

Solr应用开发

www.huawei.com





目标

- 学完本课程后，您将能够：
 - 了解**Solr**应用开发适用场景
 - 熟悉**Solr**应用开发流程
 - 熟悉并使用**Solr**常用**API**
 - 理解**Collection**设计基本原则
 - 应用开发实践



目录

1. Solr应用场景
2. Solr应用开发流程
3. Collection设计
4. 应用开发案例分析
5. 常用接口示例

Solr简介

- **Solr**是一个高性能，基于**Lucene**的全文检索服务，也可以作为**NoSQL**数据库使用。
- **Solr**对**Lucene**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，还提供了一个完善的功能管理界面。
- **SolrCloud**是从**Solr 4.0**版本开始开发出的具有开创意义的分布式索引和搜索方案，基于**Solr**和**Zookeeper**进行开发的。

Solr概念体系—总述

常见术语	描述
Config Set	Solr Core 提供服务必须的一组配置文件。包括 <code>solrconfig.xml</code> (SolrConfigXml)和 <code>schema.xml</code> (SchemaXml)等。
Core	即 Solr Core ，一个 Solr 实例中包含一个或者多个 Solr Core ，每个 Solr Core 可以独立提供索引和查询功能，每个 Solr Core 对应一个索引或者 Collection 的 Shard 的副本(replica)。
Shard	Collection 的逻辑分片。每个 hard 都包含一个或者多个 Replicas ，通过选举确定哪个是 Leader 。
Collection	在 SolrCloud 集群中逻辑意义上的完整的索引。它可以被划分为一个或者多个 Shard ，它们使用相同的 Config Set 。
Replica	Shard 下的实际存储索引的一个副本，与 Core 对应。
Leader	赢得选举的 Shard Replicas 。当索引 Documents 时， SolrCloud 会传递它们到此 Shard 对应的 Leader ， Leader 再分发它们到全部 Shard 的 Replicas 。
ZooKeeper	它在 SolrCloud 是必须的，提供分布式锁、处理 Leader 选举、管理配置等功能。

Solr的常用应用场景

- 待检索数据类型复杂：如需要查询的数据有结构化数据（关系型数据库等）、半结构化数据（网页、**XML**等）、非结构化数据（日志、图片、图像等）等，而**Solr**则可以对以上数据类型进行清洗、分词、建立倒排索引等一系列操作（建立索引），然后提供**全文检索（查询）**的能力。
- 检索条件多样化（如涉及字段太多），常规查询无法满足：全文检索（查询）可以包括简单的**词**和**短语**，或者词或短语的**多种**形式。
- 读取远多于写入数据。



目录

1. Solr应用场景
2. Solr应用开发流程
3. Collection设计
4. 应用开发案例分析
5. 常用接口示例

Solr应用开发流程



Solr应用开发流程-制定业务目标

- 业务数据规模及数据模型
 - 涉及**Collection**的**Shard**划分及**Schema**定义。
- 实时索引、查询性能要求
 - 涉及**Collection**的**Shard**划分、索引存储位置。
- 查询场景
 - 涉及**Collection**的**Schema**定义。

Solr应用开发流程-准备开发环境

准备项	说明
操作系统	Windows 系统，推荐 Windows 7 以上版本。
安装 JDK	开发环境的基本配置。版本要求： 1.7 或者 1.8 。
安装和配置 Eclipse	用于开发 Solr 应用程序的工具。
网络	确保客户端与 Solr 服务主机在网络上互通。
主机名映射	hosts 文件中添加 Solr 服务器节点的 IP 和 HostName 的映射关系。
客户端与集群时间差确认	客户端机器的时间与 FusionInsight 集群的时间差需要小于 5 分钟

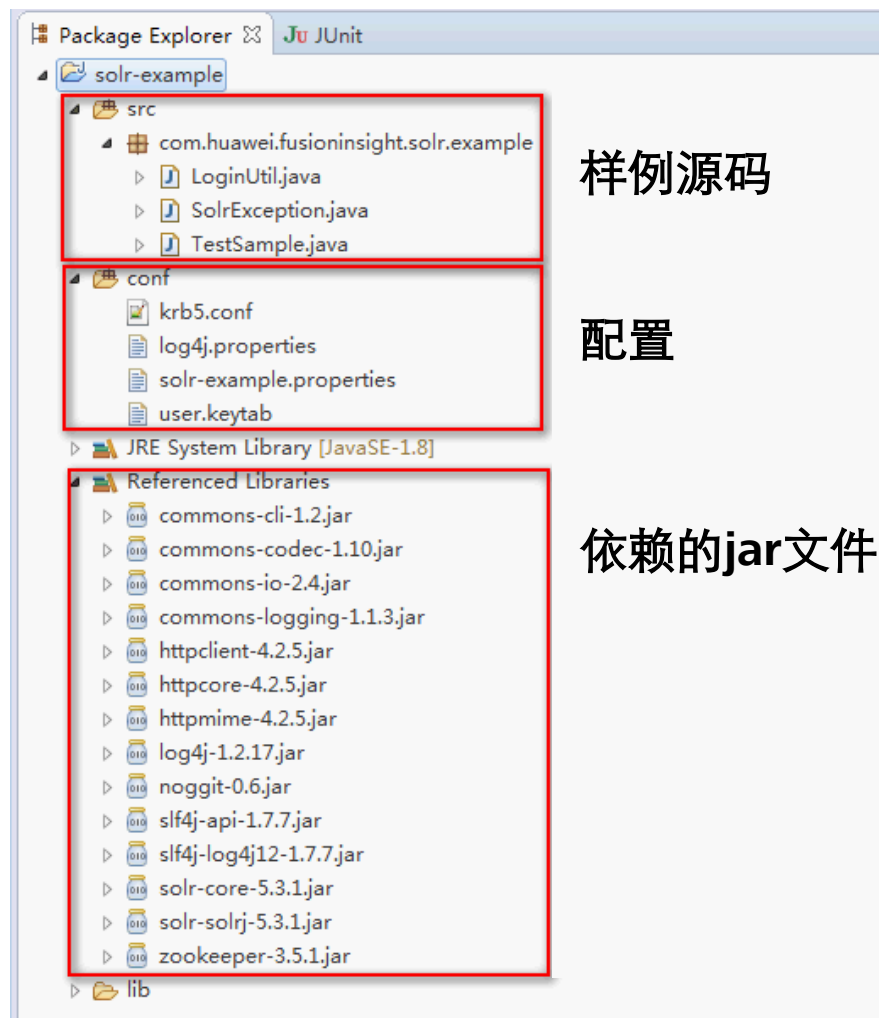
Solr应用开发流程-导入Solr样例工程

- 下载并解压**Solr**客户端压缩包。
- 导入样例工程到**Eclipse**开发环境。
- 配置**Solr**客户端样例工程“**solr-example\conf\solr-example.properties**”
 - 修改**solr-example.properties**中**ZK_URL**的值为在**Solr Admin**页面**Dashboard**下查询到的**-DzkHost**值。

Solr应用开发流程-安全配置设置

- 如果集群为安全模式：
 - **solr-example.properties**中**SOLR_KBS_ENABLED**设置为**true**。
 - 新建**Solr**用户，并下载该用户的认证凭据文件到**Solr**客户端样例工程**conf**目录下。
 - 修改**solr-example.properties**中**ZOOKEEPER_DEFAULT_SERVER_PRINCIPAL**的值为在**Solr Admin**页面**Dashboard**下查询到的**-Dzookeeper.server.principal**的值。
- 如果集群为普通模式：
 - **solr-example.properties**中**SOLR_KBS_ENABLED**设置为**false**。

Solr应用开发流程-样例工程结构



样例源码

配置

依赖的jar文件

Solr应用开发流程-初始化及安全认证

初始化并获取
配置



安全认证



```
Properties properties = new Properties();
String proPath = System.getProperty("user.dir") + File.separator +
"conf" + File.separator + "solr-example.properties";
...
SOLR_KBS_ENABLED = properties.getProperty("SOLR_KBS_ENABLED");
...
```

```
String path = System.getProperty("user.dir") + File.separator +
"conf" + File.separator;
path = path.replace("\\", "\\");
try {
LoginUtil.setJaasFile(principal, path + "user.keytab");
LoginUtil.setKrb5Config(path + "krb5.conf");
LoginUtil.setZookeeperServerPrincipal(ZOOKEEPER_DEFAULT_SERVER_
PRINCIPAL);
} catch (IOException e) {
LOG.error("Failed to set security conf", e);
throw new SolrException("Failed to set security conf");
}
```

Solr应用开发流程-连接并提交业务



获取
CloudSolrClient



```
ModifiableSolrParams params = new ModifiableSolrParams();
params.set(HttpClientUtil.PROP_MAX_CONNECTIONS, 1000);
params.set(HttpClientUtil.PROP_MAX_CONNECTIONS_PER_HOST, 500);
HttpClient httpClient = HttpClientUtil.createClient(params);

if (SOLR_KBS_ENABLED.equals("true")) {
    try {
        httpClient = new InsecureHttpClient(httpClient, params);
    } catch (Exception e) {
        LOG.error("Failed to create InsecureHttpClient", e);
        throw new SolrException("Failed to create
InsecureHttpClient");
    }
}

LBHttpSolrClient lbClient = new LBHttpSolrClient(httpClient);

CloudSolrClient cloudSolrClient = new CloudSolrClient(zkHost,
lbClient);
cloudSolrClient.setZkClientTimeout(zkClientTimeout);
cloudSolrClient.setZkConnectTimeout(zkConnectTimeout);
cloudSolrClient.connect();
```

Solr应用开发流程-连接并提交业务



调用Solr API

```
CollectionAdminResponse response =  
create.process(cloudSolrClient);  
...  
cloudSolrClient.add(documents);  
...  
QueryResponse response = cloudSolrClient.query(query);
```


Solr应用开发流程-设计Collection

- 根据业务数据的关系设计**schema.xml**
- 根据写入和查询场景设计**uniqueKey**字段
- 根据写入和查询性能要求设计**solrconfig.xml**
- 根据业务数据规模和**Solr**集群规模确定**Shard**数目
- 根据可靠性要求设定**Shard**副本数

Solr应用开发流程-根据场景开发工程

- 梳理业务场景流程
- 设计各模块接口
- 如果使用的是安全集群，需要进行安全认证
- 熟悉**Solr**提供的相应**API**
- 调用业务需要的**API**实现各功能

Solr应用开发流程-编译并运行程序

- 在开发环境**Eclipse**中，右击**TestSample.java**，单击 “**Run as > Java Application**”运行对应的样例工程。

Solr应用开发流程-查看结果与调试程序

- 在**Eclipse**的**Console**窗口可查看**Solr API**返回结果是否符合预期。
- 可以像普通**Java Application**一样，在**Eclipse**中设置断点，对**Solr**样例工程进行调试。
- 同时也可以通过登录**FusionInsight Manager**, 服务 > **Solr** > **Solr WebUI**，进入**Solr Server Admin UI**页面查看（如**Collection**是否创建成功、数据是否已写入）。



目录

1. Solr应用场景
2. Solr应用开发流程
3. Collection设计
4. 应用开发案例分析
5. 常用接口示例

Solr数据架构主要概念

术语	描述
Collection	在 SolrCloud 集群中逻辑意义上的完整的索引，对逻辑数据的逻辑存储。它可以被划分为一个或者多个 Shard ，它们使用相同的 Config Set 。
Document	存储在 SolrCloud 集群中的主要实体，是索引和查询的基本单元，可以包含一个或多个字段。文档必须包含 uniqueKey 字段。
Schema	用于定义索引数据的结构，主要包含三部分： uniqueKey 、 Field 、 FieldType 。
uniqueKey	用来标识文档唯一性的字段，更新、删除操作时会用到。
Field	Document 的主要构成单元，是更具体的信息描述，包含名称和值两部分。
FieldType	用来定义字段的类型，定义如何去处理该字段的数据，以及这个字段在查询的时如何处理。

Collection设计-索引存储位置

	索引存储在HDFS	索引存储在本地
缺点	<ul style="list-style-type: none">与存储在本地磁盘相比，性能下降30% ~ 50%。实时单节点写入速度$\leq 2\text{MB/s}$。数据膨胀率略高于存储在本地。	<ul style="list-style-type: none">需要管理磁盘，例如：选择raid等级。需要自行管理数据，例如：Solr暂时没有数据balance功能，需要规划好磁盘分区与Solr实例关系，以充分利用磁盘空间。为保证数据可靠性，需要设置多副本。
优点	<ul style="list-style-type: none">Solr设置单Replica即可，利用HDFS副本机制保障数据可靠性；数据管理由HDFS完成，包括各个节点数据balance、方便迁移。	<ul style="list-style-type: none">实时单节点写入速度在2MB/s~4MB/s之间。与存储在HDFS上相比，性能更优。

配置集solrconfig.xml

- 该文件定义**Solr**的索引、查询的处理配置和组件信息配置。
- **directoryFactory**配置定义索引存储位置：
 - 存储到HDFS: `<directoryFactory name="DirectoryFactory" class="org.apache.solr.core.HdfsDirectoryFactory">`
 - 存储本地磁盘: `<directoryFactory name="DirectoryFactory" class="org.apache.solr.core.NIOFSDirectoryFactory" />`

Collection设计-Shard划分策略

- 每个**Shard**都可以处理索引和查询请求，在设定**Shard**数目时，可从以下两方面考虑：
 - **Shard**数目最好是SolrServer实例个数的整数倍，尽可能保证每个SolrServer实例负载均衡。
 - 为了性能最优，每个**Shard**的所包含的数据不超过1亿条。

配置集Schema设计-Field定义1

- 固定字段属性：
 - **name**: 字段名
 - **type**: 字段类型
 - **indexed**: 是否被用来建立索引（关系到搜索和排序）
 - **stored**: 是否存储原值；设置为**true**，则查询时可返回原值
 - **multiValued**: 该字段在文档中是否包含多个值
 - **omitNorms**: 是否忽略标准化，可节省内存
 - **required**: 增加一个**document**时，该字段是否必须有值
 - **docValues**: 是否面向列存储，一般**uniqueKey**字段设置为**true**

配置集Schema设计-Field定义2

- 全文字段的几个属性：
 - **termVectors**: 设置为**true**时，会存储**term vector**，用于加速**MoreLikeThis**、**Highlighting**特性，有存储方面的开销。
 - **termPositions**: 是否存储**term vector**的地址信息。
 - **termOffsets**: 是否存储**term vector**的偏移量。

配置集Schema设计-Field定义3

- **dynamicField**

- 动态的字段设置，用于后期自定义字段，“*”号通配符。
- 格式：

```
<dynamicField name="*_i" type="int" indexed="true" stored="true"/>
```

- **copyField**

- 将多个字段集中到一个字段。
- 当**copyField**的**dest**字段如果有多个**source**，则该**dest**字段在定义时需**multiValued**属性设置为**true**。
- 格式：

```
<copyField source="cat" dest="text" maxChars="30000"/>
```

配置集Schema设计-普通FieldType定义

- 普通**FieldType**属性：
 - **name**: 字段类型名。
 - **class**: 字段类型实现类。
 - **positionIncrementGap**: 针对**multiValued**字段可指定一个距离，优化短语查询正确性。
 - **precisionStep**: 一般数值类型字段需要设置该值，以空间为代价换取**range search**时更快的速度。

配置集Schema设计-文本FieldType定义

- 文本**FieldType**中**analyzer**包括**tokenizer**和**filter**两部分
 - **tokenizer**用于对文本分词。
 - **filter**用于对分词的结果进行筛选过滤。
- **analyzer**可通过以下两者方式进行定义
 - 使用**org.apache.lucene.analysis.Analyzer**的子类进行定义。
 - 指定一个**TokenizerFactory**，后面跟**TokenFilterFactories**，按照所列的顺序进行处理。
 - 可通过**type**属性指定是索引还是查询阶段使用该分析器。

配置集Schema设计-示例

```
<schema name="example" version="1.5">
<field name="_version_" type="long" indexed="true" stored="true" />
<field name="id" type="string" multiValued="false" indexed="true" stored="true"
required="true" docValues="true" />
...
<field name="profile" type="text_general" multiValued="false" indexed="true" stored="false" />
<fieldType name="string" class="solr.StrField" sortMissingLast="true" />
```

```
...
<fieldType name="text_general" class="solr.TextField" positionIncrementGap="100">
<analyzer type="index">
<tokenizer class="solr.NGramTokenizerFactory" minGramSize="2" maxGramSize="2" />
</analyzer>

<analyzer type="query">
<tokenizer class="solr.NGramTokenizerFactory" minGramSize="2" maxGramSize="2" />
</analyzer>
</fieldType>
```

定义文本类型字段的在索引和查询阶段的分析器。

```
<fieldType name="ignored" stored="false" indexed="false" multiValued="true"
class="solr.StrField" />
```

文档的唯一标识，必须填写这个**field**。

```
<uniqueKey>id</uniqueKey>
```

```
<solrQueryParser defaultOperator="AND" />
</schema>
```

配置查询短语间的默认操作符，可以为“AND、OR”，如果不指定则默认为OR。

配置集创建及上传

- 在本地指定路径生成**Solr**的配置文件集
 - `solrctl confset --generate /opt/solr/test/`。
- 将设计好的**solrconfig.xml**和**schema.xml**替换至该路径**conf**目录下。
- 上传本地配置文件集给**ZooKeeper**，并指定其名称
 - `solrctl confset --create test_conf /opt/solr/test/`。
 - 配置集操作只有**solr**、**hue**或**admin**用户才有权限执行。
- 配置集创建完成后，在创建**Collection**时就可以以配置集名字**test_conf**被引用。



目录

1. Solr应用场景
2. Solr应用开发流程
3. Collection设计
4. 应用开发案例分析
5. 常用接口示例

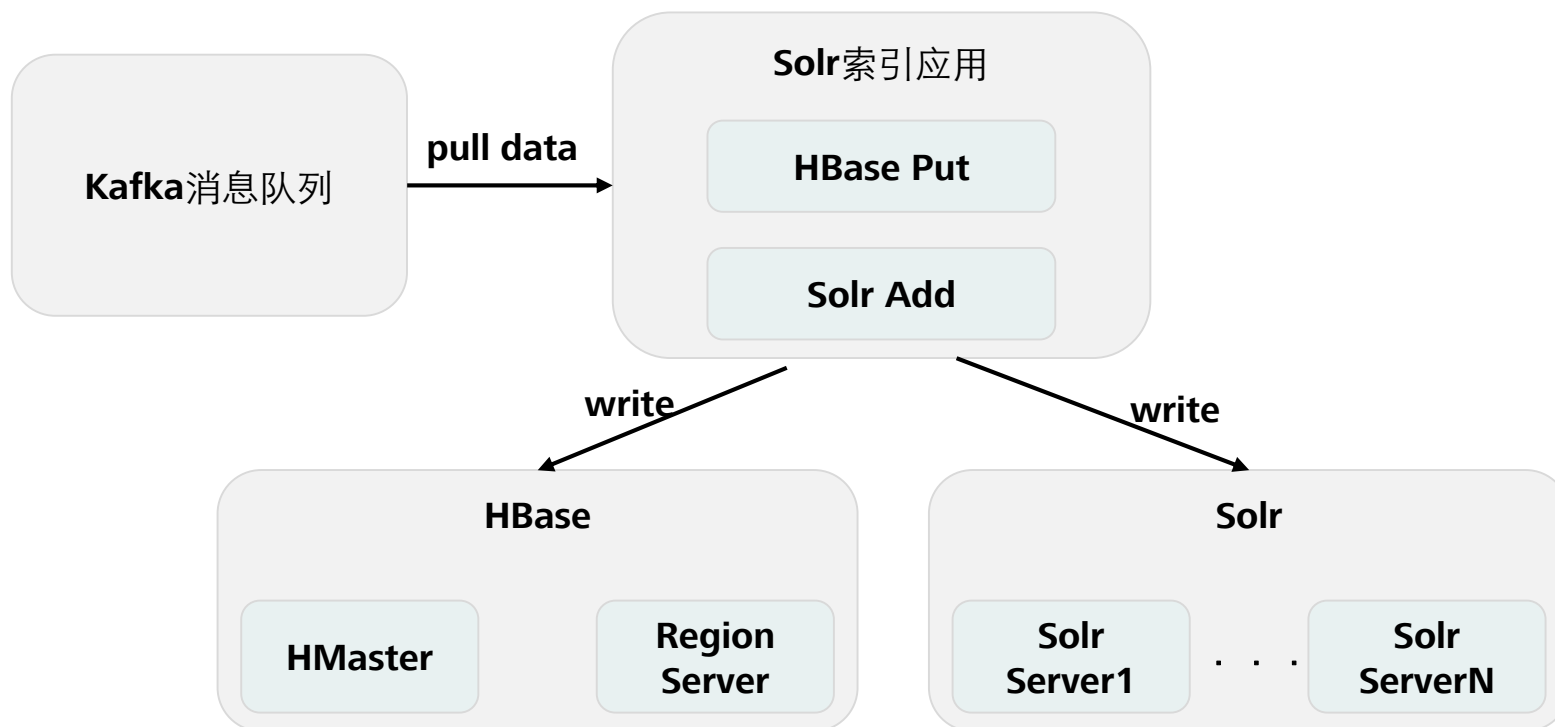
应用开发案例分析

- 业务目标：
 - 结构化数据，数据总体规模为**PB**级。
 - 支持全文检索，查询秒级响应。
 - 实时录入，写索引性能达**2MB/s**以上。
- 业务方案
 - 数据存储于**HBase**中，对查询字段在**Solr**中建立索引。
 - 涉及**RowKey**查询，通过**HBase**客户端查询；非**RowKey**字段查询通过**Solr**查询。

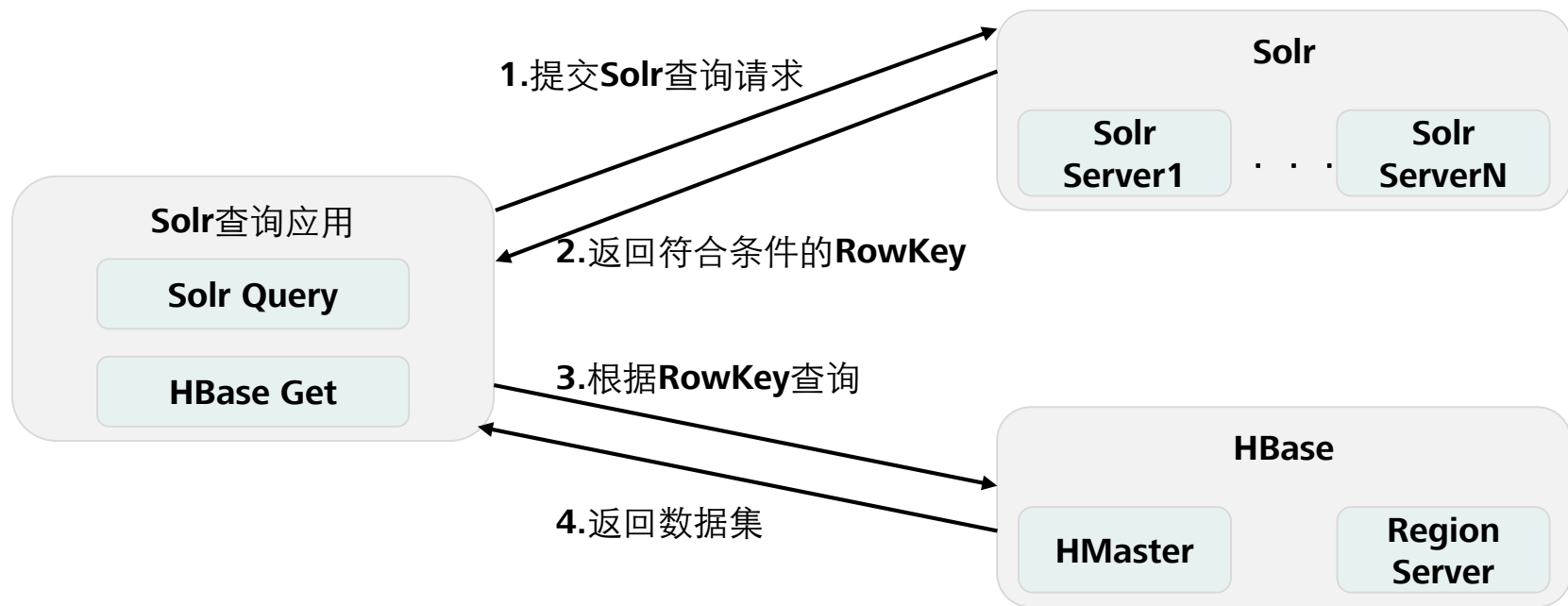
应用开发案例分析-配置集设计

- **schema**设计：
 - **uniqueKey**字段设置成**RowKey**字段。
 - 非**uniqueKey**的索引字段**stored**属性设置为**false**，不在**Solr**中保留原始值。
 - 为支持小语种及字母数字类组合查询，文本字段选择**ngram**分析器，**mingram**设置为**2**，**maxgram**同样设置为**2**。
- **solrconfig.xml**设计：
 - 由于对查询性能和索引性能要求比较高，选择索引存储在本地。

应用开发案例分析-数据录入过程



应用开发案例分析-查询过程





目录

1. Solr应用场景
2. Solr应用开发流程
3. Collection设计
4. 应用开发案例分析
5. 常用接口示例

常用Java接口-创建CloudSolrClient

```
ModifiableSolrParams params = new ModifiableSolrParams();  
params.set(HttpClientUtil.PROP_MAX_CONNECTIONS, 1000);  
params.set(HttpClientUtil.PROP_MAX_CONNECTIONS_PER_HOST,  
500);  
HttpClient httpClient =  
HttpClientUtil.createClient(params);
```

HttpClient内带有连接池，可传入参数控制并发量。

```
if (SOLR_KBS_ENABLED.equals("true")) {  
    httpClient = new InsecureHttpClient(httpClient, params);  
}  
  
LBHttpSolrClient lbClient = new  
LBHttpSolrClient(httpClient);  
// using BinaryRequestWriter, it can reduce indexing I/O.  
lbClient.setRequestWriter(new BinaryRequestWriter());
```

集群处于安全模式下，需传入经由InsecureHttpClient封装过的HttpClient。

为减少网络I/O，提高索引速度，在索引时可使用BinaryRequestWriter。

```
CloudSolrClient cloudSolrClient = new  
CloudSolrClient(zkHost, lbClient);  
cloudSolrClient.setZkClientTimeout(zkClientTimeout);  
cloudSolrClient.setZkConnectTimeout(zkConnectTimeout);  
  
cloudSolrClient.connect();
```

常用Java接口-创建Collection

```
CollectionAdminResponse rsp = new  
CollectionAdminRequest.Create()  
    .setCollectionName(collectionName)  
    .setNumShards(numOfShards)  
    .setConfigName(confSetName)  
    .setReplicationFactor(replicationFactor)  
    .setMaxShardsPerNode(maxShardsPerNode)  
    .process(cloudSolrClient);
```

要创建的**Collection**名字。

Collection中**Shard**数目。

创建**Collection**指定的配置集名字。

副本数。

每个主机上最大**Shard**数目。

常用Java接口-索引

```
UpdateRequest updateRequest = new UpdateRequest();  
Set<Map.Entry<String, String>> entrySet = null;
```

```
for (Map<String, String> keyValMap : keyValList) {  
    /* bulk submit once every 10,000 docs, it can improve  
     * indexing performance. method "getDocuments" in  
     * UpdateRequest is not recommended, because it is  
     * return a copy of the docs*/  
    if (updateRequest.getDocumentsMap().size() >= 10000)
```

```
{  
    cloudSolrClient.request(updateRequest, collName);  
    updateRequest.clear();}
```

```
entrySet = keyValMap.entrySet();  
SolrInputDocument doc = new SolrInputDocument();  
for (Map.Entry<String, String> entry : entrySet) {  
    doc.addField(entry.getKey(), entry.getValue());  
}
```

```
entrySet = keyValMap.entrySet();
```

```
SolrInputDocument doc = new SolrInputDocument();
```

```
for (Map.Entry<String, String> entry : entrySet) {  
    doc.addField(entry.getKey(), entry.getValue());  
}
```

```
updateRequest.add(doc, false);  
}
```

```
if (updateRequest.getDocumentsMap().size() > 0) {  
    cloudSolrClient.request(updateRequest, collName);  
}
```

UpdateRequest中的getDocuments()”方法，是copy了一份已被添加进去的doc对象，需使用getDocumentsMap()接口来获取已添加进去的doc数目。

构造SolrInputDocument对象，一个对象对应一条记录；该对象必须添加uniqueKey字段。

UpdateRequest中添加一个Doc对象：add(final SolrInputDocument doc, Boolean overwrite)。如果overwrite为true，则覆盖该uniqueKey对应的Doc。

常用Java接口-查询

```
SolrQuery solrQuery = new SolrQuery()
    .setStart(start).setRows(rows)
    .setFields(returnFields)
    .setQuery(queryStr)
    .setFilterQueries(filterQuery);

solrQuery.set("shards.tolerant", true);
solrQuery.set("collection", collection);

QueryResponse response =
    cloudSolrClient.query(collection, solrQuery);
SolrDocumentList docs = response.getResults();

LOG.info("Total doc num found : {}",
    docs.getNumFound());

for (SolrDocument doc : docs) {
    LOG.info("doc detail : " + doc.getFieldValueMap());
}
```

设置查询的返回字段，多个字段以逗号分隔。
示例：**"id,name,price"**。

设置主查询，示例：
"store:local AND content_type:xml"，查询
store值为local，同时字段content_type值为
xml的文档。

设置过滤查询，示例：
"mod_date:[20160401 TO 20160430 "，过
滤修改日期为2016年4月份的文档。

设置要查询的Collection。

常用Java接口-使用游标进行深度翻页

```
SolrQuery solrQuery = new SolrQuery()
    .setQuery(queryStr)
    .setFilterQueries(filterQuery)
    .setRows(rows).setSort(SortClause.asc("id"))
    .setFields(returnFields);
```

查询的条件里必须按照主键排序（升序或降序）。

```
solrQuery.set("shards.tolerant", true);
solrQuery.set("collection", collection);
```

```
boolean done = false;
QueryResponse response = null;
String cursorMark = CursorMarkParams.CURSOR_MARK_START;
String nextCursorMark = null;
```

```
while (!done) {
    solrQuery.set(CursorMarkParams.CURSOR_MARK_PARAM, cursorMark);
    response = solrClient.query(solrQuery);
    nextCursorMark = response.getNextCursorMark();
    SolrDocumentList solrDocumentList = response.getResults();
    for (SolrDocument doc : solrDocumentList) {
        // process documents
    }
    if (cursorMark.equals(nextCursorMark)) {
        done = true;
    }
    cursorMark = nextCursorMark;
}
```

查询条件里面必须有 **cursorMark** 参数，且第一次请求时必须设置为 **CursorMarkParams.CURSOR_MARK_START**；而且必须不能有 **start** 参数，即 **start** 需被设置为 0。

游标一旦读取了，就不能再返回上一次位置。

将 **nextCursorMark** 作为下一次结果集的开始位置。

常用Java接口-删除文档

// id is uniqueKey field's value
`cloudSolrClient.deleteById(id);`

通过**uniqueKey**字段，
删除单个**doc**。

// ids is uniqueKey field's values list
`cloudSolrClient.deleteById(ids);`

通过**uniqueKey**字段列
表，删除多个**Docs**。

`cloudSolrClient.deleteByQuery(collection, queryStr);`

删除查询所得的**Doc**。

常用Java接口-删除指定Collection

```
CollectionAdminResponse response = new  
CollectionAdminRequest.Delete()  
    .setCollectionName(collection)  
    .process(cloudSolrClient);
```

要删除Collection名字

常用Java接口-查看所有Collection

```
CollectionAdminResponse response = new  
CollectionAdminRequest.List()  
    .process(cloudSolrClient);  
List<String> collections =  
    (List<String>)  
response.getResponse().get("collections");
```

常用WebUI操作

- 登录**FusionInsight Manager**，选择服务管理 > **Solr**，点击“**Solr 概述**”面板中的“**Solr WebUI**”的“**SolrServerAdmin**”（两个任选一个），进入**Solr Admin Dashboard**的页面。
- **Dashboard**页面，可以查看到**SolrServerAdmin**实例启动时间、**Solr**及其底层**Lucene**版本信息、实例占用资源、**JVM**启动参数。

常用WebUI操作-查看Solr元数据信息



Dashboard

Logging

Cloud

Tree

Graph

Graph (Radial)

Dump

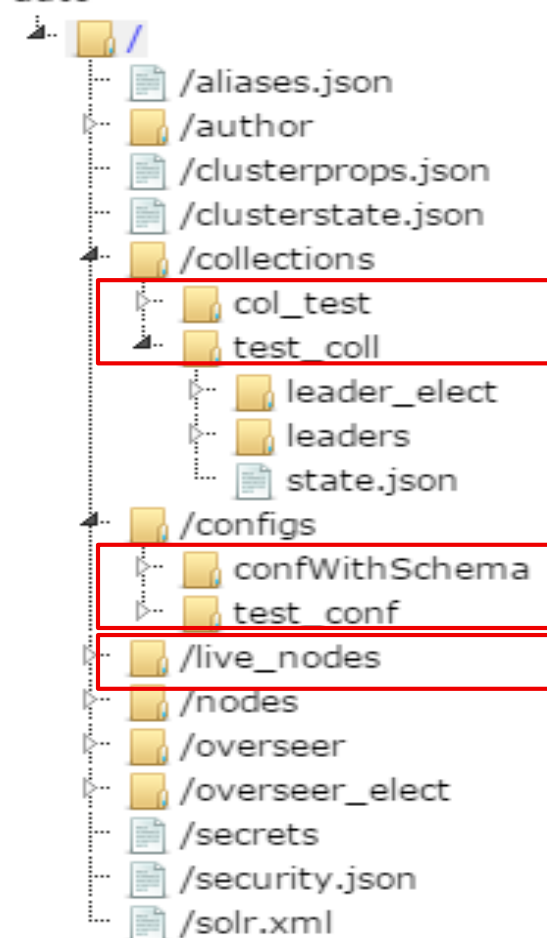
Core Admin

Java Properties

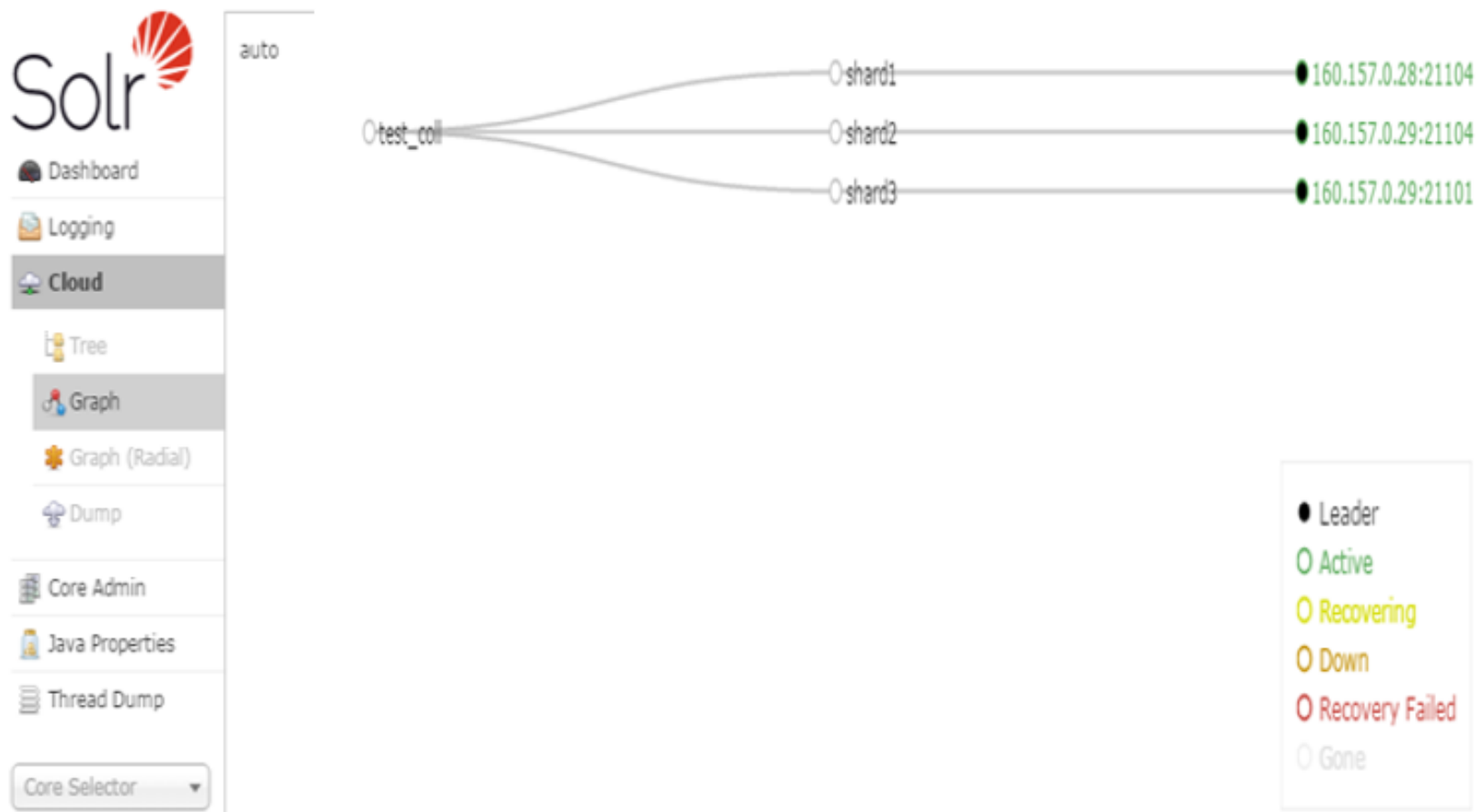
Thread Dump

Core Selector

auto




常用WebUI操作-查看Collections



常用WebUI操作-查看Replica总体情况

- 在**Solr Admin**页面,点击**Core Selector**,选择**Core**(即**Replica**),进入对应**Core**的**Overview**页面。
- 在**Overview**页面可以查看到,该**Replica**的统计数据及索引数据存放位置。

常用WebUI操作-续

 **Instance**

CWD: /opt/huawei/Bigdata/nodeagent


Instance: /srv/BigData/solr/solrserveradmin/test_coll_shard3_replica1





Data: hdfs://hacluster/user/solr/SolrServerAdmin/test_coll/core_node1/data

Index: hdfs://hacluster/user/solr/SolrServerAdmin/test_coll/core_node1/data/index

Impl: org.apache.solr.core.HdfsDirectoryFactory

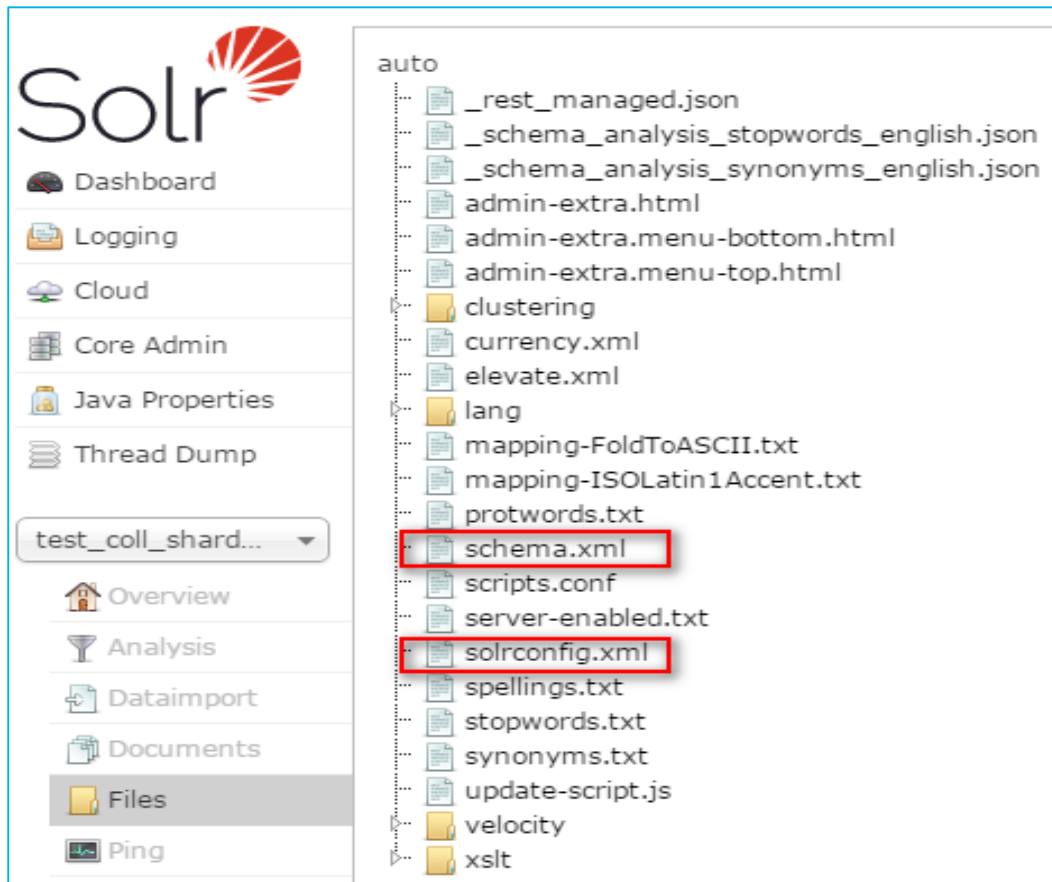
Healthcheck

Status: 

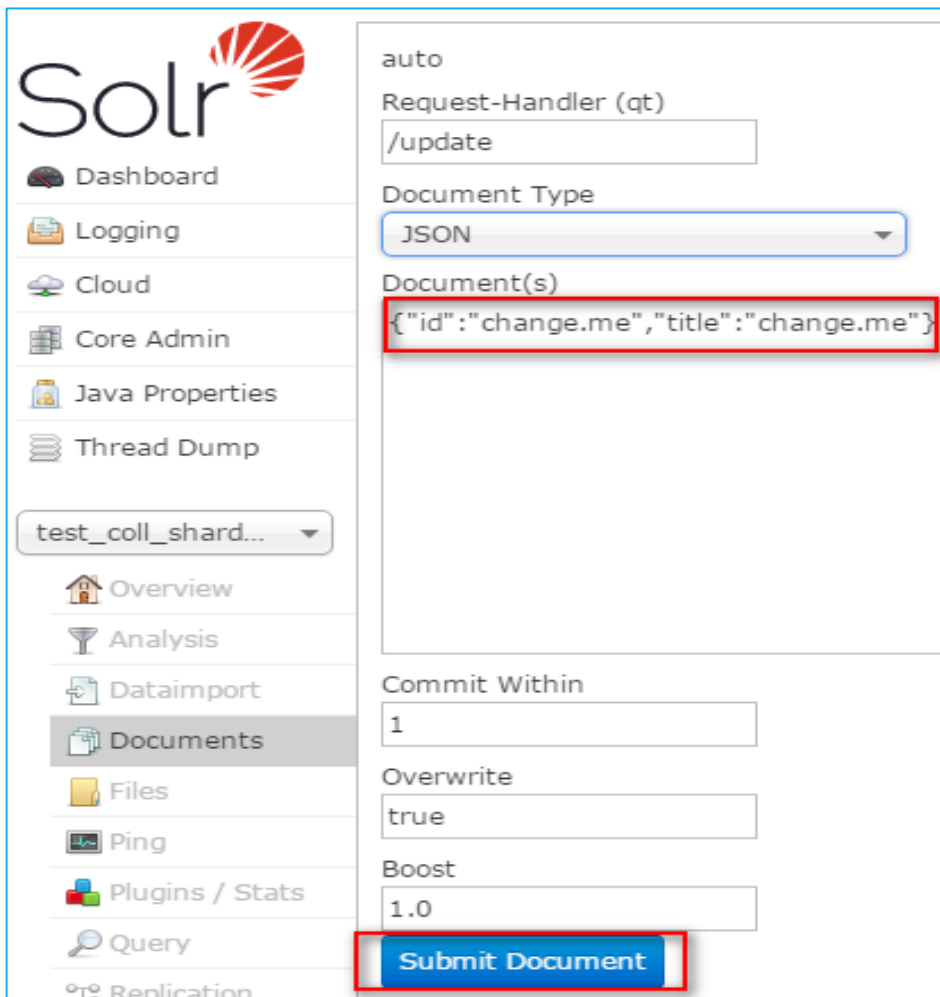
 [Documentation](#)  [Issue Tracker](#)  [IRC Channel](#)  [Community forum](#)

常用WebUI操作-查看Replica的配置集

- 在Solr Admin页面,点击Core Selector,选择Core(即Replica), 单击“Files”，可以查看该Replica所使用的配置集。



常用WebUI操作-添加文档



Solr

- Dashboard
- Logging
- Cloud
- Core Admin
- Java Properties
- Thread Dump

test_coll_shard...

- Overview
- Analysis
- Dataimport
- Documents**
- Files
- Ping
- Plugins / Stats
- Query
- Replication

auto

Request-Handler (qt)

/update

Document Type

JSON

Document(s)

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

Commit Within

1

Overwrite

true

Boost

1.0

Submit Document

- 在Solr Admin页面,点击Core Selector,选择Core(即Replica), 点击 “Documents” 。
- 在Document(s)文本框中, 以JSON格式, 输入要添加的Document。
- 点击Submit Document按钮, 提交该Document。

常用WebUI操作-查询



- Dashboard
- Logging
- Cloud
- Core Admin
- Java Properties
- Thread Dump
- test_coll_shard...
- Overview
- Analysis
- Dataimport
- Documents
- Files
- Ping
- Plugins / Stats
- Query
- Replication
- Schema Browser
- Segments info

auto
Request-Handler (qt)
/select

— common —

q
:

fq

sort

start, rows
0 10

fl

df

Raw Query Parameters
key1=val1&key2=val2

wt
json

☒ indent
☐ debugQuery

☐ dismax
☐ edismax
☐ hl
☐ facet
☐ spatial
☐ spellcheck

Execute Query

https://160.157.0.27:20026/solr/test

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 75,
    "params": {
      "q": "*:*",
      "indent": "true",
      "wt": "json",
      "_": "1465285473801"
    }
  },
  "response": {
    "numFound": 1,
    "start": 0,
    "maxScore": 1,
    "docs": [
      {
        "id": "change.me",
        "title": [
          "change.me"
        ],
        "_version_": 1536462859019485200
      }
    ]
  }
}
```

本章总结

- 本章主要介绍了**Solr**应用开发流程、**Collection**的设计基本原则以及分析了具体的应用案例，也介绍了常用的接口。学完本章后，可以根据具体的业务场景，设计**Collection**，并写入索引及进行基本的查询。

习题

1. 【单选】 客户端与集群时间差默认不小于（ ）。

A. 1分钟

B. 5分钟

C. 10分钟

D. 15分钟

2. 【单选】 定义以下哪种字段支持后期自定义增加字段（ ）。

A. dynamicField

B. copyField

C. field

D. 以上都是

习题—续

3. 【单选】 样例工程solr-example\conf\solr-example.properties中ZK_URL和ZOOKEEPER_DEFAULT_SERVER_PRINCIPAL的值可以从（ ）获得到。

- A. Solr Admin UI Dashboard JVM面板
- B. 该配置文件中自带
- C. 从FusionInsight Manager Solr服务配置页面获取
- D. A、C选项均可获取

思考题

- 设计并创建**Collection**
 - 索引性能要达到**3MB/s**;
 - 数据规模为**8亿**条;
 - 索引数据模型为**name**、**age**、**location**、**timestamp**、**msg**（文本字段）。
- 开发索引应用
- 开发查询应用
 - 查询**age**在**23~40**之间的**docs**;
 - 查询**msg**中含有指定词语的**docs**。

学习推荐

- **apache-solr-ref-guide**，详细的**schema**定义及查询语法介绍

<https://archive.apache.org/dist/lucene/solr/ref-guide/apache-solr-ref-guide-5.3.pdf>

附录：创建Solr admin role

- 登录**FusionInsight Manager**，选择 系统设置> 角色管理，点击添加角色按钮，勾选**SUPER_USER_GROUP**，点击确定，完成**Solr admin role**角色创建。



FusionInsight Manager

系统概览 服务管理 主机管理 告警管理 审计管理 租户管理 **系统设置**

系统设置 > 角色管理 > 添加角色

*角色名称: solr_admin_role ✓

权限:

服务 > Solr

视图名称
Solr Scope
<input checked="" type="checkbox"/> SUPER_USER_GROUP

当前页: 1 总页数: 1 | 跳转到 确定

描述: 绑定该角色的用户可以创建Collection，并对Solr服务内所有的Collection具有删除等权限。 ✓

确定 取消

附录：创建用户属于指定用户组及角色

- 登录FusionInsight Manager，选择 系统设置> 用户管理，点击添加用户按钮

FusionInsight Manager

系统设置 > 用户管理 > 添加用户

* 用户名: solr_user ✓

* 用户类型: 机机 ✓

* 用户组: [选择添加的用户组](#) [删除全部](#)

solr X

* 主组: solr

分配角色权限: [选择绑定角色](#) [删除全部](#)

solr_role X

电话:

邮箱:

描述:

确定 取消

单击“选择添加的用户组”，选择对应用户组将用户添加进去。

根据实际需要在“分配角色权限”，单击“选择绑定角色”为用户添加角色。

附录：资源权限管理

操作	所需权限
Collection Create	Solr group、Solr admin role
Collection Delete	Collection owner、Solr admin role
Collection Read/Write	Collection owner、Solr Role、Solr admin role
Create/Split Shard	Collection owner、Solr admin role
Replica Add/Delete	Collection owner、Solr admin role
Solr Admin UI	Solr group、Super group、Solr admin role

附录：资源权限管理-续

操作	所需权限
Shell solrctl操作	solr、hue或admin用户
Hue Search	admin用户、hue用户
对HBase数据索引	用户需要对要访问的资源有如下权限： <ul style="list-style-type: none">要操作的HBase表：读写权限，如涉及到表创建则需要创建、管理权限。要操作的Collection：读写权限，如涉及到索引创建，则需要有创建权限。
对HDFS数据索引	用户需要对要访问的资源有如下权限： <ul style="list-style-type: none">要操作的HDFS文件：读写权限，如涉及到文件创建删除则需要创建、管理权限。要操作的Collection：读写权限，如涉及到索引创建，则需要有创建权限。

Thank you

www.huawei.com