

Obviously, few departments will seek to reorganize themselves along interdisciplinary lines. Establishing interdisciplinary institutes or fellowships at interdisciplinary institutes is more likely to be feasible, but would still be a formidable task. Thus, political science departments seeking to move their methods training in an interdisciplinary direction would most likely either adopt the certification approach, send students to other departments to complete a political science methods requirement, or establish a regular “pipeline” for sending students to summer methods programs.

For departments that decide to set up a certification program for interdisciplinary methods training, begin by contacting those departments that offer classes that might be useful to political science students and proposing an interdisciplinary unit. Other social science departments (such as anthropology, communication, economics, education, geography, and sociology), the statistics department, and even the mathematics department are good candidates. Devising a general curriculum that is acceptable to all participating departments (such as 4 additional methods classes) is the next step. There is also likely to be some administrative work in getting the certification program approved above the departmental level. Establishing a clear framework for the responsibilities of each participating academic unit at this time is important. Issues such as resources and course relief for administrative work related to the program should be clearly worked out early on to avoid problems and misunderstandings down the road.

Sending students to summer methods programs on a regular basis is a comparatively low cost way to pursue an interdisciplinary approach to methods training. However, even at this relatively low level of commitment to interdisciplinary methods training there might be policies a department can establish to most effectively integrate these summer programs into a larger methods program. For instance, should departments send any students who are interested to a summer methods program, only the methods students, or only those that are deemed the best in some kind of internal competition? Should students attend at the end of the departments methods sequence, or only once the student has a clear dissertation topic? As with implementing any of the other approaches to interdisciplinary methods training outlined here, a clear accounting of goals, costs, and benefits is vital.

References

- Beck, Neal. 2000. ““Political Methodology” A Welcoming Discipline.” *Journal of the American Statistical Association* 95(450):651-654.
- McKenzie, Kenneth. 2007. “Comment on Moran’s ‘Interdisciplinarity and Political Science’.” *Politics* 27(2):119-122.
- Moran, Michael. 2006. “Interdisciplinarity and Political Science.” *Politics* 26(2):73-83.

Computing and Software

Web-Based Data Collection with PHP and MySQL

Stephen R. Haptonstahl

Washington University in St. Louis
srhapton@wustl.edu

Introduction

Computer-based forms can greatly increase the efficiency and accuracy of data collection. Collecting the data in digital form makes it easier to import it to statistical software and facilitates easy backups. Many researchers reach first for a spreadsheet like MS Excel because, as a free form tool, it is easy to get started without any programming skills and no Internet connection is needed. However, Web-based forms provide some clear advantages: more than one person can enter data at a time without fear of writing over each other’s work; the data is stored on a server where it

is (almost certainly) safer; the researcher can impose more structure on the data entered using pull-down menus, check boxes, and radio buttons. Two caveats: an Internet connection and some technical skills are required. However, if those entering the data will have Internet access and if you can create and upload a simple Web page, then this article fills in the rest, providing the tools you need to set up your own Web-based data collection form.

Here’s how it works. Talk to the administrator of your Web server to get a MySQL account and make sure phpMyAdmin is installed. Using the code book you wrote

for your data set, modify the template downloaded from my Web site (<http://haptonstahl.org/srh/>) by adding boxes called **controls** in the template for each piece of data you want to collect; this Web page you create is called the **front end** of the form. Upload this file and two other files to a folder on the Web server. Within your Web browser use phpMyAdmin to set up the database with fields corresponding to the controls in the Web form; this database is the **back end** of the form. Once set up, test everything. Those entering the data use any Internet-connected computers to enter the data simultaneously, which is then stored safely in a database on the server. Once the data is entered, use phpMyAdmin to view the data and export for use in your statistical software.

There are some skill and system prerequisites for this project. You should be comfortable editing a simple Web page using a text editor like Notepad or WinEdt, uploading it to a server, and viewing the page using your Web browser. If not, you can acquire an overview of the process for creating a Web page from <http://tinyurl.com/e5uwo> and learn basic HTML in a few minutes from <http://tinyurl.com/h922b>. The server you use must have PHP, MySQL, and phpMyAdmin installed. If you want to know what each of these programs do, I recommend starting with the Wikipedia entries on each. These are free and probably are already on your Web server, but the server administrator will know for sure. To use MySQL, you need another user name and password, possibly different from the ones you use to upload files to the server, and the name of the database assigned by the administrator. Save this information for later; a useful form for this is available at <http://tinyurl.com/m4cf4>.

Setting Up the Front End

The front end is the Web page itself, and all you have to do is modify a template I have used with other projects. Create a *working folder* where you will work on the site; this can be a folder on your local machine, in which case you upload the files to the Web server after each edition, or you can work directly on the server. Download the zip file containing the template files (<http://tinyurl.com/egvvt>) and extract the contents to this working folder. Two files, `add_record.php` and `show_records.php`, contain PHP code that you may never need to modify; leave those alone. The file you will edit for your project is `index.php`, but I will start by dissecting a simpler file, `simple-index.php`.

`simple-index.php`

```
<html>
<head>
  <title>Project Title Here - Data Entry Form</title>
<?php
  $server_host =      'localhost';
  $mysql_user_id =   'mylogin';
```

```
  $mysql_password =  'mypassword';
  $mysql_database =  'mydatabase';
  $mysql_table =     'mytable';
  $mysql_primary_key = 'id';
  include('add_record.php');
  ?>
</head>
<body>
<center>
<h2>Project Title Here</h2>
<h3>Data Entry Form</h3>
<form action="index.php"
  method='post'>
  <input type='reset'
    value='Reset the form (does not upload info)' />
  <table border cellpadding='2' align='center'>

    <tr>
      <td><b>#</b></td>
      <td><b>Variable</b></td>
      <td><b>Value</b></td>
      <td><b>Description</b></td>
      <td><b>Coding Notes</b></td>
    </tr>

    <tr>
      <td>1</td>
      <td>Test Text</td>
      <td><input type='text' name='testText'
        size='40' maxlength='255' /></td>
      <td>Plain text field</td>
      <td>Details about what to enter go here.</td>
    </tr>

  </table>
  <input type='hidden' name='action' value='add' />
  <input type='submit'
    value='Click to upload this record' />
</form>
</center>
<?php
  $number_of_records_to_show = 5;
  include('show_records.php');
  ?>
</body>
</html>
```

The two sections marked “<?php ... ?>” are PHP code, not HTML, so be careful in modifying these sections. Note that each line of PHP ends in a semicolon. The first PHP section at the top is where you put the MySQL user name, password, and database given to you by the server administrator. You can choose the name of the table, and like any “variable” the table name should contain no spaces or punctuation. Note that a table in MySQL corresponds to a worksheet in MS Excel, so you will need exactly one table for all of the data entered using a single Web-form. I like to name the table something like a one-word version of the project title. In that block you should only have to set those four values.

At the bottom of the file is a small PHP block where you can specify how many of the previously entered records will be displayed at the bottom of the screen. You can set

this to any integer 0 or higher, or to 'all'. It is useful for those doing data entry to be able to see records recently entered. Even though they will not be able to modify those records, if they see an error they can reenter the corrected record with an note to you that the erroneous record should be removed.

Between the two PHP sections is an HTML form. This is the section that you modify to set up the front end interface for the user, which appears in Figure 1.

Project Title Here

Data Entry Form

Reset the form (this does not upload typed info)

#	Variable	Value	Description	Coding Notes
1	Test Text	<input type="text"/>	Plain text field	Details about what to enter can go here.

Click when complete to upload this record

Figure 1: Simple version of the “Front End” Form

Each block of `<tr> ... </tr>` corresponds to a row in the displayed HTML table. Rows in the HTML table (front end) roughly correspond to columns (fields) in the database table (back end) but this is only approximately true; don't confuse the two kinds of tables. There are two HTML rows in the table above. The first is the header row. The second defines one control, a text box, where a piece (field) of data “Test Text” can be entered. If you want a box where up to 255 characters (letters or numbers) can be entered, copy this block. There are other controls (radio buttons, check boxes, text areas) I will discuss later.

Within each row, there are five columns delimited by `<td> ... </td>`. The first is the item number specified in your data code book (because you numbered them, right?) The second is the short title of the field. The third is a bit of HTML “form” code (cf. HTML forms) creating a control in which users can enter data; for text boxes just copy this block but rename the “name”, which in this example is 'testText'. **Each field in the database must have a distinct name that matches exactly the name of the control in the HTML form.** The fourth and fifth columns you can use as you like, however on most computers the fourth column will be completely visible but reading the fifth column may require the user to scroll right, so it is a less convenient reference.

There are three other kinds of controls that are useful instead of simple text boxes. The following sample comes from the template file `index.php` and produces the controls that appear in Figure 2.

Section from `index.php`

```
<tr>
  <td>3</td>
  <td>Test Check Boxes (check all that apply)</td>
  <td>
    <input type='checkbox' name='testCheckbox1' />
    Check this one <br/>
    <input type='checkbox' name='testCheckbox2' />
    You can check this one, too <br/>
  </td>
  <td>Check boxes</td>
  <td>Each check box is a separate field that is empty
  if not checked and set to 'value' if checked. If you
  do not specify a value, it simply gets the value 'on'
  when checked.</td>
</tr>

<tr>
  <td>4</td>
  <td>Test Radio 2<br/>(choose one)</td>
  <td>
    <input type='radio' name='testRadio2' value='1' />
    I can spell R <br/>
    <input type='radio' name='testRadio2' value='2' />
    I use R <br/>
    <input type='radio' name='testRadio2' value='3' />
    People ask me how to use R <br/>
    <input type='radio' name='testRadio2' value='88' />
    I write R packages (which ones?)
    <input type='text' name='testRadio2_notes' size='15'
    maxlength='255' /><br/>
  </td>
  <td>Radio buttons</td>
  <td>This combines the radio buttons with a text box.
  Note that the text box has a different name and gets
  stored in a different column of the database.</td>
</tr>

<tr valign='top'>
  <td>5</td>
  <td>Test Text Area</td>
  <td>
    <textarea name='testTextArea' rows='15'
    cols='45'>Default text here (leave blank if you
    like)</textarea><br/>
  </td>
  <td>Text area</td>
  <td>Type lots if you like. In phpMyAdmin I created a
  "TEXT" field for this, which can be very, very long.</td>
</tr>
```

The little square boxes above are **check boxes**. Each is a separate piece bit of data with its own name, `testCheckbox1` and `testCheckbox2`; more than one may be checked. The little circles are **radio buttons**. All of those in a group have the same name, `testRadio2`; only one at a time can be checked (just like buttons on old-style car radios.) The large box at the bottom is a **text area** which is suitable for entering text more than 255 characters.

Radio buttons are good for categorical data as long as the categories are distinct. Check boxes are good for collecting “dummy variable” type data as long as there is no missing data; if missing data is a possibility, use a radio

3	Test Check Boxes (check all that apply)	<input type="checkbox"/> Check this one <input type="checkbox"/> You can check this one, too	Check boxes	Each check box is a separate field that is empty if not checked and set to 'value' if checked. If you do not specify a value, it simply gets the value 'on' when checked.
4	Test Radio 2 (choose one)	<input type="radio"/> I can spell R <input type="radio"/> I use R <input type="radio"/> People ask me how to use R <input type="radio"/> I write R packages (which ones?) <input type="text"/>	Radio buttons	This combines the radio buttons with a text box. Note that the text box has a different name and gets stored in a different column of the database.
5	Test Text Area	Default text here (leave blank if you like)	Text area	Type lots if you like. In phpMyAdmin I created a "TEXT" field for this, which can be very, very long.

Figure 2: Check Box, Radio Button, and Text Area Controls

button with “missing” as one of three choices. I recommend including at least one text area at the end of the form for any comments the coder may have about the data. It is also a good idea to have the coder enter their name somewhere on the form if there is more than one person entering data.

You now have examples of four different controls. To create your own, find the right type of control in the template and copy everything in the `<td> . . . </td>` block surrounding it. Then customize it by setting the item number, control name, and comments. For radio buttons, you can change the values associated with each choice by setting the `value` and you can have more or fewer choices by copying or deleting a `<input type='radio' . . . />` line. Radio buttons are “grouped” simply by giving them the same name; if they have the same name, only one of them can be selected at a time.

You will have to choose names for each of the controls, and these control names must agree *exactly* with the fields you create in the database. Start by reading Jonathan Nagler’s (1995) article on “Coding Style and Good Computing Practices.” Then go through your code book and indicate what kind of control you want to use (text box, radio button, check box, or text area) and name each control (or group of radio buttons.) Keep this list handy for setting up the database.

The template is fairly simple, so you might decide to spend some time customizing it. I recommend some caution here. If you make the page “prettier” you may inadvertently make it harder to use. If you add features like validation using JavaScript or other “client-side” code you may change the form so that it does not run on all browsers. You will have to make a decision on these tradeoffs; if you make such changes, make sure you test the form on every browser (Internet Explorer, Firefox, Safari, Opera, etc.) you can before you start data collection.

Setting Up the Back End

Now that you have the front end set up to collect the data, you need a back end database to store the data. The front end uses PHP to send data to the database using `add_record.php`, but to set things up you will use a

free Web-based application called phpMyAdmin. MySQL is the database program you will use, but it only has a command line interface built in. phpMyAdmin provides an easier graphic user interface. You do not install phpMyAdmin on your machine; your administrator installs it (if it is not there already) on the Web and MySQL server. You use it through your Web browser by going to a Web address provided by your server administrator. Log in using the user name and password specifically provided for MySQL – the same ones you put in the top PHP block of the Web form.

Once logged in you will create the table that will hold the data. Click on “Databases” and then on the name of the database your administrator provided. You should see a prompt “Create new table on database” (Figure 3). Enter the name of the table that you chose earlier when setting up the front end. For the number of fields, enter one more than the number of field names you chose when going through your code book. For example, the template `index.php` needs 8 distinct data fields, so the table will need 9 fields.

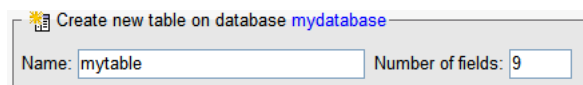


Figure 3: Creating a Table using phpMyAdmin

The next screen will allow you to create the database fields (Figure 4). There are a lot of options here, but you only need three kinds of fields to collect all of your data. First, add a field named ‘id’ (without quotes). This field will be a unique id for each record, even if all of the data fields are repeated across records. Set the type to ‘INT’, the ‘Extra’ to ‘auto_increment’ and click on the first radio button (under the key) to designate this field as the primary key of the table.

Now enter the names of all of your data fields down the first column. If you may be typing a paragraph for a field and you chose a ‘text area’ control then set the type to ‘TEXT’ which can handle about 8 full typed pages of text. For all other controls (text box, radio button, check box) leave the type set to ‘VARCHAR’ and set the length

Field	Type	Length/Values	Attributes	Null	Default	Extra					
id	INT			not null		auto_increment	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
testText	VARCHAR	255		not null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
testRadio1	VARCHAR	255		not null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
testCheckbox1	VARCHAR	255		not null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
testCheckbox2	VARCHAR	255		not null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
testCheckbox3	VARCHAR	255		not null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
testRadio2	VARCHAR	255		not null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
testRadio2_note	VARCHAR	255		not null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
testTextArea	TEXT			not null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Figure 4: Creating Fields in a Table using phpMyAdmin

to the maximum of 255 characters. This does mean that you are limited to 255 characters in text boxes, but if you are typing anywhere near that, you should probably be using a text area control and ‘TEXT’ field type. To create the table corresponding to the template file `index.php` the screen should look like Figure 4. Check the spelling of all of

the field names, including capitalization. Not matching exactly the field names between the form and the database is the most common error; this will cause data entered in that control to ‘just disappear’ and not appear in corresponding field of the database, or possibly appear in the wrong field. Click ‘Save’ to create the table.

If you have a lot of fields (10+) you might choose to enter ten or so at a time. To do this, enter ‘10’ for the number of fields when creating the database and enter the ‘id’ and first nine data fields. To add fields, click on the table name on the left side of the screen, which will show the structure of the table, then enter the number of fields you want to add and click ‘Go’ (Figure 5).

When you are finished testing, clear the data from the database. You can’t do this from the Web form, and neither can anyone else – it’s not a bug, it’s a feature. To clear the table of all data, log into phpMyAdmin, browse to the table structure, and look for the ‘Empty’ tab (Figure 6). Click it.

field(s)
 At End of Table
 At Beginning of Table
 After

Figure 5: Adding Fields to a Table

Figure 6: ‘Empty’ is Good; ‘Drop’ is Bad

Testing

After you create the front end page and back end database, upload the files all to a single folder on the Web server and view the Web page. With luck, everything looks great, but this is unlikely. Check everything thoroughly, from the spelling of page title to the positioning of each control. Try to add some data. Do you get an error when uploading the data? Does the data appear down at the bottom of the entry form? If you set `$number_of_records_to_show = 0`, you may want to set it to 10 while testing the form to make it easy to see whether the database receives correctly the data you enter.

Important: Do not click the ‘Drop’ tab; that will delete the table completely and you will have to reenter all of the fields. If you changed the `$number_of_records_to_show` setting for testing, now is the time to change it back.

A great way to make sure that a data entry screen is well-designed is for the designer of the form to use it. Go ahead and do some of the real data entry yourself. I recommend entering at least 10-20 records. This will catch more problems and help you anticipate how best to train the people who will be doing the rest of the data entry.

What Goes In Must Come Out

You may want to inspect the data while it is being entered. Log into phpMyAdmin, click on the table name, then click on the ‘Browse’ tab to view the data. If you need to make corrections in the data here you can scroll to the record you want to correct and click the pencil icon in the same row. You can also delete records (by clicking the X on

One easy method for testing the connection between the form and the database is to enter one record with data only in the first field, then another record with data only in the second field, etc. Remember to check each radio button and each check box separately. Checking radio buttons in order makes it easy to verify that you didn’t forget to change the ‘value’ from one button to the next.

the row) and you can delete all of the data or the entire table. The software will ask you for verification before deleting anything, but be careful—once something is deleted, there is no ‘undo’.

When the data has all been entered, it is easy to export the data to a more usable format. Log into phpMyAdmin and click on the table name to view the structure of the table. Click on the ‘Export’ tab. Here you will have several options for the format. To back up your data in a format that you could read back into MySQL, use the default ‘SQL’ settings. Other file formats include \LaTeX , Excel, Word, comma-separated versions (CSV), and XML.

To get your data into R choose the “CSV for MS Excel” format. Check the boxes to “Put field names in the first row” and “Save as file”, then click ‘Go’. The resulting downloaded file can be read into R using the default settings for `read.delim()`. One way to get your data to STATA is to save it from R using `write.dta(foreign)`.

Final Thoughts

How safe is this? Most likely, your administrator performs nightly backups of all MySQL databases on the server, so your data is reasonably safe from hardware failure. Another potential source of problems stems from the fact that the Web form is on the Web: theoretically, anyone can get to it. However, if nobody creates an explicit link to the form, a would-be saboteur would have to know the address to get to the page, and even then all they could do is enter extra records. As long as you have a field for the name of the person entering data, you would be able to filter out bogus data. They could possibly fill the server with data, although, unless automated, this would take a *very* long time. They could only see the last few records (however many are displayed at the bottom of the form) but they could not delete records. All of this is very unlikely, but possible. If this “security through obscurity” is not sufficient for your project, you can arrange with your server administrator to “secure” the entire folder containing the Web form so that a password is required for any access.

Collecting data via the Web takes a little more time to set up initially, but the advantages are significant. One recent project using this method used a long form to collect 180 fields of data for 1100 court cases using half a dozen coders, but this would be useful for smaller or larger projects. Go to <http://tinyurl.com/kyavr> to try both of the forms discussed in this article, download the files, and find other useful links.

References

- Nagler, Jonathan. 1995. “Coding Style and Good Computing Practices.” *The Political Methodologist* 6(2):2-8.