

Tutorial/Laboratory 06

Form Processing with HTML5, Bootstrap & PHP

1. LEARNING OUTCOMES

Upon completion of these laboratory exercises, you should be able to:

- Create responsive, user-friendly forms for collecting input.
- Implement a comprehensive strategy for validating & sanitizing form input utilizing a combination of client-side and server-side techniques.

2. REQUIRED SOFTWARE

- Apache NetBeans 11.2 (or later):
<https://netbeans.apache.org/download/index.html>
- FireFox (<https://www.mozilla.org/en-US/firefox/new/>) or Chrome (<https://www.google.com/chrome/>) web browser.
- PHP: <https://www.php.net/downloads>

3. EXERCISE 1: ADD A MEMBER REGISTRATION FORM

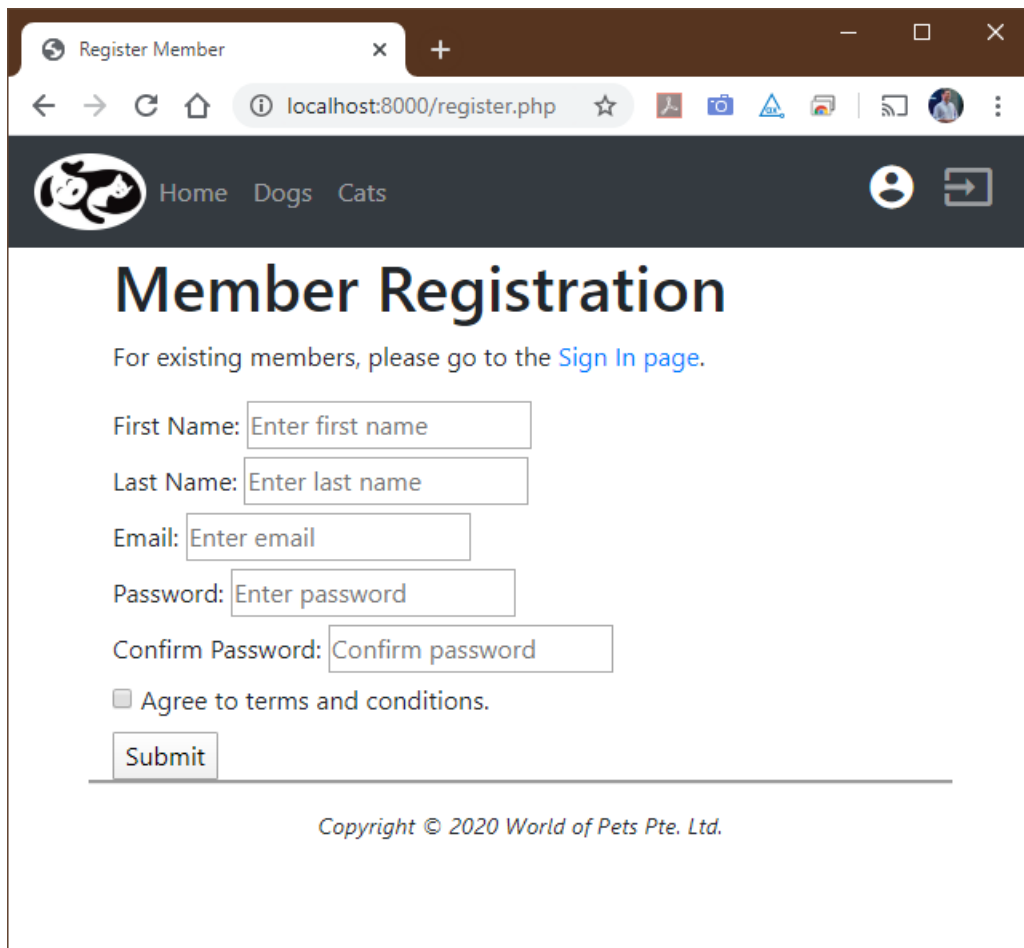
- 3.1 In this exercise, we'll add a form that will allow users to register as World of Pets members.
- 3.2 Make a copy of your latest World of Pets project from the previous Lab and add a new PHP page called **register.php**. You'll also want to add a menu item to your <nav> element that links to register.php.
- 3.3 Replace the default contents in register.php with your own <head> element (you can copy from index.php or use a PHP include), then add the following <body> element:

```
<body>
  <?php
  include "nav.inc.php";
  ?>

  <main class="container">
    <h1>Member Registration</h1>
    <p>
      For existing members, please go to the
      <a href="#">Sign In page</a>.
    </p>
    <form action="process_register.php" method="post">
      <label for="fname">First Name:</label>
      <input type="text" id="fname" name="fname"
        placeholder="Enter first name">
      <br>
      <label for="lname">Last Name:</label>
      <input type="text" id="lname" name="lname"
        placeholder="Enter last name">
      <br>
      <label for="email">Email:</label>
      <input type="email" id="email" name="email"
        placeholder="Enter email">
      <br>
      <label for="pwd">Password:</label>
      <input type="password" id="pwd" name="pwd"
        placeholder="Enter password">
      <br>
      <label for="pwd_confirm">Confirm Password:</label>
      <input type="password" id="pwd_confirm" name="pwd_confirm"
        placeholder="Confirm password">
      <br>
      <label>
        <input type="checkbox" name="agree">
        Agree to terms and conditions.
      </label>
      <br>
      <button type="submit">Submit</button>
    </form>
  </main>

  <?php
  include "footer.inc.php";
  ?>
</body>
```

This will give us a basic HTML5 form without any bells or whistles:



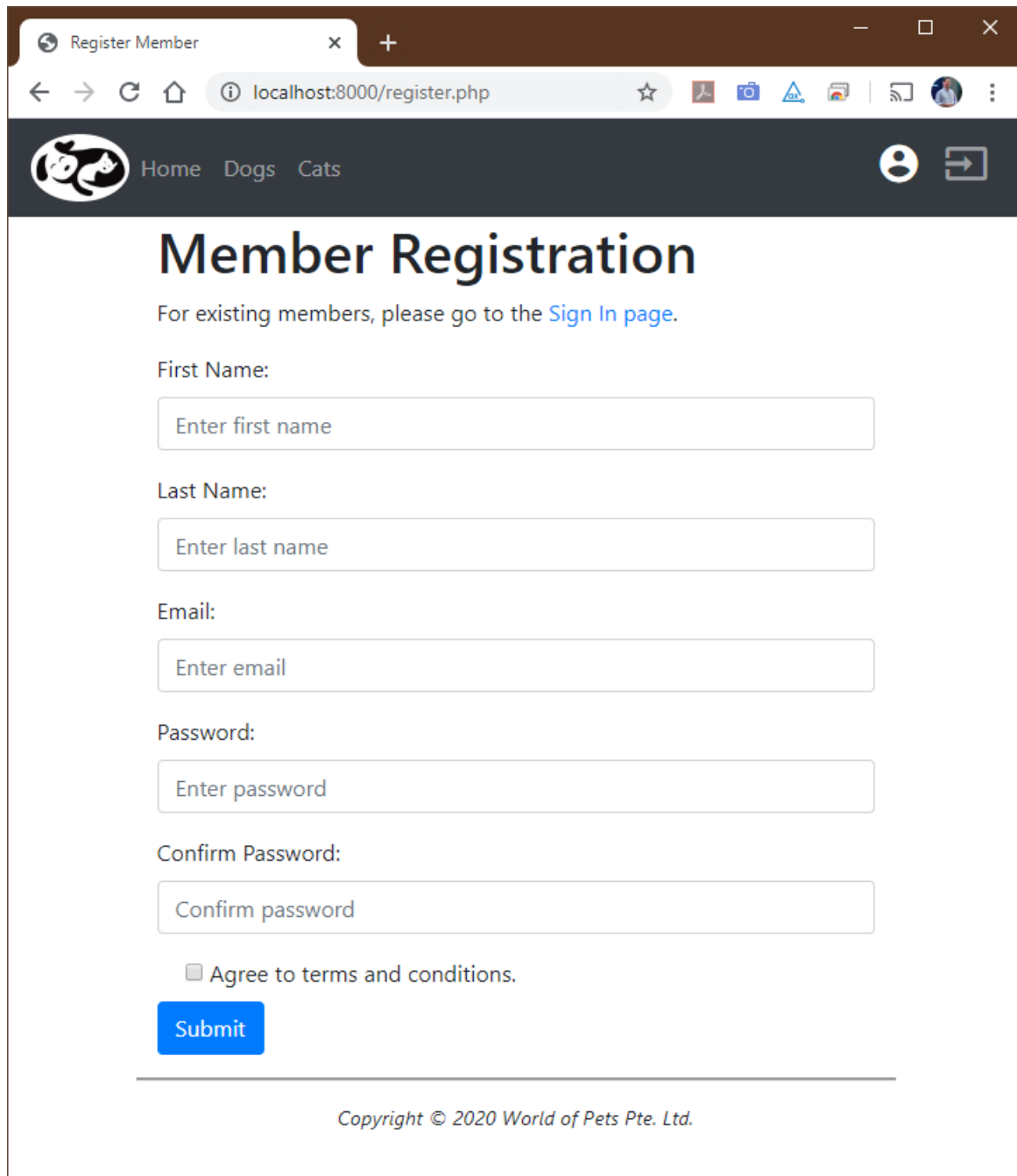
This form is semantically correct and accessible, but it is not responsive nor very presentable, so let's fix that using Bootstrap.

- 3.4 Follow these steps to add Bootstrap classes to our form in order to make it responsive and more attractive:
 - a. Remove the `
` tags separating the input groups and replace with `<div>` elements using the **form-group** class. This class allows you to group form items, such as a `<label>` and its corresponding `<input>` control, together and provides responsive formatting.
 - b. Add the **form-control** class to each of the textual input controls. This class provides styles for general appearance, focus state, sizing, and more.
 - c. Wrap the checkbox element and its outer `<label>` element in a `<div>` with the **form-check** class.
 - d. Add the **btn** and **btn-primary** classes to the submit `<button>` element.
 - e. For more details on formatting forms with Bootstrap, refer to: <https://getbootstrap.com/docs/4.3/components/forms/>.

After implementing the above changes, your form code should be similar to this:

```
<form action="process_register.php" method="post">
  <div class="form-group">
    <label for="fname">First Name:</label>
    <input class="form-control" type="text" id="fname"
      name="fname" placeholder="Enter first name">
  </div>
  <div class="form-group">
    <label for="lname">Last Name:</label>
    <input class="form-control" type="text" id="lname"
      name="lname" placeholder="Enter last name">
  </div>
  <div class="form-group">
    <label for="email">Email:</label>
    <input class="form-control" type="email" id="email"
      name="email" placeholder="Enter email">
  </div>
  <div class="form-group">
    <label for="pwd">Password:</label>
    <input class="form-control" type="password" id="pwd"
      name="pwd" placeholder="Enter password">
  </div>
  <div class="form-group">
    <label for="pwd_confirm">Confirm Password:</label>
    <input class="form-control" type="password" id="pwd_confirm"
      name="pwd_confirm" placeholder="Confirm password">
  </div>
  <div class="form-check">
    <label>
      <input type="checkbox" name="agree">
      Agree to terms and conditions.
    </label>
  </div>
  <div class="form-group">
    <button class="btn btn-primary" type="submit">Submit</button>
  </div>
</form>
```

...and the registration page should now look and behave much better:



Register Member

localhost:8000/register.php

Home Dogs Cats

Member Registration

For existing members, please go to the [Sign In page](#).

First Name:

Last Name:

Email:

Password:

Confirm Password:

☐ Agree to terms and conditions.

Submit

Copyright © 2020 World of Pets Pte. Ltd.

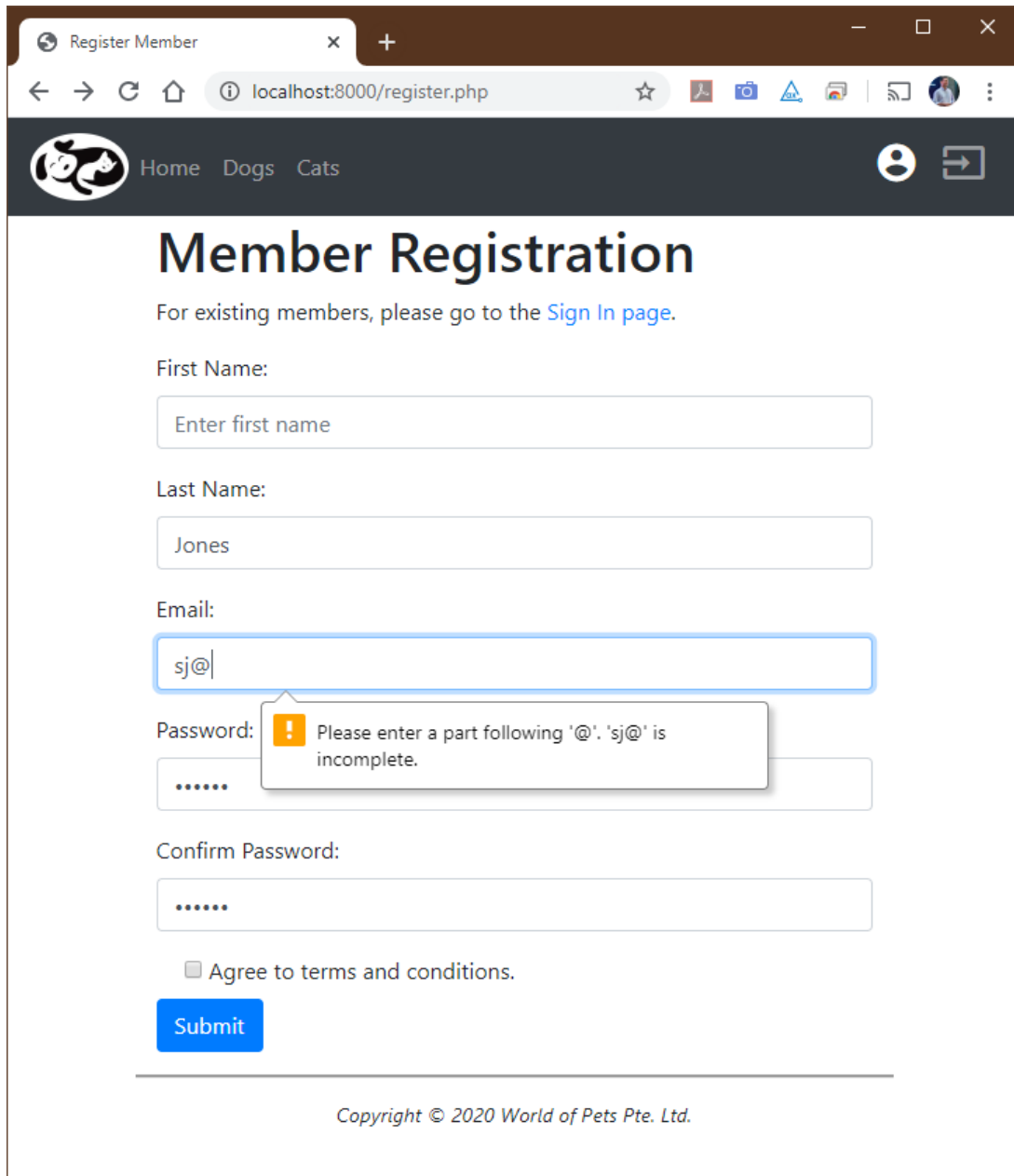
- 3.5 Lastly, now that we are adding additional pages to our website, you'll want to implement a method for highlighting the menu item corresponding to the page currently being viewed. For example, in the screenshot above, the icon for member registration is highlighted instead of the 'Home' menu item. There are a number of ways to do this using JavaScript and/or CSS - below is one way to do it using JavaScript/jQuery:

```
/*
 * This function toggles the nav menu active/inactive status as
 * different pages are selected. It should be called from $(document).ready()
 * or whenever the page loads.
 */
function activateMenu()
{
    var current_page_URL = location.href;

    $(".navbar-nav a").each(function()
    {
        var target_URL = $(this).prop("href");
        if (target_URL === current_page_URL)
        {
            $('nav a').parents('li, ul').removeClass('active');
            $(this).parent('li').addClass('active');
            return false;
        }
    });
}
```

4. EXERCISE 2: FORM VALIDATION

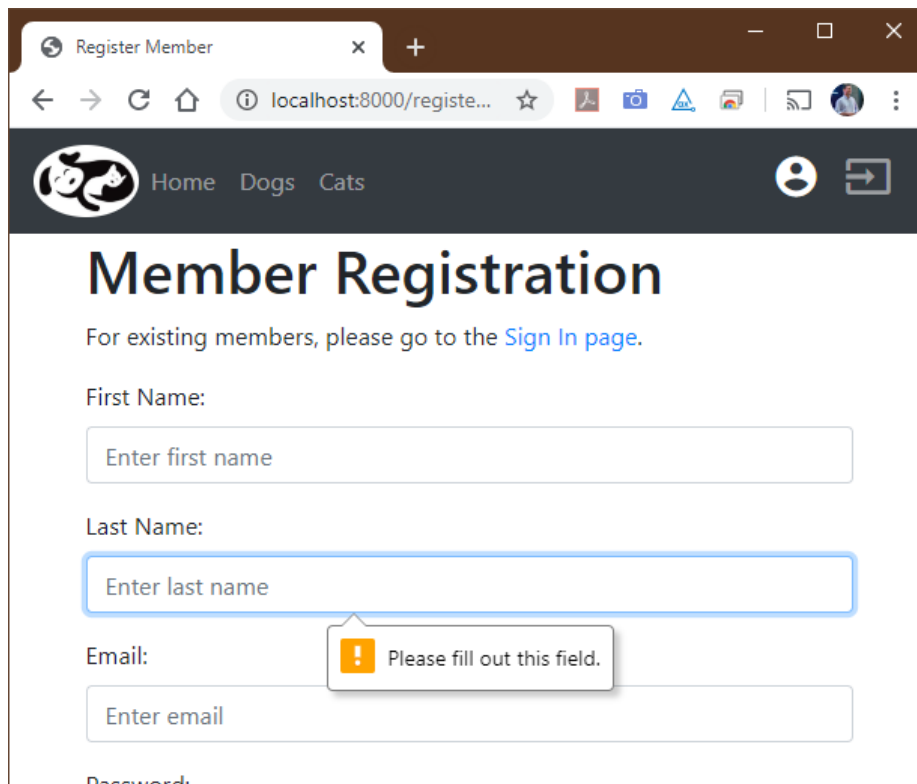
- 4.1 As it stands, there is no validation (other than the **type** attribute) for our form - users can submit empty or invalid input. Or even worse, a nefarious person could submit malicious code and hack our website! In this exercise, we'll use a combination of client-side and server-side form validation & sanitization to make our form more robust and secure.
- 4.2 Let's start by adding some HTML5 attributes to help with validation. We've already used the **type** attribute, which indicates to the browser the type of input field. Not only does this attribute help make our form semantically correct and aid accessibility, it also can provide some validation depending on the type. For example, `type="email"` will tell the browser to do some basic checking to verify if the text is a properly formatted email address. And `type="password"` will instruct the browser to hide the characters being typed. Example:



The next attribute we'll use is **required** - this specifies whether a form field needs to be filled in before the form can be submitted. In our case, we want to require all fields except for First Name, since it's possible for a person to have only one name (as is common in Indonesia). Insert the **required** attribute in all form elements except for the fname <input> and the <button>. Example:

```
<input class="form-control" type="text" id="lname"
      required name="lname" placeholder="Enter last name">
```

Now if you try to submit the form without filling in the text boxes, you'll see an error message from the browser:



- 4.3 Another useful set of HTML5 attributes are **minlength** and **maxlength**. When applied to a textual `<input>` field, they restrict the number of characters allowed. Let's set the maximum length of our name fields to 50 characters, e.g.:

```
<input class="form-control" type="text" id="lname" required
      maxlength="50" name="lname" placeholder="Enter last name">
```

- 4.4 There are a few more attributes available to help with client-side form validation, but for our case, these should be sufficient. For more information, refer to: https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation.

Note that you may also use JavaScript to perform more customized validation on the client if you have the need. Just be aware that JavaScript can be easily bypassed.

- 4.5 At this point, we have standard client-side form validation to ensure that the user does not leave any required fields blank, the email address is more or less formatted correctly, etc. The form is also responsive and provides immediate feedback to the user if they forgot to enter something.

However, this is only half the job - we must also perform *sanitization* in addition to validation, to protect against security breaches. Malicious users often employ an attack known as cross-site scripting ([XSS](#)) to inject client-side script into web pages viewed by others. [SQL injection](#) is another common exploit. Hijacking form submissions is a typical way to carry out such attacks, therefore we must **always think security** when processing PHP forms. In the remaining part of this exercise, we'll learn how to process forms securely on the server-side using PHP.

- 4.6 Create a new file called **process_register.php**. In this file, we'll write the PHP script that will process the registration form when it is submitted to the server. Here we can use the **\$_POST[]** superglobal variable to retrieve the values submitted in the form.

Example: `$_POST["fname"]` would return the first name provided in the `<input name="fname">` field.

- 4.7 Consider the following PHP script which validates and sanitizes an email address submitted by the user via a form POST:

```
<?php
$email = $errorMsg = "";
$success = true;

if (empty($_POST["email"]))
{
    $errorMsg .= "Email is required.<br>";
    $success = false;
}
else
{
    $email = sanitize_input($_POST["email"]);

    // Additional check to make sure e-mail address is well-formed.
    if (!filter_var($email, FILTER_VALIDATE_EMAIL))
    {
        $errorMsg .= "Invalid email format.";
        $success = false;
    }
}

if ($success)
{
    echo "<h4>Registration successful!</h4>";
    echo "<p>Email: " . $email;
}
else
{
    echo "<h4>The following input errors were detected:</h4>";
    echo "<p>" . $errorMsg . "</p>";
}

//Helper function that checks input for malicious or unwanted content.
function sanitize_input($data)
{
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>
```

This “bare bones” example only illustrates validation of the ‘email’ form field, but you should be able to extend this pattern to validate all of the remaining input fields. Some interesting things to take note of:

- a. The function `sanitize_input()` is used to filter any malicious or unwanted characters from the form input. As a rule, **NEVER accept user input “as-is”** – you should always employ a filtering method to make sure your website is not victimized by hackers.
- b. The `filter_var()` PHP function is used to further qualify the email address to make sure it matches standard formatting. `filter_input()` is another useful PHP function that you can look into as well.

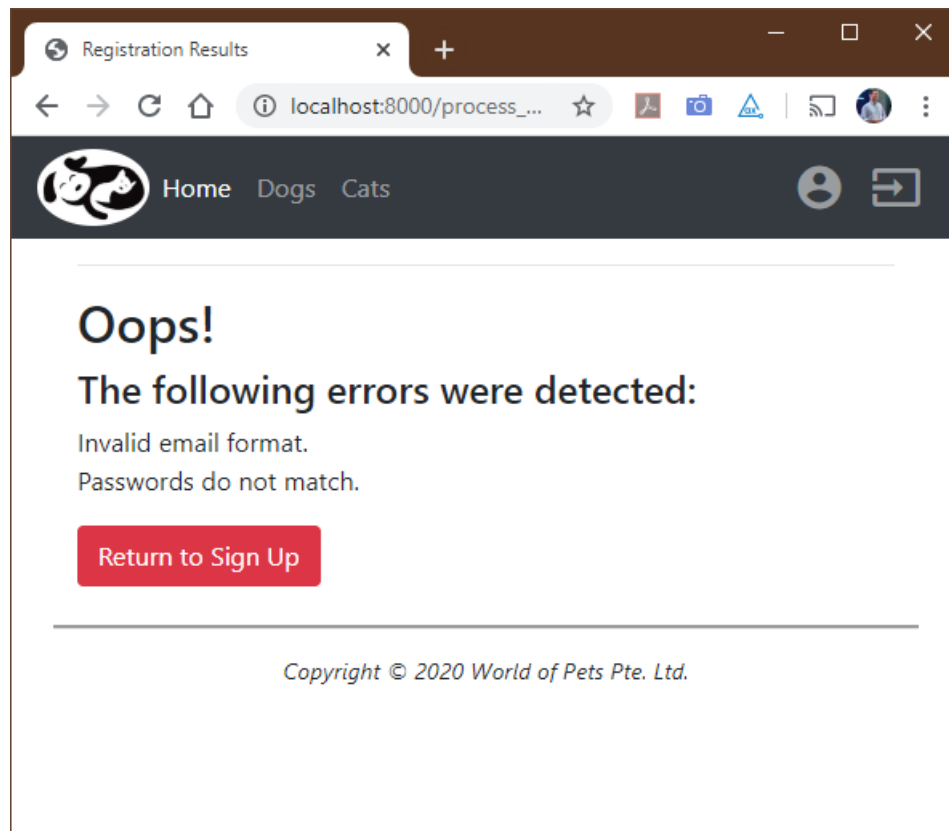
4.8 Using the above script as a guide, fill in the remaining code to validate all of the form data on the server using PHP. At a minimum, you should:

- a. Make sure required fields such as last name, email and password, have been provided.
- b. Verify that the email address is properly formatted and doesn’t contain invalid characters.
- c. Make sure the password matches the confirmation password.
- d. Sanitize/filter all input to protect against malicious attacks.

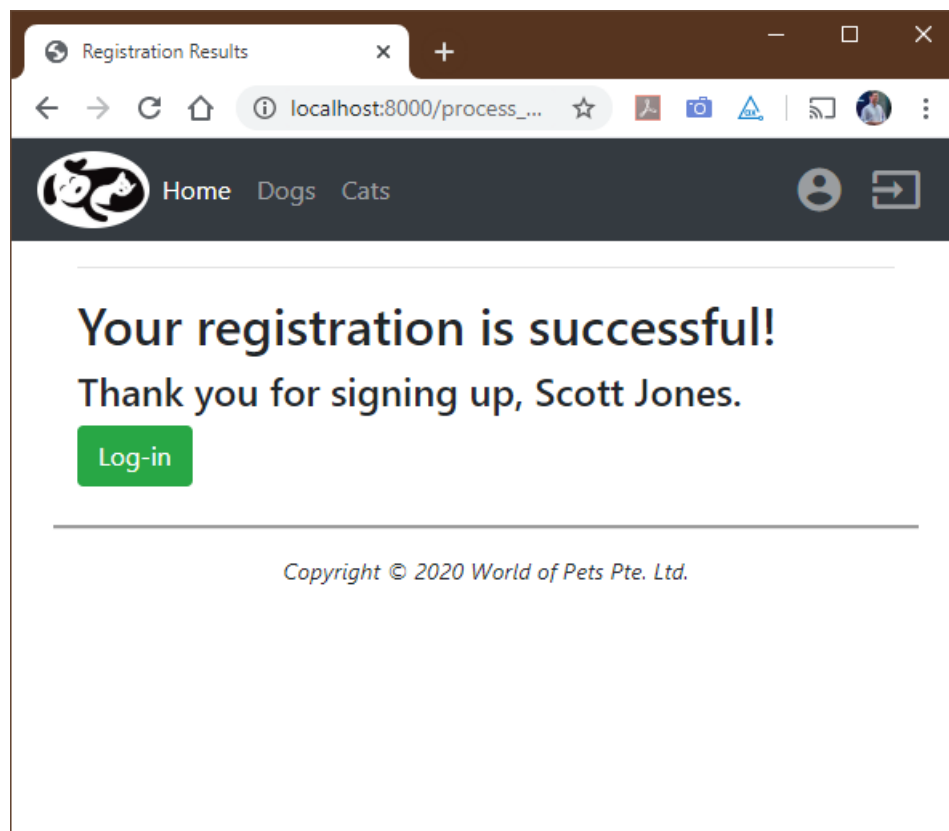
You can temporarily disable your client-side form validation in order to verify that the server-side validation is working.

4.9 Lastly, you can add styling and informative wording to the `process_register.php` page so it is more presentable.

Here’s an example with error message:



And here's an example of successful form submission:



- 4.10 In a future Lab, we'll close the loop by implementing database functionality for saving the user registration, and add a log-in page.

Discussion Questions:

- a. Why do we do validation on *both* the client and the server sides? What are the plusses and minuses of each?
- b. Some web developers will insist that sanitization should only be performed on output, never on input, so as to preserve the user's original input as submitted. What are your thoughts on this?
- c. There are two ways that form data can be submitted to the server: **POST** and **GET**. What are the differences, advantages and disadvantages of these two methods? Why did we choose the POST method for our form? **Tip:** go to https://www.w3schools.com/html/html_forms.asp and read the section titled "The Method Attribute".

5. SUBMISSION OF LAB ASSIGNMENT

- 5.1 In order to receive credit for this Lab assignment, you must submit your completed work to xSiTe LMS before the end of the Lab session. To submit your work:
- a. Save all files and close NetBeans.
 - b. In File Explorer, navigate to the location where you saved your project and right-click on the folder name, then select 'Send to -> Compressed (zipped) folder' and ZIP up your entire NetBeans project(s). **Note: only .zip format is acceptable, do not use .rar, .7z, or any other format.**
 - c. In the ICT1004 module on xSiTe, go to **Assessments->DropBox** and locate the Dropbox folder corresponding to this Lab. Click the link to open the Dropbox then hit the **Add a File** button to submit your .zip file. You may also add comments if desired. Be sure to hit **Submit** to complete your submission.
 - d. Remember to save a copy of your work as we will be building upon this website in subsequent Lab assignments.

6. ADDITIONAL PRACTICE

- 6.1 Once you've completed this Lab assignment, you are encouraged to try out the following online tutorials:
- a. HTML Forms: https://www.w3schools.com/html/html_forms.asp
 - b. Client-side Form Validation: https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation
 - c. Bootstrap Form Validation: <https://getbootstrap.com/docs/4.3/components/forms/?#validation>
 - d. PHP Form Handling: https://www.w3schools.com/php/php_forms.asp

- e. PHP Form Validation:
https://www.w3schools.com/php/php_form_validation.asp
- f. PHP Filters: https://www.w3schools.com/php/php_filter.asp