[今日课程大纲]

商品搜索功能业务分析

Solr 数据初始化

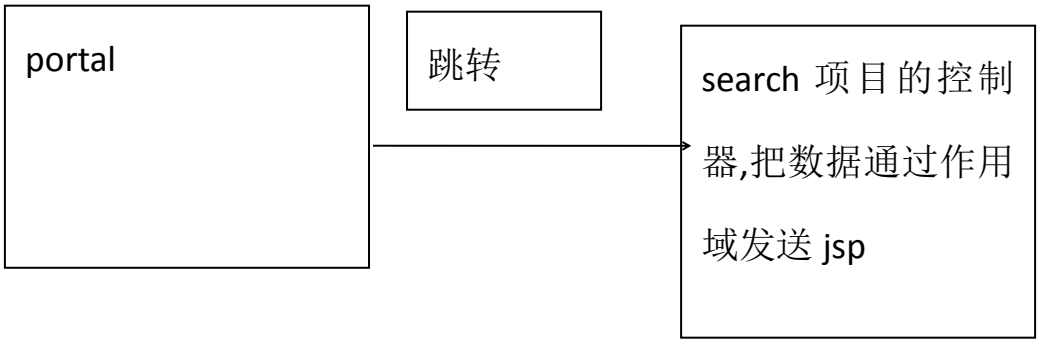
编写 Ego-Search 项目

HttpClient

[知识点详解]

一.商品搜索功能业务分析

1.搜索功能写在 ego-search 中



2. 在编写 ego-search 项目的控制器时发送给 jsp 多个值

2.1 ${query} 搜索内容

2.2 ${itemList} 搜索到商品列表

2.2.1 商品的图片是数组类型

2.3 ${totalPages} 总页数

2.4 ${page} 当前页

3. Solr 完成步骤:

3.1 先配置 field

3.1.1 注意:field 的名称不要和原有名称重复.

3.2 数据初始化.

3.3 完成搜索功能

二. 配置 Solr 的 Field

1. 已经在 Solr 中安装好 IK Analyzer

2. 在 Solrhome/collection1/conf/schema.xml 中配置配置业务字段

```
<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"/>
```

三. 创建 ego-search 项目

1. 新建 ego-search
2. 在 ego-parent 中添加

```
<dependency>
        <groupId>org.apache.solr</groupId>
        <artifactId>solr-solrj</artifactId>
        <version>5.3.1</version>
    </dependency>
```

3. 把 ego-manage 的 pom.xml 粘贴到 ego-search

    3.1 修改端口为 8083

    3.2 添加 solrj 的依赖

    3.3 删除 ego-redis 的依赖

        3.3.1 如果不删除,且 redis 服务器没有启动,会在启动 tomcat 时,

停在 init Webapplicationcontext

4. 把 ego-manage 的 webapps 下的 WEB-INF 和 META-INF 粘贴进来

4.1 删除原有 jsp 等文件夹,粘贴 search 页面文件夹

5. 把 ego-manage 的 4 个配置文件粘贴

5.1 applicationContext-dubbo.xml 修改应用程序名和注解扫描包名

四.Solr 数据初始化

1. 由于 ego-manage 没有提供 solr 初始化按钮等功能.

1.1 只能在 ego-search 编写控制器,通过浏览器访问实现数据初始化.

2. 在 dubbo 中 consumer 调用 privider 默认最大数据传输量为 8M

2.1 <dubbo:protocal/>中 payload=”8388608” 取值为 int,表示每次 consumer 向 provider 请求数据时数据响应最大量.单位字节

3. 实现步骤:

3.1 在 ego-service 中 TbItemDubboService 及实现类添加

```
/**
    * 通过状态查询全部可用数据
    * @return
    */
   List<TbItem> selAllByStatus(byte status);
```

```
@Override
```

```
public List<TbItem> selAllByStatus(byte status) {
    TbItemExample example = new TbItemExample();

example.createCriteria().andStatusEqualTo(status);
    return tbItemMapper.selectByExample(example);
}
```

3.2 在 ego-service 中 TbItemDescDubboService 及实现类添加

```
/**
 * 根据主键查询商品描述对象
 * @param itemid
 * @return
 */
TbItemDesc selByItemid(long itemid);
```

```
@Override
public TbItemDesc selByItemid(long itemid) {
    return
tbItemDescMapper.selectByPrimaryKey(itemid);
}
```

3.3 在 ego-search 中新建 applicationContext-solr.xml

```
<bean id="solrClient"
class="org.apache.solr.client.solrj.impl.CloudSolrCli
ent">
```

```
    <constructor-arg type="java.lang.String"
value="192.168.249.131:2181,192.168.249.131:2182,192.
168.249.131:2183"></constructor-arg>
    <property name="defaultCollection"
value="collection1"></property>
  </bean>
```

3.4 在 ego-search 中新建 com.ego.search.service.TbItemService 及实现类

```
public interface TbItemService {
   void init() throws SolrServerException, IOException;
}
```

```
@Service
public class TbItemServiceImpl implements TbItemService{
   @Reference
   private TbItemDubboService tbItemDubboServiceImpl;
   @Reference
   private TbItemCatDubboService
tbItemCatDubboServiceImpl;
   @Reference
   private TbItemDescDubboService
tbItemDescDubboServiceImpl;
```

```
    @Resource
    private CloudSolrClient solrClient;
    @Override
    public void init() throws SolrServerException,
IOException {
        //查询所有正常的商品
        List<TbItem> listItem =
tbItemDubboServiceImpl.selAllByStatus((byte)1);
        for (TbItem item : listItem) {
            //商品对应的类目信息
            TbItemCat cat =
tbItemCatDubboServiceImpl.selById(item.getCid());
            //商品对应的描述信息
            TbItemDesc desc =
tbItemDescDubboServiceImpl.selByItemid(item.getId());
            SolrInputDocument doc = new
SolrInputDocument();

            doc.addField("id", item.getId());
            doc.addField("item_title", item.getTitle());
            doc.addField("item_sell_point",
item.getSellPoint());
```

```
            doc.addField("item_price",item.getPrice() );
            doc.addField("item_image", item.getImage());
            doc.addField("item_category_name",
cat.getName());
            doc.addField("item_desc", desc.getItemDesc());

            solrClient.add(doc);
        }
        solrClient.commit();
    }
}
```

3.5 在 ego-search 中 新 建 控 制 器 com.ego.search.controller.TbItemController

```
@Controller
public class TbItemController {
    @Resource
    private TbItemService tbItemServiceImpl;
    /**
     * 初始化
     * @return
     */
    @RequestMapping(value="solr/init",produces="text/ht
```

```
ml;charset=utf-8")
   @ResponseBody
   public String init(){
      long start = System.currentTimeMillis();
      try {
         tbItemServiceImpl.init();
         long end = System.currentTimeMillis();
         return "初始化总时间:"+(end-start)/1000+"秒";
      } catch (Exception e) {
         e.printStackTrace();
         return "初始化失败";
      }
   }
}
```

五. 门户搜索功能

1. 在 ego-search 新建实体类 com.ego.search.pojo.TbItemChild

```
public class TbItemChild extends TbItem {
   private String [] images;
```

2. 在 ego-search 的 TbItemService 及实现类添加

```
/**
    * 分页查询
    * @param query
    * @return
    */
   Map<String,Object>    selByQuery(String    query,int page,int rows) throws SolrServerException, IOException;
```

```
@Override
   public Map<String, Object> selByQuery(String query,int page,int rows) throws SolrServerException, IOException {
      SolrQuery params = new SolrQuery();
      //设置分页条件
      params.setStart(rows*(page-1));
      params.setRows(rows);
      //设置条件
      params.setQuery("item_keywords:"+query);
      //设置高亮
      params.setHighlight(true);
      params.addHighlightField("item_title");
      params.setHighlightSimplePre("<span style='color:red'>");
```

```
        params.setHighlightSimplePost("</span>");

        QueryResponse response = solrClient.query(params);

        List<TbItemChild> listChild = new ArrayList<>();
        //未高亮内容
        SolrDocumentList listSolr = response.getResults();
        //高亮内容
        Map<String, Map<String, List<String>>> hh =
response.getHighlighting();

        for (SolrDocument doc : listSolr) {
           TbItemChild  child = new TbItemChild();


     child.setId(Long.parseLong(doc.getFieldValue("id").
toString()));
           List<String> list =
hh.get(doc.getFieldValue("id")).get("item_title");
           if(list!=null&&list.size()>0){
              child.setTitle(list.get(0));
           }else{
```

```
   child.setTitle(doc.getFieldValue("item_title").toSt
ring());
         }

   child.setPrice((Long)doc.getFieldValue("item_price"
));
         Object image = doc.getFieldValue("item_image");

   child.setImages(image==null||image.equals("")?new
String[1]:image.toString().split(","));

   child.setSellPoint(doc.getFieldValue("item_sell_poi
nt").toString());

         listChild.add(child);
      }


      Map<String,Object> resultMap = new HashMap<>();
      resultMap.put("itemList", listChild);
      resultMap.put("totalPages",
```

```
listSolr.getNumFound()%rows==0?listSolr.getNumFound()
/rows:listSolr.getNumFound()/rows+1);



      return resultMap;
   }
```

3. 在 ego-search 中 TbItemController 编写控制器

```
/**
    * 搜索功能
    * @param model
    * @param q
    * @param page
    * @param rows
    * @return
    */
   @RequestMapping("search.html")
   public String search(Model model,String
q,@RequestParam(defaultValue="1") int
page,@RequestParam(defaultValue="12") int rows){
      try {
         q = new
String(q.getBytes("iso-8859-1"),"utf-8");
```

```
        Map<String, Object> map =
tbItemServiceImpl.selByQuery(q, page, rows);
        model.addAttribute("query", q);
        model.addAttribute("itemList",
map.get("itemList"));
        model.addAttribute("totalPages",
map.get("totalPages"));
        model.addAttribute("page", page);
      } catch (Exception e) {
        e.printStackTrace();
      }
      return "search";
   }
```

六. ego-search 新增功能

1. 在 ego-search 中 TbItemService 添加

```
/**
    * 新增
    * @param item
    * @return
    */
   int add(Map<String,Object> map,String desc) throws
```

```
SolrServerException, IOException;
```

```
@Override
   public int add(Map<String,Object> map,String desc)
throws SolrServerException, IOException {
      SolrInputDocument doc = new SolrInputDocument();
      doc.setField("id", map.get("id"));
      doc.setField("item_title", map.get("title"));
      doc.setField("item_sell_point",
map.get("sellPoint"));
      doc.setField("item_price", map.get("price"));
      doc.setField("item_image", map.get("image"));
      doc.setField("item_category_name",
tbItemCatDubboServiceImpl.selById((Integer)map.get("c
id")).getName());
      doc.setField("item_desc", desc);
      UpdateResponse response = solrClient.add(doc);
      solrClient.commit();
      if(response.getStatus()==0){
         return 1;
      }
      return 0;
   }
```

2. 在 ego-search 的 TbItemController 添加

2.1 @RequestBody 把请求体中流数据转换为指定类型

```
/**
     * 新增
     * @param tbItem
     * @return
     */
    @RequestMapping("solr/add")
    @ResponseBody
    public int add(@RequestBody Map<String,Object> map){
        System.out.println(map);
        System.out.println(map.get("item"));
        try {
            return
tbItemServiceImpl.add((LinkedHashMap)map.get("item"),
map.get("desc").toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
        return 0;
    }
```

七. HttpClient 简介

1. 由 apache 推出的实现使用 java 代码完成请求/响应的一套 API
2. 实现效果:

   2.1 模拟浏览器发送请求及解析响应内容
3. 常用类:

   3.1 CloseableHttpClient :负责发送请求和接收响应.相当于浏览器

   3.2 HttpPost: 请求对象,所有请求信息都放入到这个对象中

      3.2.1 HttpGet: get 方式的请求

   3.3 CloseableHttpResponse: 响应对象,所有响应信息放入到这个类

   3.4 EntityUtils: 工具类,解析响应体

八. 实现后台商品新增时同步 Solr 数据

1. 在 ego-parent 中引入 httpclient 的依赖

```
<httpclient-version>4.4.1</httpclient-version>
<!-- httpclient -->
        <dependency>

   <groupId>org.apache.httpcomponents</groupId>
```

```
            <artifactId>httpclient</artifactId>
            <version>${httpclient-version}</version>
        </dependency>
```

2. 在 ego-commons 中添加 httpclient 依赖和 HttpClientUtil 工具类和在 commons.properties 添加 solr 的 url

```
search.url = http://localhost:8083/solr/add
```

3. 在 ego-manage 中 TbItemServiceImpl 的 save 方法最下面添加

3.1 doPostJson()设置请求头中内容类型为 json 把参数以流的形式发送给服务器

```
final TbItem itemFinal = item;
      final String descFinal = desc;
      new Thread(){
         public void run() {
            Map<String,Object> map = new HashMap<>();
            map.put("item", itemFinal);
            map.put("desc", descFinal);

            HttpClientUtil.doPostJson(url,
JsonUtils.objectToJson(map));
            //使用 java 代码调用其他项目的控制器
         };
```

```
    }.start();
```