CONTINUUITY REACTOR™ GETTING STARTED GUIDE

VERSION 1.6.0 BETA

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 and then push the apps to the Reactor Sandbox, a 90-day free account on the Continuuity cloud.

Also see the Eclipse IDE Plugin section on page 8 of this Continuuity Reactor Getting Started Guide to learn how to create, debug, run and deploy an app using the Continuuity Reactor Eclipse Plugin. And don't forget to take a look at the Continuuity Reactor Developer Guide bundled with this release in your installation directory.

1.1. Prerequisites

You'll need Java™, Node.js™, and 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.

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

Node.js

You can download the latest version of Node.js from http://nodejs.org using any of the methods given. For RHEL-based operating systems we suggest 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

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

1.2. Unpack the Developer Suite

The Continuuity Developer Suite 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:

```
// For Mac OS, from a terminal prompt, change to your home directory:
> cd ~
// Copy the ZIP file into your home directory
> cp Downloads/continuuity-test-1.6.0.zip .
// 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, the Eclipse IDE plugin, and sample applications:

```
README                                                       (a file you should read)
LICENSE                                                 (the Developer Suite license)
VERSION                                           (the version number of this release)
Continuuity Reactor Release Notes 1.6.0 Beta      (Continuuity Reactor Release Notes)
Continuuity Reactor Getting Started Guide 1.6.0 Beta      (Getting Started Tutorials)
Continuuity Reactor Developer Guide 1.6.0 Beta       (Reference Guide for Developers)
continuuity-api-1.6.0.jar                                    (the API JAR for Reactor)
continuuity-api-1.6.0-javadoc.jar                (the JavaDoc JAR for the Reactor API)
continuuity-api-1.6.0-source.jar                  (the source JAR for the Reactor API)
continuuity-eclipse-plugin-1.6.0.jar                      (the Reactor Eclipse Plugin)
lib/continuuity-test-1.6.0.jar                       (the Reactor Test Framework JAR)
bin/continuuity-reactor                                         (Local Reactor Daemon)
bin/reactor-client                                      (Command-Line Reactor Client)
bin/data-client                                         (Command-Line Dataset Client)
bin/data-format                                              (Local Data Format Tool)
bin/meta-client    (Future feature for discovering app contents; not yet implemented)
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/api                                                                (API JavaDocs)
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/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)
lib                                                                     (Lots of JARs)
logs                                                               (Directory for logs)
web-app                                                 (Reactor Dashboard application)
```

1.3. Build the Example Applications

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

```
> cd ~/continuuity-developer-edition-1.6.0/examples
> ant
```

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

```
> cd ~/continuuity-developer-edition-1.6.0/examples/WordCount
> ant
```

1.4. Start the Local Reactor

To start the local Reactor, run the `continuuity-reactor` daemon with the start command:

```
> cd ~/continuuity-developer-edition-1.6.0/
> ./bin/continuuity-reactor start
```

Your local Reactor is now running. The URL for the local Reactor Dashboard is displayed on your screen. It's also accessible via http://localhost:9999.

Note: If your browser is set to not accept cookies, you may have to create an exception for http://localhost so that your local Reactor will work properly.

You can check the status of the local Reactor or stop it:

```
> ./bin/continuuity-reactor status
> ./bin/continuuity-reactor stop
```

1.5. Deploy and Run Applications Using the Reactor Dashboard

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

1. Click ADD AN APP to open the New Application dialog.
2. Drag-and-drop your `WordCount.jar` onto the center of the Add an Application dialog.
3. Your application is now uploaded, deployed and verified and the WordCount app's name appears in the Applications section.

Start the app's flow and procedure:

1. Click the `WordCount` application in the Applications 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 run the flow.

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

1. Click WORDSTREAM at the left of the flow illustration.
2. Type a string of words in the top-right textbox and press Enter or click INJECT.
3. Watch the event get processed by the flowlets in the DAG in the Processing Rate and Busyness graphs.
4. Close the wordStream dialog.

Note: You can also send events to a stream via REST or the command-line (see the Continuuity Reactor Developer Guide in your 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 in your installation directory). You can also use the procedure dashboard to send REST requests and receive the response 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 the value, type a word that you sent to the stream without the angle brackets).
3. Click EXECUTE.
4. The results of your query are displayed in the dashboard in JSON format.

1.6. Push to Cloud

After you develop and test your application in the local Reactor, you can promote it to a 90-day free remote instance of the Reactor in the cloud – the Sandbox Reactor. The Sandbox Reactor allows you to quickly and easily deploy to a real Hadoop cluster without paying 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. Click the link to go to your app in your Sandbox Reactor. The Sandbox Reactor has the same functionality as your local Reactor.

Note: If the links in the Sandbox 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. Eclipse IDE Plugin

This section walks you through creating, running, and debugging a simple application using the Continuuity Eclipse IDE plugin.

2.1. Prerequisites

OS

Linux or Mac OS X.

Java

The latest version of the JDK or JRE version 6.xx must be installed in your environment.

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

Node.js

You can download the latest version of Node.js from http://nodejs.org using any of the methods given. For RHEL-based operating systems we suggest 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
```

Eclipse

Eclipse 4.2 (Juno) (available from http://eclipse.org/eclipse4/)

Continuuity Developer Suite

1. Download the Continuuity Developer Suite ZIP file from https://accounts.continuuity.com/ and unzip it in your home directory (double-click the ZIP file in the Finder on Mac OS).

For Mac OS, move the ZIP file to your home directory:

```
$ cd ~
$ cp Downloads/continuuity-developer-suite-1.6.0.zip .
// Double-click the continuuity-developer-suite-1.6.0.zip file in the Finder
```

Continuuity Eclipse Plugin

The Continuuity Eclipse Plugin is a JAR file packaged with the Developer Suite:

continuuity-eclipse-plugin-1.6.0.jar

2.2. Get Eclipse, Set Up the Plugin and Start Eclipse

1. Download and unpack the Eclipse IDE.
2. To install the Continuuity Eclipse plugin, copy the plugin JAR file into the “plugins” subdirectory of the unpacked Eclipse directory. For example:

```
$ cp ~/continuuity-developer-suite-1.6.0/continuuity-eclipse-plugin-1.6.0.jar $ECLIPSE_HOME/eclipse/plugins/
```

3. Start Eclipse.

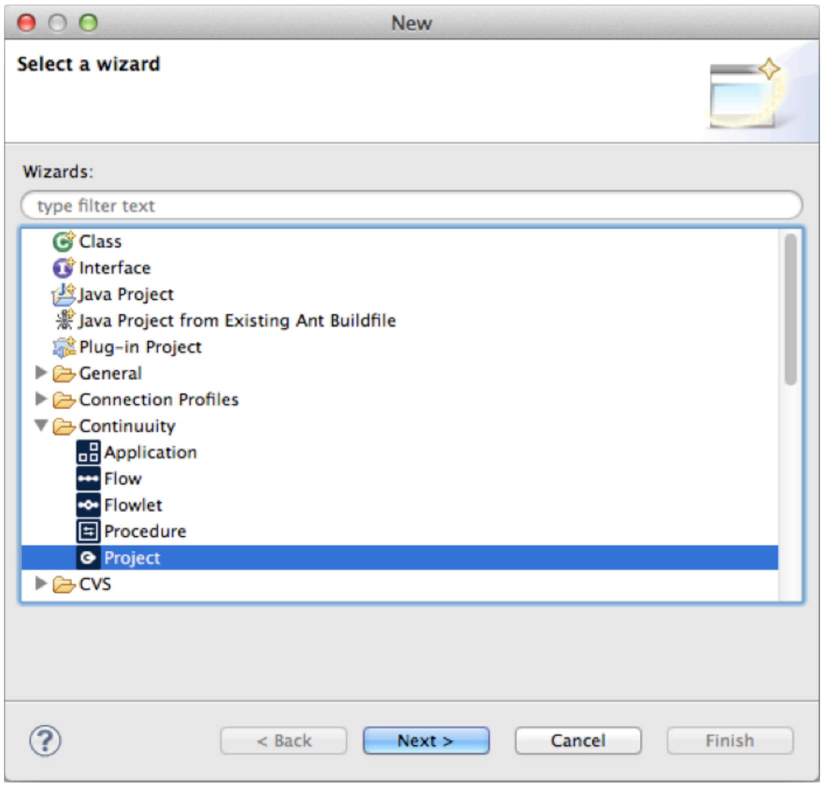
2.3. Create a Simple Application

This section guides you through creating a simple application from scratch in the Eclipse IDE.

Create a New Continuuity Project

1. Create a new project:

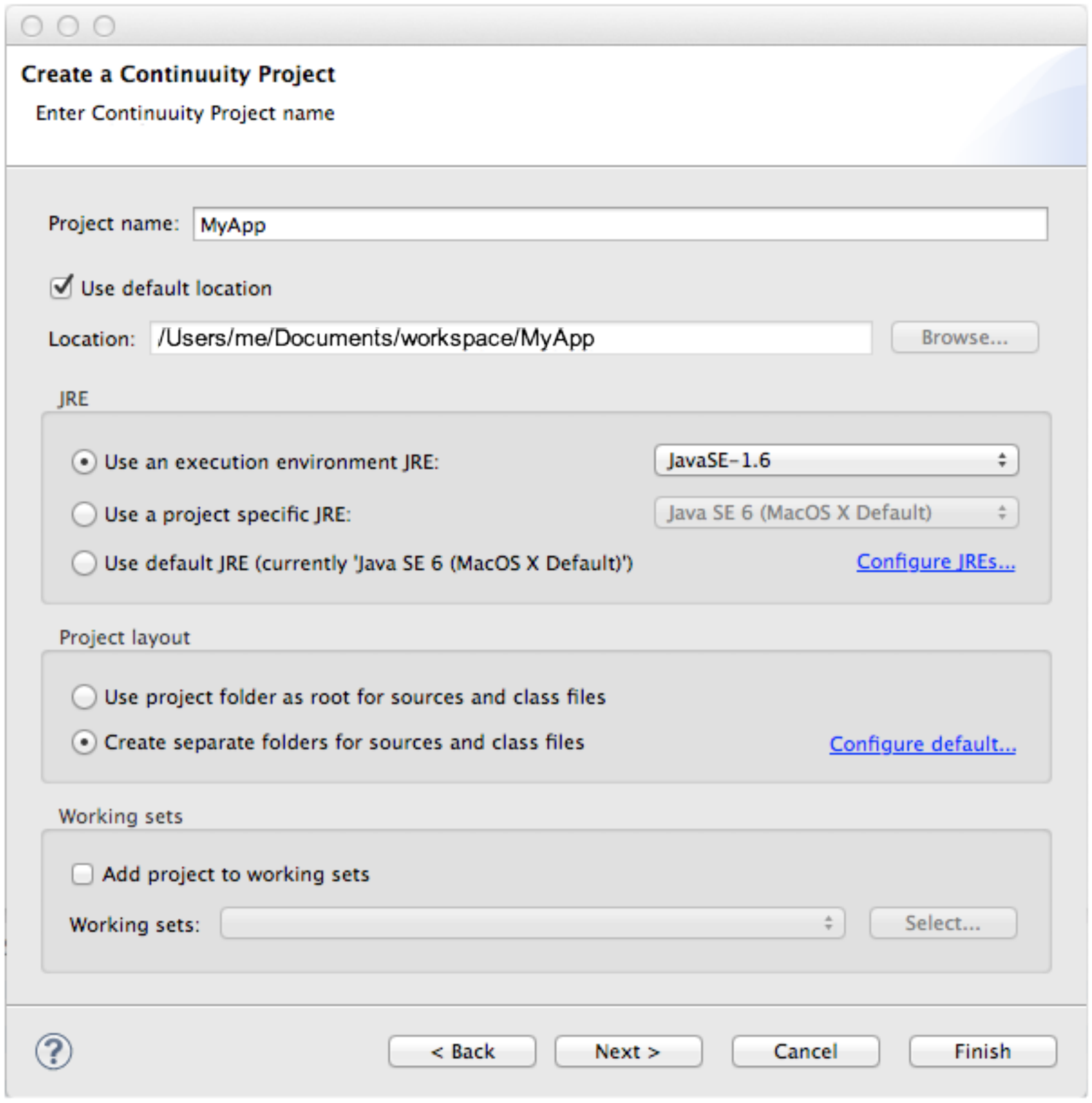
   File > New > Other... > Continuuity > Project

   NOTE: if you don't see the "Continuuity" menu item in this dialog, the plugin was not installed correctly. Please review the installation and setup instructions above.

2. Click Next.

3. Type MyApp in the Project Name field, then click Next.



Create a Continuuity Project

Enter Continuuity Project name

Project name: MyApp

[✓] Use default location

Location: /Users/me/Documents/workspace/MyApp

JRE

- (●) Use an execution environment JRE: JavaSE-1.6
- ( ) Use a project specific JRE: Java SE 6 (MacOS X Default)
- ( ) Use default JRE (currently 'Java SE 6 (MacOS X Default)')

Project layout

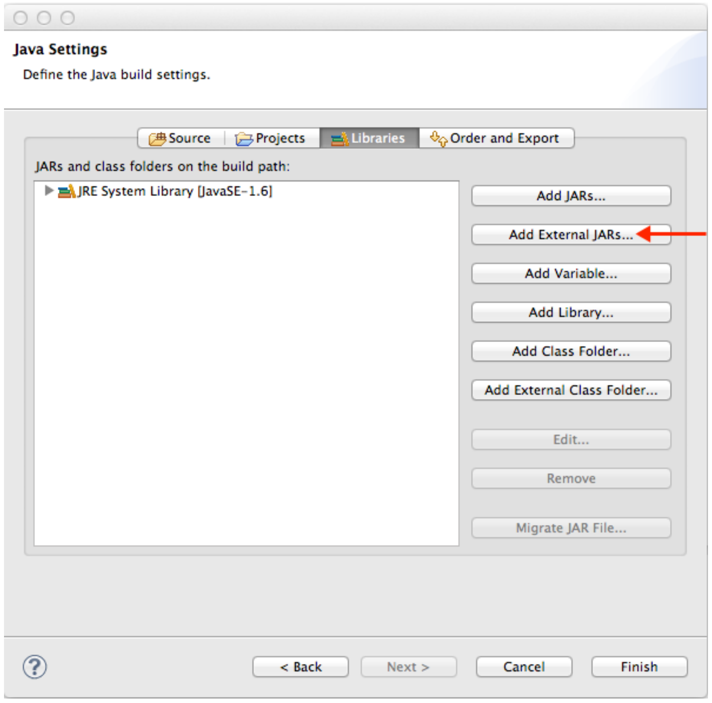
- ( ) Use project folder as root for sources and class files
- (●) Create separate folders for sources and class files

Working sets

[ ] Add project to working sets

Working sets:

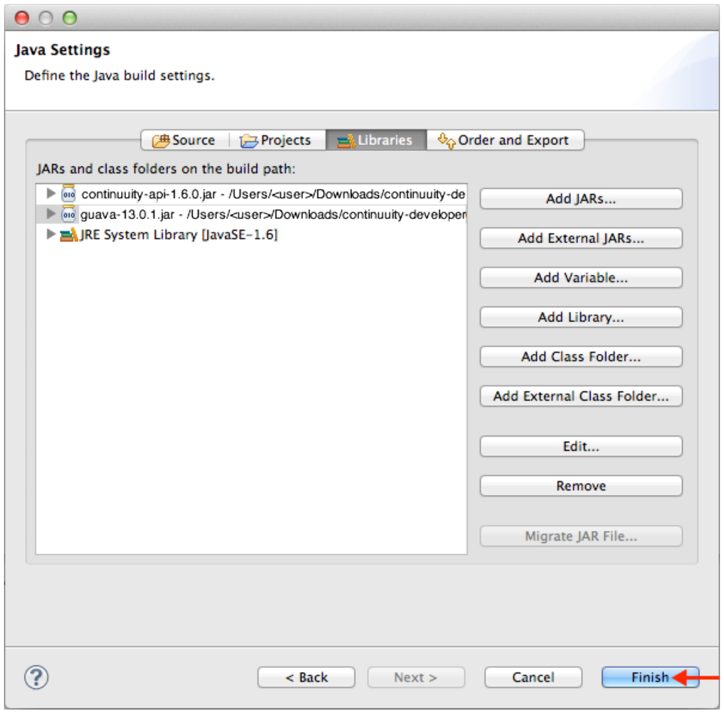
4. Click Next, click Libraries in the top menu bar, then click Add External JARS...



5. Select the continuuity-api-1.6.0.jar file located in your <home>/continuuity-developer-edition-1.6.0 directory, then click Open.

6. Click Add External JARS..., navigate to your <home>/continuuity-developer-edition-1.6.0/lib directory, select the guava-13.0.1.jar file, then click Open.

7. Click Finish.


Java Settings
Define the Java build settings.
Source
Projects
Libraries
Order and Export
JARs and class folders on the build path:
continuuity-api-1.6.0.jar - /Users/<user>/Downloads/continuuity-de
guava-13.0.1.jar - /Users/<user>/Downloads/continuuity-developer
JRE System Library [JavaSE-1.6]
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
Edit...
Remove
Migrate JAR File...
< Back
Next >
Cancel
Finish

2.4. Create a Flow

The MyApp application contains a simple Flow called MyFlow. MyFlow contains a single flowlet called MyFlowlet that consumes data from a stream called inStream.

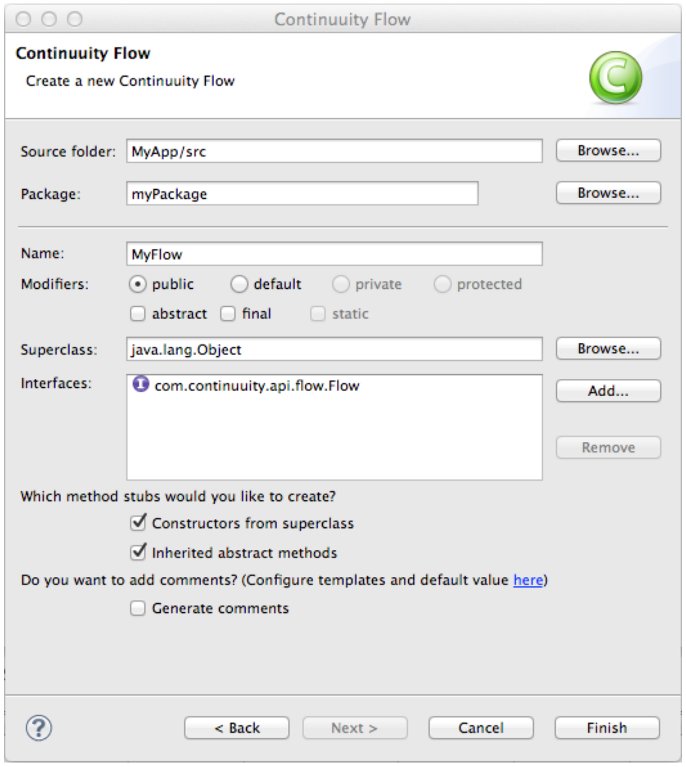
Add the Flow and Flowlet Classes

Create a Flow class stub:

1. In the Eclipse File menu, navigate to:

   File > New > Other... > Continuuity > Flow

2. Click Next.
3. In the Continuuity Flow dialog, use the Source Folder's Browse button to select the MyApp/src folder.
4. Type myPackage in the Package field.
5. Type MyFlow in the Name field.
6. Click Finish.



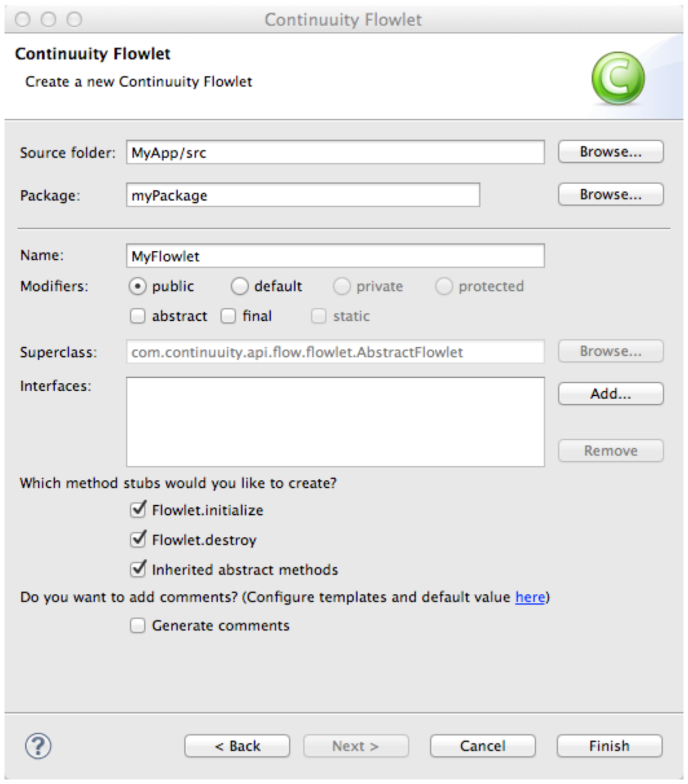
Source folder: MyApp/src
Package: myPackage
Name: MyFlow
Modifiers: [✓ public] default private protected
abstract final static
Superclass: java.lang.Object
Interfaces: com.continuuity.api.flow.Flow
Which method stubs would you like to create?
[✓ Constructors from superclass]
[✓ Inherited abstract methods]
Do you want to add comments? (Configure templates and default value here)
Generate comments

Create a Flowlet class stub:

1. In the Eclipse File menu, navigate to:

   New > Other... > Continuuity > Flowlet

2. Click Next.
3. Type MyFlowlet in the Name field.
4. Click Finish.



Source folder: MyApp/src

Package: myPackage

Name: MyFlowlet

Modifiers: [✓ public] default private protected

abstract final static

Superclass: com.continuuity.api.flow.flowlet.AbstractFlowlet

Interfaces:

Which method stubs would you like to create?

✓ Flowlet.initialize

✓ Flowlet.destroy

✓ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

Generate comments

Configure the Flow

The flow consists of one Flowlet that consumes data from a Stream and outputs the data to the output console. First, we need to configure the Flow:

1. In MyFlow.java, change the *configure()* method to:

```
@Override
public FlowSpecification configure() {
return FlowSpecification.Builder.with()
       .setName("MyFlow")
       .setDescription("MyFlow Description")
       .withFlowlets().add(new MyFlowlet())
       .connect().fromStream("inStream").to("MyFlowlet")
       .build();
}
```

The configure() method defines the identity of the flow and how the stream and the flowlet are interconnected.

- MyFlow is the name of the flow.
- MyFlowlet is the name of the flowlet
- inStream is the name of the stream that is consumed by the flowlet.
- The stream connects to the flowlet via the connect().fromStream().to() methods.

2. Save MyFlow.java.

The final MyFlow.java code:

```
Package myPackage;

import com.continuuity.api.flow.Flow;
import com.continuuity.api.flow.FlowSpecification;

public class MyFlow implements Flow {
   @Override
   public FlowSpecification configure() {
       return FlowSpecification.Builder.with()
       .setName("MyFlow")
           .setDescription("Flow description")
           .withFlowlets().add(new MyFlowlet())
           .connect().fromStream("inStream").to("MyFlowlet")
           .build();
   }
}
```

Configure the Flowlet and Add Processing Logic

The MyFlowlet flowlet consumes incoming data from a stream and outputs the data to the output console.

1. Add the following import statements to MyFlowlet.java:

```
import com.continuuity.api.common.Bytes;
import com.continuuity.api.data.OperationException;
```

2. Change the configure() method to:

```
    @Override
    public FlowletSpecification configure() {
        return FlowletSpecification.Builder.with()
            .setName("MyFlowlet")
            .setDescription("flowlet description")
            .build();
    }
```

3. Change the process() method to:

```
    public void process(StreamEvent event) {
        System.out.println("Input value: " +
            new String(Bytes.toBytes(event.getBody())));
    }
```

4. Save MyFlowlet.java. The final MyFlowlet.java code:

```
Package myPackage;

import com.continuuity.api.annotation.*;
import com.continuuity.api.common.Bytes;
import com.continuuity.api.data.OperationException;
import com.continuuity.api.flow.flowlet.*;

public class MyFlowlet extends AbstractFlowlet {

    @Override
    public void initialize(FlowletContext context) throws FlowletException {
        // TODO Auto-generated method stub
    }

    @Override
    public FlowletSpecification configure() {
        return FlowletSpecification.Builder.with()
            .setName("MyFlowlet")
            .setDescription("flowlet description")
            .build();
    }

    @Override
    public void destroy() {
        // TODO Auto-generated method stub
    }
}
    public void process(StreamEvent event) throws OperationException {
        System.out.println("Input value: " +
            new String(Bytes.toBytes(event.getBody())));
    }
}
```

2.5. Create the Application

Finally, create an application that ties everything together.

1. From the Eclipse menu, navigate to:

   File > New > Other... > Continuuity > Application

2. Click Next.
3. In the Continuuity Application dialog, verify that the package name is myPackage and the class name is MyApp.
4. Click Finish.

Configure the Application

1. Add the following import statement to MyApp.java:

```
import com.continuuity.api.data.stream.Stream;
```

2. Change the configure() method in MyApp.java to:

```
   @Override
   public ApplicationSpecification configure() {
      return ApplicationSpecification.Builder.with()
          .setName("MyApp")
          .setDescription("MyApp Description")
          .withStreams().add(new Stream("inStream"))
          .noDataSet()
          .withFlows().add(new MyFlow())
          .noProcedure()
          .noBatch()
          .build();
   }
```

3. Save MyApp.java.

The final MyApp.java code:

```
package myPackage;

import com.continuuity.api.*;
import com.continuuity.api.data.stream.Stream;

public class MyApp implements Application {

    public MyApp() {
        // TODO Auto-generated constructor stub
    }

    @Override
    public ApplicationSpecification configure() {
        return  ApplicationSpecification.Builder.with()
            .setName("MyApp")
            .setDescription("MyApp Description")
            .withStreams().add(new Stream("inStream"))
            .noDataSet()
            .withFlows().add(new MyFlow())
            .noProcedure()
            .noBatch()
            .build();
    }
}
```

Debug the Application

This section discusses debugging MyApp.

Prepare for Debugging

Notes: If the Reactor is running, stop it before continuing:

```
$ cd ~/continuuity-developer-suite-1.6.0
> continuuity-developer-suite-1.6.0 $ ./bin/continuuity-reactor stop
```

In some environments, like on Mac OS, Eclipse may override your Node configuration, which typically has a path of /usr/local/bin. To protect against this, you may need to log in as sudo and set up the following symlink from your 1.6.0 home directory:

```
> continuuity-developer-suite-1.6.0 $ sudo ln -svf /usr/local/bin/node /usr/bin/node
```

Next, kill all running Node.js processes:

```
> continuuity-developer-suite-1.6.0 $ killall -9 node
```

Set a Breakpoint

You can set breakpoints anywhere in your application code before debugging. For example, it may be useful to inspect the execution of the MyFlowlet.process() method:

1. In MyFlowlet.java, click to insert the cursor at the beginning of first line in the process() method (the line that begins with "System.out.println").
2. Click Run > Toggle Breakpoint from the Eclipse main menu to insert the breakpoint.

Create the Launch Configuration

1. Right-click MyApp in the Project Explorer (usually in the left panel). In the popup menu click:

    Debug As > Debug Configurations…

2. Right-click Local Reactor and click New.

3. Type MyApp for the Name of the launch configuration.

4. In the Reactor tab, set the Reactor Home Directory to the developer suite's home directory, typically:

/Users/<user>/Downloads/continuuity-developer-suite-1.6.0

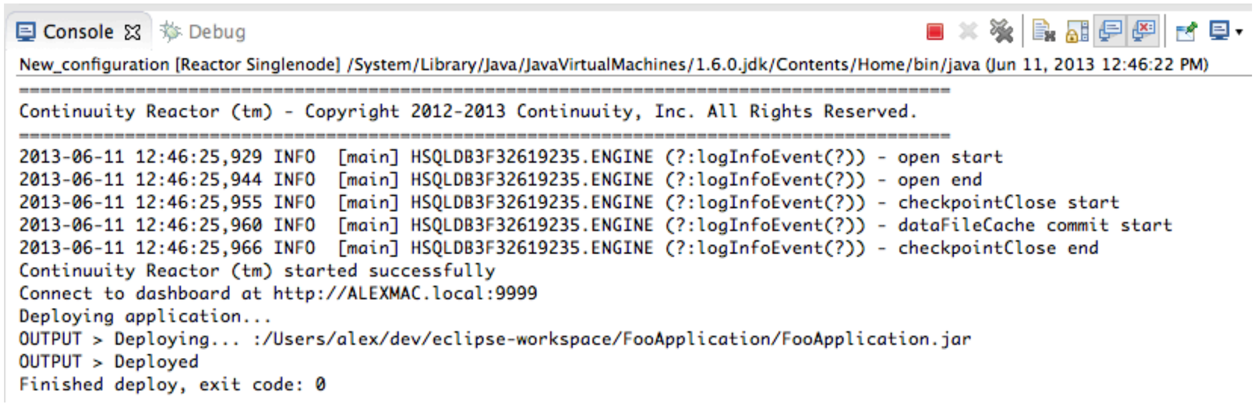
5. Specify myPackage.MyApp for the Main Application class.

START DEBUGGING

Click Debug to start the debug session and you will see the output of the launched Reactor in the Eclipse console. Wait for the Reactor to start and for the application to deploy.

If prompted to open the debug perspective, choose No unless you are familiar with it.

Note that the Eclipse output console indicates "Deployed", and the last message in the console should show "Finished deploy, exit code: 0".



```
New_configuration [Reactor Singlenode] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 11, 2013 12:46:22 PM)
===================================================================================
Continuuity Reactor (tm) - Copyright 2012-2013 Continuuity, Inc. All Rights Reserved.
===================================================================================
2013-06-11 12:46:25,929 INFO  [main] HSQLDB3F32619235.ENGINE (?:logInfoEvent(?)) - open start
2013-06-11 12:46:25,944 INFO  [main] HSQLDB3F32619235.ENGINE (?:logInfoEvent(?)) - open end
2013-06-11 12:46:25,955 INFO  [main] HSQLDB3F32619235.ENGINE (?:logInfoEvent(?)) - checkpointClose start
2013-06-11 12:46:25,960 INFO  [main] HSQLDB3F32619235.ENGINE (?:logInfoEvent(?)) - dataFileCache commit start
2013-06-11 12:46:25,966 INFO  [main] HSQLDB3F32619235.ENGINE (?:logInfoEvent(?)) - checkpointClose end
Continuuity Reactor (tm) started successfully
Connect to dashboard at http://ALEXMAC.local:9999
Deploying application...
OUTPUT > Deploying... :/Users/alex/dev/eclipse-workspace/FooApplication/FooApplication.jar
OUTPUT > Deployed
Finished deploy, exit code: 0
```

DEBUG TROUBLESHOOTING

1. If the debugger hangs with error messages indicating that Eclipse can't find a class loader, check the Java processes in a terminal window. If you see multiple StartSingleNodeAndDeployApplicationHelper messages you'll have to kill them and run debug again:

```
$ jps
3825 StartSingleNodeAndDeployApplicationHelper
3831 StartSingleNodeAndDeployApplicationHelper
3832 StartSingleNodeAndDeployApplicationHelper
3833 Jps
3586
$ kill 3825 3831 3832
$ jps
3834 Jps
3586
```

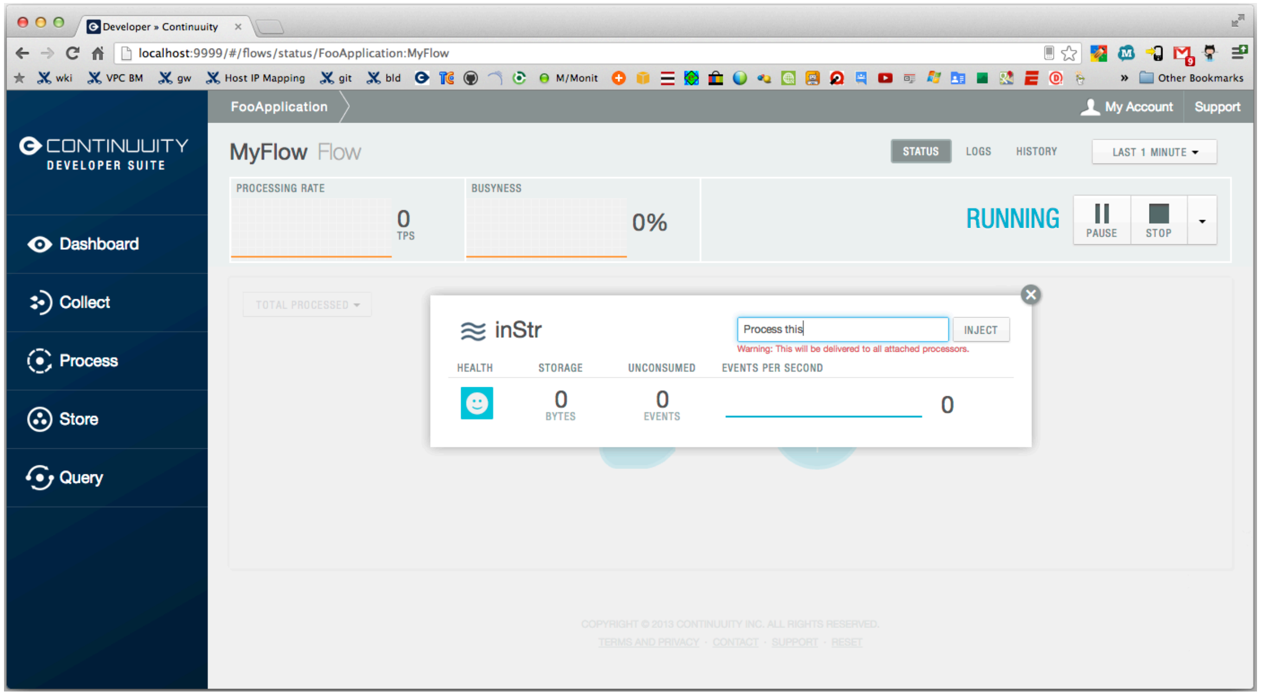
2. Eclipse also may get confused and throw error messages when upgrading to new a new release of the Reactor. If this occurs, go back to Debug Configurations (right-click on MyApp in the Project Explorer, click Debug As > Debug Configurations…, delete the MyApp Debug Configuration (click the red X above the left panel), and recreate the Debug Configuration as you did the first time.

3. You can ignore any messages like:

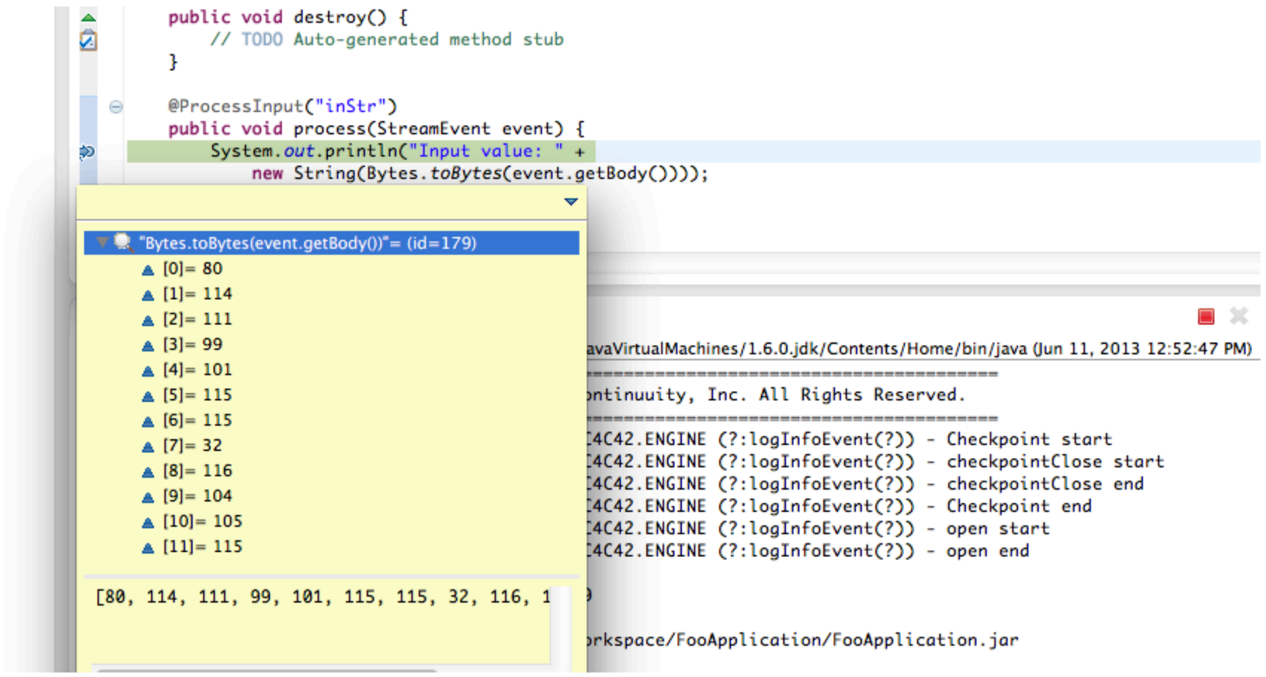
2013-08-05 14:10:31.716 java[5510:1203] Unable to load realm info from SCDynamicStore
2013-08-05 14:10:31.899 java[5510:1203] Unable to load realm info from SCDynamicStore

Work with the Application in the Reactor Dashboard

1. Go to "http://localhost:9999" in your web browser.
2. Click Process in the left panel.
3. Click MyFlow, and to start the Flow click START.
4. To inject data into the stream, click the INSTREAM icon, enter some text in the text field, then click "INJECT".



Eclipse will "catch" a breakpoint, for example in the MyFlowlet.process() method if you inserted one there:



```
    public void destroy() {
        // TODO Auto-generated method stub
    }

    @ProcessInput("inStr")
    public void process(StreamEvent event) {
        System.out.println("Input value: " +
            new String(Bytes.toBytes(event.getBody())));
```

```
"Bytes.toBytes(event.getBody())"= (id=179)
    [0]= 80
    [1]= 114
    [2]= 111
    [3]= 99
    [4]= 101
    [5]= 115
    [6]= 115
    [7]= 32
    [8]= 116
    [9]= 104
    [10]= 105
    [11]= 115

[80, 114, 111, 99, 101, 115, 115, 32, 116, 1
```

```
avaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 11, 2013 12:52:47 PM)
=========================================
ontinuuity, Inc. All Rights Reserved.
=========================================
4C42.ENGINE (?:logInfoEvent(?)) - Checkpoint start
4C42.ENGINE (?:logInfoEvent(?)) - checkpointClose start
4C42.ENGINE (?:logInfoEvent(?)) - checkpointClose end
4C42.ENGINE (?:logInfoEvent(?)) - Checkpoint end
4C42.ENGINE (?:logInfoEvent(?)) - open start
4C42.ENGINE (?:logInfoEvent(?)) - open end

orkspace/FooApplication/FooApplication.jar
```

Stop Debugging

After you're finished with your debug session , or to stop it and restart it, click the Terminate icon (the solid red square in the Eclipse Debug Console) to end the Debug session and stop the local Reactor process.

Debug Troubleshooting

If you don't click the Terminate icon as discussed above you may see an error message similar to:

```
ERROR [main:c.c.d.e.l.LevelDBOVCTable@888] - LevelDB exception on create(error
code = IO error: lock /Users/<user>/Downloads/continuuity-developer-suite-
1.6.0/data/ldb_REACTOR.MetricsTable.agg/LOCK: Resource temporarily unavailable)
```

If this happens, click the Terminate icon and try again. If the Terminate icon doesn't appear, kill all running StartSingleNodeAndDeployApplicationHelper processes from a terminal window:

```
$ jps
4403 Jps
4346 StartSingleNodeAndDeployApplicationHelper
3586
kill 4346
```

Run the Application

Running an application is similar to debugging it:

1. Right-click MyApp in the Eclipse Project Explorer.
2. Click Run As > Run Configurations…
3. Click Run.

Troubleshooting Running the Application

Note: You can ignore any messages like:

```
2013-08-05 14:10:31.716 java[5510:1203] Unable to load realm info from SCDynamicStore
2013-08-05 14:10:31.899 java[5510:1203] Unable to load realm info from SCDynamicStore
```

Stop the Application

After you run your app, or to stop it and restart it, click the Terminate icon (the solid red square in the Eclipse Debug Console) to stop the app and stop the local Reactor process.

Trougleshooting Stopping the Application

1. If you don't click the Terminate icon as discussed above you will see an error message similar to:

```
ERROR [main:c.c.d.e.l.LevelDBOVCTable@888] - LevelDB exception on create(error
code = IO error: lock /Users/<user>/Downloads/continuuity-developer-suite-
1.6.0/data/ldb_REACTOR.MetricsTable.agg/LOCK: Resource temporarily unavailable)
org.fusesource.leveldbjni.internal.NativeDB$DBException: IO error: lock
```

If this happens, click the Terminate icon and try again. If the Terminate icon doesn't appear, kill all running StartSingleNodeAndDeployApplicationHelper processes from a terminal window:

```
$ jps
4403 Jps
4346 StartSingleNodeAndDeployApplicationHelper
3586
kill 4346
```

Build a Deployable JAR File Using Ant

To build a deployable JAR file using Ant:

1. Right-click MyApp in the Eclipse Project Explorer.
2. Click Export…
3. Double-click General.
4. Click Ant Buildfiles, then click Next.
5. Click the MyApp checkbox.
6. Ensure that the checkbox for “Create target to compile project using Eclipse compiler” is selected.
7. Click Finish.
8. Open the build.xml file that was added to your MyApp project in the Eclipse Project Explorer.
9. Before the closing `</project>` element add the following code:

```
<manifest file="MANIFEST.MF">
    <attribute name="Manifest-Version" value="1.0" />
    <attribute name="Main-Class" value="myPackage.MyApp" />
</manifest>
<jar destfile="MyApp.jar" manifest="MANIFEST.MF">
    <fileset dir="bin"><include name="**"/></fileset>
</jar>
```

   Note: Don’t change the format of the fileset element shown in the code above. If you insert tabs or spaces between the angle brackets in the fileset element, instead of BUILD SUCCESSFUL you may get an error message similar to:

```
BUILD FAILED
build.xml:58: The <fileset> type doesn't support nested text data ("? ?").
```

10. Save build.xml. Note: You have to repeat this process every time you change your MyApp code.
11. Run ant build from the command line from the root folder of your project. For example, if you chose the default Eclipse working directory, depending on your platform that would be something like:

```
$ cd ~/Documents/workspace/MyApp/
$ ant build
```

12. Ant creates a new MyApp.jar that overwrites the default MyApp.jar. You can push the new MyApp.jar file to your Reactor Sandbox via the Reactor Dashboard’s Push to Cloud function (see Push to Cloud on page 7) or deploy it on your Hadoop cluster.

3. 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.

4. Technical Support

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

5. Glossary

DAG	Directed Acyclic Graph