Solr4UIMA

Solr 4 UIMA Tutorial

⚠️ Solr4.1

Solr UIMA contrib enables enhancing of Solr documents using the Unstructured Information Management Architecture (UIMA). UIMA lets you define custom pipelines of Analysis Engines which incrementally add metadata to the document via annotations. In this tutorial we first install the Eclipse UIMA toolkit, create a custom UIMA Annotator, test the Annotator using the UIMA CAS Visual debugger, create a JAR file for use with Solr 4 and setup Solr to use the Annotator.

Setup UIMA toolkit in Eclipse

More details can be found here:
http://uima.apache.org/downloads/releaseDocs/2.2.2-incubating/docs/html/overview_and_setup/overview_and_setup.html#ugr.ovv.eclipse_setup

1. Install Eclipse Modelling Framwork (EMF) from the Eclipse update site 2. Install Apache UIMA eclipse tooling from http://www.apache.org/dist /uima/eclipse-update-site 3. Install Apache UIMA from http://uima.apache.org/downloads.cgi 4. Open uimaj-examples (this will enable Run As functionality for the e.g. the JCas debugger)

- File - Import - General / Existing Projects into workspace - Select apache-uima folder
- This will automatically add uimaj-examples to the workspace

Create your own UIMA Annotator

More details can be found here:
http://uima.apache.org/doc-uima-annotator.html

1. Create a new Java project in your Eclipse workspace called RoomNumberAnnotator. To do this select "File -> New -> Java Project" and use RoomNumberAnnotator as the project name. Also, in the Project Layout section, make sure the button to "Create separate folders for sources and class files" is checked. 2. Add the UIMA nature to the project by right-clicking on the "RoomNumberAnnotator" project and choose "Add UIMA Nature". Confirm the upcoming dialogues with "Yes" to add the UIMA nature, pressing "OK", next, to confirm the status message dialog. This will create a default directory layout of folders useful for annotator component development. 3. Project - Right click - Add UIMA nature 4. Configure build path (create Variable UIMA_HOME):

- Right-click to the RoomNumberAnnotator project and choose Build Path -> Configure Build Path. * Click the "Add Variable..." button, and select the "UIMA_HOME" variable. Add new variable now, using the Configure Variables, setting it to the home directory where you have UIMA installed. * Click the "Extend..." button and chose the uima-core.jar in "lib" directory. You could add other jars from the UIMA lib, but the uima-core.jar is the only one needed for this project. * Finalize all dialogues with the "OK" button. 5. Define Annotator type * Right-click on the "desc" folder of your project and choose "New -> Other" * Select "Analysis Engine Descriptor" from the "UIMA" folder and press "Next" * Enter "RoomNumberAnnotatorDescriptor.xml" as file name, and press "Finish" 6. Add new type (RoomNumber) to the RoomNumberAnnotatorDescriptor .xml * Open the descriptor using the UIMA Component Descriptor Editor (CDE) by right-click to the "RoomNumberAnnotatorDescriptor.xml" file and choose "Open With -> Component Descriptor Editor" * Select the "TypeSystem" tab at the bottom to show the type system definition page. * Press the "Add Type" button to add the new type. Use "org.apache.uima.tutorial.RoomNumber" as type name and finish with "OK". The supertype "uima.tcas.Annotation" is correct 7. Add new feature (building) to type RoomNumber * Select the "org.apache.uima.tutorial.RoomNumber" type by clicking it. * Click the "Add..." button to add a feature to the type and specify "building" as feature name and "uima.cas.String" as range type. This means that the "building" feature is a String based feature. * Finish the dialog by clicking "OK". * Save the descriptor file 8. Automatically create Java classes: * Open the descriptor file in the Component Descriptor Editor and select the "Type System" tab. * Press the "JCasGen" button that will trigger the Java class generation. The generated classes will be added to the "src" folder of your project in a separate package. 9. Write Java code for the Annotator * Right-click on the "src" folder and select "New -> Class" * Package: org.apache.uima.tutorial.ex1 Name: RoomNumberAnnotator Superclass: org.apache.uima.analysis_component.JCasAnnotator_ImplBase 10. Test the Annotator: * Run - Run as - Run configurations - Java Application - UIMA CAS Visual debugger * Select the "User Entries" in the classpath tab and press the "Add Projects..." button * Mark the "RoomNumberAnnotator" project in the upcoming dialog and finish with "OK" * Run the CAS Visual Debugger (CVD) by selecting "Run" * Choose "Run -> Load AE" and select the RoomNumberAnnotatorDescriptor.xml file in the desc folder of your Eclipse project * Copy and past the text below for testing to the text section of the CVD

```
April 7, 2004 Distillery Lunch Seminar
UIMA and its Metadata
12:00PM-1:00PM in HAW GN-K35

April 16, 2004 KM & I Department Tea
Title: An Eclipse-based TAE Configurator Tool
3:00PM-4:30PM in HAW GN-K35

May 11, 2004 UIMA Tutorial
9:00AM-5:00PM in YKT 20-001
```

- To run the annotator on the specified text, choose "Run -> RunRoomNumberAnnotatorDescriptor" 11. Create JAR file from Project: Right-click on the Project - Export - Java - JAR file 12. Copy the JAR file to SOLR_HOME/example/solr/collection1/lib

SolrUIMA UpdateRequestProcessor

The SolrUIMA UpdateRequestProcessor is a custom UpdateRequestProcessor that takes document(s) being indexed, sends them to a UIMA pipeline and then returns the document(s) enriched with the specified metadata.

Installation

1. Download latest Solr 4.x release http://www.apache.org/dyn/closer.cgi/lucene/solr/ 2. Copy the following files from the Solr release to the Solr document location you are using (in this case solr/example/solr/collection1)

```
mkdir solr/example/solr/collection1/lib
cp solr/dist/solr-uima*.jar solr/example/solr/collection1/lib
cp solr/contrib/uima/lib/*.jar solr/example/solr/collection1/lib/
cp solr/contrib/uima/lucene-libs/lucene-analyzers-uima*.jar solr/example/solr/collection1/lib/
```

3. Modify your Solr instance config files as described in the solr/contrib/solr-uima/README.txt 4. Run your Solr instance and enjoy UIMA enriching documents being indexed

Configuration

All the SolrUIMA configuration is placed inside a <uimaConfig> element inside the solrconfig.xml.

```
<updateRequestProcessorChain name="uima" default="true">
     <processor class="org.apache.solr.uima.processor.UIMAUpdateRequestProcessorFactory">
       <lst name="uimaConfig">
         <lst name="runtimeParameters">
         </lst>
         <str name="analysisEngine">/descriptors/AggregateEngineDescriptor.xml</str>
         <bool name="ignoreErrors">false</bool>
         <lst name="analyzeFields">
           <bool name="merge">false</bool>
           <arr name="fields">
             <str>content</str>
           </arr>
         </lst>
         <lst name="fieldMappings">
           <lst name="type">
             <str name="name">com.example.uima.Any1</str>
             <lst name="mapping">
               <str name="feature">Any1</str>
               <str name="field">Any2</str>
             </lst>
           </lst>
           <lst name="type">
             <str name="name">com.example.uima.Any2</str>
             <lst name="mapping">
               <str name="feature">Any2</str>
               <str name="field">Any2</str>
             </lst>
           </lst>
           <lst name="type">
             <str name="name">com.example.uima.Any3</str>
             <lst name="mapping">
               <str name="feature">Any3</str>
               <str name="field">Any3</str>
             </lst>
           </lst>
         </lst>
       </lst>
     </processor>
     <processor class="solr.LogUpdateProcessorFactory" />
     <processor class="solr.RunUpdateProcessorFactory" />
</updateRequestProcessorChain>
```

The analysisEngine element holds the classpath to the UIMA Analysis Engine descriptor that describes which analysis block should be executed. The analysis engine referenced can be primitive or aggregate.

The analyzeFields element lists the name of fields (comma separated) which will be analyzed by the UIMA pipeline. If the attribute merge is false the field specified will be analyzed separately while if merge is true the listed fields contents will be merged and analyzed only once.

see SOLR-2129

UIMA components used

UIMA supports the use of existing analysis engines (see here and here) as long as the creation of custom components.

The current contrib/uima module uses a predefined set of components :

1. WhitespaceTokenizer 2. HMMTagger 3. OpenCalaisAnnotator 4. AlchemyAPIAnnotator

These components are arranged in a pipeline inside the OverridingParamsExtServicesAE Analysis Engine descriptor. As you can see looking at the descriptor fragment;

```
        <node>AggregateSentenceAE</node>
        <node>OpenCalaisAnnotator</node>
        <node>TextKeywordExtractionAEDescriptor</node>
        <node>TextLanguageDetectionAEDescriptor</node>
        <node>TextCategorizationAEDescriptor</node>
        <node>TextConceptTaggingAEDescriptor</node>
        <node>TextRankedEntityExtractionAEDescriptor</node>
```

the first node represent an aggregate Analysis Engine which includes the Whitespace Tokenizer and HMM Tagger (recognizing sentences), the second node uses the Open Calais Annotator to extracte named entities, the following nodes use different Alchemy API Annotator services to detect keywords, language, document category, discovered concepts and named entities.

Using other UIMA components

To use different UIMA components inside the contrib/uima module you need to:

1. import the component jar 2. use the new component Analysis Engine descriptor inside config/uimaConfig/analysisEngine element of solrconfig. xml 3. adjust Analysis Engine configuration (optional) 4. change the types and features' mapping inside config/uimaConfig/fieldMapping element of solrconfig.xml 5. deploy new apache-solr-uima.jar and component inside one of the lib directories

Import the component jar

If you're using Ant you only need put the component jar inside the solr/contrib/uima/lib directory.

If you're using Maven you need to declare the component you want to use inside the <dependencies> element in the generated pom.xml.

For example if you want to use UIMA Dictionary Annotator 2.3.1-SNAPSHOT you can either get it from snapshot repo and paste it in solr/contrib/uima/lib and run 'ant clean dist' or paste the following in the generated pom.xml (as child of the <dependencies> tag) and run 'mvn clean package'.

```
<dependency>
  <groupId>org.apache.uima</groupId>
  <artifactId>DictionaryAnnotator</artifactId>
  <version>2.3.1-SNAPSHOT</version>
</dependency>
```

Use a different UIMA descriptor

Change the descriptor to be used by this module inside config/uimaConfig/analysisEngine of the solrconfig.xml of your Solr instance.

One can use the default one bundled inside the component or create a new one.

For example to use one of the default Dictionary Annotator Analysis Engine descriptors use the following (which runs Whitespace Tokenizer and then Dictionary Annotator):

```
  <config>
    ...
    <uimaConfig>
      ...
      <analysisEngine>/AggregateAE.xml</analysisEngine>
      ...
    </uimaConfig>
    ...
  </config>
```

Adjust AE configuration (optional)

Sometimes Analysis Engines require custom parameters to be set inside their descriptor or custom resources to be imported. The easiest way to do so is to get a copy of such a descriptor, modify parameters/resources as needed and put them inside a directory which gets included in the final jar (i.e.: solr /contrib/uima/src/main/resources/org/apache/uima)

Change the types and features' mapping

Inside the solrconfig.xml go to config/uimaConfig/fieldMapping element and change <type> element according to the annotations extracted by the used component.

For example if you're using the Dictionary Annotator and you want to put the dictionary entry annotations found inside a 'lemmas' field you should configure the fieldMapping element as following:

```
<config>
  ...
  <uimaConfig>
    ...
    <fieldMapping>
      <type name="org.apache.uima.DictionaryEntry">
        <map feature="coveredText" field="lemmas"/>
      </type>
    </fieldMapping>
    ...
  </uimaConfig>
  ...
</config>
```

Deploy new jars inside one of the lib directories

Run 'ant clean dist' (or 'mvn clean package') from the solr/contrib/uima path.

Get the generated apache-solr-uima*.jar from the build directory along with the used components' jars and paste both inside one of the <lib> directories defined inside the solrconfig.xml.

You can now restart the Solr-UIMA instance to test it.

Solrcas

This is a UIMA component, see SVN and documentation

For a deepest dive into UIMA please take a look at the documentation