IRMIS Global Search Tool
Developer Notes

August 3, 2007

Overview

The global search tool enables users to quickly search a lot of data to find a match for a given keyword. The categories to search include: AOI, IOC, PLC, Component Type, Installed Components, and Epics Record. Within that more categories are available to refine the users search. This tool uses AJAX to call the server for new information, which gives the tool its quick response time.

Change Default Search Length

One of the first variables in the gst_irmis3.js is called ‘defaultSearchStringSize. That number represents the fact that the string entered has to be greater than that number.

String Response Time

If you want to change the response time set for after you type a letter, go to the function updateDisplayCheck(). The listed time is in milliseconds.

XML Fail Debug

If a line is printed to the screen mentioning a XML error, then that error is occurring over in the php files. In order to debug this in the filter functions, comment out all the xml and corresponding lines of code that work off of that xml. There is one line that is already commented out in every filter function

```
//placeObject.innerHTML = pageRequest.responseText;      // For debugging
```

Uncomment this line to display the returned text. This doesn’t do xml, so that error will go away, and this way you can see what is getting returned.

Add New Major Category

1. The only files that need to be updated are:
    a. gst_criteriaSearchResults.php
    b. gst_tabData.php
    c. gst_tableData.php
    d. gst_irmis3.js.
2. In gst_tabData.php and gst_criteriaSearchResults.php, add a new query for each search criteria.
3. In gst_tableData.php, there is one query for the whole major category.
4. There are variables in the file gst_irmis3.js that need to be updated in order to get a new major category to work
5. At the top of the file, add the new major category as a value to the array variable ‘*Category*’ in all caps. Keep “EPICS RECORD” last.
   var Category = new Array ('ALL', 'AOI', ‘SONGS’, 'EPICS RECORD');

6. The rest of the variables to modify are in the *init()* function.

7. The next array variable to change is *'minorCategory'*. It needs to be changed in two spots.
8. In the index 'ALL', add the new major category name as a value to the array.
9. Also, make the new major category as a new index in the array 'minorCategory' in which that index holds a new array.
10. In that new array, the values are all the search criteria. The first value needs to be '–ALL—' and they all need to be in caps.
    minorCategory['SONGS']=new Array('--ALL--','ARTIST', 'SONG NAME');

11. Next, do almost the same thing for the array variable 'headers' except that instead of putting search criteria as the values, put the header information for the table.
    headers['SONGS'] = new Array('SONG NAME', 'ARTIST', 'ALBUM');
12. Next, add the new major category as an index to the array variable 'webPage' which also has a new array for its value.
13. The values in that new array are corresponding websites for the header value in the 'headers' array. The' headers' and 'webPage' array need to be the same length for the same major category index. (e.g. the header title artist in the table may want a link provided for that artist name to take you to the home page of that artist)
    webPage['SONGS'] = new Array('', 'www.findartist.com', '',);
14. For each website listed in that array, a new two-dimensional index is added to the variable 'webVariable' The first index is the new major category name, and the second index is the webpage name. At this index, a new array is made which holds the variables and needed data to send with the webpage. This works in pairs. The first value is the variable name, and the second value is the corresponding index number of the header array value.
    webVariable['SONGS']['www.findartist.com'] = new Array('?artistName=', 1);
    Explanation- the number 1 is the index in the headers['SONGS'] array which corresponds to 'ARTIST'. So this format needs to be done for each website listed in the array.

Make sure to read all comments in the init() function. You have to have the EPICS RECORD be the last value in the array Category because a function is hard coded to skip that value at some point.

Flow

1. When the page is loading, the function *init()* is run which initializes a lot of variables, and some html code is created and inserted. The user has two options and those options are available at any time. He/she can hit the advance search button to run the *displayAdvanceSearch()* or start typing text into the box to run *updateDisplay()*.
2. If the button is hit, then a call to criteria.php is made to fetch the major categories select box and have it returned and displayed using *fetchData()* and *filterData()*.

3. When one of the major categories is selected, it runs *updateCriteria()* which also calls criteria.php to do the same thing, but a select box with minor categories is returned. After the user is done with choosing major categories, he/she can click on the minor categories associated with each major category.
4. For each minor category selected, *checkAll()* is run which makes sure that if any minor category was selected along with the –ALL-- option, then the ALL selection was deselected.
5. Next, *displaySearchCriteriaResults()* goes through the major category's minor select box and displays any option selected with the number of results found from searching the database with the string that the user has entered. That is done by running *fetchData()* and *filterDataCriteriaResults()* which calls the file criteriaSearchResults.php.
6. There is another way to get to the end of step 5. When the user enters text into the text box it runs *updateDisplay().* There are two options in this function. If the advanced search has not been hit yet to display the select boxes, then it runs *displayCriteriaSearchResultsALL().* It is almost the same as before but the only option that is displayed is the –ALL-- for each major category. However if the user already hit the advance search button earlier to display the select boxes, then it runs *displayCriteriaSerachResults()* and does the same thing as in step 5.
7. For every option that is displayed with a search result number higher than 0, a hyperlink is given to that number. When it is clicked, it runs *createTab(),* which creates a new tab at the top and runs *fetchData()* and *filterDataTab().*
8. The call is made to tabData.php in which the search string is sent through the query and all the names are returned and displayed . So in *FilterDataTab()* it creates a list for the data. The list contains items with the name from each row of data returned with an html checkbox next to it. At the bottom of the list, the report tool is created and displayed by running *report().*
9. At the end of that function, if the number of results from the search is different than the number of rows of data, *fetchData()* and *filterDataLog()* are called.
10. This calls logReport.php and writes to a file letting the group know that there was a mismatch during that search.
11. A lot of options are available after this point. The user can run *closeTab()* which will close and delete the current tab that the user is in by clicking on the string "Close Tab" at the top of the list created earlier.
12. The user can switch between tabs using mouse cursor to select different tab. This runs *expantTab().*
13. In the list with all the checkboxes, every time a checkbox is checked it runs *openResults()* which finds all the boxes checked and creates a table by running *fetchData()* and *fetchDataTable().* It calls tableData.php to do that.
14. Then a table is displayed to the user and displays all data for each name and the associated headers to go along with the columns.
15. There is another way to create the table and that is by checking the box 'Select All'. This does the same thing as *openResults().*
16. After the table is created, the user can use the report tool. There are three options. Display in a new window, or save data as a .txt, or as a .csv file type.

17. To display to a new window, the user runs *printView()* which displays the same table in a new window.
18. If the .txt is selected, then *txt(), fetchData(),* and *filterDataReport()* are run with txtReport.php. This will open a new file and save the data into it. After the file is created, txtOpenReportFile.php is called which opens the file.
19. If the .csv is selected, it does the same thing as step 18, but *csv(),* and csvReport.php is called along with csvOpenReportFile.php.