Izmir University CEN 345 Web Programming

Homework 2: A Simple Phone Book **19 November 2013**

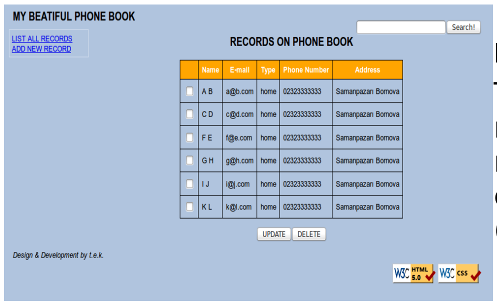
Due Date: 10 December 2013

This assignment is about making a simple online "phone book" site that processes HTML forms with PHP. You should turn in at least the following files:

- **index.php**, a PHP page which lists all the records in the "phonebook.txt" file as a table. This page must support listing only searched records in the case of search form action
- **action.php**, the page that receives data submitted by record form and performs the specified action (delete, update, and add)
- **record.php**, a page with an empty phone record form. This page is used for initiating update, add and delete operations
- **validator.js**, this JavaScript file contains the form validator functions

The files **header.html** and **footer.html**, which contain common header/footer HTML code must be included in your all pages. You should also include the complete CSS file **page.css** in all of the pages. You should be able to fully style all pages using the styles in **page.css** only. You are free to use any style without missing the requirements too much (you must have the strict layout that is seen on the images).

The details about each page's contents and behavior are described on the following sections. Screen-shots in this document are from Ubuntu GNU/Linux in Chromium Browser, which may differ from your system.



MY BEATIFUL PHONE BOOK

LIST ALL RECORDS
ADD NEW RECORD

RECORDS ON PHONE BOOK

	Name	E-mail	Type	Phone Number	Address
☐	A B	a@b.com	home	02323333333	Samanpazarı Bornova
☐	C D	c@d.com	home	02323333333	Samanpazarı Bornova
☐	F E	f@e.com	home	02323333333	Samanpazarı Bornova
☐	G H	g@h.com	home	02323333333	Samanpazarı Bornova
☐	I J	i@j.com	home	02323333333	Samanpazarı Bornova
☐	K L	k@l.com	home	02323333333	Samanpazarı Bornova

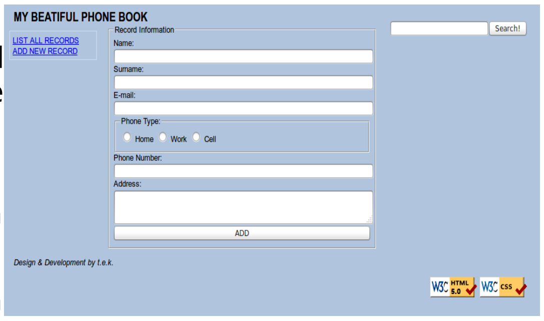
Design & Development by t.e.k.

Index Page (index.php) and Overall Site Navigation:

This page is the start of the site. It must list all the records in the file when it is requested by browser. It must be developed according to this. **footer.html** contains link to the this page and empty record page (**record.php**).

Record Page (record.php):

The **record.php** page a **form** to create a new record together with header and footer. You must write the HTML code for the form. The form contains the following labeled fields:



MY BEATIFUL PHONE BOOK

LIST ALL RECORDS
ADD NEW RECORD

Record Information
Name:
Surname:
E-mail:
Phone Type: Home Work Cell
Phone Number:
Address:

Design & Development by t.e.k.

- **Name:** A 16-character box for the user to type a name.
- **Surname:** A 16-character box for the user to type a surname.
- **E-mail:** A 32-character box for the user to type an e-mail. It must conform to the rules of the e-mail.
- **Phone type:** Radio buttons for the user to select a phone type of *Home, Work* and *Cell*. When the user clicks the text next to a radio button, that button should become checked. Initially *Home* is checked.

- **Phone number:** A 16-character box for the user to type a phone number. It can be only numbers and plus (+) sign.
- **Address:** User should enter address. It must be a text area.
- **Submit button:** This submit button value will change according to the how user has came to this page. If she came using GET method its value is going to be "ADD" and when pressed it will initiate add operation. If she came using "Update" operation from **index.php** page by a POST method then its value is going to be "UPDATE" and when pressed it will initiate update operation. These operations are explained below.

Performing the actions (action.php):

This PHP script should perform add, delete and update operations that are initiated from different parts of the page. Your site's phone book data is stored in a file **phonebook.txt**, placed in the same folder as your PHP files. The file contains data records as lines in *exactly* the following format, with the user's full name (which is a combination of name and surname), e-mail, phone type, phone number, and address, separated by tab space (\t):

```
1     A B   a@b.com     home  02323333333 Samanpazarı Bornova
2     C D   c@d.com     home  02323333333 Samanpazarı Bornova
3     F E   f@e.com     home  02323333333 Samanpazarı Bornova
4     G H   g@h.com     home  02323333333 Samanpazarı Bornova
5     I J   i@j.com     home  02323333333 Samanpazarı Bornova
6     K L   k@l.com     home  02323333333 Samanpazarı Bornova
```

(You can use this list as an initial version of phonebook.txt file)

ID on the left side are required for the delete and update operations. Finding next ID for the new record is an important part of the development of the phone book.

Searching (index.php):

In the header there is a search form. Anytime user clicks the "Search!" button, **index.php** will show only the matched results in the records table. Anything in any field must be compared with keyword and any matching record must be listed in the results table on the **index.php**.

JavaScript Validator (validator.js):

In all pages, **check for valid data** for the file's contents and form submissions. For example, no fields will be left blank or contain illegal characters (such as a tab space). No user will resubmit data for a name already in the system. You should validate form values using JavaScript (**validator.js**) validation and PHP validation after form submitted to the server (for example for the same name validation you should use PHP validation). Your JavaScript must be elegant, clean, understandable and unobtrusive. Don't mix HTML, CSS and JavaScript in the same files. Separate style, behavior and content.

Styling:

The styles you need are already given to you in **page.css**, but you still need to use proper tags and `class` attributes to make sure they are applied. Be mindful of the styles on forms and form controls. On the course web site and lecture slides there are several screen-shots of the various pages. Make sure that your form has the same width, colors, fonts, borders, etc. as in these examples. If you choose the right tags to represent your form, it should match. Make sure that form fields line up in **columns** by using a `strong` tag or `column` class so that each text label floats to the left and is 11em wide.

Use **debug `print` and `print_r` statements** to track down bugs. For example, you can `print_r($_GET);` or `$_POST` to see the query parameters submitted. Use **Firebug/Inspect Element** and also **View Source** to find HTML output problems.

Recall that form controls must have `name` attributes. Sometimes you must also add a `value` to affect how data is sent.

Implementation and Grading:

- Your HTML output for all pages must pass the W3C **HTML validator**. (Not the PHP source code itself, but the HTML output it generates.) Do not use HTML tables. Since we are using HTML **forms**, choose proper form controls and set their attributes accordingly. Properly choose between GET and POST requests for sending data.
- Your PHP code should not cause errors or warnings. Minimize use of the `global` keyword, use indentation/spacing, and avoid lines over 100 characters.
- Some HTML sections are shared redundantly between your PHP pages, found in the provided files **header.html** and **footer.html**. Include these files as appropriate in your other pages using the PHP `include` function.
- A major grading focus is **redundancy**. Use **functions, parameters/return, included files/code**, loops, variables, etc. to avoid redundancy. If you have PHP code you want to share between multiple pages, you may turn in an optional file named **common.php** containing this code. You can `include` your **common.php** in your other pages.
- For full credit, reduce the amount of large chunks of PHP code in the middle of HTML code. Replace such chunks with **functions** declared at the top or bottom of your file. You will also lose points if you use PHP `print` or `echo` statements. Insert dynamic content into the page using PHP **expression blocks**, `<?= ... ?>` , as taught in class.
- Another grading focus is PHP **commenting**. Put a descriptive comment header at the top of each file, **each function**, and each section of PHP code.
- **Format your HTML and PHP code** similarly to the examples from class. Properly use whitespace and indentation. Do not place more than one block element on a line or begin a block element past the 100th character.
- Please do not place this assignment or a solution to this assignment on-line on a publicly accessible web site, neither during nor after the school quarter is over. Doing so is considered a violation of our course academic integrity policy.
- Copying is prohibited. Don't copy other groups work.
- Maximum 3 student can team up to do the assignment. For special situations contact me!
- Please send your homework to the Moodle system as a zip file with all the required files.

Assist. Prof. Dr. Tahir Emre KALAYCI