XCas

Scott Cyphers

August 7, 2015

Abstract

XCas is a simple design pattern that augments the Apache UIMA CAS to provide modular type-safe access to external resources. Every potential user of an XCas defines the interface that it requires. A pipeline defines one or more XCas resources that provide access to implementations of the required interfaces. At appropriate points during processing, a CAS is bound to such an implementation. Annotators include the resource specification and obtain the XCas implementation from their CAS during processing.

Annotator Definition

Annotators define a public static `XCas` interface with all their required methods. The class `XCasAnnotator_ImplBase<XCas>` provides a convenient mechanism for declaring the external resource. For example, Figure **??** defines an annotator that require one XCas method. The `getDescription` method returns a description for this annotator, given a resource that implements the interface. The `XCasAnnotator_ImplBase` implements the `process(JCas jCas)` method, obtains the associated XCas, and passes both to `process`.

Resource Definition

Resource definition is also simple. The resource defines an public static `XCas` interface that extends a set of `XCas` interfaces which includes the `XCas` interfaces for the annotators that will use it. The resource is a singleton that associates a `JCas` with an implementation of its `XCas` interface. The class `XCasResource_Impl<XCas>` extends `ExternalResourceLocator` to provide methods for managing the associations, as well as a `getResourceDescription` method which creates a `ExternalResourceDescription` for this resource.

```
public class SignalWordsAnnotator
extends XCasAnnotator_ImplBase<SignalWordsAnnotator.XCas>
{
   public static interface XCas
   {
      SignalWords getSignalWords();
   }

   public static <T extends XCas>
   AnalysisEngineDescription getDescription(XCasResource<T> resource)
   throws ResourceInitializationException
   {
      return getDescription(SignalWordsAnnotator.class, resource);
   }

   @Override
   public void process(JCas jCas, XCas xCas)
   throws AnalysisEngineProcessException
   {
        xCas.getSignalWords().compute(jCas);
   }
}
```

Figure 1: Simple annotator definition

```
public class QuestionASResource
extends XCasResource_Impl<QuestionASResource.XCas>
{
   public static interface XCas extends
   ChunkVectorAnnotator.XCas,
   ClosestChunkAnnotator.QuestionXCas,
   ClosestSentenceAnnotator.QuestionXCas,
   CommentInitializer.QuestionXCas,
   Configuration.QuestionXCas,
   Doc2VecAnnotator.XCas,
   Doc2VecFeatureVectorAnnotator.QuestionXCas,
   DocumentInitializer.XCas,
   DocumentVectorAnnotator.XCas,
   QuestionCommentCASMultiplier.XCas,
   SentenceVectorAnnotator.XCas,
   TokenVectorAnnotator.XCas,
   TreeStringAnnotator.XCas,
   WordWeightAnnotator.XCas
   {}

   public final static
   QuestionASResource resource = new QuestionASResource();

   @Override
   public Object getResource()
   {
      return resource;
   }
}
```

Figure 2: `XCas` Resource definition

```
public class QuestionAS
implements QuestionASResource.XCas
{
   @Override
   SignalWords getSignalWords()
   {
      ...
   }

   ...
}
```

Figure 3: Implementation of `XCas`

XCas Implementation

Any class that implements the the resource's `XCas` interface and be bound to a `JCas` by the resource. For example, Figure **??** shows an implementation of `QuationASResource.XCas`.

Setting up a Pipeline

Binding an XCas to a JCas

XCas Strategies

Implementation