Flows and Flowlets

Module Objectives

In this module, you will learn:

- What is a Flow
- How to create a Flow
- What are Flowlets
- How these appear in the Dashboard
- How to connect Flowlets in a Flow

Flows versus Flowlets

Flows

- Developer-implemented, real-time Stream processors
- Comprised of one or more *Flowlets* that are wired together into a directed acyclic graph or DAG

Flowlets

- Each Flowlet is able to perform custom logic
- Pass DataObjects between themselves
- Can execute data operations for each individual data object processed
- All data operations happen in a consistent and durable manner

---

Flow Processing of Data Objects

When processing a single input object, all operations are executed in a transaction, including:

- The removal of the object from the input, and
- Emission of data to the outputs

This provides **ACID** properties:

- Atomicity,
- Consistency
- Isolation
- Durability

A unique and core property of the Flow system: atomic and "exactly-once" processing of each input object by each Flowlet in the DAG

Deploying A Flow

- Flows are deployed to the Reactor and hosted within containers
- Each Flowlet instance runs in its own container
- Each Flowlet in the DAG can have multiple concurrent instances, each consuming a partition of the Flowlet's inputs

To put data into your Flow, you can either:

- Connect the input of the Flow to a Stream, or
- Implement a Flowlet to generate or pull the data from an external source *(Generative Flowlet)*

Flow Interface

The interface allows you to specify:

- The Flow's metadata
- Flowlets
- Flowlet connections
- Stream to Flowlet connections
- DataSets used in the Flow

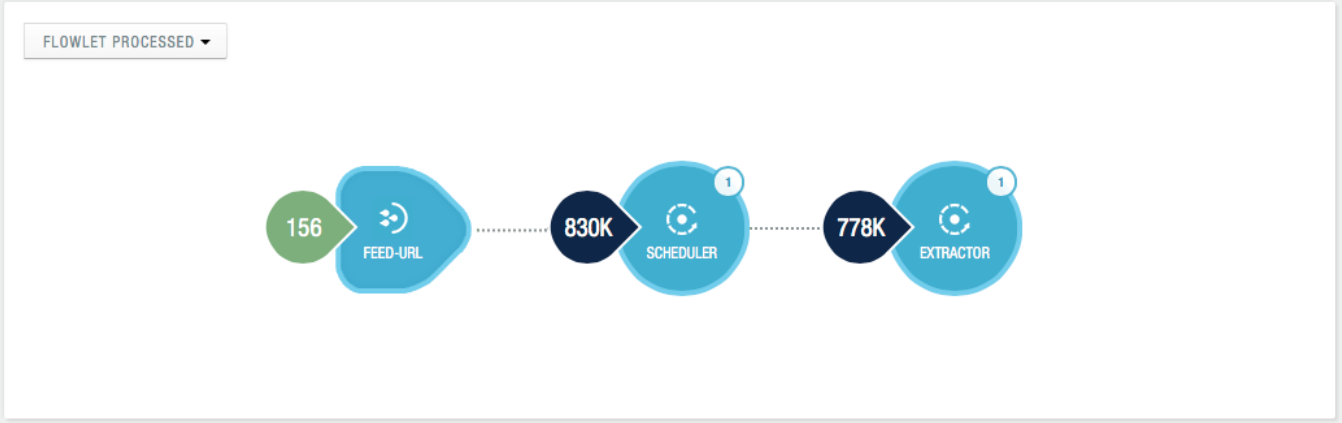
To create a Flow, implement `Flow` via a `configure` method that returns a `FlowSpecification` using `FlowSpecification.Builder()`

In this example, the *name*, *description*, *with* (or *without*) Flowlets, and *connections* are specified before building the Flow.

Flow Example

```
class MyExampleFlow implements Flow {
  @Override
  public FlowSpecification configure() {
    return FlowSpecification.Builder.with()
      .setName("mySampleFlow")
      .setDescription("Flow for showing examples")
      .withFlowlets()
        .add("flowlet1", new MyExampleFlowlet())
        .add("flowlet2", new MyExampleFlowlet2())
      .connect()
        .fromStream("myStream").to("flowlet1")
        .from("flowlet1").to("flowlet2")
      .build();
}
```

Flow Example in Dashboard



Connecting Flowlets in a Flow 1/2

- Two ways to connect the Flowlets of a Flow
- Most common form is to use the Flowlet name

Because the name of each Flowlet defaults to its class name, when building the Flow specification you can simply write:

```
.withFlowlets()
  .add(new RandomGenerator())
  .add(new RoundingFlowlet())
.connect()
  .from("RandomGenerator").to("RoundingFlowlet")
```

Connecting Flowlets in a Flow 2/2

With multiple Flowlets of the same class, give them explicit names:

```
.withFlowlets()
  .add("random", new RandomGenerator())
  .add("generator", new RandomGenerator())
  .add("rounding", new RoundingFlowlet())
.connect()
  .from("random").to("rounding")
```

Module Summary

You should now be able to:

- Implement a Flow
- Describe the difference between Flows and Flowlets
- Find a Flow Specification in an Application
- Relate the Specification to the Dashboard display
- Connect the Flowlets in a Flow

Module Completed