RSM documentation

At this point RSM is still being designed. This documentation is intended to give an idea how RSM works, and some of the details may be inaccurate due to daily changes is code.

This especially applies to:

The arguments, parameters listed for functions

Names of classes, variables, etc

For accuracy look to the current RSM code and the scripts that use RSM, not this document.

i.e. If copying code to be used in a script, copy it from a working script, not from this document.

Jared

RSM teminology

|  |  |
| --- | --- |
| Chain Object – the rsm object that controls flow of scripts as they process forms. Each script consists of one or multiple steps, each step is a form. | At start of each script is created and initialized |
| Step Object – the rsm object that represents a “page” of a script, which is a form that does some processing. | At start of each script all the possible steps (pages) are created and added to the chain object. After this the chain object controls step processing. Each step in the script is an object that extends step, which must define several abstract functions. |
| Emit  Emitter class/object | Prints to the user interface. This also includes exporting to PDF and excel. When the emitter classes are used, (eventually) exports will be automatically handled. (not all things can be exported, only a subset of what is typically on a report, but not on menus, banners, etc). So instead of “Print ‘<table class=’report’>’”, using the emitter the code is emit->table\_start(‘report’), and then (eventually) the report can be exported without special code. |
| Form  Form Object |  |

|  |  |
| --- | --- |
| Term and short explanation | Details |
| Application | Application – such as payroll, shopping, etc. Each application has a kernel and appglobals, common to each script in the application |
| Kernel emitter – shared by all applications. Kcmkrn emitter inherits rsm\_emitter | To the rsm emitter additional functions are added such as printing the banner which all of the kcm systems share |
| App Globals - kcmkrnGlobal | App-Globals not used by rsm, but passed to all the step functions. This includes security information, which is now the same for all kcm systems, but can be overridden by any system – example: payroll adds the security status of payroll-master. |
| <system> examples:, rstGlobals, adminGlobals, payGlobals, utilityGlobals , all inherits kcmkrnGlobal | There are some items which must be defined here for the kcm kernel to work, including menu setup, the name of the emitter, etc. |
| Common class/object |  |

Standard step actions terminology (only used in this document).

|  |  |
| --- | --- |
| Term | Explanation |
| SCRIPT\_START | The script is started, including coming from a menu or from self. If coming from self, there are several possibilities and this step will quickly turn into another of the other SCRIPT\_START\_\* step s |
| SCRIPT\_START\_FRESH | No hidden variable for step – this mean’s starting from scratch, such as coming from menu |
| SCRIPT\_START\_POSTDATA | Has posted data – this means called from same document from <form action=”<self>”. There must be hidden item indicating initiating step. |
| SCRIPT\_START\_REDIRECT | Has no posted data – this means had posted data, and was redirected here with some posted data |
| SUBMIT\_PRESSED | Submit button is pressed resulting in <form action=<same php script> being processed.  There should always be a hidden item specifying what the current step is |
| SUBMIT\_CALLEDFROM | The script that called this script |

Step abstract functions.

|  |  |
| --- | --- |
| Function | Purpose |
| rsmStep\_submit\_process | Place to put in code to process a submit. Usually, if applicable, check if cancelled and if so do not save posted data. Otherwise accept data from posted data and validate. If validated process data (save record, etc), otherwise copy posted data to array with a token to retrieve this data. |
| rsmStep\_submit\_accept  ($chain, $common, $rstGlobals, $data) | Place to put in code to process posted data. In some cases $data is the $\_post, and in other cases (such as a validation error), it’s the copy of the posted data that is in $\_session. |
| rsmStep\_submit\_validate | Place to put in code to |
|  |  |
| rsmStep\_init\_form | Place to put in code to define the fields on the form, including content of selects, etc. At his point may need to read in data, as the form controls can be data dependent. |
| rsmStep\_init\_output | Place to put in code to initialize the emitter. This includes initializing the banner (need the page title), initializing the menu (mostly automatically done, but there are script specific options), and customize the header for adding css stylesheets, internal styles, special java scripts, etc. |
| rsmStep\_output\_headers | Place to put in code to emit the top non-scrolling portion of the gui (not including banners, menus, error messages). Should be short (1 or two lines) to leave room for content. Good place for special instructions or filter options, |
| rsmStep\_output\_content | Place to put in code to emit main content. The most common uses are:  A list of kids, staff, etc to select one to edit.  A |
| rsmStep\_output\_footer | Place to put in code to the botton non-scrolling portion of the script. Should be short (1 or two lines) to leave room for content. Good place for submit, cancel, next buttons, or filter options. |

**Class rsm\_appData**

|  |  |
| --- | --- |
| **Class Functions** | **Example** |
| General Notes: | <stepId> is the id of the step/page/form  <varId> if the name of the field used in the form input  <variable> is the variable the value goes into |
| updateIfPosted ( <stepId>, &<variable>, <varId> )  Gets value from posted array (or copy of posted array in $\_session) | $chain->updateIfPosted (2,$this->kid->firstname,’firstname’); |
| getPosted ( <stepId> , <varIid> )  Gets a value | $kidId = $chain ->getPosted(1,‘ kidId’);  If ($kidId === NULL) {  // do not process other pages, kidId has not been posted  } |
| setPosted ( <stepId> , <varIid> , <value>)  Generally used to add a value for use by other steps. Could also be used to re-format a value | $chain->updateIfPosted (2,$this->parent->phone,’phone’);  $this->parent->phone = formatPhone(,$this->parent->phone);  $chain->setPosted ( 2 ’phone’ , $this->parent->phone) |

|  |  |
| --- | --- |
| **Abstract Functions** | **Example** |
| rsmAppData\_retrieve  ($chain, $rstGlobals)  Place to put in code to process posted data. In some cases $data is the $\_post, and in other cases (such as a validation error), it’s the copy of the posted data that is in $\_session. | Function rsmAppData\_retrieve ($chain, $rstGlobals) {  $this->kid = new kcm\_kid;  $kidId = $chain ->getPosted(1,‘ kidId’);  If ($kidId === NULL) {  Return; // still on 1st page  }  $kid->readData($rstGlobals->gb\_db,$kidId);  $chain->updateIfPosted (2,$this->kid->firstname,’firstname’);  $chain->updateIfPosted (2,$this->kid->grade,’grade’);  } |
| rsmAppData\_ validate  ($chain, $rstGlobals)  Place to put in code to validate data. | Function rsmAppData\_validate ($chain, $rstGlobals, $data) {  If ($this->kid->firstname == ‘’) {  $chain-> setError (‘kidFirst’, ‘First Name is required’);  // first argument is key used on form object  // for this item  }  If ($this->kid->firstname == ($this->kid->lastname) {  $ chain ->setError(‘kidFirst’, ‘First and last name must be different’);  }  } |
| **Recommended Functions** | **Example** |
| rsmAppData\_ save  ($chain, $rstGlobals) | Function rsmAppData\_validate ($chain, $rstGlobals) {  If ($chain->isUpdatedByOtherTab(‘kid’, $this->kidid)) {  // ???????  }  $this->kid->save($rstGlobals->gb\_db);  // create status message  $chain ->setSaved(‘kid’, $this->kidid)  } |

Step (form) processing – example of step that selects record to edit. Examples: List of kids, list of staff, game history which is report listing games, etc.

|  |  |
| --- | --- |
| User action | Resulting Sequence (same for both examples) |
| Button is submitted to select specific record (kid, game, staff, etc (possible other controls are involved such as checkboxes, selects, etc which will be posted) | This form has hidden item of “step=<current step id>”  Form submitted via “action=<same form>”.  If step one unique token is generated for start of this sequence (?? Consider tabs later)  Posted data is read and saved into $\_session keyed by token – if posted data already exists, then posted items are added to existing data (data from other pages are kept) |
| Button is submitted to change filter (list games played this week, list games of specific kid, etc) (possible other controls are involved such as checkboxes, selects, etc which will be posted) |  |
| Saving data | Check history to make sure record was not saved by another tab since data was posted  Save record – create history record for above check  Start at beginning |

Step (form) processing.

|  |  |  |
| --- | --- | --- |
| id | operation | Next operation |
| 1 | select record to edit  (examples: list of kids, game history report) | Menu option -> new script  Select record -> usually same script  No problem, even if if new tab (no specific data yet) |
| 2. | Start editing specific record(s)  Parameter In: record Id, can be none if new record  Status needed:  None if new record  \*\*\* Time stamp when records were read  (possibly save original data that was read) | (see below) |
|  | From 2 -> Menu Link |  |
|  | All submit buttons (step 1) | Form executes script in form action=”<same script>” |
|  | From 2 -> Cancel | If cancel, then proceed in script to specified page for cancel |
|  | Most submit buttons (step 1) |  |
|  | From 2 -> Back (multi page scripts, but no data involved) |  |
|  | From 2 -> Back page (multi-page data forms) |  |
|  | From 2 -> Next page (multi-page data forms) |  |
|  | From 2 -> Submit (Save) | Validate data  If invalid save posted data to $\_session and redirect to same script  Save record(s) with new data |
|  | From 2 -> Reset |  |
|  | From 2 -> Refresh |  |
|  | From 2 -> Browser back |  |
|  | From 2 -> Browser forward |  |