CWL Java SDK

Manual

Paul Grosu
pgrosu@gmail.com

Introduction

This manual will describe how to generate the CWL Java SDK.

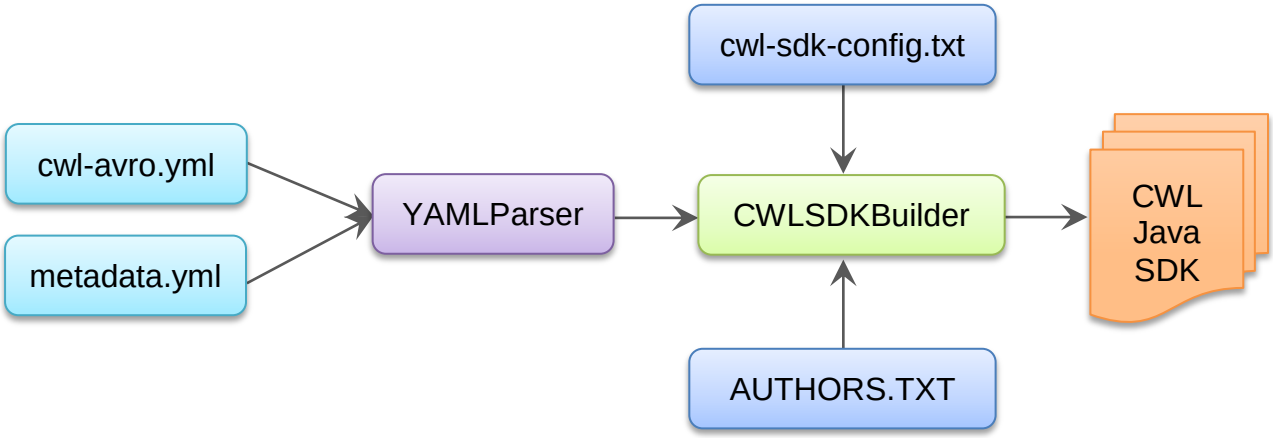
Requirements

Below are the requirements for generating the CWL Java SDK:

- Java SDK
- cwl-avro.yml
- metaschema.yml
- AUTHORS.txt
- cwl-sdk-config.txt
- CreateCWLJavaSDK.java
- YAMLParser.java
- CWLSDKBuilder.java
- compile-and-run.sh

Process of Generating the SDK

The process by which the SDK is generated is as follows:



The *YAMLParser* collects the name, type, fields, inheritance and *specializations* from the YAML files, which get provided to the *CWLSDKBuilder*. The CWLSDKBuilder in conjunction with a configuration file called *cwl-sdk-config.txt* and AUTHORS.TXT file will generate the SDK in the directory called *sdk*.

The Config File

The cwl-sdk-config.txt file has three options:

- To *override* the default class creation.
- To *skip* the creation of a class
- To assign a *namespace* to a variable

Each is specified on its own line and there can be multiple definitions. Below is a portion of the file:

```
override : Sink : interface
override : NamedType : interface
override : EnumSchema : interface
skip : PrimitiveType
skip : "Any"
namespace : _id : sld
namespace : specializeFrom : sld
namespace : specializeTo : sld
namespace : doc : sld
namespace : docParent : sld
namespace : name : foaf
namespace : mbox : foaf
namespace : cwl : https://w3id.org/cwl/cwl#
namespace : sld : https://w3id.org/cwl/salad#
namespace : avro : https://w3id.org/cwl/avro#
namespace : dct : http://purl.org/dc/terms/
namespace : rdf : http://www.w3.org/1999/02/22-rdf-syntax-ns#
namespace : rdfs : http://www.w3.org/2000/01/rdf-schema#
namespace : foaf : http://xmlns.com/foaf/0.1/
namespace : xsd : http://www.w3.org/2001/XMLSchema#
```

To *override* a default - which is to create a class – one can specify which classes should be interfaces by the following:

override : ClassName : interface

The *colon* (:) is the process of separating each aspect of the definition.

To *skip* the creation of classes for specific files, this is performed as follows:

skip : ClassName

To define a *namespace* for a specific variable, this is performed as follows:

namespace : VariableName : TheActualNamespaceAssignment

The Authors File

The AUTHORS.TXT file is where one can add author acknowledgements. Below is an example:

```
CWL Java SDK:

  * Paul Grosu <pgrosu@gmail.com>, Northeastern University

Alternate SDK (via Avro):

  * Denis Yuen <denis.yuen@gmail.com>

CWL Draft:

  * Peter Amstutz <peter.amstutz@curoverse.com>, Curoverse
  * Nebojsa Tijanic <nebojsa.tijanic@sbgenomics.com>, Seven Bridges Genomics

  Contributors:

  * Luka Stojanovic <luka.stojanovic@sbgenomics.com>, Seven Bridges Genomics
  * John Chilton <jmchilton@gmail.com>, Galaxy Project, Pennsylvania State University
  * Michael R. Crusoe <crusoe@ucdavis.edu>, University of California, Davis
  * Herve Menager <herve.menager@gmail.com>, Institut Pasteur
  * Maxim Mikheev <mikhmv@biodatomics.com>, BioDatomics
  * Stian Soiland-Reyes <soiland-reyes@cs.manchester.ac.uk>, University of Manchester
```

Compiling the Programs

To compile the programs with Java, below are the instructions:

```
javac YAMLParser.java
javac CWLSDKBuilder.java
javac CreateCWLJavaSDK.java
```

Running the Programs

To run the program in the most efficient way, just run the *compile-and-run.sh* Bash shell script as follows:

```
./compile-and-run.sh
```

This includes downloading the YAML files, compiling of the Java programs, running them and verification step of compiling the whole SDK.

To run the programs with Java, below are the instructions:

```
java CreateCWLJavaSDK.java cwl-avro.yml metaschema.yml
```

Currently there is a small fix required to the `OutputRecordField` object the draft-3 of the *cwl-avro.yml* file, in order for the SDK to compile successfully for all files. The *OutputRecordField* object is currently defined as follows:

```
- name: OutputRecordField
  type: record
  extends: "sld:RecordSchema"
  docParent: "#OutputParameter"
  specialize:
    - specializeFrom: "sld:RecordSchema"
      specializeTo: "#OutputRecordSchema"
    - specializeFrom: "sld:EnumSchema"
      specializeTo: "#OutputEnumSchema"
    - specializeFrom: "sld:ArraySchema"
      specializeTo: "#OutputArraySchema"
  fields:
    - name: outputBinding
      type: [ "null", "#Binding" ]
      jsonldPredicate: "cwl:outputBinding"
```

The *extends* line would need to be modified to the following:

```
  extends: "sld:RecordField"
```

The *compile-and-run.sh* Bash shell script will perform the rename automatically through the following line:

```
sed "921s/  extends: \"sld:RecordSchema\"/  extends:
\"sld:RecordField\"/" cwl-avro.yml > cwl-avro-fixed-
OutputRecordField.yml
```