InterSystems UIMA の使用法

Version 2024.1
2024-06-03

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

InterSystems UIMA の使用法
InterSystems IRIS Data Platform　Version 2024.1　2024-06-03

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

目次

1

UIMA サポートの使用

重要　インターシステムズ製品への UIMA の実装は非推奨になっています。インターシステムズ製品の今後のバージョンから削除される可能性があります。以下のドキュメントは、既存ユーザのみに向けたリファレンスとしています。代替のソリューションを見いだすためのサポートを必要とする既存ユーザはインターシステムズのサポート窓口にお問い合わせください。

重要　このバージョンの InterSystems IRIS® には、UIMA を実装するうえで必要な JAR ファイルが付属していません。これらのファイルの入手方法については、"更新済みの Apache UIMA JAR ファイルを構築" を参照してください。

UIMA は、ソース・テキストのアノテーションの生成に使用されます。こういったアノテーションでは、テキスト内の開始位置と終了位置でソース・テキストが参照されます。アノテーションはソース・テキストから分離されており、アノテーションによってソース・テキストが変更されることはありません。

インターシステムズでは、以下の操作を実行できます。

- InterSystems IRIS 自然言語処理 (NLP) を使用して UIMA テキスト・アノテーションを生成します。
- UIMA 標準に準拠したサードパーティのテクノロジやユーザ記述のテクノロジにアクセスして使用し、UIMA テキスト・アノテーションを生成します。
- パイプライン・アーキテクチャで複数のテクノロジを使用して、単一操作で UIMA 準拠の複数のテクノロジからアノテーションを適用します。
- UIMA 準拠の複数のテクノロジからアノテーションを保存および処理します。これらのアノテーションは、インターシステムズで生成されたものと、その他のベンダから受け取ったものの両方です。

InterSystems IRIS 自然言語処理 (NLP) は、UIMA から独立して使用できます。NLP を UIMA と連動させながら同時に使用できます。UIMA は、NLP と相互に作用できる追加テクノロジです。UIMA が以前の NLP インデックス作成や NLP 処理を変更したり、これに置き換わったりすることはありません。

ここでは、以下の UIMA 処理について説明します。

- dll アクセス用の環境変数の設定
- Java ゲートウェイの起動
- UIMA 機能インデックスの定義
  - 機能インデックスによって呼び出される UIMA 分析エンジンの指定
  - 機能インデックスによって作成される UIMA アノテーション・ストアの指定
- アノテーション・ストア SQL テーブルおよびそのフィールドの使用法

- アノテーションのテーブル、フィールド、およびインデックスを追加する XData ブロックの定義によるアノテーション・ストアの変更
- アノテーション・テーブルに配置するアノテーションのフィルタ処理
- アノテーション・テーブルへの手動によるアノテーションの追加
- UIMA 分析エンジンとしての NLP の使用法
- UIMA に対する REST インタフェースの使用法
- InterSystems UIMA タイプ・システム

1.1 UIMA とは、また UIMA の使用目的とは

UIMA は、構造化されていないデータにアノテーションを付けるときの標準を提供します。この標準を使用することによって、InterSystems IRIS 自然言語処理 (NLP) などのさまざまな未構造化データ分析テクノロジで、アノテーションが互いに干渉し合ったり、ソース・テキストの解析機能に干渉したりすることなく、同じソース・テキストにアノテーションを付けることができます。このため、UIMA フレームワークを使用すると、それぞれが 1 つのデータ分析タスクに焦点を置いている異なるテクノロジでアノテーションを組み合わせることができるようになります。UIMA では、テクノロジごとに、そのテクノロジに対するインタフェースとして分析エンジン・オブジェクトが作成されます。これにより、NLP は NLP テキスト分析エンジンを通して UIMA と対話します。一般的に、分析エンジンは、それぞれが独自のアノテーションを作成し、互いに独立して動作します。ところが、UIMA では、分析エンジンは別の分析エンジンが作成したアノテーションを使用できます。

UIMA 標準は、特定のタスク専用の UIMA 準拠テクノロジを実装するためのフレームワークを提供します。こういったテクノロジは、トークン化、意味分析 (InterSystems IRIS NLP)、人や場所の名前を識別するための固有表現認識 (NER) ツール、ルールベースの情報抽出、音声テキスト変換などの処理をサポートできます。

分析エンジンはアノテーションを作成します。アノテーションとは、ソース・テキストのフラグメントに関連付けられたエンコーディングであり、開始文字と終了文字の位置で特定されます。UIMA タイプの文字列が使用され、(必要に応じて) その他のアノテーション固有の機能が使用されます。分析エンジンによって、元のソース・テキストが変更されることはありません。

UIMA では、導入と拡張が UIMA フレームワークで処理されるので、操作が容易です。UIMA は、分散されている可能性のあるアーキテクチャ内のこれらコンポーネントのインスタンスを設定し、呼び出します。UIMA はこの共通のアノテーション形式を通して相互運用性を実現します。

インターシステムズの UIMA 実装は、Apache UIMA から入手できる追加パッケージに準拠しています。このパッケージには、非同期型スケールアウト (UIMA-AS)、分散 UIMA クラスタ・コンピューティング (UIMA-DUCC)、およびルールベースのアノテーション・ワークベンチである UIMA Ruta などが含まれています。

1.1.1 UIMA の用語集

アノテーション：ソース・テキスト内のアノテーションは、uima.tcas.AnnotationBase UIMA タイプから継承し、アノテーション・タイプ名 (Java FQN に類似)、開始位置プロパティ、および終了位置プロパティの 3 つの必須プロパティが指定されます。開始位置と終了位置のプロパティは、ソース・テキストの先頭を 0 としてカウントされる整数の文字位置として表現されます。uima.tcas.TOP UIMA タイプから継承するアノテーションは、ソース・テキストの特定のセクションには関連付けられず、代わりにソース・テキスト自体に関連付けられます。アノテーションには、実装担当者の定義どおりに、テクノロジ固有の追加プロパティを指定できます。アノテーションをネストすることも、アノテーションが他のアノテーションを参照することもできます。

CAS：共通分析構造。分析対象ソース・テキストへの共有アクセスで共通の表現とメカニズムを連動 UIMA コンポーネントに提供するメモリ内オブジェクトです。CAS は 1 つ以上の Sofa を保持するデータ構造です。CAS には常に少なくとも 1 つの Sofa が含まれます。

Sofa：CAS には、処理対象のデータに関する複数の "ビュー" を指定できます。これを Sofa (分析対象) と呼びます。例えば、Web ページの CAS に、HTML マークアップを含めた Web ページ用の Sofa を 1 つ指定して、別の Sofa を Web ページ・テキスト用のみに指定することができます。一般的に、1 つの CAS には 1 つの Sofa のみが指定されます。複数の Sofa が指定されている場合、それぞれが同一データに関する変異形を提供します。例えば、英語、フランス語、およびスペイン語に変換されるソース・テキストは、3 つの Sofa が指定された 1 つの CAS となります。アノテーションは常に個々の Sofa と関連付けられます。1 つの Sofa には、1) 処理されるソース・テキスト・データ、2) 分析エンジンから提供されるアノテーション、3) アノテーションのインデックス、4) アノテーションのタイプ・システムが含まれます。

AE：分析エンジン。構造化されていないデータを分析するテクノロジです。分析エンジンは、テクノロジ (NLP など) の対話相手の UIMA インタフェースを実装する 1 つの UIMA オブジェクトです。各テクノロジに固有の分析エンジンが指定されます。分析エンジンは、XML で UIMA コンポーネント記述子として構成されます。

パイプライン：指定された順序で実行される線系列分析エンジン。分析エンジンでは、互いに独立したアノテーションを作成することも、パイプラインで前にいずれかの分析エンジンによって生成されたアノテーションを入力として受け取り、別の分析エンジンのその生成物に基づいてアノテーションを生成することもできます。インターシステムズでは線形パイプラインがサポートされています。非線形パイプランは、集約分析エンジンによってサポートされる場合があります。非線形パイプラインはインターシステムズによって直接サポートされていませんが、インターシステムズの UIMA 実装と互換性があります。

タイプ・システム：アノテーション・タイプ名を指定するための統一標準。XML 記述子であり、記述子で内部的に使用するアノテーション語彙と出力タイプを指定するセクションが含まれます。この XML 記述子は、機能インデックス AEDESCRIPTOR で指定されます。分析エンジンを呼び出す際には、アノテーション・タイプの認識を、その分析エンジンでサポートされるタイプのサブセットに限定することができます。

1.1.2 UIMA のオンライン・リソース

Apache UIMA のホーム・ページ：http://uima.apache.org/index.html

http://uima.apache.org/d/uimaj-current/references.html#ugr.ref.json.overview

http://uima.apache.org/d/uimaj-current/references.html#ugr.ref.cas

Apache Software Foundation からダウンロード可能な UIMA 準拠のアノテータ：http://uima.apache.org/sandbox.html#UIMA%20Addons%20components

Apache cTAKES、医療患者カルテの構造化されていないデータ専用の大規模な一連のアノテータ：http://ctakes.apache.org/

1.2 IRIS Data Platform における UIMA サポートの概要

InterSystems IRIS® データ・プラットフォームの UIMA サポートは、以下のとおりです。

- UIMA の簡単な呼び出し。これにより、UIMA 準拠のテクノロジを迅速に開発できます。
- InterSystems IRIS 自然言語処理 (NLP) を UIMA 準拠のテクノロジとして使用。
- 汎用 UIMA アノテーション・ストア。InterSystems IRIS から InterSystems IRIS メソッド、SQL クエリ、または REST 操作を使用して UIMA アノテーションにアクセス。

インターシステムズでは、以下の 2 つの方法での UIMA の呼び出しがサポートされています。

- UIMA 機能インデックスを使用した呼び出し。このインデックスは、通常の InterSystems IRIS テーブルで構成できます。この機能インデックスをコンパイルすることによって、ソース・テキストが含まれるテーブルから独立したアノテーションを格納するアノテーション・ストアが作成されます。

- 直接呼び出し。文字列変数に格納されているソース・テキストで UIMA を呼び出します。アノテーションは JSON 配列として返されます。この操作は以下の 2 つの方法のいずれかで実行できます。
  - %UIMA.Utils.IndexNow() メソッド。
  - REST の使用。JSON オブジェクトに "data" プロパティと "descriptors" プロパティを指定して、アノテーションを取得します。必要に応じて "markup" プロパティを追加すると、マークアップ・バージョンの Sofa ソース・テキストも取得します。

いずれのシナリオでも、インターシステムズでは、UIMA アノテーション・ストアでの出力の永続化がサポートされます。永続的な独立したアノテーション・ストアの提供は、Apache UIMA 標準に対する InterSystems 拡張機能の 1 つです。

1.2.1 UIMA 機能インデックス

UIMA 機能インデックスは、InterSystems SQL テーブル・インデックスの 1 つです。UIMA 機能インデックスは、SQL ソース・テーブル内の 1 つの列のコンテンツにインデックスを付けます。このインデックスによって、このソース・テキスト列からデータが UIMA ワークフローに自動的にロードされます。

InterSystems UIMA 機能インデックス・クラスの **%UIMA.Index** は、**%Library.FunctionalIndex** 仕様を実装します。

機能インデックスを定義してコンパイルした後は、SQL ソース・テーブルにレコードを追加することで、ソース・テキストの列のデータが、指定された UIMA 分析エンジン (または線系列の分析エンジン (パイプライン)) によって処理されます。INSERT、UPDATE、DELETE、または %BuildIndices() 操作によって、この機能インデックスは自動的に修正され、UIMA 分析エンジン (複数可) が呼び出されます。

機能インデックスの定義については、以下で説明しています。

1.2.2 UIMA アノテーション・ストア

UIMA 機能インデックスがコンパイルされると、システムは、UIMA アノテーションを格納するための永続クラスをいくつか自動的に生成します。この UIMA アノテーション・ストアは、UIMA フレームワークに対する InterSystems 拡張機能の 1 つです。UIMA アノテーション・ストアは、ソース・テキストのテーブルとは別個であるがリンクされている永続クラス (SQL テーブル) に、UIMA 出力を格納します。InterSystems アノテーション・ストアは、複数のソース・テキストにわたる構造化されていないデータを後で分析できるように、柔軟性のある SQL ベースのアノテーション・ストレージを提供します。これにより、標準の UIMA で作成される大規模な XML アノテーション・ファイルを回避できます。

UIMA アノテーション・ストアは、機能インデックスで指定されたその他の分析エンジンや InterSystems IRIS 自然言語処理 (NLP) で生成されたアノテーション用のアノテーション・ストアとして使用できます。

UIMA アノテーション・ストアは、ソース・テキストのユーザ・ビューに基づいた追加の手動アノテーションを格納するために、機能インデックスから独立して使用することもできます。

1.2.3 UIMA 分析エンジンとしての NLP の使用法

InterSystems IRIS 自然言語処理 (NLP) を UIMA 分析エンジンとして使用して、NLP コンセプトおよびリレーション用の UIMA アノテーションを生成することができます。これらのアノテーションには、他の UIMA テクノロジで提供される UIMA アノテーションと完全な互換性があります。

UIMA から独立して NLP を使用し、NLP コンセプトおよびリレーション用の InterSystems IRIS グローバルを生成することもできます。

UIMA は、NLP 結果で UIMA 準拠インデックス作成を提供しますが、NLP エンジン自体を変更しないので、同一の構造化されていないデータ・ソースで NLP の同時並列使用ができます。詳細は、"InterSystems IRIS 自然言語処理 (NLP) の使用法" を参照してください。

1.3 UIMA 統合の使用法

1.3.1 更新済みの Apache UIMA JAR ファイルを構築

このバージョンの DOCBOOKMACRO(prod) には、UIMA の統合に必要な JAR ファイルが付属していません。

このバージョンの DOCBOOKMACRO(prod) で UIMA を統合する機能を使用するには、Apache UIMA SDK リポジトリの `main-v2` ブランチから、これらの JAR ファイルの更新済みバージョンを入手する必要があります。

Apache Maven と git をシステムにインストールしている場合は以下の手順に従います。

1. コマンド・ラインから、任意に選択したディレクトリに移動し、以下のコマンドを発行して Apache UIMA SDK GitHub リポジトリの `main-v2` ブランチを、そのディレクトリに複製します。

   ```
   git clone -b main-v2 https://github.com/apache/uima-uimaj.git
   ```

2. uima-uimaj/ ディレクトリへ移動します。
3. 以下のコマンドを発行して SDK をビルドします。

   ```
   mvn install
   ```

4. ビルドの完了後、必要な JAR ファイルを SDK キットから DOCBOOKMACRO(prod) インストール・ディレクトリの適切な場所にコピーします。そのためには、以下のコマンドを発行します。<installDir> は、DOCBOOKMACRO(prod) インスタンスのベース・ディレクトリに置き換えます。

   ```
   cp uimaj-core/target/uima-core.jar <installDir>/dev/java/lib/uima/uimaj-core-2.10.3.jar
   cp uimaj-json/target/uimaj-json*.jar <installDir>/dev/java/lib/uima/uimaj-json-2.10.3.jar
   ```

1.3.2 ライブラリの設定

Java Gateway を起動する前に、環境変数を調整して、特定の dll ライブラリ・ファイルをシステム・ローダで使用できるようにする必要があります。

InterSystems IRIS Natural Language Processing (NLP) を UIMA 分析エンジンとして実行するためのライブラリ設定：NLP エンジンは C++ で書かれており、UIMACPP ブリッジを介して UIMA 分析エンジンとして公開されます。(https://github.com/apache/uima-uimacpp を参照。)これにより、Java ベースの UIMA フレームワークで JNI を使用して C++ ライブラリを呼び出すことができます。これにはプラットフォーム固有の設定が必要です。
https://uima.apache.org/d/uimacpp-current/docs/overview_and_setup.html も参照してください。

- Windows プラットフォーム：Windows の場合、PATH 環境変数に InterSystems IRIS インストールの /bin ディレクトリが含まれていることを確認してください。これを行うには、システム環境変数として設定するか、または Java Gateway が起動される端末ウィンドウで構成します。
- Linux および UNIX プラットフォーム：Linux および UNIX の場合、InterSystems IRIS インストールの /bin サブディレクトリを、PATH および LD_LIBRARY_PATH 環境変数の両方に追加する必要があります。これは、システム・レベルまたは Java Gateway が起動される端末セッションのいずれかで行います。
- macOS プラットフォーム：macOS の SIP (Security Integral Protection) では、動的にロードできるライブラリに対して追加の制約を課しています。具体的には、Java がシステム・フォルダにインストールされている場合、SIP では、非システム・ライブラリ (NLP Analysis Engine が利用する UIMACPP ライブラリなど) をロードすることはできません。この問題を回避するには、Java を非システム・フォルダにインストールまたはコピーし、その非システム・パスを使用して Java Gateway を起動し、前述のように DYLD_LIBRARY_PATH を設定します。macOS は開発プラットフォームとしてのみサポートされており、プロダクションについてはサポートされていません。

/bin ディレクトリの場所を確認するには、%SYSTEM.Util クラスの BinaryDirectory() メソッドを呼び出します。

1.3.3 Java ゲートウェイの起動

UIMA 機能インデックスをコンパイルするには、ポート 5555 で Java ゲートウェイを実行している必要があります。この Java ゲートウェイは、InterSystems UIMA Java ゲートウェイへのアクセスを提供します。

Java ゲートウェイは、ObjectScript から Java クラスを呼び出すことができるようにする InterSystems IRIS 機能です。Java ゲートウェイは、TCP/IP ポートで待ち受けるデーモン・プロセスとして実行されます。%Net.Remote.Gateway コードで ObjectScript から送信されるメッセージで要求されるとおりに Java コードを実行するスレッドを生成します。

UIMA Java Gateway com.intersystems.uima.Gateway は、インターシステムズの UIMA 統合で ObjectScript に提供するパブリック・メソッドを保持する Java クラスです。これは、Java ゲートウェイを通して InterSystems IRIS に公開されます。

Java ゲートウェイは、InterSystems IRIS インスタンス (localhost) と同じホストで実行され、ポート 5555 で待ち受ける必要があります。JVM (Java 仮想マシン) の起動時に追加のクラスパス設定は提供されませんが、最小および最大メモリを設定するための通常の JVM パラメータは提供される場合があります。

Windows システムでは、Windows の **[ファイル名を指定して実行]** インタフェースで以下のコマンドを実行することにより Java ゲートウェイを起動できます。

```
%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
```

このコマンドが機能するには、JAVA_HOME を定義しておく必要があります。Windows システムで JAVA_HOME が定義されていない場合は、[コントロール パネル]→[システム] オプション→[詳細設定] システム設定に進みます。[環境変数] ボタンを選択します。JAVA_HOME という名前の新しいシステム変数を定義します。Java bin ディレクトリへのパスを参照し、このパスを JAVA_HOME の値として割り当てます。例：`C:\Program Files\Java\jre1.8.0_141`

1.3.4 UIMA 機能インデックスの定義

SQL テーブル列に格納されている UIMA プロセスの構造化されていないデータをユーザが使用できるようにするには、機能インデックスを定義する必要があります。UIMA 機能インデックスは、タイプが %UIMA.Index のインデックスです。

以下の例では、NYT.Articles SQL テーブルが永続クラスとして定義され (New York Times の記事のテキストのテーブル)、FullArticle 列に UIMA インデックスが定義されます。

Class Definition

```
Class NYT.Articles Extends %Persistent
{
Property NYTID As %Integer;
Property PubDate As %Date;
Property FullArticle As %String(MAXLEN=32000);
Index IdxNYTArticles On (FullArticle) As %UIMA.Index(
    AEDESCRIPTOR = "classpath:/com/intersystems/uima/annotator/iKnowEngine.xml"
    );
}
```

この機能インデックスには以下のパラメータを指定します。

- AEDESCRIPTOR ― 必須 ― 列データの処理で実行が必要な UIMA 分析エンジンを指定する UIMA コンポーネント記述子を指定します。以下のいずれかを指定します。
  - 現在のクラスにある XData ブロックの名前。ブロックには記述子の完全な XML が含まれます。例：`"iKnowEngine"`。詳細は、″XData ブロックの定義と使用″ を参照してください。
  - クラスパスでアクセス可能な記述子ファイルへのパス。`classpath:` の接頭語が付きます。例：`"classpath:/com/intersystems/uima/annotator/iKnowEngine.xml"`。

- 記述子ファイルへのフル・パス。例えば、Windows システムの場合は "C:\UIMA\Descriptors\MyAE.xml"、UNIX システムの場合は "//user/UIMA/Descriptors/MyAE.xml" です。

1 つの UIMA コンポーネント記述子を 1 つの UIMA 分析エンジン向けに引用符付き文字列として指定することも、セミコロンで区切られた複数の UIMA コンポーネント記述子文字列を指定することもできます。複数の UIMA 分析エンジンは、UIMA 処理パイプラインを作成して、指定した順序で実行されます。記述子のリストにアノテーション・ファイラを指定する必要はありません。アノテーション・ファイラは、パイプライン内の最後の分析エンジンとして自動的に指定されます。

- ADDITIONALCLASSPATH ー オプション ー AEDESCRIPTOR で記述されている分析エンジンを実行するためにクラスパスに追加ライブラリ (jarfiles) をロードする必要がある場合に、このライブラリを指定する追加のクラスパス (複数可)。複数のパスは、セミコロン区切りのリストで指定します。

  または、Java ゲートウェイの起動時に追加クラスパスを -classpath コマンド行引数に追加することによって、これらのクラスパスを Java ゲートウェイ・クラスパスに追加することもできます。

- ANNOTATIONSTOREDEF ー オプション ー アノテーション・ストアに対して構成する追加の機能とインデックスを定義する XData ブロックの名前。これを指定しない場合、既定ではインデックスと同じ名前の XData ブロックが使用されます。XData ブロックは、アノテーション・ストア定義 XML と同じクラス内に含まれている必要があります。詳細は、"XData ブロックの定義と使用" を参照してください。

機能インデックスが適用されるテーブルをコンパイルする必要があります。これにより、%FunctionalIndex フレームワークに固有の多数のメソッドがテーブル自体に生成されます。また、アノテーション・ストア記述子 XML (ANNOTATIONSTOREDEF) が検証され、合格すると、それに基づいて適切な ObjectScript クラスが生成されます。

テーブルのコンパイルでは、機能インデックスから参照される AEDESCRIPTOR のテストも実行されます。このテストは、Java ゲートウェイの呼び出しを通して実行されます。このため、既定で、UIMA 機能インデックスでテーブルをコンパイルするときに Java ゲートウェイを実行している必要があります。

インデックス・パラメータ TESTONCOMPILE=0 を設定することにより、アクティブな Java ゲートウェイなしでこのテーブルのコンパイルを実行できます。これによって AEDESCRIPTOR のテストが実行されなくなります。ただし、データをテーブルに挿入する際には、アクティブな Java ゲートウェイが必要です。

1.3.5 UIMA コンポーネントの直接呼び出し

UIMA コンポーネントを直接呼び出すには、%UIMA.Utils.IndexNow() メソッドを使用できます。

1.4 UIMA アノテーション・ストアの使用法

UIMA 機能インデックスのコンパイルにより、UIMA アノテーション・ストアが自動的に生成されます。UIMA アノテーション・ストアは、一連の永続クラス (SQL テーブル) を含むパッケージです。

アノテーション・ストアは、機能インデックスを通して提供される XML を使用して、またはアノテーション・ファイラを使用して直接、構成されます。アノテーション・ストアの構成オプションは、"%UIMA.Model.annotationStore" を参照してください。

アノテーション・ストア・クラスは、%UIMA.AnnotationStore.* スーパークラスから継承し、%UIMA.AnnotationStore.ClassGenerator によって生成されます。

1.4.1 アノテーション・ファイラ

UIMA アノテーション・ファイラ (com.intersys.uima.filer.AnnotationFiler) は、UIMA 分析エンジン・インタフェースを実装する Java クラスであるため、通常の UIMA コンポーネントとして動作します。UIMA アノテーション・ファイラ自体が分析

エンジンであり、UIMA 処理パイプラインの最後の分析エンジンです。自身でアノテーションを追加するのではなく、指定された CAS (メモリ内テキスト・オブジェクト) から既存のすべてのアノテーションを読み取ります。読み取ったアノテーションを UIMA アノテーション・ストアに格納するために、InterSystems IRIS に送信して戻します。これは、UIMA 用語では CAS コンシューマと呼ばれることがあります。

他の分析エンジンと同様に、アノテーション・ファイラは XML で UIMA コンポーネント記述子ファイルを使用して構成され、データベース接続パラメータ、データのファイリング先のアノテーション・ストアの識別子、およびアノテーション・ストアの XML 記述を含みます。

注釈　　アノテーション・ファイラのこれらのパラメータはすべて、UIMA 機能インデックスで自動的に構成されます。

1.4.2 アノテーション・ストアのテーブル

UIMA 機能インデックスのコンパイルにより、UIMA アノテーション・ストアが自動的に生成されます。UIMA アノテーション・ストアは、UIMA パイプラインで処理される指定されたデータセットのアノテーションすべてを格納する一連の InterSystems IRIS テーブルです。このパイプラインは、UIMA アノテーション・ファイラをパイプライン内の最後の分析エンジンとして、1 つ以上の UIMA 分析エンジンで構成することができます。

既定で、アノテーション・ストアには、インデックスが定義される永続クラスと同じ名前が付けられます。このため、ここで提示している例では、テーブル `NYT.Articles` の列の機能インデックスをコンパイルすると、`NYT_Articles` という名前の対応するアノテーション・ストア・パッケージが生成されます。この命名の既定値は変更することができます。

UIMA アノテーション・ストアは、(少なくとも) 以下の一連のテーブルで構成されます。

- UIMA タイプ用のテーブル。例：`NYT_Articles.Type`
- Sofa (テキスト・オブジェクト) 用のテーブル。例：`NYT_Articles.Sofa`
- アノテーション自身用のテーブル。例：`NYT_Articles.Annotation`

これらは SQL テーブルであるため、テーブルのコンテンツには標準の SQL クエリを使用してアクセスできます。

1.4.2.1 タイプ・テーブル

タイプ・テーブルは、以下のフィールドで構成されています。

フィールド	データ型	インデックス	目的
name	String	一意のインデックス	アノテーションのタイプ。
parent	Integer		タイプ・テーブルを参照します。

1.4.2.2 Sofa テーブル

Sofa テーブルは、以下のフィールドで構成されています。

フィールド	データ型	インデックス	目的
docID	Integer	ビットマップ・インデックス	ソース・テキスト・テーブルを参照します。
hasManualAnnotations	Boolean	ビットマップ・インデックス	手動アノテーションがあるかどうかを指定します。
mimeType	String		MIME タイプ（メディア・タイプとも呼ばれる）、Sofa によって表現されるデータのタイプの ISO 標準記述。
sofaID	String	ビットマップ・インデックス	
sofaString	String		この Sofa のソース・テキスト。

1.4.2.3 アノテーション・テーブル

アノテーション・テーブルは、以下のフィールドで構成されています。

フィールド	データ型	インデックス	目的
begin	Integer	%pos 標準インデックス	アノテーション・テキストの開始文字位置。位置は Unicode 文字の数で、0 からカウントされます。
coveredText	String		アノテーション・テキスト。begin および end 文字位置で特定されるソース・テキストのセクションの正確なコピー。
docID	Integer	ビットマップ・インデックス	ソース・テキスト・テーブルを参照します。
end	Integer	%pos 標準インデックス	アノテーション・テキストの終了文字位置。
isManual	Boolean	ビットマップ・インデックス	手動アノテーションにフラグを付けます。
sofaID	Integer		Sofa テーブルを参照します。
typeID	Integer	ビットマップ・インデックス	タイプ・テーブルを参照します。

最上位アノテーション・テーブルには、`begin` フィールド、`coveredText` フィールド、および `end` フィールドは含まれません。

アノテーション・テーブルでは、分析エンジンが生成した追加フィールド値を受け取ることができます。このような追加フィールド値に対してアノテーション・フィールドが定義されていない場合、それらの値は key:value ペアの JSON 配列として一般の features フィールドに格納されます。

アノテーションには、%UIMA.AnnotationStore.Store メソッドの GetAnnotations() および GetAnnotationsRS() を使用してアクセスできます。

1.4.3 アノテーション・ストアの調整

既定で、アノテーション・ストアには、インデックスが定義される永続クラスと同じパッケージ名が付けられます。必要に応じてアノテーション・ストアのパッケージ名を変更したり、Xdata ブロックを指定することによってアノテーション・ストアに機能を追加したりできます。この Xdata ブロックを使用して、追加のテーブル、列、インデックス、フィルタなど、アノテーション・ストアの追加機能を定義できます。これは、以下の 2 つの方法のいずれかで行うことができます。

- 機能インデックスと同じ名前で XData ブロックを作成し、その XData ブロックの XMLNamespace キーワードを UIMA アノテーション・ストアに設定します。例えば以下のようにします。

```
Class NYT.Articles Extends %Persistent
{
Property NYTID As %Integer;
Property PubDate As %Date;
Property FullArticle As %String(MAXLEN=32000);
Index IdxNYTArticles On (FullArticle) As %UIMA.Index(
    AEDESCRIPTOR = "classpath:/com/intersys/uima/annotator/iKnowEngine.xml"
    );
XData IdxNYTArticles [ XMLNamespace = "http://www.intersystems.com/UIMA/annotationStore" ]
  { ... }
}
```

- 異なる名前で XData ブロックを定義し、これを機能インデックスの ANNOTATIONSTOREDEF パラメータに指定します。例えば以下のようにします。

```
Class NYT.Articles Extends %Persistent
{
Property NYTID As %Integer;
Property PubDate As %Date;
Property FullArticle As %String(MAXLEN=32000);
Index IdxNYTArticles On (FullArticle) As %UIMA.Index(
    AEDESCRIPTOR = "classpath:/com/intersys/uima/annotator/iKnowEngine.xml",
    ANNOTATIONSTOREDEF = "NYTExtra"
    );
XData NYTExtra [ XMLNamespace = "http://www.intersystems.com/UIMA/annotationStore" ]
  { ... }
}
```

XData ブロックには、XML 形式のデータが含まれます。詳細は、"XData ブロックの定義と使用" を参照してください。

UIMA アノテーション・ストアには、追加アノテーション・テーブル、それらのテーブルの追加列、その追加インデックスを含めることができます。

1.4.3.1 名前の変更

以下のように、XData ブロックで、UIMA アノテーション・ストアに別のパッケージ名を指定することができます。

```
XData IdxNYTArticles [ XMLNamespace = "http://www.intersystems.com/UIMA/annotationStore" ]
  {<store package="NYT.ArticleAS">
    </store>
  }
```

以下のように、XData ブロックで、アノテーション・テーブルの名前を変更することができます。

```
XData IdxNYTArticles [ XMLNamespace = "http://www.intersystems.com/UIMA/annotationStore" ]
  {<store package="NYT.Articles">
   <tables>
     <table name="FilteredAnnotation">
     </table>
   </tables>
   </store>
  }
```

1.4.3.2 追加テーブルの指定

アノテーションを格納するための追加アノテーション・ストア・テーブルを指定できます。例えば、2 つの異なる分析エンジンからのアノテーションを別々のアノテーション・テーブルに格納するようアノテーション・テーブルを作成できます。以

下の XData ブロックでは、2 つのアノテーション・テーブルが作成されます。2 つ目のアノテーション・テーブルには、normalizedValue という名前の追加フィールドがあります。

```
XData IdxNYTArticles [ XMLNamespace = "http://www.intersystems.com/UIMA/annotationStore" ]
  {<store package="NYT.Articles">
   <tables>
     <table name="Annotation1">
     </table>
     <table name="Annotation2">
       <features storeOther="json" >
         <feature name="normalizedValue" path="normalizedValue" >
         <parameter key="MAXLEN">300</parameter>
         </feature>
       </features>
     </table>
   </tables>
    </store>
   }
```

%UIMA.Model.annotationStore で、tables プロパティに追加のアノテーション・テーブルを指定します。テーブルの定義は、%UIMA.Model.table を使用して行うことができます。

最上位アノテーション用に追加テーブルの作成が必要な場合があります。topLevel ブーリアン・プロパティを使用すると、アノテーション・テーブルにソース・テキスト内のアノテーション（開始および終了文字位置で定義されるテキスト単位に対するアノテーション）を含めるかどうか、またはソース・テキスト全体に適用される ″最上位″ アノテーションのテーブルにするかどうかを指定できます。

1.4.3.3 追加列の指定

アノテーション・テーブルへの列の追加が必要な場合があります。例えば、最上位アノテーション・テーブルに、NLP 優位性スコア用のフィールドの追加が必要な場合があります。

アノテーション・テーブルの既存フィールドに対応しないアノテーション・フィールドが分析エンジンで生成されると、これらの値は一般の features フィールドに key:value ペアの JSON 配列として格納されます。

列を追加するには、機能インデックスに提供される XData ブロックで各列を feature プロパティとして指定します。feature は、table.内の features 内に定義する必要があります。1 つ以上の追加列を定義することでも、一般の features 列が自動的に定義されます。以下の例では、3 つの追加フィールドと一般の features フィールドが定義されます。

```
XData IdxNYTArticles [ XMLNamespace = "http://www.intersystems.com/UIMA/annotationStore" ]
 {
 <store package="NYT.Articles">
 <tables>
 <table>
   <features storeOther="json" >
     <feature name="normalizedValue" path="normalizedValue" >
    <parameter key="MAXLEN">300</parameter>
    </feature>
    <feature name="occurrences" path="occurrences" type=":annotationList:Annotation" />
    <feature name="parent" path="_parent" type=":annotation" />
  </features>
 </table>
 </tables>
 </store>
 }
```

1.4.3.4 追加インデックスの指定

%UIMA.Model.table で、indices プロパティに追加のアノテーション・テーブル・インデックスを指定します。インデックスの定義は、%UIMA.Model.index を使用して行うことができます。

これらの追加インデックスは、%UIMA.Index ANNOTATIONSTOREDEF パラメータで指定する XData ブロックに入力する必要があります。index は、table.内に定義する必要があります。以下の例では、追加フィールドが定義され、そのフィールドにインデックスが作成されます。

```
XData IdxNYTArticles [ XMLNamespace = "http://www.intersystems.com/UIMA/annotationStore" ]
 {
 <store package="NYT.Articles">
 <tables>
  <table>
   <features storeOther="json" >
     <feature name="normalizedValue" path="normalizedValue" >
     <parameter key="MAXLEN">300</parameter>
     </feature>
   </features>
   <indices>
     <index properties="normalizedValue" name="NV" />
   </indices>
  </table>
 </tables>
 </store>
 }
```

1.4.4 アノテーション・フィルタの追加

既定で、生成されたアノテーションはすべてアノテーション・ストアに格納されます。分析エンジンで生成されるアノテーションに適用する 1 つ以上のフィルタを指定できます。これらのフィルタは、フィルタで指定されている UIMA タイプのアノテーションのみを格納するようアノテーション・ファイラに指示します。フィルタを適用することによって、フィルタで明示的に指定されているもの以外のすべてのアノテーションが自動的に除外されることに注意してください。

フィルタは、%UIMA.Index ANNOTATIONSTOREDEF パラメータで指定する XData ブロックに入力する必要があります。以下に、フィルタを指定する XData ブロックの例を示します。フィルタは table 内ではなく、store 内で定義されることに注意してください。

```
XData IdxNYTArticles [ XMLNamespace = "http://www.intersystems.com/UIMA/annotationStore" ]
 {
 <store package="NYT.Articles">
   <filters>
     <include>
     <exclude pattern="org.apache.uima.alchemy.ts.entity.AlchemyAnnotation" />
     </include>
   </filters>
   <tables>
     <table name="Annotation1">
     </table>
   </tables>
 </store>
 }
```

"%UIMA.Model.filters" および "%UIMA.Model.filterRule" を参照してください。

1.4.5 手動アノテーション

ソース・テキストに対して分析エンジンを実行した後に、そのソース・テキストに含めたい追加アノテーションがあることに気付く場合があります。これらのアノテーションは、手動アノテーションとしてアノテーション・ストアに直接入力できます。

アノテーション・ストアに単一アノテーションを手動で挿入するには、%UIMA.AnnotationStore.Store.FileAnnotation() メソッドを使用できます。

アノテーション・ストアにアノテーションの配列を手動で挿入するには、%UIMA.AnnotationStore.Store.FileAnnotations() メソッドを使用できます。

UIMA によって、isManual ブーリアン・フィールドを使用して、アノテーション・テーブル内の手動アノテーションにフラグが付けられます。

既定で、指定された分析エンジンが生成したアノテーションが追加される前に、機能インデックスによって、ソース・テキストの前のアノテーションがすべて削除されます。前のアノテーションを維持する場合は、これらに手動アノテーションとしてフラグを付ける必要があります。

1.5 UIMA 分析エンジンとしての NLP の使用法

InterSystems IRIS 自然言語処理 (NLP) エンジンは、UIMA アノテータとして公開され、クラスパスに既にロードされており、"classpath:/com/intersys/uima/annotator/iKnowEngine.xml" からアクセスできます。UIMA 機能インデックスを定義することにより、NLP を UIMA アノテータとして使用できます。これによって、NLP インデックスを定義する UIMA から独立した NLP の同時使用は制限されません。NLP インデックスは InterSystems IRIS グローバルとして格納されます。

NLP を UIMA アノテータとして使用する場合、NLP は NLP の構文分析の結果をコンパイル可能な UIMACPP ヘッダ・ファイルに配置した後、このデータを NLP UIMA ラッパに格納して、NLP エンジンに送信します。これにより、NLP は UIMA アノテーションに対して、これらのアノテーションが InterSystems IRIS グローバルであるかのように処理を実行できるようになります。NLP エンジン自体は変更されません。

NLP アノテーション・タイプには、コンセプト、リレーション、否定、肯定的感情、否定的感情があります。最上位アノテーションのタイプには、一意のエンティティと優位性スコアがあります。

NLP UIMA ラッパ・コードは、UIMACPP (UIMA C++) ライブラリをインポートします。このライブラリは Xerces ライブラリと APR-1 ライブラリに依存します。Xerces は既に InterSystems IRIS コード・ベースに含まれています。UIMACPP と APR-1 は InterSystems UIMA 実装の一部として提供されています。両方とも Apache Software Foundation によって維持管理されます。

1.5.1 NLP アノテーション・タイプ・システム

NLP では、メタデータ・パラメータ **language** を使用する Sofa Unicode テキスト・データ形式がサポートされます。NLP では、この **language** パラメータを使用して、対応する言語モデルが選択されます。言語は 2 文字の ISO 言語コードを使用して指定されます。例えば、英語の場合は **en** です。Sofa は包含するソース・テキストとして識別されるため、NLP 自動言語識別 (ALI) はサポートされません。Sofa に対して識別された言語は、NLP UIMA インデックスの定数と見なされます。

1.6 UIMA REST API

UIMA REST API には、Analytics や NLP でこれに相当するものと同様に、%Api.UIMA 内のさまざまなネームスペースに転送される既定の一般的な Web アプリケーションがありますが、個々のバージョンの API は、**%UIMA.REST.v1** クラスを使用して別の Web アプリにロックすることができます。

1.6.1 REST API の基礎

%UIMA.REST.v1 は、REST 経由で UIMA 機能にアクセスするためのエンドポイントを提供します。以下の URL で REST サービスを設定する必要があります。<baseURL> はインスタンスのベース URL です。

```
https:<baseURL>/api/uima/v1/<namespace>
```

以下の REST 操作がサポートされています。

- アノテーション・ストアの情報を取得します。この情報には、アノテーション・ストア・テーブルの名前、テーブルに含まれるのは標準アノテーションかそれとも最上位アノテーションか、テーブルの列、インデックス、およびフィルタが含ま

れます。必要に応じて、テーブルに含まれるアノテーション・タイプのリストやアノテーション・カウント (行カウント) などの実行時統計も含まれます。

```
GET http://localhost:52773/api/uima/v1/<namespace>/store/<annotation store name>/info
```

GET と POST の両方がサポートされています。POST オブジェクトを使用して入力できる AnnotationStoreInfo パラメータに基づいて情報を取得します。

- アノテーションを取得します。

```
GET http://localhost:52773/api/uima/v1/<namespace>/store/<annotation store
name>/annotations
```

GET と POST の両方がサポートされています。POST オブジェクトを使用して入力できる ASRequestObject パラメータに基づいて、一連のアノテーションを取得します。取得するアノテーションの最大数を指定できます。既定値は 500 です。取得するドキュメント ID、アノテーション・テーブル、列、およびアノテーション・タイプを指定できます。既定ではすべてを取得します。ドキュメント ID は、コンマ区切りの整数値のリストとして指定します。ドキュメント ID に既定値はありません。結果は、GetAnnotations() メソッドと同じパラメータが指定された配列として返されます。

- ドキュメントとその詳細を取得します。

```
GET http://localhost:52773/api/uima/v1/<namespace>/store/<annotation store
name>/document/<docID>
```

GET と POST の両方がサポートされています。ドキュメントのソース・テキストを取得します。POST オブジェクトを使用して入力される ASRequestObject パラメータに基づいて、ドキュメントのアノテーションを取得することもできます。その POST オブジェクトに Markup パラメータを追加すると、アノテーションを使用しているテキストがハイライト表示されます。既定で、Markup では CSS (カスケーディング・スタイル・シート) クラスが使用されます。

- ソース・テキストを処理します。

```
POST http://localhost:52773/api/uima/v1/<namespace>/process
```

ソース・テキストおよびそれを処理する UIMA コンポーネント記述子を指定します。ASRequestObject パラメータと Markup パラメータを指定することで、結果のアノテーションとマークアップ・バージョンのドキュメントを要求できます。

1.6.2 Swagger リファレンス・ドキュメントへのアクセス

UIMA REST API のドキュメントは、OpenAPI Specification (Swagger とも呼ばれます) を使用して完全に作成されます。YAML の記述には "/swagger" エンドポイントからアクセスでき、この API の最上部で便利な GUI 機能を提供する swagger-ui へ直接ロードできます。

それを使用するには、swagger-ui をインストールするか、http://petstore.swagger.io に移動して、このエンドポイントを指します。

- swagger-ui をダウンロードします。
- swagger-ui ダウンロードを unzip します。
- Swagger ボックスに以下のどちらかを指定します。
  - 最小のセキュリティでインストールされている InterSystems IRIS の場合 : `https:<baseURL>/api/uima/v1/samples/swagger` (`samples` を必要なネームスペースに置き換えます)。

    <baseURL> はインスタンスのベース URL です。
  - 通常のセキュリティでインストールされている InterSystems IRIS の場合 : `https:<baseURL>/api/uima/v1/samples/swagger?IRISUsername=myname&IRISPassword=mypassword`

(samples を必要なネームスペースに、myname を InterSystems IRIS ユーザ名に、mypassword を InterSystems IRIS アカウントのパスワードに置き換えます)。

- [エクスプローラ] ボタンを選択します。

1.7 リファレンス資料

1.7.1 InterSystems UIMA タイプ・システム

インターシステムズでは、ソース・テキスト内のアノテーション向けとソース・テキスト全体に適用される最上位アノテーション向けの 2 つのアノテーション・タイプ・システムがサポートされています。

1.7.1.1 テキスト内のアノテーション

- uima.tcas.Annotation :

  サブタイプ :

  - com.intersystems.uima.annotation.iknow.Base : NLP によって生成される通常のアノテーションすべてに使用される抽象スーパータイプ

    サブタイプ :

    - com.intersystems.uima.annotation.iknow.EntityOccurrence : テキスト自体でのエンティティの出現

      { entity (com.intersystems.uima.annotation.iknow.Entity) : テキストに出現するドキュメントレベルの Entity アノテーションへの参照 }

    - com.intersystems.uima.annotation.iknow.Sentence : NLP によって識別される文

    - com.intersystems.uima.annotation.iknow.Path : 文のサブセットとして意味的に関連性のあるエンティティ・シーケンス

      { entities (uima.cas.FSArray[com.intersystems.uima.annotation.iknow.EntityOccurrence]) : このパスを構成しているエンティティ }

    - com.intersystems.uima.annotation.iknow.Attribute : 1 つのパスまたは文内の複数のエンティティに影響を与える意味的属性

      サブタイプ :

      - com.intersystems.uima.annotation.iknow.Negation : 否定を表す意味的属性
      - com.intersystems.uima.annotation.iknow.PositiveSentiment : 肯定的感情を表す意味的属性
      - com.intersystems.uima.annotation.iknow.NegativeSentiment : 否定的感情を表す意味的属性

    - com.intersystems.uima.annotation.iknow.AttributeMarker : 意味的属性を示唆する句

      { scope (com.intersystems.uima.annotation.iknow.Attribute) : このマーカが示唆する意味的属性アノテーションへの参照}

      サブタイプ :

      - com.intersystems.uima.annotation.iknow.NegationMarker : 否定を表す意味的属性マーカ
      - com.intersystems.uima.annotation.iknow.PositiveSentimentMarker : 肯定的感情を表す意味的属性マーカ

- com.intersystems.uima.annotation.iknow.NegativeSentimentMarker：否定的感情を表す意味的属性マーカ

・ com.intersystems.uima.annotation.iknow.util.UDLabel：ユーザ定義ラベルの抽象スーパータイプ

サブタイプ：

- com.intersystems.uima.annotation.iknow.util.UDConcept：NLP エンジンがコンセプトとして解釈する必要がある対象のユーザ定義ラベル
- com.intersystems.uima.annotation.iknow.util.UDRelation：NLP エンジンがリレーションとして解釈する必要がある対象のユーザ定義ラベル
- com.intersystems.uima.annotation.iknow.util.UDNegation：NLP エンジンが否定マーカとして解釈する必要がある対象のユーザ定義ラベル
- com.intersystems.uima.annotation.iknow.util.UDPositiveSentiment：NLP エンジンが肯定的感情マーカとして解釈する必要がある対象のユーザ定義ラベル
- com.intersystems.uima.annotation.iknow.util.UDNegativeSentiment：NLP エンジンが否定的感情マーカとして解釈する必要がある対象のユーザ定義ラベル

1.7.1.2 最上位アノテーション

・ uima.cas.TOP：

サブタイプ：

- com.intersystems.uima.annotation.iknow.TOP：NLP によって生成されるドキュメントレベルのアノテーションすべてに使用される抽象スーパータイプ

  サブタイプ：

  ・ com.intersystems.uima.annotation.iknow.Entity：NLP によって生成される主要アノテーション・タイプで、意味的に関連性のある単語グループを識別します。サブタイプの Concept、Relation、および PathRelevant を参照してください。

  { normalizedValue (uima.cas.String)：エンティティの正規化された値 (小文字にされ、関係のない句読点が削除されます)

  dominance (uima.cas.Double)：ドキュメント内の対象エンティティの関連性を表す NLP 固有のメトリック }

  サブタイプ：

  - com.intersystems.uima.annotation.iknow.Concept：コンセプトは、テキスト作成者が表現しているとおりに、自身で意味を持つエンティティです。
  - com.intersystems.uima.annotation.iknow.Relation：リレーションは、同じパスまたは文内のコンセプト (またはパス関係) 間のリンクまたはリレーションシップを表すエンティティです。
  - com.intersys.uima.annotation.iknow.PathRelevant：パス関係の用語は、パス内でロールを持つ用語ですが、それ自体に意味はありません。これらのエンティティに優位性の値が割り当てられることはありません。

  ・ com.intersystems.uima.annotation.iknow.ProximityScore：指定ペアのエンティティの近似スコア。対象エンティティの意味的関連の近接性を表します。

  { origin (com.intersystems.uima.annotation.iknow.Concept)

  destination (com.intersystems.uima.annotation.iknow.Concept)

proximity (uima.cas.Double) }