

Desarrollo web con PHP



Plan your trip.

Travel planning



Paso de datos entre páginas





Tabla de contenido

Descripción material del programa	1
Mapa conceptual	3
1. Paso de datos entre páginas.....	4
Paso de datos por URL.....	4
Variables predefinidas \$_GET, \$_POST y \$_REQUEST.....	5
2. Transferencia de datos usando formularios.....	16
Referencias	29





Descripción material del programa

Este material está diseñado para facilitar el proceso de aprendizaje, por esta razón, los contenidos buscan que el aprendiz se apropie del conocimiento que realmente necesita para desarrollar sus habilidades y que lo haga de una forma sencilla y organizada; además de la lectura general cuenta con algunos apartes que contienen: frases o datos para recordar, segmentos de código, consejos y advertencias, estos elementos se destacan por las siguientes convenciones gráficas:

Ícono	Elemento importante
	Frases o datos para recordar: son extraídas de la lectura previa.
	Segmentos de código: pueden tomarse como base para los ejercicios propuestos.
	Consejos: buenas prácticas para el proceso de desarrollo.
	Advertencias: lo que no se recomienda hacer dentro de los procesos de desarrollo.

Fuente de imágenes: SENA





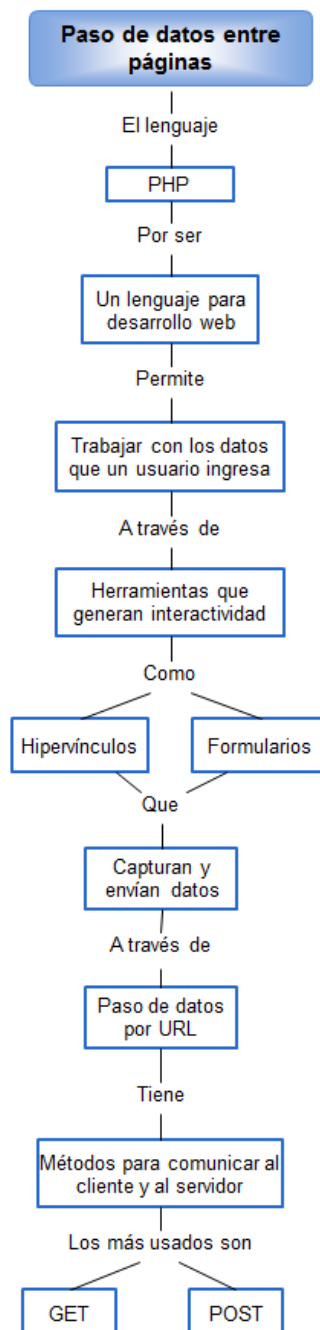
La mayor parte de los segmentos de código que aparecen en este material de formación los encuentran como archivos .php que se pueden descargar del material complementario en la siguiente ruta: Materiales del programa / Materiales de apoyo / Documentos complementarios / Documentos complementarios: Actividad de aprendizaje 4

Para usar estos segmentos de código solo se necesita copiarlos y pegarlos en el editor o entorno que esté usando para el desarrollo, o en el caso de que se encuentren etiquetados con la ruta del archivo puede abrirse directamente desde el editor o entorno. Los segmentos de código están comentados (usando los comentarios de cada lenguaje: HTML y PHP) para facilitar su comprensión y uso, dichos comentarios pueden ser modificados o retirados de ser necesario.



Mapa conceptual

En el mapa conceptual que se comparte a continuación, se evidencia la interrelación temática del contenido que se plantea en este material de formación:





1. Paso de datos entre páginas

PHP es un programa diseñado principalmente para el desarrollo web, por ello tiene herramientas muy completas para trabajar con los datos que un usuario genera al dar clic en los hipervínculos que se le presentan o al llenar los datos de un formulario.

Hasta el momento se ha trabajado con información contenida en variables, pero realmente no se ha generado una interacción con el usuario, ya que no se han tratado las herramientas que generan la interactividad entre la aplicación y el usuario, estas son: hipervínculos y formularios.

Paso de datos por URL

Antes de empezar es necesario recordar que el protocolo HTTP tiene varios métodos definidos para los procesos de comunicación entre el cliente y el servidor, dos de ellos son GET y POST: el método GET solicita recursos específicos al servidor usando una URL para identificar los recursos y el POST envía datos al servidor para que sean procesados por algún recurso (en este caso los datos se incluyen en el cuerpo de la petición y no en la URL). Aunque GET es básicamente un método para solicitar recursos, usando PHP se convierte en una forma de transferir datos al intérprete utilizando la URL.

Para esto es necesario construir un hipervínculo utilizando el objeto `<a>` de HTML cuyo atributo `href` contenga la URL a la que se hace la petición, adicionándole un símbolo de cierre de interrogación (?) después (sin dejar ningún espacio, por supuesto) se colocan las variables a transmitir de la forma `variable=valor` (nótese que no se usa el símbolo \$ para identificar las variables ya que esta forma de envío de datos no es exclusiva de PHP). Se separa con un símbolo ampersand (&) en caso de que haya varios pares `variable=valor`.

La sintaxis resultante sería la siguiente:

```
<a href = "<?php echo  
"urlReceptora.php?variable1=dato1&variable2=dato2";?>"> Texto  
hipervínculo </a>
```

También puede escribirse:





```
<?php          echo          "<a          href          =  
'urlReceptora.php?variable1=dato1&variable2=dato2';>          Texto  
hipervínculo </a>"?>
```

Aunque este método de transferencia de datos puede ser útil para solucionar ciertas necesidades pequeñas, como por ejemplo tener un hipervínculo para permitir al usuario editar cada elemento de una lista que quizás trajo de la base de datos, cada vez se desaconseja más su uso ya que la información que se envía queda expuesta fácilmente al usuario para que la puede ver en la URL, por lo tanto, no se recomienda para envío de información que se almacena en una base de datos pues da a los atacantes información sobre las estructura de la misma, además solo es recomendable para datos muy simples puesto que es complicado manejar espacios en la URL, por lo tanto enviar datos como un nombre completo sería algo engorroso.

Teniendo en cuenta que los métodos GET y POST requieren comunicación del servidor, es preciso que haya una solicitud y una respuesta, entonces es necesario recargar el servidor, ya sea para volver a abrir la misma página o para cargar la nueva página de destino (se pueden recibir los datos tanto en una página diferente a la original como en la misma página de origen de la petición), esto puede ser poco estético para el usuario pero si se usan herramientas como AJAX hace que el usuario no note las recargas y le parezca como si no hubieran recargas en ningún momento.

Variables predefinidas \$_GET, \$_POST y \$_REQUEST

Una vez que el servidor recibe la solicitud de un recurso .php y llama al intérprete de PHP para hacer el procesamiento, este último se encarga de almacenar los datos, ya sean de la URL o de un formulario en las variables predefinidas \$_GET, \$_POST o \$_REQUEST. \$_GET almacena los datos enviados por la URL o desde un formulario en el que se ponga como valor del atributo `method = "GET"`, \$_POST almacena los datos de un formulario en el que se ponga como valor del atributo `method = "POST"` y \$_REQUEST almacena todo lo contenido tanto en \$_GET como en \$_POST. Se recomienda que no se use la variable \$_REQUEST, sino que se utilicen \$_GET y \$_POST según sea el caso, sobre todo cuando se emplean formularios, de esta manera se evitan riesgos de inyección de código por URL.

Para acceder en la información que queda en los arreglos ya nombrados se utilizan índices, ya que estas variables son básicamente arreglos de PHP, por lo





tanto, si al enviar datos por URL se usó la dirección `almacenaDatos.php?nombre=Pablo`, la información se encontrará en `$_GET['nombre']` y en `$_REQUEST['nombre']` (se recomienda usar siempre las comillas simples para que PHP no intente interpretar los índices como constantes), o si en un formulario se tiene un control como el siguiente `<input type="text" name="color">` la información se encontrará en `$_POST['color']` y en `$_REQUEST['color']`. (The PHP Group, s.f.)



Fuente: SENA

Se recomienda que no se use la variable `$_REQUEST`, sino que se usen `$_GET` y `$_POST` según sea el caso, sobre todo cuando se manejan formularios, ya que de esta manera se evitan riesgos de inyección de código por URL.

En el siguiente ejemplo se muestra cómo se pasan los datos por URL:

Ejemplo 1 - Cambiar Color:



```
<!DOCTYPE html>
<html> <!-- En los archivos .php el código HTML se escribe
normalmente-->
  <head> <!-- como puedes ver toda la estructura de la página
está escrita en HTML-->
    <title>Ejemplo 1 - Cambia Color</title>
    <meta http-equiv="Content-Type"
      content="text/html; charset=ISO-8859-1" />
  </head>
  <?php
    /* En esta programa se van a usar los vínculos para pasar
    datos a través de
    * la URL con PHP, para este caso los datos se envían a la
    misma página
    */
    /* En la siguiente rutina se va a evaluar si la variable del
    sistem $_GET ya
    * contiene datos, si es así se construye la etiqueta de
    apertura del objeto
    * body de la página web con la información de color que haya
    llegado
    * por la URL, se debe recordar que la variable $_GET es la
    que
    * recibe los datos cuando se envían a través de una URL
    */
    if (isset($_GET['colorFondo'])) {
      /* Se almacena el dato que llega desde la URL en la
      variable
      * $colorFondo, el índice en el que llega el dato es el
```




```
nombre de la
    * variable que se puso en el hipervínculo
    */
$colorFondo = $_GET['colorFondo'];
echo "<body bgcolor = '$colorFondo'>";
/* Se eliminan los datos de la variable $_GET para que la
condición
    * anterior no se cumpla cuando se vuelva a cargar la
página
    */
unset($_GET);
/* Se construye un hipervínculo normal que devuelve al
usuario a la
    * página original, que en este caso es la misma pero se
va a cargar de
    * nuevo con el "menú" de colores
    */
echo "<a href = 'cambiaColor.php'>Volver al inicio</a>";
} else {
    /* Si la variable $_GET aún no se ha inicializado se
construye la
    * página con los hipervínculos necesarios para poder
pasar la información
    * cada hipervínculo lleva una variable con un valor
diferente, que
    * es la palabra del color en inglés para poder usarla al
construir el
    * objeto body en la parte anterior del condicional
    */
echo "<body>";
echo "<h3>Al dar clic en cualquiera de los siguientes
hipervínculos"
    . " podrás cambiar el color de fondo de la
página</h3>";
echo "<a href = 'cambiaColor.php?colorFondo=yellow'>Fondo
Amarillo</a>"
    . "<br />";
echo "<a href = 'cambiaColor.php?colorFondo=blue'>Fondo
Azul</a>"
    . "<br />";
echo "<a href = 'cambiaColor.php?colorFondo=red'>Fondo
Rojo</a>"
    . "<br />";
}
?>
</body>
</html>
```

Fuente: SENA

Descargue el segmento anterior del código como archivo .php del material complementario de este programa de formación en la siguiente ruta:

Materiales del programa / Materiales de apoyo / Documentos complementarios / Documentos complementarios: Actividad de aprendizaje 4 / Ejemplo 1 / Ejemplo 1 - Cambiar Color





En la siguiente figura se evidencia que al ejecutar el código anterior se obtienen los hipervínculos deseados, cada uno lleva una variable con un valor de color, puesto que la dirección del hipervínculo apunta al mismo recurso desde el que se parte la información llegará a la misma página, pero como el código contiene un condicional que cambia lo que se muestra si la variable `$_GET` está o no inicializada, parecería que son dos páginas distintas, en la figura se muestra además la barra de estado del navegador en la que puede verse la dirección a la que apunta el primer hipervínculo.



Figura 1. Ejecución del Ejemplo 1 - Cambiar Color

Fuente: SENA

Luego de dar clic en cualquier de los hipervínculos se observa lo que aparece en la Figura 2; por supuesto en cada caso con un color diferente en el fondo, aquí solo se muestra el de la primera opción que permite detallar en la barra de dirección del navegador cómo la variable y su valor se visualizan fácilmente por el usuario, esto genera inseguridad así que se recomienda usar el paso de datos por la URL solo cuando se tenga poco riesgo de facilitar un ataque al aplicativo. En algunos casos “para enviar datos un poco más sensibles” los desarrolladores utilizan encriptación de la información, lo cual puede minimizar un poco los riesgos en caso de que no se encuentre otra manera de enviar información importante.



Figura 2. Ejecución del Ejemplo 1 - Cambiar Color después de dar clic en el primer hipervínculo

Fuente: SENA

Ejercicio 1:

Utilice como base el ejemplo anterior y genere un programa que muestre hipervínculos con el nombre de 4 tipos de letra diferente y al dar clic en cualquiera de ellos muestre un mensaje con el tipo de letra seleccionado por el usuario.

En el Ejemplo 1 - Cambiar Color, los hipervínculos fueron escritos uno a uno de forma manual, pero usando las herramientas de PHP también se pueden crear de forma dinámica, en el siguiente ejemplo se van a crear dinámicamente diez hipervínculos, cada uno con su variable (la misma para todos) y con un valor numérico diferente usando un ciclo `for`.

Ejemplo 1 Seleccionar Número:

	<pre> <!DOCTYPE html> <html> <!-- En los archivos .php el código HTML se escribe normalmente--> <head> <!-- como puedes ver toda la estructura de la página está escrita en HTML--> <title>Ejemplo 1 Seleccionar Número</title> <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" /> </head> <body> <?php /* En este programa también se van a usar hipervínculos para enviar * información a la misma página, pero en este caso los hipervínculos se * van a construir dinámicamente usando un ciclo for */ if (isset(\$_GET['numeroSeleccionado'])) { \$numeroSeleccionado = \$_GET['numeroSeleccionado']; echo "<p>Usted ha seleccionado el número: </pre>
--	--



```
$numeroSeleccionado"
        . "<br />";
        unset($_GET);
        echo "<a href = 'seleccionarNumero.php'>Volver al
inicio</a>";
    } else {
        echo "<h3>Por favor seleccione uno de los
números</h3>";
        for ($i = 1; $i <= 10; $i++) {
            /* Como puede verse se usa el índice del ciclo for
para
            * construir los hipervínculos de forma dinámica
se van a crear
            * diez hipervínculos cada uno con un valor
numérico diferente
            * que se transmite en la variable
numeroSeleccionado
            */
            echo "<a href =
'seleccionarNumero.php?numeroSeleccionado=$i'>"
                . "$i</a><br />";
        }
    }
    ?>
</body>
</html>
```

Fuente: SENA

Descargue el segmento anterior del código como archivo .php del material complementario de este programa de formación en la siguiente ruta:

Materiales del programa / Materiales de apoyo / Documentos complementarios / Documentos complementarios: Actividad de aprendizaje 4 / Ejemplo 1 / Ejemplo 1 Seleccionar Número

Como en el Ejemplo 1 - Cambiar Color, la información se envía a la misma página, en la Figura 3 puede verse el resultado con los hipervínculos creados de manera dinámica y en la barra de estado se puede ver la dirección a la que apunta el primer hipervínculo.



Figura 3. Ejecución del Ejemplo 1 Seleccionar Número
Fuente: SENA

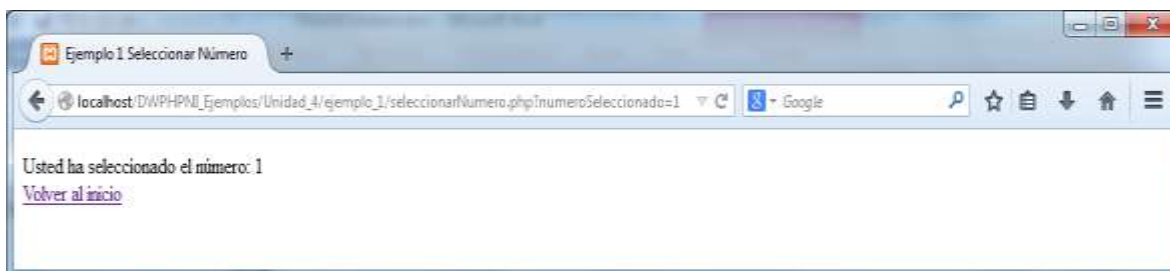



Figura 4. Ejecución del Ejemplo 1 Seleccionar Número luego de dar clic al primer hipervínculo
Fuente: SENA

A continuación se muestra el **Ejemplo 2 Seleccionar Número** en el que la información se envía realmente entre dos páginas distintas, lo cual no modifica en nada la forma en que se usan las herramientas para lograrlo, solo que una parte del código está en la página de origen y la otra parte en la página de destino; la rutina hace lo mismo que el segundo ejemplo, envía un valor numérico seleccionado desde un vínculo creado dinámicamente, ya no lo envía a la misma página sino a otra diferente que puede llamarse de destino, y una vez que el usuario, estando en la página de destino dé clic en el hipervínculo de “Volver al inicio” regresa a la página de origen.





Ejemplo 2 Seleccionar Número:

	<pre><!DOCTYPE html> <html> <!-- En los archivos .php el código HTML se escribe normalmente--> <head> <!-- como puedes ver toda la estructura de la página está escrita en HTML--> <title>Ejemplo 2 Seleccionar Número Pasar a Otra Página</title> <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" /> </head> <body> <?php /* En este programa, así como en el segundo archivo del ejemplo * anterior, se va a crear una lista de hipervínculos generados por * un ciclo for, la diferencia es que en este caso van a haber dos * páginas, una que es esta en la que se crea el "menú", y la otra * (recibeNumero.php) en la que se recibe la información */ echo "<h3>Por favor seleccione uno de los números</h3>"; for (\$i = 1; \$i <= 10; \$i++) { echo "" . "\$i
"; } ?> </body> </html></pre>
---	--

Fuente: SENA

Descargue el segmento anterior del código como archivo .php del material complementario de este programa de formación en la siguiente ruta:

Materiales del programa / Materiales de apoyo / Documentos complementarios / Documentos complementarios: Actividad de aprendizaje 4 / Ejemplo 2 / Ejemplo 2 Seleccionar Número



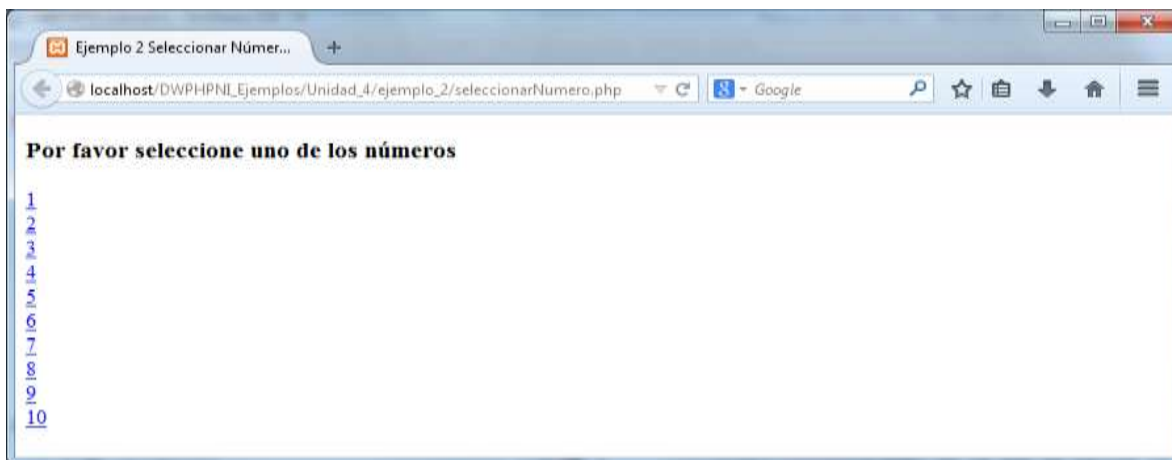


Figura 5. Ejecución del Ejemplo 2 Seleccionar Número
Fuente: SENA

El siguiente es el código que corresponde al Ejemplo 2 Recibe Número, esta página recibe como respuesta a la petición hecha en la página anterior, por lo tanto, aparece justo después de que se dé clic en el hipervínculo del número seleccionado:

Ejemplo 2 Recibe Número:

	<pre> <!DOCTYPE html> <html> <!-- En los archivos .php el código HTML se escribe normalmente--> <head> <!-- como puede ver toda la estructura de la página está escrita en HTML--> <title>Ejemplo 2 Recibe Número Desde Otra Página</title> <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" /> </head> <body> <?php /* Esta es la página de destino del programa que permite enviar * un número, seleccionado en hipervínculos, a través de la URL */ if (isset(\$_GET['numeroSeleccionado'])) { \$numeroSeleccionado = \$_GET['numeroSeleccionado']; echo "<p>Usted ha seleccionado el número: \$numeroSeleccionado" . "
"; unset(\$_GET); } else { /* A diferencia del otro ejemplo que hace lo mismo que este cuando </pre>
--	---



	<pre>* la variable \$_GET no se ha inicializado se muestra un mensaje * indicando que no se ha seleccionado un número, esto es útil * cuando el usuario pone la dirección de una página directamente * en la barra de dirección del navegador. También existen herramientas * que permiten redireccionar automáticamente al usuario si entra * a páginas por vías diferentes a las que se desea */ echo "No se ha seleccionado ning&uacute;n &uacute;mero
"; } echo "Volver al inicio"; ?> </body> </html></pre>
--	--

Fuente: SENA

Descargue el segmento anterior del código como archivo .php del material complementario de este programa de formación en la siguiente ruta:

Materiales del programa / Materiales de apoyo / Documentos complementarios / Documentos complementarios: Actividad de aprendizaje 4 / Ejemplo 2 / Ejemplo 2 Recibe Número

En la Figura 6 se observa lo que sucede cuando el usuario da clic en el primer hipervínculo de la página anterior, como se ve, la dirección es distinta porque se está solicitando un recurso diferente a aquel en el que se estaba, pero de todas maneras en la barra de dirección se nota fácilmente la variable y su valor.

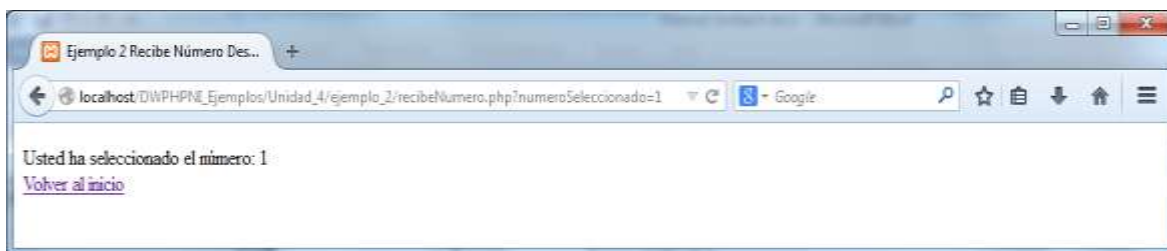


Figura 5. Segunda página del Ejemplo 2 Recibe Número que aparece al dar clic en uno de los vínculos de la página anterior

Fuente: SENA

Ahora se muestra lo que sucede si el usuario no llega a la segunda página (Ejemplo 2 Recibe Número) usando los hipervínculos de la primera (Ejemplo 2





Seleccionar Número), sino que lo hace poniendo él mismo la dirección, pero sin poner la adición para PHP de la variable y el valor, es decir que ubicaría la dirección:

```
http://localhost/DWPHPNI_Ejemplos/Unidad_4/ejemplo_2/recibeNumero.php
```



Figura 6. Segunda página del Ejemplo 2 Recibe Número cuando el usuario no da clic en los hipervínculos de la primera sino que escribe la dirección en el navegador

Fuente: SENA

Finalmente, en la Figura 8 se observa lo que sucedería si el usuario de nuevo escribe la dirección en el navegador, pero esta vez si pone la información de la variable y el valor en la URL solo que, para efectos académicos, se ve que el usuario colocar un valor diferente a los que se permitían en los hipervínculos.

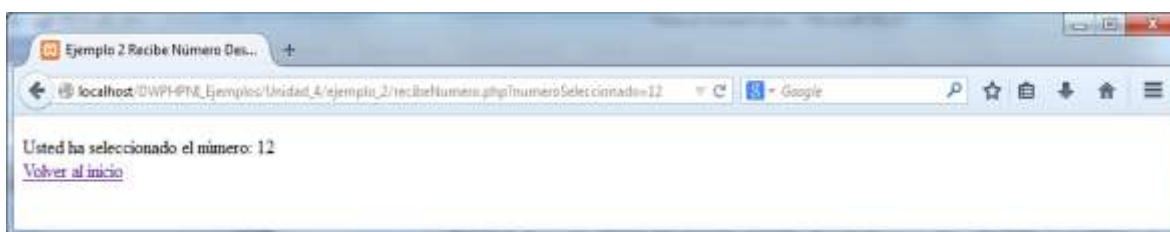


Figura 7. Segunda página del Ejemplo 2 Recibe Número, cuando el usuario no da clic en los hipervínculos de la primera sino que escribe la dirección en el navegador poniendo la variable con un valor diferente al de los hipervínculos


Fuente: SENA

Nótese que en este caso la ejecución no genera ningún error, debido a que el sistema simplemente muestra el valor que se recibió, incluso el usuario no ubicó un número sino la palabra “carro” como valor de la variable numeroSeleccionado, y el sistema simplemente hubiese impreso “Usted ha



seleccionado el número: carro”, lo cual sería un error de lógica pero no generaría un fallo del sistema (cuando menos no en este caso), pero si en lugar de simplemente imprimir el valor el sistema requiere de la información enviada para hacer otros procesos, como por ejemplo un código de producto para buscar el producto en una base de datos, entonces el hecho de tener datos erróneos podría generar un error severo.

Por las razones anteriores a la hora de empezar a interactuar con información externa (aunque no puede obviarse totalmente cuando se trabaja con información interna), ya sea enviada por URL, por formulario, recibida desde un dispositivo, otra aplicación o algún repositorio de datos como un manejador de bases de datos, se piensa primero en la validación de los datos recibidos, esto se logra principalmente con técnicas lógicas y manejo de errores (que PHP ya soporta), de esta manera las aplicaciones creadas funcionarán con mayor nivel de estabilidad.

 <p>Fuente: SENA</p>	<p>A la hora de interactuar con información externa (aunque no puede obviarse totalmente cuando se trabaja con información interna), ya sea enviada por URL, por formulario, recibida desde un dispositivo, otra aplicación o algún repositorio de datos como un manejador de bases de datos, se debe empezar a pensar en la validación de los datos recibidos.</p>
---	---

Ejercicio 2:

Cree un programa que por medio de hipervínculos envíe 2 datos y sean procesados en una página de destino diferente a la de origen.

2. Transferencia de datos usando formularios

Para que una aplicación web tenga total interacción con el usuario es necesario que permita recepción de información más compleja que la simple selección de un hipervínculo, por esta razón se hace totalmente necesario que exista la posibilidad de gestionar los controles de los formularios de HTML, un formulario es básicamente una colección de diferentes controles que permiten recibir distintos tipos de información, los objetos mediante los que HTML construye dichos controles son `<input>`, `<textarea>` y `<select>`, con estos objetos se construyen cajas de texto, cajas de selección múltiple, cajas de chequeo y combos de opciones desplegables. Al respecto HTML5 tiene nuevas opciones con base en



el objeto `<input>` que permiten la validación de los datos antes de que sean transmitidos al servidor.

Como ya se dijo antes el uso de la variable predefinida `$_REQUEST` no es recomendable, y en el caso de envío de información mediante formularios se sugiere uso absoluto del método POST en los atributos del formulario y la variable `$_POST` para recoger los datos, de esta forma se evitan que se puedan generar ataques por inyección de código en la URL, además se sugiere el uso de rutinas de validación de datos. Los controles HTML5 con prevalidación aún no son una solución generalizada, ya que no todos los navegadores soportan los nuevos controles.

Para tomar los datos desde la variable `$_POST` simplemente se pone como índice de dicho arreglo el nombre que se le dio al control en el formulario HTML mediante el parámetro name, por ejemplo si se tiene un control como el siguiente:

`<input type = "text" name = "apellidos">` en la página de destino (que puede ser o no la misma de origen) se tendrá acceso a la información en: `$_POST['apellidos']`.



Fuente: SENA

Para que una aplicación web tenga total interacción con el usuario es necesario que se pueda recibir información más compleja que la simple selección de un hipervínculo, por esta razón se hace totalmente necesario que exista la posibilidad de gestionar los controles de los formularios de HTML.

En el ejemplo que se presenta a continuación, `formularioHTML.php`, se tiene un formulario con diferentes tipos de controles, los cuales son recibidos en la página de destino `recibeDatos.php`.



Ejemplo 3: formularioHTML

	<pre> <!DOCTYPE html> <html> <head> <title>Unidad 4 - Ejemplo 3 Formulario con diferentes Controles</title> <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" /> <!-- Para la presentación de este formulario no se requiere de código PHP solo HTML, se usó una plantilla CSS gratuita descargada de internet que se vincula en la siguiente línea.--> <link rel="stylesheet" href="css/style.css"> </head> <body> <section class="container"> <div class="login"> <h1>POR FAVOR LLENE ESTE FORMULARIO</h1> <!-- Se crea un formulario con diferentes tipos de controles para que el usuario pueda diligenciar los datos solicitados los datos serán enviados a una página distinta en la cual pueden ser procesados, puede ver que el método usado es POST y el argumento action tiene el valor "recibeDatos.php" que es la página que va a recibir y procesar los datos del formulario--> <form method="post" action="recibeDatos.php"> <p> <label> Nombre <input type="text" name="nombre" placeholder="Nombre Completo" autofocus required> </label> </p> <p> <label> Fecha de Nacimiento <input type="date" name="fecha_nacimiento" placeholder="Fecha de Nacimiento"> </label> </p> <p> <label> Tel&eacute;fono <input type="text" name="telefono" placeholder="Tel&eacute;fono"> </label> </p> <p> <label> </pre>
--	---



	<pre> e-mail <input type="mail" name="email" placeholder="e-mail"> </label> </p> <p> <label> Estado Civil</label>
 <label> <input type="radio" name="estadoCivil" value="Casado"> Casado </label> <label> <input type="radio" name="estadoCivil" value="Soltero"> Soltero </label> <label> <input type="radio" name="estadoCivil" value="Unión Libre"> Unión Libre </label> </p> <p> <label> Departamento <select name="departamento"> <option value ="Amazonas">Amazonas</option> <option value ="Antioquia">Antioquia</option> <option value ="Arauca">Arauca</option> <option value ="Atlántico"> Atlántico </option> <option value ="Bolívar">Bolívar</option> <option value ="Boyacá">Boyacá</option> <option value ="Caldas">Caldas</option> <option value ="Caquetá">Caquetá</option> <option value ="Casanare">Casanare</option> <option value ="Cauca">Cauca</option> <option value ="Cesar">Cesar</option> <option value ="Chocó">Chocó</option> <option value ="Córdoba">Córdoba</option> <option value ="Cundinamarca"> Cundinamarca </option> <option value ="Guainia">Guainia</option> <option value ="Guaviare">Guaviare</option> <option value </pre>
--	--



	<pre> ="Huila">Huila</option> Guajira</option> ="Magdalena">Magdalena</option> ="Meta">Meta</option> ="Nariño">Nari&ntilde;o</option> Santander"> Norte de Santander </option> ="Putumayo">Putumayo</option> ="Quindio">Quind&iacute;o</option> ="Risaralda">Risaralda</option> Providencia"> San Andr&eacute;s y Providencia </option> ="Santander">Santander</option> ="Sucre">Sucre</option> ="Tolima">Tolima</option> <option value ="Valle del Cauca"> Valle del Cauca </option> ="Vaupés">Vaup&eacute;s</option> ="Vichada">Vichada</option> </select> </label> </p> <p> <label> <input type="checkbox" name="hijos"> &quest;Tiene hijos? </label> </p> </p> <input type="hidden" name="datoOculto" value="Esta información esta oculta al usuario"> <p class="submit"> <input type="submit" > </p> </form> </div> </section> </body> </html> </pre>
--	--

Fuente: SENA



Descargue el segmento anterior del código como archivo .php del material complementario de este programa de formación en la siguiente ruta:

Materiales del programa / Materiales de apoyo / Documentos complementarios / Documentos complementarios: Actividad de aprendizaje 4 / Ejemplo 3 (en esta carpeta comprimida encuentra el archivo **formularioHTML**).

Ejemplo 3: recibeDatos

	<pre><!DOCTYPE html> <html> <head> <title>Unidad 4 - Ejemplo 2 Recibe Datos Desde Otra P&aacute;gina</title> <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" /> </head> <body> <?php /* En esta página se reciben los datos, como puede verse solo se usa * la función print_r() para volcar al navegador el arreglo \$_POST * que contiene todos los datos del formulario de la página de origen * la idea es que usted como aprendiz complete el código de esta página * para procesar los datos recibidos según lo especificado en el * ejercicio propuesto en el material de estudio. */ print_r(\$_POST); <?> </body> </html></pre>
--	---

Fuente: SENA

Descargue el segmento anterior del código como archivo .php del material complementario de este programa de formación en la siguiente ruta:

Materiales del programa / Materiales de apoyo / Documentos complementarios / Documentos complementarios: Actividad de aprendizaje 4 / Ejemplo 3 (en esta carpeta comprimida encuentra el archivo **recibeDatos**).



Unidad 4 - Ejemplo 3 Formular...

localhost/DWPHPNL_Ejemplos/Unidad_4/ejemplo_3/formularioHTML.php

Google

POR FAVOR LLENE ESTE FORMULARIO

Nombre

Fecha de Nacimiento

Teléfono

e-mail

Estado Civil
☐ Casado ☐ Soltero ☐ Unión Libre

Departamento

☐ ¿Tiene hijos?

Enviar consulta

Figura 8. Ejecución del Ejemplo 3: formularioHTML
Fuente: SENA

Unidad 4 - Ejemplo 2 Recibe D...

localhost/DWPHPNL_Ejemplos/Unidad_4/ejemplo_3/recibeDatos.php

Google

Array ([nombre] => Juan Perez [fecha_nacimiento] => 25/03/1979 [telefono] => 3214567 [email] => jperez@hotmail.com [estadoCivil] => Soltero [departamento] => Guaviare [datoOculto] => Esta informaci3n esta oculta al usuario)

Figura 9. Ejecución del Ejemplo 3: recibeDatos
Fuente: SENA

Ejercicio 3:

Con base en el ejemplo anterior modifique el Ejemplo 3: recibeDatos, para que presente un mensaje en el que se salude al usuario utilizando los datos que puso en el formulario, diciéndole su edad con base en la información puesta en la fecha de nacimiento, la parte del final del mensaje debe decir “Esperamos que sus hijos



estén muy bien” solo en caso de que haya chequeado la caja ¿tiene hijos?, en esto último debe validarse que si no está chequeada la opción no salga ningún aviso de error del intérprete.

En el Ejemplo 4: Login que se muestra a continuación, se tiene un formulario de login y la información que se incluya será evaluada por la página que recibe los datos del formulario, dicha página no permitirá el acceso en caso de que se llegue a ella sin pasar por el login, también validará que el usuario y la contraseña coincidan con los datos almacenados en el sistema para dar acceso, de lo contrario lo negará y devolverá al usuario de la aplicación hasta la página de login.

Ejemplo 4: Login



```

<!DOCTYPE html>
<html>
  <head>
    <title>Unidad 4 - Ejemplo 4 Login</title>
    <meta http-equiv="Content-Type"
      content="text/html; charset=ISO-8859-1" />
    <!-- En este archivo se genera un formulario de login
básico
se usa la misma plantilla CSS del ejemplo anterior pero
esta
solo modifica la presentación de nuevo los datos serán
enviados
a otra página para que valide la información, por ahora
dicha validación se hará comparándola con información
contenida
en variables, pero con base en esta rutina se pueden crear
puntos de acceso a aplicaciones que usen información
contenida en base de datos para dar o negar el acceso-->
    <link rel="stylesheet" href="css/style.css">
  </head>
  <body>
    <section class="container">
      <div class="login">
        <h1>Login</h1>
        <form method="post" action="validaLogin.php">
          <p><input type="text" name="usuario"
            placeholder="Nombre de Usuario"></p>
          <p><input type="password" name="contrasena"
            placeholder="Contraseña"></p>
          <p class="remember_me">
            <label>
              <input type="checkbox"
name="recordarme"
              value="false">
              Recordar mis datos
            </label>
          </p>
          <p class="submit"><input type="submit"
value="Ingresar"></p>
        </form>
      </div>
    </section>

```



	<pre></body> </html></pre>
--	--

Fuente: SENA

Descargue el segmento anterior del código como archivo .php del material complementario de este programa de formación en la siguiente ruta:

Materiales del programa / Materiales de apoyo / Documentos complementarios / Documentos complementarios: Actividad de aprendizaje 4 / Ejemplo 4 (en esta carpeta comprimida encuentra el archivo **Login**).

Ejemplo 4: validarLogin

	<pre><?php /* En este ejemplo lo primero que se hace es validar que se haya llegado * a esta página luego de hacer un intento de login, ya que de lo contrario * se podría tratar de un intento de forzar el acceso a la aplicación, esto * puede lograrse de varias maneras, la que aquí se usa es validar si el * arreglo \$_POST y más específicamente el índice 'usuario' ya fue inicializado * si un usuario intentara llegar a esta página inyectando código por la URL * sería rechazado ya que la información llegaría al arreglo \$_REQUEST o al * arreglo \$_GET */ if (!isset(\$_POST['usuario'])) { ?> <!DOCTYPE html> <html> <head> <title>Unidad 4 - Ejemplo 4 Error de ingreso</title> <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" /> </head> <body> <?php /* Si la variable \$_POST['usuario'] no está inicializada se muestra un * aviso al usuario de que no ha hecho login correctamente, es decir, que * no llegó a esta página habiendo hecho login, y que será redireccionado * a la página de login */ echo "No se ha hecho login correctamente, ser&aacute; redireccionado a " . "la p&aacute;gina de login
"; echo "Si no es redireccionado de clic aquí"; /* la función header() permite el reenvío de los encabezados de una</pre>
--	--



```

* página o recurso web, lo que hace que pueda ser recargado
como en este
* usando una cadena que refresca el navegador, pero permite
esperar un
* determinado número de segundos durante los cuales se puede
mostrar el
* mensaje que está arriba.
*/
header("Refresh: 5; URL=Login.php");
} else {
?>

<!DOCTYPE html>
<html>
  <head>
    <title>Unidad 4 - Ejemplo 4 Valida Login</title>
    <meta http-equiv="Content-Type"
          content="text/html; charset=ISO-8859-1" />
  </head>
  <body>
    <?php
    /* Si por el contrario el usuario llega a esta página
luego de hacer
    * un intento de login, se validan los datos que puso
en el
    * formulario si son iguales que lo contenido en las
variables
    * $usuario y $contrasena se da acceso, si no se da un
mensaje que
    * indica que los datos no coinciden con los
almacenados y se
    * devuelve al usuario a la página de login. Cuando se
estudie
    * el tema de bases de datos en los siguientes niveles
del curso
    * se podrá utilizar esta base para comparar los datos
del formulario
    * con los contenidos en la base de datos, y así mismo
    * al usar sesiones se puede hacer que la próxima vez
que el usuario
    * intente ingresar a la aplicación no tenga que
volver a poner
    * los datos de login, esto siempre y cuando se tengan
las
    * garantías de seguridad necesarias
    */
$usuario = "Juan";
$contrasena = "Pablo";
if (($_POST['usuario'] != $usuario) or
    ($_POST['contrasena'] != $contrasena)) {
  echo "El usuario o la contrase&ntilde;a son
incorrectos "
  . "ser&aacute; redireccionado a la p&aacute;gina
de login<br />";
  echo "Si no es redireccionado de clic "
  . "<a href = 'login.php'>aqui</a>";
  header("Refresh: 5; URL=Login.php");
} else {
  echo "<h1>Bienvenid@</h1>";
  echo "Su usuario y contrase&ntilde;a son correctos
<br />";
  if (isset($_POST['recordarme'])) {

```





```
if ($_POST['recordarme'] == TRUE) {
    echo "Usted quiere que se recuerden su
datos";
}
}
}
?>
</body>
</html>
```

Fuente: SENA

Descargue el segmento anterior del código como archivo .php del material complementario de este programa de formación en la siguiente ruta:

Materiales del programa / Materiales de apoyo / Documentos complementarios / Documentos complementarios: Actividad de aprendizaje 4 / Ejemplo 4 (en esta carpeta comprimida encuentra el archivo **validarLogin**).

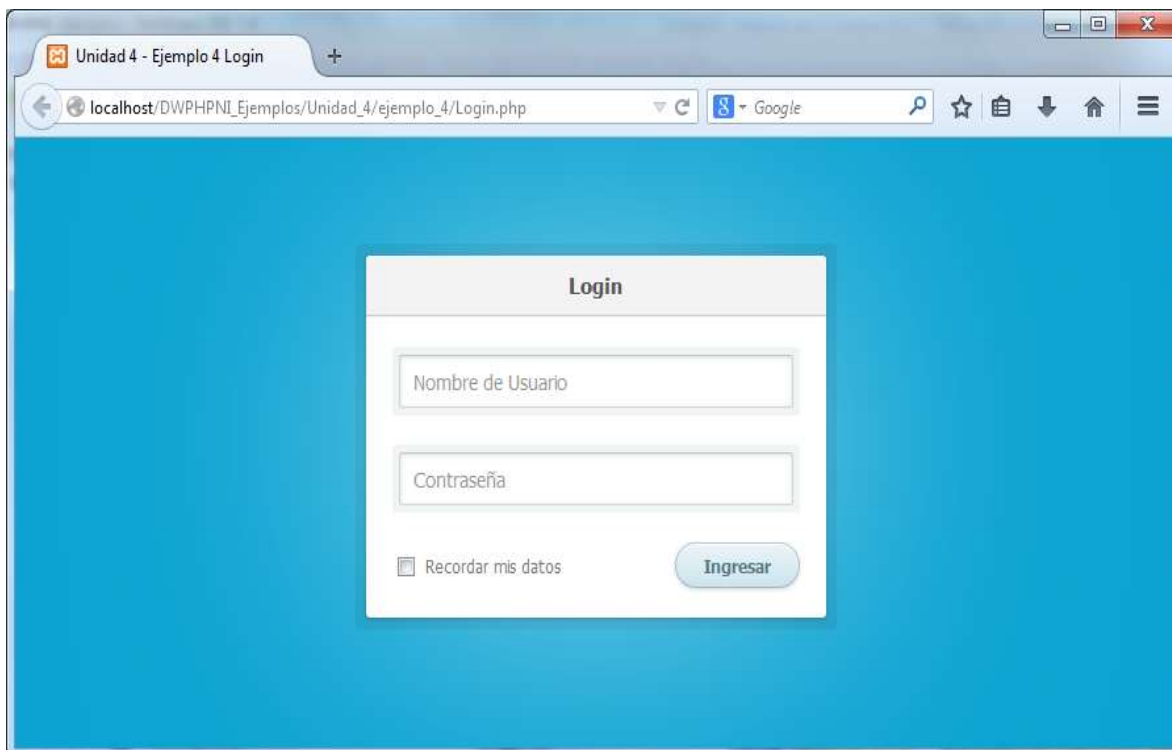


Figura 10. Ejecución del Ejemplo 4: validarLogin

Fuente: SENA



Figura 11. Ejecución del Ejemplo 4: validarLogin cuando ingresa a la página sin pasar por Login
Fuente: SENA



Figura 12. Ejecución del Ejemplo 4: validarLogin cuando en el Login se pone datos de usuario y contraseña que no coinciden con los almacenados
Fuente: SENA

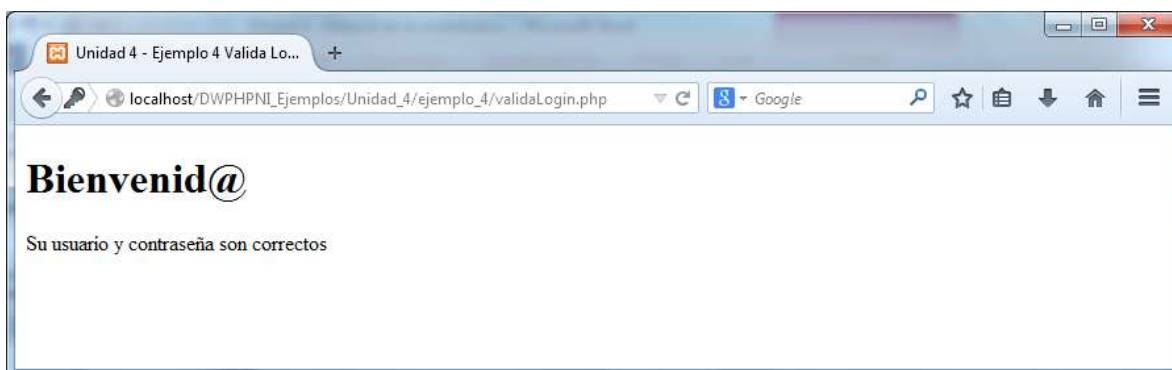


Figura 13. Ejecución del Ejemplo 4: validarLogin cuando los datos de acceso son correctos
Fuente: SENA





Con el propósito de poner en práctica los conocimientos adquiridos a través de este material de formación, consulte la guía de aprendizaje y realice todas las evidencias propuestas en ella.

Para acceder a la guía y a las evidencias diríjase al botón: Actividades / Actividad de aprendizaje 4



Referencias

The PHP Group. (s.f.). *Manual de PHP*. Consultado el 30 de junio de 2015, en <http://www.php.net/manual/es/index.php>

Control del documento

	Nombre	Cargo	Dependencia	Fecha
Autor	Jorge Luis Ballesteros Vargas	Instructor	Centro Metalmecánico Regional Distrito Capital	Diciembre de 2014
Adaptación	Paola Andrea Bobadilla Gutiérrez	Guionista - Línea de producción	Centro Agroindustrial Regional Quindío	Junio de 2015