Create a Ghub Tool with Rappture Builder

This set of instructions takes you through creating and deploying a Ghub tool based on C code. It describes how to use the Rappture builder to quickly create a GUI to deploy your tool onto Ghub.

For more information on the high-level process of deploying a tool, refer to Ghub tool development workflow

Create the Code

1. Develop a program (e.g. example.c) that accepts inputs and generates outputs. Compile and test it on your local machine.

Upload and Test on Ghub

2. Log into Ghub and access your Workspace, and click Launch Tool. In the Workspace xterm, organize your home directory in the following way, by creating directories:

```
 ~/apps/toolname/
  test/  - useful for testing basic tool code
     in this phase of dev
  repo/  - houses the local svn repository
     (checked-out code version)
```

For more information on the Hub tool directory structure, refer to Ghub Tool Directory Structure

3. Upload the working source file (example.c) into ~/apps/<toolname>/test/ on Ghub.

To upload, you can:

- sftp from your local machine to Ghub using your Ghub password. You will automatically be in your home directory on Ghub. Use the put command to transfer your source file.
- Alternately, use webdav

Compile and test the basic tool in the Ghub Workspace. Now that your code is uploaded, most of your tool creation work will take place in the Hub Workspace.

Prepare the Repository

4. From the Ghub Workspace, check out the Subversion directory structure into

~/apps/<toolname>/repo. From the ~/apps/<toolname>/ directory:

```
$ svn checkout https://theghub.org/tools/toolname/svn/trunk repo
```

This command creates the correct directory structure for the Tool.

Run Rappture Builder

5. On Ghub, access your Workspace and click ‘Launch Tool’. In the Workspace xterm, run Rappture in builder mode. This lets you use the Rappture GUI to generate a GUI for your own program. From your ~/repo/rappture/ directory, run:

```
$ rappture -builder
```

Populate the GUI

6. In the Rappture GUI, under Tool Interface, select ‘tool’. Enter a Title and Description, and ‘select the appropriate programming language from the pull-down Program menu’. This will instruct Rappture about which files to create.

7. Populate the Tool Interface Input and Output sections. Do this by dragging controls from the left-hand palette and dropping them into the right hand panel. Rename the controls to match the names of your inputs and outputs; update the Label and Description fields.

8. When the tool is populated, click ‘Save As’. From the dialog, select:

- Tool Definition File
- Skeleton Program
- Makefile

Choose names for these files as appropriate. Click Save.

Rappture builder will save the following:

- main.c (default name of the generated C program with inputs/outputs as indicated)
- tool.xml (default name of tool definition file used to make the GUI)
- Makefile (file with instructions for the C compiler)

Edit the Rappture Files

9. In Ghub’s Workspace xterm, edit (e.g. with vi) the following files:

main.c

Rappture’s generated source code. Its default name is main.c. Add functional code for your tool under ‘Add your code here…’ section.

tool.xml

Rappture’s generated tool.xml. Ensure it points to the executable in the bin/ directory.

Makefile

Rappture’s generated Makefile. Ensure that it installs the executable in`bin/`. Add whatever compile targets you like; clean, distclean, install, etc.

invoke

You may also want to look at middleware/invoke. This is the invocation script for your tool itself. Ensure the script is there; it may not need any alterations.

Build and Test

10. In Ghub’s Workspace xterm, build and run the program (build from the src/ directory):

```
$ make
```

This creates your executable. To run it using the Rappture front end, issue the following from your rappture/ directory:

```
$ rappture
```

This should display the Rappture front end generated with the Rappture builder. The tool should run when you click ‘Simulate’. Troubleshoot and patch as needed.

Invoke

To test the tool using the invoke script that Ghub will use, issue the following command from your repo/ directory:

```
$ ./middleware/invoke
```

Once invoke works, you're ready to check the tool and GUI code into Subversion.

Troubleshoot

To troubleshoot the tool or workspace session you're running, refer to ~/data/ and its subdirectories, sessions/ and results/. All user session data are written there.

Check in the Code and Change Status

11. Check the working tool into the Ghub repository using Subversion:

- add all new files (main.c, tool.xml, Makefile, etc.) to Subversion, using the command:

```
$ svn add path1/name1 path2/name2
```

- Commit all files to the Subversion repository, using the command:

```
$ svn commit
```

12. In the tool's Contribtool page, set its status as 'Uploaded'. The administrator will install it for testing in the Ghub application area.

The Ghub application area for your tool is found at:

```
/apps/<toolname>
```

These subdirectories are readable by the tool developer, and invoke calls can be issued there for troubleshooting.

Rappture References

- Rappture xml reference
- Rappture C API
- Example: Rappture and Python