Election Administration Component - Catherine

Description

The election administration component is the set of PHP scripts that generate the online election that voters participate in. The user sees three pages: a welcome page, a page with the election questions and a page showing whether or not their vote was submitted properly. In addition, the user might end up seeing an error page if something goes wrong (problems with the database, server etc...) or if they try and vote multiple times. If there is no current election in progress, the user is shown a page consisting of that information. The largest concern for this part of the project is security - the site must be secure so that the election cannot be tampered with.

Design

The design for the component can be broken down into multiple considerations/pieces:

1. Security
2. Web Pages
3. Question Generation
4. Question Verification
5. Question Submission
6. How This Is Setup In PHP

Security

We are using Brown's WebAuth security system in order to guarentee that the site is secure. The system works by using .htaccess files (files which are used by the server to set various directory dependent variables) in each directory. The .htaccess file can be used to specify what groups of students have access to the material in the directory (all undergraduates, freshman, sophomores, juniors, seniors etc...).

There are two kinds of elections: general and year-specific. General elections are ones in which all undergraduates can vote on all questions and year-specific elections are ones in which some questions are not offered to all years.

When an election is general, the user sees a welcome page with a link entitled "Go Vote!". This link leads them to a script in an election-general folder which gets the questions of the election from the XML component and then generates the HTML for each question. The .htaccess file in this folder is written such that all Brown undergraduates have access to the page.

When an election consists of different questions for different years, the welcome page of the election will ask the user to click on a button with their year on it. They will then be taken to Brown's WebAuth login page (if they are not logged in) and once they are logged in they will be allowed to view the election page only if they are the year they selected.

We cannot check what year a student is in directly, we can only grant them access to a directory if they are a given year. For each election question, there is a list of what years can and cannot vote. Therefore, there are 4 almost identical copies of election generation scripts in 4 directories on the server: election-freshman, election-sophomore, election-junior and election-senior. In each of these directories is a different .htaccess file which allows only the corresponding year access to the election-generation script inside. Each of these scripts works by getting an array of all of the election questions from the XML component. For each question the script checks if the given year can vote, and if so then the script generates the HTML for the question. This is done with a function in a Question.php script in a utility directory in the election directory on the server.

While having five copies of a script does not seem like a good software engineering practice, it is neccessary to guarentee that for year-specific elections each group of students only sees the questions appropriate for their year. This is due to the setup of the security system we are using which is a neccessary external dependency since we are using CIS's server and none of the group are security experts.

Web Pages

The web pages are all HTML with at least a small amount of PHP embedded in them so that information such as the election title can be obtained from the XML data file.

There are three main pieces to the web site the user sees during a normal election.

- **Welcome Page** This page contains a welcome message obtained from the XML file and, depending on the type of election, has the user either follow a "Go Vote!" (general election) or click on a button corresponding to their class year (year-specific election) to vote.

- **Voting Page** This is the page containing all of the election questions that the user is allowed to vote on. All of the questions are displayed as some type of form input, and the entire form is submitted when the user clicks "Vote!" at the bottom of the page. For more detail on the types of questions and their HTML representations, see the Question Generation section.

- **Vote Submission Success Page** This is the page that users see after they click "Vote!". The script for this page both verifies that the answers provided to all of the questions are valid (people only chose 4 of 7 choices etc...) and then submits the answers to the database. The user sees one of several things here:

  - **Success** The questions were all answered properly and the vote was successfully recorded in the database
  - **Multiple Votes Error** The user has already voted, and thus there vote will not be counted again.
  - **Invalid Responses** Some of the answers the user provided were not valid. The user will then be prompted to go back, fix their errors, and resubmit their ballot. For more information, see the Question Verification section.
  - **Other Error** There was a problem with the database and the vote could not be entered at this time. The user will be asked to attempt to resubmit at a later time. This should NOT happen unless the server itself is having problems, which is beyond our control to fix.

Question Generation

The generation for each question consists of generating the question text with a link for more information (optional) followed by a table of potential choices and some way of marking your preference for a given choice. Each choice is displayed with the following three pieces:

- **Picture** The picture of the candidate, if one is provided - optional.
- **Name** The text displayed for a candidate - needed.
- **More Information Link** A link to more information about the candidate, either a personal html page provided when the ballot is created, or an outside webpage - optional.

The detail for each question type is as follows:

- **CHOOSEONE** After the question text is displayed, a table is generated where each entry in the table is the information about one candidate followed by a radio button to select that candidate.
- **CHOOSEX** Identical to the CHOOSEONE display, except that each candidate has a check box next to their name.
- **OPENENDED** The question text is displayed, and then a 1 column table is generated where the text for each choice (or part) is displayed followed by a text area for the user to input the answer.
- **RANKX** After the question is displayed, a table is generated consisting of all of the choices and the information for each choice. After each choice, a line of text containing "Choice x" is added. Following the choices table, another table is generated which contains X drop-down menus (X is the number of things to rank). Each drop down menu contains "Choice 1" through "Choice n" (n is the number of total choices) as well as a "No Preference" option which is initially selected. Each drop down menu corresponds to one of the ranking spots (first, second, third etc...) and is labeled as such.

Question Verification

Question verification is used to make sure that all of the user's selections fall within the parameters of the election. Users will not be allowed to submit a vote in which they have selected 6 candidates when the question asked for 4. Due to the differing nature of form inputs, some question types will be easier to verify then others. The following lists what needs to be done for each type of question, when the verification function is given the question information and an array consisting of all of the question answers the user entered for that question.

- **CHOOSEONE** The verification needs to make sure that only one answer is provided for this question, or that no answer is provided.
- **CHOOSEX** The verification needs to make sure that the number of selected answers is less than or equal to the number of answers that the voter is allowed to select.
- **OPENENDED** The verification needs to make sure that the number of answers for the question (the number of parts of the question) is less than or equal to the number of parts the question has.
- **RANKX** The verification needs to do two things here. One is to make sure that there are X answers provided (if someone doesn't want to vote for their fifth choice, then "No Preference" will be selected). Also, the verification needs to make sure that no two choices are the same, unless both of the choices are "No Preference".

Note: For most of these questions, checking to make sure that the right number of answers is present is to guard against someone submitting answers that don't come from the election. If the answers come from the election scripts, there is no way there could be too many choices for some types of questions.

Question Submission

Question submission is fairly straight forward once it has been checked that all of the user's inputs are valid. Generating the array to submit to the database consists of iterating through each of the questions and adding an entry to the array to submit for each one. The keys of the array are simply "question n" where n is the number of the current question. The value of a given key is the value to be stored for that question in the array and varies slightly for each question type.

- **CHOOSEONE** The answer the user selected is entered into the database.
- **CHOOSEX** All of the answers the user selected are concatenated with ":::" being placed between each answer. Order does not matter.
- **OPENENDED** The text the user entered is stored in the database.

- **RANKX** All of the choices are concatenated together in the same manner as CHOOSEX questions, but this time order is important. The first choice is first in the string, the second choice second etc...

How This Works In PHP

The way that all of this is setup is as follows. All of the functions pertaining to questions are written in a Question.php script which contains a function to gnerate the question HTML, a function to validate a question, and a function to generate the string to submit to the database for each question.

The rest of the functionality is achieved through three other PHP scripts: welcome.php, election-gen.php and submit.php. The welcome.php script is responsible for seeing if an election is running, and determining what kind of election is in progress and directing the user to an appropriate election-gen.php script. The election-gen.php script lives in five different directories and each copy is slightly different (to deal with different years etc...). This script generates the HTML for all the questions and displays it. It also generates the over-arching form which submits to submit.php. The submit.php script is responsible for sorting through all of the answers passed to it from the form in election-gen.php and generating answers for one question at a time. It then validates all questions, and if validation is successful it then generates the array to submit to the database and submits it. Depending on the value returned from the database, the submit.php script then tells the user they are finished or directs them to an appropriate error page, but all three options will contain a logout button.

Interactions

Depends On:

- XML Component
- Database Component

Is Used By:

- The voters