Pemrograman Web

10. PHP: Loop, POST & GET Methods

M. Udin Harun Al Rasyid, S.Kom, Ph.D
http://lecturer.eepis-its.edu/~udinharun
udinharun@eepis-its.edu

Table of Contents

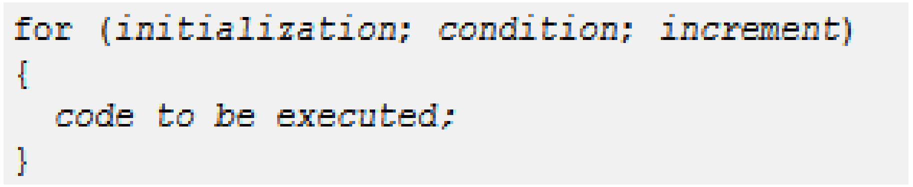
PHP Loop Types

- Loops in PHP are used to execute the same block of code a specified number of times.

- PHP supports following four loop types:
  - **for -** loops through a block of code a specified number of times.
  - **while -** loops through a block of code if and as long as a specified condition is true.
  - **do...while -** loops through a block of code once, and then repeats the loop as long as a special condition is true.
  - **foreach -** loops through a block of code for each element in an array.

The for loop statement

- The for statement is used when you know how many times you want to execute a statement or a block of statements.
- **Syntax**



```
for (initialization; condition; increment)
{
  code to be executed;
}
```

- Example

```
<html>
<body>
<?php
$a = 0;
$b = 0;

for( $i=0; $i<5; $i++ )
{
    $a += 10;
    $b += 5;
}
echo ("At the end of the loop a=$a and b=$b" );
?>
</body>
</html>
```

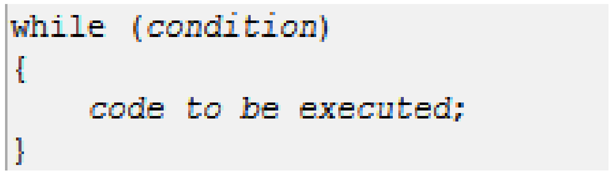
- Result

```
At the end of the loop a=50 and b=25
```

The while loop statement

- The while statement will execute a block of code if and as long as a test expression is true.

- Syntax



```
while (condition)
{
    code to be executed;
}
```

- Example

```
<html>
<body>
<?php
$i = 0;
$num = 50;

while( $i < 10)
{
   $num--;
   $i++;
}
echo ("Loop stopped at i = $i and num = $num" );
?>
</body>
</html>
```

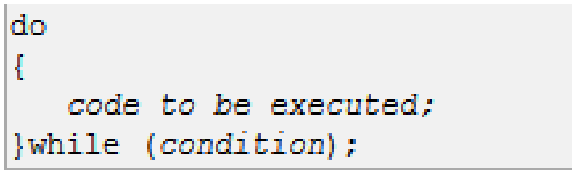
- Result

```
Loop stopped at i = 10 and num = 40
```

The do...while loop statement

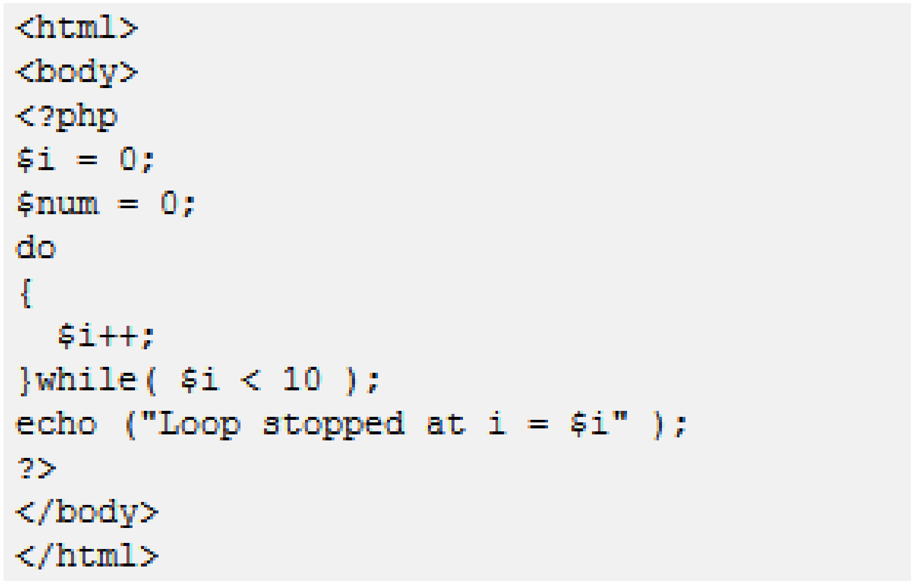
- The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.

- Syntax



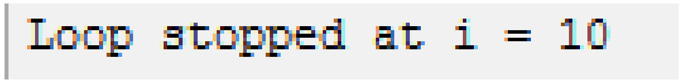
```
do
{
   code to be executed;
}while (condition);
```

- Example



```
<html>
<body>
<?php
$i = 0;
$num = 0;
do
{
  $i++;
}while( $i < 10 );
echo ("Loop stopped at i = $i" );
?>
</body>
</html>
```

- Result



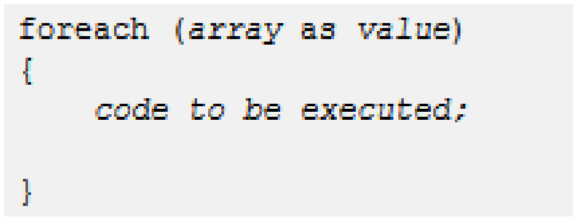
```
Loop stopped at i = 10
```

The foreach loop statement

- The foreach statement is used to loop through arrays.

- For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed.

- Syntax



```
foreach (array as value)
{
    code to be executed;

}
```

- Example

```
<html>
<body>
<?php
$array = array( 1, 2, 3, 4, 5);
foreach( $array as $value )
{
  echo "Value is $value <br />";
}
?>
</body>
</html>
```

- Result

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
```

PHP Web Concepts

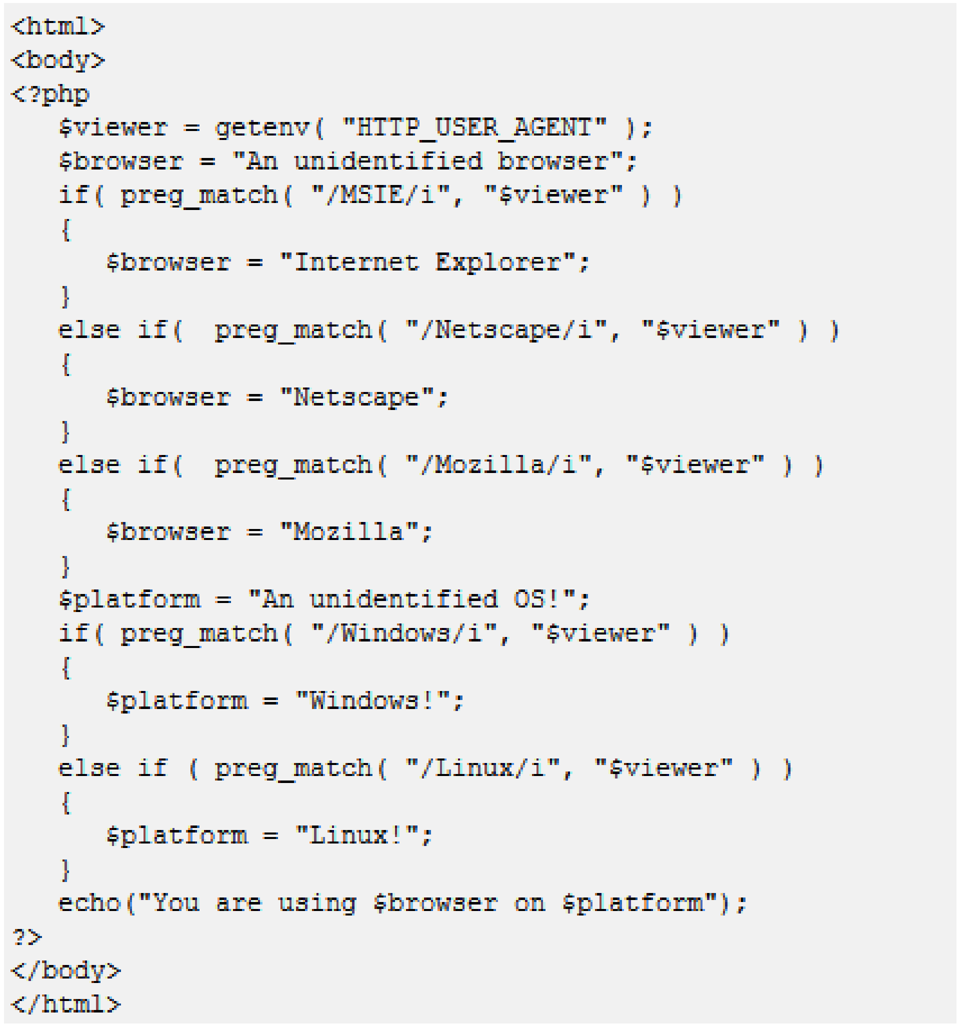
Identifying Browser & Platform

- PHP creates some useful **environment variables** that can be seen in the phpinfo.php page that was used to setup the PHP environment.

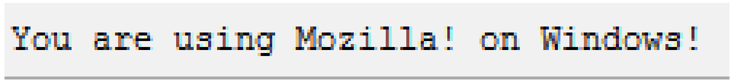
- One of the environemnt variables set by PHP is **HTTP_USER_AGENT** which identifies the user's browser and operating system.

- PHP provides a function getenv() to access the value of all the environment variables.

Example



```
<html>
<body>
<?php
   $viewer = getenv( "HTTP_USER_AGENT" );
   $browser = "An unidentified browser";
   if( preg_match( "/MSIE/i", "$viewer" ) )
   {
      $browser = "Internet Explorer";
   }
   else if(  preg_match( "/Netscape/i", "$viewer" ) )
   {
      $browser = "Netscape";
   }
   else if(  preg_match( "/Mozilla/i", "$viewer" ) )
   {
      $browser = "Mozilla";
   }
   $platform = "An unidentified OS!";
   if( preg_match( "/Windows/i", "$viewer" ) )
   {
      $platform = "Windows!";
   }
   else if ( preg_match( "/Linux/i", "$viewer" ) )
   {
      $platform = "Linux!";
   }
   echo("You are using $browser on $platform");
?>
</body>
</html>
```



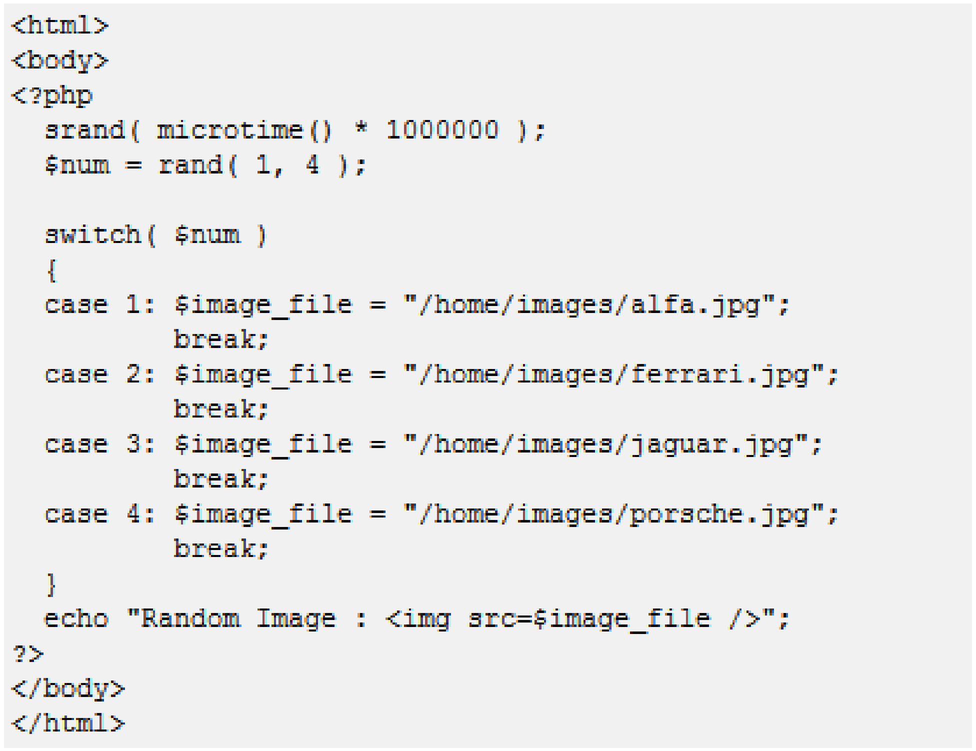
```
You are using Mozilla! on Windows!
```

Display Images Randomly

- The PHP **rand()** function is used to generate a random number.
- The function can generate numbers with-in a given range.

- The random number generator should be seeded to prevent a regular pattern of numbers being generated. This is achieved using the **srand()** function that specifiies the seed number as its argument.

Example



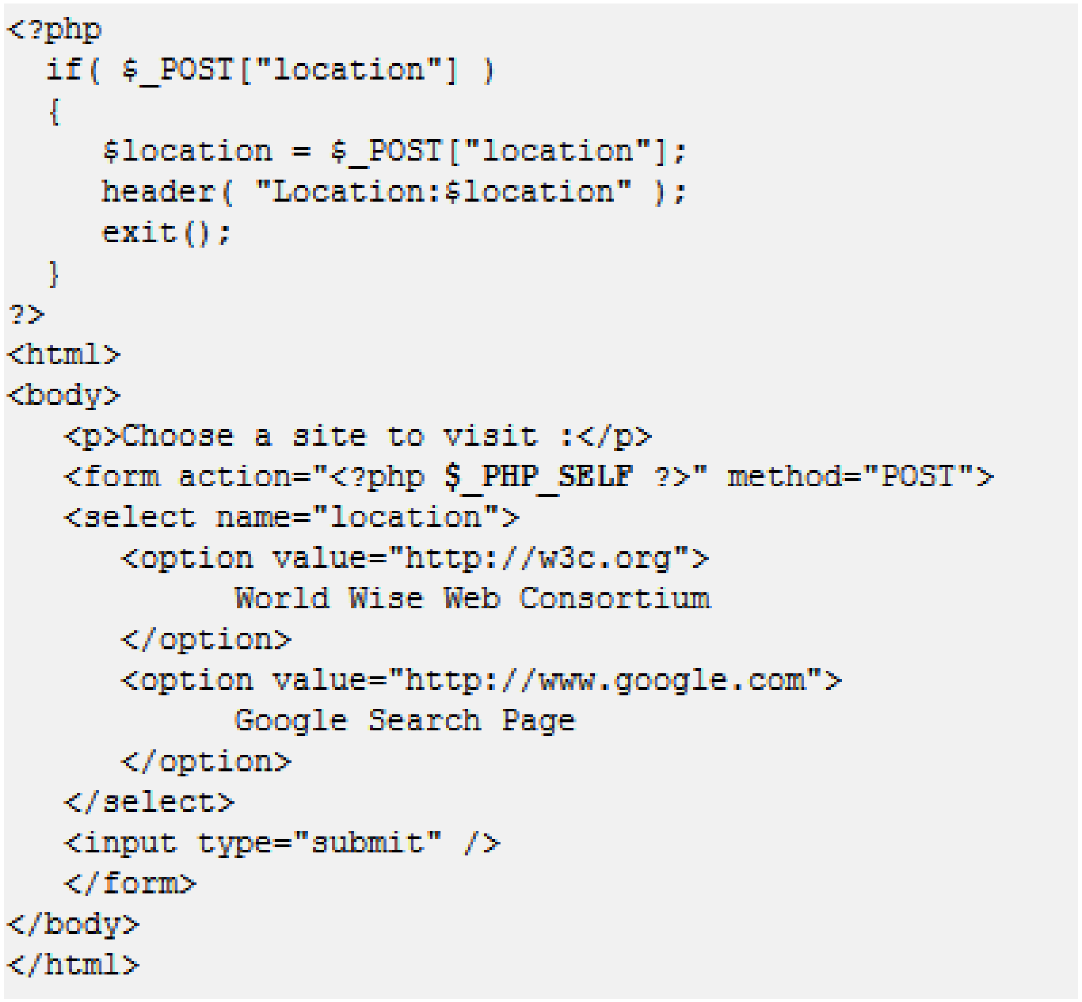
```
<html>
<body>
<?php
  srand( microtime() * 1000000 );
  $num = rand( 1, 4 );

  switch( $num )
  {
  case 1: $image_file = "/home/images/alfa.jpg";
          break;
  case 2: $image_file = "/home/images/ferrari.jpg";
          break;
  case 3: $image_file = "/home/images/jaguar.jpg";
          break;
  case 4: $image_file = "/home/images/porsche.jpg";
          break;
  }
  echo "Random Image : <img src=$image_file />";
?>
</body>
</html>
```

Browser Redirection

- The PHP **header()** function supplies raw HTTP headers to the browser and can be used to redirect it to another location.

- The target is specified by the **Location:** header as the argument to the **header()** function. After calling this function the **exit()** function can be used to halt parsing of rest of the code.

Example



```
<?php
  if( $_POST["location"] )
  {
     $location = $_POST["location"];
     header( "Location:$location" );
     exit();
  }
?>
<html>
<body>
   <p>Choose a site to visit :</p>
   <form action="<?php $_PHP_SELF ?>" method="POST">
   <select name="location">
      <option value="http://w3c.org">
            World Wise Web Consortium
      </option>
      <option value="http://www.google.com">
            Google Search Page
      </option>
   </select>
   <input type="submit" />
   </form>
</body>
</html>
```

PHP GET and POST Methods

- There are two ways the browser client can send information to the web server.
  - The GET Method
  - The POST Method

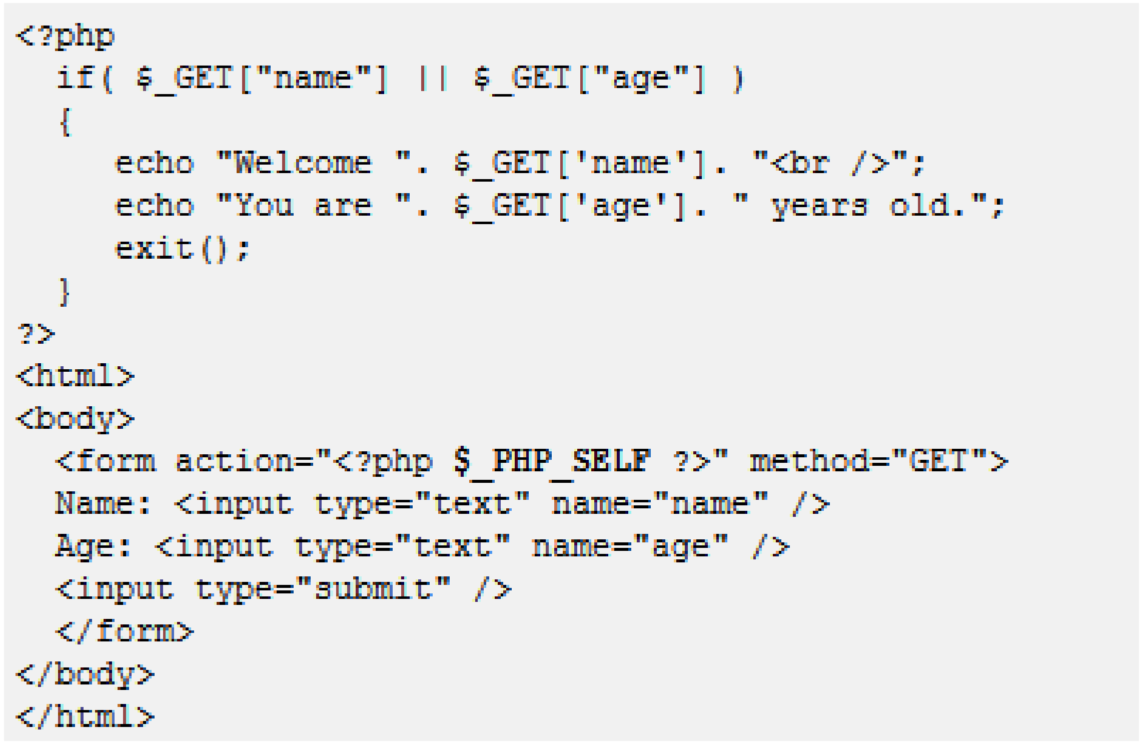
The GET Method

- The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?** character.

```
http://www.test.com/index.htm?name1=value1&name2=value2
```

- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY_STRING environment variable.
- The PHP provides **$_GET** associative array to access all the sent information using GET method.

Example

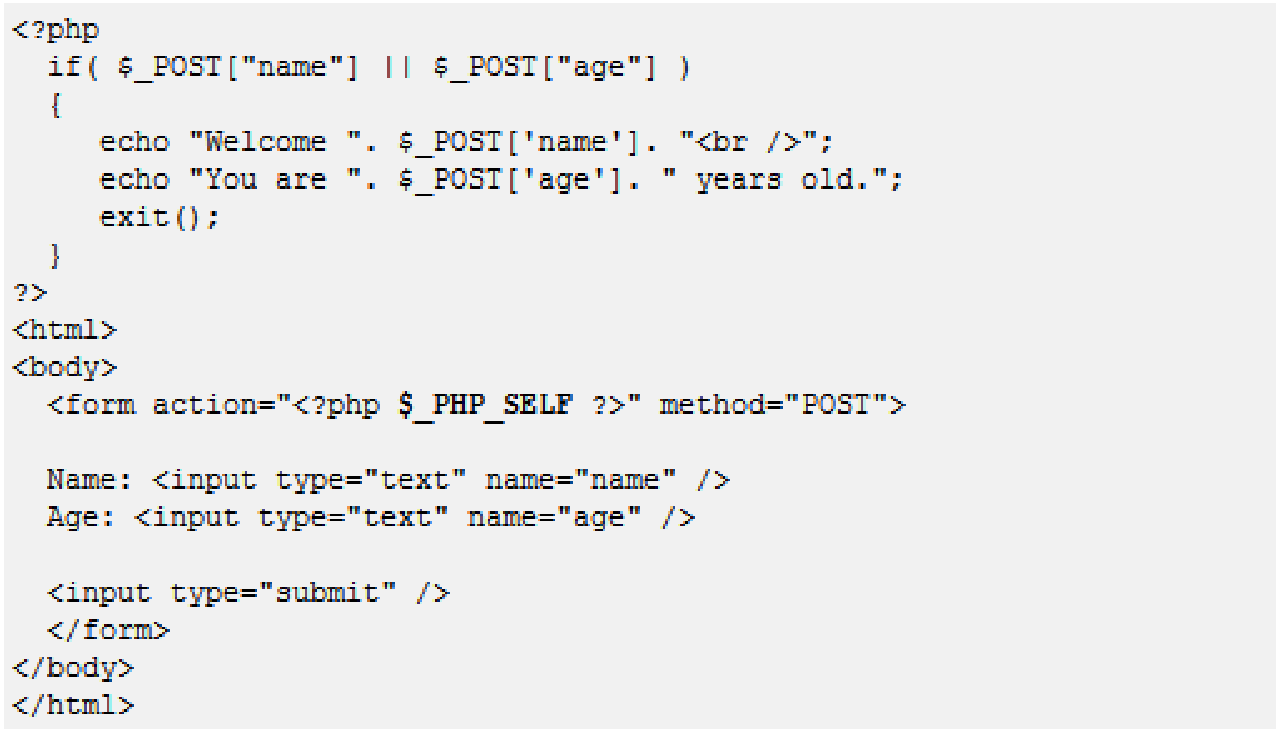


```
<?php
  if( $_GET["name"] || $_GET["age"] )
  {
     echo "Welcome ". $_GET['name']. "<br />";
     echo "You are ". $_GET['age']. " years old.";
     exit();
  }
?>
<html>
<body>
  <form action="<?php $_PHP_SELF ?>" method="GET">
  Name: <input type="text" name="name" />
  Age: <input type="text" name="age" />
  <input type="submit" />
  </form>
</body>
</html>
```

The POST Method

- The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.
- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides **$_POST** associative array to access all the sent information using GET method.

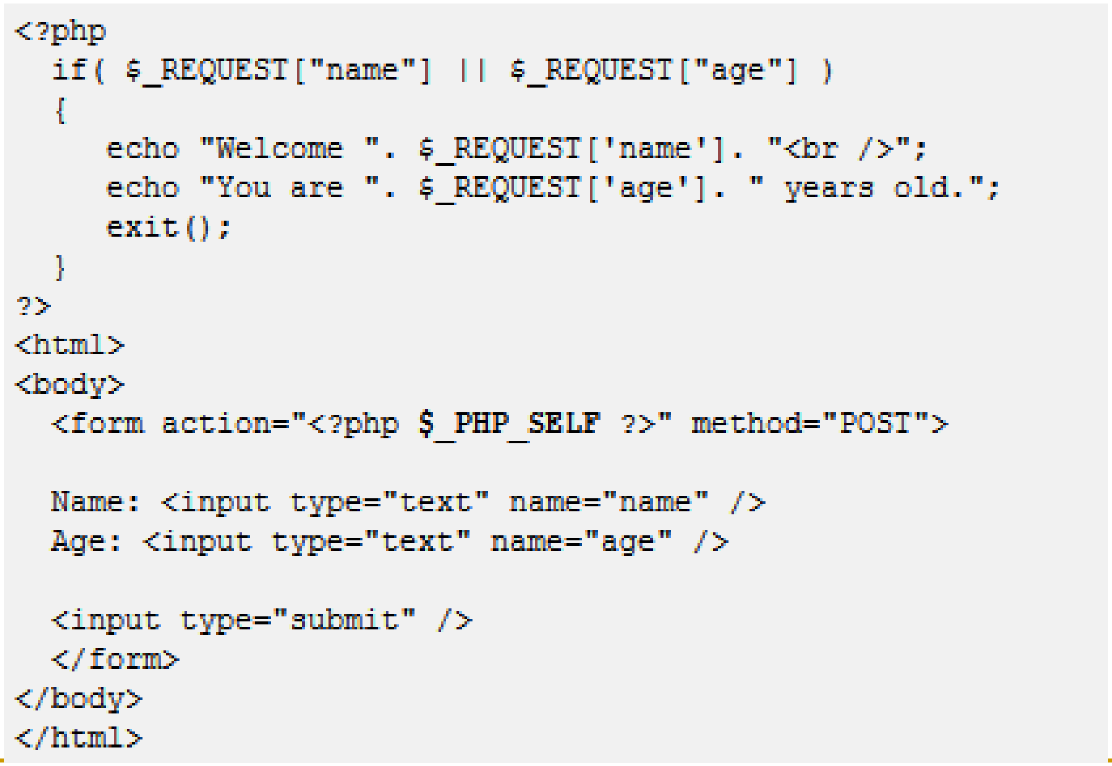
Example



```
<?php
  if( $_POST["name"] || $_POST["age"] )
  {
     echo "Welcome ". $_POST['name']. "<br />";
     echo "You are ". $_POST['age']. " years old.";
     exit();
  }
?>
<html>
<body>
  <form action="<?php $_PHP_SELF ?>" method="POST">

  Name: <input type="text" name="name" />
  Age: <input type="text" name="age" />

  <input type="submit" />
  </form>
</body>
</html>
```

The $_REQUEST variable

- The PHP $_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.



```
<?php
  if( $_REQUEST["name"] || $_REQUEST["age"] )
  {
     echo "Welcome ". $_REQUEST['name']. "<br />";
     echo "You are ". $_REQUEST['age']. " years old.";
     exit();
  }
?>
<html>
<body>
  <form action="<?php $_PHP_SELF ?>" method="POST">

  Name: <input type="text" name="name" />
  Age: <input type="text" name="age" />

  <input type="submit" />
  </form>
</body>
</html>
```

Finish

pens