Assignment Specification: Profiles, Positions, and Education

In this next assignment, you will extend our simple resume database to support Create, Read, Update, and Delete operations (CRUD) into a Education table that has a many-to-one relationship to our Profile table and a many-to-many relationship to an Institution table. We will add an jQuery autocomplete field to our user interface.

This assignment will also feature a jQuery auto-complete field when entering the name of the school.

Sample solution

You can explore a sample solution for this problem at http://www.wa4e.com/solutions/res-education/

Resources

There are several resources you might find useful:

- You might want to refer back to the resources for the previous assignment.
- An article from Stack Overflow on Add/Remove HTML Inside a div Using JavaScript
- Documentation for jQuery Autocomplete
- Sample code: jQuery, JSON, JSON Chat, JSON CRUD

Additional Tables Required for the Assignment

This assignment will add one more table to the database from the previous assignment. We will create **Education** and **Institution** tables and connect them to the **Profile** table.

```
CREATE TABLE Institution (

  institution_id INTEGER NOT NULL KEY AUTO_INCREMENT,

  name VARCHAR(255),

  UNIQUE(name)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE Education (

  profile_id INTEGER,

  institution_id INTEGER,

  rank INTEGER,

  year INTEGER,

  CONSTRAINT education_ibfk_1

    FOREIGN KEY (profile_id)

    REFERENCES Profile (profile_id)

    ON DELETE CASCADE ON UPDATE CASCADE,

  CONSTRAINT education_ibfk_2

    FOREIGN KEY (institution_id)

    REFERENCES Institution (institution_id)

    ON DELETE CASCADE ON UPDATE CASCADE,

  PRIMARY KEY(profile_id, institution_id)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

You must create the **Institution** table first so that the CONSTRAINTS in the **Education** table work properly.

Like in the **Position** table, the **rank** column should be used to record the order in which the positions are to be displayed. Do not use the **year** as the sort key when viewing the data.

You should also pre-insert some University data into the Institution table as follows:

```
INSERT INTO Institution (name) VALUES ('University of Michigan');

INSERT INTO Institution (name) VALUES ('University of Virginia');

INSERT INTO Institution (name) VALUES ('University of Oxford');

INSERT INTO Institution (name) VALUES ('University of Cambridge');

INSERT INTO Institution (name) VALUES ('Stanford University');

INSERT INTO Institution (name) VALUES ('Duke University');

INSERT INTO Institution (name) VALUES ('Michigan State University');
INSERT INTO Institution (name) VALUES ('Mississippi State University');

INSERT INTO Institution (name) VALUES ('Montana State University');
```

This will allow you to have some university names pop up when you are typing ahead in the School field.

JavaScript and CSS

To make this work use the following JavaScript / CSS in your document's head area:

Optional : As shown in the code walkthrough. Copy lines 3-11 into a new file, head.php, remembering to add the require_once(scriptname); as needed.

```
<head>
...
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6
    /css/bootstrap.min.css" integrity="sha384-1q8mTJOASx8j1Au
    +a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7" crossorigin="anonymous">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6
    /css/bootstrap-theme.min.css" integrity="sha384-fLW2N01lMqjakBkx3l
    /M9EahuwpSfeNvV63J5ezn3uZzapT0u7EYsXMjQV+0En5r" crossorigin="anonymous">
  <link rel="stylesheet" href="https://code.jquery.com/ui/1.12.1/themes/ui
    -lightness/jquery-ui.css">
  <script src="https://code.jquery.com/jquery-3.2.1.js" integrity="sha256-DZAnKJ
    /6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE=" crossorigin="anonymous"></script>
  <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js" integrity="sha256
    -T0Vest3yCU7pafRw9r+settMBX6JkKN06dqBnpQ8d30=" crossorigin="anonymous"
    ></script>
...
</head>
```

Type Ahead

If you look a the sample implementation, the actual typeahead code in the browser is pretty simple starting with the text field that we want to add the auto-complete to:

```
School: <input type="text" size="80" name="edu_school1" class="school" value=""

  />
```

This is a normal input tag to which we have added the "school" class. We attach typeahead code to this input box using the following jQuery code:

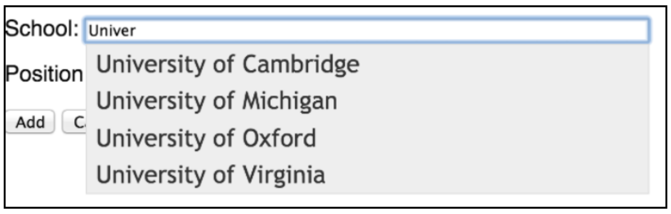
```
$('.school').autocomplete({ source: "school.php" });
```

This simply says that we want to have autocomplete active for all input tags with a class of "school" and to call the script **school.php** with the partially typed school name using the following calling sequence (make sure your are logged in to the sample application before accessing this URL):

http://www.wa4e.com/solutions/res-education/school.php?term=Univer

The **term** is whatever has been typed into the input field so far. The HTTP response from the **school.php** is a JSON array of items to displayed as the autocomplete list:

["University of Cambridge","University of Michigan", "University of Oxford","University of Virginia"]



After error checking and session checking the code to return the list of universities that match the prefix typed so far looks as follows:

```
$stmt = $pdo->prepare('SELECT name FROM Institution WHERE name LIKE :prefix');

$stmt->execute(array( ':prefix' => $_REQUEST['term']."%"));

$retval = array();

while ( $row = $stmt->fetch(PDO::FETCH_ASSOC) ) {

  $retval[] = $row['name'];

}

echo(json_encode($retval, JSON_PRETTY_PRINT));
```

Note : To return the correct HTTP response from school.php for the JSON array, it requires a Content header to be sent before the data. See the sample code json-01.zip/JSON.php, links in the Resources section at the top of this page.

The Screens for This Assignment

We will be extending the user interface of the previous assignment to implement this assignment. All of the requirements from the previous assignment still hold. In this section we will talk about the additional UI requirements.

- **add.php** You will need to have a section where the user can press a "+" button to add up to nine empty education entries. Each education entry includes a year (integer) and a school name.

- **view.php** Will show all of the education entries in an un-numbered list and the positions in another un-numbered list.
- **edit.php** Will support the addition of new position or education entries, the deletion of any or all of the existing entries, and the modification of any of the existing entries. After the "Save" is done, the data in the database should match whatever positions and education entries were on the screen and in the same order as the positions on the screen.
- **index.php** No change needed.
- **login.php** No change needed.
- **logout.php** No change needed.
- **delete.php** No change needed.

If the user goes to an add, edit, or delete script without being logged in, die with a message of "ACCESS DENIED".

Optional : You might notice that there are several common operations across these files. You might want to build a set of utility functions to avoid copying and pasting the same code over and over across several files. In the lecture these are shown as head & util.php. If you decide to do this remember to load these before any script or functions are called.

For example, Code Walkthrough: Profile, Positions, Education, and JSON @ 07:36 edit.php, the require at the top.

Data validation

In addition to all of the validation requirements from the previous assignment, you must make sure for the education entries - year, description and institution name are non-blank and that all years are numeric.

All fields are required

or

Year must be numeric

Submitting Your Assignment

As a reminder, your code must meet all the specifications (including the general specifications) above. Just having good screen shots is not enough - we will look at your code to see if you made coding errors. For this assignment you will hand in:

- A screen shot (including the URL) of your add.php showing one existing Education institution and one new education institution

- A screen shot (including the URL) of your edit.php showing one education entry modified, one education entry deleted and one new education entry
- A screen shot (including the URL) of your view.php showing the correct new education entry after the edit is complete
- A screen shot (including the URL) of your add.php showing the error message for a bad year
- A screen shot of your Education database table showing at least three rows
- A screen shot of your Institution database table showing at least one new row inserted when the institution name did not match an existing instutition
- Source code of add.php
- Source code of view.php
- Source code of edit.php
- Source code any file in your application that you choose (i.e. util.php)

General Specifications

Here are some general specifications for this assignment:

- You **must** use the PHP PDO database layer for this assignment. If you use the "mysql_" library routines or "mysqli" routines to access the database, **you will receive a zero on this assignment.**
- Your name must be in the title tag of the HTML for all of the pages for this assignment.
- All data that comes from the users must be properly escaped using the **htmlentities()** function in PHP. You do not need to escape text that is generated by your program.
- You must follow the POST-Redirect-GET pattern for all POST requests. This means when your program receives and processes a POST request, it must not generate any HTML as the HTTP response to that request. It must use the "header('Location: ...');" function and either "return" or "exit();" to send the location header and redirect the browser to the same or a different page.
- All error messages must be "flash-style" messages where the message is passed from a POST to a GET using the SESSION.
- Please do not use HTML5 in-browser data validation (i.e. type="number") for the fields in this assignment as we want to make sure you can properly do server side data validation. And in general, even when you do client-side data validation, you should still validate data on the server in case the user is using a non-HTML5 browser.