1 Practice 4.1. Forms

> This practice consists of the first of a series of practices, consisting of a Contacts application. Each practice will be based on the previous one. To start, clone the repository task as explained in the end of the practice.
>
> The inclusion of comments in the scripts will be valued.
>
> Once done, do the submission via GitHub Classroom as detailed in the end of the practice.

1.1 Part 1

Create a self-validated form, called `contact_form.php`, with the next fields:

- **ID**: a text input. We don't want the user to change its value, so set it to `readonly`.
- **Title**: a set of radio buttons with the options "Mr.", "Mrs." and "Miss". Default "Mr.".
- **First name** and **surname**: text inputs
- **Birth date**: a **date** input.
- **Phone** and **email**: text inputs.
- **Type**: a set of checkboxes with the values "Favorite", "Important" and "Archived".
- **3 input buttons**: Save, Update and Delete.

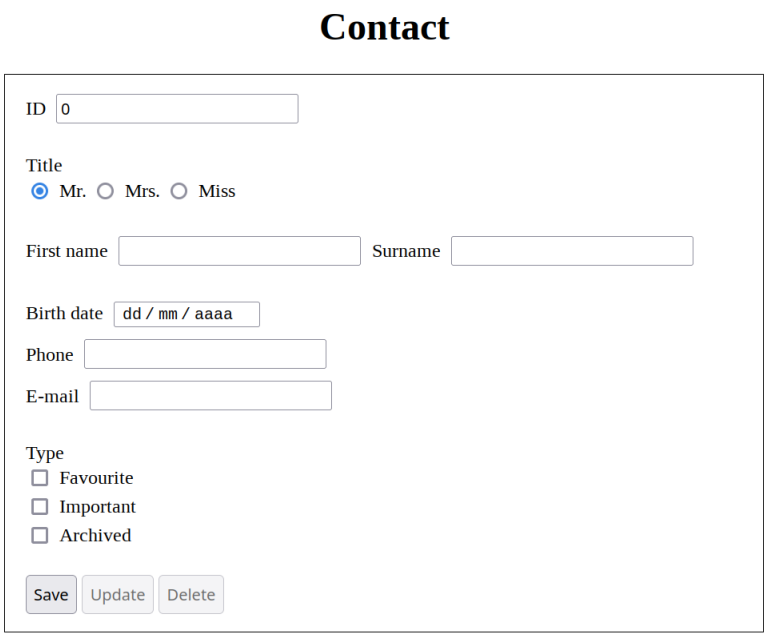

Figure 1: Contact form

Feel free to give the page the style you want. You can also use a framework like Bootstrap.

We want to do the next validations:

- **name**, **surname** and **birthdate**: required. As the date input must be of type date, we don't need validation, because it has the form "yyyy-mm-dd".
- **title**: if it has no value, assign the default value "Mr."
- **phone**: required and must be a valid phone number (use `preg_match()` and regular expressions).
- **email**: required and must be a valid email (use `filter_var()`).
- **id** and **type**: we don't need any validation.

When the user pushes the "Save" button, do the validation. If all the data is correct, open a new script, called `checkdata.php`, that shows the message "No errors!" and all the data of the new contact. Use for this **`header()`** and **session variables**.

> **Important**: If you are programming in Windows, make sure **all the folders and files are written in lowercase**, to prevent errors in Linux servers.

1.2 Part 2

Create a php page, called `contact_list.php`, which shows a table with the next data from the sample array provided in the GitHub repository:

Figure 2: Contact list

Feel free to give the page the style you want.

In each row, make an *"Edit/View"* button which, when pressed, opens the `contact_form.php` form and send to it the contact **id**. You can do this with the GET method. Then, the form will show all the data associated with the contact identified by the id (use the contacts array). When the user presses the *Update* button, the form will do all the validations.

In the `contact_list.php` page, make a button with the text "Create new contact". When pressed, it will open the `contact.php` form without data in the inputs (only the default value for *title*).

Optionally, if the form has been opened from the contact list, disable the "Save" button (attribute `disabled`). If the form has been opened directly or from the "Create new contact" button, disable the "Update" and "Delete" buttons.

You can use the `showTable` function of the previous practice to do the table.

In this practice we won't use the *Delete* button.

1.3 Part 3

Make at least 2 partials, one for the beginning and the other for the end of your pages, and change your existing pages to use the partials.

These partials must include variables for the title, author, header, etc. Add a footer to the final partial.

1.4 How to submit to GitHub Classroom

1. You need a GitHub account. Create one if you don't have any.
2. Click on the assignment invitation link. Enter your GitHub credentials and authorize GitHub Classroom to access your GiHub account.
3. Select your name from the "Join the Classroom" list.
4. Accept the assignment.
5. Go to your assignment repository. Clone it and start coding.
6. Once you finish the task, make a commit with the comment "SUBMISSION COMMIT" and push it to GitHub. Before that, you can do the commits and pushes you want. If you change your code after your submission commit, make another commit and push with the same text in the message adding the corrections you've done (for instance: "SUBMISSION COMMIT, corrected function mistake"). Make a pull-request if you want so i could know your code is submitted.
7. Once pushed to GitHub, create a tag with the name "Practice 4". Create a new tag if you make corrections after the first submission (ex. "Practice 4 corrected").

If you have any doubt in your task, you can push your code and ask me by email what's your problem. It will make it easier for both the solutions of code issues.