August 03, 2006
Created by Dawn Clemons

Instructions for Adapting the PHP Viewer Report Tool to Other Viewers

Overview

The AOI PHP viewer report tool enables a user to save query results obtained from searching with the AOI viewer. It uses an interface merged with the viewer's so that the user can select sections to report within a particular result. Those selections can then be reported in one of three ways: a text file, a printable web page, or a comma-separated value (CSV) file. This tool makes query results using the AOI viewer more mobile, facilitating areas of work like troubleshooting. The tool also allows for adaptability, such as adding other ways of reporting, handy form add-ons, etc.

This tool has been designed with compatibility with other PHP viewers in mind, so it will be easy to adapt the report tool to your viewer. Much of the code is viewer-nonspecific, so no changes should need to be made to it. A few files are viewer-specific, however, so you will need to make your own versions of them. Which files these are is explained in these directions.

Although in this document I will refer to file names belonging only to the AOI viewer for the sake of simplicity, the report tool has been implemented in the AOI, IOC, and Component Type PHP viewers.

Naming scheme, file paths, and locations

-All .php files unique to the report tool's implementation are prefixed with "`report_`" (except for `action_report_form.php`).
-All .php files unique to the report tool's implementation but specific to a PHP viewer (a subset of the above-mentioned files) are postfixed with "`_aoi`"
-All .php files unique to the report tool's implementation, specific to a PHP viewer, and specific to a particular search page within the viewer are postfixed with that search page's name ("`_ioc_network`", for example).
-Any .php file that is a form's action is prefixed with "`action_`"

Files specified in the report tool's code (for example, `include()`statements) use relative file paths, so if you move files to a different location, you may or may not need to update file paths leading to them, and file paths included within them.

All files unique to the report tool (.php and otherwise) are contained in the "Report" directory.

File interaction

Each PHP viewer comes with a diagram showing file relationships of that viewer. The report tool's place within those files is shown abstractly, since each tool's implementation in a viewer comes with its respective documentation. Consult each file relation diagram and its legend for further details.

The selection form

The report tool begins with `report_startform.php`. It is a generic file that can be used by any other PHP viewer for a report tool. The file starts a form for report creation. Its action is the file `action_report_form.php`. In the code for your PHP viewer, include `report_startform.php` where you would like the selection scope of the form to be started. Then, throughout your code for the viewer's search page, place a checkbox appropriately to represent each unit (section) of reportable information. `action_report_form.php` is expecting the checkbox name to be "`selected`[]." Checkbox values should reflect the section they indicate in an easily readable, nice-looking manner, as the value strings will be used as section labels in the generated report.

Reporting without section selection

This tool assumes you will have several sections to potentially report, a section being one or more visually grouped pieces of information. There is a check in `action_report_form.php` to make sure that at least one selection has been made.

Although the tool is designed with selection options in mind, it will still work if you prefer to report all items from a query result at once. You will simply need to make a hidden form element named `selected[]` and set it as true. Then, rather than checking for selected sections to report in your `report_specific_(report type)_(viewer).php`, you will just report all data the query returned.

Submitting the selection form

A hidden form element (it is expected it to be named "`viewName`") must be present. It identifies the PHP viewer search page that you are reporting to subsequent files so that an appropriate report can be created. It will also be used to display the viewer's name in all reports.

Finally, make sure to have submit buttons at the end of your form, one button per potential report type. The action of the form is `action_report_form.php`. Use the name of the buttons to indicate to `action_report_form.php` what type of report will be generated by clicking on it (text file: `name` = `text`, printable version: `name` = `print`, CSV: `name = csv`, etc.)

You can also add a text box for additional comments a user may wish to include in the report. It is expected that the name of the comments `textarea` will be "`comments`". No changes need to be made to any file if the form doesn't have a text box for comments.

Creating your own files for reporting specific data

Utilizing only the three report types options provided, you must still author your own way of printing information that is specific to your viewer. Inspect `report_generic_text.php` and `report_generic_print.php` to see exactly what will be left to write/display. Generic information involves: identification of IRMIS, viewer name, timestamp, and comments. The `report_generic_(report` `type).php` files also then divert based on viewer using the hidden form element `viewName`. In `report_generic_(report type).php` you must have a check for your viewer type, and then within the check include a file that will report your specific data. Files that report specific data are named `report_specific_(report type)_(viewer name).php`. Also, you must include files with result object definitions for your viewer in the `i_common.php` file in the `report` directory. Note: See the "I_common" section of these instructions.

<u>For additional report types:</u>
-You must add a button for each new report type outside of the three currently available.
-You will need to put a check for the report type in action_report_form.php that includes a `report_generic_(new report type).php.`
-You will have to write all of your own files

> -generic information:
> > `report_generic_(new report type).php`
>
> -specific information:
> > `report_specific_(new report type)_(viewer).php`
>
> -Look at
> > `report_generic_(report type).php` and
> > `report_specific_(report type)_(viewer).php`
>
> for examples.
> -If your new report type will involve file I/O, you'll also need to create a
> > `report_savedialog_(new report type).php` file.
>
> -Look to
> > `report_savedialog_(report type).php`
>
> for examples.

<u>Global Variables and `$_SESSION`</u>
The report tool currently has only one global variable, `$_SESSION['genFileName']` which specifies a server-side file name used to create a text report by writing to this file. Make sure your viewer has a `session_start()` so that this variable can be utilized. The global variable is initialized in `i_common.php.`

Temporary files for server-side file I/O (creating a file to transfer to a client, for example) must be held in `/tmp` on the Maia server, so make sure this file path precedes the file's name when assigning the path string to `$_SESSION['genFileName']`.

You will need to create this file in `/tmp` and change its permissions to have read and write access. At the shell prompt, you can use the command `% chmod a+rw filename` to do this.

<u>`I_common.php`</u>
This file resumes the current session and includes code for the objects in which the query result is stored. Only the code needed for the particular viewer used is included, so when adding includes, precede them by a check for the viewer.

JavaScript

There are two instances of client-side scripting within this report tool. The first is a dialog alert if the user attempts to create a report without having selected (if applicable) any sections to report.

The second is a script that enables selecting or deselecting all checkboxes in the form. It is a simple JavaScript function which has been provided in `selectDeselectAll.php.` The script consists of a function that takes two arguments, the form `id`, and `true` or `false` (representing selection or deselection, respectively). You can see the function in use in `report_submit_aoi_editor.php.`

Printer-friendly Version's Common Look

There are two jpg's which are necessary for the printable page to be displayed correctly. They are `irmisLogo.jpg` and `irmisText.jpg`. There is a cascading style sheet `printable_report.css` that creates simplistic tables without background color. It should fit most needs for printing. In your printer-friendly report, do not have any href's (links), and do not display any non-pertinent information like columns that contain links.

Viewers that currently have the report tool implemented in them:

Related suffixes are in parentheses

AOI:

AOI search results `(_aoi.php)`

IOC:

IOC search results `(_ioc.php)`
General search results `(_ioc_general.php)`
Network search results `(_ioc_network.php)`

Component Type:

Component Types search results `(_comp.php)`
Details search results `(_comp_details.php)`
Instances search results `(_comp_instances.php)`

Additional miscellaneous helpful information

Remove all `echo`s in all db files.
Remove trailing whitespace in `common/i_common_common.php`.

For reports that manipulate a page's header like text and csv (any report that is downloaded), there cannot be any `echo`s, `print`s, or `printf`s present, and there can be no whitespace before and after `<?php ?>`

For csv reports, put field values in quotes so that it won't matter if the value has the delimiter in it.

  Example: Hi, there → (saved as csv) "Hi, there" → (in spreadsheet) Hi, there

Put additional quotation marks around things already in quotes to preserve the original quotation marks.

  Example: "Hello there." → (saved as) ""Hello there."" → (in spreadsheet) "Hello there"

A header attachment's "Content-type" doesn't seem to matter in Windows when a file extension is provided with the file name.

"Content-type: application/vnd.ms-excel" is MSExcel.
"Content-type: application/vnd.oasis.opendocument.spreadsheet" is Open Office spreadsheet, but this type also works suffices MSExcel.

Spreadsheet Import Wizard default delimiters (Excel, Open Office):

.txt extension:	tab
.csv extension:	comma

Note: All text files are simply tab-delimited at the present time. No attempt has been made to line up the information in columns.