Runtime Arguments and Scaling Instances

Module Objectives

In this module, you will learn about:

- Runtime arguments
- Using runtime arguments
- Scaling Reactor instances
- Getting and setting the number of instances

Runtime Arguments

Flows, Procedures, MapReduce and Workflows can receive runtime arguments

- For Flows and Procedures, runtime arguments are available to the initialize method in the context
- For MapReduce jobs, runtime arguments are available to the beforeSubmit and onFinish methods in the context
- The beforeSubmit method can pass them to the Mappers and Reducers through the job configuration
- When a Workflow receives runtime arguments, it passes them to each MapReduce job in the Workflow

Runtime Argument Example (1 of 2)

• The initialize()	method in this examp	le accepts a runtime arg	nument for the Hellowor	ld Procedure
--------------------	----------------------	--------------------------	-------------------------	--------------

It ch	anges the	greeting	from a	default	"Hello"	to a	customized	"Good	Morning'
-------------------------	-----------	----------	--------	---------	---------	------	------------	-------	----------

Runtime Argument Example (2 of 2)

```
public static class Greeting extends AbstractProcedure {
 @UseDataSet("whom")
 KeyValueTable whom;
 private String greeting;
 public void initialize(ProcedureContext context) {
    Map<String, String> args = context.getRuntimeArguments();
    greeting = args.get("greeting");
   if (greeting == null) {
      greeting = "Hello";
 @Handle("greet")
 public void greet(ProcedureRequest request,
                    ProcedureResponder responder) throws Exception {
   byte[] name = whom.read(NameSaver.NAME);
   String toGreet = name != null ? new String(name) : "World";
    responder.sendJson(greeting + " " + toGreet + "!");
}
```

Scaling Instances: Flowlets

You can query and set the number of instances executing a given Flowlet by using the instances parameter with HTTP GET and PUT methods:

```
GET /v2/apps/<app-id>/flows/<flow-id>/flowlets/<flowlet-id>/instances
PUT /v2/apps/<app-id>/flows/<flow-id>/flowlets/<flowlet-id>/instances
```

with the arguments as a JSON string in the body:

```
{ "instances" : <quantity> }
```

Where:

<ahref="right"><app-id>:: Name of the application
<flow-id>: Name of the Flow
<flowlet-id>: Name of the Flowlet

<quantity>: Number of instances to be used

Scaling Instances: Flowlets Examples

1. Find out the number of instances of the Flowlet *saver* in the Flow *WhoFlow* of the application *HelloWorld*:

GET /v2/apps/HelloWorld/flows/WhoFlow/flowlets/saver/instances

2. Change the number of instances of the Flowlet saver in the Flow WhoFlow of the application HelloWorld:

PUT /v2/apps/HelloWorld/flows/WhoFlow/flowlets/saver/instances

with the arguments as a JSON string in the body:

```
{ "instances" : 2 }
```

Scaling Instances: Procedures

In a similar way to *Scaling Flowlets*, you can query or change the number of instances of a Procedure by using the instances parameter with HTTP GET and PUT methods:

```
GET /v2/apps/<app-id>/procedures/<procedure-id>/instances
PUT /v2/apps/<app-id>/procedures/<procedure-id>/instances
```

with the arguments as a JSON string in the body:

```
{ "instances" : <quantity> }
```

Where:

<quantity>: Number of instances to be used

Scaling Instances: Procedures Examples

1. Find out the number of instances of the Procedure *saver* in the Flow *WhoFlow* of the application *HelloWorld*:

GET /v2/apps/HelloWorld/flows/WhoFlow/procedure/saver/instances

2. Change the number of instances of the Procedure saver in the Flow WhoFlow of the application HelloWorld:

PUT /v2/apps/HelloWorld/flows/WhoFlow/procedure/saver/instances

with the arguments as a JSON string in the body:

```
{ "instances" : 2 }
```

Module Summary

You should be able to:

- Use runtime arguments in your applications
- Get and set the number of Reactor instances for different elements

Module Completed