Quick Start: Using an Aspire Distribution

On this page

For more information on Aspire Distributions, see Using the Maven Distribution Archetype.

The previous quick-start used a pre-built Aspire distribution which was fixed for a certain set of components.

In this quick-start, you'll learn how to build an Aspire distribution from scratch, using the Aspire Maven Distribution archetype. Also, all of the components will be automatically downloaded from Maven repositories - making system configuration and deployment a breeze.

Prerequisites

Before you begin, you need to be registered to use Aspire (go to http://aspire.searchtechnologies.com/) if you haven't already done that.

You will need your user registration name and password in order to complete this tutorial.

1: Download and Install Java and Set JAVA_HOME

The version of Java you should use depends on the Aspire version you are targeting to:

- Aspire 2.2 and up requires to run at Java 1.7

We recommend installing the Java JDK (Java Development Kit), just in case you want to create your own Aspire components/applications in the future. Only the JRE (Java Runtime Environment) is absolutely required.

1. Download and install the latest version of the Java 1.6 or 1.7 JDK as aprropiate: http://java.com /en/download/manual.jsp blocked URL
   - If you have a 64 bit machine, we recommend installing the 64 bit version of Java. That will allow you to create large-memory instances of Aspire. blocked URL
     - The Aspire framework itself does not use up that much memory (100mb or so). But some applications may store big hash tables to improve performance, so it's best to have the 64 bit JVM (Java Virtual Machine), just in case you need it someday.
2. Test that you can access the "java" command from your console.
   a. Open up a new DOS command-shell (go to the start menu, and enter "cmd" where it says "Run" or "Search for Programs", and then execute the cmd.exe program).
   b. Enter: `java -version`

Add the "JAVA_HOME" environment variable. blocked URL

- This is needed by the Maven command-line tool described below.

1. Open up your control panel
2. Go to the "System" control panel (may be inside the 'System and Security' category)
3. Open up the "Advanced System Settings"
4. Go to the "Advanced" tab (may already be selected)
5. Click on "Environment Variables"
6. Click on "New..."
7. Enter "JAVA_HOME" as the "Variable name:"
8. Locate the directory within your Program files where java was installed. Enter this directory name as the "Variable value:"

2: Download the Maven Command-Line Tool

Aspire is tightly integrated with Apache Maven, more so than any other software tool. New distributions are built using a Maven archetype and Aspire will automatically download components and application bundles stored in Maven repositories.

1. Download and install the latest version of Apache Maven: http://maven.apache.org/download.html
    a. Download the compressed binary files.
    b. Suggestion: Create a "C:\dev" directory and unpack the files there.
        - "C:\dev" is a good place to store open source tools like Apache Maven, since (on rare occasions) they will not work with path names that have spaces in them.
2. Add the M2_HOME and M2 environment variables in your System Properties and modify your Path variable:

    1. Please refer to Step 1 above for the instructions on accessing your Environment Variables via the Control Panel.

    2. Add the M2_HOME system variable:

        i. In the System Variables section, select ***New***.
        ii. Enter the following in the Variable name field: `M2_HOME`
        iii. Locate the directory within your Program files where you unpacked Maven, enter this directory name as the Variable value (for example, `C:\dev\apache-maven-2.1.0`), then click ***OK***.

    3. Add the M2 system variable: blocked URL

        a. In the System Variables section, again select ***New***.
        b. Enter the following in the Variable name field: `M2`
        c. In the Variable value field, enter the following, then click ***OK***: `%M2_HOME%\bin`

    4. Modify your Path variable:

        a. Scroll through the System Variables to locate the Path variable, select it, then click on the Edit button.
        b. Click in the "Variable Value" field to activate it, then press your End key to move your cursor to the end of the value.
        c. Without typing any spaces, enter the the bin directory from Apache Maven to your environment path (for example, `;C:\dev\apache-maven-3.0.4\bin`), then click ***OK*** .

    5. Click on the OK button to close the Environment Variables window, then click ***OK*** to close the System Properties window.

3. Test that you can access Maven. blocked URL

    1. Open up a new DOS command-shell (go to the Start menu, enter "cmd" in the "Run" or "Search for Programs" field, and then execute the cmd.exe program).
    2. At the prompt, enter the following, then press the Enter key: `mvn -version`
    Success is indicated when version information is returned, as shown in the image to the right.

3: Connect up to the Maven Repository

Now that you have Maven and Java (yay!), you will need to configure Maven so that it understands where the Maven Repository is located. This is the location on the internet where all of the Aspire components are located. The URL is http://repository.searchtechnologies.com/artifactory/simple/community-public/. It is password protected.

Note: You will need to use your registered user name and password to access the Maven repository. Go to Registration for more information.

Once Maven understands where the ST repository is located, Maven can automatically download Aspire and all of its pieces to create new Aspire distributions with a single command.

1. Create the ".m2" directory in your user-name's home directory (if it does not already exist) blocked URL

    - This is known as the "Maven Local Repository" blocked URL
    - This is the directory where Maven will store all of the components which it downloads on your local machine.
        - Windows XP: C:\Documents and Settings\{username}\.m2
        - Windows 7: C:\Users\{username}\.m2

      NOTE: Windows 7 may prevent you from creating this directory. It will say "You must type a file name."
        - If this happens, you will need to create the directory using command-line tools:
    - Open up a new DOS command-shell (go to Start, enter `cmd`"in the **Run** or **Search for Programs** field, and then execute the **cmd.exe** program).
        - If necessary, use `cd` to change directory to your user's home directory (the C:\Users\{username} directory).
    - Enter the following, then press the Enter key: `mkdir .m2`

2. Create the settings.xml file inside your new ".m2" directory

    - You will need to register to get a username and password for accessing the repository.

1. A default settings.xml file is installed with Maven and can be found in the conf directory file (for example, C:\dev\apache-maven-3.0.4); locate it and copy the file.
2. Navigate to your .m2 directory and paste the file.
3. Open the file in any editing software.
4. Go to Connect to the Maven Repository and copy the settings.xml content on the page (starting with <?xml and ending with </settings>) and paste it into your settings.xml file.
5. Replace "REGISTERED-USERNAME" and "REGISTERED-PASSWORD" with your registration data.
6. Save the changes and close the file.
    - These steps will create a settings.xml file that will only apply to the current user on the machine. To apply the settings to all users who will log onto the machine, please refer to Connect to the Maven Repository.

Step 1: Use Maven to Create a New Aspire Distribution

More detailed documentation for creating new distributions is here: Use The Maven Distribution Archetype

Now that Maven is installed, creating a new Aspire distribution is relatively easy. Our recommendations are to:

1. Open up a DOS command window (see step 1 for instructions).
2. Use "cd" to change the directory to someplace where you want to store the new Aspire distributions.
3. Execute the command shown below.

```
 mvn archetype:generate -DarchetypeGroupId=com.searchtechnologies.aspire -DarchetypeArtifactId=aspire-distribution-archetype -DarchetypeVersion=3.2 -DrepositoryId=stPublic
```

The command should be executed exactly as is. Do not modify the parameters (they specify where to locate the instructions for how to build a new Aspire Distribution).

Note that, sometimes, the "mvn" command will fail to locate the Search Technologies repository (it should be specified in your new settings.xml file). If this is a problem, add the following argument to the "mvn" command from above:

```
-DarchetypeRepository=http://repository.searchtechnologies.com/artifactory/simple/community-public/
```

After executing the command, Maven will download a bunch of different JAR files. Most of these are various Maven plug-ins which Maven itself requires to execute. These plug-ins are downloaded automatically from the "master maven repository in the sky", namely http://repo.maven.apache.org.

Finally, once Maven itself is ready to go, you will be prompted for the following information:

- **groupId** - This is the name of the organizational unit to which your distribution belongs. It's usually the same as your organization's domain, but in reverse order (for example, "com.searchtechnologies.aspire"). If the distribution belongs to a customer, you should use that customer's groupId.
    - The purpose of the groupId is to distinguish this distribution from all other maven distributions built by other organizations throughout the world (just in case somebody somewhere wants to re-use your distribution someday).
- **artifactId** - This is the name for your distribution. Technically, it should be unique name across all distributions (or any other maven project) within the organization specified by groupId. Again, it's purpose is to uniquely identify your distribution within your organization.
- **version** - This will be the version number for this distribution.
    - I usually use "0.1-SNAPSHOT".
    - I say "0.1" to not be presumptuous. In other words, projects that are just starting out should start at 0.1, until they get tested and used at least a little bit.
    - The "-SNAPSHOT" is a Maven standard which means that this is a project which is currently in development. It can be removed once the project is officially "released".
- **mavenPassword** - This will be the Maven password you got when you registered.
- **mavenUsername** - This will be the Maven username which you got when you registered.

After you enter the final information and press the Enter key, you are prompted to confirm the entered information. Enter Y and press the Enter key again.

Here is an example output:

```
>mvn archetype:generate -DarchetypeGroupId=com.searchtechnologies.aspire -DarchetypeArtifactId=aspire-distribution-archetype -DarchetypeVersion=3.2 -DrepositoryId=stPublic
[INFO] Scanning for projects...
Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.4.1/maven-clean-plugin-2.4.1.pom
.
.
.
Define value for property 'groupId': : com.searchtechnologies
Define value for property 'artifactId': : aspire-demo
Define value for property 'version':  1.0-SNAPSHOT: : 0.1-SNAPSHOT
Define value for property 'mavenPassword':  : : xxx
Define value for property 'mavenUsername':  : : johndoe@mycompany.com
[INFO] Using property: package = pom
Confirm properties configuration:
groupId: com.searchtechnologies
artifactId: aspire-demo
version: 0.1-SNAPSHOT
package: pom
Y: : y
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Archetype: aspire-distribution-archetype:2.2.2
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: com.searchtechnologies
[INFO] Parameter: artifactId, Value: aspire-demo
[INFO] Parameter: version, Value: 0.1-SNAPSHOT
[INFO] Parameter: package, Value: pom
[INFO] Parameter: packageInPathFormat, Value: pom
[INFO] Parameter: version, Value: 0.1-SNAPSHOT
[INFO] Parameter: package, Value: pom
[INFO] Parameter: groupId, Value: com.searchtechnologies
[INFO] Parameter: artifactId, Value: aspire-demo
[INFO] Parameter: mavenPassword, Value: xxx
[INFO] Parameter: mavenUsername, Value: john-doe@mylocation.com
[INFO] project created from Archetype in dir: C:\Users\pnelson\Desktop\AspireDemo\aspire-demo
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 45.292s
[INFO] Finished at: Wed Jan 26 13:28:55 EST 2014
[INFO] Final Memory: 9M/109M
[INFO] ------------------------------------------------------------------------
```

Step 2: Build the Distribution

So now you can run Aspire, right? Well, actually, there's one more step.

The purpose of the Maven archetype (the previous step) was to build a Maven distribution project. Essentially this is a template distribution. We will now need to use Maven to build the distribution itself.

The sequence goes like this:

```
Aspire Distribution Archetype --mvn--> Aspire Distribution Project --mvn--> Aspire Distribution
```

Why is it so complicated? Well, Aspire is meant for industrial-strength production installations. It is not a toy system. This means:

1. There needs to be a solid, repeatable method for building deployments.
2. The same distribution will likely need to be built for multiple environments:
    - Development
    - Quality Assurance
    - Staging
    - Production

The Maven Distribution Project amply fulfills these needs. It provides a template for building copies of your distribution. Maven "profiles" can be used to customize the template for different environments, specifying different server addresses and database names for the Development environment vs the Staging environment, for example.

Okay, so let's actually build our Aspire distribution. Fortunately, this is very easy.

1. Change your working directory into your Maven distribution project; this is the same as what you entered in step 5 for the artifactId.
2. In the command window, execute: **mvn clean package**

Again, since this is your first time executing "mvn package", Maven will download a bunch of plug-ins, which it needs to do its job. But once that's done, it will build the Aspire distribution and you're ready to go.

```
> cd aspire-demo
> mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Aspire Distribution Project 0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-resources-plugin/2.5/maven-
resources plugin-2.5.pom
Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-resources-plugin/2.5/maven-
resources plugin-2.5.pom (7 KB at 15.1 KB/se c)
Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/19/maven-plugins-19.pom
.
.
.
[INFO] Reading assembly descriptor: distribution.xml
[INFO] Copying files to C:\Users\pnelson\Desktop\AspireDemo\aspire-demo\target\aspire-demo-0.1-SNAPSHOT-
distribution
[WARNING] Assembly file: C:\Users\pnelson\Desktop\AspireDemo\aspire-demo\target\aspire-demo-0.1-SNAPSHOT-
distribution is not a regular file (it may be a directory). It cannot be attached to the project build for
installation or deployment.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3:12.857s
[INFO] Finished at: Wed Jan 25 13:46:04 EST 2012
[INFO] Final Memory: 9M/51M
[INFO] ------------------------------------------------------------------------
```

Messages you could see when building the distribution

There are a number of warning message you may see when building the distribution. Don't let this worry you. Basically, if you see the following message, everything worked OK :

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

However, here's an explanation of warnings you may see.

This warning refers to the fact we don't hard code the encoding for the ascii files we use. We tried to do this in the very early stages and it gave us more problems than it fixed. This shouldn't cause any issues as long as you're not working in any languages that use extended character sets.

```
 [WARNING] Using platform encoding (MacRoman actually) to copy filtered resources, i.e. build is platform
dependent!
```

We recently changed the build process so that you could include dependencies in the pom file to cause "application bundles" to be copied from Maven in to the target bundles/aspire directory at build time. This allows Aspire to use those application bundles on hosts which have no internet access. You've always been able to do that with component bundles. The message below tells you that you've not got any dependencies with the group id *com.searchtechnologies.appbundles*.

```
 [WARNING] The following patterns were never triggered in this artifact inclusion filter:
   o  'com.searchtechnologies.appbundles:*'
```

Some versions of Maven's mvn command (mainly 2.x) produce an unwanted file and copy it to the target bundles/aspire directory . This warning indicates your version didn't (as this line warns you the file is not being excluded when things are being copied to the target directory)

```
 [WARNING] The following patterns were never triggered in this artifact exclusion filter:
   o  'com.searchtools:aspire-demo:jar:0.3-SNAPSHOT'
```

This warning just refers to the fact that the build is copying to a directory and therefore can not produce something that can be place back in a Maven repository.

```
 [WARNING] Assembly file: /Users/st/aspire-test-3/aspire-demo/target/aspire-demo-0.3-SNAPSHOT-distribution is
not a regular file (it may be a directory). It cannot be attached to the project build for installation or
deployment.
```

mvn install

In Maven 2.x, you can get away with running the following command to build the distribution:

```
 mvn install
```

In Maven 3.x, you can't run this command to build a distribution, so make sure you use **mvn package** as detailed above. If you do use **install**, the build will fail and you'll see the following:

```
 [ERROR] Failed to execute goal org.apache.maven.plugins:maven-install-plugin:2.3.1:install (default-install)
on project test-project: The packaging for this project did not assign a file to the build artifact -> [Help 1]
 [ERROR]
 [ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
 [ERROR] Re-run Maven using the -X switch to enable full debug logging.
 [ERROR]
 [ERROR] For more information about the errors and possible solutions, please read the following articles:
 [ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionException
```

What do we have?

The following diagram shows the directory structure you have created within your Maven distribution project so far:

blocked URL

Step 3: Launch Default Distribution

The default Aspire distribution is a fully functional version of Aspire. Let's launch Aspire.

1. In the command window, change your working directory to: target\aspire-demo-0.1-SNAPSHOT-distribution
2. Execute: bin\aspire.bat

Here is **example output**:

```
Removing Felix-Cache and AppBundle-Cache directories

************************************************************************
*
* ASPIRE BOOTLOADER
*
* Bundle id : 10
*
* Location  : file:bundles/boot/aspire-bootloader-3.2.jar
*
.
.
.
2014-02-04T20:38:43Z INFO [/Workflow]: Installed component: /Workflow/WfManager
2014-02-04T20:38:43Z INFO [aspire]: Successfully started appBundle: /Workflow (location: com.
searchtechnologies.aspire:app-workflow-manager)
AUTOSTART: No applications to start
```

NOTE: The aspire-dcm-enterprise component is available only with Enterprise systems (and is used for Distributed Processing and Failover).

aspire-dcm-enterprise THIS ITEM IS BEING DEPRECATED.

At this point, you can use the standard administration interface (http://localhost:50505) to install pre-packaged applications such as connectors and search engine publishers. Go to UI Introduction for more details.

Errors?

If you see an error like this:

```
2014-11-19T10:20:35Z INFO [BOOTLOADER]: Fetching: com.searchtechnologies.aspire:aspire-application:2.0.3
2014-11-19T10:20:43Z ERROR [BOOTLOADER]: Cannot get file from repository (http://repository.searchtechnologies.
com/artifactory/simple/community-public/) - check the component is properly deployed. File: http://repository.
searchtechnologies.com/artifactory/simple/community-public/com/searchtechnologies/aspire/aspire-application/3.2
/aspire-application-3.2.jar
org.apache.maven.wagon.authorization.AuthorizationException: Access denied to: http://repository.
searchtechnologies.com/artifactory/simple/community-public/com/searchtechnologies/aspire/aspire-application/3.2
/aspire-application-3.2.jar
       at org.apache.maven.wagon.providers.http.LightweightHttpWagon.fillInputData(LightweightHttpWagon.java:
119)
       at org.apache.maven.wagon.StreamWagon.getInputStream(StreamWagon.java:116)
       at org.apache.maven.wagon.StreamWagon.getIfNewer(StreamWagon.java:88)
       at org.apache.maven.wagon.StreamWagon.get(StreamWagon.java:61)
       at org.sonatype.aether.connector.wagon.WagonRepositoryConnector$GetTask.run(WagonRepositoryConnector.
java:511)
       at java.util.concurrent.ThreadPoolExecutor$Worker.runTask(ThreadPoolExecutor.java:895)
       at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:918)
       at java.lang.Thread.run(Thread.java:662)
```

and Aspire exits:

```
ERROR: Failed to load Aspire application
Shutting down OSGI
```

the first thing to check is that you have the correct username and password in the *config/settings.xml* file. See here for details.

Step 4: Launch the Example application.xml File

blocked URL

A simple application.xml file is provided with your application. Let's go to the debug interface and start it up to see what it does.

1. Point your web browser to http://localhost:50505/aspire (the debug interface).
2. Where it says "Load a new Application", enter "config/application.xml"
3. Then click "start"

This will start up the sample application. Aspire will take a minute or two to launch the first time because (just like Maven) Aspire will automatically download components from the Search Technologies Maven Repository. These are stored in the Maven local repository (the .m2 directory) just like Maven does.

You should see something that looks like this:

```
> cd target\aspire-demo-0.1-SNAPSHOT-distribution
> bin\aspire.bat
Removing Felix-Cache and AppBundle-Cache directories
->Removing Felix-Cache and AppBundle-Cache directories


********************************************************************
*
* ASPIRE BOOTLOADER
*
* Bundle id : 10
*
* Location  : file:bundles/boot/aspire-bootloader-3.2.jar
*
.
.
Starting bundle: 13 - file:/C:/Users/pnelson/.m2/repository/com/searchtechnologies/aspire/aspire-extract-domain
/2.2.2/aspire-extract-domain-2.2.2.jar
Started component factory: aspire-extract-domain (bundle 13)
Registering component: /FeedOneExample/StandardPipeManager/ExtractDomain
Registering component: /FeedOneExample/StandardPipeManager/PrintToFile
AUTOSTART: Complete
```

blocked URL

Once the sample application is started, you can return to the debug console: http://localhost:50505/aspire

The debug-console provides status information and some low-level admin controls for Aspire. Through the debug console you can:

- **Load New Applications**
  - Each aspire application will be an application.xml file which has components and pipelines for processing content.
  - Multiple configurations can be loaded into the same instance of Aspire.
    For example, you can have a separate configuration for each database that you want to process with Aspire.
- **Refresh Applications**
  - This will reload the application.xml file for a configuration, loading all new configuration settings and automatically loading components as needed
- **Browse Applications**
  - Every component in an Aspire Application Configuration has a separate web page within the Aspire home page.
- **Component Status / Component Controls**
  - Many of the components will use the debug console to display additional status about the component or to allow for administrator control over the component (reloading XSL transforms, etc.)
    For example, you can turn on statistics for pipeline managers to see which pipeline stages are the slowest.
- **Update Components**
  - Not only can component configurations be updated, but the binary Java code for individual components can be updated as well. Click on "Check For Updates". If a new version of a component is available from the Maven Repository, you will be given the option to update the component.
    This is possible because Aspire is built on top of OSGi, which allows for Jar files to be dynamically reloaded as needed. The entire Aspire system has been built to allow for dynamic component updates.
- **View the OSGi Web Console**
  - The OSGi web console allows you to interact with Apache Felix directly. The username/password is "admin/admin".
- **View the Health of the System**
  - Clicking on the "Health" Bar will show you detailed health about the system.
    Note that health checks need to be configured in the system, and are not turned on by default (other than the application startup health)

So you can see, the debug console contains quite a lot of useful functionality. Spend some time exploring it before going on to the next (and final) step.

Step 5: Process a URL

Let's process a URL with the default pipeline.

blocked URL

1. Go back to the home page: http://localhost:50505/aspire
2. Click on the "/FeedOneExample" Component Manager.
3. Click on the "FeedOne" sub component.
    - NOTE: As an alternative, you can jump directly to the FeedOne component by clicking here: http://localhost:50505/aspire/FeedOneExample/FeedOne/
4. Enter "http://www.searchtechnologies.com" where it says "URL to Feed:" and then click on the "feed" button.
    - Note: Do **not** press your RETURN or ENTER key.

After clicking on "feed", you should see the "Job status" table that indicates your job has been submitted to the Aspire pipeline.

blocked URL

The steps above submit a URL down the default pipeline, which will:

- Fetch the URL from the remote web server
- Extract text content from the HTML page
- Extract the domain name from the host name

So, the pipeline doesn't actually send the document to a search engine to index it, but it certainly could. That would only take adding one additional stage to the pipeline.

The job should take only a small fraction of a second to process. You can see that it's done (and that everything worked correctly) by refreshing the page. If it was successful, you should see "Successful" in neon green text.

blocked URL

You can click on the "detail" link in the job status to see the results of the pipeline. What you should see (you may need to scroll down a bit) is an XML representation of Aspire's internal document structure, which contains the result of all document processing. It will contain a <content> tag which contains the text content (from the text extraction component), http headers and web server information (from communicating with the web server at searchtechnologies.com), the <content> (added by the text extractor), and the <domainName> (extracted from the host name of the URL).

Step 6: Where to Go From Here

Congratulations! You have built a brand-new distribution of Aspire from scratch using the distribution archetype, and you launched the default Aspire installation and processed a URL.

Here are some suggestions on what to do next:

1. Explore your distribution. Especially look into the config directory and read the "settings.xml" and "application.xml" files.
    - For more information on the settings.xml file, go here: Settings Configuration
    - For more information on the application.xml file, go here: Application Configuration.
2. Try adding a Groovy Scripting stage to your component.
    - Groovy scripting allows you to print / read / modify the metadata for documents processed by Aspire. It is the most common way to add custom processing to an Aspire pipeline.
    - More information on Groovy Scripting with Aspire can be found here: Use Groovy
    - Note that you'll need to add your new Groovy Scripting component to your Aspire pipeline. See here: Pipeline Manager to learn more about pipelines

For example, the following is a minimal Groovy Scripting component:

```
<component name="SampleGroovyScript" subType="default" factoryName="aspire-groovy">
  <config>
    <script>
      <![CDATA[
        println "Hello World!";
        println "The Aspire Document is this: ";
        println doc.toString(/*pretty:*/true);
      ]]>
    </script>
  </config>
</component>
```

You can add this component to your default Aspire configuration using the following steps:

1. Add the component to your "application.xml" file.
    - Put it after the <component name="ExtractDomain"> component
2. Then add "SampleGroovyScript" as a stage to your pipeline.
    - Put it after <stage component="ExtractDomain" />
3. Now reload your configuration
    - Do this by going to Aspire debug console home: http://localhost:50505/aspire
    - Then click on the circular arrow button next to the "/FeedOneExample" configuration
4. Now go back to the FeedOne component and resubmit the http://www.searchtechnologies.com URL to the pipeline (see above).
    - Look at the console window where you first ran "aspire.bat". You should see "Hello World!" followed by a JSON representation of the Aspire document.

- For more information on the Groovy scripting language, go here: http://groovy.codehaus.org/
- For more information on methods available on the "doc" variable in your Groovy script go to:
    - AspireObject
    - The AspireObject javadoc page