

# Building An Application Using Streams and Flows

---

## Exercise Objectives

In this exercise, you will:

- Start with the application created by the `maven` archetype
  - Specify a Stream in the application
  - Specify a Flow in the application
  - Build and deploy the application
  - Run the application and use it to ingest data into Reactor
-

## Exercise Steps

- A Stream and Flow were defined as part of the `ApplicationSpecification`
- The Stream was defined and named as `sentence`
- Add required imports
- Implement the Flowlet classes
- Build and deploy

Add these imports:

```
import com.continuity.api.annotation.Batch;
import com.continuity.api.common.Bytes;
import com.continuity.api.flow.flowlet.FlowletSpecification;
import java.util.Iterator;
```

---

# SentimentAnalysisFlow

Add the `SentimentAnalysisFlow` with its three Flowlets:

```
public static class SentimentAnalysisFlow implements Flow {
    @Override
    public FlowSpecification configure() {
        return FlowSpecification.Builder.with()
            .setName("analysis")
            .setDescription("Analysis of text to generate sentiments")
            .withFlowlets()
                .add(new Normalization())
                .add(new Analyze())
                .add(new Update())
            .connect()
                .fromStream("sentence").to(new Normalization())
                .from(new Normalization()).to(new Analyze())
                .from(new Analyze()).to(new Update())
            .build();
    }
}
```

---

# Normalization

Add the Flowlet Normalization:

```
/**
 * Normalizes the sentences.
 */
public static class Normalization extends AbstractFlowlet {
    private OutputEmitter<String> out;

    @ProcessInput
    public void process(StreamEvent event) {
        String text = Bytes.toString(Bytes.toBytes(event.getBody()));
        if (text != null) {
            out.emit(text);
        }
    }
}
```

---

## Flowlet `Analyze`

Add the Flowlet `Analyze`; it currently just passes through its input:

```
/**
 * Analyzes the sentences.
 */
public static class Analyze extends AbstractFlowlet {

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

    @ProcessInput
    public void process(String sentence) {
        sentiment.emit(sentence);
    }
}
```

---

## Flowlet `Update`

Add the Flowlet `Update`; it currently does nothing:

```
public static class Update extends AbstractFlowlet {

    @Override
    public FlowletSpecification configure() {
        return FlowletSpecification.Builder.with()
            .setName("update")
            .setDescription("Updates the sentiment counts")
            .build();
    }

    @Batch(1)
    @ProcessInput("sentiments")
    public void process(Iterator<String> sentimentItr) {
        while (sentimentItr.hasNext()) {
            String text = sentimentItr.next();
        }
    }
}
```

---

## Build and Run

Start Reactor

Build App using `mvn clean package` and deploy by dragging and dropping

Start the `analysis` Flow

Send sentences using a `curl` command (on one line) and watch them run through the Flow system:

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

Note: You can either modify the sentence inside the `curl`, or send the same one repeatedly

---



## Exercise Summary

You should now be able to:

- Start with an application created by the maven archetype
  - Specify a Stream to an application
  - Specify a Flow to an application
  - Implement Flowlets
  - Build and deploy the resulting application
  - Run the application and use it to ingest data into Reactor
-

# Exercise Completed

[Chapter Index](#)