

Refer the following link, view the video (Get, Post and Request Method Explained)

<https://www.youtube.com/watch?v=Z9aePaXve6s>

9.17 Using Form Elements with Multiple Options Problem

You have form elements that let a user select multiple choices, such as a drop-down menu or a group of checkboxes, but PHP sees only one of the submitted values.

Solution

End the form element's name with a pair of square brackets ([]).

Example 9-27 shows a properly named group of checkboxes.

Example 9-27. Naming a checkbox group

```
<input type="checkbox" name="boroughs[]" value="bronx"> The Bronx  
<input type="checkbox" name="boroughs[]" value="brooklyn"> Brooklyn  
<input type="checkbox" name="boroughs[]" value="manhattan">  
Manhattan  
<input type="checkbox" name="boroughs[]" value="queens"> Queens  
<input type="checkbox" name="boroughs[]" value="statenisland">  
Staten Island
```

Then, treat the submitted data as an array inside of `$_GET` or `$_POST`, as in Example 9-28.

Example 9-28. Handling a submitted checkbox group

```
print 'I love ' . join(' and ', $_POST['boroughs']) . '!';
```

Discussion

Putting [] at the end of the form element name tells PHP to treat the incoming data as an array instead of a scalar. When PHP sees more than one submitted value assigned to that variable, it keeps them all. If

the first three boxes in Example 9-27 were checked, it's as if you'd written the code in Example 9-29 at the top of your program.

Example 9-29. Code equivalent of a multiple-value form element submission

```
$_POST['boroughs'][] = "bronx";  
$_POST['boroughs'][] = "brooklyn";  
$_POST['boroughs'][] = "manhattan";
```

A similar syntax also works with multidimensional arrays. For example, you can have a checkbox such as

```
<input type="checkbox" name="population[NY][NYC]"  
value="8336697">.
```

If checked, this form element sets `$_POST['population']['NY']['NYC']` to 8336697.

9.18 Creating Drop-Down Menus Based on the Current Date Problem

You want to create a series of drop-down menus that are based automatically on the current date.

Solution

Create a `DateTime` object and then loop through the days you care about, modifying the object with its `modify()` method.

Example 9-30 generates `<option/>` values for today and the six days that follow. In this case, 'today' is April 8, 2013.

Example 9-30. Generating date-based drop-down menu options

```
$options = array();  
$when = new DateTime();  
// print out one week's worth of days  
for ($i = 0; $i < 7; ++$i) {  
    $options[$when->getTimestamp()] = $when->format("D, F j, Y");  
    $when->modify("+1 day");  
}  
foreach ($options as $value => $label) {  
    print "<option value='$value'>$label</option>\n";  
}
```

When run on April 8, 2013, Example 9-30 prints:

```
<option value='1365450257'>Mon, April 8, 2013</option>
```

**<option value='1365536657'>Tue, April 9, 2013</option>
<option value='1365623057'>Wed, April 10, 2013</option>
<option value='1365709457'>Thu, April 11, 2013</option>
<option value='1365795857'>Fri, April 12, 2013</option>
<option value='1365882257'>Sat, April 13, 2013</option>
<option value='1365968657'>Sun, April 14, 2013</option>**

Discussion

In Example 9-30 we set the value for each date as its Unix timestamp representation because we find this easier to handle inside our programs. Of course, you can use any format you find most useful and appropriate.

Using `DateTime#modify()` and `DateTime#format()` frees you from any concerns about time zone math. Whatever the appropriate summer time transitions are for the relevant time zone will be handled properly.