Solr 课程内容教案

1、Solr概述

1.1、什么是 Solr

Solr 是一个高性能，采用 Java5 开发，基于 Lucene 的全文搜索服务器。同时对其进行了扩展，提供了比 Lucene 更为丰富的查询语言，同时实现了可配置、可扩展并对查询性能进行了优化，并且提供了一个完善的功能管理界面，是一款非常优秀的全文搜索引擎。

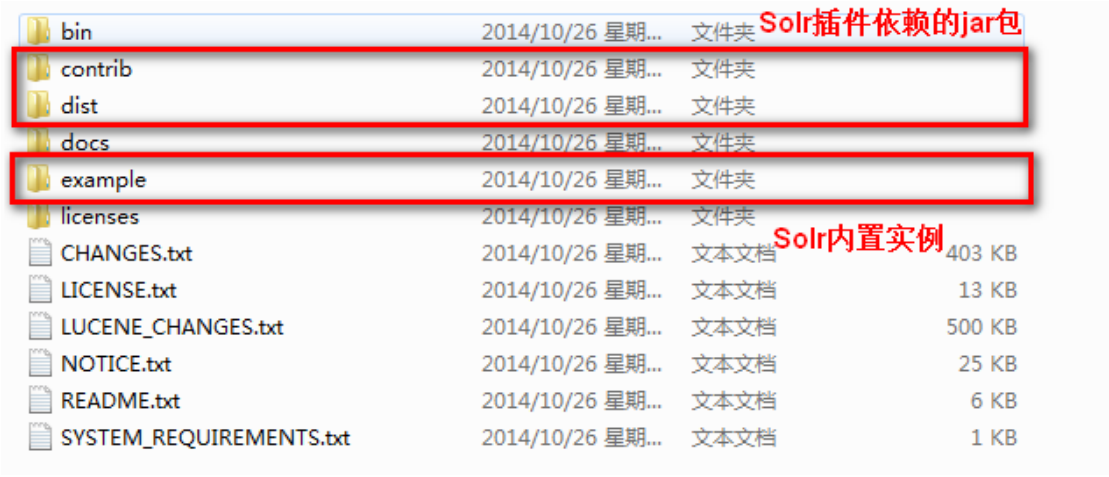
1.2、Solr 的下载

Solr 是 Apache 下的一个开源项目，由于是基于 Lucene 的一个实现产品，因此将 Solr 放在 Lucene 项目当中，官方网站为 :http://lucene.apache.org/solr/。最新版本为 7.1.0，我们这里使用于 Lucene 相匹配的 4.10.2 版本进行我们的学习。

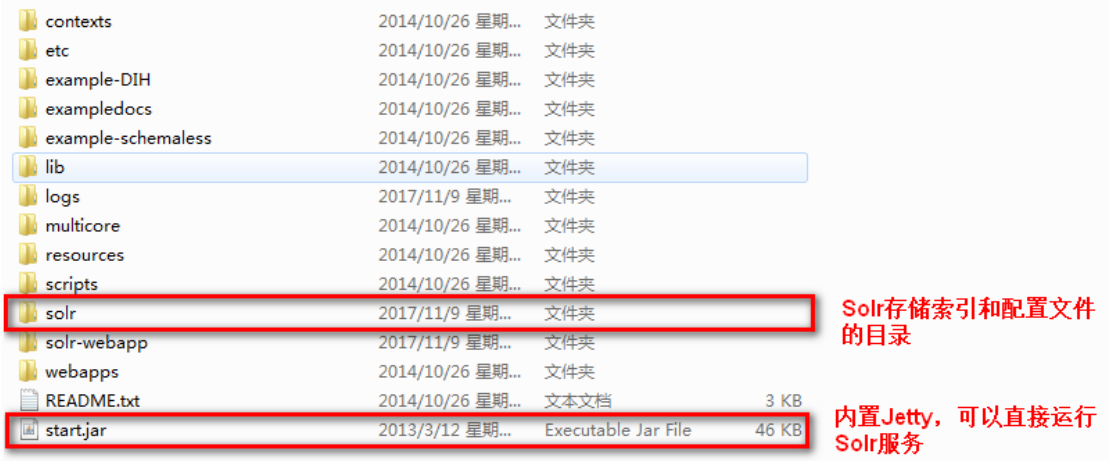
Solr 的历史下载地址为：http://archive.apache.org/dist/lucene/solr/，我们可以从中下载 4.10.2 版本，我们这里已经下载完成了。

1.3、Solr 的下载

将下载的 Solr 文件解压缩即可获取 Solr 的内容，我们看到目录结构如下。



在 example 中包含 Solr 的完整案例，我们需要了解。



这里我们需要注意的是：

Solr 目录用来存放 Solr 的 Core，主要包含索引和配置文件。

solr-webapp 目录中管理者 Jetty 运行的 Solr 服务去程序。

Webapps 目录中的 solr.war 为 solr 服务的文本应用程序。

2、运行 Solr 服务

Solr 实际上是一个 web 服务器应用程序，因此要启动 Solr 需要文本服务器，而在 Solr 实例代码中已经直接集成了 Jetty 服务器，我们可以直接使用，也可以单独配置和使用 Tomcat 服务器来启动 Solr 服务。

2.1、启动 Solr 服务的方式

2.1.1、使用 Solr 内置的 Jetty 服务器启动 Solr

使用内置的 Jetty 来启动 Solr 服务器只需要在 example 目录下，执行 start.jar 程序即可，我们可以直接执行命令：java –jar start.jar。

当服务启动后，默认发布在 8983 端口，所以可以访问该端口来访问 Solr 服务。

2.1.2、将 Solr 部署到 Tomcat 中

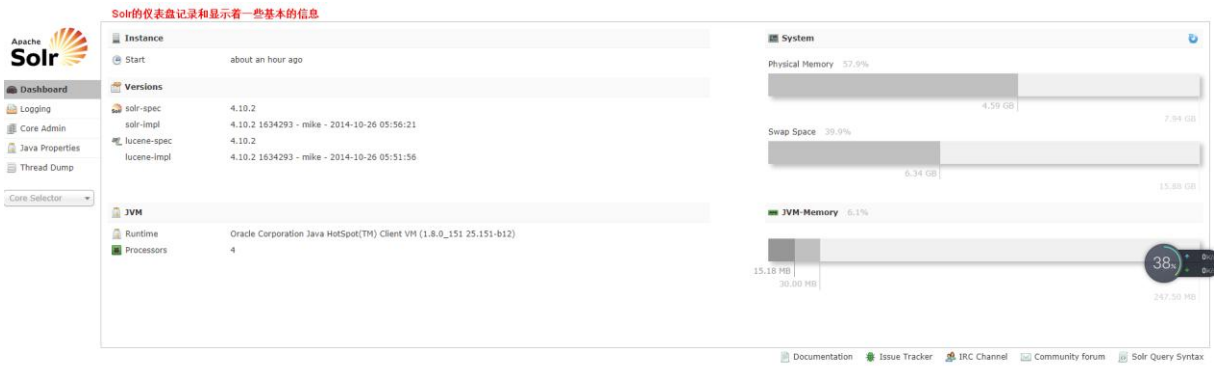
在 Tomcat 中部署 Web 服务，将 solr-4.10.2/example/webapps/solr.war 复制到自己的 tomcat/webapps 目录中，进行并解压。

将 Solr 用到的 jar 包拷贝到项目的 lib 目录下，在启动前还需要设置 solr.solr.home 启动参数，在 catalina.bat 文件添加配置 :set "JAVA_OPTS=-Dsolr.solr.home=d:/test/solr"。

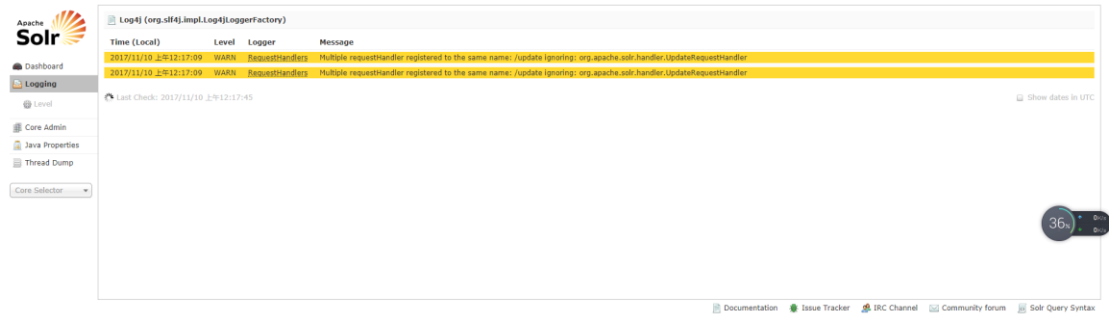
启动 Tomcat 则 Solr 的服务器发布就完成了。

2.2、Solr 管理页面

2.2.1、DashBoard



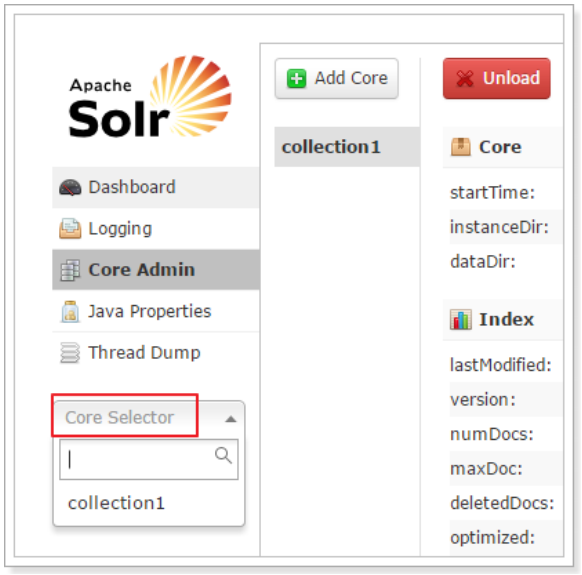
2.2.2、Logging



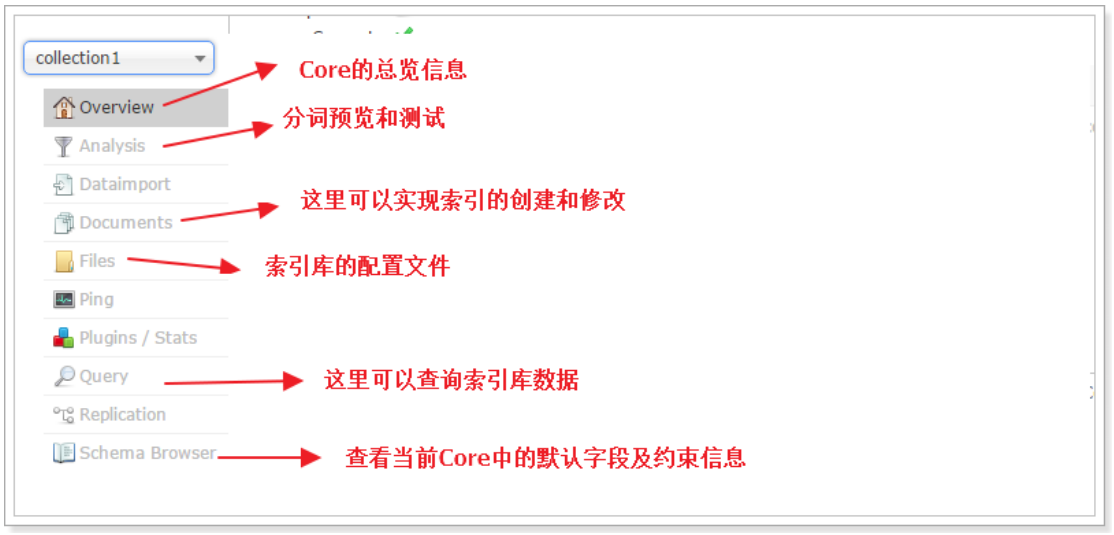
2.2.3、Core Admin

在 Solr 中，每一个 Core，代表一个索引库，里面包含索引数据及其信息。Solr 中可以拥有多个 Core ,也就同时管理多个索引库 !就像在 MySQL 中可以有多个 database 一样。

2.2.4、CoreSelector

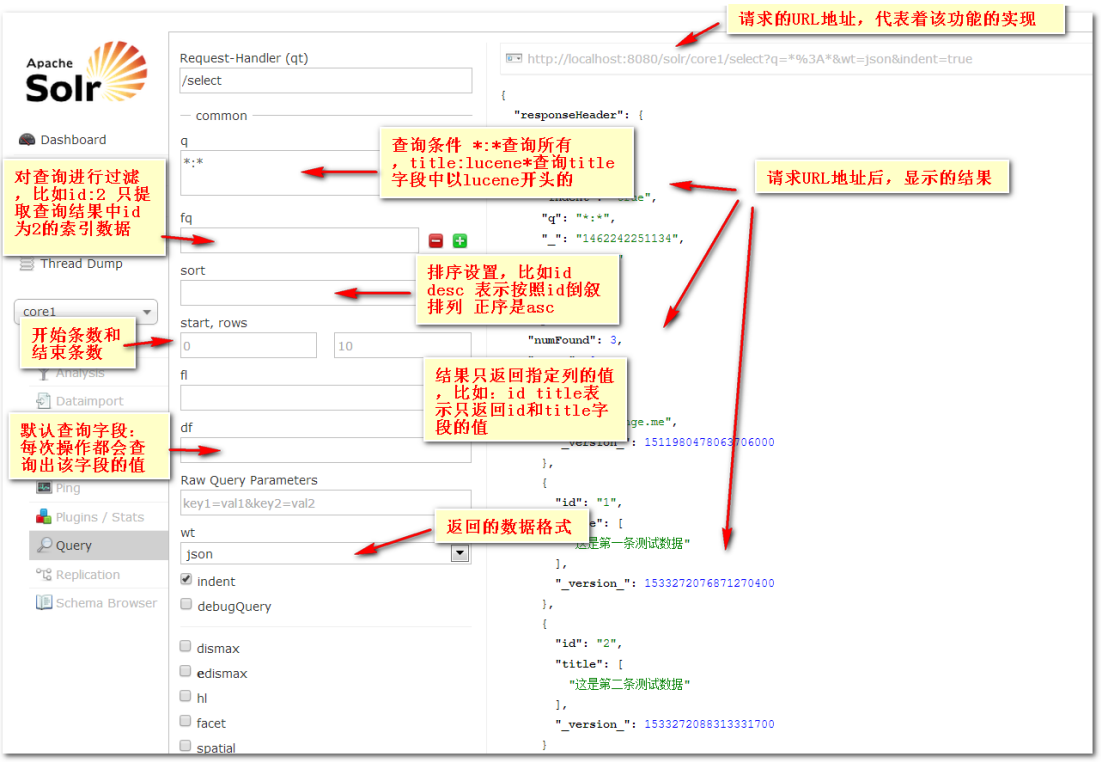


选择一个 Core，可以进行更详细的操作：



2.2.4.1、通过 Solr 管理界面添加索引数据

2.2.4.2、通过 Solr 管理界面查询索引数据



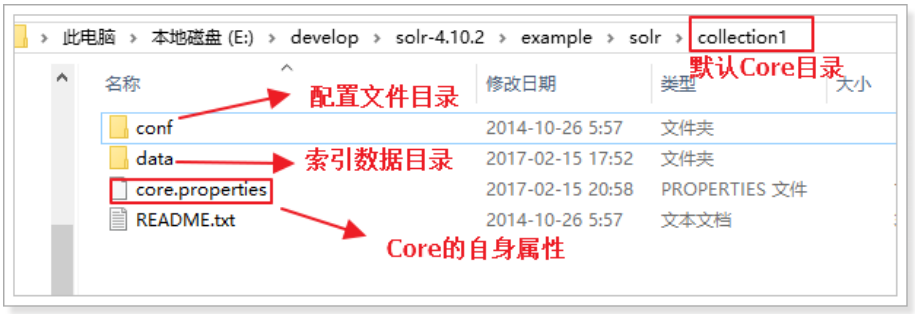
3、Solr中的Core详解

3.1、 Core 的概念

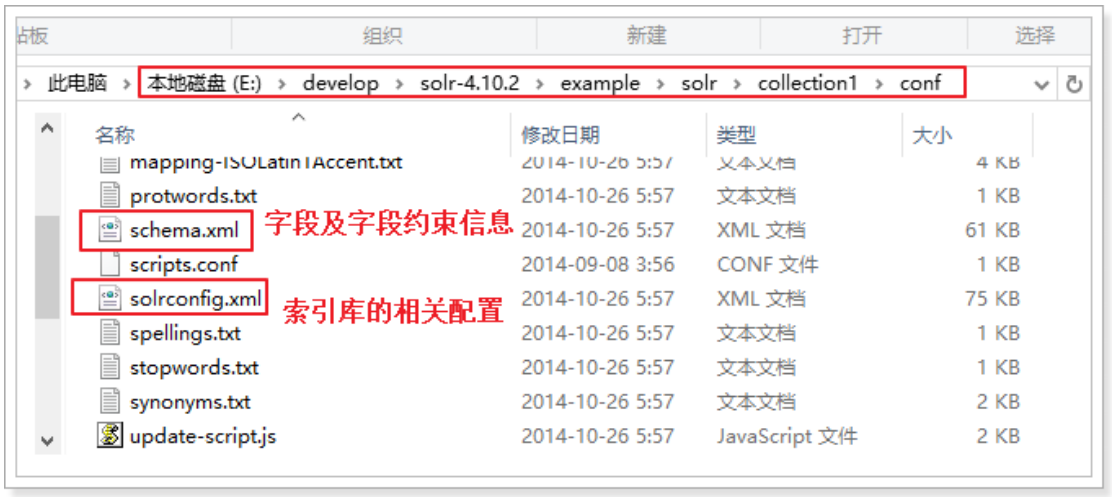
在 Solr 中一个 core 就是一个索引库，里面包含索引信息和配置文件。

3.2、 目录结构

Core 中有两个重要目录：conf 和 data



conf 目录中有两个非常重要的配置文件：schema.xml 和 solrconfig.xml

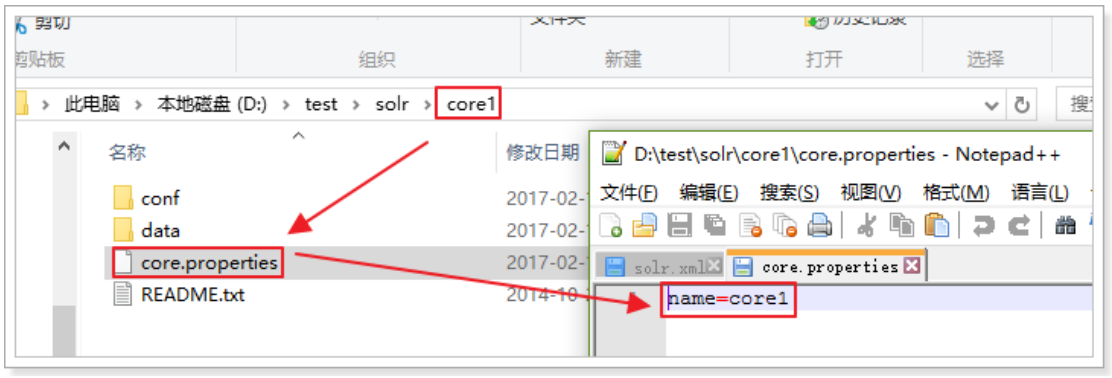


3.3、Core 的配置文件详解

3.3.1、core.properties

3.3.1.1、修改 core 名称

Core 的属性文件，记录当前 core 的名称、索引位置、配置文件名称等信息。一般要求 Core 名称跟 Core 的文件夹名称一致！这里都是 Collection1，我们可以手动修改这个属性。



此时重启 Tomcat，可以看到 core 的名字已经改变！

3.3.1.2、添加多个 core

在 solr 目录下创建新的文件夹 core2,作为新的 core 目录，创建 conf 目录和 data 目录，并且创建文件 core.properties，从 core1/conf 目录下复制配置文件 core2/conf/下。

3.3.2、schema.xml

Solr 中会提前对文档中的字段进行定义，并且在 schema.xml 中对这些字段的属性进行约束，例如：字段数据类型、字段是否索引、是否存储、是否分词等等。

3.3.2.1、通过 Field 字段定义字段的属性信息

属性及含义：

name：字段名称 type：字段类型，指向的是本文件中的<fieldType>标签 indexed：是否创建索引 stored：是否被存储 multiValued：是否可以有多个值，如果字段可以有多个值，设置为 true

注意：在本文件中，有两个字段是 Solr 自带的字段，绝对不要删除：_version 节点和 _root 节点。

3.3.2.2、通过 FieldType 指定数据类型

属性及含义：

name：字段类型的名称，可以自定义，<field>标签的 type 属性可以引用该字段，来指定数据类型 class：字段类型在 Solr 中的类。StrField 可索引不可分词。TextField 字段可索引，可以分词，所以需要指定分词器 <analyzer>：这个子标签用来指定分词器

3.3.2.3、唯一主键

Lucene 中本来是没有主键的。删除和修改都需要根据词条进行匹配。而 Solr 却可以设置一个字段为唯一主键，这样增删改操作都可以根据主键来进行。

3.3.2.4、动态字段

动态字段可以动态匹配，一般使用比较少，使用 dynamicField 来定义。

3.3.3、solrconfig.xml

这个配置文件主要配置跟索引库和请求处理相关的配置。

3.3.3.1、<lib/>标签

用途：配置插件依赖的 jar 包

注意事项：

- 如果引入多个 jar 包，要注意包和包的依赖关系，被依赖的包配置在前面
- 这里的 jar 包目录如果是相对路径，那么是相对于 core 所在目录

```
<lib dir="../../../contrib/extraction/lib" regex=".*\.jar" />
<lib dir="../../../dist/" regex="solr-cell-\d.*\.jar" />

<lib dir="../../../contrib/clustering/lib/" regex=".*\.jar" />
<lib dir="../../../dist/" regex="solr-clustering-\d.*\.jar" />

<lib dir="../../../contrib/langid/lib/" regex=".*\.jar" />
<lib dir="../../../dist/" regex="solr-langid-\d.*\.jar" />

<lib dir="../../../contrib/velocity/lib" regex=".*\.jar" />
<lib dir="../../../dist/" regex="solr-velocity-\d.*\.jar" />
```

3.3.3.2、<requestHandler/>标签

用途：配置 Solr 处理各种请求（搜索/select、更新索引/update、等）的各种参数

主要参数：

- name：请求类型，例如：select、query、get、update
- class：处理请求的类
- initParams：可选。引用<initParams>标签中的配置
- <lst name="defaults">：定义各种缺省的配置，比如缺省的 parser、缺省返回条数

4、SolrJ的使用

4.1、概述

SolrJ 是 Apache 官方提供的一套 Java 开发的，访问 Solr 服务的 API，通过这套 API 可以让我们的程序与 Solr 服务产生交互，让我们的程序可以实现对 Solr 索引库的增删改查。

添加依赖：

```
<!-- Junit单元测试 -->
      <dependency>
          <groupId>junit</groupId>
          <artifactId>junit</artifactId>
          <version>4.12</version>
      </dependency>
      <dependency>
          <groupId>org.apache.solr</groupId>
          <artifactId>solr-solrj</artifactId>
          <version>4.10.2</version>
      </dependency>
      <!-- Solr底层会使用到slf4j日志系统 -->
      <dependency>
          <groupId>org.slf4j</groupId>
          <artifactId>slf4j-log4j12</artifactId>
          <version>1.7.22</version>
      </dependency>
      <dependency>
          <groupId>commons-logging</groupId>
          <artifactId>commons-logging</artifactId>
          <version>1.2</version>
      </dependency>
```

4.2、使用 SolrJ 添加或修改索引库数据

4.2.1、以 Document 形式添加或修改数据

```
@Test
public void testInsertIndexByDocument() throws Exception {
   //创建服务器对象
   HttpSolrServer server = new
HttpSolrServer("http://localhost:8983/solr/collection2");

   //创建文档对象
```

```
    SolrInputDocument document = new SolrInputDocument();

    //添加索引数据
    document.addField("id", "1");
    document.addField("title", "这是第一条测试数据");

    //添加文档
    server.add(document);

    //提交请求，如果ID对应的数据存在则进行更新操作，如果不存在则创建索引
    server.commit();
}
```


```
{
  "responseHeader": {
    "status": 0,
    "QTime": 1,
    "params": {
      "q": "*:*",
      "indent": "true",
      "wt": "json",
      "_": "1510248319031"
    }
  },
  "response": {
    "numFound": 1,
    "start": 0,
    "docs": [
      {
        "id": "1",
        "title": [
          "这是第一条测试数据"
        ],
        "_version_": 1583610064790356000
      }
    ]
  }
}
```


4.2.2、使用注解和 JavaBean 添加或修改数据

创建 JavaBean，并且用注解标明要添加到索引库的字段，直接通过 SolrServer 添加 JavaBean。

```
@Test
public void testInsertIndexByBean() throws Exception {
    //创建服务器对象
    HttpSolrServer server = new
HttpSolrServer("http://localhost:8983/solr/collection2");

    //添加数据
    server.addBean(new Person("1","张三","张三是一个好学生"));
```

```
    //提交请求，如果ID对应的数据存在则进行更新操作，如果不存在则创建索引
    server.commit();
}
```


```
{
  "response": {
    "numFound": 1,
    "start": 0,
    "docs": [
      {
        "id": "1",
        "name": "张三",
        "description": "张三是一个好学生",
        "_version_": 1583610334510317600
      }
    ]
  }
}
```


4.3、使用 SolrJ 删除索引库数据

删除索引可以根据 ID 删除，也可以写一个查询条件，匹配到条件的都会被删除

```
@Test
public void testDleteIndex() throws Exception{
    //创建服务器对象
    HttpSolrServer server = new
HttpSolrServer("http://localhost:8983/solr/collection2");

    //根据条件进行删除
    server.deleteByQuery("name:张三");

    //提交请求
    server.commit();
}
```

4.4、使用 SolrJ 查询索引库数据

4.4.1、以 Document 形式返回查询结果

```
@Test
public void testQueryIndex() throws Exception{
    //创建服务器对象
    HttpSolrServer server = new
HttpSolrServer("http://localhost:8983/solr/collection2");
    //定义查询条件
    SolrQuery query = new SolrQuery("name:李四");
```

```
    //进行查询处理
    QueryResponse queryResponse = server.query(query);
    SolrDocumentList results = queryResponse.getResults();

    System.out.println("找到了"+results.size()+"条记录");

    //便利数据
    for (SolrDocument solrDocument : results) {
        System.out.println(solrDocument.get("id"));
        System.out.println(solrDocument.get("name"));
        System.out.println(solrDocument.get("description"));
    }
}
```

4.4.2、以 JavaBean 形式返回查询结果

```
@Test
public void testQueryIndexBean() throws Exception{
    //创建服务器对象
    HttpSolrServer server = new
HttpSolrServer("http://localhost:8983/solr/collection2");
    //定义查询条件
    SolrQuery query = new SolrQuery("name:李四");
    //进行查询处理
    QueryResponse queryResponse = server.query(query);
    List<Person> beans = queryResponse.getBeans(Person.class);

    for (Person person : beans) {
        System.out.println("id:"+person.getId());
        System.out.println("name:"+person.getName());

    System.out.println("description:"+person.getDescription());
    }
}
```

4.4.3、SolrQuery 对象的高级查询

在创建 SolrQuery 时，我们填写的 Query 语句，可以有以下高级写法：

查询语句中如果有特殊字符，需要转义，可以使用：""

1、匹配所有文档：*:* （通配符？和*："*"表示匹配任意字符；"？"表示匹配出现的位置）

2、布尔操作：AND、OR 和 NOT 布尔操作（推荐使用大写，区分普通字段）

3、子表达式查询（子查询）：可以使用“()”构造子查询。 比如：(query1 AND query2) OR (query3 AND query4)

4、相似度查询：

（1）默认相似度查询：title:appla~ ，此时编辑举例是 2

（2）指定编辑举例的相似度查询：对模糊查询可以设置编辑举例，可选 0~2 的整数。

5、范围查询（Range Query）：Lucene 支持对数字、日期甚至文本的范围查询。结束的范围可以使用“*”通配符。

（1）日期范围（ISO-8601 时间 GMT）：a_begin_date:[1990-01-01T00:00:00.000Z TO 1999-12-31T24:59:99.999Z]

（2）数字：salary:[2000 TO *]

（3）文本：entryNm:[a TO a]

6、日期匹配：YEAR, MONTH, DAY, DATE (synonymous with DAY) HOUR, MINUTE, SECOND, MILLISECOND, and MILLI (synonymous with MILLISECOND)可以被标志成日期。

（1）r_event_date:[* TO NOW-2YEAR]：2 年前的现在这个时间

（2）r_event_date:[* TO NOW/DAY-2YEAR]：2 年前前一天的这个时间

4.4.4、SolrQuery 实现排序

```
5、@Test
6、public void testQueryIndexSort() throws Exception{
7、 //创建服务器对象
8、 HttpSolrServer server = new
   HttpSolrServer("http://localhost:8983/solr/collection2");
9、 //定义查询条件
10、    SolrQuery query = new SolrQuery("*:*");
11、    //设置按ID排序
```

```
12、    query.setSort("id", ORDER.desc);
13、    //进行查询处理
14、    QueryResponse queryResponse = server.query(query);
15、    SolrDocumentList results = queryResponse.getResults();
16、
17、    System.out.println("找到了"+results.size()+"条记录");
18、
19、    //便利数据
20、    for (SolrDocument solrDocument : results) {
21、        System.out.println(solrDocument.get("id"));
22、        System.out.println(solrDocument.get("title"));
23、    }
24、}
```

4.4.5、 SolrQuery 实现分页

```
5.      @Test
6.      public void testQueryIndexPages() throws Exception{
7.          //要查询的页数
8.          int pageNum = 2;
9.          //每页显示条数
10.         int pageSize = 2;
11.         //当前页的起始条数
12.         int start = (pageNum - 1) * pageSize;
13.
14.         //创建服务器对象
15.         HttpSolrServer server = new
        HttpSolrServer("http://localhost:8983/solr/collection2")
        ;
16.         //定义查询条件
17.         SolrQuery query = new SolrQuery("*:*");
18.         //设置按ID排序
19.         query.setSort("id", ORDER.desc);
20.
21.         //设置起始条数
22.         query.setStart(start);
23.         //设置每页条数
24.         query.setRows(pageSize);
25.
26.         //进行查询处理
27.         QueryResponse queryResponse = server.query(query);
28.         SolrDocumentList results = queryResponse.getResults();
29.
30.         System.out.println("当前第" + pageNum + "页，本页共" +
```

```
        results.size() + "条数据。");
31.
32.        //便利数据
33.        for (SolrDocument solrDocument : results) {
34.            System.out.println(solrDocument.get("id"));
35.            System.out.println(solrDocument.get("title"));
36.        }
37.    }
```

4.4.6、 SolrQuery 实现高亮显示

```
5.     @Test
6.     public void testQueryIndexHighlighting() throws Exception{
7.        //创建服务器对象
8.        HttpSolrServer server = new
       HttpSolrServer("http://localhost:8983/solr/collection2")
       ;
9.        //定义查询条件
10.       SolrQuery query = new SolrQuery("title:测试");
11.
12.       //设置高亮的标签
13.       query.setHighlightSimplePre("<em>");
14.       query.setHighlightSimplePost("</em>");
15.       //高亮字段
16.       query.addHighlightField("title");
17.
18.       //进行查询处理
19.       QueryResponse queryResponse = server.query(query);
20.       SolrDocumentList results = queryResponse.getResults();
21.
22.       //获取高亮字段
23.       Map<String, Map<String, List<String>>> highlighting =
       queryResponse.getHighlighting();
24.
25.       System.out.println("找到了"+results.size()+"条记录");
26.
27.       //便利数据
28.       for (SolrDocument solrDocument : results) {
29.           String id = solrDocument.get("id").toString();
30.           System.out.println("id:"+id);
31.
          System.out.println(highlighting.get(id).get("title").
       get(0));
32.       }
```

```
33.         }
```