First Look: InterSystems IRIS and UIMA

Version 2019.4
2020-01-28

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

First Look: InterSystems IRIS and UIMA
InterSystems IRIS Data Platform Version 2019.4 2020-01-28

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

First Look: InterSystems IRIS and UIMA

This First Look provides a quick introduction on how InterSystems IRIS® data platform implements and complements the Unstructured Information Management Architecture (UIMA). After a brief overview of UIMA and how InterSystems IRIS complements it, you have the opportunity to work through a basic hands-on exercise to see InterSystems IRIS in action.

To browse all of the First Looks, including those that can be performed on a free evaluation instance of InterSystems IRIS, see InterSystems First Looks.

1 About UIMA

UIMA is a standard that governs the analyzing of unstructured information such as text and video. With unstructured information, computers usually need a few steps to turn the information into actionable structured data. For example, a scanned document needs OCR before the text becomes machine-readable, and even then a computer does not work particularly well with natural language text until additional NLP strategies are applied. Because a process like this includes steps that are very different in nature, it's unlikely that a single tool can handle them all. More likely, this process includes individual modules, implemented by different parties using different technologies, that need to work together. In UIMA, these modules are called *analysis engines*.

Because UIMA-compliant analysis engines all comply with the same standards, they can be combined into a series of analyzers (a *UIMA analysis pipeline*), each doing what it does best. The source unstructured data is not altered as it makes its way through this UIMA analysis pipeline, but rather *annotations* are generated along the way. The UIMA standard ensures that the annotations from one analysis engine do not interfere with the annotations from a different analysis engine. For text, these annotations are based on character position within the text. The interoperability of UIMA allows you to combine analysis engines from different vendors and technologies into a single pipeline without writing any custom code, and because the analysis engines refer to character positions in the original source data, their annotations can be combined, compared, and reasoned with. The UIMA standard includes a framework implementation in Java that runs these analysis engines.

In addition to providing interoperability, UIMA provides a framework for scaling and deploying these analysis engines. This allows vendors to focus on developing analysis engines without worrying about scaling and deploying their solutions. The UIMA standard also provides the framework to invoke these analysis engines in a distributed architecture.

Each UIMA-compliant analysis engine must be accompanied by an XML descriptor file that contains basic identifying information such as name and vendor of the analysis engine. It also defines the annotation types that categorize the annotations that the analysis engine produces.

2 How InterSystems IRIS Complements UIMA

InterSystems IRIS complements UIMA in three ways. It:

- Introduces a functional index to create a UIMA analysis pipeline and to automatically feed the pipeline with new text when a record is inserted or updated in an InterSystems IRIS table.
- Stores annotations generated by the UIMA analysis pipeline in a clear, SQL-accessible Annotation Store on InterSystems IRIS.

- Ensures that InterSystems IRIS Natural Language Processing (NLP) complies with the UIMA standard and can be used as an analysis engine in a UIMA analysis pipeline.

2.1 Creating and Invoking a UIMA Analysis Pipeline

InterSystems IRIS uses a *functional index* to create a UIMA analysis pipeline using InterSystems IRIS concepts without needing to worry about implementing Java interfaces. A functional index is a feature of the InterSystems IRIS database that allows a function to be executed when a record is inserted or updated in a table. In this case, the functional index is defined on a table column that contains the unstructured data that you want analyzed by the UIMA analysis pipeline. Setting up the pipeline is as easy as adding the location of the analysis engines' descriptor files to the functional index definition.

Once the functional index is defined, InterSystems IRIS automatically feeds unstructured data into the UIMA analysis pipeline whenever new data is inserted or updated in the indexed table column. For example, if the functional index is defined on a column that contains reports, then a new report would be analyzed as soon as it is added to the table. Without this special functionality in InterSystems IRIS, you would need to send unstructured data through the pipeline programmatically in Java every time you wanted to analyze the data.

2.2 Annotation Store

By default, the results of a UIMA analysis pipeline are captured in verbose and cumbersome XML files. Because the UIMA standard does not provide a more sophisticated method of storing the annotations, InterSystems IRIS extends an UIMA analysis pipeline by using flexible, SQL-based storage to put the annotations in uniform, persistent tables for later retrieval. This storage system is called the *Annotation Store*.

This Annotation Store is created automatically the first time you compile the class that contains the functional index you defined to create the UIMA analysis pipeline. It is linked directly to the column in the original table that contains the unstructured data.

Architecturally, the Annotation Store is produced by adding a special analysis engine as the last component of a UIMA analysis pipeline. This happens automatically when you add a UIMA functional index to an InterSystems IRIS class. It's also possible for a UIMA analysis pipeline developed outside of InterSystems IRIS to add this special analysis engine to the end of the pipeline to create an Annotation Store. Such an implementation is beyond the scope of this First Look.

You can also customize the Annotation Store using an XData block in the class that contains the functional index. For example, you can define additional columns and indices per table. You can also filter annotation types to keep them out of the Annotation Store.

2.3 InterSystems IRIS NLP

InterSystems IRIS Natural Language Processing (NLP) is embedded into InterSystems IRIS® data platform and allows you to perform text analysis on unstructured text without any upfront knowledge of the subject matter. It does this by applying language-specific rules that identify semantic entities. Because these rules are specific to the language, not the content, InterSystems IRIS NLP can provide insight into the contents of texts without using a dictionary or ontology.

You can use InterSystems IRIS NLP as a UIMA analysis engine, generating UIMA annotations for NLP concepts and contexts. These annotations are fully compatible with UIMA annotations supplied by other UIMA analysis engines.

3 Tour of UIMA in InterSystems IRIS

Now that you have some basic information about UIMA, it's time to take a hands-on tour to see how it works in InterSystems IRIS. You will need to setup the environment before taking the tour.

3.1 Before You Begin

To get started, perform the following preliminary setup tasks:

1. Install the Java Runtime Environment.
2. Install InterSystems IRIS.
3. Create a new InterSystems IRIS namespace.
4. Add InterSystems libraries to your environment variables.
5. Start the Java Gateway.

3.1.1 Installing the Java Runtime Environment

InterSystems IRIS' implementation of a UIMA analysis pipeline requires that the Java Runtime Environment (JRE) be installed. It also requires an environment variable that points to the location of the JRE installation.

1. If you do not already have the JRE installed on your machine, download and install the latest version from Oracle®.
2. Create an environment variable called `JAVA_HOME` that points to the location of the JRE installation. For example, on Windows®, use the Control Panel to create the `JAVA_HOME` environment variable and define its path to the location of the JRE installation.

3.1.2 Installing InterSystems IRIS

To run the demo of the UIMA analysis pipeline, you'll need a running, licensed instance of InterSystems IRIS.

For instructions on how to install and license a development instance of InterSystems IRIS, see InterSystems IRIS Basics: Installation.

3.1.3 Creating a New Namespace

As part of the tour in this First Look, you will add a new class file to a namespace in InterSystems IRIS. To keep this sample data separate from the pre-defined namespaces, create a new namespace called SAMPLES to hold the code and data associated with this First Look. To create a new namespace:

1. Open the Management Portal in your browser using the URL for your instance, as described in InterSystems IRIS Connection Information in *InterSystems IRIS Basics: Connecting an IDE*.
2. Select **System Administration** > **Configuration** > **System Configuration** > **Namespaces**.
3. On the **Namespaces** page, select **Create New Namespace**.
4. On the **New Namespace** page, enter `SAMPLES` as the name for the new namespace.
5. Next to the **Select an existing database for Globals** drop-down menu, click **Create New Database**. This displays the **Database Wizard**.
6. On the first page of the **Database Wizard**, in the **Enter the name of your database** field, enter the name of the database you are creating, such as `Samplesdb`.
7. Enter a directory for the database, such as `C:\InterSystems\IRIS\mgr\Samplesdb`.
8. Click **Next**.
9. Click **Finish**.
10. Back on the **New Namespace** page, in the **Select an existing database for Routines** drop-down menu, select the database you just created.

11. Click **Save** near the top of the page and then click **Close** at the end of the resulting log.

3.1.4 Adding InterSystems Libraries to Your Path

Because the UIMA integration requires certain system libraries to be available when invoked through its Java framework, you must add the bin directory of the InterSystems IRIS installation to your path before running the Java Gateway (for example, C:\InterSystems\IRIS\bin). On Windows, add the bin directory to the PATH environment variable. For UNIX® and Linux platforms, add the bin directory to both the PATH and LD_LIBRARY_PATH environment variables.

3.1.5 Running the Java Gateway

The Java Gateway can instantiate an external Java object and manipulate it as if it were a native object within InterSystems IRIS. InterSystems IRIS' UIMA strategy uses the Java Gateway, which can be started from the command line. For example, on Windows:

1. Open the Run dialog.
2. Enter the following command:

   %JAVA_HOME%\bin\java -classpath "C:\InterSystems\IRIS\dev\java\lib\JDK18*;C:\InterSystems\IRIS\dev\java\lib\jackson*;C:\InterSystems\IRIS\dev\java\lib\uima*" com.intersystems.gateway.JavaGateway 5555

   Where:

   - `JAVA_HOME` is an environment variable that points to the location of the installation directory for the Java Runtime Environment (JRE).
   - `C:\InterSystems\IRIS` is the directory where you installed InterSystems IRIS.
   - `JDK18` corresponds to your version of the JRE.

   If you are running on UNIX®, remember that the syntax for `-classpath` uses a colon for the separator.

4 Taking the Tour of a UIMA Analysis Pipeline

Now that you've taken care of the preliminaries, you are ready to see a UIMA analysis pipeline in action. In this tour, you will:

- Add a class file that contains a UIMA functional index
- Compile the class that contains the functional index.
- Look at the Annotation Store's tables.
- Add unstructured data to the sample database.
- Browse the Annotation Store for new data generated by the analysis pipeline.

4.1 Adding Class File with a UIMA Functional Index

You add an analysis engine to the UIMA analysis pipeline by defining a functional index for the table that contains the unstructured text. In this tour, you are adding the InterSystems IRIS NLP analysis engine to the pipeline.

In this part of the tour, you are creating a new class file. You can create the class file in your favorite text editor if you do not have the Atelier IDE set up.

1. Create a new file in Atelier or a text editor.
2. Copy and paste the following into the class file:

```
Class Sample.MyData Extends %Persistent
{
Property MyText As %String;
Index MyIndex On (MyText) As %UIMA.Index(AEDESCRIPTOR =
"classpath:/com/intersystems/uima/annotator/iKnowEngine.xml");
}
```

   where:

   - `MyText` is the column of the Sample.MyData table that contains the unstructured text.
   - `MyIndex` is the UIMA functional index.
   - `iKnowEngine.xml` is the descriptor file for the InterSystems IRIS NLP analysis engine.

3. Save the file as `sample.cls`.

4.2 Compiling the Table Class

To automatically generate the Annotation Store, you simply compile the class that contains the functional index. If you created sample.cls in Atelier, simply compile the file.

If you made the changes in a text editor, use the InterSystems Terminal to load and compile the class.

Tip: When working with the InterSystems Terminal, you can paste the contents of your clipboard to the Terminal command prompt using `Shift+Insert`. This is useful for copying commands from this guide and pasting them in the Terminal to reduce errors.

To load and compile the class:

1. Open the InterSystems Terminal. For information about opening Terminal for your instance, see InterSystems IRIS Connection Information in *InterSystems IRIS Basics: Connecting an IDE*.
2. Switch to the namespace that you created for this demo. For example:

   set $namespace="samples"

3. Enter the following command to load the class file into the namespace:

   do $system.OBJ.Load("<sample-dir>\sample.cls")

   where <*sample-dir*> is the location where you saved the samples.cls class file.

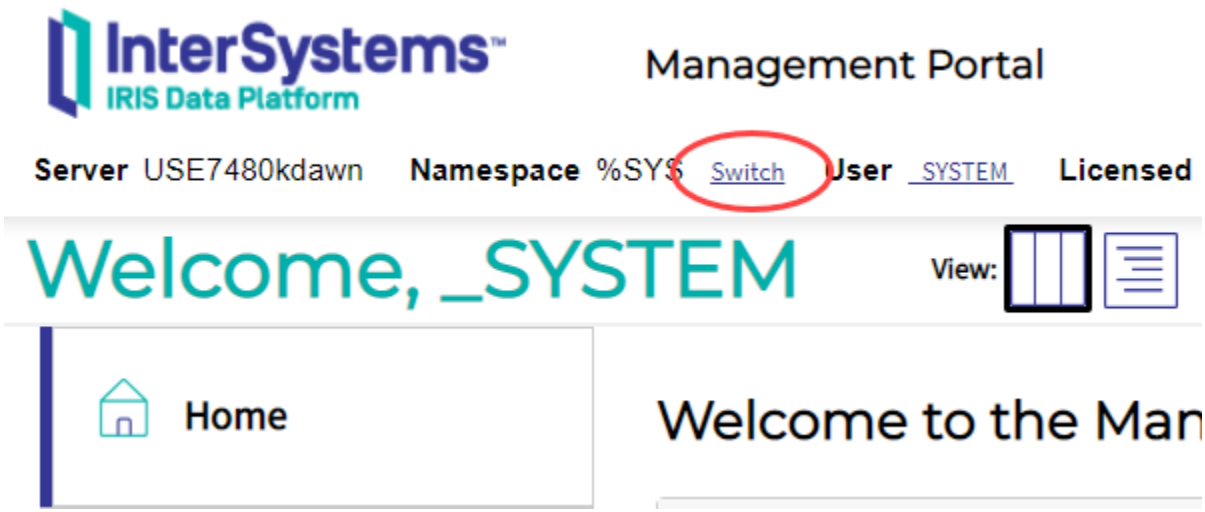
4. Enter the following command to compile the Sample.MyData class that you pasted into sample.cls:

   do $system.OBJ.Compile("Sample.MyData")

4.3 Browsing the Annotation Store

Now that you have compiled the class with the UIMA functional index, you can browse the Annotation Store that was created to preserve the annotations generated by InterSystems IRIS NLP.

1. Open the Management Portal in your browser using the URL for your instance, as described in InterSystems IRIS Connection Information in *InterSystems IRIS Basics: Connecting an IDE*.
2. Switch to the **Samples** namespace using the link in the header.



3. Go to **System Explorer > SQL**.
4. Expand the **Tables** list in the left-hand pane.

   You can see the three tables of the Annotation Store. The naming convention of these tables corresponds to the table (Sample.MyData) that contains the unstructured text that was analyzed.

   - Sample_MyData.Type — Contains an overview of the Annotation Types used in this store.
   - Sample_MyData.Sofa — Contains sofas, which are the text objects that were analyzed by the UIMA analysis engine.
   - Sample_MyData.Annotation — Contains the annotations generated by the analysis engine.

   You can modify the functional index definition to create multiple annotation tables, and then channel the output into the right table based on the annotation type.

4.4 Sending New Text Through the Analysis Pipeline

The power of the UIMA analysis pipeline in InterSystems IRIS is that new unstructured text is automatically sent through the pipeline for analysis and the results added to the Annotation Store. Now that you've created the Annotation Store, you can see how new records added to the Sample.MyData table results in new entries being added to the Annotation Store.

4.4.1 Adding a Record to the Aviation.Event Table

In this step, you will use SQL to add some unstructured text to the Sample.MyData table in the sample database. Remember that this is the table that contains the MyText column on which you defined the functional index. As you will see, annotations are generated automatically when you make this insertion.

1. On the **System Explorer > SQL** page, expand the **Tables** list in the left-hand pane.
2. Select **Sample.MyData**, which is the table that contains the unstructured text that gets sent through the analysis pipeline.
3. In the right-hand pane, click the **Execute Query** tab.
4. To insert a new entry into the sample database, enter the following query into the text box:

   ```
   INSERT INTO Sample.MyData (MyText) VALUES ('First Look unstructured text')
   ```

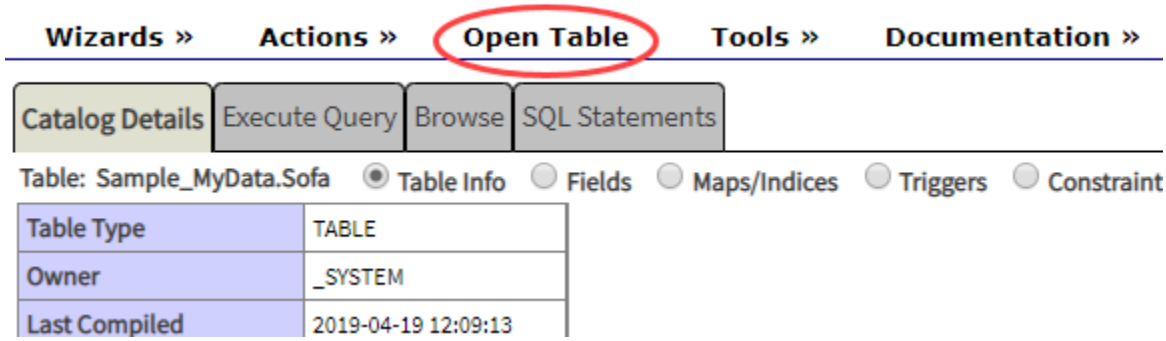
5. Click **Execute**.

   This puts the phrase "First Look unstructured text" into the MyText column of the Sample.MyData table.

4.4.2 Viewing New Entries in the Annotation Store

Now that you have added new unstructured text into the samples database, you can look at the Annotation Store to see how this text was automatically sent through the analysis pipeline. You can see that both the new unstructured text and the annotations from InterSystems IRIS NLP were added to the Annotation Store.

1. On the **System Explorer > SQL** page, expand the **Tables** list in the left-hand pane.
2. Select the **Sample_MyData.Sofa** table.
3. In the right-hand pane, click **Open Table**.



   You can see the new record that was added to the Annotation Store. The **sofaString** is the piece of unstructured text that was processed by the analysis pipeline.

4. Click **Close Window**.
5. In the left-hand pane, select the **Sample_MyData.Annotation** table.
6. Click **Open Table**.

   In the **coveredText** column, you can see the annotations that were generated by the InterSystems IRIS NLP analysis engine.

5 Learn More About UIMA

To learn more about how InterSystems IRIS implements and complements UIMA, see Using InterSystems UIMA.

For a detailed overview of the frameworks, infrastructure, and components of the UIMA standard, see the Apache UIMA home page.