Web Interfaces to Backend Databases
Using PHP+MySQL

Dee Dee Daniel
Pace University Seidenberg School of CSIS, White Plains, NY 10606, USA

ABSTRACT
There are currently about 80 Doctor of Professional Studies (DPS) in Computing dissertations posted to the Pace University library catalog. This work extends earlier work to organize the DPS dissertation listings using a database and web interfaces. The database was to be used to organize the dissertation data in order to make the retrieval of DPS information easier via the web interface and an easy-to-use web interface to add, delete, and modify tables and queries was provided.

1. INTRODUCTION
The web interface created earlier with PHP logic and MySQL was functional. However, the UI (User Interface) was not suitably designed for the end users. The goal of this new effort was to:

- Modify the existing tables and queries and fix UI issues
- Fix errors in the database such as advisor list, committee members, and date of defense
- Check each dissertation on the library site and make sure all are added to the database
- Update database by using the CMS admin site and add missing dissertations from the library catalog

2. USER INTERFACE DESIGN
2.1. Team 1: Web Interface Design
The UI was built using PHP, HTML, a CSS grid system, jQuery, along with a few jQuery libraries. The grid system is used in order to avoid having to style every item in order to position throughout the page. Using a grid system helps by placing a class such as grid_6 to specify which container the DOM element will be placed in. jQuery was used because of its powerful selector based system. Using a DOM element’s id one can select an Item the element and manipulate it to one’s specification. jQuery also allows the use of classes for selectors. This allows one to manipulate multiple items using the same class just as it would be done using CSS.

The theme in the UI was built using jQuery’s theme roller. This is used in conjunction with the jQuery framework. Using those selectors styles were added to many different DOM elements by including a UI library and the custom CSS class (Figure 1).

![Figure 1. UI library and the custom CSS class](image)

2.2. Database Management
A. Accessing the Database
Figure 2 shows the PHP code that is used to connect to the MySQL schema. The first line creates a variable with the connection parameters which are the server, the username, and the password that would be used to access the schema. In this instance the server is localhost, the username is cs691-team06-2012, and the password is 397j1C3d. The second line consists of a conditional which asks if the connection was correct. If it is not achieved, the connection is killed and an error is displayed if logging is enabled. The last line uses the connection and selects the database within the MySQL schema.

![Figure 2. PHP code used to connect to MySQL schema](image)
B. Database Querying

Once the connection has been established and the database has been selected an SQL query can be run in order to obtain information from the database to display on the page. Figure 3 shows the select statement that is used to grab the information for the Advisor/Dissertation report. The result of this query is stored in a variable called $sql.

```
$sql = "SELECT b.Name, count(a.author) dissertations, avg(a.monthsToCompletion / 12) yearsToCompletion, count(a.committeeMember1) + count(a.committeeMember2) + count(a.committeeMember3) + count(a.committeeMember4) + count(a.committeeMember5) sumPub, count(a.primarySubjectCategory) + count(a.secondarySubjectCategory) + count(a.tertiarySubjectCategory) sumCatCount, count(a.primaryMethod) + count(a.secondaryMethod) + count(a.tertiaryMethod) sumCount, count(a.methodNeedsCount) WHERE b.Name = a.committeeMember1 GROUP BY b.Name";
```

Figure 3. SQL Query - Advisor/Dissertation report

C. Using Database Data

Once we store the result in a PHP variable we are able to iterate through it and display it on the page as necessary the format to do so is “$.row[‘Column Name’] .”

```
while($row = mysql_fetch_array($result)) {
    echo"<tr class='body'>;
    echo"<td"> $row[‘name’] ."</td>;
    echo"<td"> $row[‘dissertations’] ."</td>;
    echo"<td"> $row[‘yearsToCompletion’] ."</td>;
    echo"<td"> $row[‘subjectCategory’] ."</td>;
    echo"<td"> $row[‘methodNeeds’] ."</td>;
    echo"<td"> $row[‘countAdv’] ."</td>;
    echo"<td"> $row[‘pub’] ."</td>;
    echo"<td"> $row[‘subCatCount’] ."</td>;
    echo"</tr>
    }
```

Figure 4. Display Data Code

D. End Result

The information from the Database is displayed in the UI within two sections, tables and reports. Within the tables menu, you can select each table and view a full dump of the selected tables. Within the reports menu you can select individual reports which are Averages, Dissertation Count, Subject/Years, and Method/Years.

Every page uses a table to display the information. The table uses a smart library that allows for pagination and the amount of information displayed per page is set to 10 rows by default. The smart library also uses allows for an intelligent search which searches through every column.

File Structure in $site_root = ‘~/f13-it691-s01/dps’:

It is always expected that HTML provides an illustrative layout, it was therefore critical that coding be done to enable administration views. A total of seven pages were created comprising of one log in page, 3 view pages with “person, dissertation, and external publication”. Html was used in creating forms in regards to coding of the edit pages. PHP was used to process the data entered in the various fields available and then stored in the database in the form of My SQL.

All the information gathered from the original database were inserted in the ‘easy to read table’ using HTML. This application was also used not only for data entry for the languages but also for purposes of displaying the content to the users.

In creating a visually appealing page that would excite the users, the CSS code was utilized and the results were pleasing to the users. The use of CSS was chosen because it has the merit of allowing the website designer to format the web pages and organize them as he/she deems fit. In this project, CSS therefore helped in the creation of all the link areas, the forms, tables as well as the colors.

After establishing the field requirements for each web page, HTML was utilized for purposes of generating the ‘edit’ views. For instance, in the page “Create person”, only two fields were provided for but in the page “Create external publication”, seven fields were created.

Just like the ‘listing views’, the whole process of development was just similar. It was however necessary that the specific titles and the number of rows that each table required had to be defined prior to creating the remaining pages. Figure 5 shows a sample Dissertation Listing page.
2.2. UI and Front-end updates to public website

Implementing the auto-completion, with the titles ‘author’, ‘committee member’, ‘advisor’, ‘category’, the author’s preset ID. Through this, irrelevant values are not returned as answers to the run query.

2.3. Team 2: Web Interface Design

PHP implementation

The application is built using a Model-View-Controller paradigm, wherein separation is maintained between logic that affects the data objects, the core application functionality and the display generation.

Application File Structure

```
<table>
<thead>
<tr>
<th></th>
<th>ajax</th>
<th>_controllers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>autocomplete.php</td>
<td>authentication.php</td>
</tr>
<tr>
<td></td>
<td></td>
<td>database.php</td>
</tr>
<tr>
<td></td>
<td></td>
<td>output.php</td>
</tr>
<tr>
<td></td>
<td></td>
<td>validation.php</td>
</tr>
<tr>
<td></td>
<td>_objects</td>
<td></td>
</tr>
<tr>
<td></td>
<td>_object_base.php</td>
<td>category.php</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dissertation.php</td>
</tr>
<tr>
<td></td>
<td></td>
<td>externalPublication.php</td>
</tr>
<tr>
<td></td>
<td></td>
<td>method.php</td>
</tr>
<tr>
<td></td>
<td></td>
<td>person.php</td>
</tr>
<tr>
<td></td>
<td></td>
<td>_include</td>
</tr>
<tr>
<td></td>
<td></td>
<td>core.php</td>
</tr>
<tr>
<td></td>
<td></td>
<td>header.php</td>
</tr>
</tbody>
</table>
```

‘method’ and ‘external publication’ fields, required the utilization of Java Script hence this project utilized jQuery and the jQuery UI libraries. These were very instrumental in creating and managing the dissertations. Figure 7 below shows “External Publication Listing” which utilizes jQuery UI libraries.

Through the use of jQuery, the event listeners find themselves bound by the conditions used to construct the fields since they can only enter characters that meet the author’s established ‘Unique ID’. Using jQuery UI application “Auto complete”, these are well displayed at the bottom of the text box. Whenever a value is picked, the text box is filled out. Figure 8 below shows how “Auto complete” feature works during data entry.

The real look-up of this effort is then done when you run a PHP script and it executes My SQL query that in turn searches the database. Using the field type as its parameter, the utilized PHP script runs a query that is just unique to that field and only returns values that match

- footer.php
  - assets
    - admin.css
      - dissertation
        - create.php
        - delete.php
        - form.php
      - manage.php
        - external_publication
          - create.php
          - delete.php
          - form.php
        - manage.php
          - person
          - create.php
          - delete.php
          - form.php
          - manage.php
        - home.php
        - index.php

Script structuring

Footer and header scripts are necessary for every view with the main purpose of making them work properly. This should be done at the beginning and the end respectively in the file. “The scripts encapsulate the page in the HTML template along with load all of the application functionality included in (include/main.php). It is important to use this methodology in order to centralize the location of the HTML template (including it in only one header/footer file) and to ensure that the same core application logic permeates through all views”..

Core Script Structure

```
//begin session session_start();
//global variable setup
```
$db_host = 'localhost';
$db_user = 'cs691-team03-2012';
$db_pass = 'xxxxxxxxxxxx';
$db_name = 'cs691-team03-2012_development';
$url_root = '/~cs691-team03-2012/admin';

$site_root = '/home/cs691-team03-2012/public_html/admin';

//include all application scripts
include $site_root.'/_controllers/authentication.php';
include $site_root.'/_controllers/database.php';
include $site_root.'/_controllers/output.php';
include $site_root.'/_objects/_object_base.php';
include $site_root.'/_objects/dissertation.php';
include $site_root.'/_objects/externalPublication.php';
include $site_root.'/_objects/person.php';
include $site_root.'/_objects/method.php';
include $site_root.'/_objects/category.php';

$obj_auth = new authentication();
$obj_auth->check();

//load data objects
$dissertation = new dissertation();
$externalPublication = new externalPublication();
$person = new person();
$method = new method();
$category = new category();

Object classes

The abstraction level required between core application and database interaction is brought well using the object classes. The object ‘base class’ forms the foundation of functionality for all objects.

- object_base
  - var fields
  - var idfield
  - var lookupField
  - var tableName
  - function construct()
  - function create()
  - function modify()
  - function delete()
  - function lookup()
  - function autocomplete()
  - function get()
  - function listing()
  - function getName()

//load and setup core functionality
$db = new database();

//connect and select database
$db->connect($db_host,$db_user,$db_pass);
$db->setDB($db_name);

//check for authentication status
function manageCommittee()
{
  method
  function construct()
    person
      function construct()

Object constructor setup

Each object class’ constructor for each object class is used so as to provide the base object class with the data it needs to influence the correct tables

function                   constructor()
{
  category
    dissertation
      function construct()
    function retrieveCommittee()
    function manageCommittee()
    externalPublication
      function construct()
      function retrieveCommittee()
    $this->fields =
      array('id', 'title', 'author_id', 'other_citing_information', 'dissertation_id');
    $this->idField = 'id';
    $this->lookupField = 'title';
    $this->tableName = 'external_publication';
}

Object class extensions

Normally, the base ‘create’, ‘modify’, ‘delete’, ‘get’ methods require extension and hence necessitating the need for an advanced functionality.

Through the controller classes, collections needed for core functionality applications are availed. This serves to unify the code bases and also allows for the ready extension of the applications.

- authentication - All authentication functionality
  - check()
Front end implementation

It is always expected that HTML provides an illustrative layout, it was therefore critical that coding be done to enable administration views. A total of seven pages were created comprising of one log in page, 3 view pages with “person, dissertation, and external publication”. Html was used in creating forms in regards to coding of the edit pages. PHP was used to process the data entered in the various fields available and then stored in the database in the form of My SQL.

All the information gathered from the original database were inserted in the ‘easy to read table’ using HTML. This application was also used not only for data entry for the languages but also for purposes of displaying the content to the users.

In creating a visually appealing page that would excite the users, the CSS code was utilized and the results were pleasing to the users. The use of CSS was chosen because it has the merit of allowing the website designer to format the web pages and organize them as he/she deems fit. In this project, CSS therefore helped in the creation of all the link areas, the forms, tables as well as the colors.

After establishing the field requirements for each web page, HTML was utilized for purposes of generating the ‘edit’ views. For instance, in the page “Create person”, only two fields were provided for but in the page “Create external publication”, seven fields were created.

Just like the ‘listing views’, the whole process of development was just similar. It was however necessary that the specific titles and the number of rows that each table required had to be defined prior to creating the remaining pages. Figure 6 below shows a sample “Dissertation Listing” page.

![Figure 6. Dissertation Listing](image)

Implementing the auto-completion, with the titles ‘author’, ‘committee member’, ‘advisor’, ‘category’, ‘method’ and ‘external publication’ fields, required the utilization of Java Script hence this project utilized jQuery and the jQuery UI libraries. These were very instrumental in creating and managing the dissertations. Figure 7 below shows “External Publication Listing” which uses jQuery UI libraries.

![Figure 7. External Publication Listing](image)

Through the use of jQuery, the event listeners find themselves bound by the conditions used to construct the fields since they can only enter characters that meet the author’s established ‘Unique ID’. Using jQuery UI application “Auto complete”, these are well displayed at the bottom of the text box. Whenever a value is picked,
the text box is filled out. Figure 8 shows how “Auto complete” feature works during data entry.

Figure 8. “Auto complete”

The real look-up of this project is then done when you run a PHP script and it executes My SQL query that in turn searches the database. Using the field type as its parameter, the utilized PHP script runs a query that is just unique to that field and only returns values that match the author’s preset ID. Through this, irrelevant values are not returned as answers to the run query.

3. RESULTS

The system basically consists of two sets of pages. The first set includes all the pages that display information from the dissertation database either as a table or as a report. The second set of pages provides functionality to add and edit data. Data is validate at each data entry page to ensure that the data entered is valid and confirms to the specification.

4. FUTURE ENHANCEMENTS

There are a number of ways that this project can be extended to increase the breadth and utility of the data it manages. An important first step is “information portability” or more clearly the ability to organize and extract the data in various formats. This would allow for the information to be quickly imported into other systems or a generic table structure for data analysis. By creating an API (application programming interface) information could not only be readily extracted in a machine readable format (such as XML or JSON), the same API can also allow for other software to directly add and modify information. Another possible solution is to provide a simple CSV (comma separated value) or Excel formatted export. It would not allow for interaction that is nearly as robust, but would be sufficient for a basic method of extracting data.

The basic premise of this software is to manage information about documents and their relationship to various persons. Specifically we are currently concerned with dissertations, published documents, authors, advisors, and committee members. This structure could be further generalized to allow it to contain relationships between numerous types of content and people, along with the associated metadata (publication date, years to completion, etc.). If this software was standardized it could be used to track information about all of the documents (papers, reports, dissertations, etc.) a student produces in the course of his graduate and undergraduate academic career. For archiving purposes, it would also not be difficult to include the actual paper itself in data storage, making it quickly available to that student, administrator or appropriate peers.

5. CONCLUSIONS

The web interface created as part of this project is a simple, but powerful tool to efficiently enter and maintain data in the growing dissertation database. Compared to the existing system, the new user interface with the underlying data structure sets new high standards for the data integrity. Tests around the database showed that the data is stored in the tables with 99% accuracy.

In summary:
- The study involved some changes to the original database design.
- Database entities were rearranged to better represent real world relationships between dissertations, external publications and the people involved.
- The new structure is flexible enough to handle future expansions.
- Validations provided by the user interface ensure error free data entry.