName" type="string" indexed="true" stored="true"/>

<field name="address" type="string" indexed="true" stored="true"/>

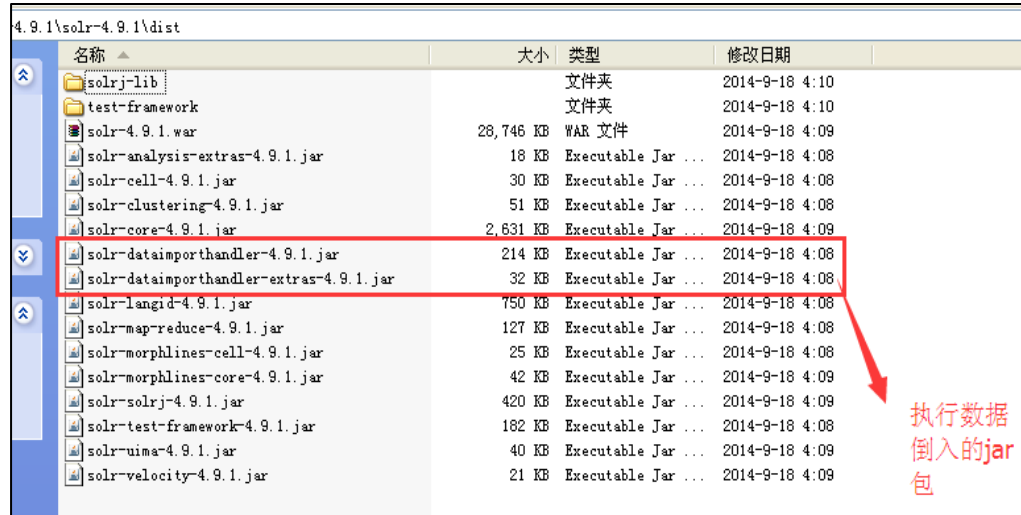
<uniqueKey>id</uniqueKey>

```

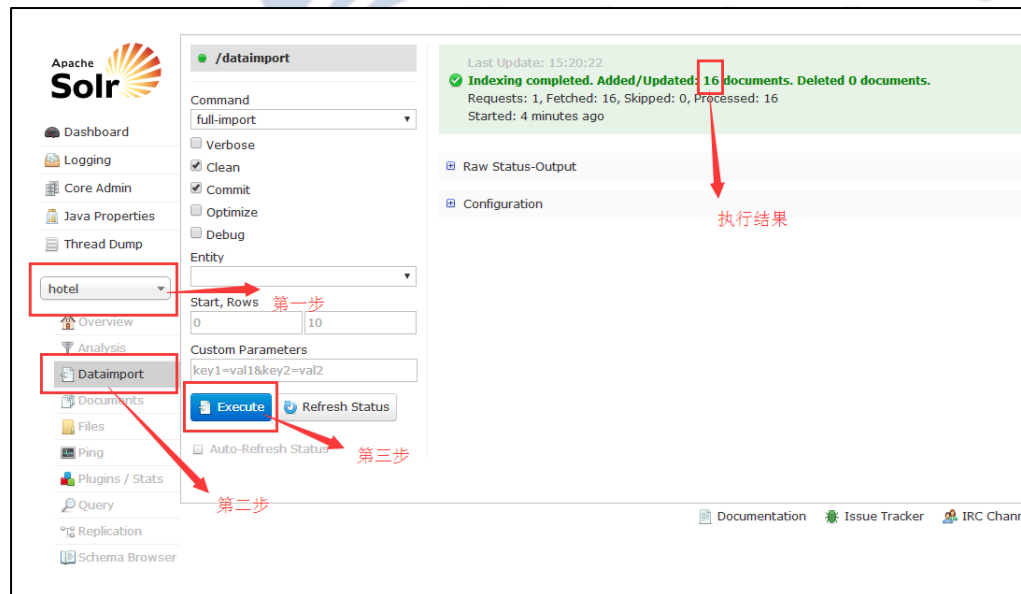


```
<field name="text" type="text_general" indexed="true" stored="false" multiValued="true"/>
```

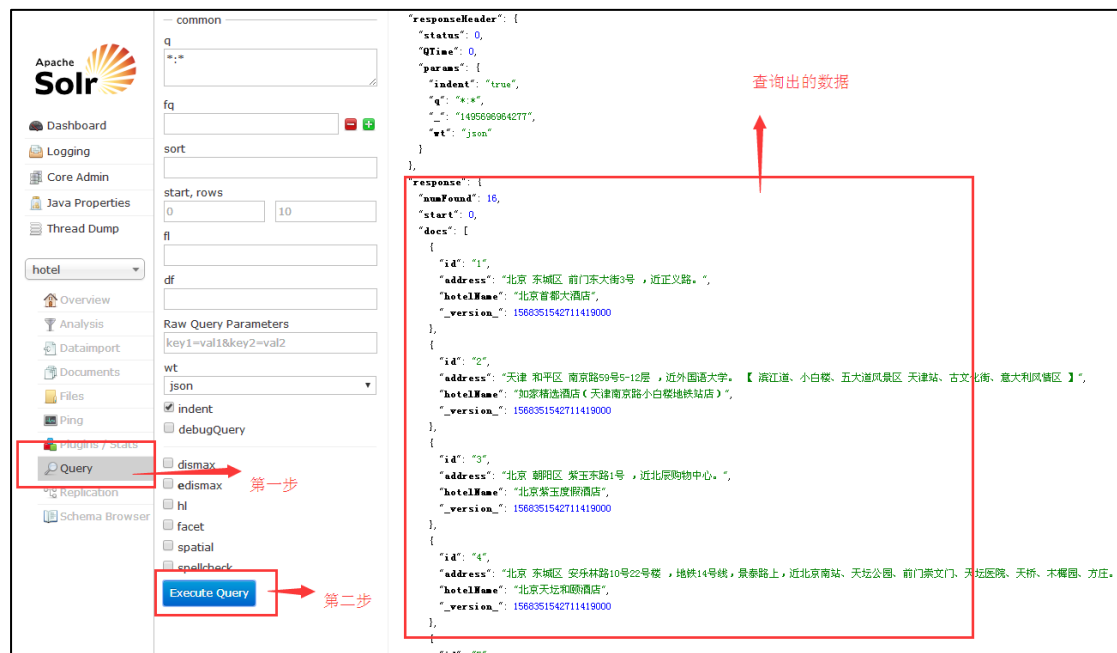
## 6. 将导入数据的JAR包拷贝到webapps/solr的lib目录下



## 7. 启动Tomcat，执行数据导入。



## 8. 查询数据



到目前为止，我们已经将数据库的数据导入到solr当中了，并且已经查询成功。

## 1.2.4 配置增量更新

以上的步骤我们实现了如何将数据库的数据导入到solr中，接下来需要配置solr的增量更新，即定时将数据库的数据导入到solr中。

1. 将资料中提供的apache-solr-dataimports-cheduler.jar包添加至solr的lib目录下

**注：apachesolrdataimportscheduler.jar的jar包是apache提供的用于增量更新的jar包，但apache提供的原jar包中，代码有BUG。该bug在教学资料提供的jar包中已经被修复，具体可参考提供的apachesolrdataimportscheduler的源码，在此不再赘述。**

2. 将资料中提供的apache-solr-dataimports-cheduler.jar包添加至solr的lib目录下。
3. 增加增量更新配置文件,在solr\_home文件夹下新建conf文件夹，并新建名为dataimport.properties的配置文件，配置如下,标红的地方为需要修改的地方

```
#####

#                                     #

#      dataimport scheduler properties      #
```

```
# #

#####

# to sync or not to sync

# 1 - active; anything else - inactive

syncEnabled=1

# which cores to schedule

# in a multi-core environment you can decide which cores you want synchronized

# leave empty or comment it out if using single-core deployment

syncCores=test,hotel

# solr server name or IP address

# [defaults to localhost if empty]

server=localhost

# solr server port

# [defaults to 80 if empty]

port=8080

# application name/context

# [defaults to current ServletContextListener's context (app) name]

webapp=solr

# 增量索引的参数

# URL params [mandatory]

# remainder of URL

params=/dataimport?command=delta-import&clean=false&commit=true
```

```
# 重做增量索引的时间间隔

# schedule interval

# number of minutes between two runs

# [defaults to 30 if empty]

interval=1

# 重做全量索引的时间间隔，单位分钟，默认7200，即5天;

# 为空,为0,或者注释掉:表示永不重做索引

#reBuildIndexInterval=7200

# 重做索引的参数

reBuildIndexParams=/dataimport?command=full-import&clean=true&commit=true

# 重做索引时间间隔的计时开始时间，第一次真正执行的时间

=reBuildIndexBeginTime+reBuildIndexInterval*60*1000;

# 两种格式：2012-04-11 03:10:00 或者 03:10:00，后一种会自动补全日期部分为服务启动时的日期

reBuildIndexBeginTime=03:10:00
```

4. 新增增量更新数据的监听器，在solr的web.xml中加入以下监听器

```
<listener>

  <listener-class>

    org.apache.solr.handler.dataimport.scheduler.ApplicationListener

  </listener-class>

</listener>
```

5. 修改导入数据查询SQL

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<dataConfig>

<dataSource type="JdbcDataSource" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://127.0.0.1:3306/itripdb" user="root" password="root" />

<document name="hotel_doc">

    <entity name="hotel" pk="id" query="select id,hotelName,address from itrip_hotel"

        deltaImportQuery="select id,hotelName,address from itrip_hotel where id
        = '${dih.delta.id}'"

        deltaQuery="SELECT id as id FROM itrip_hotel where modifyDate >
        '${dih.last_index_time}'">

        <field column="id" name="id"/>

        <field column="hotelName" name="hotelName"/>

        <field column="address" name="address"/>

    </entity>
</document>
</dataConfig>

```

说明：deltaQuery是根据dataimport.properties配置文件中的更新时间，从数据库中查询出，修改日期在最后一次更新日期之后的酒店数据，并记录其id，而deltaImportQuery的目的是将deltaQuery查询出的数据导入到solr中。

#### 6. 启动Tomcat进行测试

- 启动Tomcat，访问hotel模块
- 修改数据库中的酒店数据并同时修改该数据的modifyDate时间，
- 1分钟后查询酒店数据，确定数据是否更新

### 1.2.5 配置分词器

- 分词器: 是从用户输入的一段文本中提取关键词, 用于其它业务操作。
- 常见的 JAVA 分词器: word 分词器、Ansj 分词器、Stanford 分词器、IKAnalyzer 分词器
- 课程选用分词器: IKAnalyzer 分词器
- solr 如果是 3.x 版本的用 IKAnalyzer2012\_u6.zip 如果是 4.x 版本的用 IK Analyzer 2012FF\_hf1.zip, 一定要对应上, 要不然会配置失败。
- IK 分词器下载地址: <http://download.csdn.net/download/tjcyjd/8420639>

1. 首先, 下载IKAnalyzer。
2. 将ik的所有jar文件 拷贝到 webapps\solr\WEB-INF\lib 目录下

名称	大小	类型	修改日期
doc		文件夹	2012-10-23 16:59
IKAnalyzer2012FF_u1.jar	1,139 KB	Executable Jar ...	2012-10-26 20:46
IKAnalyzer.cfg.xml	1 KB	XML 文件	2012-2-14 11:21
IKAnalyzer中文分词器V20...	822 KB	PDF 文件	2012-10-24 11:47
LICENSE.txt	18 KB	文本文档	2012-1-17 10:22
NOTICE.txt	1 KB	文本文档	2012-1-19 23:38
stopword.dic	1 KB	文本文档	2011-4-15 16:39

3. 在webapps\solr\WEB-INF\下新建classes文件夹, 将IKAnalyzer.cfg.xml和stopword.dic 文件拷贝到改文件夹下。
4. 在 solr\_home\hotel\conf\schema.xml 增加如下配置

```
<fieldType name="text_ik" class="solr.TextField">
  <analyzer type="index" isMaxWordLength="false"
class="org.wltea.analyzer.lucene.IKAnalyzer"/>
  <analyzer type="query" isMaxWordLength="true"
class="org.wltea.analyzer.lucene.IKAnalyzer"/>
</fieldType>
```

5. 修改solr\_home\hotel\conf\schema.xml将hotelName和address指定成为text\_ik类型

```
<field name="hotelName" type="text_ik" indexed="true" stored="true"/>
<field name="address" type="text_ik" indexed="true" stored="true"/>
```

6. 重启Tomcat, 访问solr测试分词器

Field Value (Index)  
中华人民共和国

Field Value (Query)

Analyze Fieldname / FieldType: address

Verbose Output

Analyze Values

Field	Value	Start	End	Position	Length	Type
中华人民共和国	[e4 b8 ad e5 8d 8e e4 ba ba e6 b0 91 e5 85 b1 e5 92 8e e5 9b bd]	0	7	1	1	CN_WORD
中华人民共和国	[e4 b8 ad e5 8d 8e e4 ba ba e6 b0 91]	4	4	1	1	CN_WORD
中华人民共和国	[e4 b8 ad e5 8d 8e]	2	2	1	1	CN_WORD
中华人民共和国	[e5 8d 8e e4 ba ba]	3	3	1	1	CN_WORD
中华人民共和国	[e4 ba ba e6 b0 91 e5 85 b1 e5 92 8e e5 9b bd]	2	5	1	1	CN_WORD

第一步

第二步 输入中文字符串

第三步 选择指定为分词类型的字段

第四步 进行分词分析

第五步: 查看分析结果

7. 如果分词器出现上边的显示结果，则表示分词器配置成功。

## 1.3 常见异常

### 1.3.1 删除了默认字段(text 或者 \_version\_)

```
org.apache.solr.common.SolrException: undefined field text
    at org.apache.solr.schema.IndexSchema.getDynamicFieldType(IndexSchema.java:1267)
    at org.apache.solr.schema.IndexSchema$SolrQueryAnalyzer.getWrappedAnalyzer(IndexSchema.java:433)
    at org.apache.lucene.analysis.AnalyzerWrapper.initReader(AnalyzerWrapper.java:117)
    at org.apache.lucene.analysis.Analyzer.tokenStream(Analyzer.java:178)
    at org.apache.lucene.util.QueryBuilder.createFieldQuery(QueryBuilder.java:207)
    at org.apache.solr.parser.SolrQueryParserBase.newFieldQuery(SolrQueryParserBase.java:375)
    at org.apache.solr.parser.SolrQueryParserBase.getFieldQuery(SolrQueryParserBase.java:743)
    at org.apache.solr.parser.SolrQueryParserBase.handleBareTokenQuery(SolrQueryParserBase.java:542)
    at org.apache.solr.parser.QueryParser.Term(QueryParser.java:299)
    at org.apache.solr.parser.QueryParser.Clause(QueryParser.java:185)
    at org.apache.solr.parser.QueryParser.Query(QueryParser.java:107)
    at org.apache.solr.parser.QueryParser.TopLevelQuery(QueryParser.java:96)
    at org.apache.solr.parser.SolrQueryParserBase.parse(SolrQueryParserBase.java:152)
    at org.apache.solr.search.LuceneQParser.parse(LuceneQParser.java:50)
    at org.apache.solr.search.QParser.getQuery(QParser.java:141)
    at org.apache.solr.handler.component.QueryComponent.prepare(QueryComponent.java:148)
    at org.apache.solr.handler.component.SearchHandler.handleRequestBody(SearchHandler.java:197)
    at org.apache.solr.handler.RequestHandlerBase.handleRequest(RequestHandlerBase.java:135)
    at org.apache.solr.core.SolrCore.execute(SolrCore.java:1962)
    at org.apache.solr.core.QuerySenderListener.newSearcher(QuerySenderListener.java:64)
    at org.apache.solr.core.SolrCore$5.call(SolrCore.java:1734)
    at java.util.concurrent.FutureTask.run(FutureTask.java:262)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
    at java.lang.Thread.run(Thread.java:745)
```

### 1.3.2 没有加入增量更新的 jar 包

```
java.lang.ClassNotFoundException: org.apache.solr.handler.dataimport.scheduler.ApplicationListener
    at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1720)
    at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1571)
    at org.apache.catalina.core.DefaultInstanceManager.loadClass(DefaultInstanceManager.java:506)
    at org.apache.catalina.core.DefaultInstanceManager.loadClassMaybePrivileged(DefaultInstanceManager.java:488)
    at org.apache.catalina.core.DefaultInstanceManager.newInstance(DefaultInstanceManager.java:115)
    at org.apache.catalina.core.StandardContext.listenerStart(StandardContext.java:4909)
    at org.apache.catalina.core.StandardContext.startInternal(StandardContext.java:5492)
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:150)
    at org.apache.catalina.core.ContainerBase.addChildInternal(ContainerBase.java:901)
    at org.apache.catalina.core.ContainerBase.addChild(ContainerBase.java:877)
    at org.apache.catalina.core.StandardHost.addChild(StandardHost.java:649)
    at org.apache.catalina.startup.HostConfig.deployDirectory(HostConfig.java:1245)
    at org.apache.catalina.startup.HostConfig$DeployDirectory.run(HostConfig.java:1895)
    at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:471)
    at java.util.concurrent.FutureTask.run(FutureTask.java:262)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
    at java.lang.Thread.run(Thread.java:745)
```

### 1.3.3 没有加入增量更新的配置文件

```
严重: Exception sending context initialized event to listener instance of class org.apache.solr.handler.dataimport.scheduler.ApplicationListener
java.lang.NullPointerException
    at org.apache.solr.handler.dataimport.scheduler.ApplicationListener.contextInitialized(ApplicationListener.java:90)
    at org.apache.catalina.core.StandardContext.listenerStart(StandardContext.java:4994)
    at org.apache.catalina.core.StandardContext.startInternal(StandardContext.java:5492)
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:150)
    at org.apache.catalina.core.ContainerBase.addChildInternal(ContainerBase.java:901)
    at org.apache.catalina.core.ContainerBase.addChild(ContainerBase.java:877)
    at org.apache.catalina.core.StandardHost.addChild(StandardHost.java:649)
    at org.apache.catalina.startup.HostConfig.deployDirectory(HostConfig.java:1245)
    at org.apache.catalina.startup.HostConfig$DeployDirectory.run(HostConfig.java:1895)
    at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:471)
    at java.util.concurrent.FutureTask.run(FutureTask.java:262)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
    at java.lang.Thread.run(Thread.java:745)
```

## 1.4 调试技巧

在 solr 配置的过程中，难免会碰到各种各样的问题。如，在配置过程中，出现问题，可以通过以下日志对配置进行排查。

### 1.4.1 solr 日志



			F:\solr_home\hotel\...\contrib\clustering\lib).
2017/5/25 下午4:54:28	WARN	SolrResourceLoader	Can't find (or read) directory to add to classloader: ..../dist/ (resolved as: F:\solr_home\collection1\...\dist).
2017/5/25 下午4:54:28	WARN	SolrResourceLoader	Can't find (or read) directory to add to classloader: ..../dist/ (resolved as: F:\solr_home\test\...\dist).
2017/5/25 下午4:54:28	WARN	SolrResourceLoader	Can't find (or read) directory to add to classloader: ..../dist/ (resolved as: F:\solr_home\hotel\...\dist).
2017/5/25 下午4:54:28	WARN	SolrResourceLoader	Can't find (or read) directory to add to classloader: ..../contrib/langid/lib/ (resolved as: F:\solr_home\collection1\...\contrib\langid\lib).
2017/5/25 下午4:54:28	WARN	SolrResourceLoader	Can't find (or read) directory to add to classloader: ..../contrib/langid/lib/ (resolved as: F:\solr_home\test\...\contrib\langid\lib).
2017/5/25 下午4:54:28	WARN	SolrResourceLoader	Can't find (or read) directory to add to classloader: ..../contrib/langid/lib/ (resolved as: F:\solr_home\hotel\...\contrib\langid\lib).
2017/5/25 下午4:54:28	WARN	SolrResourceLoader	Can't find (or read) directory to add to classloader: ..../dist/ (resolved as: F:\solr_home\collection1\...\dist).
2017/5/25 下午4:54:28	WARN	SolrResourceLoader	Can't find (or read) directory to add to classloader: ..../dist/ (resolved as: F:\solr_home\test\...\dist).
2017/5/25 下午4:54:28	WARN	SolrResourceLoader	Can't find (or read) directory to add to classloader: ..../dist/ (resolved as: F:\solr_home\hotel\...\dist).
2017/5/25 下午4:54:28	WARN	SolrResourceLoader	Can't find (or read) directory to add to classloader: ..../contrib/velocity/lib (resolved as: F:\solr_home\collection1\...\contrib\velocity\lib).
2017/5/25 下午4:54:28	WARN	SolrResourceLoader	Can't find (or read) directory to add to classloader: ..../contrib/velocity/lib (resolved as: F:\solr_home\test\...\contrib\velocity\lib).
2017/5/25 下午4:54:28	WARN	SolrResourceLoader	Can't find (or read) directory to add to classloader: ..../contrib/velocity/lib (resolved as: F:\solr_home\hotel\...\contrib\velocity\lib).
2017/5/25 下午4:54:28	WARN	SolrResourceLoader	Can't find (or read) directory to add to classloader: ..../dist/ (resolved as: F:\solr_home\collection1\...\dist).
2017/5/25 下午4:54:28	WARN	SolrResourceLoader	Can't find (or read) directory to add to classloader: ..../dist/ (resolved as: F:\solr_home\test\...\dist).
2017/5/25 下午4:54:28	WARN	SolrResourceLoader	Can't find (or read) directory to add to classloader: ..../dist/ (resolved as: F:\solr_home\hotel\...\dist).
2017/5/25 下午4:54:29	WARN	ManagedResource	No stored data found for /rest/managed
2017/5/25 下午4:54:29	WARN	ManagedResource	No registered observers for /rest/managed
2017/5/25 下午4:54:29	WARN	ManagedResource	No stored data found for /rest/managed
2017/5/25 下午4:54:29	WARN	ManagedResource	No registered observers for /rest/managed
2017/5/25 下午4:54:29	WARN	ManagedResource	No stored data found for /rest/managed
2017/5/25 下午4:54:29	WARN	ManagedResource	No registered observers for /rest/managed

## 1.4.2 tomcat 日志

名称	大小	类型	修改日期
catalina.2017-05-25.log	64 KB	文本文档	2017-5-25 16:54
catalina.out	0 KB	OUT 文件	2017-5-25 16:32
host-manager.2017-05-25...	0 KB	文本文档	2017-5-25 16:34
localhost.2017-05-25.log	9 KB	文本文档	2017-5-25 16:54
localhost_access_log.20...	42 KB	文本文档	2017-5-25 17:02
manager.2017-05-25.log	0 KB	文本文档	2017-5-25 16:34

## 1.5 solr 应用

### 1.5.1 爱旅行项目搜索分析

以上我们已经成功的搭建了solr的服务，那么solr服务如何集成到我们项目当中，首先我们来分析爱旅行项目中的酒店搜索的需求：

- 搜索条件包括目的地、入住时间、退房时间、关键词、位置、价格、酒店级别、酒店特色。
- 关键词和目的地需要对多个字段进行检索，比如酒店名称、酒店地址、酒店描述等。
- 酒店搜索关联多个数据库表，这其中包括酒店表、区域表、酒店区域关联表、房间表、

特色表、酒店特色关联表、酒店评论表、图片表。

- 搜索过程中包括一些复杂搜索，包括平均分计算、评论人数统计、点评人数统计、库存计算等。
- 搜索过程中，需要对关键词。

如果我们使用传统的技术来实现以上的酒店搜索的功能会有以下问题：

- 传统的like关键词匹配大字段效率低下。
- 每次查询都执行复杂SQL，数据库压力很大。

您所在的位置：酒店预订>北京酒店

目的地：北京市

入住：2016-12-08

退房：2016-12-20

关键词：

搜索

位置：☒ 天安门、王... ☐ 中关村、五... ☐ 西单、金融... ☐ 首都机场 ☐ 亚运村

价格：☐ ¥150以下 ☒ ¥150-300 ☐ ¥301-450 ☐ ¥450以上

星级：☐ 二星级及以下 ☒ 三星级/舒适 ☐ 四星级/高档 ☐ 五星级/高档

特色：☐ 休闲度假 ☒ 青年旅社 ☐ 精品酒店 ☐ 商务出行 ☐ 会

8 家酒店 您已选择：北京 x

最受欢迎 评分 价格 星级

☐ 立即确认 ☒ 可订 ☐ 促销 ☐ 在线付 ☐ 闪住 ☐ 钟点房



北京盘古七星酒店

☒ 商务出行 ☒ 会议酒店 ☒ 精品酒店

朝阳区北四环中路27号，盘古大观，近奥运村国家体育馆。【亚运村、奥体中心地区】

交通地图 周边街景

4.7 /5分

98%用户推荐  
源自1653位住客点评

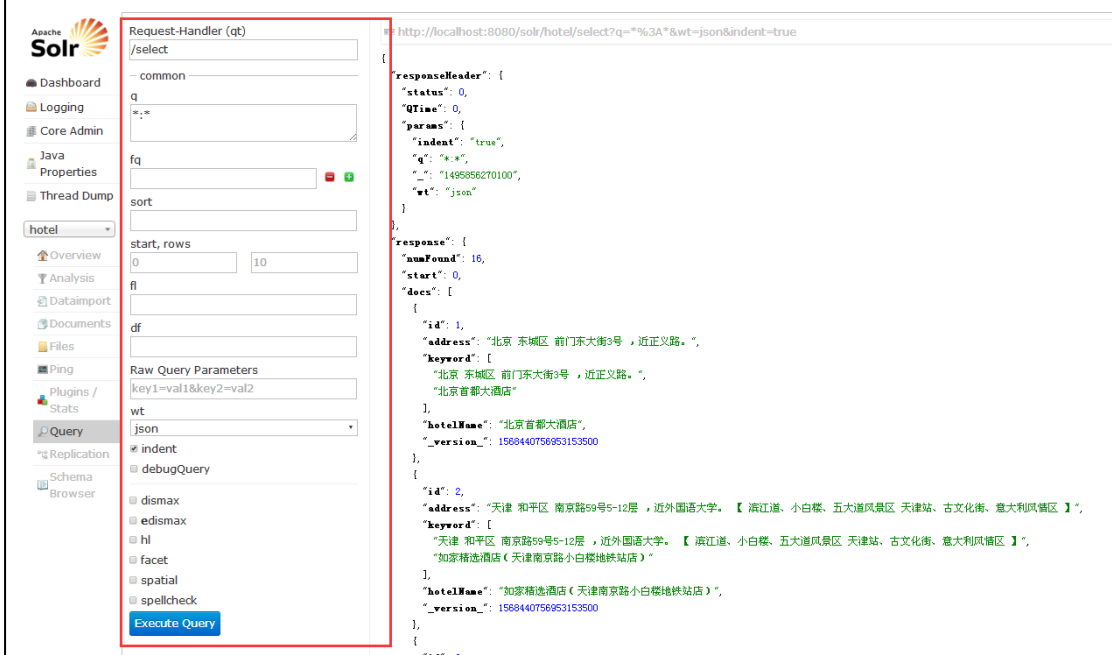
¥2059起

查看详情

## 1.5.2 solr 基本查询语法

- q – 查询字符串。
- fl – 指定返回那些字段内容，用逗号或空格分隔多个。
- start – 返回第一条记录在完整找到结果中的偏移位置，0 开始，一般分页用。
- rows – 指定返回结果最多有多少条记录，配合 start 来实现分页。
- sort – 排序，格式：sort=<field name>+<desc|asc>[,<field name>+<desc|asc>]...。示例：  
(inStock desc, price asc) 表示先 “inStock” 降序，再 “price” 升序，默认是相关性降序。
- wt – (writer type)指定输出格式，可以有 xml, json, php, phps, 后面 solr 1.3 增加的，要用通知我们，因为默认没有打开。
- fq – (filter query) 过滤查询，作用：在 q 查询符合结果中同时是 fq 查询符合的，例

如: q=mm&fq=date\_time:[20081001 TO 20091031], 找关键字 mm, 并且 date\_time 是 20081001 到 20091031 之间的。



The screenshot shows the Apache Solr Admin UI. On the left, the 'Query' tab is selected. The main panel displays the 'Request-Handler (qt)' section with a query 'q=mm&fq=date\_time:[20081001 TO 20091031]'. The response shows search results for 'mm'.

### 1.5.3 solr 多字段匹配

针对关键词多字段的搜索, solr中提供了相应的检索机制。

1. 在hotel/conf/schema.xml文件中新增filed字段存储多字段的值(红色内容为新增部分)

```
<field name="_version_" type="long" indexed="true" stored="true"/>

<field name="id" type="long" indexed="true" stored="true"/>

<field name="hotelName" type="text_ik" indexed="true" stored="true"/>

<field name="address" type="text_ik" indexed="true" stored="true"/>

<field name="keyword" type="text_ik" indexed="true" stored="true" multiValued="true"/>

<copyField source="hotelName" dest="keyword"/>

<copyField source="address" dest="keyword"/>
```

2. 重新执行数据导入, 并查询, 出现以下结果则多字段配置正确。

```

"response": {
  "numFound": 16,
  "start": 0,
  "docs": [
    {
      "id": 1,
      "address": "北京 东城区 前门东大街3号 ,近正义路。",
      "keyword": [
        "北京 东城区 前门东大街3号 ,近正义路。",
        "北京首都大酒店"
      ],
      "hotelName": "北京首都大酒店",
      "_version_": 1568519626969579500
    },
    {
      "id": 2,
      "address": "天津 和平区 南京路59号5-12层 ,近外国语大学。【滨江道、小白楼、五大道风景区 天津站、古文化街、意大利风情区】",
      "keyword": [
        "天津 和平区 南京路59号5-12层 ,近外国语大学。【滨江道、小白楼、五大道风景区 天津站、古文化街、意大利风情区】",
        "如家精选酒店(天津南京路小白楼地铁站)"
      ],
      "hotelName": "如家精选酒店(天津南京路小白楼地铁站)",
      "_version_": 1568519626969579500
    },
    {
      "id": 3,
      "address": "北京 朝阳区 紫玉东路1号 ,近北辰购物中心。",
      "keyword": [
        "北京 朝阳区 紫玉东路1号 ,近北辰购物中心。",
        "北京紫玉度假酒店"
      ],
      "hotelName": "北京紫玉度假酒店",
      "_version_": 1568519626969579500
    },
    {
      "id": 4,

```

### 3. 关键词搜索，查看查询结果是否正确

Request-Handler (qt)

/select

common

q

keyword:北京

fq

sort

start, rows

0 10

fl

df

Raw Query Parameters

key1=val1&key2=val2

wt

json

indent

debugQuery

dismax

edismax

hl

facet

spatial

spellcheck

Execute Query

http://localhost:8080/solr/hotel/select?q=keyword%3A%E5%8C%97%E4%BA%AC&wt=json&ind

```

{
  "responseHeader": {
    "status": 0,
    "QTime": 515,
    "params": {
      "indent": "true",
      "q": "keyword:北京",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 6,
    "start": 0,
    "docs": [
      {
        "id": 1,
        "address": "北京 东城区 前门东大街3号 ,近正义路。",
        "keyword": [
          "北京 东城区 前门东大街3号 ,近正义路。",
          "北京首都大酒店"
        ],
        "hotelName": "北京首都大酒店",
        "_version_": 1568519626969579500
      },
      {
        "id": 3,
        "address": "北京 朝阳区 紫玉东路1号 ,近北辰购物中心。",
        "keyword": [
          "北京 朝阳区 紫玉东路1号 ,近北辰购物中心。",
          "北京紫玉度假酒店"
        ],
        "hotelName": "北京紫玉度假酒店",
        "_version_": 1568519626969579500
      },
      {

```

## 1.5.4 solrj 集成 solr 应用

以上的操作中，我们根据业务，完善了solr服务器，接下来我们将学习，如何把solr项目集成到项目中来。solr本身提供了对外调用的Http接口，利用Http请求可以直接从solr中获取数据。

为了方便Java程序员调用solr，Apache提供了基于solr操作的solrj程序包。

### 1. 下载solrj程序包

- 系统中选用的solrj版本:5.3.1
- solrj下载: maven方式

```
<dependency>

    <groupId>org.apache.solr</groupId>

    <artifactId>solr-solrj</artifactId>

    <version>5.3.1</version>

</dependency>
```

### 2. 在程序中创建solr查询的接收对象（省略get、set）

```
import org.apache.solr.client.solrj.beans.Field;
import java.io.Serializable;
public class ItripHotel implements Serializable {
    @Field
    private Long id;
    @Field
    private String hotelName;
    @Field
    private String address;
```

引入field注解

于solr中的返回数据的字段名称保持一致

### 3. 创建main程序调用solr应用

```

public static void main(String[] args) throws IOException, SolrServerException {
    //连接URL
    String url = "http://localhost:8080/solr/hotel/";
    HttpSolrClient httpSolrClient=new HttpSolrClient(url);
    httpSolrClient.setParser(new XMLResponseParser()); // 设置响应解析器
    httpSolrClient.setConnectionTimeout(500); // 建立连接的最长时间
    SolrQuery solrQuery = new SolrQuery(); //新建查询
    solrQuery.setQuery("keyword:北京");
    QueryResponse queryResponse = httpSolrClient.query(solrQuery);
    List<ItripHotel> hotelList= queryResponse.getBeans(ItripHotel.class);
    for (ItripHotel hotel:hotelList){
        System.out.println(hotel.getHotelName());
    }
}

```

本地solr访问地址  
ip: 端口号/solr/模块名

执行Http请求的客户端

用于设置查询的对象

指定返回的对象的类型

## 1.6 上传服务器

在本地配置好solr和solr\_home后，可以将装载有solr的tomcat和solr\_home文件夹直接拷贝到Linux服务器。此处注意要修改solr中的web.xml的solr\_home的地址修改为solr\_home在服务器的实际目录。

详细部署文档请参考《itrip安装配置及备份方案》。

