

基于SpringBoot的Data Solr搜索引擎开发

关于Apache Solr的简介

Solr它是一种开放源码的、基于 Lucene Java 的搜索服务器，易于加入到 Web 应用程序中。Solr 提供了层面搜索(就是统计)、命中醒目显示并且支持多种输出格式（包括XML/XSLT 和JSON等格式）。它易于安装和配置，而且附带了一个基于HTTP 的管理界面。可以使用 Solr 的表现优异的基本搜索功能，也可以对它进行扩展从而满足企业的需要。Solr的特性包括：

- 高级的全文搜索功能
- 专为高通量的网络流量进行的优化
- 基于开放接口（XML和HTTP）的标准
- 综合的HTML管理界面
- 可伸缩性 – 能够有效地复制到另外一个Solr搜索服务器
- 使用XML配置达到灵活性和适配性
- 可扩展的插件体系
- 支持像英语，德语，中国，日本，法国和许多主要语言

Apache Solr和Lucene的关系

Solr 与 Lucene 并不是竞争对立关系，恰恰相反Solr 依存于Lucene，因为Solr 底层的核心技术是使用 Apache Lucene 来实现的，简单的说Solr 是Lucene 的服务器化。需要注意的是Solr 并不是简单的对 Lucene 进行封装，它所提供的大部分功能都区别于Lucene。

Apache Solr 相关目录说明

名称	修改日期	大小	种类
bin	2016年12月30日 下午9:26	--	文件夹
CHANGES.txt	2017年2月6日 下午5:24	567 KB	纯文本文稿
contrib	2017年2月9日 上午2:14	--	文件夹
dist	2017年8月8日 下午4:56	--	文件夹
docs	2017年8月8日 下午4:56	--	文件夹
example	2016年6月24日 下午9:10	--	文件夹
LICENSE.txt	2016年6月24日 下午9:10	13 KB	纯文本文稿
licenses	2017年2月6日 下午5:24	--	文件夹
LUCENE.CHANGES.txt	2017年2月8日 下午10:42	595 KB	纯文本文稿
NOTICE.txt	2016年6月24日 下午9:10	27 KB	纯文本文稿
README.txt	2016年6月24日 下午9:10	7 KB	纯文本文稿
server	2017年8月8日 下午4:59	--	文件夹
contexts	2016年6月24日 下午9:10	--	文件夹
etc	2016年6月24日 下午9:10	--	文件夹
lib	2017年8月8日 下午5:02	--	文件夹
modules	2016年6月24日 下午9:10	--	文件夹
README.txt	2016年6月24日 下午9:10	4 KB	纯文本文稿
resources	2016年12月30日 下午9:26	--	文件夹
jetty-logging.properties	2016年6月24日 下午9:10	68 字节	Java Properties
log4j.properties	2016年12月30日 下午9:26	956 字节	Java Properties
scripts	2016年6月24日 下午9:10	--	文件夹
solr	2017年8月8日 下午5:09	--	文件夹
configsets	2016年6月24日 下午9:10	--	文件夹
README.txt	2016年6月24日 下午9:10	3 KB	纯文本文稿
solr.xml	2016年12月30日 下午9:26	2 KB	XML
zoo.cfg	2016年6月24日 下午9:10	501 字节	Config...tion file
solr-webapp	2017年8月8日 下午5:00	--	文件夹
webapp	2017年2月9日 上午2:13	--	文件夹
start.jar	2015年7月31日 上午1:32	109 KB	Java JAR 文件

Solr程序包的结构说明

- bin：solr相关运行脚本
- docs：相关API参考文档和wiki资料等
- licenses：存放了solr相关证书
- contrib：solr相关扩展的jar包
- dist：存放Solr 构建完成的JAR 文件、WAR 文件和Solr 依赖的JAR 文件。
- example：是一个安装好的Jetty 中间件，其中包括一些样本数据和Solr 的配置信息。

- example/etc：Jetty 的配置文件。
- example/multicore：当安装Solr multicore 时，用来放置多个Solr 主目录。
- example/solr：默认安装时一个Solr 的主目录。
- example/example-DIH：数据库做索引数据源的示例
- example/webapps：Solr 的WAR 文件部署在这里。
- src：Solr 相关源码。
- src/java：Solr 的Java 源码。
- src/scripts：一些在大型产品发布时一些有用的Unix bash shell 脚本。
- src/solrj：Solr 的Java 客户端。
- src/test：Solr 的测试源码和测试文件。
- src/webapp：Solr web 管理界面。管理界面的jsp 文件都放在web/admin/ 下面，可以根据你的需要修改这些文件。

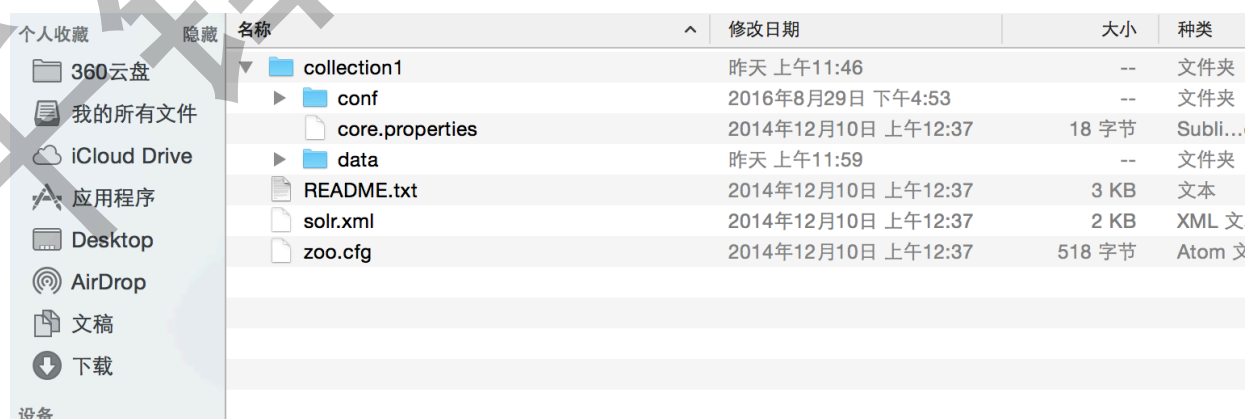
Solr 主目录结构说明

- bin：建议将集群复制脚本放在这个目录下。
- conf：放置配置文件。
- conf/schema.xml：建立索引的schema 包含了字段类型定义和其相关的分析器。
- conf/solrconfig.xml：这个是Solr 主要的配置文件。
- conf/xslt：包含了很多xslt 文件，这些文件能将Solr 的XML 的查询结果转换为特定的格式，比如：Atom/RSS。
- data：放置Lucene 产生的索引数据。
- lib：放置可选的JAR 文件比如对Solr 扩展的插件，这些JAR 文件将会在Solr 启动时加载。

第一章：搭建Apache Solr开发环境

准备步骤：Tomcat 8以上、JDK7以上、solr5.5.4版本

1. 将 solr 压缩包解压，并将solr-5.5.4\solr_webapps下的文件夹webapp，将之复制到 Tomcat\webapps\目录下，并且改名为solr(名字自己定义)
2. 将 solr 压缩包中 solr-5.5.4\server\lib\ext所有jar包 全部复制到 Tomcat\ webapps\solr\WEB-INF\lib目录中
3. 将 solr 压缩包中 solr-5.5.4\ server\resources \log4j.properties 复制到Tomcat\ webapps\solr\WEB-INF 目录中
4. 创建一个solr_home 的目录(用户可以根据操作系统决定)，并将 solr 压缩包中example下的solr 目录除了bin所有文件复制D:\solr_home目录下，如图所示：



名称	修改日期	大小	种类
collection1	昨天 上午11:46	--	文件夹
conf	2016年8月29日 下午4:53	--	文件夹
core.properties	2014年12月10日 上午12:37	18 字节	Subli...
data	昨天 上午11:59	--	文件夹
README.txt	2014年12月10日 上午12:37	3 KB	文本
solr.xml	2014年12月10日 上午12:37	2 KB	XML 文
zoo.cfg	2014年12月10日 上午12:37	518 字节	Atom 文

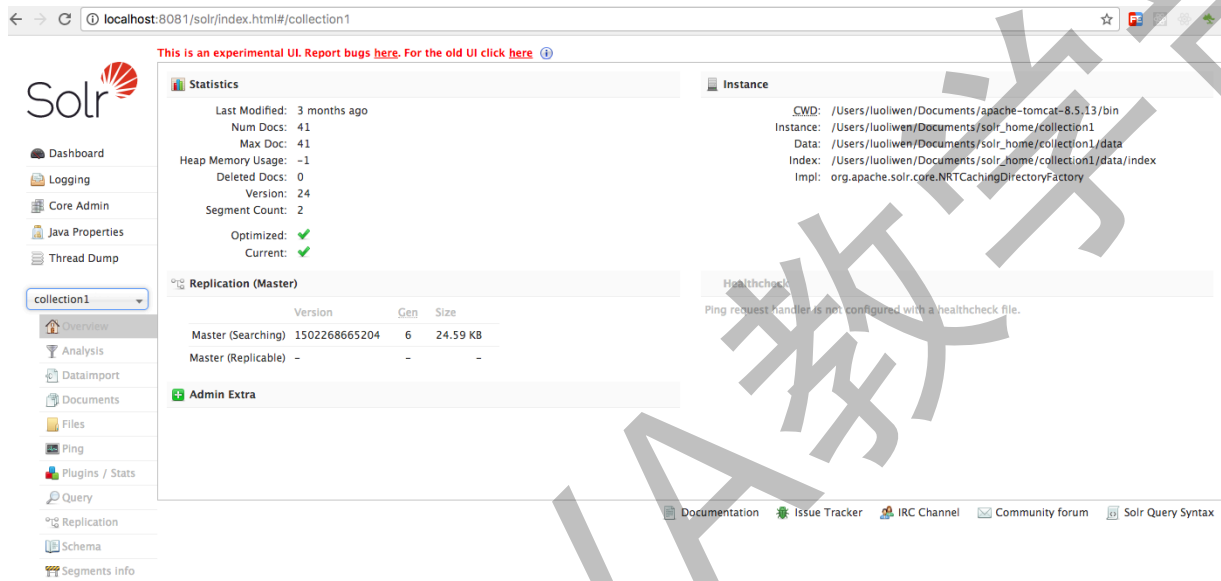
5. 打开Tomcat/webapps/solr/WEB-INF下的web.xml，增加如下配置内容（初始状态下该内容是被注释掉的）：

```
<env-entry>
  <env-entry-name>solr/home</env-entry-name>
  <env-entry-value>D:/solr_home</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
```

将<env-entry-value>中的内容改成你的solrhome路径，这里是D:/solr_home

这项配置,主要是建立tomcat 与solr之间的关系的,它的作用是让tomcat找到你所配置的solr 目录。

6. 启动tomcat，从地址栏输入如下网址：<http://localhost:8080/solr>会出现如下界面：



第二章：Apache Solr 从MySQL数据库中导入数据

1. 导入solr自带的数据导入包放到tomcat 中的solr工程的web-inf下lib中 在collection1的目录中添加lib包，把data-import需要用的包拷贝到lib中 其中包括mysql包
2. 在磁盘中的solr_home 打开solrconfig.xml 添加如下代码

```
<requestHandler name="/dataimport"
  class="org.apache.solr.handler.dataimport.DataImportHandler">
  <lst name="defaults">
    <str name="config">data-config.xml</str>
  </lst>
</requestHandler>
```

3. 在同级目录下创建一个data-config.xml文件，添加如下内容

```
<?xml version="1.0" encoding="UTF-8" ?>
<dataConfig>
<dataSource type="JdbcDataSource" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost/mydb" user="root" password="root" />
  <document name="userinfo">
    <entity name="userinfo" query="select id,name,address,birthday,age from
userinfo ">
      <field column="ID" name="user_id" />
      <field column="NAME" name="user_name" />
      <field column="ADDRESS" name="user_address" />
      <field column="BIRTHDAY" name="user_birthday" />
      <field column="AGE" name="user_age" />
    </entity>
  </document>
</dataConfig>
```

备注说明：

document：表示搜索引擎的文档

entity：表示对应的实体类

query：执行的SQL查询语句

field：搜索引擎对应的域

column：数据库对应的列

name：搜索引擎对应的域名

4. 在同级目录下schema.xml下配置如下代码，将数据库的字段和域——对应关系配置好。

```
<field name="user_id" type="int" indexed="true" stored="true"/>
<field name="user_name" type="string" indexed="true" stored="true"/>
<field name="user_address" type="string" indexed="true" stored="true"/>
<field name="user_birthday" type="date" indexed="true" stored="true"/>
<field name="user_age" type="int" indexed="true" stored="true"/>
```

第三章：Apache Solr配置中文分词器

1、中文分词器采用的是lucene-analyzers-smartcn来做为Apache Solr的分词器，将该jar包拷贝到tomcat下的webapps-solr

工程中的WEB-INF的lib中。

2、在配置的solr_home目录中打开配置好的collection1下的conf的managed-schema文件添加如下代码

启动控制台，从地址栏中输入<http://localhost:8080/solr>会出现如下界面：

[illegible]

第一步：配置SpringBoot的属性文件

```
spring.data.solr.host=http://localhost:8081/solr
spring.thymeleaf.cache=false
spring.thymeleaf.encoding=utf-8
spring.thymeleaf.mode=HTML
spring.thymeleaf.suffix=.html
```

第二步：编写Apache Solr对应的实体类

```
package com.sudojava.springboot_solr.domain;

import org.apache.solr.client.solrj.beans.Field;

public class Product {

    @Field
    private String id;
    @Field
    private String price;
    @Field
    private String title;
    @Field
    private String pic_url;
    @Field
    private String selling_price;
    @Field
    private String sales_volume;
    @Field
    private String coupon_title;

    @Override
    public String toString() {
        return "Product{" +
            "id='" + id + '\'' +
            ", price='" + price + '\'' +
            ", title='" + title + '\'' +
            ", pic_url='" + pic_url + '\'' +
            ", selling_price='" + selling_price + '\'' +
            ", sales_volume='" + sales_volume + '\'' +
            ", coupon_title='" + coupon_title + '\'' +
            '}';
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
```

```
        this.id = id;
    }

    public String getPrice() {
        return price;
    }

    public void setPrice(String price) {
        this.price = price;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getPic_url() {
        return pic_url;
    }

    public void setPic_url(String pic_url) {
        this.pic_url = pic_url;
    }

    public String getSelling_price() {
        return selling_price;
    }

    public void setSelling_price(String selling_price) {
        this.selling_price = selling_price;
    }

    public String getSales_volume() {
        return sales_volume;
    }

    public void setSales_volume(String sales_volume) {
        this.sales_volume = sales_volume;
    }

    public String getCoupon_title() {
        return coupon_title;
    }

    public void setCoupon_title(String coupon_title) {
        this.coupon_title = coupon_title;
    }
}
```

```
}  
}
```

第三步：编写产品查询列表的Service以及实现类

```
package com.sudojava.springboot_solr.service;  
  
import com.sudojava.springboot_solr.domain.Product;  
  
import java.util.List;  
  
public interface ProductService {  
  
    public List<Product> searchProductByName(String title);  
  
}
```

```
package com.sudojava.springboot_solr.service;  
  
import com.sudojava.springboot_solr.common.StringTrim;  
import com.sudojava.springboot_solr.domain.Product;  
import org.apache.solr.client.solrj.SolrClient;  
import org.apache.solr.client.solrj.SolrQuery;  
import org.apache.solr.client.solrj.SolrServerException;  
import org.apache.solr.client.solrj.response.QueryResponse;  
import org.apache.solr.common.SolrDocument;  
import org.apache.solr.common.StringUtils;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Map;  
import java.util.function.Consumer;  
  
@Service  
public class ProductServiceImp implements ProductService {  
  
    @Autowired  
    private SolrClient client;  
  
    @Override  
    public List<Product> searchProductByName(String title) {  
        SolrQuery query = new SolrQuery();  
        query.setQuery("title:" + title);  
        query.set("df", "title");//设置查询的域  
    }  
}
```



```

query.setHighlight(true);
//设置高亮显示的标签
query.setHighlightSimplePre("<font color=\"red\">");
query.setHighlightSimplePost("</font>");
query.addHighlightField("title");
query.setHighlightFragSize(150); //高亮的标题长度
query.setHighlightSnippets(1); //获取高亮片段数，一般搜索词可能分布在文章中的不同位置，其所在一定长度的
//语句即为一个片段，默认为1，但根据业务需要有时候需要多取出几个片段

QueryResponse response;
List<Product> list = new ArrayList<>();
try {
    response = client.query("collection1", query);

    Map<String, Map<String, List<String>>> highlighting =
response.getHighlighting();

    if (response.getStatus() == 0) {
        response.getResults().forEach(new Consumer<SolrDocument>() {
            @Override
            public void accept(SolrDocument entries) {
                Product product = new Product();
                product.setId(StringTrim.trim(entries.get("id")));

                product.setCoupon_title(StringTrim.trim(entries.get("coupon_title")));

                product.setPic_url(StringTrim.trim(entries.get("pic_url")));
                product.setPrice(StringTrim.trim(entries.get("price")));

                product.setSales_volume(StringTrim.trim(entries.get("sales_volume")));

                product.setSelling_price(StringTrim.trim(entries.get("selling_price")));
                Map<String, List<String>> map =
highlighting.get(entries.get("id"));
                List<String> result = map.get("title");
                if (result != null && result.size() > 0) {
                    product.setTitle(StringTrim.trim(result.get(0)));
                } else {
                    product.setTitle(StringTrim.trim(entries.get("title")));
                }

                list.add(product);
            }
        });
    }
} catch (SolrServerException e) {
    e.printStackTrace();
}

```

```

        } catch (IOException e) {
            e.printStackTrace();
        }
        return list;
    }
}

```

第四步：编写产品检索的Controller类以及Index初始化页面、工具类

```

package com.sudojava.springboot_solr.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class IndexController {

    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String index(){
        return "/index";
    }
}

```

```

package com.sudojava.springboot_solr.controller;

import com.sudojava.springboot_solr.domain.Product;
import com.sudojava.springboot_solr.service.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import java.util.List;

@Controller
public class ProductSearchController {

    @Autowired
    private ProductService service;

    @RequestMapping(value = "/search", method = RequestMethod.POST)
    public String searchProduct(String title, Model model) {
        System.out.println("---title-->" + title);
        List<Product> list = service.searchProductByName(title);
        System.out.println("----->" + list);
        model.addAttribute("list", list);
    }
}

```

```
        return "/index";
    }

}
```

```
package com.sudojava.springboot_solr.common;

public class StringTrim {

    public static String trim(Object str) {
        if (null == str) {
            str = "";
        }
        return str.toString();
    }
}
```

第五步：编写查询的界面，采用Thymeleaf模板引擎显示数据

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Solr查询产品界面</title>

    <meta charset="utf-8"/>
    <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
    <meta name="viewport" content="width=device-width,initial-scale=1,user-
scalable=no"/>
    <!--引入CSS样式-->
    <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css"/>
    <style>
        .mainCSS {
            margin: 0 auto;
            width: 1050px;
        }
        * {
            padding: 0;
            margin: 5px;
            list-style: none;
        }

        html, body {
            width: 100%;
        }

        #wrap {
            width: 1050px;
```

[illegible]

```

        <p>
            <strong>介绍: </strong><span th:utext="${pro.title}"></span>
        </p>
        <p style="text-decoration: line-through;color: red;">
            <strong>原价: </strong><span th:text="${pro.price}"></span>
        </p>
        <p><strong>售价: </strong><span th:text="${pro.selling_price}"></span>
        </p>
        <p><strong>销量: </strong><span th:text="${pro.sales_volume}"></span>
        </p>
    </li>

</ul>
</div>
</body>
</html>

```

第五章：使用SolrClient对Solr的索引库进行添加、删除、和修改操作

本章节主要是讲解使用MockMVC集成单元测试对Solr的索引库进行单元测试。

测试步骤如下：

第一步：定义对Solr的操作接口以及实现类

```

package com.sudojava.springboot_solr.service;

import com.sudojava.springboot_solr.domain.Product;

import java.util.List;

public interface ProductService {

    public List<Product> searchProductByName(String title);

    public int addProduct(Product product);

    public int deleteProduct(String id);

}

```

```

package com.sudojava.springboot_solr.service;

import com.sudojava.springboot_solr.common.StringTrim;
import com.sudojava.springboot_solr.domain.Product;
import org.apache.solr.client.solrj.SolrClient;
import org.apache.solr.client.solrj.SolrQuery;

```

```

import org.apache.solr.client.solrj.SolrServerException;
import org.apache.solr.client.solrj.response.QueryResponse;
import org.apache.solr.client.solrj.response.UpdateResponse;
import org.apache.solr.common.SolrDocument;
import org.apache.solr.common.SolrInputDocument;
import org.apache.solr.common.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.function.Consumer;

@Service
public class ProductServiceImp implements ProductService {

    @Autowired
    private SolrClient client;

    @Override
    public int addProduct(Product product) {
        //第一种添加索引方式
        SolrInputDocument document = new SolrInputDocument();
        document.addField("id", product.getId());
        document.addField("coupon_title", product.getCoupon_title());
        document.addField("pic_url", product.getPic_url());
        document.addField("price", product.getPrice());
        document.addField("sales_volume", product.getSales_volume());
        document.addField("selling_price", product.getSelling_price());
        UpdateResponse response = null;
        try {
            response = client.add("collection1", document);
            if (response.getStatus() == 0) {
                client.commit("collection1");
            }
        } catch (SolrServerException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return response.getStatus();
    }

    @Override
    public int deleteProduct(String id) {

```

```

UpdateResponse response = null;
try {
    response = client.deleteById("collection1",id);
    if (response.getStatus()==0) {
        client.commit("collection1");
    }
} catch (SolrServerException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
return response.getStatus();
}

@Override
public List<Product> searchProductByName(String title) {
    SolrQuery query = new SolrQuery();
    query.setQuery("title:" + title);
    query.set("df", "title");//设置查询的域
    query.setHighlight(true);

    //设置高亮显示的标签
    query.setHighlightSimplePre("<font color=\"red\">");
    query.setHighlightSimplePost("</font>");
    query.addHighlightField("title");
    query.setHighlightFragSize(150);//高亮的标题长度
    query.setHighlightSnippets(1);//获取高亮分片数，一般搜索词可能分布在文章中的不同位置，其所在一定长度的
    //语句即为一个片段，默认为1，但根据业务需要有时候需要多取出几个分片

    QueryResponse response;
    List<Product> list = new ArrayList<>();
    try {
        response = client.query("collection1", query);

        Map<String, Map<String, List<String>>> highlighting =
response.getHighlighting();

        if (response.getStatus() == 0) {
            response.getResults().forEach(new Consumer<SolrDocument>() {
                @Override
                public void accept(SolrDocument entries) {
                    Product product = new Product();
                    product.setId(StringTrim.trim(entries.get("id")));

                    product.setCoupon_title(StringTrim.trim(entries.get("coupon_title")));

                    product.setPic_url(StringTrim.trim(entries.get("pic_url")));
                    product.setPrice(StringTrim.trim(entries.get("price")));
                }
            });
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return list;
}

```

```

product.setSales_volume(StringTrim.trim(entries.get("sales_volume")));

product.setSelling_price(StringTrim.trim(entries.get("selling_price")));
        Map<String, List<String>> map =
highlighting.get(entries.get("id"));
        List<String> result = map.get("title");
        if (result != null && result.size() > 0) {
            product.setTitle(StringTrim.trim(result.get(0)));
        } else {

product.setTitle(StringTrim.trim(entries.get("title")));
        }

        list.add(product);
    }
    });
}
} catch (SolrServerException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
return list;
}
}

```

Solr同时也支持对索引的实体类进行插入操作

```

@Override
    public int addProduct(Product product) {
        //第一种添加索引方式
        // SolrInputDocument document = new SolrInputDocument();
        // document.addField("id", product.getId());
        // document.addField("coupon_title", product.getCoupon_title());
        // document.addField("pic_url", product.getPic_url());
        // document.addField("price", product.getPrice());
        // document.addField("sales_volume", product.getSales_volume());
        // document.addField("selling_price", product.getSelling_price());
        UpdateResponse response = null;
        try {
            response = client.addBean("collection1", product);
            if (response.getStatus() == 0) {
                client.commit("collection1");
            }
        } catch (SolrServerException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

```



```
    }  
    return response.getStatus();  
}
```

第三步：创建添加、删除的Controller控制器

```
package com.sudojava.springboot_solr.controller;  
  
import com.sudojava.springboot_solr.domain.Product;  
import com.sudojava.springboot_solr.service.ProductService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestMethod;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.servlet.ModelAndView;  
  
import java.util.List;  
  
@Controller  
public class ProductSearchController {  
  
    @Autowired  
    private ProductService service;  
  
    @RequestMapping(value = "/search", method = RequestMethod.POST)  
    public ModelAndView searchProduct(String title) {  
        System.out.println("---title-->" + title);  
        ModelAndView modelAndView = new ModelAndView();  
        List<Product> list = service.searchProductByName(title);  
        modelAndView.addObject("list", list);  
        modelAndView.setViewName("/index");  
        return modelAndView;  
    }  
  
    @RequestMapping(value = "/add", method = RequestMethod.POST)  
    public String addProduct(Product product, Model model) {  
        int flag = service.addProduct(product);  
        System.out.println("----->" + flag);  
        model.addAttribute("flag", "addProduct is success!!!" + flag);  
        return "/index";  
    }  
  
    @RequestMapping(value = "/delete", method = RequestMethod.POST)  
    public String deleteProduct(String id, Model model) {  
        int flag = service.deleteProduct(id);  
        System.out.println("----->" + flag);  
        model.addAttribute("flag", "deleteProduct is success!!!" + flag);  
    }  
}
```

```

        return "/index";
    }

}

```

第四步：创建单元测试类进行测试操作

```

package com.sudojava.springboot_solr;

import com.sudojava.springboot_solr.common.StringTrim;
import com.sudojava.springboot_solr.domain.Product;
import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.context.annotation.Bean;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.test.context.web.WebAppConfiguration;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
import org.springframework.test.web.servlet.result.MockMvcResultHandlers;
import org.springframework.test.web.servlet.setup.MockMvcBuilders;
import org.springframework.util.LinkedMultiValueMap;
import org.springframework.util.MultiValueMap;
import org.springframework.web.context.WebApplicationContext;

import java.io.IOException;

@RunWith(SpringJUnit4ClassRunner.class)
@WebAppConfiguration
@SpringBootTest
public class SpringbootSolrApplicationTests {

    private MockMvc mockMvc;

    @Autowired
    private WebApplicationContext context;

    @Before
    public void setup() {
        mockMvc = MockMvcBuilders.webAppContextSetup(context).build();
    }

    @Test
    public void searchProduct() throws Exception{

```

```

        mockMvc.perform(MockMvcRequestBuilders.post("/search")
            .param("title", "女装")
        )

        .andDo(MockMvcResultHandlers.print());
    }

    @Test
    public void deleteProduct() throws Exception{
        mockMvc.perform(MockMvcRequestBuilders.post("/delete")
            .param("id", "10010234")
        )

        .andDo(MockMvcResultHandlers.print());
    }

    @Test
    public void add() throws Exception {

        Product product = new Product();
        product.setId("10010234");
        product.setTitle("衣服打折销售了");
        product.setSelling_price("1000");
        product.setSales_volume("200");
        product.setPrice("100");
        product.setCoupon_title("很好了www");

        product.setPic_url("http://img1.tbcdn.cn/tfscom/i1/TB1qBvJOXXXXXcwapXXXXXXXXXX_!!
0-item_pic.jpg_450x10000.jpg");

        //      LinkedMultiValueMap<String, String> map = new LinkedMultiValueMap<>();
        //      map.add("id", product.getId());
        //      map.add("coupon_title", product.getCoupon_title());
        //      map.add("pic_url", product.getPic_url());
        //      map.add("price", product.getPrice());
        //      map.add("sales_volume", product.getSales_volume());
        //      map.add("selling_price", product.getSelling_price());

        mockMvc.perform(MockMvcRequestBuilders.post("/add").
            param("id", product.getId()).
            param("coupon_title", product.getCoupon_title()).
            param("pic_url", product.getPic_url()).
            param("price", product.getPrice()).
            param("sales_volume", product.getSales_volume()).
            param("selling_price", product.getSelling_price())
        )

        .andDo(MockMvcResultHandlers.print());
    }
}

```

第六章：基于SpringBoot JPA 的Solr 数据查询

本章节主要是介绍SpringBoot 关于Solr查询的JPA规范。Spring家族提供的spring-data适用于关系型数据库和nosql数据库

例如 Spring Data JPA, Spring Data Hadoop, Spring Data MongoDB , Spring Data Solr 等。

他们的共同特点是给我们提供了框架代码，spring Data能自动创建实体dao的实现类和自定义查询，不用我们去定义了，这样简化了对dao层的操作。

值得注意的是如果使用Spring Data Solr操作的话，必须要在Solr对应的实体类中添加一个标签

```
import org.apache.solr.client.solrj.beans.Field;
import org.springframework.data.solr.core.mapping.SolrDocument;

@SolrDocument(solrCoreName = "collection1")
public class Product {
    .....省略代码....
}
```

第一步：声明一个接口继承SolrCrudRepository接口

```
package com.sudojava.springboot_solr.repository;

import com.sudojava.springboot_solr.domain.Product;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.solr.repository.SolrCrudRepository;

public interface ProductRepository extends SolrCrudRepository<Product,String> {
    Page<Product> findByTitle(String title, Pageable pageable);
}
```

第二步：分别声明对应的JPA Service、ServiceImpl、以及Controller类

```
package com.sudojava.springboot_solr.jpa;

import com.sudojava.springboot_solr.domain.Product;
import org.springframework.data.domain.Page;

public interface ProductJPAService {

    Page<Product> findByTitle(String title);

    public void addProduct(Product product);
}
```

```
        public void deleteProductByID(String id);
    }
}
```

```
package com.sudojava.springboot_solr.jpa;

import com.sudojava.springboot_solr.domain.Product;
import com.sudojava.springboot_solr.repository.ProductRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.stereotype.Service;

@Service
public class ProductJPAServiceImp implements ProductJPAService {

    @Autowired
    private ProductRepository repository;

    @Override
    public Page<Product> findByTitle(String title) {

        return repository.findByTitle(title, PageRequest.of(1,6));
    }

    @Override
    public void addProduct(Product product) {
        repository.save(product);
    }

    @Override
    public void deleteProductByID(String id) {
        repository.deleteById(id);
    }
}
```

```
package com.sudojava.springboot_solr.jpa;

import com.sudojava.springboot_solr.domain.Product;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class ProductJPASearchController {
```

```

@Autowired
private ProductJPAService service;

//做分页处理
@RequestMapping(value = "/jpa/search", method = RequestMethod.POST)
public String search(String title, Model model) {
    Page<Product> page = service.findByTitle(title);
    model.addAttribute("products", page.getContent());
    model.addAttribute("page", page);
    return "/index";
}

@RequestMapping(value = "/jpa/add", method = RequestMethod.POST)
public String add(Product product) {
    service.addProduct(product);
    return "/index";
}

@RequestMapping(value = "/jpa/delete", method = RequestMethod.POST)
public String delete(String id) {
    service.deleteProductByID(id);
    return "/index";
}
}

```

第四步：声明单元测试类

```

package com.sudojava.springboot_solr;

import com.sudojava.springboot_solr.common.StringTrim;
import com.sudojava.springboot_solr.domain.Product;
import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.context.annotation.Bean;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.test.context.junit4.SpringRunner;
import org.springframework.test.context.web.WebAppConfiguration;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
import org.springframework.test.web.servlet.result.MockMvcResultHandlers;
import org.springframework.test.web.servlet.setup.MockMvcBuilders;
import org.springframework.util.LinkedMultiValueMap;

```

```
import org.springframework.util.MultiValueMap;
import org.springframework.web.context.WebApplicationContext;

import java.io.IOException;

@RunWith(SpringJUnit4ClassRunner.class)
@WebAppConfiguration
@SpringBootTest
public class SpringbootSolrApplicationTests {

    private MockMvc mockMvc;

    @Autowired
    private WebApplicationContext context;

    @Before
    public void setup() {
        mockMvc = MockMvcBuilders.webAppContextSetup(context).build();
    }

    @Test
    public void searchProduct() throws Exception{
        mockMvc.perform(MockMvcRequestBuilders.post("/jpa/search")
            .param("title", "女装")
        )
            .andDo(MockMvcResultHandlers.print());
    }

    @Test
    public void deleteProduct() throws Exception{
        mockMvc.perform(MockMvcRequestBuilders.post("/delete")
            .param("id", "10010234")
        )
            .andDo(MockMvcResultHandlers.print());
    }

    @Test
    public void add() throws Exception {

        Product product = new Product();
        product.setId("10010234");
        product.setTitle("衣服打折销售了");
        product.setSelling_price("1000");
        product.setSales_volume("200");
        product.setPrice("100");
        product.setCoupon_title("很好了www");
    }
}
```

```
product.setPic_url("http://img1.tbcdn.cn/tfscom/i1/TB1qBvJOXXXXXcwapXXXXXXXXXX_!!
0-item_pic.jpg_450x10000.jpg");

//      LinkedMultiValueMap<String, String> map = new LinkedMultiValueMap<>();
//      map.add("id", product.getId());
//      map.add("coupon_title", product.getCoupon_title());
//      map.add("pic_url", product.getPic_url());
//      map.add("price", product.getPrice());
//      map.add("sales_volume", product.getSales_volume());
//      map.add("selling_price", product.getSelling_price());

mockMvc.perform(MockMvcRequestBuilders.post("/jpa/add").
    param("id", product.getId()).
    param("coupon_title", product.getCoupon_title()).
    param("pic_url", product.getPic_url()).
    param("price", product.getPrice()).
    param("sales_volume", product.getSales_volume()).
    param("selling_price", product.getSelling_price())
)
    .andDo(MockMvcResultHandlers.print());
}
}
```