TASK

PHP -

HTML Form Handling

Visit our website

Introduction

MAKE IT EASY FOR THE USER

In the last task, you saw one way for PHP to take input — with GET request parameters. But this isn't very user friendly. Luckily there is another well-supported way of getting user input that's much simpler and easier to use. In this task, we'll cover what HTML forms are, and how to use them to get user input.

BASIC HTML FORMS

HTML comes built with support for many different kinds of user-input forms — things you see on every web form like checkboxes, selection spinners, text forms, and buttons. With no additional scripting required, standard HTML forms can be used directly to send information to your PHP script. It does this by automatically making a POST request and converting user input to appropriate parameters.

To declare a form, add the following to an HTML file:

```
<form>
   <label for="input_title">Title:</label><br>
   <input type="text" id="input_title" name="title"><br>
   <label for="input_lname">Last name:</label><br>
   <input type="text" id="input_lname" name="last_name">
</form>
```

HTML forms are always enclosed with **`form`** tags. Inside a form you will typically have several **`input`**s and **`label`**s for those inputs. The **`name`** attribute specifies what the parameter name will be when making the POST request. The **`id`** attribute isn't strictly necessary, but is useful for associating labels and if you want to style your forms in CSS. By convention, **`input`** tags are never closed.

To give the form functionality, specify the **`action`** and **`method`** attributes of the form tag, and add a **`submit`** button inside it:

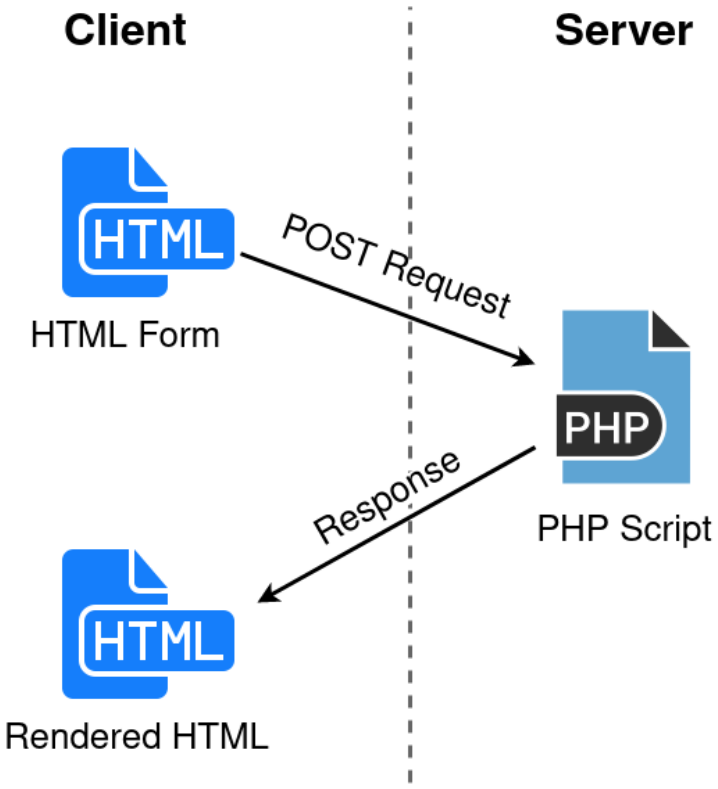
```
<form action="form-handler.php" method="post">
   <!-- inputs and labels ... -->
   <button type="submit">Submit</button>
</form>
```

The action attribute specifies where the user input will be sent, and the method specifies how (what kind of request to make).

To accept the request, we first need to create the PHP file to which it's being sent (`form-handler.php`). The submitted data can be found in the `$_POST` superglobal.

```
<?php
echo '<h1>Greetings, '. $_POST['title'] . ' ' .
    $_POST['last_name'] . '.</h1>';
?>
```

Go ahead and test it out. Load up your HTML file in a browser, fill in the details, and click Submit to see the magic happen. Just to be clear, this is what the "magic" entails:

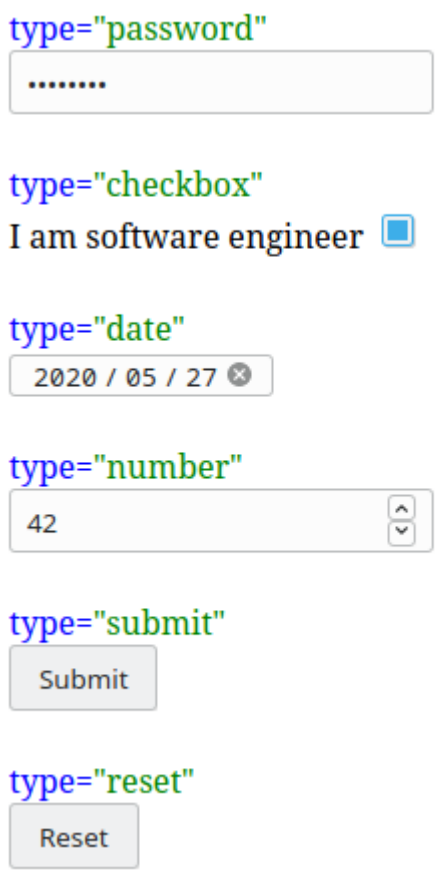


The HTML form's submit button triggers a POST request with all the user input to the server. The server executes the target PHP script, which uses the request parameters to generate a response. This response is then sent back to the client.

Note that the POST request is made by the main browser window, meaning that the window's URL will change to the POST destination and the response will be rendered directly in-browser. This is one reason many developers don't use HTML forms to submit data. Instead, they write some JavaScript code to make the request in the background and return the response to the same page (without having to redirect the user). We won't be covering this method in this task, but it's good to know there are alternatives.

MORE INPUTS

A comprehensive list of form input types can be found **here**. Below are some commonly used input types:



Text-based inputs allow for the `placeholder` attribute, which can greatly help a user understand what to enter into a text box.

```
<form>
    <label for="animal">Favourite animal:</label>
```

```
    <input type="text" name="animal" id="animal" placeholder="Dragon">
</form>
```

Favourite animal: Dragon

COMPLEX INPUTS

Some inputs require more than one element to be functional. For instance, to make a radio group:

```
<form>
   <input type="radio" id="male" name="gender" value="male">
   <label for="male">Male</label>
   <input type="radio" id="female" name="gender" value="female">
   <label for="female">Female</label>
   <input type="radio" id="other" name="gender" value="other">
   <label for="other">Other</label>
</form>
```

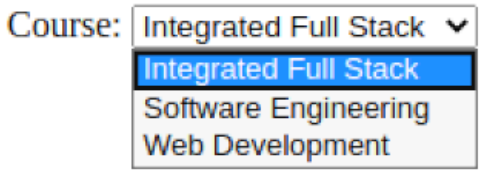
would look something like:

Male Female Other

To make sure only one option is selectable, specify all the input **name**s to be the same. Note that the selected radio's **value** will be POSTed, not its **label**. If you want more than one radio group in a form, be sure to make the name different per group.

To make a dropdown box:

```
<form>
   <label for="sel">Course:</label>
   <select name="course" id="sel">
       <option value = "ifs" selected>Integrated Full Stack</option>
       <option value = "se">Software Engineering</option>
       <option value = "webdev">Web Development</option>
   </select>
</form>
```



As with radio buttons, the selected option's **`value`** will be POSTed, not its text content. Add the **`selected`** attribute (no value needed) to specify which option should be selected by default.

INPUT RESTRICTIONS

You can optionally impose input restrictions to help reduce user input error before the data is submitted. A full list is available at the **same link as above**, but here are some common usages:

```
<form>
   <!-- quantity with minimum 0 and maximum 100 -->
   <input type="number" name="quantity" min="0" max="100">

   <!-- limit name length to 50 characters -->
   <input type="text" name="full_name" maxlength="50">

   <!-- this checkbox cannot be altered -->
   <input type="checkbox" name="is_developer" disabled>
</form>
```

Note that some input types like date, email, and url automatically impose formatting restrictions.

In the next task, we'll look at more rigorous ways of input checking.

Compulsory Task 1

Create an HTML form that captures data about a user's place of work.
The folder "Compulsory Task 1" contains a PHP file that's ready to take POST requests. Inspect it and create an appropriate HTML form.

- Make sure a user can click "Submit" to have their input automatically sent to the PHP script.
- Ensure no fields are left out.

Compulsory Task 2

Create a PHP file that parses data captured about a user's place of residence and renders it in neat HTML. The folder "Compulsory Task 2" contains an HTML file with an HTML form ready to submit said data. Inspect it and create an appropriate PHP form handling script.

- Make sure to parse all fields and render it neatly so that the user can easily confirm that their details have been entered correctly.

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.