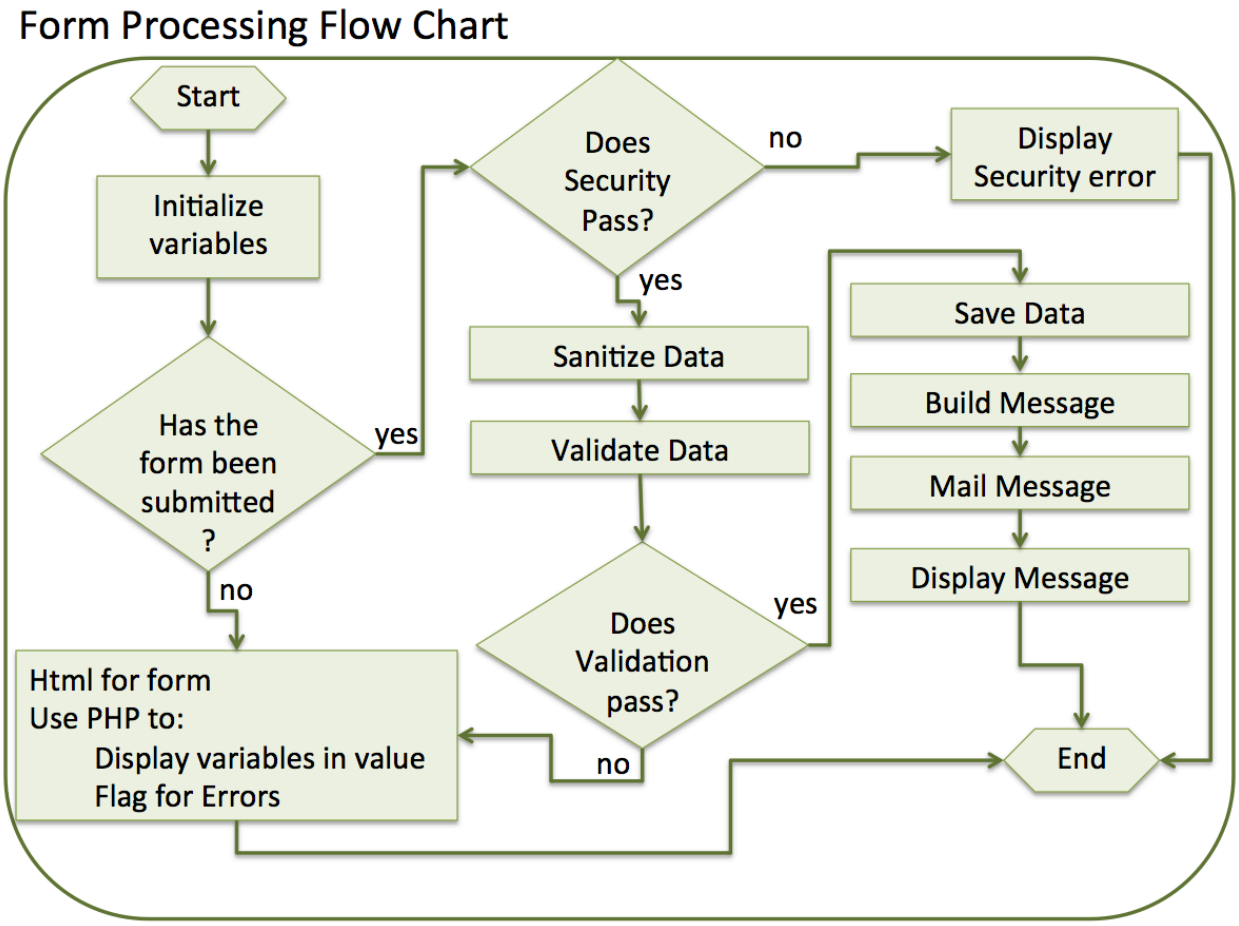
Chapter 10 Form Processing – Adding Elements



At this point we have our form process complete! Of course not many forms just ask for your email address. When we expand our form, we want to follow the same building process of small amounts of code at one time making sure it works before moving on. So we will add to our form one element at a time. Let's start with just adding another text box. What we want to see is the six things we need to do:

1. SECTION: 1c initialize a variable for the element
2. SECTION: 1d initialize an error flag for the element
3. SECTION: 2b Sanitize the value from the POST array
4. SECTION 2c Validate the data. Remember this is a three step process that can be repeated as many times as needed.
    a. Perform your validation check ex: if($firstName=="")
    b. Create your error message display ex: $errorMsg[]="Please enter your first name"
    c. Set Error Flag to true
    d. Repeat for each validation check.
5. SECTION: 2e put the value in the dataRecord array to be saved
6. SECTION 3c Add the HTML for the element

The rest of the code will just work since it is based on loops and arrays so we don't need to change anything else. Each time we add a new element our csv will get a new column. This does mean that any existing rows in the csv file will not have the new column and you should start fresh with a new csv file. The easy way to do this is create an empty file and ftp the file to replace the now old csv file (this will delete any data existing).

The code is very similar for all the form elements but let's go over them and talk about the little things in each one. All elements must have the name attribute or it will not show in the post array. Let's start with just adding another text box since that is what we already have. I want to add First Name.

Text Box

Looking at the six steps I need to make a variable for First Name that sounds like a good name to me so lets take out the blank space and use camel case. We put this code in section 1c.

```
$firstName = "";
```

Technically the order does not matter however in some cases it will make a difference so the best order to use is to initialize your variables in the order they will appear on your from. In my example I am going to put this before $email.

The second step is to initialize an error flag variable. No sense in being creative on a name we can just tack Error to the end of the variable. This code goes in section 1d.

```
$firstNameERROR = "";
```

Since I want to have my code "clean" I am going to add this before the flag for email. I am going to keep the order the same in each step.

The third step is to sanitize our data. For the email address we had a php built in function but for first name we are just going to use html entities like we did in Chapter 4. Of course for this code I need to know the html entity name so I am going to just use firstName but tack on a three letter prefix txt for a text box to come up with txtFirstName. This code goes in section 2b.

```
$firstName = htmlentities($_POST["txtFirstName"], ENT_QUOTES,
"UTF-8");
```

The Fourth step is validate our data. We need to think of what can we check for a first name? We can check to see if it is empty if we require the user to enter it. We can also check if the name has only letters with an occasional number. Luckily we have a function we

made from Chapter 7. Since we have two checks we will use a an if else programming statement. This code goes in section 2c.

```
if ($firstName == "") {
    $errorMsg[] = "Please enter your first name";
    $firstNameERROR = true;
} elseif (!verifyAlphaNum($firstName)) {
    $errorMsg[] = "Your first name appears to have extra
character.";
    $firstNameERROR = true;
}
```

Remember the three step process for validating:

1. What are you going to check for?
2. Prepare your error message for the user.
3. Flag that this element has a mistake.

Do you see the steps above repeated twice?

The Fifth step is prepare our data to be saved. To do this we put the value into the array for saving. I created an array called $dataRecord for this purpose. The order is important as this is the order that the data will be put into excel. It dictates the column order. I am going to stay with the same order which I based on the order the elements appear on the form. So I will put this code before the $email again. This code goes in Section 2e.

```
$dataRecord[] = $firstName;
```

The Six and last step is to create the actual html for the element in section 3c.

```
<p>
     <label class="required text-field" for="txtFirstName">First
Name</label>
     <input <?php  if ($firstNameERROR) print 'class="mistake"';
?>
            id="txtFirstName"
            maxlength="45"
            name="txtFirstName"
            onfocus="this.select()"
            placeholder="Enter your first name"
            tabindex="100"
```

```
            type="text"
            value="<?php print $firstName; ?>"
    >
</p>
```

That completes the process. At this point you would ftp your code and check to make sure it works. There are three things to verify.

1. If you don't add a first name it prompts you with the error message
2. If you type a first name the error message gets cleared. I test this by deleting the email address so I still have a mistake.
3. When the data is correct and submit the form the message displays the value you entered.
4. The email message contains the value you entered.
5. Open the csv file and make sure the value was saved.

Once you have verified that your code works as you expect it to then move on to the next form element that you need. In this case it might be last name ☺ which the process above is the same. I am going to skip to the next form element I want to go over.

Text Area

The next element to talk about is a text area which in many ways is the same as a text box since they contain text just more of it. So here is the code for the six steps:

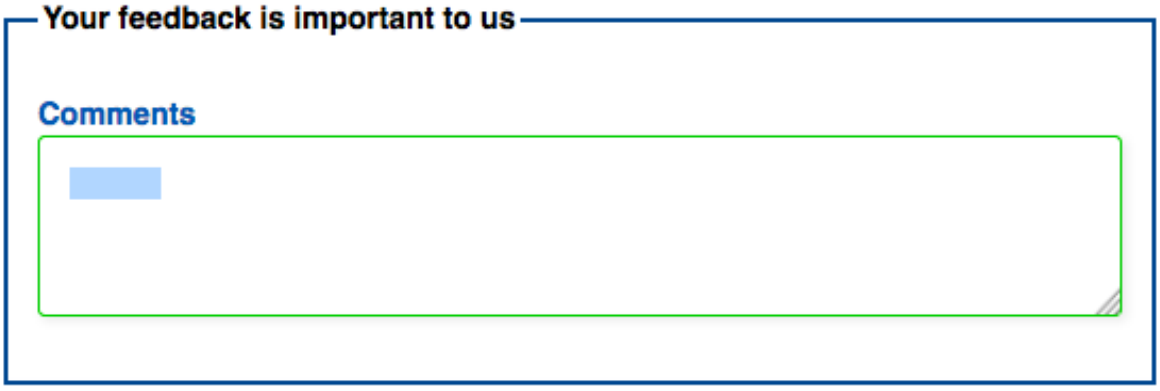
1. ```
   // Initialize variable: SECTION 1c.
   $comments="";
   ```
2. ```
   // Initialize error flag: SECTION 1c.
   $commentsERROR = false;
   ```
3. ```
   // Sanitize the data: SECTION 2b.
   $comments = htmlentities($_POST["txtComments"], ENT_QUOTES,
   "UTF-8");
   ```
4. ```
   // Validate the data: SECTION 2c.
   // Note that this if statements mean the comments are not
   required
   if ($comments != "") {
        if (!verifyAlphaNum($comments)) {
            $errorMsg[] = "Your comments appear to have extra
   characters that are not allowed.";
            $commentsERROR = true;
   ```

```
        }
    }
5. // Prepare to save the data: SECTION 2e.
   $dataRecord[] = $comments;
6. // Create the html for the text area: SECTION 3c.
   <p>
      <label class="required"
             for="txtComments">Comments</label>
      <textarea <?php if ($commentsERROR) print
   'class="mistake"'; ?>
      id="txtComments"
      name="txtComments"
      onfocus="this.select()"
      tabindex="200"><?php print $comments; ?></textarea>
   <!-- NOTE: no blank spaces inside the text area, be sure to
   close the text area directly -->
   </p>
```

If I do I put the html code on separate lines which seems normal like this:

```
tabindex="200">
<?php print $comments; ?>
</textarea>
```

My text area will contain blank spaces. You can see them when the text area receives the focus because they will be highlighted meaning the text area is not empty as you can see by the highlighted blue below.



Radio buttons

Radio buttons are used to help prevent user input errors by giving the user a chance to choice which one option they want. Gender is a common one for radio buttons. So let's go over the six steps. We are letting the user choose one option so we need one variable.

1.
```php
// Initialize variable: SECTION 1c.
$gender="Female";
```
2.
```php
// Initialize error flag: SECTION 1c.
$genderERROR = false;
```
3.
```php
// Sanitize the data: SECTION 2b.
We need to check to make sure there is a value in the post
array, otherwise a php error would result.
if (isset($_POST["radGender"])) {
  $gender = htmlentities($_POST["radGender"], ENT_QUOTES,
"UTF-8");
}else {
  $gender = "";
}
```
4.
```php
// Validate the data: SECTION 2c.
// Note even though we as the programmer provide the value
we need to validate it in case someone tries to sneak
something by.
if($gender != "Female" AND $gender != "Male" AND $gender !=
"Prefer"){
     $errorMsg[] = "Please choose a gender";
     $genderERROR = true;
}
```
5.
```php
// Prepare to save the data: SECTION 2e.
$dataRecord[] = $gender;
```
6.
```php
// Create the html for the radio buttons: SECTION 3c.
<p>
   <label class="radio-field">
   <input type="radio"
          id="radGenderFemale"
    name="radGender"
    value="Female"
    tabindex="582"
```

```
            <?php if ($gender == "Female") echo '
checked="checked" '; ?>
   > Female</label>
</p>
<p>
   <label class="radio-field">
   <input type="radio"
            id="radGenderMale"
            name="radGender"
            value="Male"
            tabindex="584"
            <?php if ($gender == "Male") echo ' checked="checked"
'; ?>
   > Male</label>
</p>
<p>
   <label class="radio-field">
   <input type="radio"
            id="radGenderPrefer"
            name="radGender"
            value="Prefer"
            tabindex="586"
            <?php if ($gender == "Prefer") echo '
checked="checked" '; ?>
   > Prefer not to say</label>
</p>
```

When the form is submitted the radio button that is checked, that value will be in the post array. If nothing is chosen then the radio button is not in the post array.

Check Boxes

Check boxes are used to help prevent user input errors by giving the user a chance to choose every option they want. Check boxes are different than radio buttons because you have one variable for every checkbox (radio buttons have one variable for all radio buttons). So let's go over the six steps. You will need to do the six steps for each checkbox you have. In this example I am going to set the check box to check by assigning true to the variable.

```
1. // Initialize variable: SECTION 1c.
   $promotionalMaterial = true;
```

2. `// Initialize error flag: SECTION 1c.`

   Technically with one check box there are no errors. However sometimes we want a user to select at least one in a group so that is that this is for.

```
$preferenceERROR = false;
$totalPreferenceChecked = 0;
```

3. `// Sanitize the data: SECTION 2b.`

   Technically we are not sanitizing anything but we are setting the value. If the checkbox is not checked then it is not in the post array.

```
if (isset($_POST["chkPromotionalMaterial"])) {
  $promotionalMaterial = true;
}else {
  $promotionalMaterial = false;
}
```

4. `// Validate the data: SECTION 2c.`

   Many times for check boxes there is nothing to validate but if you wanted to make sure the user checked at least one item this is how we can do that.

```
if($totalPreferenceChecked < 1){
   $errorMsg[] = "Please choose at least one preference.";
   $preferenceERROR = true;
}
```

5. `// Prepare to save the data: SECTION 2e.`

```
$dataRecord[] = $promotionalMaterial;
```

6. `// Create the html for the check box: SECTION 3c.`

```
<fieldset class="checkbox <?php if($activityERROR) print '
mistake'; ?>">
<p>
   <label class="check-field">
   <input <?php if ($promotionalMaterial) print " checked
"; ?>
          id="chkPromotionalMaterial"
          name="chkPromotionalMaterial"
          tabindex="620"
          type="checkbox"
          value="Promotional Material"> Would you like to
receive promotional material.</label>
</p>
</fieldset>
```

Notice how I set the field set to a mistake so that the whole area is highlighted. Be sure to make sure when there is a mistake that a user can still the check boxes.

List Boxes

A list box is used when you want the user to choose one of a large list, if your list is small radio buttons may be the best option. However try to image 50 radio buttons to choose a state. There are multiple select list boxes but they go beyond the scope of this text.

So let's go over the six steps.

1.
```php
// Initialize variable: SECTION 1c.
$mountain = "Camels Hump";
```
2.
```php
// Initialize error flag: SECTION 1c.
$mountainERROR = false;
```
3.
```php
// Sanitize the data: SECTION 2b.
$mountain = htmlentities($_POST["lstMountain"], ENT_QUOTES,
"UTF-8");
```
4.
```php
// Validate the data: SECTION 2c.
```
If you build the list box from an array you can check to see if the value is in the array. So this example assumes there is an array called $peaks
```php
if(!in_array($mountain, $peaks)){
   $errorMsg[] = "Please choose a favorite mountain.";
   $mountainERROR = true;
}
```
5.
```php
// Prepare to save the data: SECTION 2e.
$dataRecord[] = $mountain;
```
6.
```php
// Create the html for the list box: SECTION 3c.
<p>
   <legend>Favorite Mountain</legend>
   <select id="lstMountains"
           name="lstMountains"
           tabindex="520" >
      <option <?php if($mountain=="HayStack Mountain")
print " selected "; ?>
           value="HayStack Mountain">HayStack
Mountain</option>

      <option <?php if($mountain=="Camels Hump") print "
selected "; ?>
```

```
            value="Camels Hump">Camels Hump</option>
        <option <?php if($mountain=="Laraway Mountain") print
" selected "; ?>
            value="Laraway Mountain">Laraway
Mountain</option>
   </select>
</p>
```

Summary

Now your form is complete. The nice part is that most of the code you have already written as part of the process will work. You do not need to add anything more to save the information to a cs file. Your existing code will mail the information to the user. Any error messages will be displayed.

Of course this is always more customization that you can do especially with the message in regards to check boxes. As with most customization it involves if statements. Previously we looked at removing the submit button and you can do something similar to create a custom part for check boxes. I hope this has helped you to understand not only the form process but forms in general.

Self Test Questions

1. **Text boxes**
   Create a form with 3 text boxes for Street Address, City, and Zip code. On submit, display the submitted values as one address string. (i.e. 123 Street Rd., Big City, 12345)

2. **Text Area**
   Create a form with a text area. On submit, parse through the submitted text and replace any occurrence of the letter “a” with the letter “z”.

3. **Radio Buttons**
   Create a form with three radio buttons. Selecting option 1 will mail a message to your email, and selecting option 2 will mail a different message to your email. Selecting option 3 will not mail a message at all. Load each message from a respective file.

4. **Check Boxes**
   Create a form with 5 check boxes. Write some php code that displays a different image on the resulting page depending on which of the 5 boxes are selected.

5. **List Boxes**
   Create a form with a list box that contains all 50 states in the US. Display a google maps embed on the resulting page that is centered on the state selected.