Query Builder

- The query builder element was designed to allow clients to perform ad hoc queries.
- Still under development - expect potential changes in the short term.

```
query_builder
  query_builder_fields
    query_builder_field
    ...
  query_builder_data
```
Query Builder element

- The query builder element is added using the add_element method.

```php
$qb =& $body->add_element('query_builder', $params);
```

- $params is a associative array of attributes:
  - name - name of the query builder
  - submit_handler - URL to direct to when submit is pressed
  - query_builder - 'on' or 'off'
  - report_builder - 'on' or 'off'

Query Builder Fields

- Query builder fields must be enclosed in a query_builder_fields element.

```php
$qbf =& $qb->add_element('query_builder_fields', $params);
```

- $params is a associative array of attributes:
  - default_table - default table to use for query_builder_field elements
Query Builder Field

- query_builder_field elements are used to describe your table structure to the query builder

```php
$qbf-&gt;add_element("query_builder_field", params, display)
```

Query Builder Field

- `params` is an associative array of attributes:
  - name - column name
  - table - table the column is located in
  - input_type - type of data - currently only text and list
  - sortable - '0' or '1' - if no report section, specifies that this field can be used for sorting
  - convert - allows conversion of the data
  - list_source - array of data to be used with list input_type
  - suppress_search - '0' or '1' - suppress from search section
  - suppress_report - '0' or '1' - suppress from reporting section
  - format - format for data (dates only)
  - display_format - format for the output (XSLT format string)
  - input_hint - hint for the user on the format of the data
  - default_criteria - starting criteria in the search section

Query Builder Field

```php
search.php
<?php

$qbf-&gt;add_element("query_builder_fields", array("default_table" =&gt; "orgs"));
$qbf-&gt;add_element("query_builder_field", array(
    "name"=&gt;"shortname",
    "input_type"=&gt;"text"), "Short Name");
$qbf-&gt;add_element("query_builder_field", array(
    "name"=&gt;"description",
    "input_type"=&gt;"text"), "Description");
$qbf-&gt;add_element("query_builder_field", array(
    "name"=&gt;"advisor",
    "input_type"=&gt;"text"), "Advisor");

display($presentation, "Search Organizations");
```
Query Builder Data

search.php
<?php
[...]
$qbf=&$qb->add_element("query_builder_fields", array("default_table" => "orgs"));
[...]
$qbf->add_element("query_builder_field", array(
  "name"=>"advisor",
  "input_type"=>"text"), "Advisor");
$qb->add_element("query_builder_data");
display($presentation, "Search Organizations");
?>

Processing the report

- The report should be processed if
  $_REQUEST['query_builder_name']['run'] or
  $_REQUEST['query_report_name']['navigation']['start_record'] is defined

Processing the report

- The select class of the database library was designed specifically to work with the query builder
- The select class contains the following methods:
  tables() - defines the tables used by your select
  add_limit() - defines intelligent limits (where clause)
  report_from() - handles query from query builder
  execute() - executes the query
  add_output() - defines additional tabulator attributes

Processing the report

search.php
<?php
[...]
$dbh->select();
$select->tables('orgs');
$select->report_from($_REQUEST['orgs_query']);
$select->execute();
$select->add_output($body, array(
  "title":"Organizations",
  "name":"orgs_query_output"));
} display($presentation, "Search Organizations");
?>
Debugging databases

```php
$search.php
<?php

$search = $dbh->select();
$search->db->statement_tracking = true;
...
$body->add_element('para','','<sourcecode>' .
    htmlentities(wordwrap($select->db->statements())) .
    '</sourcecode>');

->
```

Processing the report

```php
$search.php
<?php

$qbf =& $qb->add_element("query_builder_fields",
    array("default_table" => "orgs"));
$qbf->add_element("query_builder_field", array(
    "name" => "shortname",
    "input_type" => "text",
    "convert" => "all:lowercase"), "Short Name");
$qbf->add_element("query_builder_field", array(
    "name" => "description",
    "input_type" => "text",
    "convert" => "all:lowercase"), "Description");
$qbf->add_element("query_builder_field", array(
    "name" => "advisor",
    "input_type" => "text",
    "convert" => "all:lowercase"), "Advisor");

display($presentation, "Search Organizations");
?>
```

Processing the report

```php
$search.php
<?php

$fields->add_element("query_builder_field", array(
    "name" => "startdate",
    "input_type" => "text",
    "convert" => "value:date",
    "format" => "%m/%d/%Y",
    "input_hint" => "e.g. 1/15/2003"), "Start Date");
$fields->add_element("query_builder_field", array(
    "name" => "enddate",
    "input_type" => "text",
    "convert" => "value:date",
    "format" => "%m/%d/%Y",
    "input_hint" => "e.g. 1/15/2003"), "End Date");

```

Processing the report

```php
$search.php
<?php

$fields->add_element("query_builder_field", array(
    "name" => "advisor",
    "input_type" => "text",
    "default_criteria" => "eq",
    "convert" => "both:lowercase"), "Advisor");

display($presentation, "Search Organizations");
?>
```

Processing the report

```php
$search.php
<?php

$fields->add_element("query_builder_field", array(
    "name" => "startdate",
    "input_type" => "text",
    "convert" => "value:date",
    "format" => "%m/%d/%Y",
    "input_hint" => "e.g. 1/15/2003"), "Start Date");
$fields->add_element("query_builder_field", array(
    "name" => "enddate",
    "input_type" => "text",
    "convert" => "value:date",
    "format" => "%m/%d/%Y",
    "input_hint" => "e.g. 1/15/2003"), "End Date");
```

Processing the report

```php
$search.php
<?php

$fields->add_element("query_builder_field", array(
    "name" => "advisor",
    "input_type" => "text",
    "convert" => "all:lowercase"), "Advisor");

display($presentation, "Search Organizations");
?>
```

Processing the report

```php
$search.php
<?php

$fields->add_element("query_builder_field", array(
    "name" => "startdate",
    "input_type" => "text",
    "convert" => "value:date",
    "format" => "%m/%d/%Y",
    "input_hint" => "e.g. 1/15/2003"), "Start Date");
$fields->add_element("query_builder_field", array(
    "name" => "enddate",
    "input_type" => "text",
    "convert" => "value:date",
    "format" => "%m/%d/%Y",
    "input_hint" => "e.g. 1/15/2003"), "End Date");
```

Processing the report

```php
$search.php
<?php

$fields->add_element("query_builder_field", array(
    "name" => "advisor",
    "input_type" => "text",
    "convert" => "all:lowercase"), "Advisor");

display($presentation, "Search Organizations");
?>
```
Processing the report

```php
$search.php

<?php

$advisors = $dbh->queryall_list(
    'select unique advisor, name from orgs, person where
    advisor=uniqueId');

$fields->add_element("query_builder_field", array(
    "name" => "advisor",
    "input_type" => "list",
    "list_source" => $advisors), "Advisor");

display($presentation, "Search Organizations");
?>
```

Query Builder Futures

- The following have been discussed as future enhancements to Query Builder
  - Cleanup interface
  - Improved input controls (date, validation, etc.)
  - Default values and default report fields
  - Saved queries and reports
  - Other output options (CSV, PDF, etc.)
  - Query Builder Class - simplified programmer interface

Logging out

- The end_session() function can be used to end a session.
- The function returns true on success or false on failure.
function addnav(&$presentation) {
    $navmenu = array("Home" => "index.php",
                    "View" => "view.php",
                    "Search" => "search.php",
                    "Logout" => "logout.php");
    $presentation->add_navigation($navmenu);
    $suptext = array("For help, please contact the ".
                    '<link href="http://www.muohio.edu/supportdesk/"MCIS
                    Support Desk"></link>");
    $presentation->add_supplemental($suptext);
    $relmenu = array(
        array("title" => "MCIS Home Page",
               "content" => "The main MCIS webpage",
               "href" => "http://www.muohio.edu/mcis/"),
        array("title" => "Time Entry",
               "content" => "The MCIS Time Entry Application",
               "href" => "http://www.muohio.edu/teps/"));
    $presentation->add_related($relmenu);
}
addnav($presentation);

function end_session() {
    return false;
}

if (end_session()) {
    $body->add_element('para',NULL,'You have successfully logged out. ');
} else {
    $body->add_element('para',NULL,
                       'An unexpected error has occurred. You may not have been logged out. ');
}
display($presentation, "Logout");

- The application developed in class was a simple example. In a real application, the following should be done differently:
  - Better error handling:
    print "An error occurred";
  - New Organizations, Delete Organizations
  - WAS enabled
    'was_enabled' => '1'
  - Better organization
    separate organization logic into it's own file
  - classes/OOP
  - Possibly cache database results

Questions?
Library change

- The attribute() method has been retired.
- We used the attribute() method in view.php

```php
$modifiable = $auth->check_authorization("PHP Training", "general", "modify");
$table =& $body->add_element('tabulator', array(
    'title' => "Organizations",
    'name' => 'orgs",
    'navigation' => "on",
    'record_count' => $recordCount,
    'start_record' => $start,
    'show_records' => $show,
    'sort_by' => $sort_by,
    'sort_direction' => $sort_direction,
    'link_using' => ($modifiable === true ? 'shortname' : '')
));
```