# 一、解决Cookie跨域的问题

参考文档

# 二、注销功能的实现

保证客户端携带的cookie和注销的cookie的路径是相同

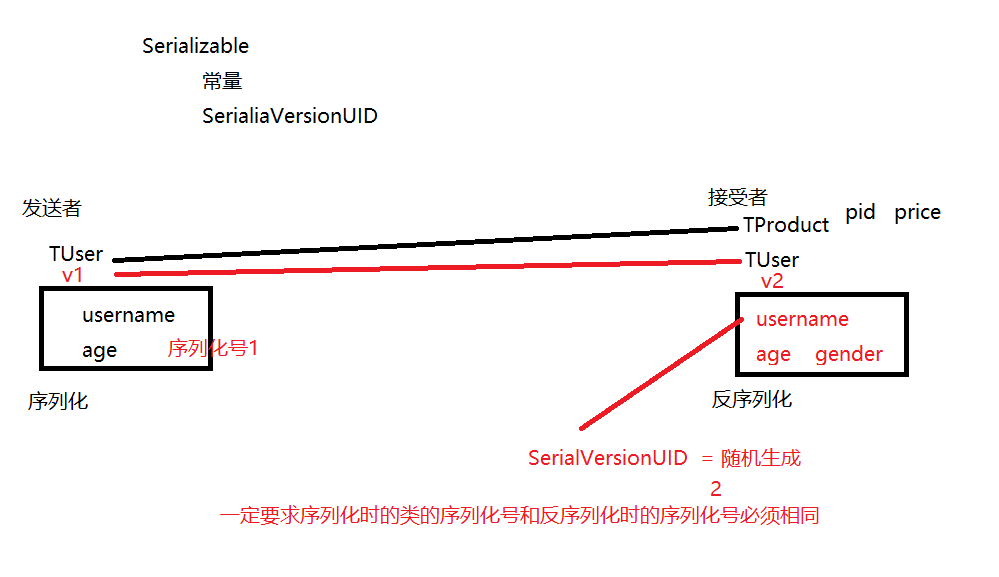
# 三、分布式项目的二级域名问题

hosts文件的路径：

C:\Windows\System32\drivers\etc



# 四、关于序列化版本号的问题



# 五、解决使用二级域名时cookie携带的问题

*//设置cookie在qf.com的域名下包含二级域名..等的情况下都会携带  
//但是： 对于www.baidu.com localhost等这些域名就不会携带。*cookie.setDomain(**"qf.com"**);  
*//该cookie只能被后端访问，提高cookie的安全性*cookie.setHttpOnly(**true**);

注意：前端ajax请求的路径就不能使用localhost，而是用sso.qf.com这样的域名。

# 六、solr中如何自定义字段：

1.sql

|  |
| --- |
| SELECT  a.`id` AS t\_id,  a.name AS t\_name,  a.price AS t\_price,  a.`sale\_point` AS t\_sale\_point,  b.`product\_desc` AS t\_product\_desc,  a.`image` AS t\_image  FROM  t\_product AS a  LEFT JOIN t\_product\_desc AS b  ON a.id = b.pid |

2.加入自定义字段

需要在solr的分词器ik域中设置这些自定义字段，只有这些自定义字段，才能使用这些字段来存放数据

修改 managed-schema文件里面加入字段

|  |
| --- |
| <field name="t\_id" type="plong" indexed="true" stored="true"/>  <field name="t\_name" type="text\_ik" indexed="true" stored="true"/>  <field name="t\_price" type="pfloat" indexed="true" stored="true"/>  <field name="t\_sale\_point" type="text\_ik" indexed="true" stored="true" />  <field name="t\_image" type="string" indexed="true" stored="true" /> |

3.加入复制域

为了方便一个关键字，在多个字段中查询

4.实现删除

通过id来删除

<delete>

<id>1002</id>

</delete>

<commit/>

通过查询结果来删除

<delete>

<query>\*:\*</query>

</delete>

<commit/>

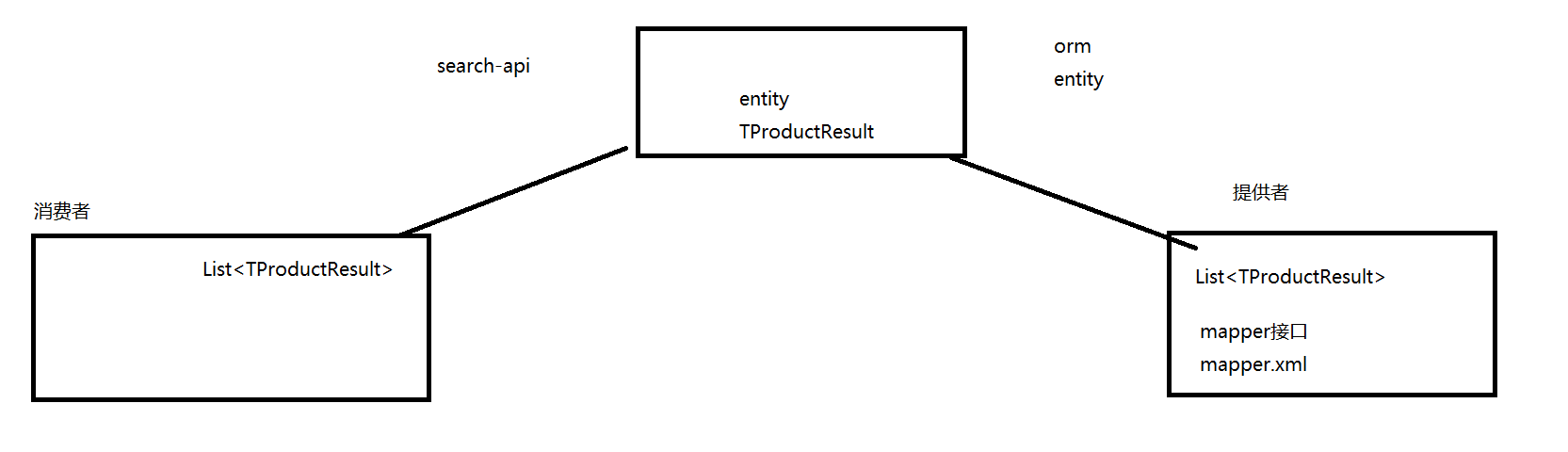
# 七、在项目中使用solr来提供搜索服务

## 1.准备工作

1）创建dubbo结构（服务提供者、服务消费者、api）

2）创建实体类

|  |
| --- |
| SELECT  a.`id`,  a.name AS t\_name,  a.price AS t\_price,  a.`sale\_point` AS t\_sale\_point,  b.`product\_desc` AS t\_product\_desc,  a.`image` AS t\_image  FROM  t\_product AS a  LEFT JOIN t\_product\_desc AS b  ON a.id = b.pid |



3）创建映射关系

4）实现实体数据的封装和获取

mapper.xml

|  |
| --- |
| <**mapper namespace="com.qf.v4.search.service.mapper.TProductResultMapper"**>   <**resultMap id="BaseResultMap" type="com.qf.v4.search.api.entity.TProductResult"**>  <**id column="id" jdbcType="BIGINT" property="id"** />  <**result column="t\_name" jdbcType="VARCHAR" property="tName"** />  <**result column="t\_price" jdbcType="DECIMAL" property="tPrice"** />  <**result column="t\_image" jdbcType="VARCHAR" property="tImage"** />  <**result column="t\_sale\_point" jdbcType="VARCHAR" property="salePoint"** />  <**result column="t\_product\_desc" jdbcType="VARCHAR" property="productDesc"** />  </**resultMap**>    <**select id="selectList" resultMap="BaseResultMap"**>  SELECT  a.`id`,  a.name AS t\_name,  a.price AS t\_price,  a.`sale\_point` AS t\_sale\_point,  b.`product\_desc` AS t\_product\_desc,  a.`image` AS t\_image  FROM  t\_product AS a  LEFT JOIN t\_product\_desc AS b  ON a.id = b.pid  </**select**>   </**mapper**> |

2.如何实现数据的插入到solr数据库中

|  |
| --- |
| *//添加一个数据到solr* @Test **public void** testInsertSingleData(){   *//得到数据* TProductResult product = **new** TProductResult();  product.setId(1001L);  product.settName(**"娃娃"**);  product.settPrice(**new** BigDecimal(20099));  product.setSalePoint(**"好玩！"**);  product.settImage(**"pic/1.jpg"**);    *//把数据封装成document对象* SolrInputDocument document = **new** SolrInputDocument();  document.setField(**"id"**,product.getId());  document.addField(**"t\_name"**,product.gettName());  document.setField(**"t\_price"**,product.gettPrice().intValue());  document.setField(**"t\_sale\_point"**,product.getSalePoint());  document.setField(**"t\_image"**,product.gettImage());   **try** {  *//需要通过solr客户端将这一条数据插入到solr中* **solrClient**.add(document);  *//需要提交* **solrClient**.commit();  } **catch** (SolrServerException e) {  e.printStackTrace();  } **catch** (IOException e) {  e.printStackTrace();  }   } |

## 2.删除数据

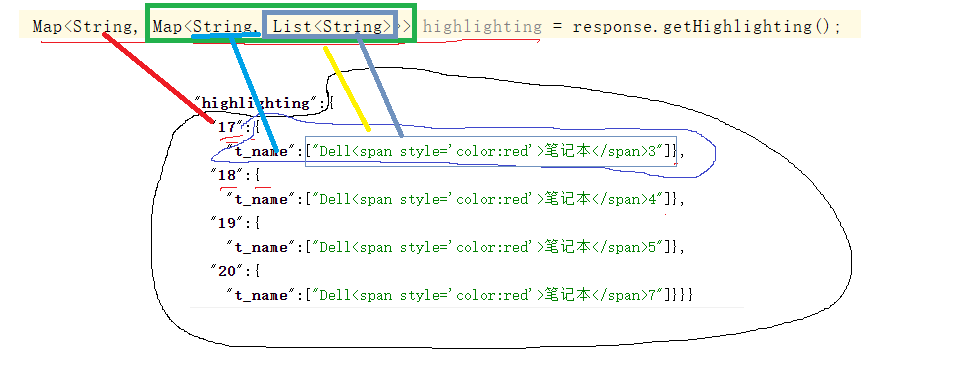
|  |
| --- |
| *//根据id来删* @Test **public void** testDelete(){  **try** {  **solrClient**.deleteById(**"1001"**);  **solrClient**.commit();  } **catch** (SolrServerException e) {  e.printStackTrace();  } **catch** (IOException e) {  e.printStackTrace();  } } |

## 3.查询数据

1）不带高亮查询

|  |
| --- |
| *//查* @Test **public void** testQueryByKeywords() **throws** IOException, SolrServerException {   SolrQuery query = **new** SolrQuery();  *//将查询条件封装进query对象中   //分页* query.setStart(0);  query.setRows(12);   *//设置复制域* query.set(**"df"**,**"t\_keywords"**);  *//name 或 salepoint 或 productdesc 只要有“笔记本”这样的关键词命中，就能被检索出来。  //设置关键字* query.setQuery(**"笔记本"**);  *//高亮   //通过客户端来根据查询条件来查询，得到响应结果* QueryResponse response = **solrClient**.query(query);   *//获取响应结果中的内容  //通过getResults得到所有的数据，每个数据是存放在一个document对象中，通过拆封document对象来获取数据==》TProductResult对象中* SolrDocumentList documents = response.getResults();  *//创建一个所有搜集结果的集合* List<TProductResult> products = **new** ArrayList<>();  **for** (SolrDocument document : documents) {  TProductResult product = **new** TProductResult();  product.setId(Long.*parseLong*((String)document.getFieldValue(**"id"**)));  product.settName((String)document.getFieldValue(**"t\_name"**));  product.settImage((String)document.getFieldValue(**"t\_image"**));  product.setSalePoint((String)document.getFieldValue(**"t\_sale\_point"**));  product.settPrice(**new** BigDecimal((Float)document.getFieldValue(**"t\_price"**)));  product.setProductDesc((String)document.getFieldValue(**"t\_product\_desc"**));  products.add(product);  }  System.***out***.println(products);   } |

2）带高亮查询



|  |
| --- |
| *//带高亮的查* @Test **public void** testQueryByHighLight(){   SolrQuery query = **new** SolrQuery();   query.set(**"df"**,**"t\_keywords"**);   query.setStart(0);  query.setRows(12);   query.setQuery(**"笔记本"**);   *//设置高亮* query.setHighlight(**true**);  query.addHighlightField(**"t\_name"**);  query.setHighlightSimplePre(**"<span style='color:red'>"**);  query.setHighlightSimplePost(**"</span>"**);    **try** {  QueryResponse response = **solrClient**.query(query);  *//获得所有的记录的集合* SolrDocumentList documents = response.getResults();  *//获得所有记录的高亮数据的集合* Map<String, Map<String, List<String>>> highlighting = response.getHighlighting();   List<TProductResult> products = **new** ArrayList<>();  **for** (SolrDocument document : documents) {   TProductResult product = **new** TProductResult();  product.setId(Long.*parseLong*((String)document.getFieldValue(**"id"**)));  product.settName((String)document.getFieldValue(**"t\_name"**));  product.settImage((String)document.getFieldValue(**"t\_image"**));  product.setSalePoint((String)document.getFieldValue(**"t\_sale\_point"**));  product.settPrice(**new** BigDecimal((Float)document.getFieldValue(**"t\_price"**)));  product.setProductDesc((String)document.getFieldValue(**"t\_product\_desc"**));   *//从highlighting这个map里把高亮的数据拿到，把product的name替换掉* Map<String, List<String>> map = highlighting.get(document.getFieldValue(**"id"**));  List<String> tnameList = map.get(**"t\_name"**);  **if**(tnameList!=**null**&&tnameList.size()>0){  String tname\_data = tnameList.get(0);  product.settName(tname\_data);  }  products.add(product);    }   System.***out***.println(products);    } **catch** (SolrServerException e) {  e.printStackTrace();  } **catch** (IOException e) {  e.printStackTrace();  }     } |

# 八、创建搜索页面，实现搜索功能

# 九、nginx的三大功能

1.反向代理

通过访问nginx，由nginx访问到目标主机。

除了可以防止外网对内网服务器的恶性攻击、缓存以减少服务器的压力和访问安全控制之外，还可以进行负载均衡，将用户请求分配给多个服务器。

2.负载均衡

3.静态资源服务器

