Solr的安装

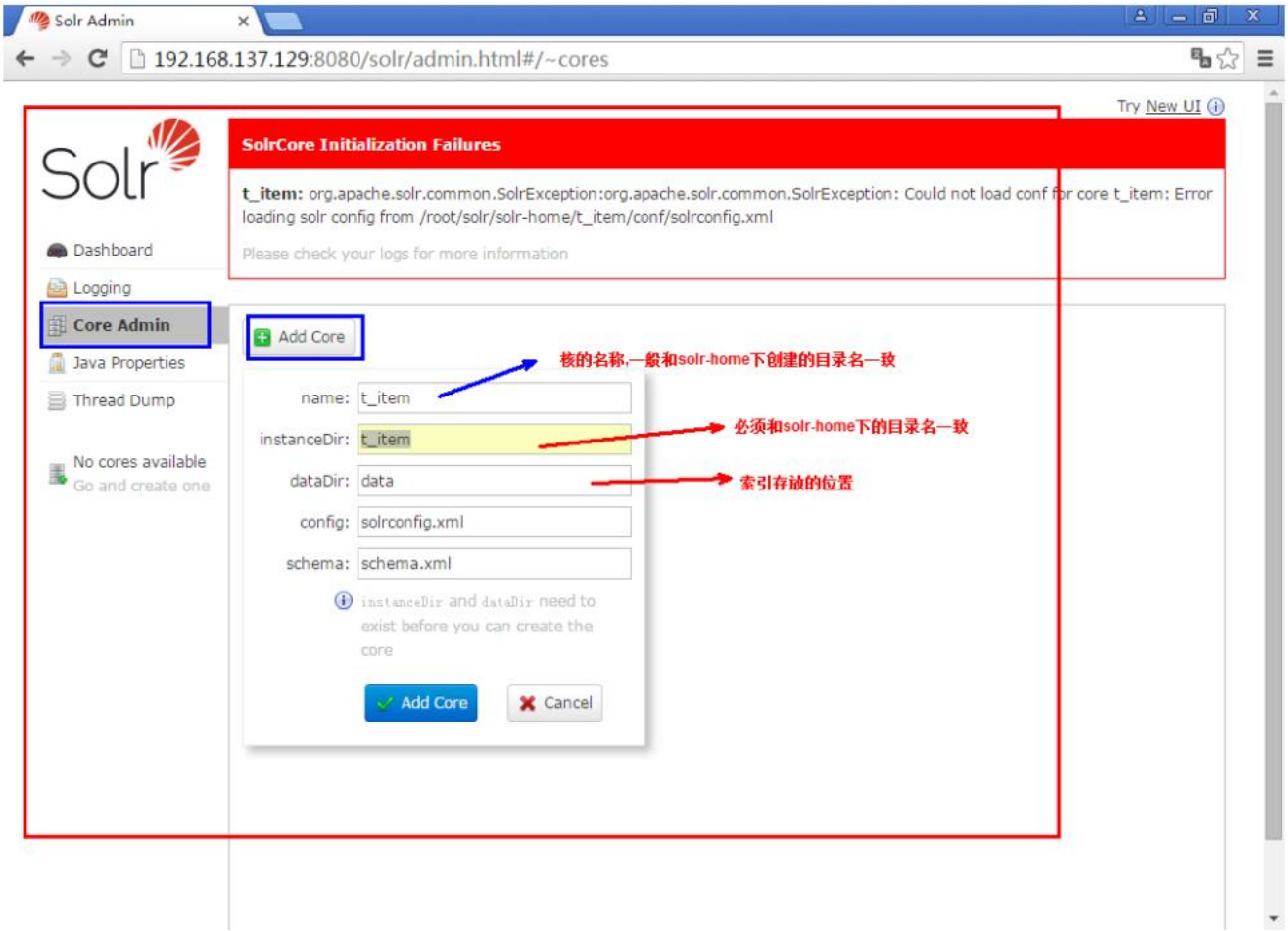
```
1.环境要求
        jdk1.7+tomcat8+solr5.5.0
2.将以上的软件包上传到服务器
3.
        安装tomcat8(解压文件)
                tar -zxvf apache-tomcat-8.5.24.tar.gz
4.
        解压solr
                tar -zxvf solr-5.5.0.tgz

    1.将solr的服务拷贝到tomcat的webapps下
    cp -r  solr-5.5.0/server/solr-webapp/webapp  tomcat8/webapps/solr
    2.需要将solr需要日志的jar拷贝到tomcat8/webapp/solr/WEB-INF/lib
    cp -r solr-5.5.0/server/lib/ext/*.jar tomcat8/webapps/solr/WEB-INF/lib
    3.需要将日志的log4j配置文件拷贝到项目中
    mkdir -p tomcat8/webapps/solr/WEB-INF/classes
    cp -r solr-5.5.0/server/resource/log4j.properties tomcat8/webapps/WEB-INF/classes
    4.配置solrHome（存放索引）
    mkdir solr-home
    vi tomcat8/webapps/solr/WEB-INF/web.xml
    打开如下配置
    <env-entry>
       <env-entry-name>solr/home</env-entry-name>
       <env-entry-value>/put/your/solr/solr-home</env-entry-value>
       <env-entry-type>java.lang.String</env-entry-type>
    </env-entry>
    拷贝solr的配置文件到solr-home
    cp -r solr-5.5.0/server/solr/*  /root/solr/solr-home

  5.启动tomcat
        tomcat8/bin/startup.sh

  6.http://192.168.137.129:8080/solr/admin.html
```

核心概念

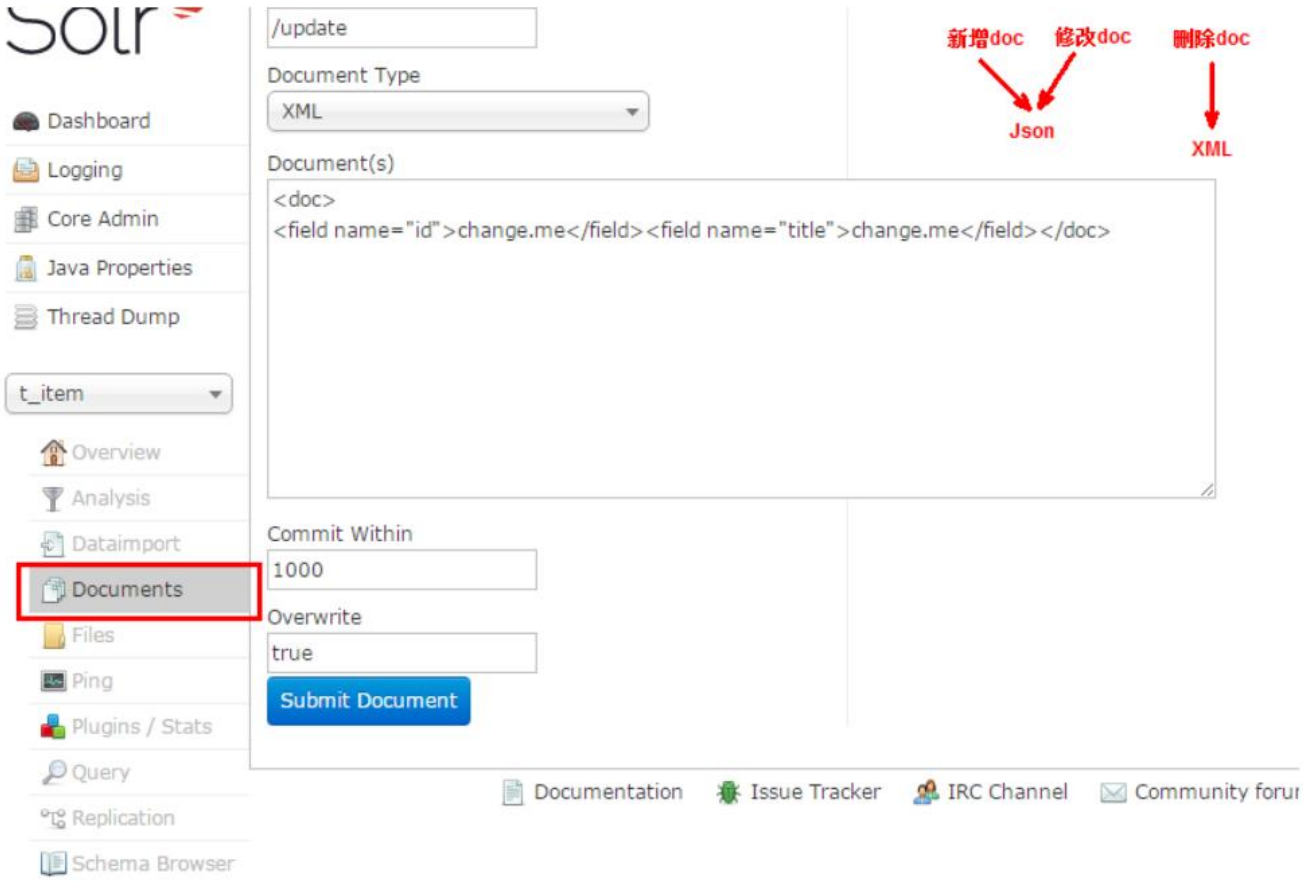
```
1.solr服务器  -->理解为数据库
2.核          -->表
3.field       -->表中字段
```

创建核

```
1.在solr-home下创建一个目录（目录名随意,例如t_item）
2.拷贝核中所需要的配置文件(solr-home下configsets里面)
                cp -r configsets/sample-techproducts-configs/*  t_item
3.主界面配置（见下图）
```


Solr Admin
192.168.137.129:8080/solr/admin.html#/~cores
Try New UI
Solr
SolrCore Initialization Failures
t_item: org.apache.solr.common.SolrException:org.apache.solr.common.SolrException: Could not load conf for core t_item: Error loading solr config from /root/solr/solr-home/t_item/conf/solrconfig.xml
Please check your logs for more information
Dashboard
Logging
Core Admin
Java Properties
Thread Dump
No cores available
Go and create one
Add Core
核的名称,一般和solr-home下创建的目录名一致
name: t_item
instanceDir: t_item
必须和solr-home下的目录名一致
dataDir: data
索引存放的位置
config: solrconfig.xml
schema: schema.xml
instanceDir and dataDir need to exist before you can create the core
Add Core
Cancel

系统界面操作



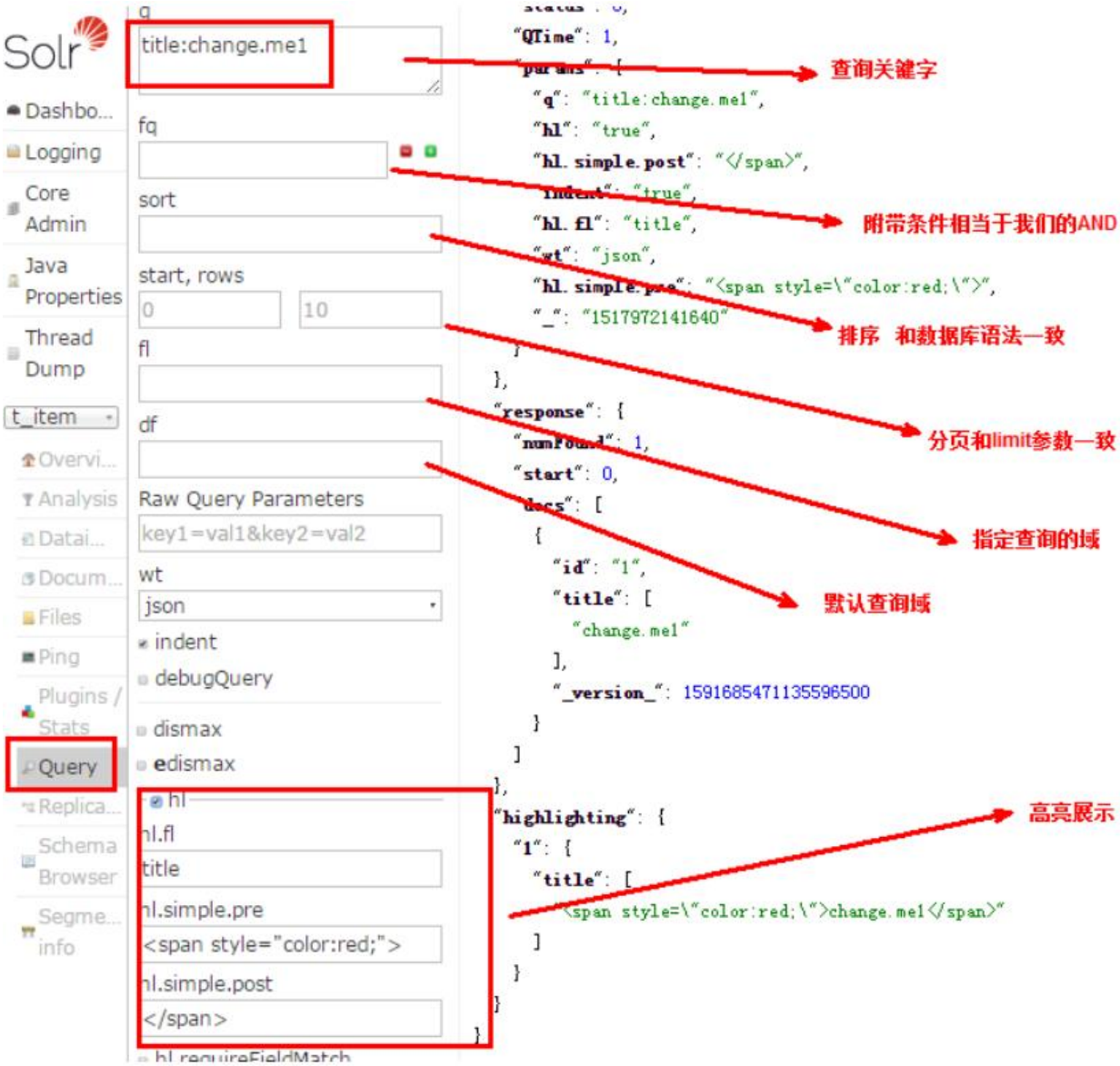
1.新增和修改（id在索引库中存在就是修改，不存在就是新增）

```
{"id":"change.me","title":"change.me"}
```

> id必须存在，可以理解为主键
>
> key必须先配置（managed-schema）再使用

```
2.删除使用xml
<delete>
    <id>change.me</id>
</delete>
<commit/>
```

3.查询(见下图)



java中操作Solr

```
1.添加依赖
<dependency>
  <groupId>org.apache.solr</groupId>
  <artifactId>solr-solrj</artifactId>
  <version>5.5.0</version>
</dependency>
2.使用API
    1.获取和服务器连接
        String baseURL = "http://192.168.137.129:8080/solr/t_item"; //服务器地址，最后一个表示核
       SolrClient solrClient = new HttpSolrClient(baseURL);
    2.调用
        solrClient中的增删改查操作
```

Solr添加域

```
1.需要在solr-home，找到对应的核的目录，conf/managed-schema
        vi conf/managed-schema
        <!--配置自定义的域-->
<field name="content_ik" type="text_ik" indexed="true" stored="true"/>
<fieldType name="text_ik" class="solr.TextField">
  <analyzer class="org.wltea.analyzer.lucene.IKAnalyzer"></analyzer>
</fieldType>

2.将IK分词器拷贝到tomcat8/webapps/solr/WEB-INF/lib
        cp ik-analyzer-5.3.0.jar  tomcat8/webapps/solr/WEB-INF/lib

3.重启服务器


IK分词器需要添加扩展词典
    将配置文件IKAnalyzer.cfg.xml和词典上到 tomcat8/webapps/solr/WEB-INF/classes

```

在工作中使用Solr

```
1.创建业务域
     <!--业务域类型定义-->
   <field name="content_ik_nostore" type="text_ik" indexed="true" stored="false"/>
   <fieldType name="text_ik" class="solr.TextField">
      <analyzer class="org.wltea.analyzer.lucene.IKAnalyzer">
      </analyzer>
   </fieldType>

   <!--业务域名-->
   <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_created" type="date" indexed="true" stored="true"/>

   <!--拷贝域-->
   <field name="item_keywords" type="text_ik" indexed="true" stored="true"
multiValued="true"/>
   <copyField source="item_title" dest="item_keywords"/>
   <copyField source="item_sell_point" dest="item_keywords"/>

2.加载数据
    使用加载的数据，包裹以上域名包装为SolrInputDocument
    调用SolrClinet中add方法添加Solr库中
```

java调用solr服务器的相关代码

managed-schema配置文件修改

solr-home/某核心/conf/managed-schema文件中添加的配置

```
<!-- 自定义 field -->
<field name="content_ik" type="text_ik" indexed="true" stored="true" />

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

<!-- 自定义 fieldType -->
<fieldType name="text_ik" class="solr.TextField" >
  <analyzer class="org.wltea.analyzer.lucene.IKAnalyzer" ></analyzer>
</fieldType>

<!-- copyField的使用 -->
<field name="keywords_ik" type="text_ik" indexed="true" stored="true" multiValued="true" />
<copyField source="title_ik" dest="keywords_ik" />
<copyField source="info_ik" dest="keywords_ik" />
```

Properties配置文件:

```
solr.url=http://localhost:8181/solr/t_user
solr.connectionTimeout=500
```

在junit测试类中写的代码

初始化

```
//相关初始化
private Properties p = new Properties();
private HttpSolrClient solrClient;
//private SolrClient solrClient;//使用此类时无法设置connectionTimeout等属性
@Before
public void before(){
  try {
    p.load(SolrTest.class.getClassLoader().getResourceAsStream("solrclient.properties"));
  } catch (IOException e) {
    e.printStackTrace();
  }
  //HttpSolrClient:启动web服务器使用的,通过http请求的
  solrClient = new HttpSolrClient(p.getProperty("solr.url"));
  int connectionTimeout = Integer.parseInt(p.getProperty("solr.connectionTimeout"));
  solrClient.setConnectionTimeout(connectionTimeout);
}
```

高亮查询

```java
public void queryHighlight(){
  //关键字查询    域:值      *:*匹配所有
  //想完全匹配必须要用双引号
  //String q = "title_ik:是  AND info_ik:\"中国人\"";
  String q = "keywords_ik:国人";//keywords_ik是copyField
  SolrQuery sq = new SolrQuery();
  sq.setQuery(q);
  sq.setHighlight(true);//开启高亮显示
  sq.setHighlightSimplePre("<span color='red' >");//高亮格式开头
  sq.setHighlightSimplePost("</span>");//高亮格式结尾
  //sq.setStart(0).setRows(100);//分页

  //hl.fl表示属性高亮, String... values表示哪些field需要高亮, copyField无法高亮
  sq.setParam("hl.fl", "title_ik", "info_ik", "keywords_ik");
  try {
    QueryResponse response = solrClient.query(sq);

    SolrDocumentList results = response.getResults();
    System.out.println("总数:" + results.getNumFound());
    int size = results.size();
    System.out.println("当前总数:" + size);
    for (int i = 0; i < size; i++) {
      SolrDocument doc = results.get(i);
      System.out.println(doc);
      String id = (String) doc.get("id");
      //在solr这里对需要加高亮的字段必须要在索引中的store=true才行
      //获取对应的document高亮的key=value
      Map<String, List<String>> map = response.getHighlighting().get(id);
      List<String> titles = map.get("title_ik");
      List<String> infos = map.get("info_ik");
      if(null == titles){
        titles = new ArrayList<String>();
        String title = (String) doc.get("title_ik");
        if(null != title){
          titles.add(title);
        }
      }
      if(null == infos){
        infos = new ArrayList<String>();
        String info = (String) doc.get("info_ik");
        if(null != info){
          infos.add(info);
        }
      }
      System.out.println(id + " ,title_ik= " + titles + " ,info_ik= " + infos + "
,=keywords_ik= " + doc.get("keywords_ik"));
    }
  } catch (SolrServerException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }

}
```

添加**Document**

```
public void addDoc(SolrInputDocument doc){
  //id是必填的,并且是String类型的
  //id是唯一的主键,当多次添加的时候,最后添加的相同id会覆盖前面的域
  try {
    solrClient.add(doc);
    solrClient.commit();
  } catch (SolrServerException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

修改**Document**

```
public void updateDocById(SolrInputDocument doc){
  //修改实际上还是增加索引,只不过指定id,把相同id的filed覆盖掉
  try {
    solrClient.add(doc);
    solrClient.commit();
  } catch (SolrServerException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

删除**Document**

```java
public void deleteByIds(List<String> ids){
  //根据多个id删除多个document
  //List<String> ids = new ArrayList<String>();
  try {
    solrClient.deleteById(ids);
    solrClient.commit();
  } catch (SolrServerException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
}

public void deleteById(String id){
  //根据id删除document
  try {
    solrClient.deleteById(id);
    solrClient.commit();
  } catch (SolrServerException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
}

public void deleteByQuery(String query){
  //删除查询的结果
  //String query = "title_ik:人";
  try {
    solrClient.deleteByQuery(query);
    solrClient.commit();
  } catch (SolrServerException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```