Tool Builder

The Eclipse-LAMS ToolBuilder Plug-In

The Eclipse-LAMS ToolBuilder Plug-In is a handy tool for developers wishing to create or modify LAMS tools. It works as a plug-in extension to the Eclipse integrated development environment so that once you install the plug-in you will be able to create new LAMS tool projects within Eclipse. The idea behind the ToolBuilder is to provide developers with the basis of a LAMS tool so that they can focus on their tool's business logic instead of wasting time trying to get the tool to work. The new tool projects created contain all the necessary hooks and calls to the LAMS core code, they will compile and run immediately after creation.

How it Works

The ToolBuilder works by translating files from selected templates using the meta-data that entered in the project wizard. The files are carefully translated so the new tool will be unique and follow the guidelines outlined in the LAMS Tool Contract. The ToolBuilder is designed so the new tools will will run alonside the other tools in your LAMS environment as soon as they are created. At first, the tool will work as a copy of the tool template that is chosen, so all you will need to do is add your own functionalities to the tool to make it your own.

The templates chosen for the ToolBuilder were chosen both for their simplicity and for their well-structured code. Keeping the tool templates simple will make the learning process quicker for creating new tools. At present the following tools are available as templates:

- Forum (lams_tool_forum)
- Notebook (lams_tool_notebook)
- Noticeboard (lams_tool_nb)
- Share Resources (lams_tool_rsrc)

If you want to base your new tool off another existing LAMS tools, then you will need to use our old RenameTool. But the RenameTool is not as nice as the plug-in, so use this plug-in if possible.

One of the pieces of information you need is your tool signature. If you are just playing with LAMS then you can make one up. If you are likely to keep the code please contact Fiona Malikoff or Ernie Ghiglione about specifying a tool signature, also make sure you let the community know by posting a message in the LAMS Community. The tool signature is used in naming your database tables and it can be tedious to change once you have a large amount of code. We are not trying to discourage you or control who writes tools - we are just trying to save you having to change working code later!

Eclipse Workspace Requirements

Check Your Eclipse Workspace
Tool templates are taken from the Eclipse workspace, you will need at least one of these projects in the workspace along with the LAMS core projects for the Toolbuilder to work.

Configure your cvs client as follows to download the LAMS tools:

- access method: pserveruser
- name: anonymous
- server name: code.lamsfoundation.org
- location: /usr/local/cvsroot

The following tool projects are required in your Eclipse workspace for the ToolBuilder to run. You must also have at least one of the tool templates listed in the above section in your workspace so the ToolBuilder has something to copy from.

- lams_admin
- lams_build
- lams_central
- lams_common
- lams_contentrepository
- lams_learning
- lams_monitoring
- lams_tool_deploy
- lams_www

Download and Installation

Getting Eclipse

You can download the latest Eclipse release here.
For more information about setting up the LAMS development environment in Eclipse see the Development Environment page.

Installing the ToolBuilder Plugin

There are two ways to install the LAMS ToolBuilder into Eclipse:

1. (Recommended) Download using Eclipse's inbuilt plug-in download manager.
2. Download the binary, then manually copy the jar into the Eclipse plugin directory.

To install using the **first** method follow these steps:

1. Open up Eclipse.
2. Go to help->software updates->find and install->new remote site.
3. Enter a name for the site (eg LAMS)
4. Copy the following URL for the remote site location: http://saturn.melcoe.mq.edu.au/installers/eclipse/org.lams.toolbuilder.updatesite
5. Then just follow the prompts to install the ToolBuilder.

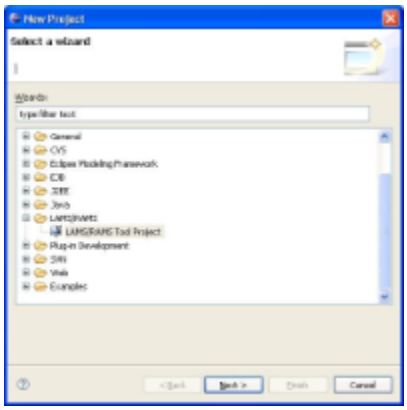
To install using the **second** method, please follow these steps (WARNING, this method does not ensure successful installation because of dependencies).

1. Download the org.lams.toolbuilder_1.0.x.jar binary.
2. If Eclipse is running, shut it down.
3. Copy org.lams.toolbuilder_1.0.x.jar to the eclipse/plugins directory of your Eclipse installation.
4. Restart Eclipse. The tool should now appear in the list of available plugins.

How to use it

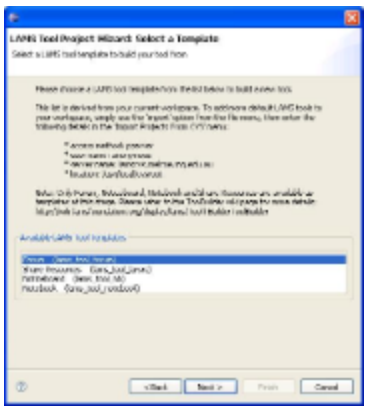
Using the ToolBuilder is quite simple. The following shows the normal steps behind creating a new LAMS tool project using the ToolBuilder

1. In Eclipse goto file -> new -> project, the following screen should appear



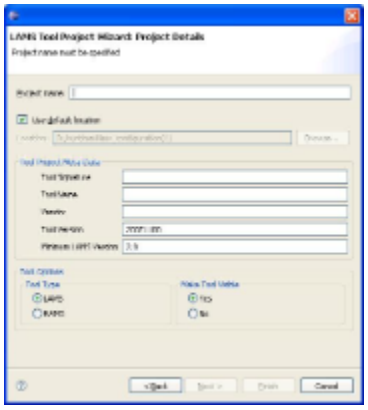
Click to enlarge

2. Open the 'LAMS/RAMS' folder in the New Project Wizard, select 'LAMS/RAMS Tool Project' and click 'next'.
3. The next screen should appear, you are now in the Toolbuilder wizard.
4. Select the template you wish to make your new LAMS tool from and click the 'next' button.



Click to enlarge

5. Now you are in the project meta-data page, you need to enter the some information about your new tool.



Click to enlarge

6. Enter the new project name, this will be the name of the folder that contains the LAMS tool project in eclipse, for example lams_tool_forum
7. Enter the tool signature, the tool signature should follow the form prescribed here.
8. Enter your tool name, this will be used to translate the files in your chosen template.
9. Enter the vendor, this can be the organisation you work for or your name, or whatever you choose to call it. Note that if you change the vender name from lamsfoundation, you will need to edit the beanRefContext.xml and commonContext.xml in lams_common/src/java/org/lamsfoundation/lams to point to your tool. In beanRefContext.xml there is a line like this:

```
<value>classpath*:org/lamsfoundation/lams/tool/**/*pplicationContext.xml</
value>
```

add another line like this replacing "lamsfoundation" with your vendor. In commonContext.xml you will find a like like this:

```
<value>classpath*:org/lamsfoundation/lams/tool/\**/**/*.hbm.xml</value>
```

again add another line like this replacing "lamsfoundation" with your vendor.

10. Enter the tool version, this is an 8-digit number representing the date of tool creation in the form YYYYMMDD
11. Enter the minimum LAMS server version, if your tool requires a certain version of LAMS, be sure to enter it here.
12. Select the type of tool you wish to make, LAMS or RAMS.
13. If you don't want the tool to be displayed as a standalone tool in author or learner, select no in the 'Make Tool Visible' box.
14. Clicking 'finish' will start the project creation process, this should take a few moments to finish.

Deploying the New Tool

Once you have created your new tool, it should be able to compile and run like all the other tools. First, you should build the LAMS core projects before deploying the new tool.

Run the following ant tasks from the lams_build/build.xml file. Alternatively you can simply rum the 'lams-cruise' task.

- rebuild-db
- assemble-ear
- deploy-ear
- copy-files

This will build the core. If you want the other tools, do "deploy-tools".

Once you have successfully deployed the LAMS core code, you can then run the 'deploy-tool' task from your new tool's build.xml file.

Known issues

The Tool Builder is designed to get you started with tool development. It doesn't promise to give you a 100% working tool out of the box. As we find issues, we will note them here.

- Copying the share resources tool. When you add a new item (in authoring) it may throw a Javascript error like "submit<tooldesc>Item is not defined". This is because the file rsrc<tooldesc>Item.js hasn't been renamed to match references to it in the .jsp files. You will probably have another javascript error telling you that a file cannot be found - that is the expected name for the rsrc<tooldesc>Item.js. So rename the file and it should work.

Alternatively

If this doesn't work for you, you can use good old unix commands to rename an existing tool and start from scratch.

```
# Renaming the lams_tool_bbb tool in a new tool called XYZ


 find . -name *.java -print0| xargs -0 sed -i 's/Bbb/Xyz/g'
 find . -name *.sql -print0| xargs -0 sed -i 's/Bbb/Xyz/g'
 find . -name *.xml -print0| xargs -0 sed -i 's/Bbb/Xyz/g'
 find . -name *.jsp -print0| xargs -0 sed -i 's/Bbb/Xyz/g'
 find . -name *.properties -print0| xargs -0 sed -i 's/Bbb/Xyz/g'
 find . -name *.MF -print0| xargs -0 sed -i 's/Bbb/Xyz/g'

 find . -name *.java -print0| xargs -0 sed -i 's/bbb/xyz/g'
 find . -name *.sql -print0| xargs -0 sed -i 's/bbb/xyz/g'
 find . -name *.xml -print0| xargs -0 sed -i 's/bbb/xyz/g'
 find . -name *.jsp -print0| xargs -0 sed -i 's/bbb/xyz/g'
 find . -name *.properties -print0| xargs -0 sed -i 's/bbb/xyz/g'

 find . -name *.project -print0| xargs -0 sed -i 's/bbb/xyz/g'
 find . -name *.mymetadata -print0| xargs -0 sed -i 's/bbb/xyz/g'
 find . -name *.MF -print0| xargs -0 sed -i 's/bbb/xyz/g'


# Renaming file names

find .  -type d -print0| xargs -0 rename 's/bbb/xyz/g'
find .  -type f -print0| xargs -0 rename 's/bbb/xyz/g'
find .  -type f -print0| xargs -0 rename 's/Bbb/Xyz/
```