关于金融事件抽取应用的部署、启动、调用及更新维护

目录

[项目部署 1](#_Toc528341976)

[前期准备： 1](#_Toc528341977)

[启动Tomcat生成编译好的项目文件 1](#_Toc528341978)

[关闭Tomcat编辑项目配置文件 2](#_Toc528341979)

[配置hownet解析器路径 3](#_Toc528341980)

[项目启动与关闭 4](#_Toc528341981)

[接口详解 6](#_Toc528341982)

[健康测试接口信息 6](#_Toc528341983)

[主接口信息 6](#_Toc528341984)

[调用示例 7](#_Toc528341985)

[代码示例（以主接口调用为例）： 7](#_Toc528341986)

[返回结果示例： 9](#_Toc528341987)

[解析结果节点介绍 11](#_Toc528341988)

[日志信息详解 12](#_Toc528341989)

[日志文件介绍： 12](#_Toc528341990)

[日志路径及保存天数配置 12](#_Toc528341991)

[日志文件内容示例： 13](#_Toc528341992)

[项目更新及维护 18](#_Toc528341993)

[替换操作 18](#_Toc528341994)

[替换词典和规则文件 18](#_Toc528341995)

[替换完成后重启 20](#_Toc528341996)

[“RelationMappingTalbe.txt”文件 20](#_Toc528341997)

[文件内容介绍 20](#_Toc528341998)

[文件更新 20](#_Toc528341999)

# 项目部署

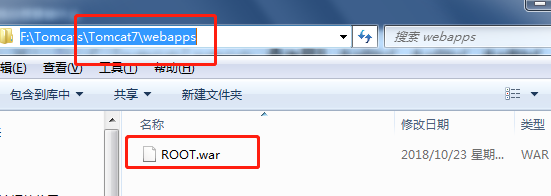
## 部署环境：

1. 64位Windows系统（由于HowNet目前只支持Windows环境，并且在短时间内无法开发出一套可以应用于Linux的版本）
2. Microsoft Visual C++运行库
3. Jdk 1.7.x版本（必须为32位）
4. Tomcat 7.x 版本（32位/64位均可）

硬件环境则根据部署的HowNet数量来定，单机HowNet最低配置2核4G，硬件配置将直接影响HowNet的解析速率

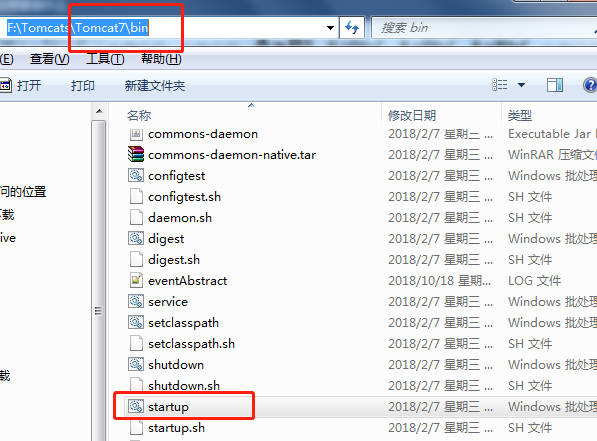
## 前期准备：

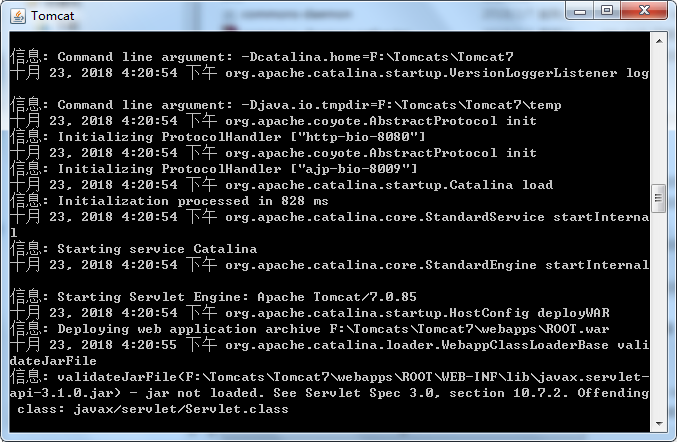
配置好Tomcat端口号（Tomcat端口号配置请自行百度，此处不再讲解）。然后清空”Tomcat/webapps/”下的内容，然后将项目（war包）放入”Tomcat/webapps/”下，并重命名为“ROOT”，如下图：



## 启动Tomcat生成编译好的项目文件

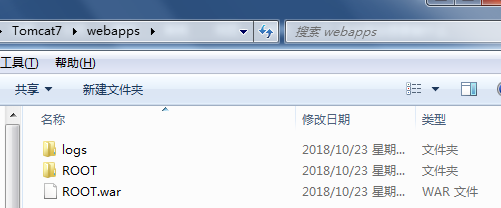
双击”Tomcat\bin”目录下的startup.bat文件,并弹出黑窗口：





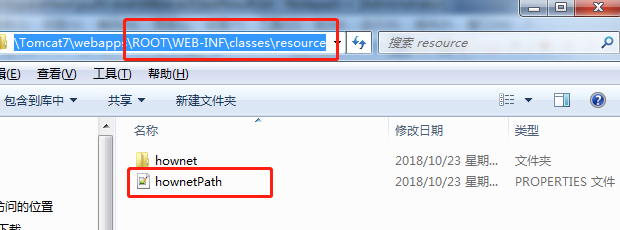
## 关闭Tomcat编辑项目配置文件

等Tomcat加载信息打印完毕后，关掉此窗口。此时，返回到“Tomcat\webapps”目录下，”ROOT.war”项目已经编译生成ROOT文件夹，此时可将 “ROOT.war”文件删除，只保留ROOT文件夹即可。

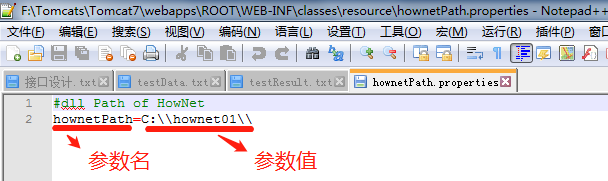


### 配置hownet解析器路径

1. 进入ROOT文件夹，在“ROOT\WEB-INF\classes\resource”目录下，找到“hownetPath.properties”文件：

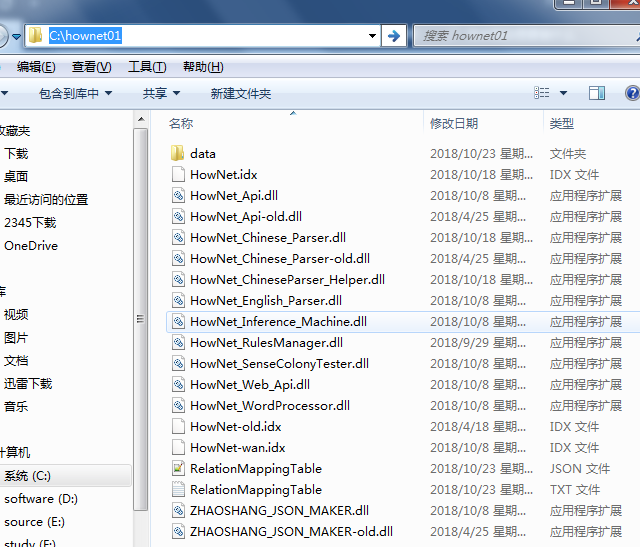


1. 编辑该文件，配置HowNet动态库的路径，只需改动等号后面的值。此时的参数值为“C:\\hownet01\\”。请务必按照“参数配置图1”中的格式配置路径，尤其是“\\”！！！



参数配置图 1

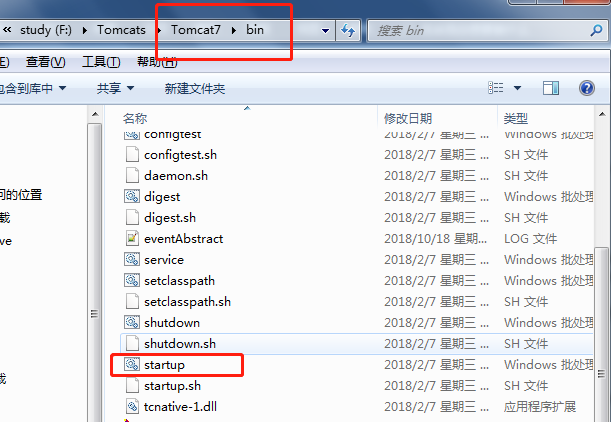
以上配置对应的是存放在c盘下的hownet01文件夹下的HowNet动态库：

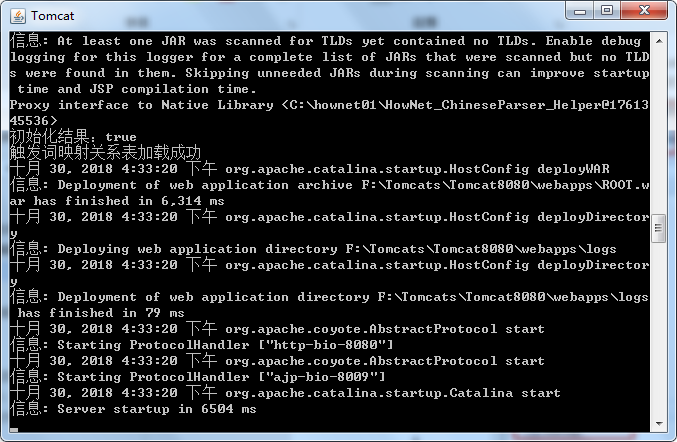


更改完毕后保存，自此，单套应用部署完成。

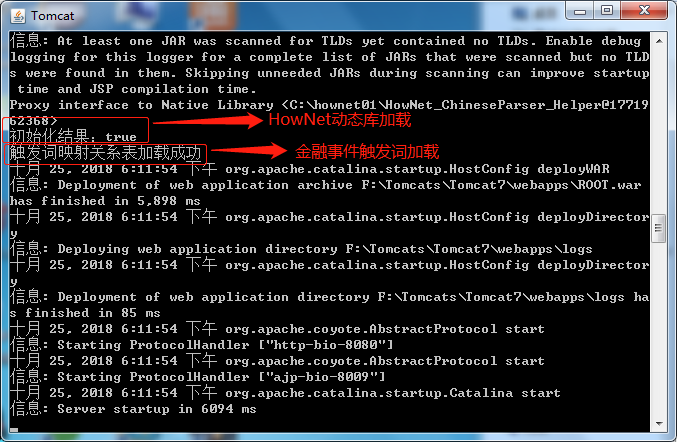
# 项目启动与停止

1. 进入刚刚部署好的Tomcat中，在“Tomcat\bin”目录下，找到“startup.bat”文件，双击运行，弹出黑窗口。停止服务时，只需将黑窗口关闭即可。





1. 若Tomcat黑窗口中正常打印了如下信息，并且没有抛出异常信息，则说明Tomcat正常启动。（备注：“初始化结果：true”信息表示HowNet动态库初始化成功；“触发词映射关系表加载成功”信息则表示触发词关系映射表加载完成；二者缺一不可。）



# 接口详解

## 健康测试接口信息

**接口地址：**<http://IP:Port/api/keepAlived.do>

**请求方式：**HTTP POST请求

**接口参数：**无

**接口返回值：**

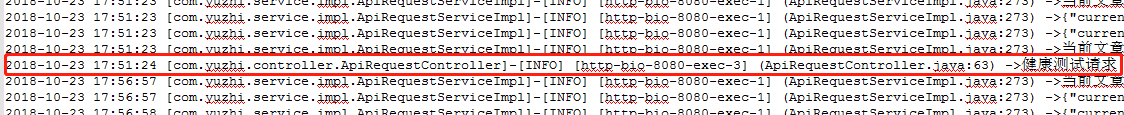
若应用正常，则返回字符串“Alived”。

若Java应用不正常，则无响应。

若Java应用正常，但是HowNet无响应，则返回“HowNet is Dead!!”(此类情况很少见)。

**接口日志打印：**

当用户发送“keepAlived.do”请求时，Java应用打印“健康测试请求”。如下图：



## 主接口信息

**接口地址：**<http://IP:Port/api/eventAbstract.do>

**请求方式：**HTTP POST请求

**接口参数：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名称 | 参数介绍 | 参数类型 | 参数示例 | 备注 |
| articleID | 文章的唯一标识，仅用于后期查看解析日志信息 | String（字符串，建议为UUID） | 4b5a0c0a-5077-4069-9b99-8b365f7cc60c | 必填参数 |
| article | 文章内容，待解析文本 | String | 金融事件文章（建议500字符以内） | 必填参数 |

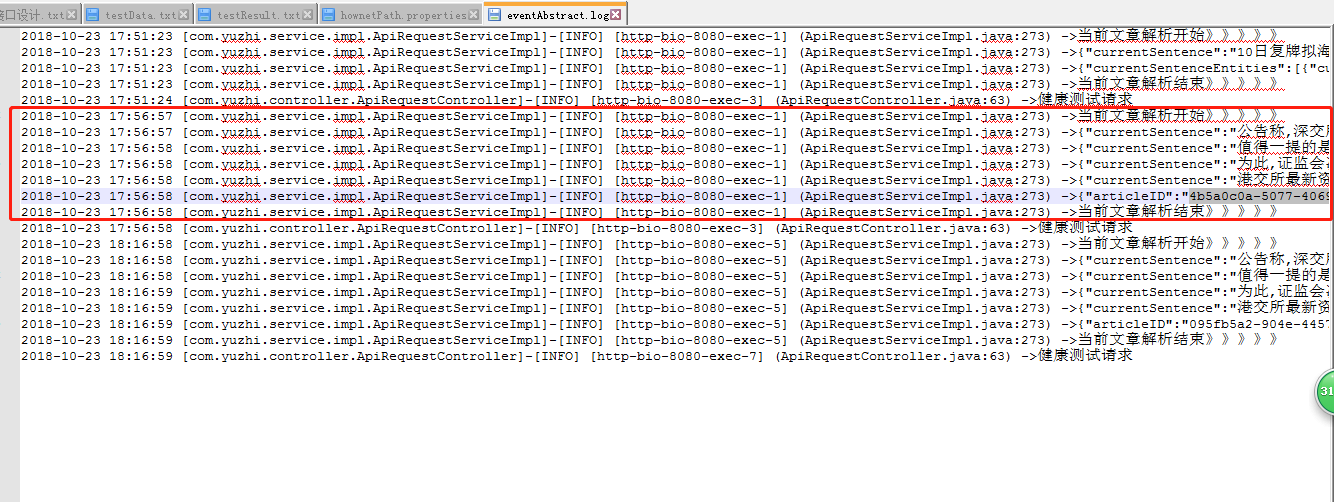
**接口返回值（返回值详情请参照该文档后面的【调用示例】一项）：**

若项目正常运行，返回Json格式的字符串

若项目未正常运行，如HowNet或Tomcat挂掉，则无响应

**接口日志打印（返回值详情请参照该文档后面的【日志信息详解】一项）：**

当用户发送“eventAbstract.do”请求时，Java应用打印文章中每一句的解析日志，并在解析结束之前最后一行，打印整句话的日志信息，日志信息是可读的Json字符串。如下图，红框中的内容为一次正常解析的日志打印。若中途HowNet挂掉，则日志信息会停留在文章中最后一句完成正确解析的句子处。



# 调用示例

## 代码示例（以主接口调用为例）：

|  |
| --- |
| **public** **static** **void** main(String[] args){  // 创建Httpclient对象  CloseableHttpClient httpclient = HttpClients.*createDefault*();  // 创建http POST请求  HttpPost httpPost = **new** HttpPost("http://99.13.33.27:8080/api/eventAbstract.do");  // 设置2个post参数，一个是articleID、一个是article  List<NameValuePair> parameters = **new** ArrayList<NameValuePair>();  parameters.add(**new** BasicNameValuePair("articleID", UUID.*randomUUID*().toString()));  //示例  parameters.add(**new** BasicNameValuePair("article", "百度公司今年利润出现大幅下滑，为此，深交所向公司发出了问询函。公告称,深交所于4月12日向公司发出了重组问询函,公司就问询函提及的问题进行了反馈。值得一提的是,美利股价异常波动情况引起了深交所关注,并在3月7日向美利股份有限公司发出了《问询函》。为此,证监会决定对人保财险宝应支公司罚款20万元。港交所最新资料显示,华(00628)主要股东ChinWanYi场内减持公司好仓195.2万股,成交均价1.803港币。"));  CloseableHttpResponse response = **null**;  **try** {  // 构造一个form表单式的实体  UrlEncodedFormEntity formEntity;  formEntity = **new** UrlEncodedFormEntity(parameters, "UTF-8");  // 将请求实体设置到httpPost对象中  httpPost.setEntity(formEntity);  // 执行请求  response = httpclient.execute(httpPost);  // 判断返回状态是否为200  **if** (response.getStatusLine().getStatusCode() == 200) {  //调用结果  String content = EntityUtils.*toString*(response.getEntity(), "UTF-8");  //打印返回的信息  System.***out***.println(content);  //通过Alibaba的Fastjson，将解析结果提取出来  JSONObject object = JSONObject.*parseObject*(content);  String result ="";  **if**(object.get("success")!=**null**&&!"".equals(object.get("success"))){  result = object.get("success").toString();  }**else** **if**(object.get("error")!=**null**&&!"".equals(object.get("error"))){  result = object.get("error").toString();  }**else**{  System.***out***.println("返回为空");  }  //打印解析结果  System.***out***.println(result);  }  } **catch** (Exception e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  } **finally** {  **try** {  **if** (response != **null**) {  response.close();  }  httpclient.close();  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  }  } |

## 返回结果示例：

1. **正常返回结果：**

|  |
| --- |
| **[**  **{**  **"sentence"**: "百度公司今年利润出现大幅下滑，为此，深交所向公司发出了问询函。"**,**  **"tags"**: **[**  **{**  **"type"**: "问询"**,**  **"trigger"**: **[**  "问询函"  **],**  **"trigger\_position"**: **[**  **[**  27**,**  30  **]**  **],**  **"sub"**: **[ ],**  **"obj"**: **[**  "深交所"  **],**  **"date"**: **[ ],**  **"amount"**: **[ ]**  **},**  **{**  **"type"**: "利润下滑"**,**  **"trigger"**: **[**  "利润"**,**  "大幅下滑"  **],**  **"trigger\_position"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif  **[**  6**,**  8  **],**  **[**  10**,**  14  **]**  **],**  **"sub"**: **[**  "百度公司"**,**  "大幅下滑"  **],**  **"obj"**: **[ ],**  **"date"**: **[**  "今年"  **],**  **"amount"**: **[ ]**  **}**  **]**  **},**  **{**  **"sentence"**: "公告称,深交所于4月12日向公司发出了重组问询函,公司就问询函提及的问题进行了反馈。"**,**  **"tags"**: **[**。。。**]**  **},**  **{**  **"sentence"**: "值得一提的是,美利股价异常波动情况引起了深交所关注,并在3月7日向美利股份有限公司发出了《问询函》。"**,**  **"tags"**: **[**。。。**]**  **},**  **{}。。。**  **]** |

1. **若解析的文章中没有打上标签，则返回：**

|  |
| --- |
| **[ ]** |

1. **若参数格式不正确，导致解析错误，则返回：**

|  |
| --- |
| ["文章ID不能为空！"] |

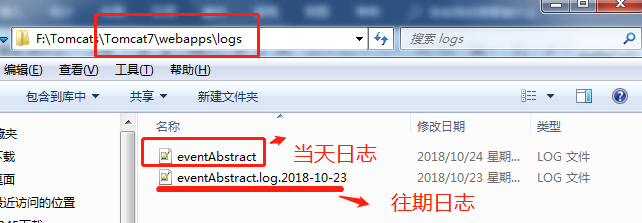
## 解析结果节点介绍

|  |
| --- |
| [  {  #文章中含有金融事件的第一个句子，被HowNet预处理过的  "sentence": unicode string  #该句里面所有的事件标签列表  "tags":[  # 每个事件标签的json体  {  # 类型  u"type": unicode string,  # 触发词  u"trigger": [unicode string],  # 触发词在(HowNet处理过的原文)句子中的位置，第一个整型为第一个字符index（以0开始），第二个整型为最后一个字符index+1  # 与触发词字段一一对应  u"trigger\_position": [ [int, int] , [int, int] ],  # 主体，可以为空  u"sub": [unicode string],  # 客体，可以为空  u"obj": [unicode string],  # 时间，可以为空  u"date": [unicode string],  # 程度，可以为空  u"amount": [unicode string]  },  {  ...  }  ]  },  {  #文章中含有金融事件的第二个句子，被HowNet预处理过的  "sentence": "",  "tags": [ ]  }  ] |

# 日志信息详解

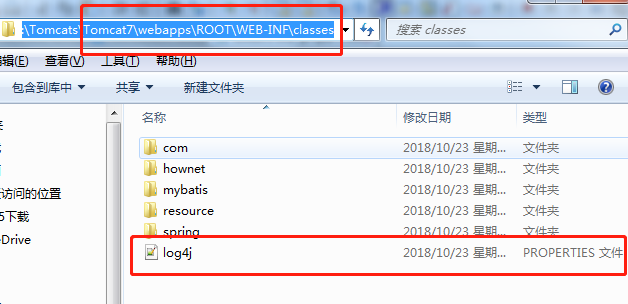
## 日志文件介绍：

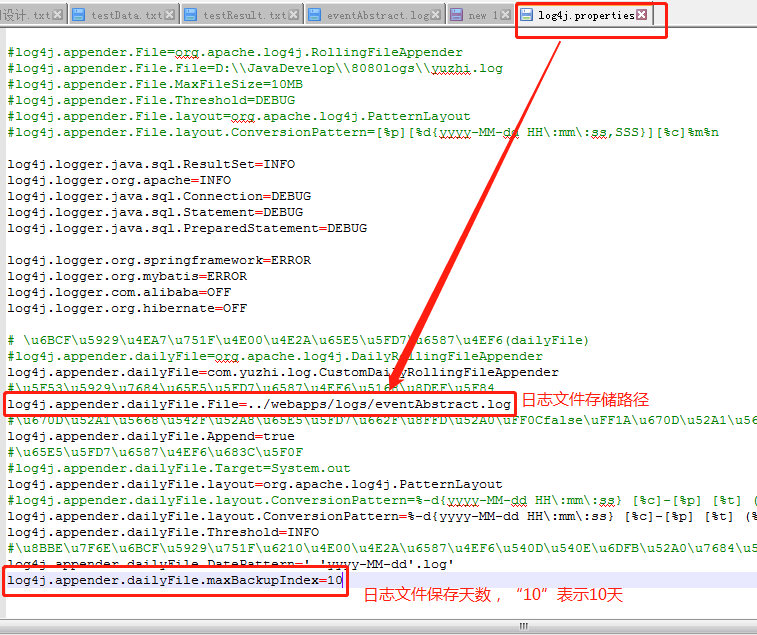
本应用日志借助Log4j进行的打印，日志文件存储目录为：Tomcat\webapps\logs\



## 日志路径及保存天数配置

日志文件的存储路径和保留天数可在“Tomcat\webapps\ROOT\WEB-INF\classes\”目录下的”log4j.properties”中配置：





## 日志文件内容示例：

### 日志头信息

这是每行日志的开头信息，这个格式是固定的。

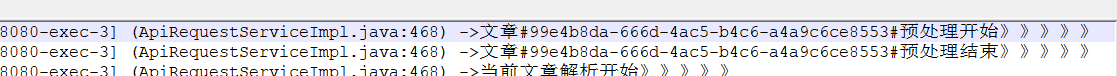
|  |
| --- |
|  |

如上图，“2018-10-24 09:45:41 [com.yuzhi.service.impl.ApiRequestServiceImpl]-[INFO] [http-bio-8080-exec-3] (ApiRequestServiceImpl.java:273) ->”此部分内容为固定格式，包含时间日期、日志打印源（代码中哪一行打印的，上面示例中的“ApiRequestServiceImpl.java:273”表示：ApiRequestServiceImpl.java类中的第273行代码发出的打印信息），日志的正文信息以“->”为起始标志。

### 预处理模块日志

这是预处理模块的日志信息打印。

开始和结束的打印格式分别为“文章#articleID#预处理开始》》》》》”和“文章#articleID#预处理结束》》》》》”。两行中间正常情况下不会打印其他信息。



### 解析模块日志

接下来，是解析模块的日志信息打印。

|  |
| --- |
|  |

如上图，任何一篇文章的解析日志信息，总是以“当前文章解析开始》》》》》”开始打印，并以“当前文章解析结束》》》》》”的字样结束打印。以下为详解：

1. 开始解析后，前几行日志打印的是单句解析日志。这些日志用于HowNet死机后，去日志中寻找因为哪句话导致的死机（即最后打印完整的一句话，紧挨着的下一句话），示例如下：

|  |
| --- |
| **{**http://tool.oschina.net/tools/json_format/Expanded.gif  #当前解析句子实体  #句子内容  **"currentSentence"**: "公告称,深交所于4月12日向公司发出了重组问询函,公司就问询函提及的问题进行了反馈。"**,**  #句子分词情况  **"nodes"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif  **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"aspect"**: ""**,**  **"dP"**: "4"**,**  **"deepLog"**: "InfoSource"**,**  **"deepSon"**: ""**,**  **"eS"**: ""**,**  **"fH"**: "004"**,**  **"gramInfo"**: ""**,**  **"key"**: ""**,**  **"key\_1"**: ""**,**  **"key\_2"**: ""**,**  **"key\_IR"**: ""**,**  **"nER"**: ""**,**  **"pOS"**: "noun"**,**  **"son"**: ""**,**  **"tempInfo"**: "abso"**,**  **"unitID"**: "44299"**,**  **"yS"**: ""**,**  **"id"**: "1"**,**  **"expression"**: "公告"**,**  **"log"**: "agent"**,**  **"pos\_begin"**: "0"**,** #当前词起始位置  **"pos\_end"**: "2"#当前词终止位置  **},**  **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"aspect"**: "+subj"**,**  **"dP"**: ""**,**  **"deepLog"**: "InfoMeans"**,**  **"deepSon"**: ""**,**  **"eS"**: ""**,**  **"fH"**: ""**,**  **"gramInfo"**: "MustObj,thatobj"**,**  **"key"**: ""**,**  **"key\_1"**: ""**,**  **"key\_2"**: ""**,**  **"key\_IR"**: ""**,**  **"nER"**: ""**,**  **"pOS"**: "verb"**,**  **"son"**: "001"**,**  **"tempInfo"**: "abso,UB"**,**  **"unitID"**: "186943"**,**  **"yS"**: ""**,**  **"id"**: "4"**,**  **"expression"**: "称"**,**  **"log"**: ""**,**  **"pos\_begin"**: "2"**,**  **"pos\_end"**: "3"  **},**  **。。。（此处省略相似内容）**  **],**  #句子所走的规则列表  **"rulesEntities"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif  #一对“{}”表示一条规则  **{**http://tool.oschina.net/tools/json_format/Expanded.gif  #分词ID，即哪个词走了下面这条规则，这里用ID标识，虽然是字符串，但其实是整数  **"cN"**: "12"**,**  #规则内容  **"rule"**: "~~ZhaoShangYH\_Preprocess 0024 CN[\*def=={time|时间:TimeSect={month|月},modifier={specific|特定}}];R1[\*def=={cardinal|基数}];R2[\*def>={time|时间:TimeSect={day|日}}]$@bind(CN,R2)CN[\*NewDef=={time|时间},\*log==`time`]. //2013年 "  **},**  **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"cN"**: "1"**,**  **"rule"**: "DefineSegment 0000 CN[\*PartNum==`1`]$. // 没有歧义"  **},**  **。。。（此处省略相似内容）**  **]**  **}** |

1. 在每个句子的解析日志打印结束之后，在出现“当前文章解析结束》》》》》”字样之前，将会打印整篇文章全部的解析日志信息，这些信息可以直接提取并分析整篇文章所走规则。示例如下：

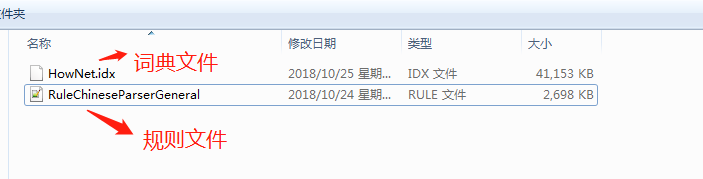
|  |
| --- |
| **{**http://tool.oschina.net/tools/json_format/Expanded.gif  #当前文章的ID  **"articleID"**: "2a8669c1-bb8d-430b-98ea-adac8ff115fa"**,**  #文章分句及解析列表  **"currentSentenceEntities"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif  #每个分句的日志（以下节点内容）  **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"currentSentence"**: "为此,证监会决定对人保财险宝应支公司罚款20万元。"**,**  **"nodes"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif  **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"aspect"**: ""**,**  **"dP"**: ""**,**  **"deepLog"**: ""**,**  **"deepSon"**: ""**,**  **"eS"**: ""**,**  **"fH"**: "005"**,**  **"gramInfo"**: ""**,**  **"key"**: ""**,**  **"key\_1"**: ""**,**  **"key\_2"**: ""**,**  **"key\_IR"**: ""**,**  **"nER"**: ""**,**  **"pOS"**: "prep"**,**  **"son"**: ""**,**  **"tempInfo"**: "abso"**,**  **"unitID"**: "168156"**,**  **"yS"**: ""**,**  **"id"**: "1"**,**  **"expression"**: "为此"**,**  **"log"**: "",  **"pos\_begin"**: "2"**,**  **"pos\_end"**: "3"  **},**  **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"aspect"**: ""**,**  **"dP"**: ""**,**  **"deepLog"**: ""**,**  **"deepSon"**: ""**,**  **"eS"**: ""**,**  **"fH"**: ""**,**  **"gramInfo"**: "comma"**,**  **"key"**: ""**,**  **"key\_1"**: ""**,**  **"key\_2"**: ""**,**  **"key\_IR"**: ""**,**  **"nER"**: ""**,**  **"pOS"**: "pun"**,**  **"son"**: ""**,**  **"tempInfo"**: "abso,UB,quasihead"**,**  **"unitID"**: "10409"**,**  **"yS"**: ""**,**  **"id"**: "4"**,**  **"expression"**: ","**,**  **"log"**: "",  **"pos\_begin"**: "2"**,**  **"pos\_end"**: "3"  **},**  **。。。（此处省略相似内容）**  **],**  **"rulesEntities"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif  **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"cN"**: "26"**,**  **"rule"**: "~~ZhaoShangYH\_Preprocess 0010 CN[\*pos==`num`];R1[\*pos==`num`]$@bind(CN,R1).//合并数字表示 "  **},**  **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"cN"**: "1"**,**  **"rule"**: "DefineSegment 0000 CN[\*PartNum==`1`]$. // 没有歧义"  **},**  **。。。（此处省略相似内容）**  **]**  **},**  **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"currentSentence"**: "港交所最新资料显示,华(00628)主要股东ChinWanYi场内减持公司好仓195.2万股,成交均价1.803港币。"**,**  **"nodes"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif  **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"aspect"**: ""**,**  **"dP"**: ""**,**  **"deepLog"**: ""**,**  **"deepSon"**: ""**,**  **"eS"**: ""**,**  **"fH"**: "008"**,**  **"gramInfo"**: "proper"**,**  **"key"**: ""**,**  **"key\_1"**: ""**,**  **"key\_2"**: ""**,**  **"key\_IR"**: "CorpName"**,**  **"nER"**: ""**,**  **"pOS"**: "noun"**,**  **"son"**: ""**,**  **"tempInfo"**: "abso,UB"**,**  **"unitID"**: "63169"**,**  **"yS"**: ""**,**  **"id"**: "1"**,**  **"expression"**: "港交所"**,**  **"log"**: "restrictive",  **"pos\_begin"**: "2"**,**  **"pos\_end"**: "3"  **},**  **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"aspect"**: ""**,**  **"dP"**: ""**,**  **"deepLog"**: ""**,**  **"deepSon"**: ""**,**  **"eS"**: ""**,**  **"fH"**: "008"**,**  **"gramInfo"**: ""**,**  **"key"**: ""**,**  **"key\_1"**: ""**,**  **"key\_2"**: ""**,**  **"key\_IR"**: ""**,**  **"nER"**: ""**,**  **"pOS"**: "adj"**,**  **"son"**: ""**,**  **"tempInfo"**: "abso"**,**  **"unitID"**: "216440"**,**  **"yS"**: ""**,**  **"id"**: "5"**,**  **"expression"**: "最新"**,**  **"log"**: "modifier",  **"pos\_begin"**: "2"**,**  **"pos\_end"**: "3"  **},**  **。。。（此处省略相似内容）**  **],**  **"rulesEntities"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif  **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"cN"**: "40"**,**  **"rule"**: "~~ZhaoShangYH\_Preprocess 0010 CN[\*pos==`num`];R1[\*expr==`.`];R2[\*pos==`num`];R3[\*pos==`num`]$@bind(CN,R3).//合并数字表示 "  **},**  **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"cN"**: "53"**,**  **"rule"**: "~~ZhaoShangYH\_Preprocess 0010 CN[\*pos==`num`];R1[\*expr==`.`];R2[\*pos==`num`]$@bind(CN,R2).//合并数字表示 "  **},**  **。。。（此处省略相似内容）**  **]**  **}**  **]**  **}** |

# 项目更新及维护

项目上线之后，后期的规则增加与修改，HowNet词典加词等操作是由本地的语言分析师，在本地完成加词、改规则等操作之后，将词典及规则文件上传至服务器，然后在每个HowNet动态库文件夹下找到对应的文件，复制替换一下即可。但由于词典文件过大，上传生产环境流程繁琐，最终决定在线上修改词典及规则文件，由于这部分操作用文字描述不清楚，因此将在后期招行员工培训中体现。这里我们只讲一下如何替换，详细操作如下：

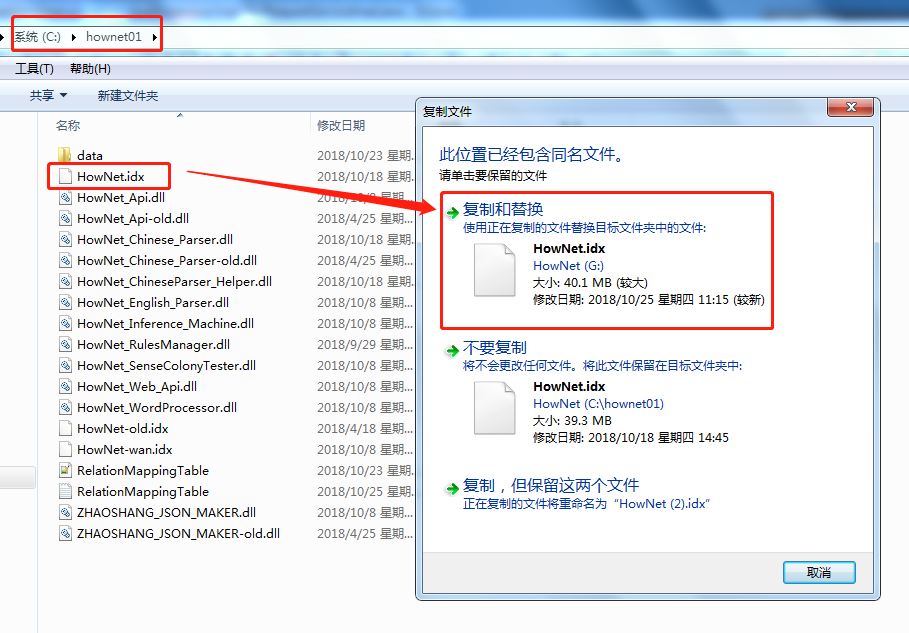
## 替换操作

语言分析师将改好的规则文件发给运维人员，运维人员完成以下替换操作。请务必在替换时，将Java应用暂时关闭！！！目前只需要两个文件，一个是idx格式的词典，一个是RULE格式的规则文件。还有一个“RelationMappingTable.txt”

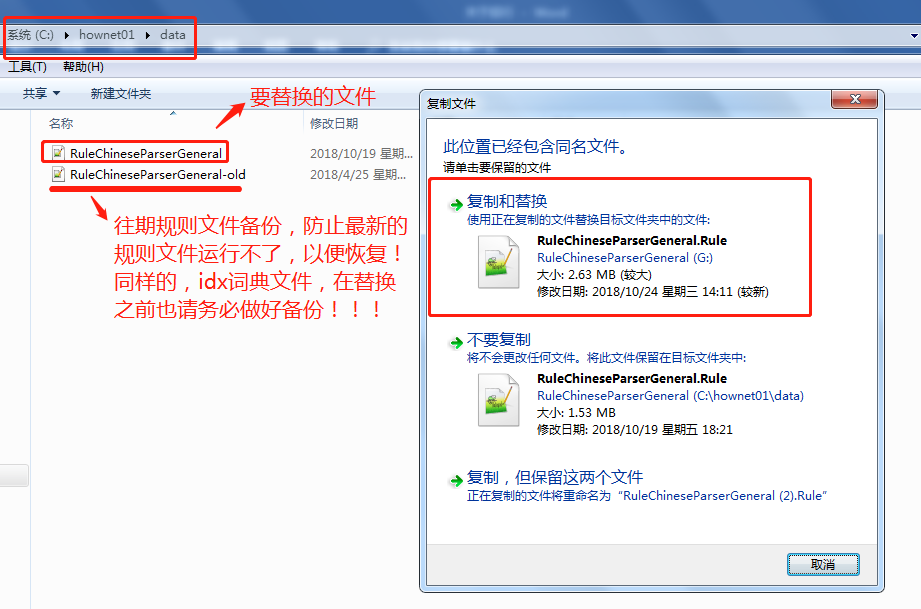


## 替换词典和规则文件

将上述两个文件分别复制到服务器上HowNet动态库的文件夹下。如下图，还以hownet01这套动态库为例。idx文件放入“C：\hownet01\”目录下：



RULE格式文件则放在“C:\hownet01\data\”目录下：



## 替换完成后重启

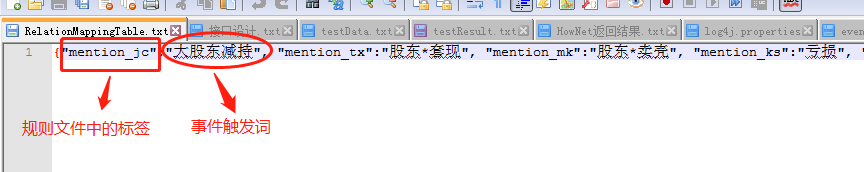
替换完成后，来到对应的Java应用（Tomcat）目录下，进入“\tomcat\bin\”目录下，双击startup.bat程序文件进行重启（详情参考【项目启动与关闭】）

## “RelationMappingTalbe.txt”文件

### 文件内容介绍

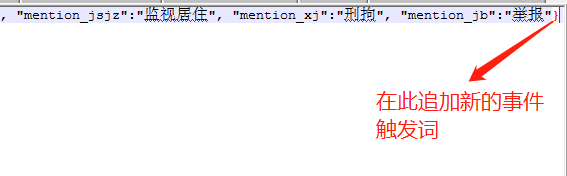
该文件位于“C:\hownet01\”目录下，与idx词典同目录。

文件中存放的是金融事件的触发词对应关系表，格式为“键值对”的形式，中间逗号分隔。键部分的英文表示，触发词在规则文件中的标签；值部分的中文则为触发词。该表主要用于Java应用中根据解析结果确定事件类型。

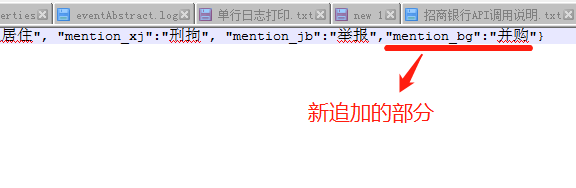


### 文件更新

一般不用更新，若有新加的事件，才需要更新此文件。只需将新加词，按照文件中的格式（键值对），追加在末尾（括号之前）即可。文件可以换行，但是务必保持格式正确！

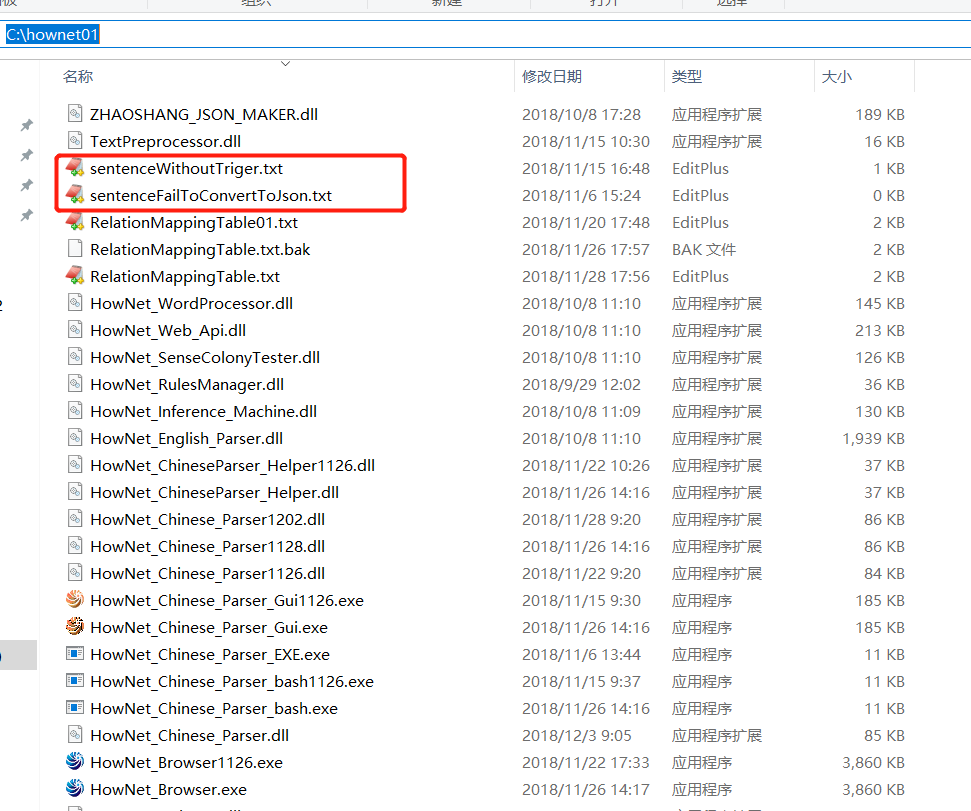


我们以“mention\_bg”:“并购”为例，直接在“}”前追加即可，如下图：



## “sentenceFailToConvertToJson.txt”和“sentenceWithoutTriger.txt”文件

此两个文件位于hownet动态库的根目录下(即与HowNet\_Browser.exe、HowNet\_Chinese\_Parser.dll等，处于同目录下)，但是在默认情况下，该目录下是没有这两个文件的。如下图：



### 文件介绍及生成条件

sentenceFailToConvertToJson.txt文件，是用于记录在文章解析过程中，解析结果错乱导致Java应用提取结果失败的句子。例如，“\语知科技\预计将于明年三月份’并购’未来科技有限公司。”,该句话中的“\”会导致HowNet解析器返回的Json格式的解析结果错乱，从而导致Java应用提取结果失败。这时，该句话就会被记录在sentenceFailToConvertToJson.txt文件中。

sentenceWithoutTriger.txt文件，是用于记录在文章解析过程中，抽取结果中没有事件触发词，却有事件主、客体等内容的句子。

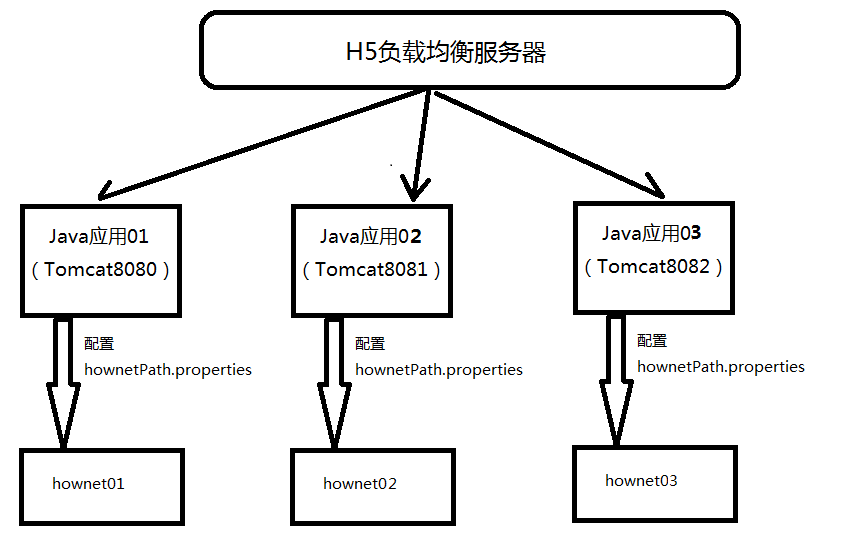
### 文件的维护与删除

由于此两个文件记录的信息量并不大，经商讨，这些文件默认不会自动删除，需要运维人员定期（几个月甚至半年）去查看一次，若有上述两个文件，则将此文件备份至本地后将线上的清空或删除。备份于本地的则反馈给开发人员进行相应修改。

# 应用扩展

## Java应用与动态库的对应关系

目前，HowNet支持横向扩展。及可以复制出多份HowNet，正如测试服务器上的8080、8081所对应的hownet01、hownet02。他们的对应关系如下：



通过配置java应用的resource目录下面的hownetPath.properties文件，即可实现Java应用于HowNet动态库之间的关联。

## 手动扩展一套HowNet

背景：

已知：当前服务器下已经有一套正在运行的Java应用和hownet动态库，他们的位置分别为“F:\Tomcats\Tomcat8080\webapps\ROOT”和“C:\hownet01”。并且已经通过配置ROOT\WEB-INF\classes\resource\目录下的hownetPath.properties文件，实现了连接。如下图1和图2：

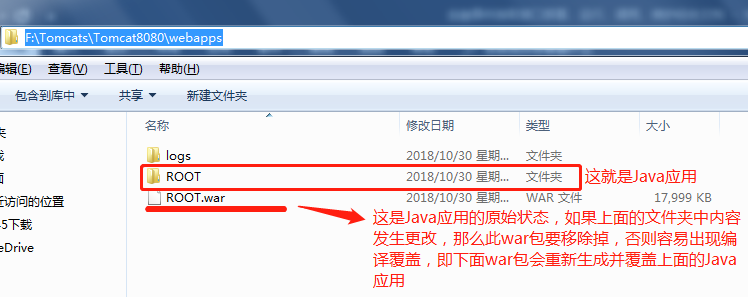


图1

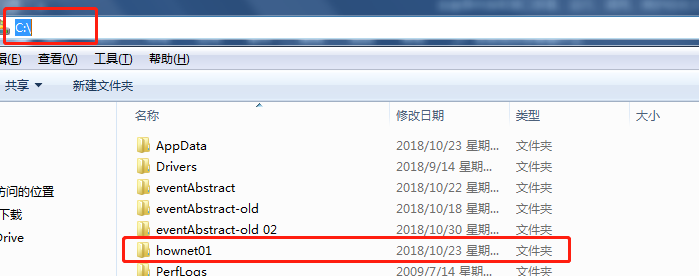
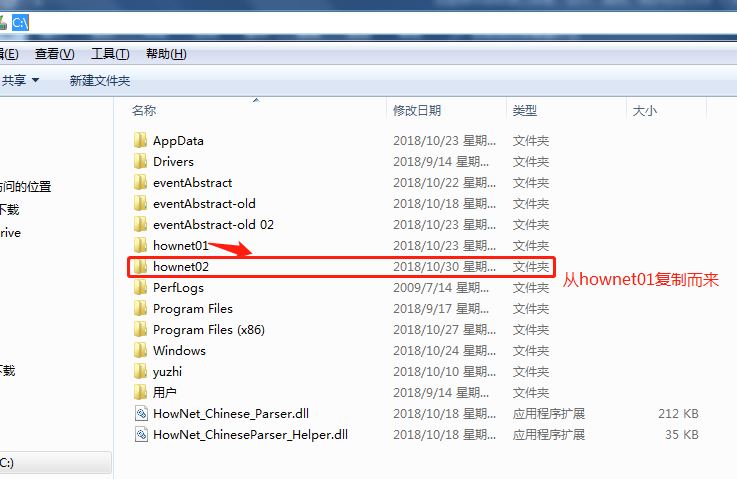


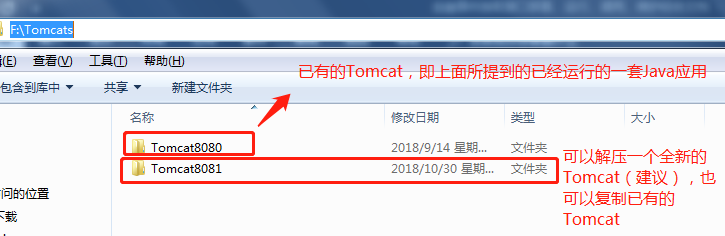
图2

那么现在我们就手动扩展一套HowNet！！！敲黑板！！！运维必考题！！！！

1. 第一步，复制一份HowNet动态库文件出来，我们暂且命名为hownet02，并放在自己确定好的目录下，这里就还放置在C盘根目录下。（不管放在哪个目录下，只要自己记得就好，需要往hownetPat.properties文件里面写。）



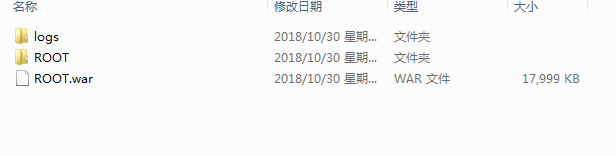
1. 第二步，配置一个全新的Tomcat，命名为“Tomcat+端口号”(方便区分)。并通过修改Tomcat\conf\文件夹下的server.xml文件来配置不同于已有的端口号，至于如何更改Tomcat端口号，请自行百度，此处就不再介绍了。这里将新的Tomcat命名为“Tomcat8081”，如下图：



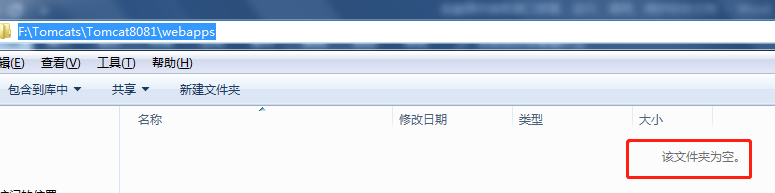
1. 第三步，配置完了新的Tomcat，接下来进入Tomcat8081\webapps\目录：
2. 若在上一步中，新解压了一个全新的Tomcat，那么该目录下应该是这个样子：



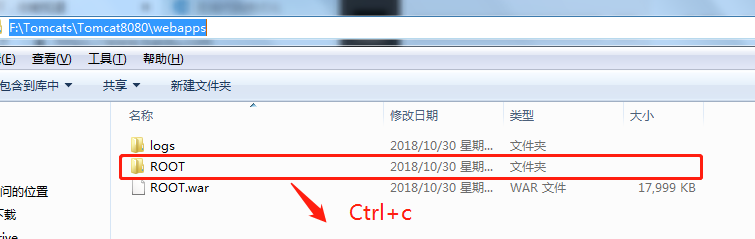
1. 若在上一步中，直接从运行的Tomcat复制出来的，那么该目录下应该是这个样子：



1. 为了保证万无一失，以上两种情况，我们均直接清空webapps文件夹，并重新配置。



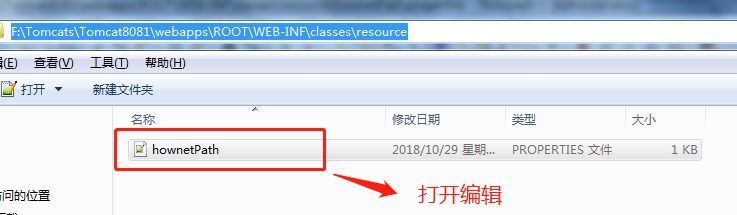
1. 接下来，将开发人员提供的可用的Java应用，ROOT文件夹或ROOT.war（二选一即可，如果有已经在运行的Java应用，之后可以直接从正在运行的Tomcat里面拷贝），这里我们选择拷贝。拷贝步骤如下：
2. 找到已有的或者正在运行的Tomcat，这里我们刚才提到，我们是有一套正在运行的应用的，即Tomcat8080。进入Tomcat8080\webapps\目录，复制ROOT：



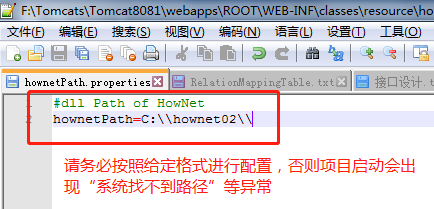
1. 返回Tomcat8081\webapps\目录下，粘贴：



1. 然后进入到ROOT\WEB-INF\classes\resource\目录下，编辑hownetPath.properties文件：



1. 将“=”后面的值改为你要对应的HowNet动态库，我们这里要对应的是hownet02动态库。刚才我们把hownet02动态库文件夹，放在了C盘的根目录下，那么这里“=”后面就应该写入“C:\\hownet02\\”。再强调一遍：请务必按照给定格式配置路径，否则项目启动时会报“系统找不到指定的路径”异常。如图3：



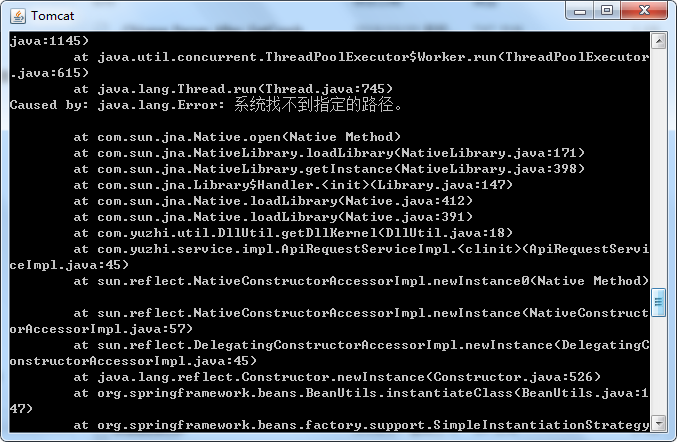


图3

1. 编辑完成后保存。自此扩展完成！接下来就是启动Tomcat，详情参考【项目启动与关闭】一项。

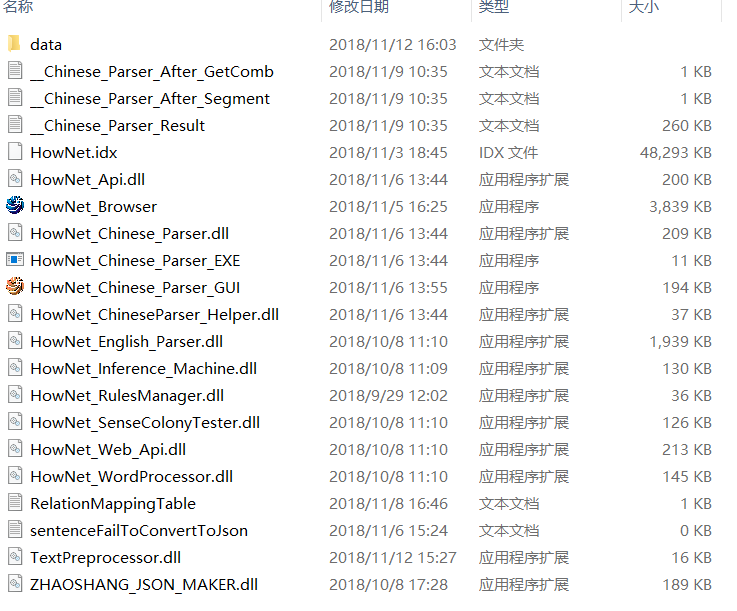
# 关于Java调用动态库的演示

## 准备条件

### 导入调用动态库的Jar包：JNative.jar和jna.jar



### C++工程师那边所提供的动态库一套（即上边频繁提到的HowNet动态库）



### C++工程师所提供的接口文档

这里需要为你提供接口（动态库）的工程师给你一个接口列表。这里我以金融事件抽取接口为例：

动态库初始化接口 🡺🡺🡺 const bool initHowNet(char\* ApPath);

金融事件抽取主接口 🡺🡺🡺 const bool parseText(char\* ApPath);

金融事件抽取结果获取 🡺🡺🡺 const char\* getJsonRet();

这里需要对C++做一些简单的了解，或者直接询问他以上方法的每个关键字什么意思。

## Java代码编写

接下来进入正题，编写调用动态库的编码。

### 创建一个调用动态库的工具类，详细代码如下

|  |
| --- |
| **package** com.yuzhi.util;  **import** com.sun.jna.Native;  **import** com.sun.jna.win32.StdCallLibrary;  **import** com.yuzhi.entity.ParamConstant;  /\*\*  \* 这是一个调用动态库的工具类  \* **@author** Hunter  \*  \*/  **public** **class** DllUtil {  //编写一个interface接口，名字随便起  //让此接口继承com.sun.jna.win32.StdCallLibrary接口  **public** **interface** EventAbstract **extends** StdCallLibrary {  //将c++提供的所有接口，用Java代码实现一下    //HowNet初始化接口，对应c++接口列表里的 const bool initHowNet(char\* ApPath);  **public** **boolean** initHowNet(String Apath);  //解析接口，对应c++接口列表里的 const bool parseText(char\* ApPath);  **public** **boolean** parseText(String input);  //获取解析结果接口，对应c++接口列表里的 const char\* getJsonRet();  **public** String getJsonRet();  }    //声明一个接口变量  **private** **static** EventAbstract *dllKernel*;    //创建接口实例的方法，相当于单例模式下的 getInstance()方法  **public** **static** EventAbstract getDllKernel() {  //加载本地dll接口  **if** (*dllKernel*==**null**) {//如果变量不为空，说明已经加载，则直接返回  //com.sun.jna.Native类有一个loadLibrary()方法，通过反射的方式来加载动态库，此处的“ParamConstant.INITIAL\_PATH”就是我们之前部署的时候所配置的HowNet路径，即动态库的路径  *dllKernel* = (EventAbstract) Native.*loadLibrary*(ParamConstant.*INITIAL\_PATH*+"HowNet\_ChineseParser\_Helper",  EventAbstract.**class**);  }  //返回实例  **return** *dllKernel*;  }    } |

### 编写一个测试类

|  |
| --- |
| package com.yuzhi.test;  import com.yuzhi.entity.ParamConstant;  import com.yuzhi.util.DllUtil;  import com.yuzhi.util.DllUtil.EventAbstract;  /\*\*  \* 动态库调用测试类  \* @author Hunter  \*  \*/  public class DllUtilTest {  //定义HowNet时间抽取接口  private static EventAbstract eventAbstract;  public static void main(String[] args) {  //调用工具类中加载动态库的方法  eventAbstract = DllUtil.getDllKernel();  //加载完之后初始化，此处的“ParamConstant.INITIAL\_PATH”就是我们之前部署的时候所配置的HowNet路径，即动态库的路径  boolean init = eventAbstract.initHowNet(ParamConstant.INITIAL\_PATH);  //初始化成功之后即可调用  if(init){  //解析  boolean parseText = eventAbstract.parseText("百度公司今年利润出现大幅下滑，为此，深交所向公司发出了问询函。");  //如果返回true,说明解析成功，获取解析结果  if(parseText){  String jsonRet = eventAbstract.getJsonRet();  //打印结果到控制台  System.out.println("解析结果》》》"+jsonRet);  }  }else{  System.out.println("初始化失败！！");  }  }  } |

控制台打印结果

|  |
| --- |
| 解析结果》》》{"sentence":[此处省略。。。]} |

自此，调用动态库成功。