CONTINUUITY REACTOR GETTING STARTED GUIDE

Version 2.0.0

Continuuity, Inc.

Table of Contents

1. Getting Started with the Local Reactor

In this section you'll build, deploy, and run one or more of the provided sample apps using the Local Reactor and the Reactor dashboard. Then you push an app to the Sandbox Reactor, a 30-day free account on the Continuuity cloud.

Don't forget to take a look at the *Continuuity Reactor Developer Guide* located in the docs directory in your Reactor installation directory and the Continuuity Reactor Javadocs located in the javadocs directory.

1.1. Prerequisites

You'll need Java™ and Node.js™.

The Reactor example apps are pre-compiled, but if you want to modify and compile an app, you'll also need Apache Ant installed on your system as discussed below.

OS

You'll need Linux or Mac OS X.

Java

The latest version of the JDK or JRE version 6 must be installed in your environment. Only Java6 is currently supported.

- Click here to download the Java Runtime for Linux and Solaris.
- On Mac OS X, the JVM is bundled with the operating system.
- Set the JAVA_HOME environment variable after installing Java.

Node.js

This section discusses installing Node.js on MAC and RHEL systems.

Note: The version of Node.js must be v0.8.16 or greater.

MAC OS

You can download the latest version of Node.js from http://nodejs.org using any of the methods they suggest.

RHEL

For RHEL-based operating systems, consider installing Node.js using RPM:

```
$ wget http://mirrors.xmission.com/fedora/epel/6/i386/epel-release-6-8.noarch.rpm
$ rpm -i epel-release-6-8.noarch.rpm
$ yum install npm
```

Apache Ant

The example apps are pre-compiled. You'll only need Apache Ant if you want to modify and then compile an app.

You can get the latest version of Apache Ant from http://ant.apache.org.

Note: For MAC OS, there is no need for further configuration after the install.

1.2. Unpack the Reactor Development Kit

The Continuuity Reactor Development Kit is bundled as a ZIP file and contains everything you need to build and run Big Data applications on your local machine. Copy the ZIP file into your home directory and unzip it. On Mac OS:

```
// For Mac OS, from a terminal prompt, change to your home directory:
$ cd ~
// Copy the ZIP file into your home directory. Note the dot at the end of the command.
$ cp Downloads/continuuity-Reactor-development-kit-2.0.0.zip .     // note the "." at the end
// In the Finder, double-click the ZIP file in your home directory to unzip it.
```

The ZIP file includes the documentation, various JAR files, tools, and example applications:

```
README                                                    (a file you should read)
LICENSES/                 (the Continuuity Reactor license and open source licenses)
VERSION                                          (the version number of this release)
continuuity-api-2.0.0.jar                                   (the API JAR for Reactor)
continuuity-api-2.0.0-javadoc.jar              (the JavaDoc JAR for the Reactor API)
continuuity-api-2.0.0-source.jar                (the source JAR for the Reactor API)
lib/continuuity-test-2.0.0.jar and other libs       (the Reactor Test Framework JAR)
bin/continuuity-reactor                                    (Local Reactor Daemon)
bin/data-client                                      (Command-Line Dataset Client)
bin/data-format                                           (Local Data Format Tool)
bin/Reactor-client    (Command-Line Tools for deploying, deleting, and managing apps)
bin/stream-client                                      (Command-Line Stream Client)
data/                                                (Directory with metrics items)
docs/       (Directory for Release Notes, Getting Started Guide, and Developer Guide)
conf/continuuity-site.xml                              (Local Reactor Configuration)
conf/logback.xml                                    (Local Reactor Log Configuration)
examples/CountAndFilterWords                         (Sample Word Filter Application)
examples/CountCounts                                (Sample Number Counter Application)
examples/CountOddAndEven                                (Sample Odd/Even Number App)
examples/CountRandom                               (Sample Random Number Generator App)
examples/CountTokens                                    (Sample String Counting App)
examples/HelloWorld                                        (Sample Hello World App)
examples/Purchase                      (Sample Purchase History MapReduce Application)
examples/SimpleWriteAndRead                                 (Sample Dataset Using App)
examples/Ticker                                  (Import and Query Stock Trade Data)
examples/WordCount                                   (Sample Word Count Application)
examples/ant-common.xml                 (Common scripts used by all of the examples)
examples/build.xml                             (Ant build.xml for building examples)
javadocs/                                       (Directory for Reactor API Javadocs)
lib/                                                                  (Lots of JARs)
logs/                                                            (Directory for logs)
```

1.3. Build the Example Applications

The example applications are pre-built. However, if you experiment with them by changing the code you can rebuild them using Apache Ant.

Building the example applications is simple using Ant. You can get the latest version of Ant from http://ant.apache.org.

```
$ cd ~/continuuity-Reactor-development-kit-2.0.0/examples
$ ant
```

This will generate a JAR file for each of the sample applications. You can also individually build a single example:

```
$ cd ~/ continuuity-Reactor-development-kit-2.0.0/examples/WordCount
$ ant
```

1.4. Start the Local Reactor

Start the Local Reactor:

```
$ cd ~/continuuity-reactor-development-kit-2.0.0/
$ ./bin/continuuity-reactor start
```

Your Local Reactor is now running.

To run Reactor in debug mode so that you can connect to an IDE remote debugger, start it with this command:

```
$ ./bin/continuuity-Reactor start --enable-debug 5005
```

For more information see the Debugging Reactor Applications section on page 9.

You can check the status of the Local Reactor, stop it, or restart it:

```
$ ./bin/continuuity-reactor status
$ ./bin/continuuity-reactor stop
$ ./bin/continuuity-reactor restart
```

1.5. Open the Local Reactor in Your Browser

With your Local Reactor started from the command line, navigate to the URL for the Local Reactor Dashboard that is displayed on your screen. The Dashboard for the Local Reactor is also accessible via http://localhost:9999.

Notes:

- This version of the Reactor requires a Chrome v30+, Safari v7+, or Firefox v25+ browser.
- If your browser is configured to not accept cookies, you may have to create an exception for http://localhost so that your Local Reactor will work properly.

1.6. Deploy and Run Applications Using the Reactor Dashboard

Now that the Local Reactor instance is running on localhost and you have accessed your local Reactor Dashboard, you can easily deploy and run one or more of the bundled sample applications. For example, deploy the `WordCount` application by drag-and-dropping the `WordCount.jar` file onto the Reactor Dashboard.

The WordCount.jar file is located in your ~/continuuity-reactor-development-kit-2.0.0/examples/WordCount directory.

The WordCount application is now deployed and verified and the app's name appears in the Apps section.

Start the app's flow and procedure:

1. Click the `WordCount` application in the Apps section.
2. All of the elements of this application display: a stream in the Collect section, a flow in the Process section, four datasets in the Store section, and a procedure in the Query section.
3. Click the `WordCounter` flow in the Process section to open the flow visualization.
4. Click START in the top-right corner to start the flow.

The flow is running, so it can process incoming data from the stream. To send a sample event to the flow:

1. Click the WORDSTREAM stream icon at the left of the flow illustration.
2. Type a string of words in the textbox and press Enter or click INJECT. For example, type "It's a beautiful day." (without the quotes), then close the dialog.
3. Watch the event get processed by the flowlets in the Directed Acyclic Graph (DAG) in the Processes and Busyness graphs.
4. Note the numbers indicating the activity in each flowlet.
5. Close the wordStream dialog.

Note: You can also send events to a stream via the HTTP or the command-line APIs (see the *Continuuity Reactor Developer Guide* in the docs directory in your Reactor installation directory).

We've just processed some text data from the stream via the flow, now we'll use a procedure to read the results of the processing:

1. Click Query.
2. Click the `RetrieveCounts` procedure.
3. Click START in the top-right of the RetrieveCounts screen to run the procedure.

The procedure is now running and is ready to accept new requests. Procedures bind to REST interfaces, so it's easy to query them using any HTTP-based tools or libraries (see the *Continuuity Reactor Developer Guide*). You can also use the procedure dashboard to send HTTP requests and receive the responses directly in the dashboard.

1. Type `getCount` in the METHOD text box.
2. Type the JSON string `{"word":"<a word in your stream>"}` in the PARAMETERS text box, for example type {"word":"beautiful"}
3. Click EXECUTE.
4. The results of your query are displayed in the dashboard in JSON format:

   {"assocs":{"Its":1,"a":1,"day":1},"count":1,"word":"beautiful"}

1.7. Push to Cloud

After you develop and test your application in the Local Reactor, you can promote it to a 30-day free remote instance of the Reactor in the cloud – the Sandbox Reactor. The Sandbox Reactor has the same functionality as your Local Reactor. It allows you to quickly and easily deploy to a real Hadoop cluster without having to pay a fee to deploy to a Continuuity Hosted Reactor or a Continuuity Enterprise Reactor.

You can create your Sandbox Reactor at https://accounts.continuuity.com if you haven't already done so. At that time you will receive an **API Token** that authenticates you with the Sandbox Reactor. You'll need this API token to promote an application to the cloud.

1. Click Overview to return to the Dashboard.
2. Click WordCount in the Applications section.
3. Click PUSH in the top-right of the screen.
4. Enter your API key and your Sandbox Reactor will appear in the dialog. To find your API key, click the Profile Page link in the Push to Cloud dialog and supply your login information. In the Profile tab of the My Account page, click Show Key, enter your password into the API Key field, and click Show Key.
5. Copy your API Key, paste it into the API Key field in the Push to Cloud dialog, then click PUSH to promote your app to your Sandbox Reactor.
6. The success confirmation message displays the link for your app in your Sandbox Reactor. Click the link to go to your app in your Sandbox Reactor.

Note: If the links in the Sandbox Reactor don't work, try reloading the page. This is especially true if you have a lot of apps, browsers, and/or browser tabs open.

2. Reactor Maven Archetype

Maven is a very popular Java build and dependencies management tool for creating and managing a Java application projects. A simple way to start is to use a Maven archetype to generate a skeleton for the Java project.

This Maven archetype generates a Reactor application Java project with the proper dependencies and sample code as a base to start writing your own Big Data application. To generate a new project, execute the following command:

```
$ mvn archetype:generate \
  -DarchetypeCatalog=https://repository.continuuity.com/content/groups/releases/archetype-catalog.xml \
  -DarchetypeGroupId=com.continuuity \
  -DarchetypeArtifactId=Reactor-app-archetype \
  -DarchetypeVersion=2.0.0
```

In the interactive shell that appears, specify some basic properties for the new project. For example, to create a new project called MyFirstBigDataApp:

```
Define value for property 'groupId': : org.myorg
Define value for property 'artifactId': : MyFirstBigDataApp
Define value for property 'version':  1.0-SNAPSHOT: :
Define value for property 'package':  org.myorg: :
Confirm properties configuration:
groupId: org.myorg
artifactId: MyFirstBigDataApp
version: 1.0-SNAPSHOT
package: org.myorg
 Y: : Y
```

After you confirm the settings, the directory MyFirstBigDataApp is created under the current directory.

To build the project:

```
$ cd MyFirstBigDataApp
$ mvn clean package
```

This creates MyFirstBigDataApp-1.0-SNAPSHOT.jar in the target directory. This JAR file is a skeleton Reactor application that is ready to deploy to the Reactor. Just drag and drop it anywhere on the Reactor Dashboard.

3. Debugging Reactor Applications

Any Reactor application can be debugged in the Local Reactor by attaching a remote debugger to the Reactor JVM.

To enable remote debugging, start the Local Reactor with the --enable-debug option specifying port 5005:

```
$ cd ~/continuuity-Reactor-development-kit-2.0.0/
$ ./bin/continuuity-Reactor start –-enable-debug 5005
```

The Reactor confirms that the debugger port is open:

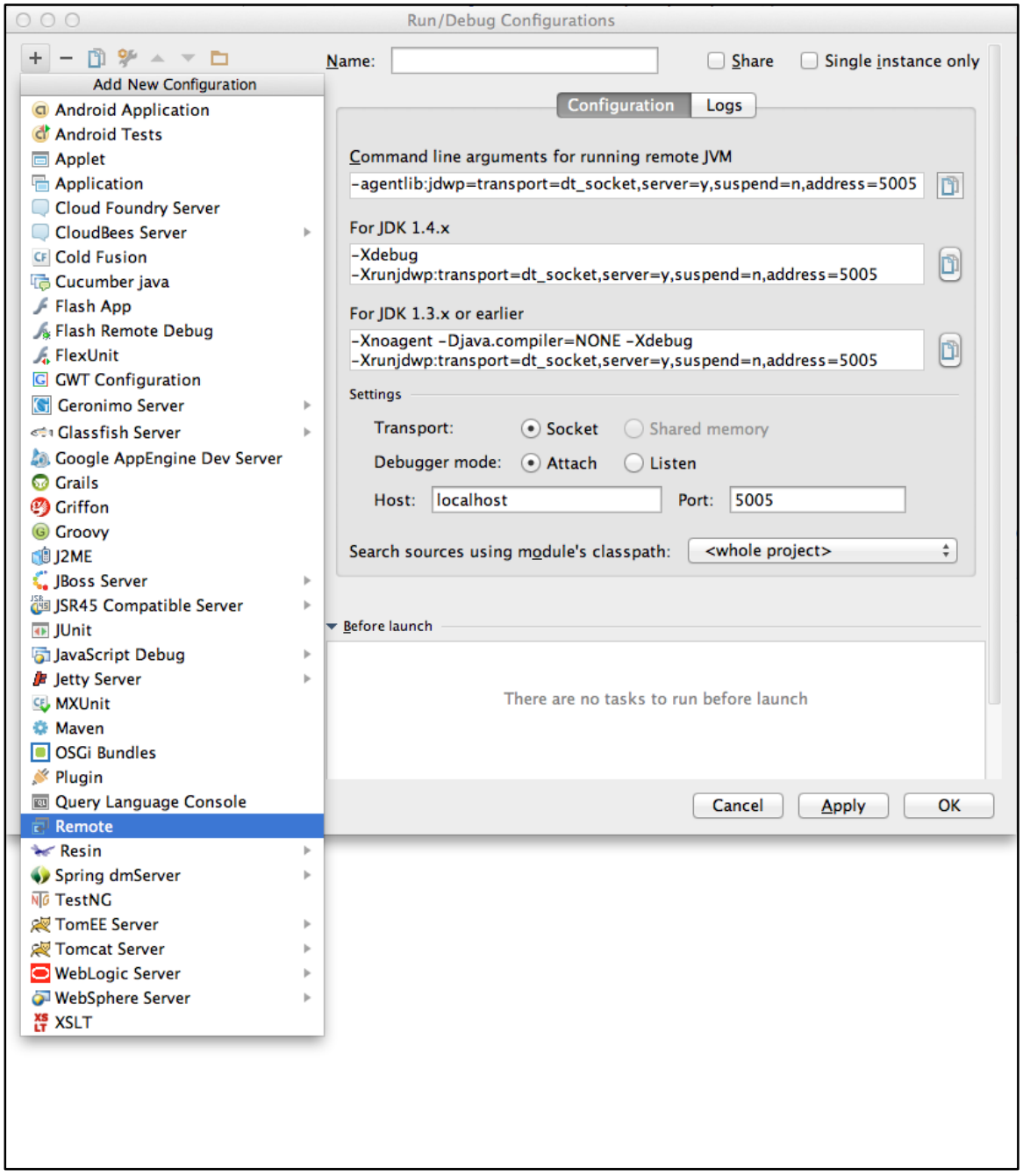
"*Remote debugger agent started on port 5005*"

1. Deploy the HelloWorld application to the Reactor by dragging and dropping the HelloWorld.jar file from the /examples/HelloWorld directory onto the Reactor Dashboard.

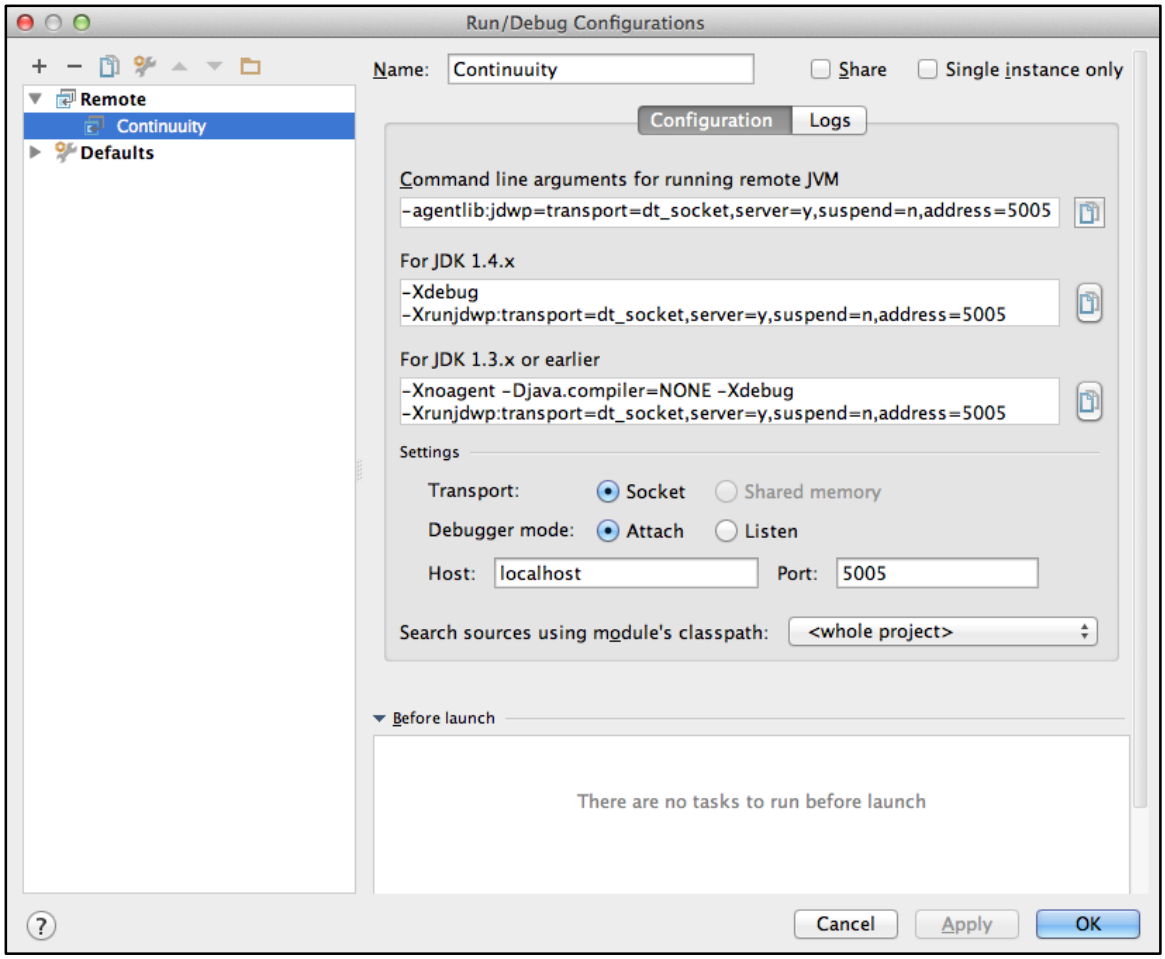
2. Open the HelloWorld application in an IDE and connect to the remote debugger. For more information, see Debugging with IntelliJ on page 10 or Debugging with Eclipse on page 12.

3.1. Debugging with IntelliJ

1. From the IntelliJ toolbar, select Run -> Edit Configurations.
2. Click + and choose Remote Configuration:



3. Create a debug configuration by entering a name, for example, Continuuity.
4. Enter 5005 in the Port field:



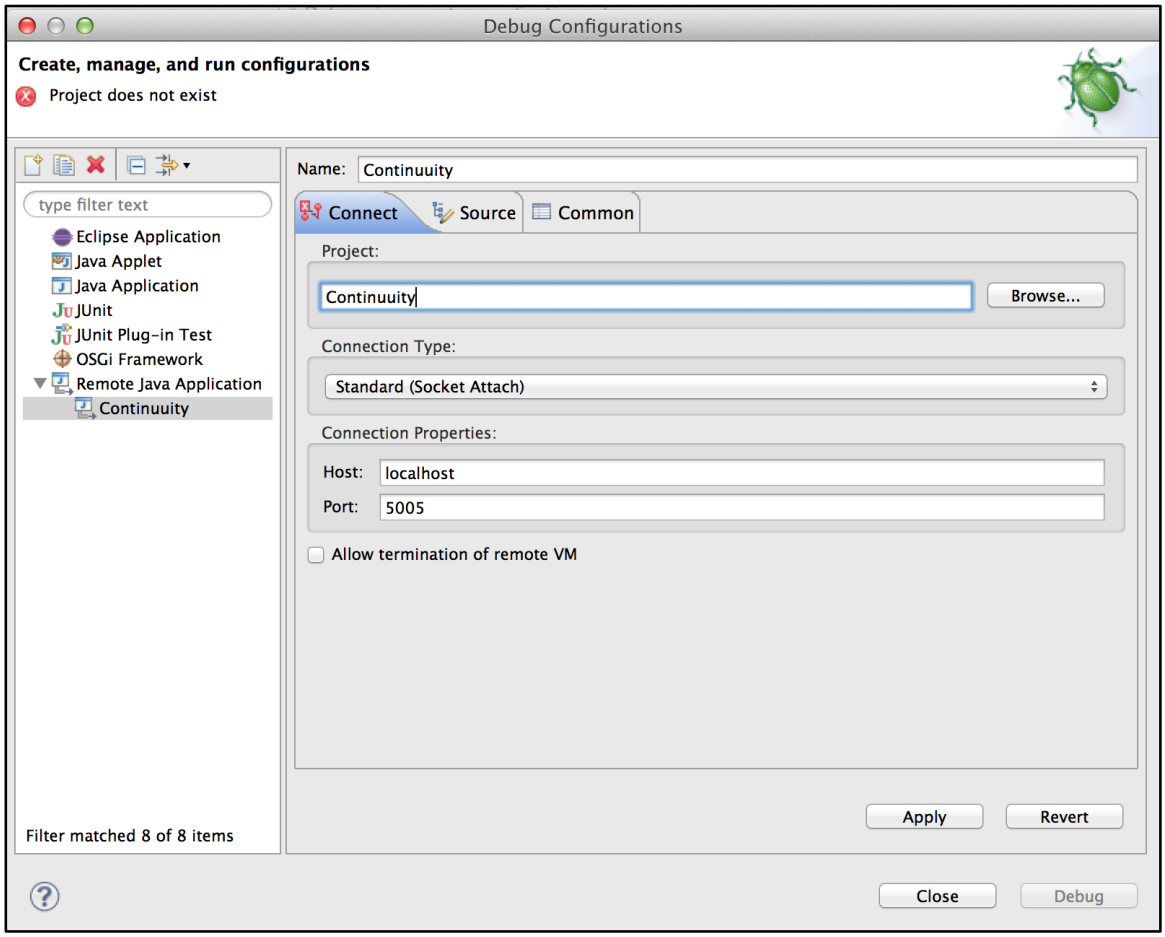
5. To start the debugger, select Run -> Debug -> Continuuity.
6. Set a breakpoint in any code block, for example, a flowlet method:

```
71      @ProcessInput
72      public void process(StreamEvent event) {
73        byte[] name = Bytes.toBytes(event.getBody());
74        if (name != null && name.length > 0) {
75          whom.write(NAME, name);
76        }
77        if (name.length > 10) {
78          flowletMetrics.count("names.longnames", 1);
79        }
80        flowletMetrics.count("names.bytes", name.length);
81      }
82    }
```

7. Start the flow in the Dashboard.
8. Send an event to the stream. The control will stop at the breakpoint and you can proceed with debugging.

3.2. Debugging with Eclipse

1. In Eclipse, select Run-> Debug configurations.
2. In the pop-up, select Remote Java application.
3. Enter a name, for example, Continuuity.
4. In the Port field, enter 5005.
5. Click Debug to start the debugger:



6. Set a breakpoint in any code block, for example, a flowlet method:

```
    public void process(StreamEvent event) {
        byte[] name = Bytes.toBytes(event.getBody());
        if (name != null && name.length > 0) {
          whom.write(NAME, name);
        }
        if (name.length > 10) {
```

7. Start the flow in the Dashboard.
8. Send an event to the stream.
9. The control stops at the breakpoint and you can proceed with debugging.

4. Next Steps

For a more in-depth understanding of how to develop Big Data applications with the Continuuity Reactor, see the *Continuuity Reactor Developer Guide* in the /docs directory in your Reactor installation directory.

5. Technical Support

If you need any help from us along the way, you can reach us at http://support.continuuity.com.