Using IntelliJ with the Broad Sting Repository

Aaron Mckenna

March 15, 2009

1 Overview

Using JetBrains IntelliJ with the Broad String repository is a relatively simple process.

1.1 Getting the source

The source can be checked out from the repository, from the following link:

https://svnrepos/sting

Using the command:

```
svn co http://svnrepos/sting ./sting
```

Replacing the second directory with the location you'd like the code to reside in on your local machine.

1.2 Getting IntelliJ

Licensing of JetBrains IntelliJ at MIT is done one license at a time, we don't have a site license for it. You'll need to contact help, (help@broad.mit.edu) and they'll retrieve a license for you. In the the mean time you can download the IntelliJ demo, which is a fully featured 30 day trial that the license can be entered into. You can download it from their site:

http://www.jetbrains.com/idea/download

When you do have your license, you can enter the license into IntelliJ by selecting help, and then the register drop down option.

2 Working with the code in IntelliJ

2.1 Setting up the project

IntelliJ doesn't need to know that the project is an ant project specifically, instead it knows how to build an ant project once it's loaded. To setup a new project, create a new project, selecting the *Create project from existing sources* option. It's best to work with either the playground or the core sources one at a time, and all development work should be done in the playground. The best option in IntelliJ is to choose the java directory in playground as your source, and let IntelliJ create the project directory there.

1. Choose *New Project* from the file menu

2. Choose the *Create project from existing sources* option.

3. Name your project, something sensical like Playground, and choose the source directory *sting/playground/java*.

4. Choose the .idea (directory format) project storage. This is just a recommendation, you can do either.

5. After clicking next, IntelliJ should detect that the java directory already has a src directory, click through the next few windows and finally select finish.

To have IntelliJ recognize that the project is an Ant project, select the Ant tab on the far right of the IntelliJ window. When the window pane opens, click the plus symbol, and select the build.xml in the /playground/java directory. Now the Ant build target window should have a list of the targets that were loaded from the build file.

2.2 Building the code

To build the code, select the compile list options from the Ant window. Ivy should automatically build the dependancy list, and fetch any libraries that aren't already loaded. To transfer the code to one of our machines once a jar file is created.

Useful targets include:

compile Compiles all java code in the source tree. Places generated classes in the build directory.

dist Generates jar files, suitable for running via java -jar YOUR_JAR. Places resulting jars in the dist subdirectory.

resolve Resolves third-party dependencies. Downloads all third-party dependencies to the lib directory.

javadoc Generates javadoc for the source tree. Places javadoc in the javadoc directory.

clean Removes artifacts from old compilations / distributions.

View all available ant targets by running 'ant -projecthelp' in the directory containing build.xml.

2.3 Adding external sources

To easily debug into third-party dependencies like picard and samtools, add their source to the dependency list as follows:

1. Right-click one of the Sting modules and select 'Module Settings'.
2. Select the 'dependencies' tab.
3. Right-click your library directory. Hit 'Edit...'.
4. Click 'Attach Sources...'.
5. In the 'Choose Source Roots' directory, if the project contains a test directory, unselect it.
6. Click 'Okay' or 'Apply' until all dialogs are closed.

2.4 Debugging remotely on gsa1 / gsa2

To debug remotely on gsa1 or gsa2:

1. Select 'Edit Configurations' from the 'Run' menu.
2. Click the '+' button to add a new configuration. Select 'Remote'.
3. Type in 'gsa1' or 'gsa2' in the Host field. Choose a port randomly (please choose a value ¿= 1024).

4. Before launch, run ant target 'dist'.

5. Make a note of the command-line settings required to initiate remote debugging:

```
-Xdebug
-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=5005
```

6. Start the application on gsa1 as follows:

```
java -Xdebug \
     -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=5005 \
     -Xmx{mem_required} \
     -jar {your jar} {your options}
```

7. Select the 'Debug' icon from the toolbar. Press the 'Debug' button in the resulting dialog.

2.5 Setting up the header to contain the license file

3 Platform-specific Notes

3.1 Using IntelliJ on Linux

If IntelliJ hangs or crashes, try changing the default arguments specified in the ${INTELLIJ_HOME}/idea.vmoptions file to the following:

```
-server
-Xms768m
-Xmx1248m
-Xmn170m
-XX:MaxPermSize=300m
-ea
```