

# Building Your First Continuity Reactor Application

---

## Exercise Objectives

In this exercise, you will:

- Use the Continuity Reactor `maven` archetype
  - Build a Continuity Reactor Application
  - Deploy and run the application
  - Start an application used for the remaining examples
-

## Exercise Steps: Create the Project

- In a command-line window, create a new, empty directory
- Run the maven archetype command:

```
$ mvn archetype:generate \  
-DarchetypeCatalog=http://tinyurl.com/ndoa5I2 \  
-DarchetypeGroupId=com.continuity \  
-DarchetypeArtifactId=reactor-app-archetype \  
-DarchetypeVersion=2.2.0
```

- For *groupId* use `com.example`
  - For *artifactId* (the project name) use `SentimentAnalysis`
-

## Exercise Steps: Build and Deploy

- Build the resulting project

```
$ cd SentimentAnalysis  
$ mvn clean package
```

- Deploy the resulting application to a Reactor
  - Open the maven project in IntelliJ by opening the `pom.xml` file from within IntelliJ [Extra credit!]
-

## Exercise Steps: Sentiment Analysis

- The `SentimentAnalysis` project will be used to create a complete Reactor application
- A copy of the completed application is included in the Continuuity Reactor SDK's `/examples/SentimentAnalysis` directory
- The completed application performs sentiment analysis of sentences using an external Python natural language toolkit
- The application will be called `SentimentAnalysisApp`

In the `pox.xml` change the `app.main.class` from `WordCountApp` to `SentimentAnalysisApp`:

```
<app.main.class>com.example.SentimentAnalysisApp</app.main.class>
```

Change the name of the source file `WordCountApp.java` to `SentimentAnalysisApp.java`

Change the class inside from `WordCountApp` to `SentimentAnalysisApp`

---

## Exercise Steps: ApplicationSpecification

Replace the existing ApplicationSpecification with:

```
public ApplicationSpecification configure() {  
    return ApplicationSpecification.Builder.with()  
        .setName("SentimentAnalysisApp")  
        .setDescription("Application for Sentiment Analysis")  
        .withStreams()  
            .add(new Stream("sentence"))  
        .noDataSet()  
        .withFlows()  
            .add(new SentimentAnalysisFlow())  
        .noProcedure()  
        .noMapReduce()  
        .noWorkflow()  
        .build();  
}
```

---

## Exercise Steps: ApplicationSpecification

Change its javadoc comment to read:

```
Application that analyzes sentiment of sentences as positive, negative or neutral.
```

Inside `SentimentAnalysisApp`, change the `LOG` variable to set the log as the `SentimentAnalysisApp.class` rather than `WordCount.class`

Remove all other classes in the `SentimentAnalysisApp`

---

## Exercise Steps: Changing the Test File

Change the name of the source file `WordCountTest.java` to `SentimentAnalysisTest.java`

Change the class inside from `WordCountTest` to `SentimentAnalysisTest`

Comment out the contents of the `try` block of the test so the tests build and runs but does nothing; a later exercise will add tests

---



## Exercise Summary

You should now be able to:

- Use the Continuity Reactor `maven` archetype
  - Create a simple Continuity Reactor Application
  - Deploy it in Reactor
  - Open it in an IDE such as IntelliJ
  - Modify a template for a new project
-

# Exercise Completed

[Chapter Index](#)