Building An Application Using External Processes

Exercise Objectives

In this exercise, you will:

- Add external processing to the example Reactor Application
- Modify the build system to include the external files
- Build, deploy and run the modifications

Adding External Processing to An Application

- To complete the Sentiment Analysis Application, add external processing using a natural language toolkit (NLTK)
- The NLTK is included as part of the Continuuity Reactor SDK examples

Copy from the Sentiment Analysis example in the Continuuity Reactor SDK these directories:

```
/examples/SentimentAnalysis/lib
/examples/SentimentAnalysis/sentiment
```

- Place them in the top level of your `SentimentAnalysis`'s directory
- The `lib` directory contains two archives with natural language toolkit and a supporting data serialization implementation, both written in Python
- The `sentiment` directory contains data for the natural language toolkit

---

The `Unzipper`

Copy from the Sentiment Analysis example in the Continuuity Reactor SDK:

```
/examples/SentimentAnalysis/src/main/java/.../Unzipper.java
```

This file is used to unzip the archives of the `lib` directory

- Place in the matching location in your project
- Update the package statement in the files to reflect its new location

```
package com.example;
```

Update the `pom.xml` (1 of 2)

Update the `pom.xml` so these files get built and included in the Jar

In the `pom.xml` add to the `properties` element a property `archive`:

```
<archive>sentiment-process.zip</archive>
```

To the `dependencies` element, add an additional dependency:

```
<dependency>
    <groupId>org.apache.ant</groupId>
    <artifactId>ant-compress</artifactId>
    <version>1.2</version>
</dependency>
```

Update the `pom.xml` (2 of 2)

To the `build` element, add two plugins to the list of plugins:

```xml
<plugin>
     <artifactId>maven-resources-plugin</artifactId>
     <version>2.6</version>
     <executions>
          ...
     </executions>
</plugin>

<plugin>
     <groupId>org.codehaus.mojo</groupId>
     <artifactId>truezip-maven-plugin</artifactId>
     <version>1.1</version>
     <executions>
          ...
     </executions>
</plugin>
```

Copy the complete `plugins` from the SDK's `SentimentAnalysis pom.xml`

Using the NLTK

To use the NLTK, modify the `Analyze` Flowlet to pass sentences through the NLTK for suffixing with a sentiment

Add these imports:

```
import com.continuuity.flow.flowlet.ExternalProgramFlowlet;
import java.io.File;
import com.continuuity.api.flow.flowlet.FlowletContext;
import java.io.InputStream;
import com.google.common.base.Throwables;
import org.apache.commons.io.FileUtils;
```

Revised `Analyze` Flowlet

```
public static class Analyze extends ExternalProgramFlowlet<String, String> {
  private static final Logger LOG = LoggerFactory.getLogger(Analyze.class);

  @Output("sentiments")
  private OutputEmitter<String> sentiment;

  private File workDir;

  @Override
  protected ExternalProgram init(FlowletContext context) {...}
  @Override
  protected String encode(String input) {...}
  @Override
  protected String processResult(String result) {...}
  @Override
  protected OutputEmitter<String> getOutputEmitter() {...}
  @Override
  protected void finish() {...}

}
```

`init` method

```
@Override
protected ExternalProgram init(FlowletContext context) {
  try {
    InputStream in = this.getClass().getClassLoader()
                         .getResourceAsStream("sentiment-process.zip");
    if (in != null) {
      workDir = new File("work");
      Unzipper.unzip(in, workDir);
      File bash = new File("/bin/bash");
      if (!bash.exists()) {
        bash = new File("/usr/bin/bash");
      }
      if (bash.exists()) {
        File program = new File(workDir, "sentiment/score-sentence");
        return new ExternalProgram(bash, program.getAbsolutePath());
      }
    }
    throw new RuntimeException("Unable to start process");
  } catch (IOException e) {
    throw Throwables.propagate(e);
  }
}
```

`encode` method

```
/**
 * This method will be called for each input event to transform the given input
 * into string before sending to external program for processing.
 *
 * @param input The input event.
 * @return A UTF-8 encoded string of the input, or null if to skip this input.
 */
@Override
protected String encode(String input) {
  return input;
}
```

`processResult` method

```
/**
 * This method will be called when the external program returns the result. Child
 * class can do its own processing in this method or could return an object of type
 * for emitting to next Flowlet with the
 * {@link com.continuuity.api.flow.flowlet.OutputEmitter} returned by
 * {@link #getOutputEmitter()}.
 *
 * @param result The result from the external program.
 * @return The output to emit or {@code null} if nothing to emit.
 */
@Override
protected String processResult(String result) {
  return result;
}
```

`getOutputEmitter` method

```
/**
 * Child class can override this method to return an OutputEmitter for writing data
 * to the next Flowlet.
 *
 */
@Override
protected OutputEmitter<String> getOutputEmitter() {
  return sentiment;
}
```

`finish` method

```
@Override
protected void finish() {
  try {
    LOG.info("Deleting work dir {}", workDir);
    FileUtils.deleteDirectory(workDir);
  } catch (IOException e) {
    LOG.error("Could not delete work dir {}", workDir);
    throw Throwables.propagate(e);
  }
}
```

Build and Deploy

Build the updated project using:

```
mvn clean package
```

Reset the Continuuity Reactor by starting it up (if it is not already running) and using the `Reset` link on the `Overview` tab of the Dashboard

Drag and drop the application jar on the Dashboard

Run Modified Application

Send sentences (without sentiments) using `curl` (each a single line) and watch them run through the Flow system:

```
curl -o /dev/null -sL -w "%{http_code}\\n" -d
  "Continuuity Reactor is awesome"
    http://localhost:10000/v2/streams/sentence

curl -o /dev/null -sL -w "%{http_code}\\n" -d
  "I have hard time building apps on Hadoop"
    http://localhost:10000/v2/streams/sentence

curl -o /dev/null -sL -w "%{http_code}\\n" -d
  "Hadoop is a Big Data platform"
    http://localhost:10000/v2/streams/sentence
```

Exercise Summary

You should now be able to:

- Add external processing to a Reactor Application
- Modify the build system to include external files
- Build, deploy and run the modifications

Exercise Completed

Chapter Index