Store1 Restoration

Software Technical Specifications

# Summary

Store1 Restoration Utility is software that allows the user to analyze, restore, and recover files or folders from mass store 1 backup. The software includes various functions like search by date range, custom directories search, and individual file/folder history. The goal of this software is to have an application that will make recovering data from the backup more efficient and less time consuming. Currently the backup is made arranged by folder whom names are the date of the backup. This makes searching for a particular file or folder that was backed up over 7 days hard to track (would involve going into each days folder and then subfolders to find what you are looking for) and also find the most recent version to restore. In this scenario the competition for the software would be windows explorer which has decent search capabilities for files and folders. To compete, the Store1 Restoration Utility has been specifically customised for store1 making the search faster than windows explorer and often easier if the user knows what they are looking for.

# Fundamental Processes

# Runtime

Several processes are done when the user initially starts the program. The program is run from the Restore class which creates a GUI object. When the GUI object is created it creates a Restore object which is used for the searching process. The idea behind this was so GUI and Restore can pass variables between each other, ex. the search result list which is passed from restore to GUI to be displayed. After those objects are created the Store1 backup directory is analyzed with a method called getAllDirectories() this method finds the date range of the backup and displays it on screen so the user knows the range they can search. The method also creates a date list which is an array list of date objects representing the range of the backup. Using this list a global beginning and end date are set and used to validate the dates entered by the user. Method setBackupPath() is run to set the global path where the store1 backup is located. This can be changed in the text “BackupPath” if it turns out the original backup path has been changed (the path of the Store1 backup could also vary by user).

# File Search

The file search functionality works by searching a string accompanied with a glob pattern against an Array List of paths. An inner class called “Finder” searches each path one at a time and adds the files that match the glob pattern to an Array List of Files. Once all the paths are searched and all results are matched with the glob pattern the file list is passed to the GUI class to be displayed in the JList list model.

# Folder Search

The folder search functionality works somewhat the same as file search. Folder search uses an array list of paths and searches each path one at a time to make a result list of file objects. To search, the method uses the new NIO library to walk the paths from the path list and obtain all of their subfolders. The depth of the subfolders is defined to be 6 levels deep meaning any subdirectory more than 6 deep will not be found. Through testing average speed and file discovery, six seemed to be the best fit so the user isn’t searching unwanted files and losing precious time (search speed ultimately depends on the systems file disk specifications i.e. hard drive RPM, flash storage, etc.).

Once all the subdirectories are found they are added to a temporary list that will be searched for with the users input string (folder name). If a match is found it is added to a results array list that will be passed to the GUI class to be displayed. The main reason for using three lists in this scenario is for increased speed. The regular IO library is way too slow to search a directory for folders and would encompass the developer searching files and folders and checking for only folders. Therefore NIO was chosen for its speed capabilities when searching a directory for folders.

# Backup Recovery

The backup recovery process allows the user to create a recovery image of the backup between two dates. The process itself is threaded so the user has an indication what is running with updates on what folder is being transferred to the recovery destination. The reason behind threading the recovery process is to stop the GUI from freezing until the process has been completed. Inside the thread an external class called DirectoryCopy is called to copy the source folders to the recovery destination, this is done in a for loop that iterates through an array list of files which contains all the backup folders between the user selected date range. The file list is arranged from oldest to newest so when the copy is done the recovery image holds all the newest versions of the files if an overwrite occurs.

# Restore

The restore process allows the user to restore a particular file or folder to a specific directory. By default a restore directory is created in the users my documents. Clicking the restore button will open a JFileChooser window that defaults to the “RestoredFiles\_Store1” directory that was created in the users my documents. When the user selects save, JFileChooser checks if the file already exists and prompts for an overwrite. When the selection is approved the process checks if the file or folder radio button is checked. If file is checked the copy is done by apache FileUtils from the source file path to the destination file path. If the folder radio button is checked a DirectoryCopy object is created and used with apache FileUtils.copyDirectory to copy the source to the destination.

To get the correct file when the user selects a specific index in the JList a parallel array list is created. One list of type String contains the information the user sees in the GUI being the file name and last modified date. The second array list is of type file which holds the path to the file with that string. This makes it so when the user double clicks a file or folder in the JList the program just has be get the selected index and apply to the file list to open and view the file or folder.

# Classes

# Restore

The restore class handles primarily searching. Two fundamental methods “searchFile” and “searchFolder” are used to find the corresponding folders or files based on a search string. Restore also contains an inner class called “Finder” that uses the nio library to walk the file system and match a glob pattern.

# GUI

The GUI class essentially holds everything together with the buttons included in it acting as individual methods. Since the class primarily initializes graphical objects it will be explained with help of photos to demonstrate the various options.

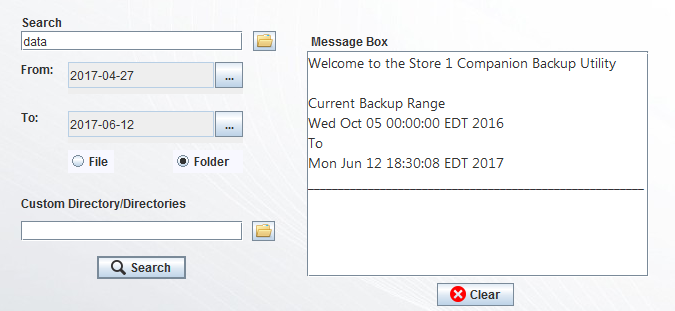


Figure 1, utility search area

Above in figure 1 shows the area where the user can search the backup. The user can either search by date range by using the “From” and “To” date pickers which promptly opens a graphical calendar view for easier selection or the option to search by custom selected directories. By clicking the file icon beside the “Custom Directory/Directories” search box the user can select their desired directories to search (selecting multiple directories requires the user to hold ‘Ctrl’ and click the desired directories). At runtime the backup is initially analyzed and the GUI class retrieves the beginning and end date of the backup judging by the folder names. Using the beginning and end date the backup range is displayed to the user for easier searching.

The file icon beside the search bar allows the user to select an existing file in store1 to search for backed up versions of it. Once all information is inputted the user can press search which passes the text from the search bar to the searchFile method in the Restore class to determine the correct glob pattern and find matching results.

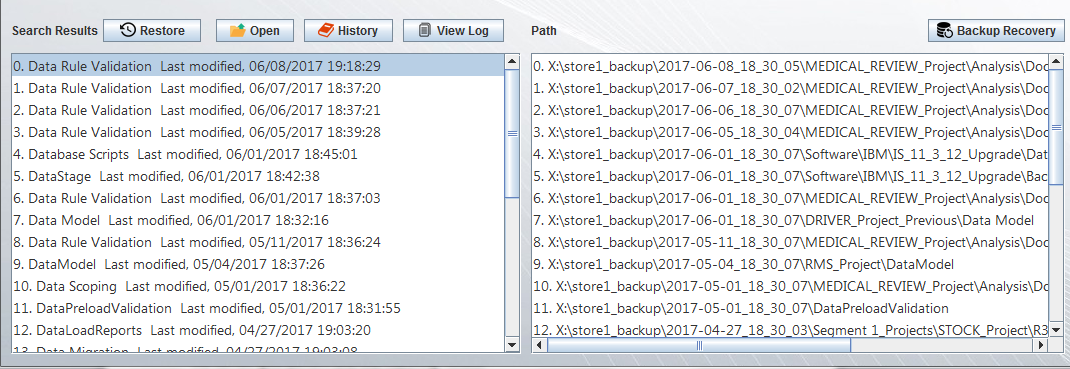


Figure 2, search result area

The above image shows found folder results for the String “data.” At the runtime the two JLists are already created however to show the search data a global DefaultListModel of type String is created which is filled with elements when the results file list is set by the Restore class with the results (fileList is a global array list of file objects in the GUI class and is set by the Restore class). Making the list model global allows the program to changes the contents of the JList dynamically which ultimately allows several searches to be made. Each time a search is made the list model is cleared and filled with the results of the new search this allows the JList to only show current results and also keeps the list model parallel with the file list (file list array list is parallel with list model so when the user selects an index on the JList the corresponding file can be easily retrieved using the same index on the file list). On right side displays the path of each file or folder so the user can track where each results lies if there are duplicates. The Path list is mainly for tracking so there is no action methods associated with it. If a user double clicks the file or folder it will be opened in windows explorer. For files specifically clicking “Open” will open the file with its associated program if applicable.

# InfoGui

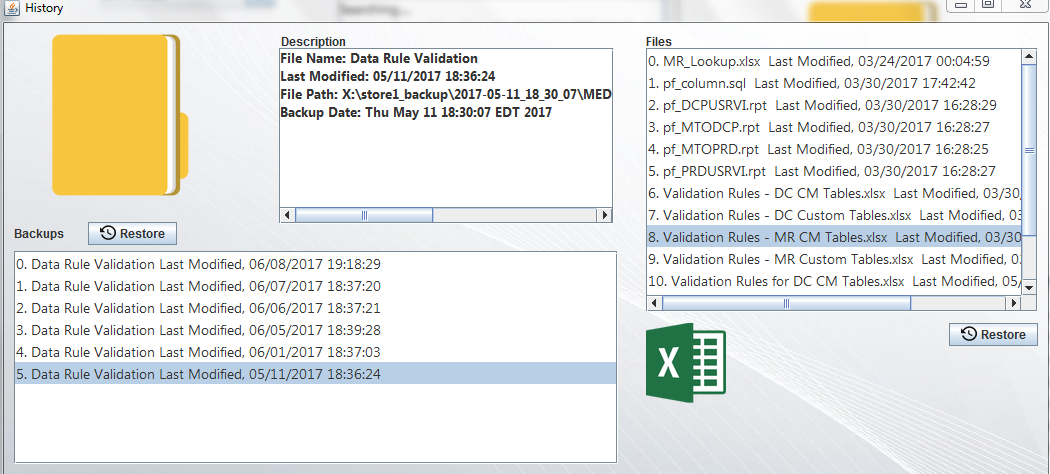
The inforGui class is used to show specific information regarding a file or folder. More geared toward folders the InfoGui class can show the history of a folder as it has been backed up over several days. With this functionality the user can track specific files and recover different versions without having to know the exact name of the file. The InfoGui class also includes restore functionality which acts the same as in the GUI class. The class is invoked by the GUI class when the user clicks the “History” button on the main GUI. 

Figure 3, history view with folder option

In the image above we see multiple versions of Data Rule Validation and the several different days the folder and part of its contents were backed up. To get only the selected file and the days it’s been backed up a temporary array list of file objects is created that only accepts elements from the GUI classes file list that match the selected file name. The temporary list is then passed to the InfoGui class where it sets its own file list which is used to display to the user.

The right side of the view shows the files in that specific folder. Often times a folder will have multiple different subfolders which could make seeing the difference in backup days difficult if just done in windows explorer. With this option when a user selects an older or newer version of the folder backed up the program gets every file in the directory regardless of subfolders. The InfoGui class uses the same inner class as Restore (Finder) to get all the files in that particular folder.

At the top center of the screen a description box is displayed which shows specific description for that file or folder. File name, last modified, and path are easily retrieved with the included methods with the file java library however to get the backup date a regular expression is used to match the date within path string and set it to a date string variable which is then formatted to a date variable using simple date format and then displayed in the description box (description box is a JTextArea).

# Backup Recovery

The backup recovery class handles only one task which is to create a backup image between two dates. To do so the class creates a file list of the backup folders between the date range and copies them from oldest to newest using the DirectoryCopy class which inherently uses apache FileUtils directory copy to complete the operation. The class is invoked by the GUI class when the user clicks “Backup Recover”.

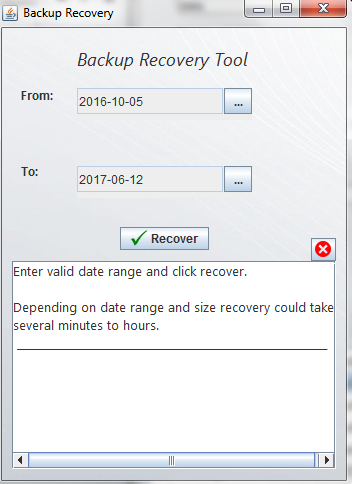
As seen in figure 4 the dates are set to be the range of the backup. It’s important to note that recovery the whole backup would most likely take several hours for the reason being the raw file size of some of the files and also the hardware it is running on. When the user clicks recover with a reasonable backup range a Store1\_Recovery folder is created by default in the Store1\_RestoredFiles folder is the user chooses to use the default settings. The recovery folder will hold the all the files and folders within that date range.

Figure , backup recovery view

# Find Image

The find image class is used to determine the appropriate icon to be displayed when a file or folder is selected. More geared towards files the class will determine the icon based on the extension. For example xlsx would be an excel file and docx would be a word file. To do this several if and else if statements are used and if there is no match then a default file icon is used. The purpose behind the class is to give the user a visual representation of what the file is without having to read each extension. The method “getIconStr” returns the path to icon depending on the file type, after being invoked by the GUI class that same method is used to as the path for the fileIcon jLabel. Example: fileIcon.setIcon(new ImageIcon(InfoGui.class.getResource(fi.getIconStr()))); (in this case ‘fi’ would be the Find Image object)

The size of the file icons are 162x162 and are used in the main GUI class as well as the InfoGui Class.

# Find Image 2

Acts the same as the original Find Image, except it returns the paths to icons which are 80x80. The decision to make two classes that basically do the same thing was a choice of simplicity. Since this class is only used once in the InfoGui class to show the file icons for files in a specific folder (in the history option) the work needed to create flags and switches for differentiating icon size was not worth it for one time use.

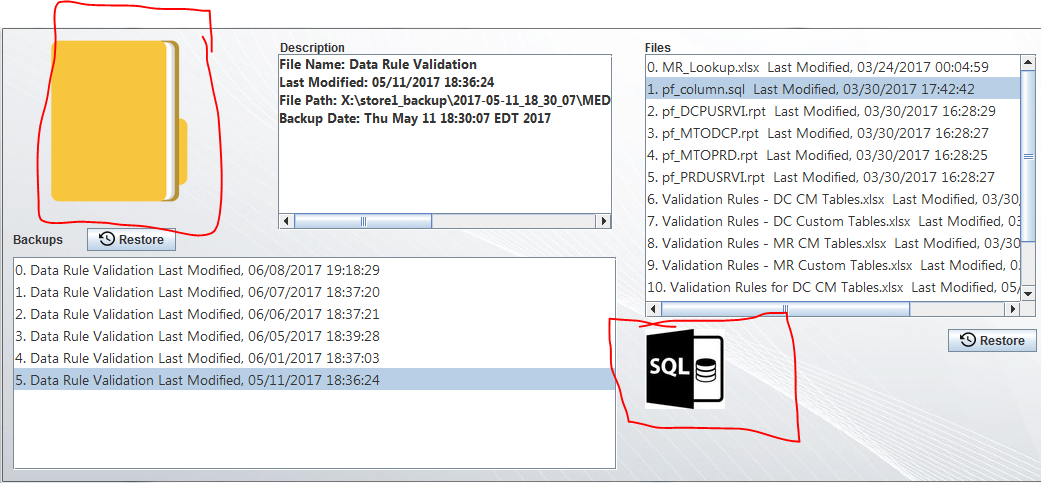


Figure 5, FindImage class use on left and FindImage2 class use on right

# Date Label Formatter

Used to format how the date is displayed in the JDatePicker text box. There are the two overrides stringToValue() and valueToString(). This class was used from a JDatePicker tutorial: <http://www.codejava.net/java-se/swing/how-to-use-jdatepicker-to-display-calendar-component>. Explanation: “As you can see, this class overrides the stringToValue() method to parse a String to a Date object; and overrides the valueToString() method to format the Calendar object to a String. The date pattern to use is **yyy-MM-dd**.

# Directory Copy

This class is used to perform a directory copy from a source to a destination. The classe uses the apache FileUtils library and only requires a source path string and a destination path string to perform the copy. It is used by GUI, InfoGui, and BackupRecovery classes when a directory needs to be copied to a specific location. The class keeps all file metadata intact like date created and date modified so when an overwrite occurs the original file metadata is the same.