Table of Contents

andy_straw@urmc.rochester.edu
2015-03-10

Eclipse Projects for LabKey Source

Here is one way to set up Eclipse projects for LabKey Server Java source. Be sure to build the LabKey source (using the "ant build" target) before setting up your Eclipse projects.

The basic approach is to use one project for the combination of server/api and server/internal, one project for the Java Client API (i.e., remoteapi), and then use one additional project for each module. Attached to this wiki page are .project and .classpath files you can use to set up the server/api + server/internal project, the client (remote) API, as well as projects for the following modules:

- Experiment
- Study

Beware of the following dependencies:

- To set up a module project, you must also set up the server/api + server/internal project, since the modules depend on these.
- Since 14.1, you also need the remoteapi/java project set up, because the server/api+server/internal project depends on it. (This dependency seems to be the result of just one or two classes in server/*.)

If you're curious about how the .project and .classpath files were created in the first place, see the page Eclipse Projects for LabKey Source - Details.

For each project, the steps are basically the same, except:

- Use the appropriate .project and .classpath files
- Put the .project and .classpath files in the appropriate LabKey source folder.
- Change the value of LabKey_Root appropriately
- Change the value of TOMCAT_LIB variable appropriately (necessary only in ServerAPI.project)
- Change the project name appropriately (this is optional)

Values for each of these for each project are in this table:

LabKey source folder(s)	Description	.project and .classpath files to use	Default project name (XX_Y is LabKey version)
server/api + server/internal	Core server code	ServerAPI.project ServerAPI.classpath	LabKey_XX_Y_ServerAPI
remoteapi/java	Java client API	RemoteAPI.project RemoteAPI.classpth	LabKey_XX_Y_RemoteAPI
server/modules/experiment	Experiment module	Modules_Experiment.project Modules_Experiment.classpath	LabKey_XX_Y_Modules_Experiment
server/modules/study	Study module (includes Assay)	Modules_Study.project Modules_Study.classpath	LabKey_XX_Y_Modules_Study

Here are the general instructions for each project:

1. Drop the attached 14_3_whatever.project and 14_3_whatever.classpath files into the appropriate folder of your LabKey source (see table above). Use the files for the appropriate version of LabKey.

- Rename the files to .project and .classpath
- Edit the .project file:
  - Change the value of the LabKey_Root variable to the location of the root of your LabKey source folder (that is, the folder that has server, external, tools, etc. as sub-folders). The attached .project files uses a value like "C:/home/dev/code/labkey_3" (which is what I named the folder where I have 14.3 source).
  - For the ServerAPI project only, change the value of the TOMCAT_LIB variable to the location of the lib sub-folder of your CATALINA_HOME. The attached .project file usea the value "file:/C:/home/dev/tools/tomcat/apache-tomcat-7.0.47/lib" (which is the lib directory inside my CATALINA_HOME folder).
  - Re-name the project, if you like. The attached .project file names the project as indicated in the table above.

2. In Eclipse, in Package Explorer, right-click and choose "Import...".

- Open "General" and choose "Existing Projects into Workspace". Click "Next".
- Select "Select root directory", hit "Browse..." and select the appropriate folder of your LabKey source (see table above).
- When you return to the main Import dialog, a project with name that matches what's in the .project file should be listed. Check the check box next to it, and hit "Finish".
- You should now have a new project that builds properly in Eclipse - Except for these errors that you can ignore:
  - ServerAPI will complain that LogPrintWriter is defined twice (once in server/api/src, once in server/internal/src).
  - Modules_Experiment will complain about a type mismatch in the return statement in org.labkey.experiment.DerivedSamplePropertyHelper.getNamePDs().

Note about Spring source code

LabKey uses Spring, and the jars for Spring core and web-mvc are included in external/lib/server. However, they don't include source for Spring, and that can be pretty useful sometimes. If you need it, I have attached the jar of that source to this page. Download the jar to your local machine, and then modify the source attachment location of spring-2.5.5.jar and spring-webmvc-2.5.5.jar to be wherever you stored the Spring source jar on your local machine. Here are details of how to do that:

- Project->Properties, then select "Java Build Path" on the left, then select "Libraries" tab on right.
- Expand the tree node of the spring-2.5.5.jar. You should see "Source attachment" and "Javadoc location".
- Double click "Source attachment". Choose "External File...". Browse to location of spring-sources.jar on your local machine, and select it. Click "OK" twice.
- Do the same for spring-webmvc-2.5.5.jar

Note that the .classpath files attached to this page define a source attachment for the Spring jars, but the location is specific to where I (Andy) happen to have put the spring-sources.jar file.

Eclipse Projects for LabKey Source - Details

This page describes how Eclipse projects for LabKey source were set up originally. If you just want to set up the projects quickly, see the page Eclipse Projects for LabKey Source instead.

We define path variables, and linked resources using those path variables, which are used to define the Java build path (source folders, libraries this project depends on, class file folders, projects this one depends on, etc.). This allows the .classpath file to be shared, even though different developers have different locations for their LabKey source. The variables also make the .classpath entries more succinct and clear.

RemoteAPI

This is the simplest project to set up. Despite it's name and purpose, this project has now become (as of 14.1) a dependency of the ServerAPI project (for one or two classes only).

1. Create a Java Project. Name it LabKey_13_1_RemoteAPI (replacing "13_1" with whatever version of LabKey source you're using). Uncheck "Use default location" and set "Location" to the remoteapi/java folder of your LabKey source.

2. When you hit "Finish", Eclipse will build the project, and that should succeed because all the dependencies are in the lib/ directory under remoteapi/java.

ServerAPI + Server Internal

Ideally, we would have one Eclipse project for server/api, and one for server/internal. Unfortunately, these two seem to have circular dependencies, which means neither will build without the other. But Eclipse can't handle a circular project dependency, so we have to combine the two into one project, using source from both as the source for the one project. Here's how we create that combined project.

1. Create a Java Project. Name it LabKey_13_1_ServerAPI (replacing "13_1" with whatever version of LabKey source you're using). Uncheck "Use default location" and set "Location" to the server/api folder of your LabKey source.

2. When you hit "Finish", Eclipse will try to build the project, but will have lots of errors due to unresolved dependencies. We'll fix these.

2b. Add the LabKey_XX_Y_RemoteAPI project as a project this one depends on.

3a. Create a new linked folder so we can add server/internal source to this project, as follows:

- New->Folder
- Click the "Advanced" button.
- Click the "Link to alternate location (Linked Folder)" radio button
- Click "Variables..."
- Click "New..." to define a new variable for LabKey_Root
    - Name: "LabKey_Root"
    - Location: Click "Folder..." and browse to the root folder of your LabKey source.
    - Click "OK"
- Select LabKey_Root from the list of variables, and click "New..." to define a new variable for LabKey_server_internal based on LabKey_Root
    - Name: "LabKey_server_internal"
    - Location: Click "Variable..."
    - Select "LabKey_Root" and click "Extend..."
    - Browse to, and select, server/internal, and click "OK"
    - You should now see Location has value "${LabKey_Root}\server\internal". Click "OK"
- Select LabKey_server_internal and Click "OK"
- You should now be back at the "New Folder" dialog, with LabKey_server_internal as the name of the Linked Folder to create. Click "Finish".
- You should now have a new linked folder named "LabKey_server_internal" in your Package Explorer under your _ServerAPI your project.

3b. Add source folders from server/internal

- In Package Explorer, expand LabKey_server_internal. You should see gwtsrc and src, among other sub-folders.
- Right-click on "src", and choose "Build Path->Use as Source Folder".
- Do the same for "gwtsrc".
- This will get rid of some compile errors, but introduce others.

3c. Add jars from server/internal/lib (i.e., LabKey_server_internal/lib) to build path. (See step 4b for details about how to do this.)

4a. Create a new linked folder so we can add jars from external/lib/server/ and external/lib/build to our build path

As above, create a new linked folder named "LabKey_external_lib" based on a variable by the same name, which extends LabKey_Root with external/lib:

- New->Folder
- Click the "Advanced" button.
- Click the "Link to alternate location (Linked Folder)" radio button
- Click "Variables..."
- Click "New..." to define a new variable for LabKey_external_lib
    - Name: "LabKey_external_lib"
    - Location: Click "Variable..."
    - Select "LabKey_Root" and click "Extend..."
    - Browse to, and select, external/lib, and click "OK"
    - You should now see Location has value "${LabKey_Root}\external\lib". Click "OK"
- Select LabKey_external_lib and Click "OK"
- You should now be back at the "New Folder" dialog, with LabKey_external_lib as the name of the Linked Folder to create. Click "Finish".
- You should now have a new linked folder named "LabKey_external_lib" in your Package Explorer under your _ServerAPI your project.

4b. Add jars from external/lib/server/ (i.e., LabKey_external_lib/server), external/lib/build/ (i.e., LabKey_exernal_lib/build) ***(do NOT include gwt-dev.jar or yuicompressor-2.3.5.jar)***, and external/lib/common/ (i.e., LabKey_external_lib/common) ***(include ONLY mail.jar)*** to build path. (Note: starting with 14.1, it's not "external/lib/common", but rather "external/lib/tomcat".)

In this step, when adding jars to the build path, do NOT include any files with extensions other than .jar, and don't include files with "src", "source" or "javadoc" in their name. There are some files with .txt, .xml, and .dtd extensions that you don't want, and the "src", "source", "javadoc" and .zip are source or javadoc files. The source should be attached as source, to the corresponding library jar.

- Add all jars from external/lib/server. If they have source, attach it..
- Do the same for jars from external/lib/build.

Details, if you need them:

- To add jars to the build path:
  - Project->Properties, then select "Java Build Path" on the left, then select "Libraries" tab on right.
  - Click "Add JARs", and browse to the appropriate folder (use the linked folders we created), and select jars. (It's probably easiest to select all files at first. Than after they all get added to the list of Libraries, select ones that you don't want (.zip, source, javadoc, etc.) and hit "Remove".)
- To attach source to a library:
  - Project->Properties, then select "Java Build Path" on the left, then select "Libraries" tab on right.
  - Expand the tree node of the library (i.e., .jar file). You should see "Source attachment" and "Javadoc location".
  - Double click "Source attachment". Choose "Workspace...". Browse to location of corresponding zip or jar file that contains the source for the selected library, and select it. Click "OK" twice.
- To attach javadoc to a library: (This isn't really necessary, if the library has source attached, because Eclipse parses the javadoc from the source. If you have a library with javadoc, but not source, attach the javadoc as follows.)
  - Project->Properties, then select "Java Build Path" on the left, then select "Libraries" tab on right.
  - Expand the tree node of the library. You should see "Source attachment" and "Javadoc location".
  - Double click "Javdoc location". Choose "Javadoc in archive" and then "Workspace file".
  - Hit "Browse..." next to "Archive path:". Browse to location of corresponding zip or jar file that contains the javadoc for the selected library, and select it. Click "OK" twice.

5a. Create a new linked folder so we can add jars from our tomcat lib directory to our build path

As above, create a new linked folder named "TOMCAT_LIB" based on a variable by the same name:

- New->Folder
- Click the "Advanced" button.
- Click the "Link to alternate location (Linked Folder)" radio button
- Click "Variables..."
- Click "New..." to define a new variable for TOMCAT_LIB
  - Name: "TOMCAT_LIB"
  - Location: Click "Folder..."
  - Browse to tomcat's home directory, select, the "lib" sub-directory, and click "OK"
  - You should now see Location has value "${LabKey_Root}\external\lib". Click "OK"
- Select TOMCAT_LIB and Click "OK"
- You should now be back at the "New Folder" dialog, with TOMCAT_LIB as the name of the Linked Folder to create. Click "Finish".
- You should now have a new linked folder named "TOMCAT_LIB" in your Package Explorer under your _ServerAPI your project.

5b. Add jars the following jars from TOMCAT_LIB:

- el-api.jar
- jasper.jar
- jsp-api.jar

6a. Create a new linked folder so we can add build/xb as a class folder to our build path

As above, create a new linked folder named "LabKey_build_xb" based on a variable by the same name, which extends LabKey_Root with build/xb:

- New->Folder
- Click the "Advanced" button.
- Click the "Link to alternate location (Linked Folder)" radio button
- Click "Variables..."
- Click "New..." to define a new variable for LabKey_build_xb
  - Name: "LabKey_build_xb"
  - Location: Click "Variable..."
  - Select "LabKey_Root" and click "Extend..."
  - Browse to, and select, build/xb, and click "OK"
  - You should now see Location has value "${LabKey_Root}\build\xb". Click "OK"
- Select LabKey_build_xb and Click "OK"
- You should now be back at the "New Folder" dialog, with LabKey_build_xb as the name of the Linked Folder to create. Click "Finish".
- You should now have a new linked folder named "LabKey_build_xb" in your Package Explorer under your _ServerAPI your project.

6b. Add build/xb as class folder to build path

- Project->Properties, then select "Java Build Path" on the left, then select "Libraries" tab on right.
- Select "Add Class Folder...", and select the check box next to LabKey_build_xb, under your project.
- Click "OK" twice.

Experiment module

- New project - point at server/modules/experiment
  - Should automatically include both src and gwtsrc as source folders
- Add the following variables and linked folders:
  - LabKey_Root = root folder of your LabKey source (variable only - no linked folder)
  - LabKey_external_lib = ${LabKey_Root}/external/lib/
  - LabKey_build_xb = ${LabKey_Root}/build/xb/
- Add the following to the Java Build Path:
  - Project dependencies:
    - LabKey_13_1_ServerAPI (or whatever you called the server/api + server/internal Eclipse project)
  - Library dependencies (attach source, if available):
    - All jars from LabKey_external_lib/server
    - servlet-api.jar from LabKey_external_lib/build
    - jsp-api.jar from TOMCAT_LIB
  - Class folders:
    - LabKey_build_xb

Study module

- New project - point at server/modules/study
  - Should automatically include both src and gwtsrc as source folders
- Add the following variables and linked folders:
  - LabKey_Root = root folder of your LabKey source (variable only - no linked folder)
  - LabKey_external_lib = ${LabKey_Root}/external/lib/
  - LabKey_build_xb = ${LabKey_Root}/build/xb/
  - LabKey_build_modules = ${LabKey_Root}/build/modules/
- Add the following to the Java Build Path:
  - Project dependencies:
    - LabKey_13_1_ServerAPI (or whatever you called the server/api + server/internal Eclipse project)
  - Library dependencies (attach source, if available):
    - All jars from LabKey_external_lib/server
    - mail.jar from LabKey_external_lib/common
    - servlet-api.jar, jsp-api.jar, gwt-*.jar, gxt.jar from LabKey_external_lib/build
    - jsp-api.jar from TOMCAT_LIB
    - schemas.jar from LabKey_build_modules/study/explodedModule/lib
  - Class folders:
    - LabKey_build_xb