THE UNIVERSITY OF AKRON
Theoretical and Applied Mathematics

The Methodology used for Summarizing the TEX/LATEX Usage Survey

D. P. Story

1. Introduction

On January 31, 2005, I published a **TEX/LATEX Online Usage Survey** at

`http://www.math.uakron.edu/~dpstory/eqExam/tex_survey.pdf`[1]

as a demo file to my LATEX package eqExam, and as a service to the TEX community. The document `tex_survey_rep.pdf` has a tabulation of the multiple choice questions and `longresp.tex.pdf` lists the responses to the short and extended response questions. This short article describes the methodology used to create these two documents. The method is a desktop solution that requires a Windows platform with Acrobat 7.0 Professional.

The source files `tex_survey_rep.tex` and `longresp.tex.tex` are also available. The interested reader should feel free to download them and to study the LATEX and JavaScript code.[2] The reader may someday use eqExam to create his own survey (or questionnaire), the code and the methodology can come in handy at that time.

2. Methodology

The survey was constructed using the eqExam package with the `email` option. The way eqExam is designed to work with this option is that the server-side script, `eqAttach.asp`,[3] receives the form data from the client, creates an email message, attaches the data as a FDF file, and sends the message to the designated recipient. As the designated recipient of the survey, I've received over 200 emails to date.

Gathering the Data. The attachments were saved to a folder on my desktop as they arrived for later processing.[4]

How do we take all these FDF files, extract the form data, and save the data to a database? There is a new feature of Adobe Acrobat 7.0 Professional that I used to handle this problem. Under the `File > Form Data` menu, there is an item

[1]The survey is ongoing, so feel free to respond.

[2]The JavaScript is a good example of how to access a database, extract data, make calculations, and populate a PDF document. It can be modified and used in applications.

[3]This script is part of the distribution of eqExam.

[4]I've programmed my mail client, Pegasus Mail, to automatically detect these emails (by scanning the subject), saving the FDF attachment the designated folder, then moving the email to a special email folder, reserved for the survey results. This requires no daily intervention on my part.

in the drop-down menu titled "Create Spreadsheet from Data File…". I selected this item, in the subsequent dialog, I chose all the FDF files in the folder were I kept the survey results, and Acrobat did the rest. Upon completion, Acrobat produced a `.csv` file, a comma delimited file that Microsoft Excel can read. I opened the new file in Excel and saved it as a `.xls` file.

All of the field names of the survey contain "dots"; for example, the radio button fields of question 1(a) have a name `eqexam.Part1.1.parta`. Ultimately, I wanted to move the data to Microsoft Access where field names containing "dots" are not legal names. In Excel, I changed the column headings, from, for example, `eqexam.Part1.1.parta` to `1_parta`. I then opened Microsoft Access and imported the Excel spreadsheet, and saved as `report.mdb`.

Extracting and Reporting the Data. The next problem was extracting the data in some way. For a number of years, I've been fascinated by the interplay between LaTeX, the form elements of a PDF document, and Acrobat JavaScript, so I thought I'd do something innovative. `:-)`

First, I made a copy of the source file, `tex_survey.tex`, to form the base document to report the results of the survey. This file, `tex_survey_rep.tex`, and its PDF counterpart accompany this article.

By redefining some of the commands from eqExam and the AcroTeX Bundle, the radio button fields were changed to text fields.

Within the source document, `tex_survey_rep.tex`, I wrote some JavaScript to connect to the Access database, `report.mdb`, that contains the form data. When the document `tex_survey_rep.tex` is first opened in Acrobat 7.0 Pro, the JavaScript retrieves the data using Acrobat's ADBC plug-in.[5] The JavaScript compiles a count of each response to every multiple choice question, and reports the count in the newly created text fields. A verbatim listing of this JavaScript follows:

```
% Get a connection object to the MS Access DB through ODBC
connect = ADBC.newConnection("texsurvey");
% Get a statement object
statement = connect.newStatement();
%
% Create a function to perform the repeated task of extracting
% data from the database and populating the appropriate fields
```

[5] Acrobat Database Connectivity, available on the Windows version of Acrobat since version 5.0.

```
function getMultiPart( field )
{
    var f = this.getField(field);
    var n = f.getArray().length;
    var row, result, nChoice;
    var alpha = "abcdefghijklmnopqrstuvwxyz";
%
% This array will hold the counts of each of the choices.
% We initialize the count to zero.
%
    var aCnt = new Array(n);
    for ( var i=0; i<n; i++) aCnt[i]=0;
%
% Here, we execute a SQL statement to get the data we want.
%
    statement.execute("SELECT \""+field+"\" FROM \"report\"")
    try {
        while (true) {
%
% Loop through the data, row by row. An exception is thrown
% when there are no more rows.
%
            statement.nextRow();
            row = statement.getRow();
% get the value for the current row, of the requested field
            result = row[field].value;
            if ( result == "Off" ) aCnt[n-1]++;
            else {
                nChoice=result.charCodeAt(1)-alpha.charCodeAt(0);
                aCnt[nChoice]++;
            }
        }
    } catch(e) {console.println("Done with " + field);}
% we are done extracting data and computing the counts of each
% response, now let’s populate the corresponding field in
% tex_survey_rep.pdf.
    for ( var i=0; i < n-1; i++)
        this.getField(field+"."+alpha[i]).value = aCnt[i];
%
% I added an extra field to hold the count for the number of
% times the respondent did not respond to this question.
%
    this.getField(field +".Off").value = aCnt[n-1];
}
% The array of the (Access) field names of the multiple
% choice questions.
var aMC = new Array("1_parta","1_partb","2","3","4_parta",
    "4_partc", "5_parta","5_partb","6_parta","6_partb",
    "6_partc","7_parta","7_partb","8_parta","8_partb",
    "10","12", "14","16","17","18","19"
);
% Finally, loop through all multiple choice field names,
% getting the data and populating the text fields of
% tex_survey_rep.pdf.
for ( var i=0; i < aMC.length; i++) getMultiPart(aMC[i]);
```

In addition to the multiple choice questions, there are a number of “fill-in” questions. I wrote JavaScript to extract these responses from the database as well and to write a listing of the responses to a LaTeX file, `longresp.tex`, and to attach this newly created LaTeX file to `tex_survey_rep.pdf`. This is a new feature of Acrobat 7.0 Professional, the ability to programmatically create a file and attach it to a PDF document. The details of this code can be found in the source file `tex_survey_rep.tex`.

Building the Report Document. The final step of this whole process is to `latex` the file `tex_survey_rep.tex`, and convert the `.dvi` output to PostScript. Distilling the postscript file produces the PDF document. When the document is opened for the first time in Acrobat 7.0 Professional, that’s when the JavaScript just described leaps into action. It extracts the data from the database, populates the text fields of `tex_survey_rep.pdf` with the frequency counts, creates the LaTeX document listing, `longresp.tex`, of the extended responses and attaches it to `tex_survey_rep.pdf`.

The only thing left to do is to save the attachment, `longresp.tex`, to a folder, open it, `latex` it,[6] and convert it to a PDF. I’ve made the two documents `longresp.tex` and `longresp.pdf` available for download.

In Conclusion. The system developed can be repeated as new responses to the survey arrive. I can easily rebuild all the documents and publish the latest compiled statistics, the hard part was doing it the first time.

Hope you’ve found this discussion worthwhile, and now I simply must get back to work. 𝔇𝔓

[6]Not quite as easy as that, I had to go through this LaTeX file and fix some of the responses of the respondents so that file compiled correctly. The respondents were not asked to enter LaTeX code.