NA_WorkSheet Developer's Guide

July 4, 2006

1 Introduction

1.1 About the NA_WorkSheet

The NA_WorkSheet is a collection of numerical algorithms organized into one place. The worksheet attempts to create an easy to use interface that can quickly be used to carry out numerous numerical analysis algorithms. A selected algorithm once executed in the worksheet will give a result and intermediate data if so desired. The NA_WorkSheet also incorporates a graphing function and some limited import and export of data.

1.2 General Information

Developers can find general information about becoming a contributor in the SourceForge site project's forum under Developers General. Posts can be made here by the public and will be responded to if they have to do with questions about becoming a developer. Developers may choose to contribute by taking on an open task or discussing with an admin a new feature/enhancement. Once a contribution has been made then the individual will be added to the project as an official developer. An official developer will have open access to the project's CVS repository and have their name included as such in the next release. If you decide to take on an open task and have questions please ask. The NA_WorkSheet project is oriented around various numerical analysis techniques, if you do not have some background in this area then it will be hard for you to contribute. The project will though have some openings for other more general tasks and Java code development that is not so orientated around numerical analysis.

2 Technical Overview

2.1 Main Classes

The NA_WorkSheet is comprised of five main classes. These classes general constitute the user interface and control of the worksheet.

- NumericalWorkSheet - The main JApplet/JFrame.
- MainInputPanel - Equation input, algorithm selection, and execution.
- MethodPanel - Content pane for the algorithms' input panels.
- GraphPanel - Equation/data plotting.
- OutputPanel - Algorithm output, evaluation, and graph control.

Several other classes are included in the main source code that are really just off shoots of the five main classes. Three of these are adapters for the GraphPanel. Any processing of the mouse or keyboard input events are handled here and routed appropriately. The WorkSheetJMenuBar, AboutDialog, and HelpFrame all have to with either setting up the worksheet's main menu bar or creating context for the menu bar. Three other classes, ExecuteApproximation, MainInputPanelUpdate and MethodPanelUpdate, house methods that either help switch the worksheet when choosing algorithms or execute the algorithm. Two of these classes in particular will need to be edited anytime a new algorithm is added to the worksheet. Greater detail will be given later with regard to those edits. One other class is include with the main grouping, but is presently not being used. The class is the WorkSheetMouseAdapter.

2.2 Approximation Classes

The approximation classes are the actual algorithm objects in the worksheet. They might also be referred to by some as a package group. All the algorithms are not stand alone objects that can be instantiated. The constructors do not provide that ability generally. A couple of the algorithms are created partially in this way because they are called upon to help other approximations with their execution. This is a feature that might need to be corrected

in the future.

The approximations come into existence for only the during of the execution of the algorithm. Once they have completed their execution and have provided output they are nulled out for garbage collection. An approximation algorithm for the worksheet should generally follow the sample Algorithm Template given in the docs section of the project. The algorithm class should have at lease three main methods:

- getCheckData - Obtains input from the algorithms input panel and checks that data.
- processData - The actual algorithm process and output.
- main(ALGORITHM_NAME) - Method that is called by the worksheet to instantiate the object.

All algorithm classes will need an associated ID that is used to coordinate with the OutputPanel for displaying any results that have been generated. The MethodPanelUpdate holds the sequence numbers for the algorithm IDs.

2.3 Approximation Panels Classes

The approximation panels are independent JPanels that each algorithm class uses to obtain its input parameters. The panels are part of a cardlayout in the MethodPanel. The card selections are made through the MainInputPanel radio buttons and algorithm type combobox. Each algorithm accesses its associated input panel through its getCheckData method. Each approximation input panel should provide methods to getData, clearTextFields data, and getStateCheckbox. The latter is used to determine the output of intermediate data during the execution of the algorithm.

2.4 Expression/Function Classes

The Expression group of classes comprise the equation parser that is used to evaluate values in the function provided by the Equation for Analysis. Some of the algorithms use this parser, others do not. Any equation that is graphed will use this set of classes to evaluate points in the plot.

The Function classes are the various mathematical operations that can be included in an equation. These include trigonometric, and many more standard features that might be used in evaluating an expression.

2.5 Utilities Classes

The Utilities classes in the NA_WorkSheet are miscellaneous methods that are used by many of the algorithms. The methods are recurring in natural throughout the processing of data in the worksheet.

- NA_Utils - Several class methods that perform calls to the expression parser, multiply functions, chop data, string manipulation, and grid bag constrains. This class extends the NumericalWorkSheet class.
- ReadDataFile - Used by several of the algorithms to import data. The class is most often called directly from an approximation's getCheck-Data method.
- TextAreaSort - Algorithms use this class to tokenize the input string obtained from its associated input panel textarea. The class returns a string array that holds not only the data, but also information about this data.
- TextFieldSort - The TextFieldSort class performs the same function as the TextAreaSort except with textfields in the input panels.

2.6 Legendre Classes

The Legendre classes are used to perform the creation of a Legendre table of roots and weights that are used in the Gaussian Quadrature.

3 Source Code Development

3.1 Tools

Various ways can be used to develop code for the project. Each person is going to have personal preferences in tools and methods to create code. An outline will be given here to get you started in this direction for code deployment at the SourceForge site.

These days most programmers use some kind of IDE, integrated development environment, to coordinate the writing, compiling, testing, and packaging of software programs. The NA_WorkSheet was created with the Eclipse IDE. Eclipse is an open source tool that can be found on the Internet at http://www.eclipse.org. The files in the CVS, concurrent version system, for the NA_WorkSheet project can be checked out and immediately ported into an Eclipse project. Just place the files in a known directory on your system and create a new project in Eclipse pointing at that directory. Eclipse comes with an excellent help document that can get you aquatinted with the tool's functionality.

In order to get the NA_WorkSheet source code files on your system I recommend a manual approach. There are some more efficient tools that can help this process like WinCvs, which is available on SourceForge. I use though the PuTTY ssh, secure shell, client to log on to the SourceForge shell server. The project files than can be checked out from the CVS repository into your home directory. Once the project files are in your home directory than they can be downloaded with a standard ftp program like FileZilla or WinSCP. Both of these programs are also open source software that again are available at SourceForge.

Recommended Tools

- Eclipse - IDE http://www.eclipse.org.
- PuTTY - SSH client http://www.chiark.greenend.org.uk/ sgtatham/putty/.
- FileZilla or WinSCP - FTP open source programs at SourceForge.

3.2 Development Cycle

The development of code general follows a cyclic pattern. A project will be setup in your IDE. Once the project is established then source code will be created, compiled, and then executed. A review will then be performed to determine if an expected behavior is obtained. If so your finished after maybe some more testing. Ha! If only we were all so good. Generally the expected behavior will not be obtained so modifications will need to be made to the source and the process will start all over again. Once a satisfactory result has been obtained though, then the code should be committed to the CVS repository. At this point other developers might also have been

working with the code so an update should be performed on your working directory source. If no other modifications conflict with the files that you are updating then commit your changes to the CVS. Conflicts should be resolved by communications with the development team.

The cyclic development pattern described above, requires some discipline. Generally local files should be committed only after some certainty is achieved in accomplishing some predefined goal or task. At the same time commits to the CVS should not be delayed so long as to make one uncertain about what exactly is being committed. Notes taken between long commit times, can help to remind you of exactly which files you have modified. Use these notes appropriately. Focus only on source code files that you need to edit to achieve your goal or task. Please do not make a hay day[1] of the project source code files.

3.3 Getting Started

3.3.1 CVS

In order to get the NA_WorkSheet files on your system you should get to know CVS at the SourceForge site. When you signed up as a register user with SourceForge an account was created with the Project Shell Server. Access to this account can be obtained through a ssh, secure shell, client. A common one that is freely available is PuTTY. The address for the SourceForge shell server is shell.sourceforge.net. When you connect up to the shell server you will be in a UNIX type of OS environment. All commands take place in a command prompt medium. Some common ones are:

- pwd - Present working directory.
- ls - List current directory contents.
- cd - Change directory, just like in DOS.
- cp - Copy file, or files.
- rm - Remove files/directory. Careful there is no trash can on this system.

[1] If you do not know what a hay day is go get a pitch fork and move a pile of hay from one place to another. It can get messy especially if its windy.

- man "Command" - Help information on a command.

The NA_WorkSheet project files can be created in your home directory, by checking out a working copy from CVS. If you are not yet a developer than you can do so anonymously from the pserver. At the shell console type in the following command when logged into the SourceForge shell server:
"cvs -z3 -d:pserver:anonymous@na-worksheet.cvs.sourceforge.net:/cvsroot/na-worksheet checkout na-worksheet3.0"
In your present working directory you should now have a directory named na-worksheet3.0 with all the files from the CVS repository. In the future the only part of this command that might change is the module name, na-worksheet3.0. A check of the CVS files for the project can be done via a web browser at the project's home page with SourceForge. The top directory will always be the module name. If there is more than one top directory than consult with an admin to understand which branch of the source code you should use.

An anonymous checkout of the NA_WorkSheet project files will not allow you to commit any changes back into the CVS repository. Inform an admin of your desire to contribute source code and after a review someone will commit those changes for you. Once you have made a contribution then you will be assigned as an official developer. As an official developer you can then make updates yourself to the project's source code.

A developer, needs to checkout the project's source code from the CVS repository in a slightly different way. First remove any remnants of the anonymous checkout by using the remove command. At the shell console window type in the command, "rm -r na-worksheet3.0". Then checkout the CVS project files with the command,
"cvs -z3 -d:ext:YOURUSERNAME@na-worksheet.cvs.sourceforge.net:/cvsroot/na-worksheet checkout na-worksheet3.0"
A password will be required this time to checkout the files, use your registered user password.

3.3.2 Downloading the Working Project Code

The working directory of the project source code should now be in your home directory with the SourceForge shell server. These files can be downloaded via a ftp client. The WinSCP client tool can do this for you. The address for the connection to the SF shell server is shell.sourceforge.net. Use the SFTP,

port 22, feature with WinSCP to get the files on your local system. The ftp client Filezilla can also be used for this purpose. A couple of the files that will be needed by the Eclipse IDE are normally hidden on a shell account. Any file starting with a period are hidden. Make sure you download the .classpath and .project files if you are going to use Eclipse. Now that a copy of the project's source code is on your local system you can begin to develop code with a chosen IDE.

3.3.3 Developing Code

The developing of code for the project can be done by choosing a task designated by the NA_WorkSheet team or altogether independently. If you would like to enhance or create a new feature feel free to do so. Any improvements to the worksheet is appreciated. The goal of the NA_WorkSheet from the beginning was to create a easy to use interface that could execute numerical algorithms that were presented in a academic environment. Java was chosen because it allowed the software to be available as a web applet and a more versatile application on a multitude of platforms. The NA_WorkSheet was not created to compete with commercial applications along the same lines, but to a limited audience of educators and students. It so happens though that the application might be used outside of this context to solve a limited range of problems.

3.3.4 Source Code Inclusion

Once code has been developed, then to be included with the project's CVS repository it needs to be made available. If you are not yet a developer e-mail an admin to inform the team of a wish to contribute code, hopefully you will already talked with the team about such a contribution. Source code files can be e-mailed to an admin via a standard client which supports attachments. The SourceForge messaging script does not support attachments. A paste would have to be made into the text body. The e-mail address for and admin will be available from the project's SourceForge page or the home web page.

If you are a developer than the source code you have created can be uploaded directly to your working directory at the SourceForge shell server. Just use the WinSCP or Filezilla programs to do so. The WinSCP tool provides a easy way to compare your local system files to the remote location. In this way you should be able to easily identify files that have been modified.

Of course you should already know this, but it serves as a good sanity check.

3.3.5 Updating and Committing Code as a Developer

An official developer for the project should be able to commit code to the CVS repository without the help from the team. SourceForge provides documentation that can help you get starting using CVS. Look under the docs section of SourceForge to find this documentation. The basic commands that you should become familiar with are update, add, and commit. All these activities will be performed on your SourceForge shell user account so use PuTTY to login.

With multiple developers working on a project an update should be performed periodically to see if others are changing code that might compromise your efforts. Do an update before you commit especially if you are working with existing files. If you are adding files then the importance lessens. To perform an update of the entire module change to the working module directory, "cd na-worksheet3.0". Execute the the CVS command "cvs update". CVS will cycle though the project's repository making an comparison of your working copy to its own. Any differences will be highlighted. The update command can also be used to just make an adjustment to a single file or directory in the module. See the documentation for CVS, or try "info CVS" when at your shell console account. "q" quits this latter.

Now to actually make an addition to the project's source code repository, use either the add or commit command. If you are adding a new class then make sure the file is in the appropriate folder and enter "cvs add 'CLASSNAME.java'". CVS will make the note of the addition and inform you of the need to make a commit to finish the inclusion of the file into the repository. So to finalize the operation type in the command "cvs commit -m 'MESSAGE ABOUT THE COMMIT' 'CLASSNAME.java'". The -m option should be used to give others a brief idea of what modifications were made to the file. Please see the documentation section to gather more information about recommended requirements about log_messages. The commit option can be used to modify more than one file or an entire directory, please see the documentation.

3.4 Approximation Algorithm Creation

The creation of an algorithm class is relatively easy. The difficulties normally arise in the processing of output data and obtaining input. If the input variables are few than the input can normally be taken directly from the input panel for the algorithm. If the input requirements are say for a linear system solver than it would be tedious for the user to input a large number of variables and input should be obtained from an import option also. Consider these aspects when creating your algorithm.

The steps given below can be used as a basis to incorporate a new approximation into the worksheet. Once you have created the algorithm and input panel classes and would like help in integrating it into the worksheet please ask.

1. Follow the given Algorithm Template sample using it as basis to construct your class.

2. Create an input panel using the Algorithm Input Panel Template sample. If this sample does not meet your needs then look for another that relates more to your requirements.

3. Edit the MethodPanel class to include your approximation's input panel in the CardLayout.

4. Edit the MainInputPanel class to create a new radio button for your algorithm in the appropriate category.

5. Edit the MethodPanelUpdate class so that your panel will be brought up once it is selected through the MainInputPanel radio button selection process. The only trick here is to use the same String that was used to create the new card. Also here is where an ID sequence number for your algorithm can be obtained. Just increment the last one used by one.

6. At this point you should be able to run a recompiled version of the worksheet and be able to bring up your algorithm's input panel.

7. Finally edit the ExecuteApproximation class to include your algorithm to be executed when it is selected. Again use the same string that was

used to create your radio button in the MainInputPanel. I like to keep the card and radio button strings the same.

8. Test your algorithm against know input/output data.

9. Check any boundary conditions. Make sure that only valid data is excepted.

10. Congratulations you have contributed to the project. Thank you.

3.5 Documentation

Documenting code should always be done to help you and others understand what is being performed. It is extremely helpful when later modifications take place on the program. The NA_WorkSheet has been idle several times for extended periods. Each time I came back to continue work I was able to relatively easily figure out what I had done and begin again. The templates for developers in the docs section of the project provide a basis for what I usually do for comments within code. The header comment at the beginning of each class is required by the project. Please include this section in your code which encompasses all the way down to and embodies the Revision History. Put your own information as you deem necessary. The version history provides a easy means for others and yourself to quickly see what level of source code is being dealt with. If you edit an existing class please update the revision history. I use the revision history comment as the basis for the message_log when committing to the CVS repository.

Each class that exists in the project has a short JavaDoc comment at the beginning of the class to provide an inclusion list. The code at this time does not embrace the JavaDoc standard. The rest of the commenting within the code files generally are limited to each method and major blocks of code. Rarely is there more unless a specific routine is doing something that is a little complicated. If you get a feeling that what you are doing is a little strange and you might not remember it in the future provide a note. The actual approximation class that is implementing a specific algorithm should be documented at the beginning of that algorithm with a reference to its origin. Please see the sample algorithm class template.

Some other areas that provide a more organized readable code is the use of indentation and line breaks. Each block of code should be indented. Line breaks between methods help to highlight them. One other major difference

that you will recognize in my templates is the use of curly braces in blocks. The left curl goes after the method, class, or block not on the same line. The habit came from a requirement by a professor in a Pascal programming class. Document/comment as you go, otherwise it just usually gets left by the wayside. If you think that you do not have to comment than I guarantee you that at some point in a programming career a lot of people are going to be pissed off when they try to modify your code. Also no matter how brilliant you are, age is going weaken that youthful mind and recollection is going to diminish.

4 Summary

The developer guide has tried to provide a means to answer most of the questions that may be asked by a new developer. The guide tries to generally cover the overall structure of the NA_WorkSheet and how to go about contributing to the project. Each individual will have there own preferences to tools and techniques. This document just highlights one particular way to get your source into the CVS repository to be shared by others. The focus has taken a more manual approach than might be required. Many use WinCvs to accomplish the task more efficiently. The Eclipse IDE also supports remote repositories for managing your code in a team. Templates have been provided for the specific case of sharing an additional algorithm along with integrating the approximation into the worksheet. As a developer you should understand the basic concepts of CVS and be able to commit code. All developers should form the habit of creating legible code with comments that others can understand.