UNIVERSITY OF ALASKA ANCHORAGE

College of Arts and Sciences

Department of Mathematics and Statistics

Pensive: An Application for Visualizing Continuous Spectral Content of Volcanic Seismicity

by

Tom Parker

Supervisor:

Prof. Kirk Scott, PhD

A CAPSTONE PROJECT SUBMITTED TO THE DEPARTMENT OF MATHEMATICS AND STATISTICS, AT UNIVERSITY OF ALASKA ANCHORAGE, FOR THE DEGREE OF BACHELOR OF SCIENCE.

Anchorage AK, April 2015

©Copyright 2015
by
Tom Parker

tparker10@alaska.edu

Version 0.9

Abstract

The aim of this applied software development capstone is to produce a new application for visualizing, in near real-time, continuous seismic waveforms from a large number of sensors deployed on active volcanoes. The application is intended to be used in routine volcano monitoring operations on a daily basis, where it will assist in the assessment volcanic unrest.

The developed application presents the user with a mosaic comprised of individual tiles showing the spectral content of waveforms at a volcano. Each tile can be viewed individually with a higher resolution to have a better view of a shorter time span.

This combination of views permits an analyst to quickly scan an entire day of data, while still allowing a closer look at specific events of interest

Design requirements were designed to minimize cost of the infrastructure required without sacrificing flexibility. This goal was met allowing the application to run anywhere a seismic wave server is accessible.

Acknowledgements

This capstone would not have been possible without the kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

I am highly indebted to Professor Kirk Scott for his guidance and constant supervision throughout this project. I also want to thank Professor Adriano Calvacanti for patience and support while completing this project.

Several organizations helped with testing of Pensive, without their assistance and insightful feedback Pensive could not have been successful. I would like to the scientists at Alaska Volcano Observatory, Hawaii Volcano Observatory, Cascades Volcano Observatory, Servicio Geológico Colombiano, Pacific Northwest Seismic Network, and National Earthquake Information Center.

Most importantly, I want to thank the authors of the libraries Pensive relies on. Without their contribution to open source software Pensive and many other applications would not be possible.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Spectral analysis of seismic signals has proven to be a useful method of distinguishing signals indicative of increasing volcanic unrest from other sources of quasi-continuous ground vibrations, such as microseisms and wind-generated noise[14]. Figure 1.1 shows a spectrogram generated during a missile launch from Kodiak Island observed on instruments maintained by Alaska Volcano Observatory. Figure 1.2 shows volcanic tremor observed on instruments maintained by Alaska Volcano Observatory. During the 1991 eruption of Mt. Pinatubo, spectral analysis of seismic signals allowed scientists to characterize phases of the eruption[12].

Existing tools for continuous spectral analysis are either obsolete or encumbered by licensing or infrastructure requirements preventing wide-spread adoption. This project will produce a new cross-platform application which permits volcano observatories world-wide to integrate spectral analysis of volcanic seismicity into their daily monitoring operations.

This aim will be accomplished by breaking the project into the following distinct objectives.

- Minimize number of installation steps

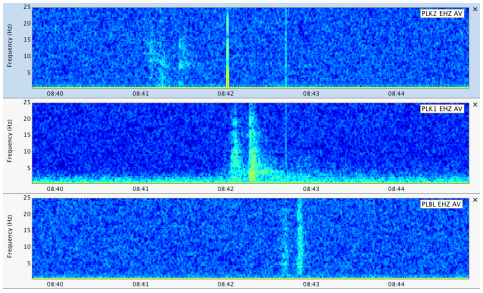

Figure 1.1: Missile launch from Kodiak recorded on seismometers installed on Peulik

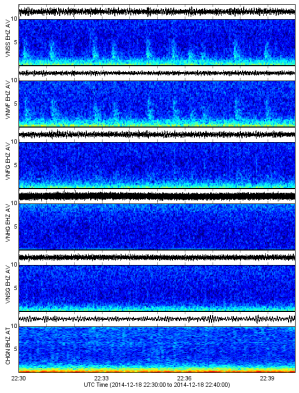

Figure 1.2: Low frequency tremor at Veniaminof Volcano

- Minimize complexity without sacrificing flexibility
- Run as a persistent application without relying on an external scheduling system
- Provide visualization of data without a dedicated application.
- Provide navigation of data both temporarily and spatially

1.2 Existing applications

Several applications have been created to facilitate analyzing the spectral content of seismic signals. They have worked well in the organizations they were created for, but none have gained wide-spread adoption. I believe this is because none of them fully meet the objectives of this project. Table 1.1 compares existing applications with the budget and infrastructure limitations Pensive intends to fulfill.

System	Available without fee	Usable without server infrastructure
MATLAB/PHP	☐	☑
sgram	☑	☐
SSAM	☑	☐
Java/Static HTML	☑	☑

Table 1.1: Comparison of existing applications

1.2.1 SSAM

SSAM, Seismic Spectral Amplitude Measurement, was a tool used by responders to the 1989-1990 eruption at Redoubt Volcano[14] and the 1991 eruption of Mt. Pinatubo[12] . The application ran on PC-based hardware with a digital signal processor and the MDETECT software package. The output provided by the application was useful and showed the promise of the technique, but was limited by the available technology of the time. Figure 1.3 shows an image produced by SSAM and used for analysis during the 1991 eruption of Mount Pinatubo.

1.2.2 sgram

A successor to SSAM was created by Northern California Seismic Network at Menlo Park, California. Their new tool, sgram[7], produces a color spectrogram of a single channel without the need of a dedicated hardware DSP.

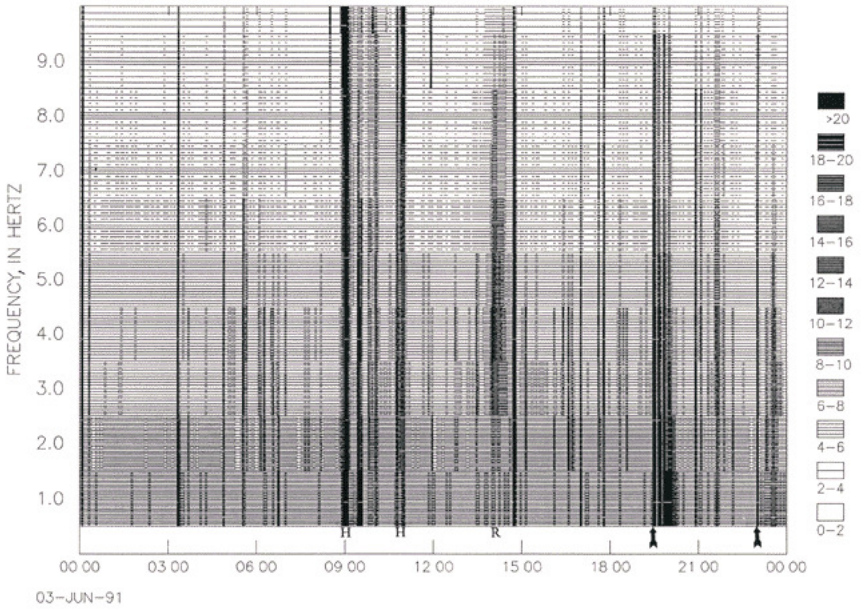

Figure 1.3: SSAM record

Figure 1.4 shows a 12-hour sgram record for a single seismic component of an instrument on Sotara volcano.

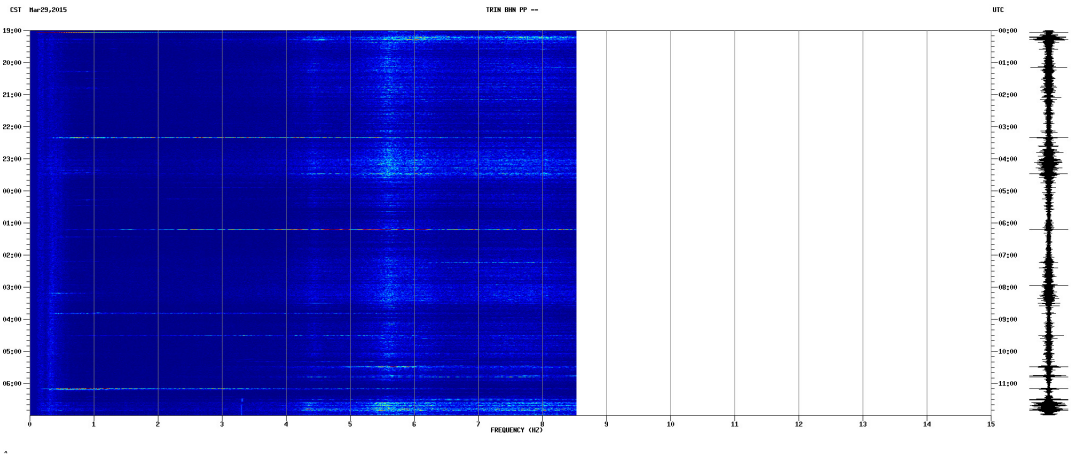

Figure 1.4: sgram plot from Servicio Geológico Colombiano

1.2.3 matlab spectrograms

The Geophysical Institute at University of Alaska Fairbanks, created an application which stacked spectrograms for an entire subnet of seismic stations on a single plot. Their implementation used matlab to produce the plots and PHP to display server them to a web browser. This implementation is still in use at Alaska Volcano Observatory. Figure 1.5 shows a spectrogram mosaic for 3 hours at Redoubt volcano.

To allow the application to both gain widespread acceptance and be deployed quickly in response to a volcanic crisis, it is necessary that an individual scientist be able to run the application without requiring the assistance of a professional Systems Administrator. Further, pricing for commercial software capable of producing the plots is prohibitively expensive for developing countries and smaller government organizations. Matlab currently costs $2150 for a single user license[18].

1.3 Functional specifications

1.3.1 Application

Installation of Pensive must require minimal effort on the part of the user. Pensive should be made available in a single archive file with no other dependencies to install separately. It will be assumed that the user already has a Java virtual machine installed. The pensive application will be responsible for populating any directories that are required. The user must not be expected to copy files on behalf of the application.

Configuration will require no more than one file. Reasonable defaults and inheritance will be used to permit the configuration to be as concise as possible. While only a single file will be required, users will have the option to separate the configuration into multiple files. This will permit more complex configurations while keeping the configuration maintainable.

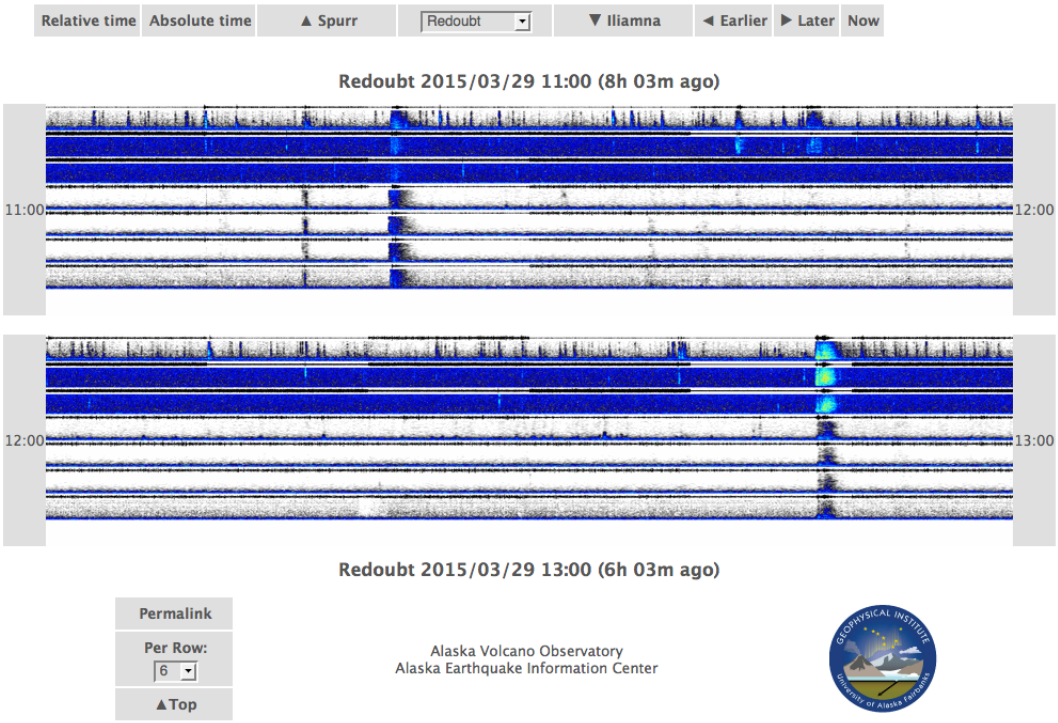

Figure 1.5: Spectrograms courtesy of Geophysical Institute at University of Alaska Fairbanks

Pensive will run persistently, handling the scheduling of plots internally. This is intended to avoid the need to rely on an external scheduling service provided by the operating system, such as cron.

1.3.2 Data visualization and navigation

Visualization of the data must be simple and intuitive. The user must not be required to use a dedicated client to view and navigate the plots, nor can Pensive require that server infrastructure be available.

Data should be navigable both temporally and spatially. The user should be able to easily move forward and backwards through time as the data is viewed. The user should also be able to move to adjacent volcanoes at the

same time period.

1.4 Project development requirements

During development, the project will need access to real-time seismic data and a repository for the source code. While many publicly-facing seismic wave servers would meet the need, the server provided by the Alaska Volcano Observatory was selected. GitHub was selected to host the source code repository. GitHub provides a well-known service at no cost[5].

Chapter 2

System Integration and Modeling

2.1 Technology

Pensive is written in Java and complies with Java SE 7 as specified in JSR 336[3]. The Java language provides the required cross-platform capabilities and Java 7 and while not the most recent version, ensures that the application will be able to run on the largest number of systems. Including parts of the world where current technology lags behind, such as developing countries where obsolete operating systems are still in use.

Pensive relies on an external server to provide seismic wave data. Both Earthworm WaveServerV and Winston Wave Server are supported. These two servers were selected because they are both freely available and have wide-spread adoption in the seismological community.

Plots produced by pensive will be written to disk in the Portable Network Graphics (PNG)[13] format. This format is well supported, available without intellectual property encumbrances, and is well suited to the type of graphics present in the plots.

Display and navigation of the plots relies on Dynamic HTML. Using

Dynamic HTML allows the images to be viewed directly on the system that the java app runs on, or remotely through a web server.

2.2 Design

2.2.1 Top-down Design

Pensive is a persistent java application, which wakes up every 10 minutes to request seismic data from an external source and plot the returned data. The plots reside on a data store allowing later viewing using a web browser. Figure 2.1 shows this data flow.

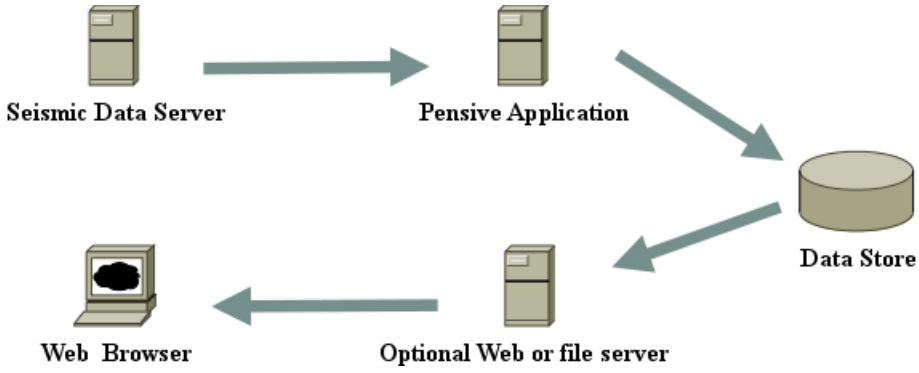

Figure 2.1: Pensive data flow

Pensive supports two seismic data server protocols. Data may be retrieved from either a Winston Wave Server or an Earthworm WaveServerV. The plots produced will be the same regardless of the protocol used to retrieve the data. Pensive will write its plots to the specified location in the local filesystem. The may be either a local filesystem or a filesystem mounted from a remote server.

Primary configuration is performed through a text file. The file should be simple to ease make it easy to run Pensive the first time. However, it also must be very flexible to allow the plots to adapt to varying instrumentation,

network design, and user preferences.

2.2.2 Bottom-up Design

Pensive is intended to support the production a large number of plots on a continuous basis, each of which will require data to be retrieved from a server. These data requests may be prohibitively slow if performed sequentially. Pensive must be able to process these requests concurrently. However, sending an excessive number of concurrent requests to a single server may impact other clients of that system. To balance these needs, a bounded pool of connections to each server will be maintained. To ensure plots are produced in a time-ordered sequence, each plotting job will be added to a synchronized priority queue based on the start time of the plot. Figure 2.2 shows an example of threads created with 3 connections to one wave server and 2 connections to a second wave server.

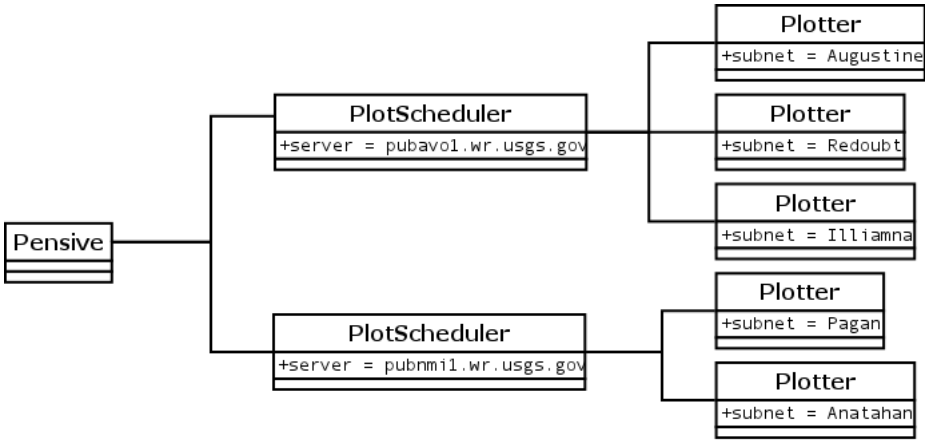

Figure 2.2: An example of threads created with 3 connections to one wave server and 2 connections to a second wave server.

Pensive will produce an unbounded number of plots. To make archival and navigation of these images simple, they will be placed into hierarchy based on date. The specific structure of the hierarchy will be configurable

by the user using Java SimpleDate format strings.

The initial project schedule is shown in Figure 2.3. It is expected to evolve over the course of the project.

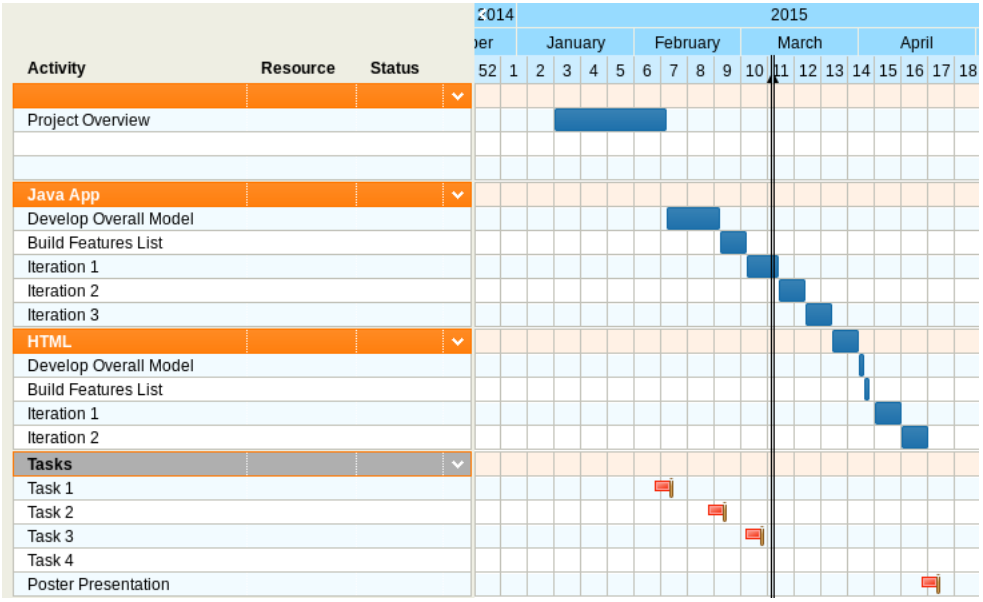

Figure 2.3: Project Schedule

2.3 Components Used in Development

Development of Pensive was performed using Eclipse IDE for Java Developers, an open source Java integrated development environment[16].

Documentation of the Pensive code is created using the freely available Javadoc[2] tool, which parses the declarations and documentation comments in a set of source files and produces a set of HTML pages describing the classes, interfaces, constructors, methods, and fields. Model diagrams were created with Dia. Dia is an application for creating technical diagrams. Its interface and features are loosely patterned after the Windows program Visio. Features of Dia include multiple-page printing, export to many formats (EPS,

SVG, CGM and PNG), and the ability to use custom shapes created by the user as simple XML descriptions. Dia is useful for drawing UML diagrams, network maps, and flowcharts[8].

In addition to the development tools, several libraries were used in constructing Pensive.

JSAP

JSAP - Java Simple Argument Parser, syntactically validates command line arguments and converts those arguments into objects[9].

FreeMarker

FreeMarker is a template engine designed to be practical for the generation of HTML Web pages.[4].

jQuery

jQuery is a fast, small, and feature-rich JavaScript library designed to make things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler[17].

leanModal.js

leanModal.js is a lightweight javascript library designed for producing modal dialog boxes in javascript[15]

usgs.jar

usgs.jar is a library used by projects designed at the Alaska Volcano Observatory. Plotting and data retrieval rely on the USGS Java libraries. These libraries provide existing well-tested functionality to retrieve seismic data and produce plots.

2.4 Coding Methodology

2.4.1 Agile

Agile coding methodology emphasizes iteration and flexibility over more traditional software development methodologies. Agile grew out of a meeting of developers which produced The Agile Manifesto[8] to document the four core values which define Agile. Figure 2.4 shows the Agile lifecycle.

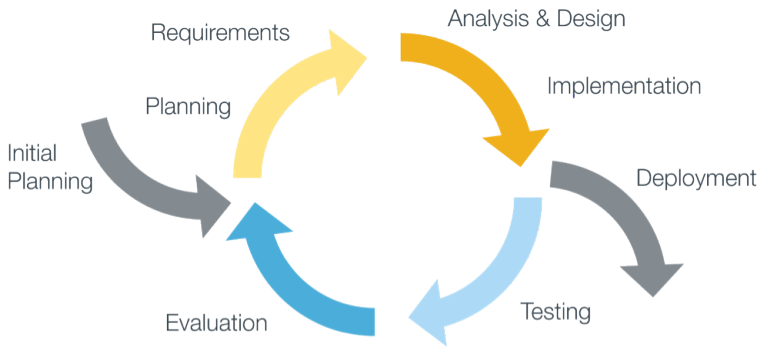

Figure 2.4: Agile lifecycle. Copyright iMedia Communications Inc.

2.4.2 Feature Driven Development

This project used the Feature Driven Development methodology as described by Jeff De Luca and Peter Coad in Java Modeling in Color with UML[1]. Feature Driven Development is an agile methodology described as a "client-centric, architecture-centric, and pragmatic software process"[1]. Figure 2.5 shows the five activities which iteratively drive the process. This methodology was selected because its combination of early model development followed by iterative feature development suited the needs of this project particularly well.

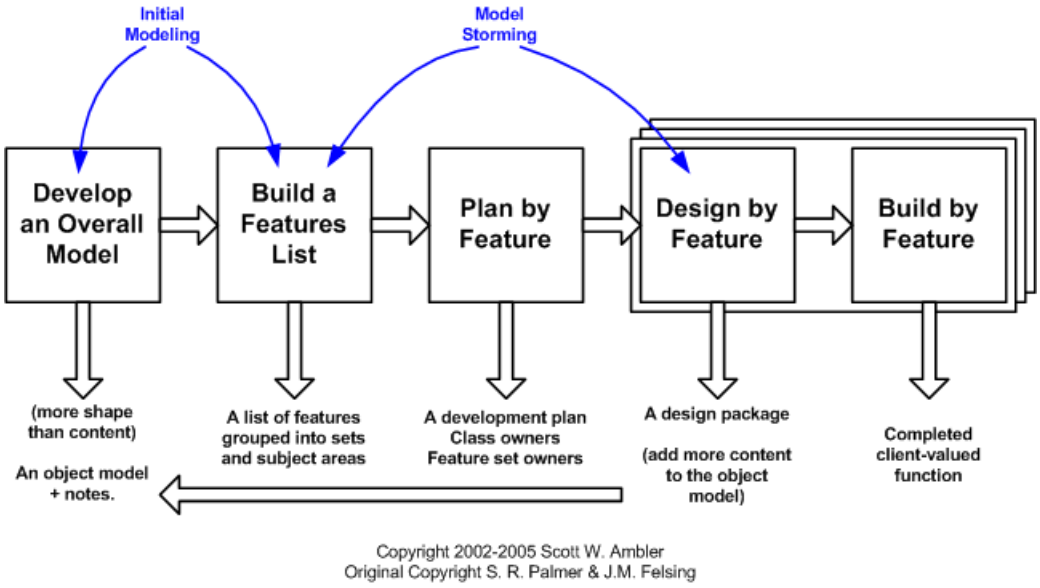

Figure 2.5: Feature Driven Development activities

Chapter 3

User Interface and Testing Methodology

3.1 User Interface

Users will primarily interact with the Pensive application through its configuration file. The complexity of Pensive's configuration will vary with the requirements of its users. A short concise configuration will work for many users, while users desiring a greater degree of customization are also supported. Users will interact with the output generated by Pensive through a single page application, viewed in their web browser.

Pensive users will view and navigate the plots through a single page application, viewed in the web browser of their choice. The single page application can be used either locally or remotely.

3.1.1 Configuration

Pensive is configured through a text configuration file, containing an unordered list of case sensitive [key]=[value] pairs. This list is parsed into a tree whose child nodes inherit values from their parent. Figure 3.1 shows the

structure of the configuration tree.

Configuration of Pensive is performed using a single configuration file named pensive.config. This file contains a hierarchical set of configuration stanzas Figure 3.1 shows the tree layout. This permits both a very expressive configuration and also a very simple configuration, depending on the users need.

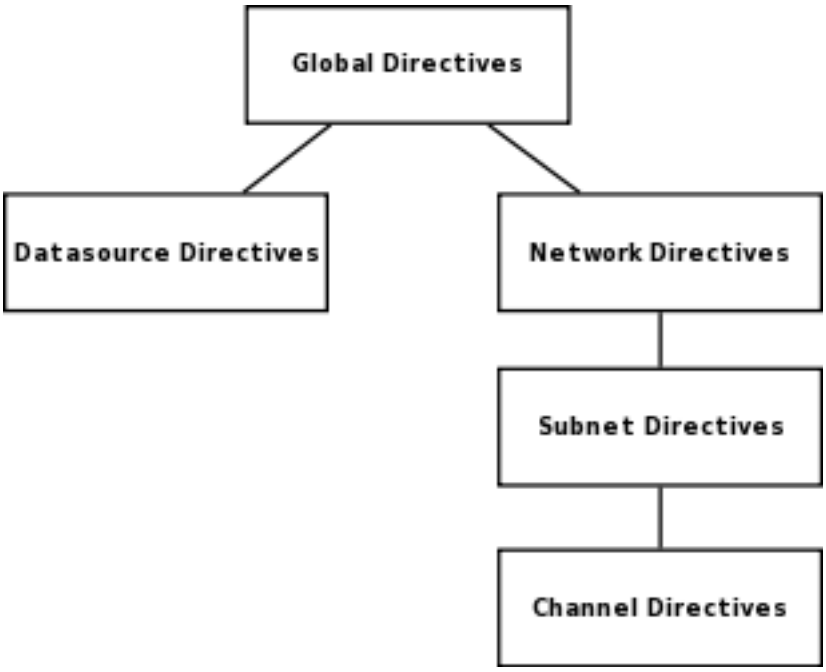

Figure 3.1: Configuration Tree

3.1.2 Plot display

Display and navigation of the plots produced by Pensive is done through a web browser. The user will either load the Pensive HTML page by opening it directly on their workstation or through a URL pointing to a remote server. The Pensive page is divided into three sections. The top bar contains buttons to select the overall state of the display; a middle bar that displays the subnet

and timespan shown, along with buttons to move to an adjacent display either spatially or temporally; and the main plot display.

The top bar provides a provides controls to set the overall state of the display. Selection lists are provided to choose the network and subnet that is displayed. Following the selection lists are three buttons labeled "Absolute time", "Permalink", and a third that is dependent on the type of plot displayed.

The Absolute time box sets the start time of the displayed data. The button will display a modal dialog box with a single text input field and a submit button. Pensive populates the text files with a properly formatted example time to provide the user direction on an acceptable input format. Pensive parses the input field as each character is entered and provides feedback by changing the color of the text and disabling the submit button if the filed is not valid.

Pensive uses dynamic HTML to display plots and respond to users input. At the top of the page is a series of buttons which allows the user to

the single window, Pensive has two display modes; mosaic and single image. Both modes use a consistent header bar which allows the user to choose a different network or subnet to view, without altering either the display mode or time span displayed.

The mosaic display (see Figure 3.2) mode allows the user to quickly scan several hours of data at once. This mode displays the name of the subnet at the top of the page with buttons on either side used to switch to the previous or next subnet in the list. Below the subnet name is the time span displayed. This line has buttons on either side of the timespan allowing the user to move forward or backwards in time, along with a third button which moves to the most recent time span.

Below the time span, there are a series of rows of images. Each row begins with the time of the first sample displayed on the plot and ends with the first sample not shown. This method of displaying the end time was selected to

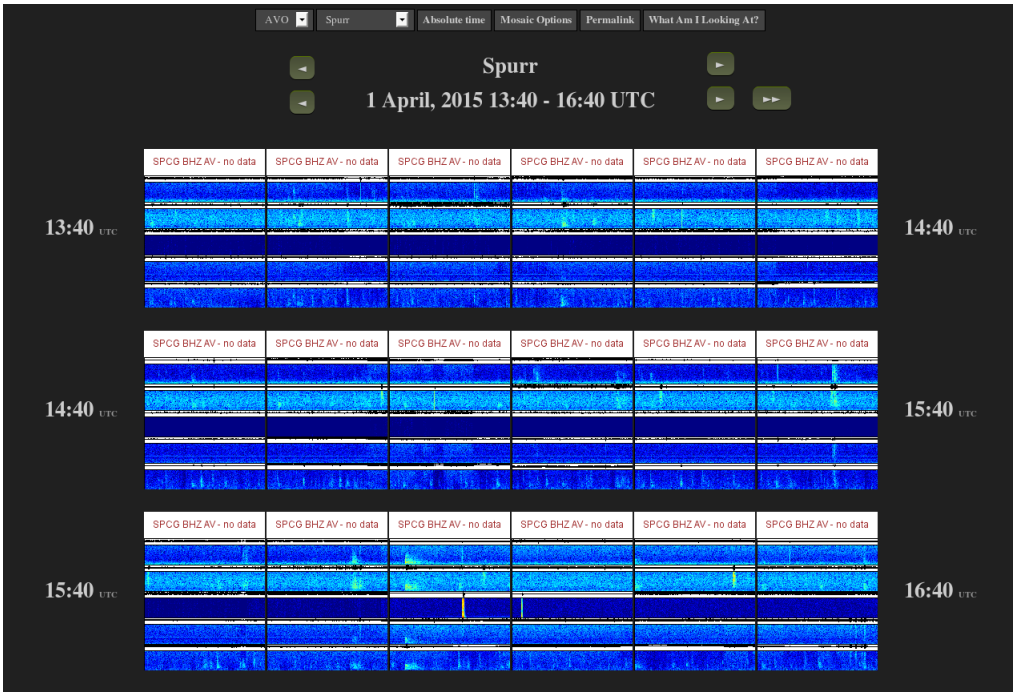

Figure 3.2: A 3-hour mosaic of Spurr

avoid unnecessarily precise time displays and to provide continuity between rows. Between the timestamps is a mosaic of thumbnail images of the plots without axis decoration. These are presented with a thin line between images to help in estimating time within the row.

The single plot display (see Figure 3.3) allows the user to take a closer look at a short time span. The header is similar to that of the mosaic display. In this mode, a single image is displayed with fully decorated axes.

3.2 Testing Methodology

Testing of Pensive was divided into classifications by purpose[11]. These classifications were correctness, performance, reliability, and security.

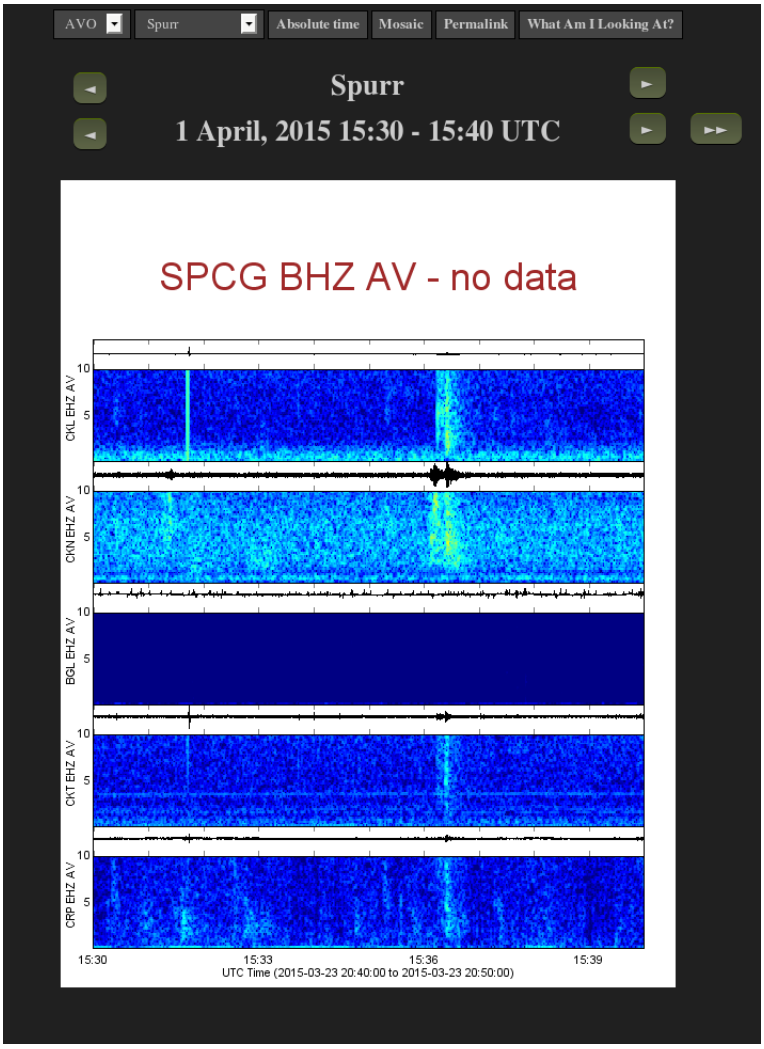

Figure 3.3: A 10-minute plot of Spurr

3.2.1 Correctness Testing

Correctness is the minimum requirement of software, the essential purpose of testing[11]. Pensive underwent a combination of black-box and white-box correctness testing to verify that the application works as expected with differing configuration scenarios and that it met the needs observatory scientists and was useful in monitoring volcanic activity.

Black-box testing is driven by the applications functional requirements and data, without consideration of the underlying structure of the code[11]. Black-box testing was performed by putting Pensive into use at Alaska Volcano Observatory. Pensive was configured to produce plots for every monitored volcanic region in Alaska and allowed to run for several weeks. During this time, observatory scientists regularly used Pensive in their daily monitoring activities. Feedback from the users was integrated into the application and testing was completed with no unresolved bug reports.

White-box testing is performed with knowledge of the structure of the code[11]. White-box testing was performed by creating several configuration files, attempting to exercise the full range of configuration options the Pensive supports. Configurations including invalid or nonsensical directives were included to assess how Pensive handles error conditions. While there is room to improve feed back to the user, Pensive handled all configurations in an acceptable manner.

3.2.2 Performance Testing

Scalability was an important goal for Pensive. Users should be able to configure as many networks and subnets required to cover the volcanoes they monitor. This goal requires Pensive to use system and network resources efficiently.

Pensive was configured with two networks which had a combined total of 27 subnets, containing plots for 148 channels of continuous seismic data. This configuration was allowed to run for several days and monitored for both excessive instantaneous resource consumption and increasing resource consumption over time. It is expected that disk space is the only resource need which should increase with time.

A full compliment of subnets for the Alaska Volcano Observatory was configured and run on a modest desktop computer. Pensive was started and allowed to run continuously for one week. During this period the application

was profiled using the YourKit Java Profiler[2]. The heap size of the running JVM was watched over a period of two weeks. After 8 days heap consumption had remained nearly constant as shown in Figure 3.4. Thread count remained constant as shown by Figure 3.5

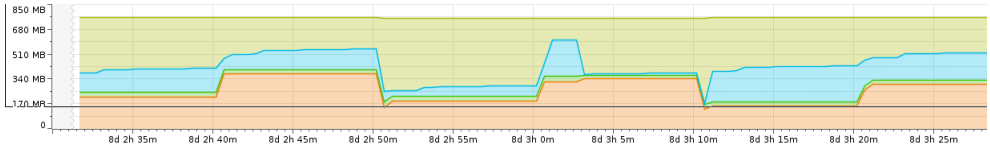

Figure 3.4: Heap usage after 8 days

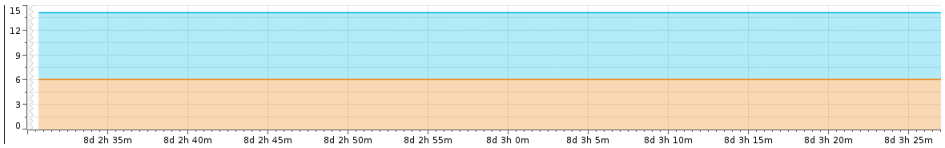

Figure 3.5: Thread count after 8 days

Network performance was analyzed with the profiler and by inspecting requests received by the upstream server. The profiler showed that no network traffic was generated when Pensive was not requesting data to plot. An analysis of requests received by the upstream server showed that each request was received only once and requested only the data that was necessary to produce the plot.

3.2.3 Reliability Testing

Reliability testing of Pensive focused on how the app behaved when faced with various failures of its network and upstream data source. The app was run continuously for several days as its environment changed and its response was monitored.

The app failed gracefully when its upstream data provider became unreachable or unresponsive, and recovered on its own when those conditions cleared up.

3.2.4 Security Testing

The portions of Pensive vulnerable to attack have intentionally been kept small. Care was taken to make the system robust and all known security issues have been addressed.

Pensive will do its best to obey the provided configuration file. This was a deliberate design decision to allow maximum flexability to Pensive's users. However, this requires that the configuration file be appropriately protected by the underlying operating system.

The Pensive web application accepts HTTP form variables to set the initial state of the display. These variables are sanitized using a white list permitting only characters appropriate for each field. Specifically, Pensive was designed to avoid vulnerability to cross-site scripting[10] attacks.

3.3 Agile Project Management

Agile project management attempts to integrate the philosophy of Agile development into the project management process. Agile Leadership Network, an organization dedicated to applying agile leadership principles and values[1]. In their Declaration of Interdependence[6] they describe the goals of agile project management as:

- We *increase return on investment* by making continuous flow of value our focus.
- We *deliver reliable results* by engaging customers in frequent interactions and shared ownership.
- We *expect uncertainty* and manage for it through iterations, anticipation, and adaptation.
- We *unleash creativity and innovation* by recognizing that individuals are the ultimate source of value, and creating an environment where

they can make a difference.

- We *boost performance* through group accountability for results and shared responsibility for team effectiveness.
- We *improve effectiveness and reliability* through situationally specific strategies, processes and practices.

This project did not use formal agile project management.

Chapter 4

User Manual

4.1 Introduction

4.2 Configuration

4.2.1 Configuration file format

Pensive is configured through a text configuration file. The name and path of the configuration may be passed as an argument on the command line when launching the application. If no configuration file is given on the command line, Pensive will look for its configuration in a file named pensive.config in the current working directory.

The configuration file is an unordered list of case sensitive [key]=[value] pairs. This list is parsed into a tree whose child nodes inherit values from their parent. An entry's key describes its place in the tree. Leading and trailing whitespace is ignored, as are any lines beginning with a # character.

After whitespace is trimmed from the beginning of the line, all remaining characters up to the fist equals sign (=) will be the key.

Similarly, after whitespace is trimmed from the end of the line, all remaining characters following the first equals sign will comprise the value.

There is a special token that can be used to set values which span multiple lines. If the value is *begin-multiline*, the key will be comprised of the following lines up to a line containing only *end-multiline*.

A like containing the token *@include* followed by a file name will be replaced by the content of the named file. This can be used to separate the configuration into multiple files and may ease maintenance of large or complex configurations.

4.2.2 Global Directives

Pensive has several configuration directives which will apply to all aspects of its operation. The value of these directives cannot be overridden by children of the root node of the configuration tree.

debug If specified, this directive will increase the logging output of the covered sections. There is not default value for this directive.

pathRoot This is the root of the directory structure the plots are placed into. If unspecified, this will default to *html/*.

filePathFormat This is the string used to format paths to the plots. The strings follows Java's SimpleDateFormat. If unspecified, this will default to *yyyy/MM/dd*.

fileNameSuffixFormat This is the string used to format the file name suffix of plots produced. If unspecified, this will default to *_yyyyMMss-HHmm*.

writeHtml If true, Pensive will write a HTML file when launched. If false, Pensive will only produce plots. If unspecified, this will default to *true*.

selectedNetwork This is the network that will be selected by default when the Pensive HTML page is loaded. If unspecified, the default will be the first network listed in the configuration file.

waveSource A named source of seismic data. There will be one *waveSource* directive for each data source.

4.2.3 Data Source Directives

The Data source directives inform Pensive of known sources of seismic data. Each directive is prepended with the data source name provided in a matching *waveSource* directive, followed by a period. Directives that are specified without a data source name will apply to all data sources unless overridden.

type The type of data server. Can be either wws or wsv. If unspecified, the value will default to *wws*.

host The IP address, hostname, or fully qualified domain name of the server.

port The TCP port the server is listening on. If unspecified, this value will default to *16022*.

threads The number of concurrent connections kept open to the server. If unspecified, this value will default to *5*.

4.2.4 Network Directives

There are no network-specific directives. Directives may be prepended with the network name to indicate default values for all subnets in that network.

4.2.5 Subnet Directives

Subnet directives apply to all channels on a single plot. They may not be overridden by specifying individual channel values.

embargo Pensive will wait this many seconds after the end of a plotting period before requesting data. This directive is intended to allow Pensive to produce plots with complete data with as little latency as possible. If unspecified, this value defaults to *5*.

dataSource The name of the source of data for the subnet. This value has no default.

4.2.6 Channel Directives

waveRatio The ratio, in percent, of the height of a waveform plot to a spectrogram plot. If unspecified, this will default to *25*.

logPower If true, plot power on a log scale.

minFreq The minimum frequency to plot on the spectrogram.

maxFreq The maximum frequence to plot on the spectrogram.

minPower The minimum power to plot on the spectrogram.

maxPower The maximum power to plot on the spectrogram.

binSize The number of samples in each bin. If unspecified, this will default to *256*.

overlap The overlap used in the FFT calculation, specified as a decimal ratio to the bin size.

nfft The FFT length. If set to 0, the FFT length will match the bin size. If unspecified, this will default to *0*.

4.3 Starting and Stopping Pensive

Pensive is meant to run persistently. It will produce plots every ten minutes for as long as it's running. Start Pensive by executing pensive.bat on Windows operating systems, and pensive.sh on all other operating systems.

Start Pensive by executing pensive.bat on Windows operating systems and pensive.sh on all other operating systems. Most users will integrate

this into the startup actions on their operating system once they've verified correct operation and configuration of pensive.

While Pensive is running, it will print its status to STDOUT. To stop Pensive, press ctrl-C.

4.4 Viewing the data

The plots produced by Pensive are viewed and navigated in a web browser. Pensive does not require additional plugins to be installed or typical browser configuration settings to be changed. All modern browsers are supported in their default configuration. Pensive output may reside on either on a standalone workstation or on a central server.

If running Pensive on a stand alone workstation, use your browsers *open file* option to locate the file named index.html at the root of the directory pointed to by the *pathRoot* directive in the configuration file. If the *pathRoot* directive is not set in the configuration file, it will default to *html* inside of the working directory. Once the page is loaded, it may be bookmarked to simplify opening the page in the future.

If Pensive is being run on a server, the plots may be viewed through a web browser or a file server. In either case, set the *pathRoot* directive to place Pensive's output in an accessible location on the server and note it's location to be opened in a web browser.

Chapter 5

Conclusion

5.1 Summary

This capstone project was intended to produce an application which would permit volcanologists to integrate spectral analysis of continuous seismic data into their daily operational duties. Spectral analysis of seismic data has been demonstrated to be useful [14, 12] in forecasting near-term volcanic unrest. Extensive infrastructure requirements have limited adoption of existing tools for continuous spectral analysis. A tool is needed which is suitable for use by users who have budgetary or IT personnel constraints. Pensive is expected to fulfill that need.

Pensive is currently being tested in daily operations at the Alaska Volcano Observatory. Initial testing has been favorable. Positive feedback has been received by observatory geophysicists and there are no outstanding bug reports. A prototype of Pensive is in use at five other organizations with volcano monitoring responsibilities. Feedback from those organizations has been incorporated into the final application which is expected to be used in daily operations at those sites once testing has completed.

5.2 Future Development

Pensive is fully complete and suitable for use, but there is always room for improvement. Here I make some suggestions for future development.

The spectrogram plot does not display a color bar. The color bar is unnecessary when reviewing the plots for changes within a subnet but would be useful when comparing different subnets. Because the rage of the color bar is determined by the run-time configuration, it can not be a static image. One possible solution is to use a HTML 5 canvas element to dynamically create the color bar on the client.

Creating plots in near real-time works well most of the time, but there is no other way to create plots. Server availability limitations mean that sometimes the application will not be able to create plots in near real-time. Researchers may also wish to create a series of spectrograms for archival data either to reanalyze an old sequence or to assess how their application configuration would display known sequences of interest.

Two widely available seismic data servers are supported. This is sufficient for wide-spread adoption, but also leaves some users unable to use the application. Adding other data sources, including SEEDlink, would reduce the unserved population.

Pensive does not include scripts to automatically start the application at system boot. This is a simple task for large observatories and unnecessary for an individual researcher. However, there may be users between these two extremes who would benefit from having the scripts provided.

Pensive also does not include scripts to monitor it's health. Expanding the output to include a state of health file which could be monitored by a systems monitoring application would be useful for larger installations.

Appendix A

UML

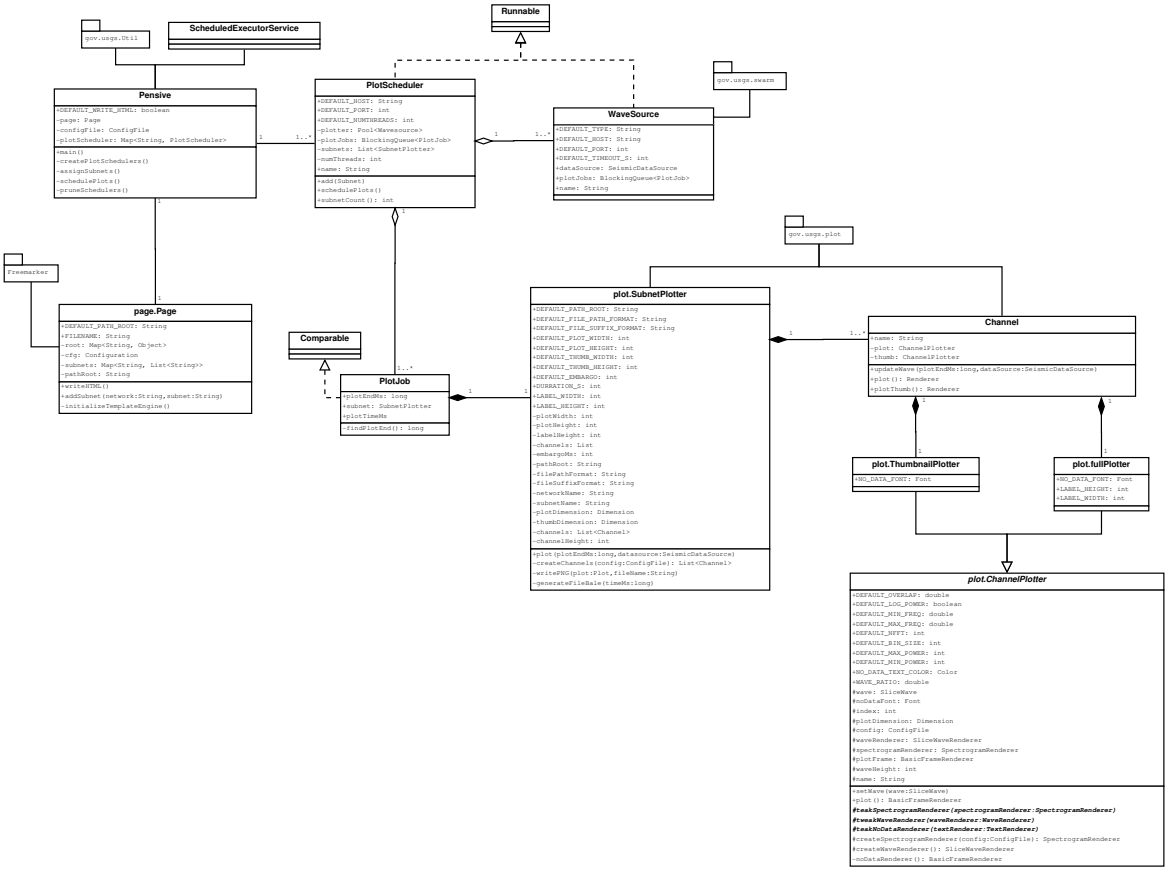

Figure A.1: Pensive Class Diagram

Appendix B

License

Creative Commons CC0 1.0 Universal

CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS DOCUMENT DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE USE OF THIS DOCUMENT OR THE INFORMATION OR WORKS PROVIDED HEREUNDER, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM THE USE OF THIS DOCUMENT OR THE INFORMATION OR WORKS PROVIDED HEREUNDER.

Statement of Purpose

The laws of most jurisdictions throughout the world automatically confer exclusive Copyright and Related Rights (defined below) upon the creator and subsequent owner(s) (each and all, an "owner") of an original work of authorship and/or a database (each, a "Work").

Certain owners wish to permanently relinquish those rights to a Work for the purpose of contributing to a commons of creative, cultural and scientific works ("Commons") that the public can reliably and without fear of later

claims of infringement build upon, modify, incorporate in other works, reuse and redistribute as freely as possible in any form whatsoever and for any purposes, including without limitation commercial purposes. These owners may contribute to the Commons to promote the ideal of a free culture and the further production of creative, cultural and scientific works, or to gain reputation or greater distribution for their Work in part through the use and efforts of others.

For these and/or other purposes and motivations, and without any expectation of additional consideration or compensation, the person associating CC0 with a Work (the "Affirmer"), to the extent that he or she is an owner of Copyright and Related Rights in the Work, voluntarily elects to apply CC0 to the Work and publicly distribute the Work under its terms, with knowledge of his or her Copyright and Related Rights in the Work and the meaning and intended legal effect of CC0 on those rights.

1. Copyright and Related Rights. A Work made available under CC0 may be protected by copyright and related or neighboring rights ("Copyright and Related Rights"). Copyright and Related Rights include, but are not limited to, the following:

i. the right to reproduce, adapt, distribute, perform, display, communicate, and translate a Work;

ii. moral rights retained by the original author(s) and/or performer(s);

iii. publicity and privacy rights pertaining to a person's image or likeness depicted in a Work;

iv. rights protecting against unfair competition in regards to a Work, subject to the limitations in paragraph 4(a), below;

v. rights protecting the extraction, dissemination, use and reuse of data in a Work;

vi. database rights (such as those arising under Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, and under any national implementation thereof,

including any amended or successor version of such directive); and

vii. other similar, equivalent or corresponding rights throughout the world based on applicable law or treaty, and any national implementations thereof.

2. Waiver. To the greatest extent permitted by, but not in contravention of, applicable law, Affirmer hereby overtly, fully, permanently, irrevocably and unconditionally waives, abandons, and surrenders all of Affirmer's Copyright and Related Rights and associated claims and causes of action, whether now known or unknown (including existing as well as future claims and causes of action), in the Work (i) in all territories worldwide, (ii) for the maximum duration provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or promotional purposes (the "Waiver"). Affirmer makes the Waiver for the benefit of each member of the public at large and to the detriment of Affirmer's heirs and successors, fully intending that such Waiver shall not be subject to revocation, rescission, cancellation, termination, or any other legal or equitable action to disrupt the quiet enjoyment of the Work by the public as contemplated by Affirmer's express Statement of Purpose.

3. Public License Fallback. Should any part of the Waiver for any reason be judged legally invalid or ineffective under applicable law, then the Waiver shall be preserved to the maximum extent permitted taking into account Affirmer's express Statement of Purpose. In addition, to the extent the Waiver is so judged Affirmer hereby grants to each affected person a royalty-free, non transferable, non sublicensable, non exclusive, irrevocable and unconditional license to exercise Affirmer's Copyright and Related Rights in the Work (i) in all territories worldwide, (ii) for the maximum duration provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or

promotional purposes (the "License"). The License shall be deemed effective as of the date CC0 was applied by Affirmer to the Work. Should any part of the License for any reason be judged legally invalid or ineffective under applicable law, such partial invalidity or ineffectiveness shall not invalidate the remainder of the License, and in such case Affirmer hereby affirms that he or she will not (i) exercise any of his or her remaining Copyright and Related Rights in the Work or (ii) assert any associated claims and causes of action with respect to the Work, in either case contrary to Affirmer's express Statement of Purpose.

4. Limitations and Disclaimers.

a. No trademark or patent rights held by Affirmer are waived, abandoned, surrendered, licensed or otherwise affected by this document.

b. Affirmer offers the Work as-is and makes no representations or warranties of any kind concerning the Work, express, implied, statutory or otherwise, including without limitation warranties of title, merchantability, fitness for a particular purpose, non infringement, or the absence of latent or other defects, accuracy, or the present or absence of errors, whether or not discoverable, all to the greatest extent permissible under applicable law.

c. Affirmer disclaims responsibility for clearing rights of other persons that may apply to the Work or any use thereof, including without limitation any person's Copyright and Related Rights in the Work. Further, Affirmer disclaims responsibility for obtaining any necessary consents, permissions or other rights required for any use of the Work.

d. Affirmer understands and acknowledges that Creative Commons is not a party to this document and has no duty or obligation with respect to this CC0 or use of the Work.

Appendix C

Source Code

Pensive.java

```
package net.stash.pensive;

import gov.usgs.util.ConfigFile;
import gov.usgs.util.Log;
import gov.usgs.util.Util;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;
import java.util.logging.Level;
import java.util.logging.LogManager;
import java.util.logging.Logger;

import net.stash.pensive.page.Page;
import net.stash.pensive.plot.SubnetPlotter;

import com.martiansoftware.jsap.JSAP;
import com.martiansoftware.jsap.JSAPResult;
```

```
import com.martiansoftware.jsap.Parameter;
import com.martiansoftware.jsap.SimpleJSAP;
import com.martiansoftware.jsap.Switch;
import com.martiansoftware.jsap.UnflaggedOption;

/**
 * An application to produce a continuous collection of subnet spectrograms.
 *
 * @author Tom Parker
 *
 *         I waive copyright and related rights in the this work worldwide
 *         through the CC0 1.0 Universal public domain dedication.
 *         https://creativecommons.org/publicdomain/zero/1.0/legalcode
 */
public class Pensive {


        public static final boolean DEFAULT_WRITE_HTML = true;
        public static final String DEFAULT_CONFIG_FILENAME = "pensive.config
            ";

        /** my logger */
        private static final Logger LOGGER = Log.getLogger("gov.usgs");

        /** my configuration file */
        private ConfigFile configFile;

        /** My SPA */
        private Page page;

        /** One plot scheduler per wave server */
        private Map<String, PlotScheduler> plotScheduler;

        // JSAP related stuff.
        public static final String JSAP_PROGRAM_NAME = "java -jar net.stash.
            pensive.Pensive";
        public static final String JSAP_EXPLANATION_PREFACE = "I am the
            Pensive server";

        private static final String DEFAULT_JSAP_EXPLANATION = "\n";

        private static final Parameter[] DEFAULT_JSAP_PARAMETERS = new
            Parameter[] {
                        new Switch("create-config", 'c', "create-config",
                                    "Create an example config file in
                                        the curent working directory."),
```

```
                    new Switch("verbose", 'v', "verbose", "Verbose
                        logging."),
                    new UnflaggedOption("configFilename", JSAP.
                        STRING_PARSER, DEFAULT_CONFIG_FILENAME, JSAP.
                        NOT_REQUIRED,
                                JSAP.NOT_GREEDY, "The config file
                                    name.") };

/**
 * Class constructor
 *
 * @param configFile
 *            my config file
 */
public Pensive(ConfigFile configFile) {

            this.configFile = configFile;
            long now = System.currentTimeMillis();
            configFile.put("applicationLaunch", "" + now);
            LOGGER.log(Level.INFO, "Launching Pensive at " + (new Date(
                now)));
            Logger.getLogger("global").setLevel(Level.OFF);

            page = new Page(configFile);

            createPlotSchedulers();
            assignSubnets();
            pruneSchedulers();

            boolean writeHtml = Util.stringToBoolean(configFile.
                getString("writeHtml"), DEFAULT_WRITE_HTML);
            if (writeHtml)
                    page.writeHTML();
    }

    /**
     * Create one PlotScheduler per wave server, each running in its own
         thread.
     */
    private void createPlotSchedulers() {
            plotScheduler = new HashMap<String, PlotScheduler>();

            for (String server : configFile.getList("waveSource")) {
                    ConfigFile c = configFile.getSubConfig(server, true)
                        ;
                    LOGGER.log(Level.INFO, "Creating plot scheduler for
```

```
                    " + server);
                plotScheduler.put(server, new PlotScheduler(server,
                    c));
        }
}

/**
 * Assign subnets to it wave server
 */
private void assignSubnets() {
        List<String> networks = configFile.getList("network");
        if (networks == null)
                throw new RuntimeException("No network directives
                    found.");

Iterator<String> networkIt = networks.iterator();
while (networkIt.hasNext()) {
    String network = networkIt.next();
    ConfigFile netConfig = configFile.getSubConfig(network, true);
    List<String> subnets = netConfig.getList("subnet");
    if (subnets == null) {
        LOGGER.log(Level.WARNING, "No subnet directives for network
            " + network + " found. Skipping.");
        networkIt.remove();
        continue;
    }

    Iterator<String> subnetIt = subnets.iterator();
    while (subnetIt.hasNext()) {
        String subnet = subnetIt.next();
        ConfigFile subnetConfig = netConfig.getSubConfig(subnet,
            true);
        if (subnetConfig.getList("channel") == null) {
            LOGGER.log(Level.WARNING, "No channel directives for
                subnet " + subnet + " found. Skipping.");
            subnetIt.remove();
            continue;
        } else {
            page.addSubnet(network, subnet);
        }

                        String dataSource = subnetConfig.getString("
                            dataSource");
                        PlotScheduler scheduler = plotScheduler.get(
                            dataSource);
                        LOGGER.log(Level.INFO, "Assigning subnet " +
```

```
                                subnet + " to " + dataSource);
                        scheduler.add(new SubnetPlotter(network,
                            subnet, subnetConfig));
                }
                netConfig.putList("subnet", subnets);
        }
        configFile.putList("network", networks);

        Iterator<String> schedulerIt = plotScheduler.keySet().
            iterator();
        while (schedulerIt.hasNext()) {
                String server = schedulerIt.next();
                PlotScheduler ps = plotScheduler.get(server);
                if (ps.subnetCount() < 1) {
                        LOGGER.log(Level.WARNING, "No subnets
                            feeding from " + ps.name + ". I'll prune
                             it.");
                        schedulerIt.remove();
                }
        }
}

/**
 * Prune PlotSchedulers that have no subnets assigned to them.
 */
private void pruneSchedulers() {
        Iterator<String> schedulerIt = plotScheduler.keySet().
            iterator();
        while (schedulerIt.hasNext()) {
                String server = schedulerIt.next();
                PlotScheduler ps = plotScheduler.get(server);
                if (ps.subnetCount() < 1) {
                        LOGGER.log(Level.WARNING, "No subnets
                            feeding from " + ps.name + ". I'll prune
                             it.");
                        schedulerIt.remove();
                }
        }
}


/**
 * Schedule a recurring call to produce plot jobs.
 */
private void schedulePlots() {
        ScheduledExecutorService scheduler = Executors.
```

```
                newScheduledThreadPool(1);
        for (PlotScheduler ps : plotScheduler.values()) {

                // schedule first plot immediately
                new Thread(ps).start();

                // satrt automated plots at the top of the next
                    period
                int delay = SubnetPlotter.DURATION_S;
                delay -= (System.currentTimeMillis() / 1000) %
                    SubnetPlotter.DURATION_S;
                LOGGER.fine("delay: " + delay);
                scheduler.scheduleAtFixedRate(ps, delay,
                    SubnetPlotter.DURATION_S, TimeUnit.SECONDS);
        }
}

public static JSAPResult getArguments(String[] args) {
        JSAPResult config = null;
        try {
                SimpleJSAP jsap = new SimpleJSAP(JSAP_PROGRAM_NAME,
                    JSAP_EXPLANATION_PREFACE +
                    DEFAULT_JSAP_EXPLANATION,
                                DEFAULT_JSAP_PARAMETERS);


                config = jsap.parse(args);
                if (jsap.messagePrinted()) {
                        // The following error message is useful for
                             catching the case
                        // when args are missing, but help isn't
                            printed.
                        if (!config.getBoolean("help"))
                                System.err.println("Try using the --
                                    help flag.");

                        System.exit(1);
                }
        } catch (Exception ex) {
                ex.printStackTrace();
                System.exit(1);
        }
        return config;
}

public static void createConfig() throws IOException {
```

```
        InputStream is = Pensive.class.getResourceAsStream("pensive.
            config");
        FileOutputStream os = new FileOutputStream(
            DEFAULT_CONFIG_FILENAME);
        try {
                byte[] buffer = new byte[1024];
                int length;
                while ((length = is.read(buffer)) > 0) {
                        os.write(buffer, 0, length);
                }
        } finally {
                is.close();
                os.close();
        }
}

/**
 * Where it all begins
 *
 * @param args
 */
public static void main(String[] args) {
        JSAPResult config = getArguments(args);

        // create the package-level logger so inheritance works
        Log.getLogger("gov.usgs");

        Logger logger = LogManager.getLogManager().getLogger("gov.
            usgs");
        if (config.getBoolean("verbose"))
                logger.setLevel(Level.ALL);
        else
                logger.setLevel(Level.FINE);

        Log.attachFileLogger(LOGGER, "pensiveLog", 100000, 10, true)
            ;
        LOGGER.setLevel(Level.INFO);

        if (config.getBoolean("create-config")) {
                try {
                        LOGGER.warning("Creating example config " +
                            DEFAULT_CONFIG_FILENAME);
                        Pensive.createConfig();
                } catch (IOException e) {
                        LOGGER.warning("Cannot write example config.
```

```
                                        " + e.getLocalizedMessage());
                        }
                        System.exit(0);
                }

                String fn = config.getString("configFilename");
                ConfigFile cf = new ConfigFile(fn);
                if (!cf.wasSuccessfullyRead()) {
                        LOGGER.warning("Can't parse config file " + fn + ".
                            Try using the --help flag.");
                        System.exit(1);
                }

                LOGGER.finest("starting Pensive");

                Pensive pensive = new Pensive(cf);
                pensive.schedulePlots();
        }
}
```

PlotScheduler.java

```
package net.stash.pensive;

import gov.usgs.util.ConfigFile;
import gov.usgs.util.Log;
import gov.usgs.util.Pool;
import gov.usgs.util.Util;

import java.util.LinkedList;
import java.util.List;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.logging.Level;
import java.util.logging.Logger;

import net.stash.pensive.plot.SubnetPlotter;

/**
 * Create a pool of connections to a single server and assign plot jobs to
 * connections as they become available.
 *
 * @author Tom Parker
 *
 * I waive copyright and related rights in the this work worldwide through
     the CC0 1.0 Universal public domain dedication.
 * https://creativecommons.org/publicdomain/zero/1.0/legalcode
 */
public class PlotScheduler implements Runnable {

        /** my logger */
        private static final Logger LOGGER = Log.getLogger("gov.usgs");

        public static final String DEFAULT_HOST = "localhost";
        public static final int DEFAULT_PORT = 16022;
        public static final int DEFAULT_NUMTHREADS = 5;

        /** pool of plotters, each with their own wave server connection */
        private final Pool<WaveSource> plotter;

        /** Queue of plot jobs awaiting an available plotter */
        private final BlockingQueue<PlotJob> plotJobs;

        /** list of subnets that feed from my wave server */
        private final List<SubnetPlotter> subnets;
```

```
/** number of connections to my wave server */
private final int numThreads;

/** name of this server */
public final String name;

/**
 * Class constructor
 *
 * @param name
 *            name given to this wave server in the config file
 * @param config
 *            My configuration stanza
 */
public PlotScheduler(String name, ConfigFile config) {

        this.name = name;
        numThreads = Util.stringToInt(config.getString("threads"),
            DEFAULT_NUMTHREADS);
        subnets = new LinkedList<SubnetPlotter>();
        plotJobs = new LinkedBlockingQueue<PlotJob>();

        plotter = new Pool<WaveSource>();
        for (int i = 0; i < numThreads; i++) {
                String n = name + "-" + i;
                WaveSource p = new WaveSource(n, plotJobs, config);
                plotter.checkin(p);
                Thread t = new Thread(p);
                t.setName(n);
                t.start();
        }
}

/**
 *
 * @param subnet
 *            The subnet to be added
 */
public void add(SubnetPlotter subnet) {
        subnets.add(subnet);
}

/**
 *
 * @return count of subnets I have
 */
```

```
        public int subnetCount() {
                return subnets.size();
        }

        /**
         * Schedule the next plot for each subnet.
         */
        public void schedulePlots() {
                for (SubnetPlotter subnet : subnets) {
                        try {
                                LOGGER.log(Level.FINE, "Scheduling subnet "
                                    + subnet.subnetName);
                                plotJobs.put(new PlotJob(subnet));
                        } catch (InterruptedException e) {
                                LOGGER.log(Level.WARNING, "Interrupted.
                                    Unable to schedule " + subnet.subnetName
                                    );
                        }
                }
        }

        /**
         * Schedule the next set of plots. Try to catch all exceptions,
         * ScheduledExecutorService does the wrong thing with exceptions.
         */
        @Override
        public void run() {
                try {
                        LOGGER.log(Level.INFO, "scheduling plots for " +
                            name);
                        schedulePlots();
                } catch (Exception e) {
                        LOGGER.log(Level.SEVERE, "Caught exception heading
                            for scheduler. " + e.getLocalizedMessage());
                }
        }
}
```

WaveSource.java

```
package net.stash.pensive;

import gov.usgs.swarm.data.DataSourceType;
import gov.usgs.swarm.data.SeismicDataSource;
import gov.usgs.util.ConfigFile;
import gov.usgs.util.Log;
import gov.usgs.util.Time;
import gov.usgs.util.Util;

import java.util.concurrent.BlockingQueue;
import java.util.logging.Level;
import java.util.logging.Logger;

import net.stash.pensive.plot.SubnetPlotter;

/**
 * Retrieve data and produce plot
 *
 * @author Tom Parker
 *
 * I waive copyright and related rights in the this work worldwide through
     the CC0 1.0 Universal public domain dedication.
 * https://creativecommons.org/publicdomain/zero/1.0/legalcode
 */
public class WaveSource implements Runnable {

        /** my logger */
        private static final Logger LOGGER = Log.getLogger("gov.usgs");

        public static final String DEFAULT_TYPE = "wws";
        public static final String DEFAULT_HOST = "localhost";
        public static final int DEFAULT_PORT = 16022;
        public static final int DEFAULT_TIMEOUT_S = 15;

        /** source of wave data */
        private final SeismicDataSource dataSource;

        /** Jobs to be plotted */
        private BlockingQueue<PlotJob> plotJobs;

        /** my name */
        public final String name;

        /**
```

```
 * Class constructor
 *
 * @param name
 *            My name
 * @param plotJobs
 *            Queue containing jobs to plot
 * @param config
 *            My config stanza
 */
public WaveSource(String name, BlockingQueue<PlotJob> plotJobs,
    ConfigFile config) {
        this.plotJobs = plotJobs;

        this.name = name;
        String type = Util.stringToString(config.getString("type"),
            DEFAULT_TYPE);
        String host = Util.stringToString(config.getString("host"),
            DEFAULT_HOST);
        int port = Util.stringToInt(config.getString("port"),
            DEFAULT_PORT);
        int timeout = Util.stringToInt(config.getString("timeout"),
            DEFAULT_TIMEOUT_S);
        int compress = 1;

        String dsString = name + ";" + type + ":" + host + ":" +
            port + ":" + timeout * 1000 + ":" + compress;
        dataSource = DataSourceType.parseConfig(dsString);
        dataSource.establish();
        dataSource.setUseCache(false);
}

/**
 * Take plot jobs and produce files.
 */
@Override
public void run() {
        while (true) {
                PlotJob pj = null;
                try {
                        pj = plotJobs.take();
                        if (System.currentTimeMillis() < pj.
                            plotTimeMs) {
                            Thread.sleep(1000);
                            plotJobs.put(pj);
                            continue;
                        }
```

```
                                SubnetPlotter subnet = pj.subnet;

                                LOGGER.log(Level.FINE, "Ploting " + subnet.
                                    subnetName + " from " + name + " at " +
                                    Time.toDateString(System.
                                    currentTimeMillis()));
                                subnet.plot(pj.plotEndMs, dataSource);
                            } catch (InterruptedException noAction) {
                                continue;
                            }
                        }
                    }
}
```

SubnetPlotter.java

```
package net.stash.pensive.plot;

import gov.usgs.plot.Plot;
import gov.usgs.plot.PlotException;
import gov.usgs.swarm.data.SeismicDataSource;
import gov.usgs.util.ConfigFile;
import gov.usgs.util.Log;
import gov.usgs.util.Time;
import gov.usgs.util.Util;

import java.awt.Dimension;
import java.io.File;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.regex.Matcher;

import net.stash.pensive.Channel;

/**
 * A single subnet.
 *
 * @author Tom Parker
 *
 * I waive copyright and related rights in the this work worldwide through
     the CC0 1.0 Universal public domain dedication.
 * https://creativecommons.org/publicdomain/zero/1.0/legalcode
 */
public class SubnetPlotter {

    /** my logger */
    private static final Logger LOGGER = Log.getLogger("gov.usgs");

    public static final String DEFAULT_PATH_ROOT = "html";
    public static final String DEFAULT_FILE_PATH_FORMAT = "yyyy/DDD";
    public static final String DEFAULT_FILE_SUFFIX_FORMAT = "_yyyyMMdd-HHmm
        ";

    public static final int DEFAULT_PLOT_WIDTH = 576;
    public static final int DEFAULT_PLOT_HEIGHT = 756;
    public static final int DEFAULT_THUMB_WIDTH = 151;
    public static final int DEFAULT_THUMB_HEIGHT = 198;
    public static final int DEFAULT_EMBARGO = 5;
```

```
/** width of plot decorations in pixels */
public static final int LABEL_HEIGHT = 35;

/** height of plot decorations in pixels */
public static final int LABEL_WIDTH = 30;

/** The duration of a single plot */
public static final int DURATION_S = 10 * 60;

/** Root of plot directory */
private final String pathRoot;

/** format of file path */
private final String filePathFormat;

/** format of file name */
private final String fileSuffixFormat;

/** Network this subnet belongs to */
public final String networkName;

/** my name */
public final String subnetName;

/** Delay image production by this amount */
public final int embargoMs;

/** plot dimension */
private final Dimension plotDimension;

/** thumbnail dimension */
private final Dimension thumbDimension;

/** Channels on this plot */
private final List<Channel> channels;

/** height of a single channel plot */
private int channelHeight;

/**
 * Class constructor
 *
 * @param networkName
 *            My network name
 *
```

```
     * @param subnetName
     *            My subnet name
     *
     * @param config
     *            My configuration stanza
     */
    public SubnetPlotter(String networkName, String subnetName, ConfigFile
        config) {
        this.subnetName = subnetName;
        this.networkName = networkName;

        pathRoot = Util.stringToString(config.getString("pathRoot"),
            DEFAULT_PATH_ROOT);
        filePathFormat = Util.stringToString(config.getString("
            filePathFormat"), DEFAULT_FILE_PATH_FORMAT);
        fileSuffixFormat = Util.stringToString(config.getString("
            fileSuffixFormat"), DEFAULT_FILE_SUFFIX_FORMAT);
        embargoMs = Util.stringToInt(config.getString("embargo"),
            DEFAULT_EMBARGO) * 1000;

        plotDimension = new Dimension();
        plotDimension.width = Util.stringToInt(config.getString("plotWidth")
            , DEFAULT_PLOT_WIDTH);
        plotDimension.height = Util.stringToInt(config.getString("plotHeight
            "), DEFAULT_PLOT_HEIGHT);

        thumbDimension = new Dimension();
        thumbDimension.width = Util.stringToInt(config.getString("thumbWidth
            "), DEFAULT_THUMB_WIDTH);
        thumbDimension.height = Util.stringToInt(config.getString("
            thumbHeight"), DEFAULT_THUMB_HEIGHT);

        channels = createChannels(config.getSubConfig(subnetName, true));
    }

    /**
     * Create Channel objects for each channel on this plot
     *
     * @param config
     *            My configuration stanza
     *
     * @return List object conatining my channels
     */
    private List<Channel> createChannels(ConfigFile config) {

        List<Channel> channels = new ArrayList<Channel>();
```

```
    List<String> chans = config.getList("channel");

    Dimension plotChanDimension = new Dimension();
    plotChanDimension.height = (plotDimension.height - 2 * FullPlotter.
        LABEL_HEIGHT) / chans.size();
    plotChanDimension.width = plotDimension.width;

    Dimension thumbChanDimension = new Dimension();
    thumbChanDimension.height = thumbDimension.height / chans.size();
    thumbChanDimension.width = thumbDimension.width;

    int idx = 0;
    for (String channel : chans) {
        boolean decorateX = (idx == chans.size() - 1) ? true : false;
        ConfigFile c = config.getSubConfig(channel, true);
        Channel chan = new Channel(channel, idx, plotChanDimension,
            thumbChanDimension, decorateX, c);
        channels.add(chan);
        idx++;
    }

    return channels;
}

/**
 * Produce both a full and a thumbnail PNG representing my subnet
 *
 * @param plotEndMs
 *            time of last sample on plot
 *
 * @param dataSource
 *            source of wave data
 */
public void plot(long plotEndMs, SeismicDataSource dataSource) {
    Plot plot = new Plot(plotDimension.width, plotDimension.height);
    Plot thumb = new Plot(thumbDimension.width, thumbDimension.height);

    for (Channel channel : channels) {
        channel.updateWave(plotEndMs, dataSource);
        plot.addRenderer(channel.plot());
        thumb.addRenderer(channel.plotThumb());
    }

    String fileBase = generateFileBase(plotEndMs);
    new File(fileBase).getParentFile().mkdirs();
```

```
        writePNG(plot, fileBase + ".png");
        writePNG(thumb, fileBase + "_thumb.png");
    }

    /**
     * Write a plot to a PNG file
     *
     * @param plot
     *            The plot to write
     *
     * @param fileName
     *            The file to write
     */
    private void writePNG(Plot plot, String fileName) {
        LOGGER.log(Level.FINE, "writting " + fileName);
        try {
            plot.writePNG(fileName);
        } catch (PlotException e) {
            LOGGER.log(Level.SEVERE, "Cannot write " + fileName + ": " + e.
                getLocalizedMessage());
        }
    }

    /**
     * Generate a file file by applying a SimpleDateFormat
     *
     * @param timeMs
     *            end time of plot
     * @return generated file path
     */
    private String generateFileBase(long timeMs) {
        StringBuilder sb = new StringBuilder();
        sb.append(pathRoot + '/');
        if (networkName != null)
            sb.append(networkName + '/');
        sb.append(subnetName + '/');
        sb.append(Time.format(filePathFormat, timeMs));

        sb.append('/' + subnetName);
        sb.append(Time.format(fileSuffixFormat, timeMs));

        String name = sb.toString();
        name = name.replaceAll("/+", Matcher.quoteReplacement(File.separator
            ));

        return name;
```

```
    }
}
```

PlotJob.java

```
package net.stash.pensive;

import gov.usgs.util.Log;

import java.util.logging.Logger;

import net.stash.pensive.plot.SubnetPlotter;

/**
 * A single plot job
 *
 * @author Tom Parker
 *
 *I waive copyright and related rights in the this work worldwide through
     the CC0 1.0 Universal public domain dedication.
 * https://creativecommons.org/publicdomain/zero/1.0/legalcode
 */
public class PlotJob implements Comparable<PlotJob> {

        /** my logger */
        private static final Logger LOGGER = Log.getLogger("gov.usgs");

        /** Time of last sample plotted */
        public final long plotEndMs;

        /** when to create the plot */
        public final long plotTimeMs;

        /** my subnet */
        public final SubnetPlotter subnet;

        /**
         * Class constructor
         *
         * @param plotEnd
         *            Time of last sample to me plotted
         * @param subnet
         *            My subnet
         */
        public PlotJob(SubnetPlotter subnet, long plotEndMs) {
                this.plotEndMs = plotEndMs;
                this.subnet = subnet;

                plotTimeMs = plotEndMs + subnet.embargoMs;
```

```
        }

        /**
         * Class constructor which uses the most recent time slice as the
             time of
         * the last sample to be plotted.
         *
         * @param subnet
         *            my subnet
         */
        public PlotJob(SubnetPlotter subnet) {
                this.subnet = subnet;
                this.plotEndMs = findPlotEnd();
                plotTimeMs = plotEndMs + subnet.embargoMs;
        }

        /**
         * Calculate the time of the last sample in the most recent time
             slice.
         *
         * @return Time of the last sample in the most recent time slice.
         */
        private long findPlotEnd() {
                long startTime = System.currentTimeMillis();
                startTime -= startTime % (SubnetPlotter.DURATION_S * 1000);

                return startTime;
        }

        /**
         * Order plots by increasing last sample time
         *
         * @param o
         *      The PlotJob to compare to
         */
        @Override
        public int compareTo(PlotJob o) {
                return (int) (plotTimeMs - o.plotTimeMs);
        }
}
```

Channel.java

```
package net.stash.pensive;

import gov.usgs.plot.data.SliceWave;
import gov.usgs.plot.data.Wave;
import gov.usgs.plot.render.Renderer;
import gov.usgs.swarm.data.SeismicDataSource;
import gov.usgs.util.ConfigFile;
import gov.usgs.util.Log;
import gov.usgs.util.Util;

import java.awt.Dimension;
import java.util.logging.Logger;

import net.stash.pensive.plot.ChannelPlotter;
import net.stash.pensive.plot.FullPlotter;
import net.stash.pensive.plot.SubnetPlotter;
import net.stash.pensive.plot.ThumbnailPlotter;

/**
 * A single channel of seismic data on a single subnet plot.
 *
 * @author Tom Parker
 *
 * I waive copyright and related rights in the this work worldwide through
     the CC0 1.0 Universal public domain dedication.
 * https://creativecommons.org/publicdomain/zero/1.0/legalcode
 */
public class Channel {

    /** my logger */
    private static final Logger LOGGER = Log.getLogger("gov.usgs");

    /** channel name in config file format */
    public final String name;

    /** My full ChannelPlotter */
    private final ChannelPlotter plot;

    /** My thumbnail ChannelPlotter */
    private final ChannelPlotter thumb;

    /**
     * Class constructor
     *
```

```
     * @param channel
     *            my channel
     * @param index
     *            my index into the plot
     * @param plotDimension
     *            Dimension of the full plot
     * @param thumbDimension
     *            Dimension of the thumbnail plot
     * @param decorateX
     *            If true decorate x-axis on full plot
     * @param config
     *            My config stanza
     */
    public Channel(String channel, int index, Dimension plotDimension,
        Dimension thumbDimension, boolean decorateX,
            ConfigFile config) {
        this.name = channel;

        plot = new FullPlotter(channel, index, plotDimension, decorateX,
            config);
        thumb = new ThumbnailPlotter(channel, index, thumbDimension, config)
            ;

    }

    /**
     * Gather new wave data and offer to plotters
     *
     * @param plotEndMs
     *            Time of last sample of waveform
     * @param dataSource
     *            Who to ask for data
     */
    public void updateWave(long plotEndMs, SeismicDataSource dataSource) {
        double t2 = Util.ewToJ2K(plotEndMs / 1000);
        double t1 = t2 - SubnetPlotter.DURATION_S;
        Wave w = dataSource.getWave(name.replace('_', ' '), t1, t2);
        SliceWave wave = null;
        if (w != null) {
            wave = new SliceWave(w);
            wave.setSlice(t1, t2);
        }
        plot.setWave(wave);
        thumb.setWave(wave);
    }
```

```
    /**
     * Create a full plot
     *
     * @return The plot Renderer
     */
    public Renderer plot() {
        return plot.plot();
    }

    /**
     * Create a thumbnail plot
     *
     * @return The thumbnail Renderer
     */
    public Renderer plotThumb() {
        return thumb.plot();
    }

}
```

ChannelPlotter.java

```
package net.stash.pensive.plot;

import gov.usgs.plot.data.SliceWave;
import gov.usgs.plot.render.BasicFrameRenderer;
import gov.usgs.plot.render.TextRenderer;
import gov.usgs.plot.render.wave.MinuteMarkingWaveRenderer;
import gov.usgs.plot.render.wave.SliceWaveRenderer;
import gov.usgs.plot.render.wave.SpectrogramRenderer;
import gov.usgs.util.ConfigFile;
import gov.usgs.util.Util;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;

/**
 *
 * @author Tom Parker
 *
 * I waive copyright and related rights in the this work worldwide through
     the CC0 1.0 Universal public domain dedication.
 * https://creativecommons.org/publicdomain/zero/1.0/legalcode
 */
public abstract class ChannelPlotter {

    /** Font used to indicate no data available */
    public static final Color NO_DATA_TEXT_COLOR = new Color(160, 41, 41);

    /** The ratio of a waveform plot to its spectrogram plot */
    public static final double WAVE_RATIO = .25;

    public static final double DEFAULT_OVERLAP = 0.859375;
    public static final boolean DEFAULT_LOG_POWER = true;
    public static final double DEFAULT_MIN_FREQ = 0;
    public static final double DEFAULT_MAX_FREQ = 10;
    public static final int DEFAULT_NFFT = 0;
    public static final int DEFAULT_BIN_SIZE = 256;
    public static final int DEFAULT_MAX_POWER = 120;
    public static final int DEFAULT_MIN_POWER = 30;

    /** my wave data */
    protected SliceWave wave;

    /** Font to use for no data message */
```

```
protected Font noDataFont;

/** Channel position in plot */
protected int index;

/** Dimension of channel plot */
protected Dimension plotDimension;

/** my config stanza */
protected ConfigFile config;

/** my wave renderer */
private final SliceWaveRenderer waveRenderer;

/** my spectrogram renderer */
protected final SpectrogramRenderer spectrogramRenderer;

/** my frame renderer */
private final BasicFrameRenderer plotFrame;

/** height of the wave panel */
protected final int waveHeight;

/** my name */
protected final String name;

/** make any type-specific modifications to the SpectrogramRenderer */
protected abstract void tweakSpectrogramRenderer(SpectrogramRenderer
    spectrogramRenderer);

/** make any type-specific modifications to the SliceWaveRenderer */
protected abstract void tweakWaveRenderer(SliceWaveRenderer waveRenderer
    );

/** make any type-specific modifications to the no-data Renderer */
protected abstract void tweakNoDataRenderer(TextRenderer textRenderer);

/**
 * Class constructor
 *
 * @param name
 *            My name
 *
 * @param index
 *            My position on the subnet plot
 *
```

```
     * @param plotDimension
     *            The dimension of the plot
     *
     * @param config
     *            My configuration stanza
     */
    public ChannelPlotter(String name, int index, Dimension plotDimension,
        ConfigFile config) {
        this.name = name;
        this.index = index;
        this.plotDimension = plotDimension;
        this.config = config;
        waveHeight = (int) (plotDimension.height * WAVE_RATIO);

        waveRenderer = createWaveRenderer();
        spectrogramRenderer = createSpectrogramRenderer(config);

        plotFrame = new BasicFrameRenderer();
        plotFrame.addRenderer(waveRenderer);
        plotFrame.addRenderer(spectrogramRenderer);
    }

    /**
     * Create a SpectrogramRendere and apply my settings
     *
     * @param config
     *            my config stanza
     *
     * @return my SpectrogramRenderer
     *
     */
    protected SpectrogramRenderer createSpectrogramRenderer(ConfigFile
        config) {

        SpectrogramRenderer sr = new SpectrogramRenderer();

        // y-axis labels will sometimes not be displayed if x-axis tick
            marks
        // are not displayed. Note sure why.
        sr.yTickMarks = false;
        sr.yTickValues = false;
        sr.xTickMarks = false;
        sr.xTickValues = false;
        sr.xUnits = false;
        sr.xLabel = false;
```

```
        sr.setOverlap(Util.stringToDouble(config.getString("overlap"),
            DEFAULT_OVERLAP));
        sr.setLogPower(Util.stringToBoolean(config.getString("logPower"),
            DEFAULT_LOG_POWER));
        sr.setMinFreq(Util.stringToDouble(config.getString("minFreq"),
            DEFAULT_MIN_FREQ));
        sr.setMaxFreq(Util.stringToDouble(config.getString("maxFreq"),
            DEFAULT_MAX_FREQ));
        sr.setNfft(Util.stringToInt(config.getString("nfft"), DEFAULT_NFFT))
            ;
        sr.setBinSize(Util.stringToInt(config.getString("binSize"),
            DEFAULT_BIN_SIZE));
        sr.setMinPower(Util.stringToInt(config.getString("minPower"),
            DEFAULT_MIN_POWER));
        sr.setMaxPower(Util.stringToInt(config.getString("maxPower"),
            DEFAULT_MAX_POWER));
        sr.setTimeZone("UTC");

        tweakSpectrogramRenderer(sr);

        return sr;
    }

    /**
     * Create a WaveRenderer and apply my settings
     *
     * @return my SliceWaveRenderer
     */
    protected SliceWaveRenderer createWaveRenderer() {

        SliceWaveRenderer wr = new MinuteMarkingWaveRenderer();

        wr.xTickMarks = false;
        wr.xTickValues = false;
        wr.xUnits = false;
        wr.xLabel = false;
        wr.yTickMarks = false;
        wr.yTickValues = false;
        wr.setColor(Color.BLACK);
        tweakWaveRenderer(wr);

        return wr;
    }

    /**
     * wave mutator method
```

```
 *
 * @param wave
 */
public void setWave(SliceWave wave) {
    this.wave = wave;
}

/**
 * Produce the plot
 *
 * @return frame renderer containing plot or error message
 */
public BasicFrameRenderer plot() {

    if (wave == null) {
        return noDataRenderer();

    } else {
        double plotStart = wave.getStartTime();
        double plotEnd = wave.getStartTime() + SubnetPlotter.DURATION_S;

        waveRenderer.setMinY(wave.min());
        waveRenderer.setMaxY(wave.max());
        waveRenderer.setWave(wave);
        waveRenderer.setViewTimes(plotStart, plotEnd, "UTC");
        waveRenderer.update();

        spectrogramRenderer.setWave(wave);
        spectrogramRenderer.setViewStartTime(plotStart);
        spectrogramRenderer.setViewEndTime(plotEnd);
        spectrogramRenderer.createDefaultFrameDecorator();
        spectrogramRenderer.update();
        return plotFrame;
    }
}

/**
 * Produce a graphical error message indicating that no data is
     available
 *
 * @return A graphical error message
 */
private BasicFrameRenderer noDataRenderer() {
    BasicFrameRenderer fr = new BasicFrameRenderer();
    int top = index * plotDimension.height;
    TextRenderer tr = new TextRenderer(plotDimension.width / 2, top +
```

```
            plotDimension.height / 2, name + " - no data");

        tr.horizJustification = TextRenderer.CENTER;
        tr.vertJustification = TextRenderer.CENTER;
        tr.color = NO_DATA_TEXT_COLOR;
        tr.font = noDataFont;
        tweakNoDataRenderer(tr);

        fr.addRenderer(tr);
        return fr;
    }
}
```

ThumbnailPlotter.java

```
package net.stash.pensive.plot;

import gov.usgs.plot.render.TextRenderer;
import gov.usgs.plot.render.wave.SliceWaveRenderer;
import gov.usgs.plot.render.wave.SpectrogramRenderer;
import gov.usgs.util.ConfigFile;

import java.awt.Dimension;
import java.awt.Font;

/**
 *
 * @author Tom Parker
 *
 * I waive copyright and related rights in the this work worldwide through
     the CC0 1.0 Universal public domain dedication.
 * https://creativecommons.org/publicdomain/zero/1.0/legalcode
 */
public class ThumbnailPlotter extends ChannelPlotter {

    /** Font used to indicate no data available */
    public static final Font NO_DATA_FONT = Font.decode("dialog-PLAIN-12");

    /**
     * Class constructor
     *
     * @param name
     *            My name
     * @param index
     *            My position on the subnet plot
     * @param plotDimension
     *            My configuration stanza
     * @param config
     */
    public ThumbnailPlotter(String name, int index, Dimension plotDimension,
         ConfigFile config) {
        super(name, index, plotDimension, config);

        noDataFont = NO_DATA_FONT;
    }

    /**
     * Apply any settings needed to the Spectrogram renderer
     *
```

```
     * @param spectrogramRenderer
     *            my SpectrogramRenderer
     */
    protected void tweakSpectrogramRenderer(SpectrogramRenderer
        spectrogramRenderer) {

        // y-axis labels will sometimes not be displayed if x-axis tick
            marks
        // are not displayed. Note sure why.
        spectrogramRenderer.yTickMarks = false;
        spectrogramRenderer.yTickValues = false;
        spectrogramRenderer.xTickMarks = false;
        spectrogramRenderer.xTickValues = false;
        spectrogramRenderer.xUnits = false;
        spectrogramRenderer.xLabel = false;

        int top = index * plotDimension.height + waveHeight;
        spectrogramRenderer.setLocation(0, top, plotDimension.width,
            plotDimension.height - waveHeight);
    }

    /**
     * Apply any settings needed to the WaveRenderer
     *
     * @param waveRenderer
     *            my WaveRenderer
     */
    @Override
    protected void tweakWaveRenderer(SliceWaveRenderer waveRenderer) {
        int top = index * plotDimension.height;
        int width = plotDimension.width;
        waveRenderer.setLocation(0, top, width, waveHeight);

    }

    /**
     * Apply any tweaks needed to the textRenderer
     *
     * @param textRenderer
     *            My text renderer
     */
    protected void tweakNoDataRenderer(TextRenderer textRenderer) {
        // no tweaks needed
    }

}
```

FullPlotter.java

```
package net.stash.pensive.plot;

import gov.usgs.plot.render.TextRenderer;
import gov.usgs.plot.render.wave.SliceWaveRenderer;
import gov.usgs.plot.render.wave.SpectrogramRenderer;
import gov.usgs.util.ConfigFile;

import java.awt.Dimension;
import java.awt.Font;

/**
 *
 * @author Tom Parker
 *
 * I waive copyright and related rights in the this work worldwide through
     the CC0 1.0 Universal public domain dedication.
 * https://creativecommons.org/publicdomain/zero/1.0/legalcode
 */
public class FullPlotter extends ChannelPlotter {

    /** Font used to indicate that no data was available */
    public static final Font NO_DATA_FONT = Font.decode("dialog-PLAIN-36");

    /** height of plot decorations in pixels */
    public static final int LABEL_HEIGHT = 35;

    /** width of plot decorations in pixels */
    public static final int LABEL_WIDTH = 30;

    /**
     * Class constructor
     *
     * @param name
     *            My name
     *
     * @param index
     *            My position on the subnet plot
     *
     * @param plotDimension
     *            The dimension of my plot
     *
     * @param decorateX
     *            If true, decorate the X-axis
     *
```

```
     * @param config
     *            My configuration stanza
     */
    public FullPlotter(String name, int index, Dimension plotDimension,
        boolean decorateX, ConfigFile config) {
        super(name, index, plotDimension, config);

        spectrogramRenderer.xTickValues = decorateX;
        spectrogramRenderer.xUnits = decorateX;
        spectrogramRenderer.xLabel = decorateX;

        noDataFont = NO_DATA_FONT;
    }

    /**
     * Apply any settings needed to my SpectrogramRenderer
     *
     * @param my
     *            SpectrogramRenderer
     */
    protected void tweakSpectrogramRenderer(SpectrogramRenderer
        spectrogramRenderer) {
        spectrogramRenderer.yTickMarks = true;
        spectrogramRenderer.yTickValues = true;
        spectrogramRenderer.xTickMarks = true;
        spectrogramRenderer.setYLabelText(name);

        int top = index * plotDimension.height;
        spectrogramRenderer.setLocation(LABEL_WIDTH, top + waveHeight +
            LABEL_HEIGHT, plotDimension.width
                - (2 * LABEL_WIDTH), plotDimension.height - waveHeight);
    }

    /**
     * Apply any settings needed to my WaveRenderer
     *
     * @param my
     *            WaveRenderer
     */
    protected void tweakWaveRenderer(SliceWaveRenderer waveRenderer) {
        int top = index * plotDimension.height + LABEL_HEIGHT;
        int width = plotDimension.width;
        waveRenderer.setLocation(LABEL_WIDTH, top, width - 2 * LABEL_WIDTH,
            waveHeight);

        waveRenderer.xTickMarks = true;
```

```
    }

    /**
     * Apply any settings needed to my TextRenderer
     *
     * @param my
     *             TextRenderer
     */
    protected void tweakNoDataRenderer(TextRenderer textRenderer) {
        textRenderer.y += LABEL_HEIGHT;
    }

}
```

Page.java

```
package net.stash.pensive.page;

import freemarker.cache.ClassTemplateLoader;
import freemarker.template.Configuration;
import freemarker.template.DefaultObjectWrapper;
import freemarker.template.Template;
import freemarker.template.TemplateException;
import freemarker.template.TemplateExceptionHandler;
import freemarker.template.Version;
import gov.usgs.util.ConfigFile;
import gov.usgs.util.Log;
import gov.usgs.util.Util;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.regex.Matcher;

import net.stash.pensive.plot.SubnetPlotter;

/**
 *
 * @author Tom Parker
 *
 * I waive copyright and related rights in the this work worldwide through
     the CC0 1.0 Universal public domain dedication.
 * https://creativecommons.org/publicdomain/zero/1.0/legalcode
 */
public class Page {
    /** my logger */
    private static final Logger LOGGER = Log.getLogger("gov.usgs");

    public static final String DEFAULT_PATH_ROOT = "html/";

    /** filename of html file */
    public static final String FILENAME = "index.html";

    /** Freemarker settings */
```

```
private Map<String, Object> root;

/** my configuration */
private Configuration cfg;

/** My subnets */
private Map<String, List<String>> subnets;

/** root of output*/
private final String pathRoot;

/**
 * Class constructor
 *
 * @param config
 *          My configuration stanza
 */
public Page(ConfigFile config) {
    pathRoot = Util.stringToString(config.getString("pathRoot"),
        DEFAULT_PATH_ROOT);

    root = new HashMap<String, Object>();

    subnets = new HashMap<String, List<String>>();
    root.put("subnets", subnets);

    root.put("refreshPeriod", SubnetPlotter.DURATION_S);
    root.put("filePathFormat", Util.stringToString(config.getString("
        filePathFormat"), SubnetPlotter.DEFAULT_FILE_PATH_FORMAT));
    root.put("fileSuffixFormat", Util.stringToString(config.getString("
        fileNameSuffixFormat"), SubnetPlotter.DEFAULT_FILE_SUFFIX_FORMAT
        ));
    root.put("selectedNetwork", config.getString("selectedNetwork"));

    try {
        initializeTemplateEngine();
    } catch (IOException e) {
        LOGGER.log(Level.SEVERE, "cannot write HTML");
    }

}

/**
 * Initialize FreeMarker
 * @throws IOException
 */
```

```
protected void initializeTemplateEngine() throws IOException {
    cfg = new Configuration();
    cfg.setTemplateLoader(new ClassTemplateLoader(getClass(), "/net/
        stash/pensive/page"));
    DefaultObjectWrapper obj = new DefaultObjectWrapper();
    obj.setExposeFields(true);
    cfg.setObjectWrapper(obj);

    cfg.setDefaultEncoding("UTF-8");
    cfg.setTemplateExceptionHandler(TemplateExceptionHandler.
        HTML_DEBUG_HANDLER);
    cfg.setIncompatibleImprovements(new Version(2, 3, 20));
}

/**
 * Write my html page
 */
public void writeHTML() {
    try {
        File pRoot = new File(pathRoot);
        if (!pRoot.exists())
            pRoot.mkdirs();

        Template template = cfg.getTemplate("pensive.html");
        String file = pathRoot + '/' + FILENAME;
        file.replace("/+", "/");
        file.replace("/", Matcher.quoteReplacement(File.separator));
        FileWriter fw = new FileWriter(file);
        template.process(root, fw);
        fw.close();
    } catch (IOException e) {
        LOGGER.log(Level.SEVERE, e.getLocalizedMessage());
    } catch (TemplateException e) {
        LOGGER.log(Level.SEVERE, e.getLocalizedMessage());
    }
}

/** add a subnet to a network list */
public void addSubnet(String network, String subnet) {
    List<String> s = subnets.get(network);
    if (s == null) {
            s = new ArrayList<String>();
            subnets.put(network, s);
    }
    s.add(subnet);
}
```

```
}
```

dialogs.html

```
<!--
    I waive copyright and related rights in the this work worldwide through
        the CC0 1.0 Universal public domain dedication.
    https://creativecommons.org/publicdomain/zero/1.0/legalcode
-->
<!--
                MOSAIC OPTIONS
-->
<div id="mosaicOptions" class="dialog">
        <h1>Mosaic Options</h1>
        <form id="mosaicOptionsForm" name="mosaicOptionsForm">
        <label for="dailyMosaic">Daily mosaic</label>
        <input name="dailyMosaic" id="dailyMosaic" type="checkbox" />
    <label for="mosaicSpan">Hours per mosaic</label>
        <input name="mosaicSpan" type="text" id="mosaicSpanH" value="3" size
            ="2" class="txtfield positiveInt" />

        <label for="mosaicRowSpan">Minutes per row</label>
        <input name="mosaicRowSpan" type="text" id="mosaicRowSpanM" value
            ="60" size="2" class="txtfield positiveInt" />
        <div class="center"><input type="submit" name="mosaicOptionsBtn" id
            ="mosaicOptionsBtn" class="flatbtn-blu hidemodal" value="Update"
             onClick="updateMosaicOptions();"></div>
        </form>
</div>

<!--
                ABSOLUTE TIME
-->
<div id="absoluteTime" class="dialog">
        <h1>Absolute Time</h1>
        <form id="absoluteTimeForm" name="absoluteTimeForm">
                <label for="ATDate">Start (UTC)</label>
                <input name="ATDate" type="text" id="ATDate" size="4" class
                    ="txtfield" />
                <div class="center"><input type="submit" name="
                    absoluteTimeBtn" id="absoluteTimeBtn" class="flatbtn-blu
                     hidemodal" value="Update" onClick="updateAbsoluteTime()
                    ;"></div>
        </form>
</div>

<!--
                PERMALINK
```

```
-->
<div id="permalink" class="dialog" style="width: auto;">
        <h1>Permalink</h1>
        <div class="permalink">
                This page can always be found at:<br>
                <span id="permalinkURL"></span><br>
        </div>
</div>
```

footer.html

```
<!--
    I waive copyright and related rights in the this work worldwide through
        the CC0 1.0 Universal public domain dedication.
    https://creativecommons.org/publicdomain/zero/1.0/legalcode
-->
  <script>
    <#include "jQuery.js">
    <#include "simpledateformat.js">
    <#include "jquery.leanModal.min.js">
        <#include "util.js">
    <#include "pensive.js">
   $(document).ready(init);

  </script>
</body>
</html>
```

header.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<!--
    I waive copyright and related rights in the this work worldwide through
        the CC0 1.0 Universal public domain dedication.
    https://creativecommons.org/publicdomain/zero/1.0/legalcode
-->

<head>  <meta http-equiv="content-type" content="text/html;charset=UTF-8"
    />

  <style>
  <#include "pensive.css">
   </style>

   <title></title>
</head>

<body>
  <!-- Begin top navbar -->
  <div class="nav">
    <ul class="nav">

          <!-- networks -->
          <select id="network" class="nav">
           <#list subnets?keys as network>
             <option value="${network}" class="nav">${network}</option>
           </#list>
                 </select>

             <!-- subnets -->
             <#list subnets?keys as network>
               <select id="${network}Subnets" class="nav subnet">
               <#assign subs = subnets[network]>
            <#list subs as subnet>
              <option value="${subnet}" class="nav">${subnet}</option>
            </#list>
               </select>
        </#list>

          <a class="nav" title="Load a specific time" name="absoluteTime"
              href="#absoluteTime" rel="leanModal">Absolute time</a>
```

```
          <a class="nav" title="mosaicOptions" name="mosaicOptions" id="
              mosaicOptionsButton" href="#mosaicOptions" rel="leanModal">
              Mosaic Options</a>
          <a class="nav" title="mosaic" name="mosaic" id="mosaicButton">
              Mosaic</a>
          <a class="nav" title="permalink" name="permalink" id="
              permalinkButton" href="#permalink" rel="leanModal">Permalink</
              a>


          <a href="" shape="rect" ><li class="nav" title="next mosaic">What
              Am I Looking At?</li></a>

          <!-- <li class="nav" href="#" onClick="toggle_visibility('colorbar
              ');" shape="rect">Colorbar</li> -->
    </ul>
</div>
<!-- End top navbar -->

<div class="clear"></div>

<form method="get" id="absolutetime" class="hidden">
<span id="absolutetime">
  End time : <input type="text" name="month" value="MM" size="2" /> / <input
       type="text" name="day" value="DD" size="2" /> / <input type="text"
      name="year" value="YYYY" size="4" />
  <input type="text" name="hour" value="HH" size="2" />:<input type="text"
      name="minute" value="MM" size="2" />

  <input type="button" name="submit" value="Go" onClick="loadAbsoluteTime(
      this.form)" />
</span>
</form>



<div class="clear"></div>

<!--
                TOP NAVIGATION
-->
<center>
<div id="title">
  <div id="perviousSubnet" class="navLeft">
    <a title="previousSubnetName"><span class="button" title="previous
        subnet">&#9668;</span></a>
```

```
  </div>
  <div id="nextSubnet" class="navRight">
    <a title="nextSubnetName"><span class="button" title="next subnet" id="
        navRight" style="width: 1em; float: left;">&#9658;</span></a>
  </div>
  <div id="subnetName" style="width: 100%;"></div>

  <div class="clear" style="height: .25em;"></div>

  <div class="navLeft">
    <a title="previous image" id="previousImage"><span class="button" id="
        navLeft" title="previous image">&#9668;</span></a>
  </div>
  <div class="navRight">
    <a title="next image" id="nextImage"><span class="button" style="width:
        1em; float: left;" title="next image">&#9658;</span></a>
    <a title="current image" id="currentImage"><span class="button" style="
        width: 2em; float: right;" title="current image">&#9658;&#9658;</span
        ></a>
  </div>
  <div id="timeSpan" style="width: 100%;"></div>
</div>

</center>
```

pensive.css

```
/*
    I waive copyright and related rights in the this work worldwide through
        the CC0 1.0 Universal public domain dedication.
    https://creativecommons.org/publicdomain/zero/1.0/legalcode
*/
body {
        font-size: 10pt;
        background-color:#202020;
        color: #cccccc;
        font-family: Georgia, Times, "Times New Roman", serif;
}

a:link,a:active,a:visited{
        text-decoration:none;
        color:#cccccc;
        outline: none;
}

#title {
        font-size: 200%;
        font-weight: bold;
        text-align: center;
        margin: 1em 0em 1em 0em;
        width: 700px;
        white-space: nowrap;
}

div.navLeft {
        float: left;
        text-align: right;
        width: 15%;
}

div.navRight {
        float: right;
        width: 15%;
}

span.navLeft {
        width: 1em;
        margin-right: 0px;
}

span.navRight {
```

```
        width: 1em;
        float: left;
}

.dialog {
  width: 300px;
  padding-top: 15px;
  padding-right: 20px;
  padding-bottom: 15px;
  padding-left: 20px;
  background-color: #f3f6fa;
  background-image: none;
  background-repeat: repeat;
  background-attachment: scroll;
  background-position: 0% 0%;
  background-clip: border-box;
  background-origin: padding-box;
  background-size: auto auto;
  border-top-left-radius: 6px;
  border-top-right-radius: 6px;
  border-bottom-right-radius: 6px;
  border-bottom-left-radius: 6px;
  box-shadow: 0px 1px 5px rgba(0, 0, 0, 0.5);
  display: none;
}

.button {
   border-top: 1px solid #567300;
   background: #5e6647;
   background: -webkit-gradient(linear, left top, left bottom, from(#474d36)
       , to(#5e6647));
   background: -webkit-linear-gradient(top, #474d36, #5e6647);
   background: -moz-linear-gradient(top, #474d36, #5e6647);
   background: -ms-linear-gradient(top, #474d36, #5e6647);
   background: -o-linear-gradient(top, #474d36, #5e6647);
   padding: 5px 10px;
   -webkit-border-radius: 8px;
   -moz-border-radius: 8px;
   border-radius: 8px;
   -webkit-box-shadow: rgba(0,0,0,1) 0 1px 0;
   -moz-box-shadow: rgba(0,0,0,1) 0 1px 0;
   box-shadow: rgba(0,0,0,1) 0 1px 0;
   text-shadow: rgba(0,0,0,.4) 0 1px 0;
   color: #cccccc;
   font-size: 14px;
   font-family: Georgia, serif;
```

```
    text-decoration: none;
    vertical-align: middle;
    }

.button:hover {
    border-top-color: #28597a;
    background: #28597a;
    color: #ccc;
    }
.button:active {
    border-top-color: #1b435e;
    background: #1b435e;
    }

/* Used for the top menu bar */
div.nav{
        float:left;
        width:100%;
        font-weight: bold;
}

div#mosaicOptions {
  width: 300px;
  padding-top: 15px;
  padding-right: 20px;
  padding-bottom: 15px;
  padding-left: 20px;
  background-color: #f3f6fa;
  background-image: none;
  background-repeat: repeat;
  background-attachment: scroll;
  background-position: 0% 0%;
  background-clip: border-box;
  background-origin: padding-box;
  background-size: auto auto;
  border-top-left-radius: 6px;
  border-top-right-radius: 6px;
  border-bottom-right-radius: 6px;
  border-bottom-left-radius: 6px;
  box-shadow: 0px 1px 5px rgba(0, 0, 0, 0.5);
}

#mosaicOptions label
{
  display: block;
  font-size: 1.1em;
```

```
  font-weight: bold;
  color: #7c8291;
  margin-bottom: 3px;
}

.permalink
{
  display: block;
  font-size: 1.1em;
  font-weight: bold;
  color: #7c8291;
  margin-bottom: 3px;
}

a:link.permalink,a:active.permalink,a:visited.permalink{
        text-decoration:none;
        color:#7c8291;
        outline: none;
  margin-top: 1em;
}

div#absoluteTime {
  width: 300px;
  padding-top: 15px;
  padding-right: 20px;
  padding-bottom: 15px;
  padding-left: 20px;
  background-color: #f3f6fa;
  background-image: none;
  background-repeat: repeat;
  background-attachment: scroll;
  background-position: 0% 0%;
  background-clip: border-box;
  background-origin: padding-box;
  background-size: auto auto;
  border-top-left-radius: 6px;
  border-top-right-radius: 6px;
  border-bottom-right-radius: 6px;
  border-bottom-left-radius: 6px;
  box-shadow: 0px 1px 5px rgba(0, 0, 0, 0.5);
}

#absoluteTime label
{
  display: block;
  font-size: 1.1em;
```

```
  font-weight: bold;
  color: #7c8291;
  margin-bottom: 3px;
}

table.mosaic
{
        border-collapse: collapse;
}
td.mosaic
{
        margin:0px;
        border:0px;
        padding:.5em 0px .5em 0px;
        overflow: auto;
        empty-cells: show;
}

td.mosaicTitle
{
        font-size: 200%;
        font-weight: bold;
        text-align: center;
        margin: 1em 0em 1em 0em;
        white-space: nowrap;
    width: 151px;
}

td.mosaic:hover
{
/*      background-color: #50384a; */
        background-color: #28597a;
}

img.mosaic
{
        vertical-align: middle;
        background-color: #202020;
        padding:.5em 0px .5em 0px;
}

div.image
{
        text-align: center;
}
.clear { clear: both; }
```

```
.hidden { display:none; }
img {
        border: none;
        border-image-width: 0 0 0 0;
}


li.nav:hover{
        color:#cccccc;
        background-color: #666666;
        font-size:100%;
}

a.nav:hover{
        color:#cccccc;
        background-color: #666666;
        font-size:100%;
}

ul.nav
{
        text-align: center;
        padding: 0px;
        margin: 0px;
        list-style: none;
}

li.nav
{
        text-decoration:none;
        display: inline;
        border:1px solid #000000;
        background-color:#444444;
        line-height:1.75em;
        padding:0.25em 0.5em 0.25em 0.5em;
}
a.nav
{
        text-decoration:none;
        display: inline;
        border:1px solid #000000;
        background-color:#444444;
        line-height:1.75em;
        padding:0.25em 0.5em 0.25em 0.5em;
}
select.nav
```

```
{
                text-decoration:none;
        display: inline;
        border:1px solid #000000;
        background-color:#444444;
        line-height:1.75em;
        padding:0.25em 0.5em 0.25em 0.5em;
                color: #cccccc;
        font-family: Georgia, Times, "Times New Roman", serif;
                font-size:100%;
}

div.center{
        align:center;
          display: block;
  text-align: center;

}

form#absolutetime
{
        text-align:center;
        margin: .5em;
}

span#absolutetime
{
        text-align:center;
        background-color:#666666;
        padding: .5em;
}

.tdimg
{
        /*overflow: hidden;*/
        margin:0px;
        border:0px;
        padding:0px;
        overflow: auto;
        height: 150px;
        width: 96px;
}

.center {
        margin-left: auto;
        margin-right: auto;
```

```
        align: center;
}

/*
ul.mosaic{
  list-style: none;
  width:604px;
  background:#f00;
  padding:0;
  overflow:hidden
}
li.mosaic{
 width:151px;
 height:198px;
 margin:0 0px 10px 0;
 background:#FFF;
 padding:0;
 float:left;
}


li.mosaic:nth-child(6n+7) {
  margin-right:0px;
  clear:left;
}

*/

#lean_overlay {
    position: fixed;
    z-index:100;
    top: 0px;
    left: 0px;
    height:100%;
    width:100%;
    background: #000;
    display: none;
}
/** page structure **/
#w {
  display: block;
  width: 750px;
  margin: 0 auto;
  padding-top: 30px;
}
```

```
#content {
  display: block;
  width: 100%;
  background: #fff;
  padding: 25px 20px;
  padding-bottom: 35px;
  -webkit-box-shadow: rgba(0, 0, 0, 0.1) 0px 1px 2px 0px;
  -moz-box-shadow: rgba(0, 0, 0, 0.1) 0px 1px 2px 0px;
  box-shadow: rgba(0, 0, 0, 0.1) 0px 1px 2px 0px;
}

/** custom login button **/
.flatbtn-blu {
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
  display: inline-block;
  outline: 0;
  border: 0;
  color: #edf4f9;
  text-decoration: none;
  background-color: #4f94cf;
  border-color: rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.25);
  font-size: 1.3em;
  font-weight: bold;
  padding: 12px 26px 12px 26px;
  line-height: normal;
  text-align: center;
  vertical-align: middle;
  cursor: pointer;
  text-transform: uppercase;
  text-shadow: 0 1px 0 rgba(0,0,0,0.3);
  -webkit-border-radius: 3px;
  -moz-border-radius: 3px;
  border-radius: 3px;
  -webkit-box-shadow: 0 1px 1px rgba(0, 0, 0, 0.2);
  -moz-box-shadow: 0 1px 1px rgba(0, 0, 0, 0.2);
  box-shadow: 0 1px 1px rgba(0, 0, 0, 0.2);
}
.flatbtn-blu:hover {
  color: #fff;
  background-color: #519dde;
}
.flatbtn-blu:active {
  -webkit-box-shadow: inset 0 1px 5px rgba(0, 0, 0, 0.1);
  -moz-box-shadow:inset 0 1px 5px rgba(0, 0, 0, 0.1);
```

```
  box-shadow:inset 0 1px 5px rgba(0, 0, 0, 0.1);
}
/** modal window styles **/
#lean_overlay {
    position: fixed;
    z-index:90;
    top: 0px;
    left: 0px;
    height:100%;
    width:100%;
    background: #000;
    display: none;
}

#loginmodal {
  width: 300px;
  padding: 15px 20px;
  background: #f3f6fa;
  -webkit-border-radius: 6px;
  -moz-border-radius: 6px;
  border-radius: 6px;
  -webkit-box-shadow: 0 1px 5px rgba(0, 0, 0, 0.5);
  -moz-box-shadow: 0 1px 5px rgba(0, 0, 0, 0.5);
  box-shadow: 0 1px 5px rgba(0, 0, 0, 0.5);
}

#loginform { /* no default styles */ }

#loginform label { display: block; font-size: 1.1em; font-weight: bold;
    color: #7c8291; margin-bottom: 3px; }

.txtfield {
  display: block;
  width: 100%;
  padding: 6px 5px;
  margin-bottom: 15px;
  font-family: 'Helvetica Neue', Helvetica, Verdana, sans-serif;
  color: #7988a3;
  font-size: 1.4em;
  text-shadow: 1px 1px 0 rgba(255, 255, 255, 0.8);
  background-color: #fff;
  background-image: -webkit-gradient(linear, left top, left bottom, from(#
      edf3f9), to(#fff));
  background-image: -webkit-linear-gradient(top, #edf3f9, #fff);
  background-image: -moz-linear-gradient(top, #edf3f9, #fff);
  background-image: -ms-linear-gradient(top, #edf3f9, #fff);
```

```
  background-image: -o-linear-gradient(top, #edf3f9, #fff);
  background-image: linear-gradient(top, #edf3f9, #fff);
  border: 1px solid;
  border-color: #abbce8 #c3cae0 #b9c8ef;
  -webkit-border-radius: 4px;
  -moz-border-radius: 4px;
  border-radius: 4px;
  -webkit-box-shadow: inset 0 1px 2px rgba(0, 0, 0, 0.25), 0 1px rgba(255,
      255, 255, 0.4);
  -moz-box-shadow: inset 0 1px 2px rgba(0, 0, 0, 0.25), 0 1px rgba(255, 255,
       255, 0.4);
  box-shadow: inset 0 1px 2px rgba(0, 0, 0, 0.25), 0 1px rgba(255, 255, 255,
       0.4);
  -webkit-transition: all 0.25s linear;
  -moz-transition: all 0.25s linear;
  transition: all 0.25s linear;
}

.txtfield:focus {
  outline: none;
/*  color: #525864; */
  border-color: #84c0ee;
  -webkit-box-shadow: inset 0 1px 2px rgba(0, 0, 0, 0.15), 0 0 7px #96c7ec;
  -moz-box-shadow: inset 0 1px 2px rgba(0, 0, 0, 0.15), 0 0 7px #96c7ec;
  box-shadow: inset 0 1px 2px rgba(0, 0, 0, 0.15), 0 0 7px #96c7ec;
}

.txtfield:disabled {
        background: #dddddd;
}

.small {
        font-size: 8pt;
        color: #999;
        font-family: Georgia, Times, "Times New Roman", serif;
}

td.badImage {
        color: #703E3E;
        font-weight: bold;
        text-align: center;
        margin: 1em 0em 1em 0em;
        white-space: nowrap;
        margin-top: 0px;
  margin-right: 0px;
  margin-bottom: 0px;
```

```
  margin-left: 0px;
  border-top-width: 0px;
  border-right-width-value: 0px;
  border-bottom-width: 0px;
  border-left-width-value: 0px;
  border-top-style: none;
  border-right-style-value: none;
  border-bottom-style: none;
  border-left-style-value: none;
  border-image-source: none;
  border-image-slice: 100% 100% 100% 100%;
  border-image-width: 1 1 1 1;
  border-image-outset: 0 0 0 0;
  border-image-repeat: stretch stretch;
  padding-top: 0.5em;
  padding-right: 0px;
  padding-bottom: 0.5em;
  padding-left: 0px;
  overflow-x: auto;
  overflow-y: auto;
  empty-cells: show;
}

a:link.badImage {
        color: #703E3E;
        font-weight: bold;
        text-align: center;
        margin: 1em 0em 1em 0em;
        white-space: nowrap;
        margin-top: 0px;
}

input.invalid {
        background: #FCC;
}

input:disabled {
        background: #D3DCE3;
        color: #D3DCE3;
}
```

pensive.html

```
<!--
    I waive copyright and related rights in the this work worldwide through
        the CC0 1.0 Universal public domain dedication.
    https://creativecommons.org/publicdomain/zero/1.0/legalcode
-->
<#include "header.html">
<div id="top">
<div id="mainFrame" class="image"></div>
</div>
<#include "dialogs.html">
<#include "footer.html">
```

pensive.js

```
/**
 * pensive.js
 *
 * A faithful JavaScript implementation of Java's SimpleDateFormat's format
 * method. All pattern layouts present in the Java implementation are
 * implemented here except for z, the text version of the date's time zone.
 *
 * Author: Tom Parker
 *
 *
 * I waive copyright and related rights in the this work worldwide through
     the CC0 1.0 Universal public domain dedication.
 * https://creativecommons.org/publicdomain/zero/1.0/legalcode
 */

/* ------------------------------------------------------------------------
     */

/** time span of a single image in ms. */
var refreshPeriodMs = 1000 * +'${refreshPeriod}';

var DAY_MS = 24 * 60 * 60 * 1000;

/** format used to display time tags */
var timeFormatter = new SimpleDateFormat("HH:mm");

/** format used to display long time string at top of page */
var dateFormatter = new SimpleDateFormat("d MMMM, yyyy");

/** format used to display initial date in absolutre time box */
var ATDateFormatter = new SimpleDateFormat("yyyy/MM/dd HH:mm");

/** format used to form path to images. Passed in from config file */
var pathFormatter = new SimpleDateFormat("${filePathFormat}");

/** format used to form image suffix. Passed in from config file */
var fileFormatter = new SimpleDateFormat("${fileSuffixFormat}");

/** end time */
var mosaicEnd;

var cellEnd;

/** start time displayed */
```

```
var startTime;

/** end time displayed */
var endTime;

/** span of a single mosaic row in ms. */
var rowSpanMs;

/** span of a mosaic in ms */
var mosaicSpanMs;

var mode;

var dailyMosaic;

var mosaicPosition;

// var dateFormats =
// [/^((19|20)\d{2})[-/]?((1[0-2]|0\d)[-/]?([0-2]\d|3[0-1]))(T|\s*)+([01]\d
    |2[0-3]):?([0-5]\d)$/
// /^\d{4}[-/]?\d{2}[-/]?\d{2}\s*(\d{2}:?\d{2})?$/,
// /^\d{4}-(0\d|1[0-2])-[01]\dT[01]\d:[0-5]\d{}$/
// ];
// /^((19|20)\d{2})[-/]?(1[0-2]|0\d)[-/]?([0-2]\d|3[0-1])(T|\s*)+([01]\d
    |2[0-3]):?([0-5]\d)$/;
var ATDateFormat = /^((19|20)\d{2})[-/]?(1[0-2]|0\d)[-/]?([0-2]\d|3[0-1])(T
    |\s*)+([01]\d|2[0-3]):?([0-5]\d)$/;

/**
 * Handle all initialization stuff here.
 */
function init() {
        mode = "mosaic";
        mosaicPosition = 0;
        mosaicEnd = new Date(getMostRecentEnd());
        cellEnd = new Date(getMostRecentEnd());
        rowSpanMs = hToMs(1);
        mosaicSpanMs = hToMs(3);
        endTime = new Date();
        startTime = new Date();
        if ($("#network option").size() == 1) {
                $("#network").hide();
        }

        parseParameters();
        registerEventHandlers();
```

```
        initalizeDialogs();

        // leanModal init
        $('form').submit(function(e) {
                return false;
        });
        $("a[rel*='leanModal']").leanModal({
                top : 110,
                overlay : 0.75,
                closeButton : ".hidemodal"
        });

        // fire subnet change trigger to get things rolling
        <#if selectedNetwork??>
                $("#network").val("${selectedNetwork}").prop('selected',
                    true);
        </#if>
        $("#network").trigger("change");

}

function initalizeDialogs() {
        // Absolute date
        $("#ATDate").val(ATDateFormatter.format(startTime));
        $("#ATDate").addClass("valid");

        // Mosaic options
        dailyMosaic = $("#dailyMosaic").not(':checked');

}

function getMostRecentEnd() {
        var endTime = new Date();
        var n = endTime.getTime();
        n += endTime.getTimezoneOffset() * 60 * 1000
        return (n - (n % (refreshPeriodMs)));
}

function updateMainFrame(e) {
        if (e != null) {
                if (e.data.mode != null)
                        mode = e.data.mode;
                if (e.data.cellEnd != null)
                        cellEnd = e.data.cellEnd;
        }
```

```
        if (mode == "singlePlot") {
                mosaicPosition = $(document).scrollTop();
                displayPlot();
        } else {
                displayMosaic();
        }
}
/**
 * Parse request parameters. Only used when page is first loaded.
 */
function parseParameters() {
        var param;
        var network;

        param = getUrlParameter("mode");
        if (param != null)
                mode = param;

        param = getUrlParameter("network");
        if (param != null) {
                network = decodeURIComponent(param);
                $("#network").val(network).prop('selected', true);
        }

        param = getUrlParameter("subnet");
        if (param != null)
                $("#" + network + "Subnets").val(decodeURIComponent(param)).
                    prop(
                                'selected', true);

        param = getUrlParameter("cellEndM");
        if (param != null)
                cellEnd = new Date(mToMs(param));

        param = getUrlParameter("mosaicEndM");
        if (param != null)
                mosaicEnd = new Date(mToMs(param));

        /** a single URL parameter */
        param = getUrlParameter("rowSpanM");
        if ($.isNumeric(param))
                rowSpanMs = mToMs(param);

        param = getUrlParameter("mosaicSpanH");
        if ($.isNumeric(param))
                mosaicSpanMs = hToMs(param);
```

```
}

/**
 * register my event handlers.
 */
function registerEventHandlers() {
        $("#nextSubnet").on('click', {
                step : 1
        }, incrementSubnet);
        $("#perviousSubnet").on('click', {
                step : -1
        }, incrementSubnet);
        $("#currentImage").on('click', {
                step : (Math.pow(2, 32) - 1)
        }, incrementTime);
        $("#nextImage").on('click', {
                step : 1
        }, incrementTime);
        $("#previousImage").on('click', {
                step : -1
        }, incrementTime);
        $("#mosaicButton").on('click', {
                mode : "mosaic"
        }, updateMainFrame);
        $(".subnet").on('change', updateSubnet);
        $("#network").on('change', updateNetwork);
        $("#permalinkButton").on('click', populatePermalink);
        $(".positiveInt").on('keyup', function() {
                this.value = this.value.replace(/[^0-9]/g, '');
        });
        $("#ATDate").on('keyup', validateATInput);
        $("#dailyMosaic").on('click', updateDailyMosaic);
}

function validateATInput() {
        var inputField = $("#ATDate");
        var input = inputField.val();

        // var valid = dateFormats.some(function(regEx){return regEx.test(
            input)});

        var valid = ATDateFormat.test(input);

        if (valid)
                inputField.removeClass("invalid");
```

```
        else
                inputField.addClass("invalid");

        $("#absoluteTimeBtn").prop("disabled", !valid);
}

function updateDailyMosaic() {
        dailyMosaic = $("#dailyMosaic").is(":checked");
        $("#mosaicSpanH").prop("disabled", dailyMosaic);
}

function populatePermalink() {
        var URL = window.location.href

        var network = $("#network option:selected").text();
        var subnet = $('#' + network + 'Subnets option:selected').text();

        var q = URL.indexOf('?');
        if (q != -1)
                URL = URL.substring(0, q);

        URL += "?mode=" + mode;
        URL += "&network=" + encodeURIComponent(network);
        URL += "&subnet=" + encodeURIComponent(subnet);

        if (mode == "singlePlot") {
                URL += "&cellEndM=" + msToM(cellEnd.getTime());

        } else {
                URL += "&mosaicEndM=" + msToM(mosaicEnd.getTime());
                URL += "&rowSpanM=" + msToM(rowSpanMs);
                URL += "&mosaicSpanH=" + msToH(mosaicSpanMs);
        }

        var link = $(document.createElement('a'));
        $("#permalinkURL").empty();
        $("#permalinkURL").append(link);
        link.attr('href', URL);
        link.addClass('permalink');
        link.text(URL);
}

/* The subnet changed, now what? */
function updateSubnet() {
        var network = $("#network option:selected").text();
        var subnet = $('#' + network + 'Subnets option:selected').text();
```

```
        $("#subnetName").text(subnet);
        updateMainFrame();
}

function updateNetwork() {
        var network = $("#network option:selected").text();
        $('.subnet:not(#' + network + 'Subnets)').hide();
        $('#' + network + 'Subnets').show();

        $('#' + network + 'Subnets').trigger("change");
}

function updateMosaicOptions() {
        var spanMS;
        var now = getMostRecentEnd();

        if ($("#dailyMosaic").is(":checked")) {
                var tzOffset = endTime.getTimezoneOffset() * 60 * 1000;

                now -= tzOffset
                now += DAY_MS - (now % DAY_MS);
                now += tzOffset;
                mosaicEnd = new Date(now);
                spanMs = hToMs(24);
        } else {
                if (mosaicEnd.getTime() > now)
                        mosaicEnd = new Date(now);

                spanMs = hToMs($("#mosaicSpanH").val());
        }

        mosaicSpanMs = spanMs;

        spanMs = mToMs($("#mosaicRowSpanM").val());
        rowSpanMs = refreshPeriodMs * Math.ceil(spanMs / refreshPeriodMs);

        updateMainFrame();
}

function updateAbsoluteTime() {
        var date = $("#ATDate").val();
        date = date.replace(/(-\d{2}) (\d{2}:)/, '$1T$2');

        date = date.replace(ATDateFormat, '$1-$3-$4T$6:$7');

        dateMs = new Date(date).getTime();
```

```
        dateMs -= dateMs % refreshPeriodMs;

        if (mode == "singlePlot") {
                var end = Math.min(getMostRecentEnd(), dateMs +
                    refreshPeriodMs);
                cellEnd = new Date(end);
        } else {
                var end = Math.min(getMostRecentEnd(), dateMs + mosaicSpanMs
                    );
                mosaicEnd = new Date(end);
        }
        updateMainFrame();
}

/* move selected subnet up or down */
function incrementSubnet(e) {
        var network = $("#network option:selected").text();
        var subnetSelector = '#' + network + 'Subnets';
        var step = e.data.step;
        var idx = $(subnetSelector).prop("selectedIndex");
        var count = $(subnetSelector + ' option').length;
        $(subnetSelector).prop("selectedIndex", (idx + step + count) % count
            );
        $(subnetSelector).trigger("change");
}

/* The time changed, now what? */
function updateTimeLabel() {
        $("#timeSpan").text(
                        dateFormatter.format(startTime) + " "
                                        + timeFormatter.format(startTime) +
                                            " - "
                                        + timeFormatter.format(endTime) + "
                                            UTC");
}

function incrementTime(e) {
        var step = e.data.step;
        var now = getMostRecentEnd();

        if (mode == "singlePlot") {
                var newEndMs = cellEnd.getTime() + (step * refreshPeriodMs);

                if (newEndMs <= now)
                        cellEnd.setTime(newEndMs);
                else
```

```
                        cellEnd.setTime(now);
        } else {
                var newEndMs = mosaicEnd.getTime() + (step * mosaicSpanMs);

                if (newEndMs <= now)
                        mosaicEnd.setTime(newEndMs);
                else
                        mosaicEnd.setTime(now);

                if ($("#dailyMosaic").is(":checked")) {
                        var tzOffset = mosaicEnd.getTimezoneOffset() * 60 *
                            1000;

                        var newEndMs = mosaicEnd.getTime();
                        if (newEndMs == now)
                                newEndMs += DAY_MS;

                        newEndMs -= tzOffset;
                        newEndMs -= newEndMs % DAY_MS;
                        newEndMs += tzOffset;
                        mosaicEnd = new Date(newEndMs);

                }
        }
        updateMainFrame();
}

function displayMosaic() {
        $("#mosaicButton").hide();
        $("#mosaicOptionsButton").show();
        mode = "mosaic";

        var network = $("#network option:selected").text();
        var subnet = $('#' + network + 'Subnets option:selected').text();

        var frame = $("#mainFrame");
        frame.empty();

        var mosaicEndMs = mosaicEnd.getTime();
        var mosaicStartMs = mosaicEndMs - mosaicSpanMs;
        startTime.setTime(mosaicStartMs);
        endTime.setTime(mosaicEndMs);
        updateTimeLabel();
        var table = $(document.createElement('table'));
        frame.append(table);
        table.addClass('center');
```

```
var rowStartMs = mosaicStartMs;
while (rowStartMs < mosaicEndMs) {
        var cell;

        var rowStart = new Date(rowStartMs);
        var rowEndMs = rowStartMs + rowSpanMs;

        var row = $(document.createElement('tr'));
        table.append(row);
        row.addClass("mosaic");

        cell = $(document.createElement('td'));
        row.append(cell);
        cell.addClass("mosaicTitle");
        cell.html(timeFormatter.format(rowStart)
                        + " <span class=\"small\">UTC</span>");

        var cellEndMs = rowStart.getTime() + refreshPeriodMs;
        while (cellEndMs <= rowEndMs) {
                cell = $(document.createElement('td'));
                row.append(cell);
                cell.addClass("mosaic");

                var cellEnd = new Date(cellEndMs);
                var url = network + "/" + subnet + "/"
                                + pathFormatter.format(cellEnd) +
                                    "/" + subnet
                                + fileFormatter.format(cellEnd) + "
                                    _thumb.png";

                var image = $(document.createElement('img'));
                cell.append(image);
                image.addClass("mosaic");
                image.attr('src', url);
                image.on('click', {
                        mode : "singlePlot",
                        cellEnd : cellEnd
                }, updateMainFrame);
                image.on('error', imageNotFound);

                cellEndMs += refreshPeriodMs;
        }

        cell = $(document.createElement('td'));
        row.append(cell);
```

```
                cell.addClass("mosaicTitle");
                cell.html(timeFormatter.format(new Date(rowEndMs))
                                + " <span class=\"small\">UTC</span>");

                rowStartMs += rowSpanMs;
        }

        if (mosaicPosition != 0) {
                $(document).scrollTop(mosaicPosition);
                mosaicPosition = 0;
        }
}

function imageNotFound(e) {
        var URL = $(e.target).attr('src');
        var cell = $(e.target).parent();
        cell.empty();
        cell.removeClass("mosaic");
        cell.addClass("badImage");
        var link = $(document.createElement('a'));
        cell.append(link);
        link.attr('href', URL);
        link.html("Image not<br>available");
        link.addClass("badImage");
}

function displayPlot() {
        $("#mosaicButton").show();
        $("#mosaicOptionsButton").hide();

        endTime.setTime(cellEnd);
        startTime.setTime(cellEnd - refreshPeriodMs);
        updateTimeLabel();

        var frame = $("#mainFrame");
        frame.empty();

        var image = $(document.createElement('img'));
        frame.append(image);

        var network = $("#network option:selected").text();
        var subnet = $("#" + network + "Subnets option:selected").text();
        var url = network + "/" + subnet + "/" + pathFormatter.format(
            cellEnd)
                        + "/" + subnet + fileFormatter.format(cellEnd) + ".
                            png";
```

```
        image.attr('src', url);
}
```

util.js

```
/**
    I waive copyright and related rights in the this work worldwide through
        the CC0 1.0 Universal public domain dedication.
    https://creativecommons.org/publicdomain/zero/1.0/legalcode
*/
function msToM(ms) {
        return ms / 1000 / 60;
}

function mToMs(m) {
        return m * 60 * 1000;
}

function hToMs(h) {
        return h * 60 * 60 * 1000;
}

function msToH(ms) {
        return ms / 1000 / 60 / 60;
}

/* Code from:
 * http://www.jquerybyexample.net/2012/06/get-url-parameters-using-jquery.
     html */
function getUrlParameter(sParam)
{
    var sPageURL = window.location.search.substring(1);
    var sURLVariables = sPageURL.split('&');
    for (var i = 0; i < sURLVariables.length; i++)
    {
        var sParameterName = sURLVariables[i].split('=');
        if (sParameterName[0] == sParam)
        {
            return sParameterName[1];
        }
    }
}
```

Bibliography

[1] Agile Leadership Network. *About Us — Agile Leadership Network*. 2014. URL: http://www.agileleadershipnetwork.org/about/.

[2] anon. *Javadoc Tool Home Page*. 2014. URL: http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html.

[3] anon. *JSR 336: Java SE 7 Release Contents*. 2015. URL: https://jcp.org/en/jsr/detail?id=336.

[4] FreeMarker project. *FreeMarker Java Template Engine - Overview*. 2015. URL: http://freemarker.org/.

[5] GitHub, Inc. *Plans and Pricing*. 2015. URL: https://github.com/pricing.

[6] Jim Highsmith. *Declaration of Interdependence*. 2005. URL: http://pmdoi.org/.

[7] Instrumental Software Technologies Inc. *Earthworm Modules: Sgram Overview*. 1999. URL: http://www.earthwormcentral.org/documentation/ovr/sgram_ovr.html.

[8] Steffen Macke. "Dia Manual". In: 2014. Chap. Introduction. URL: http://dia-installer.de/doc/en/intro-chapter.html.

[9] Martian Software, Inc. *JSAP: Java Simple Argument Parser*. 2012. URL: http://www.martiansoftware.com/jsap/.

[10] OWASP Foundation. *Cross-site Scripting (XSS)*. 2014. URL: http://users.ece.cmu.edu/~koopman/des_s99/sw_testing.

[11] Jiantao Pan. *Software Testing*. 1999. URL: http://users.ece.cmu.edu/~koopman/des_s99/sw_testing.

[12] John A. Power et al. "Preliminary Observations of Seismicity at Mount Pinatubo by use of the Seismic Spectral Amplitude Measurement (SSAM) System, May 13-June 18, 1991". In: *FIRE and MUD: Eruptions and Lahars of Mount Pinatubo, Philippines*. Ed. by Christopher G. Newhall and Raymundo S. Punongbayan. 1997.

[13] Greg Roelofs. *PNG (Portable Network Graphics) Home Site*. 2014. URL: http://www.libpng.org/pub/png/.

[14] Christopher D. Stephens et al. "Seismological aspects of the 1989-1990 eruptions at Redoubt Volcano, Alaska: the SSAM perspective". In: *Journal of Volcanology and Geothermal Research* (1994).

[15] Ray Stone. *leanModal - a JQuery modal plugin that works with your CSS*. 2012. URL: http://leanmodal.finelysliced.com.au/.

[16] The Eclipse Foundation. *About the Eclipse Foundation*. 2015. URL: http://eclipse.org/org/#about.

[17] The jQuery Foundation. *jQuery*. 2015. URL: https://jquery.com/.

[18] The MathWorks, Inc. *MATLAB and Simulink Pricing and Licensing Overview*. 2015. URL: http://www.mathworks.com/pricing-licensing/.