Spectra Thumbnails

# Overview

One of the pieces of functional that was required by the SPECCHIO application was the ability to view thumbnails for any given spectra before the user commits to opening a report for the given spectra. This data had to be inserted into an existing part of the graphical user interface within the Query Builder frame with all backend logic being dealt with in a separate class.

Topics to be covered

* Overview of the design choices and class layout
* General implementation of the thumbnails
* Implementation issues
* Error catching

# Design Decisions including class layout decisions

When the program hits live with loading all thumbnails at once

# Implementation

The implementation began by passing in the selected spectra’s from the query builder frame into a thread that is called by a tree listener within a file browser that was already part of the programs core functionality. It was tasked that this tree listener be updated to pass the unsorted selected spectra id’s into a thread, this thread was created to handle four pass variables.

1. ArrayList of spectra id’s
2. Boolean of by sensor
3. Boolean of by sensor and unit
4. String of current order

The variables are then passed into a class that’s sole purpose is to create a spectral plot for given id’s. Within this class a method was created that checks the id’s against the database’s listings for the given id. It would then pass the information to a series of ArrayLists and Hashtables that were created by the applications creators (add reference) which sort the ids and handle the meta data for each associated id.

A separate function set displayed index is called and checks against a JSpinner with a change listener for the currently selected id and creates a new spectral line plot from MATLAB’s library with a defined width and height set for correct viewing within the GUI.

The selected spectral plot is then drawn to a new JPanel and redrawn and revalidated to ensure correct showing within the GUI. This JPanel is then passed back out to the Query Builder and updated and shown.

# Implementation Issues

When creating the thumbnails a number of issues when dealing with the collection and representation of correct spectral id’s and plots had to be dealt with, along with this Java swing also had limitations with handling the panel redrawing and revalidation when adding the spectral plots as well as the spinner having a correct change listener implemented.

A list of known implementation issues are as follows.

1. Swing and layered panels
2. Swing override of panel sizes
3. Single drawing of plots as to reduce program load
4. Implementing a spinner number model to stop index out of bounds exceptions

### Swing and layered panels

When creating new instances of spectral plots they had to be cast to a panel which was being passed from the query builder. These plots were then being drawn to a panel called spectra\_thumbnail\_panel but after the first had been drawn an issue was found that any subsequent panel would be drawn behind the original panel thus causing unusable plots when multiple spectra had been selected.

A new check had to be implemented when the tree’s action listener was called which would remove any current instance of the spectra\_thumbnail\_panel and redraw and revalidate a new panel when the spinner had been given a new value either typed or clicked on. This change also relied heavily on the design decision to only draw one panel and one spectra at a time in order to reduce load on the system when a huge array of spectra had been selected.

### Swing override of panel sizes

Within the swing framework any child panel will inherit the sizes of the parent even when specifically setting a size for the child panel. This meant that when the panel had no current listing for a spectra thumbnail the panel would be a single pixel wide on initial creation and would not update. It was later found that setting a preferred size for a child node would automatically update the parent node to allow for the child’s preferred size.

### Single drawing of plots to reduce program load

The choice to only draw a single spectra at a time was decided upon after a 200 spectra test showed that rendering every drawing at once would cause small hangs in the application even with the database being run on the local development machine.

Due to this issue the spectra spinner was altered to pass a single value to the spectral plotting method. This substantially reduced the load and reduced the hang in the system to a fraction of a second. With the later addition of a thread to handle solely the spectral thumbnail plots the system has no visible slowdown.

### Implementing a spinner number model to stop index out of bounds exceptions

The JSpinner created had the ability to have any number set by the user and due to the change of the spinner directly influencing the drawing of a spectral plot this would create index out of bounds exceptions. A spinner number model was implemented that had an upper limit of the maximum amount of spectra which is determined by checking the ids ArrayList that is passed into the AddSpectralPlot class. The size of this ArrayList was then checked for its size and then added to the spinner number model. This spinner number model also doesn’t allow for input outside of the models value range which would have also caused an index out of bounds exception.

# Error Catching