Internet Programming: Lab 4 - Form Processing

This lab's purpose is to give you some practice writing a PHP script that processes an HTML form. You will also practice server-side form validation using PHP.

In this week's lab, we'll write code to complete a web service that allows students to submit their credit card numbers in exchange for good grades in the class. (Note: This is just a joke! We do not suggest that you actually try to bribe us for grades...) The HTML page `buyagrade.html` will contain a form that submits to `sucker.php`. After submission, `sucker.php` will do some checks to validate the information submitted, and it will output a web page that either confirms the submission or informs the user to try submitting again.

Exercise 1: Creating a Form (roughly 15 minutes)

This exercise involves creating an HTML form that POSTs its submitted data to a PHP program on a server. Download `buyagrade.html` to your disk (right-click the link and choose Save Link As...). You need to modify this HTML file by turning it into an HTML form. You will need to give `name` attributes to the form controls so they will be sent to the server as query parameters; the names are up to you. Also, some form controls (such as radio buttons) need `value` attributes. Fill the Section drop-down list with choices MA through MH.

When you're done with this exercise, your form should look roughly like the screenshot at the top of this handout.

Test your form to see that it is submitting the proper parameters by temporarily setting its `action` attribute to:

http://httpbin.org/get OR http://httpbin.org/post

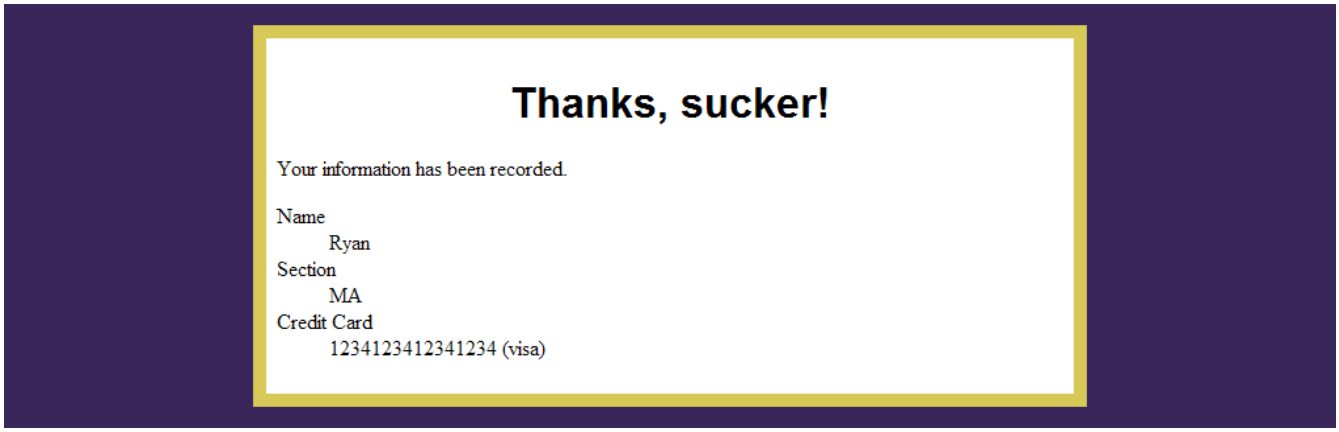
~~`http://webster.cs.washington.edu/params.php`~~

Exercise 2: Displaying Input Data (roughly 10 minutes)

In this exercise, you will write the PHP page that will handle the submitted form data. Tell you `buyagrade.html` to POST to `sucker.php`. The `sucker.php` page will receive the parameters from `buyagrade.html` and will output an HTML confirmation page. Here is a skeleton version of this page that does not actually display the data submitted by the user. Modify it to display the submitted data.

- sucker.php

Your page should at least display the submitter's name, credit card number, and credit card type (Visa or MasterCard) in the confirmation page. For now, your page doesn't actually need to save this information in any way on the server.



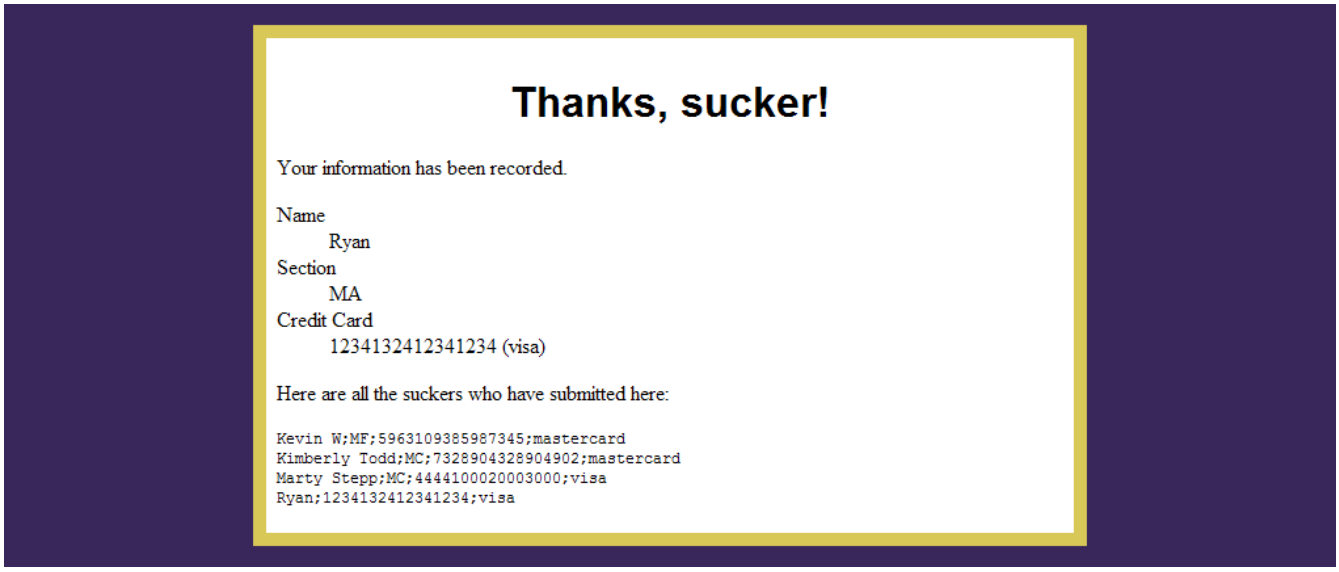
To keep your code clean, as much as possible you should embed variables' values in HTML using PHP expression blocks such as `<?= expression ?>`.

Exercise 3: Save the Form Data (roughly 10 minutes)

Modify your `sucker.php` page to save the submitted data to the file `suckers.txt`. This file should have a format similar to the following:

```
Ryan;MA;1234123412341234;visa
Kevin W;MF;5963109385987345;mastercard
Kimberly Todd;MC;7328904328904902;mastercard
Marty Stepp;MC;4444100020003000;visa
```

Also change your page's output to show the complete contents of this file to the user. Place the file contents into an HTML `<pre>` element to preserve the whitespace.



Recall that you can read and write the contents of a file using the PHP `file_get_contents` and `file_put_contents` functions.

Exercise 4: Basic Data Validation (roughly 10 minutes)

Update your `sucker.php` file to verify that the user did not leave any fields blank when submitting the form. You can check whether a particular parameter has been passed using the PHP function `isset`.

However, `isset($_REQUEST["name"])` will only check if the `$_REQUEST` associative array has `"name"` as one of its keys; it does not check to see if the value of `$_REQUEST["name"]` is a non-empty value. Remember that `$_REQUEST["name"] = ""` would give a falsey value and `$_REQUEST["name"] = "abc"` would give a truthy value.

If the user has not filled in every field, show an error message like the one below instead of displaying their submitted data:

Give them the chance to try again by linking back to your HTML page.