

# Tecnicatura Superior en Análisis de Sistemas

## PROGRAMACIÓN V

### ***Tema 8- Archivos externos. Sesiones.***

Una forma práctica y prolija de trabajar en PHP, es la inclusión de archivos externos. Generalmente nuestros programas van a ser más largos de los que estamos haciendo ahora, y seguramente nuestro código se extienda mucho más. Para que esto no suceda, podemos utilizar otros archivos, que pueden contener funciones y procedimientos, y llamarlos solamente cuando necesitemos utilizarlos.

Supongamos por ejemplo que necesitamos crear una base de datos con varias tablas, seguramente ese esquema de datos lo vamos a utilizarlo varias veces, no solamente una vez, si necesitamos hacer un insert, mostrar datos de una o varias formas, etc...

Entonces cada vez que hacemos el llamado a la base de datos, tendríamos que escribir el código de conexión a la misma. Si la llamamos 800 veces tendríamos que escribir la conexión 800 veces. Para evitar esto se utilizan **archivos externos**, para que en vez de escribir la conexión 800 veces, la escribamos **solamente una vez** y cada vez que la necesitemos llamemos al archivo de conexión. Esto nos ahorra tiempo, código y queda mucho más prolijo y entendible.

Otro ejemplo, puede ser algo muy común y hasta diría obligatorio en las páginas web, como lo son los pies de página (**footer** en inglés), cada vez que hacemos una página nueva de nuestro sitio web debemos agregarle el pie de página, y esto sería repetir código. Entonces ¿por qué no dejar el footer en un archivo y cada vez que lo necesite sólo llamarlo? Bueno, también para esto se utilizan archivos externos.

En PHP hay varias formas de llamar a archivos externos:

- **Include.**
- **Include\_once.**
- **Require.**
- **Require\_once.**

La sentencia **include** incluye y evalúa el archivo especificado, para que esto suceda nuestro archivo debe tener la extensión **.php**. Y para incluir el archivo debemos hacerlo de la siguiente manera:

```
include 'inc/conexion.php' ;
```

En este ejemplo anterior, estamos llamando a un archivo que se encuentra en el directorio **inc** y el mismo se llama **conexion.php**.

Este archivo debería tener la conexión a la base de datos, de esta manera siempre que necesitemos conectar a la base, solamente hacemos el llamado a este archivo (conexion.php).

### **Ejemplo práctico:**

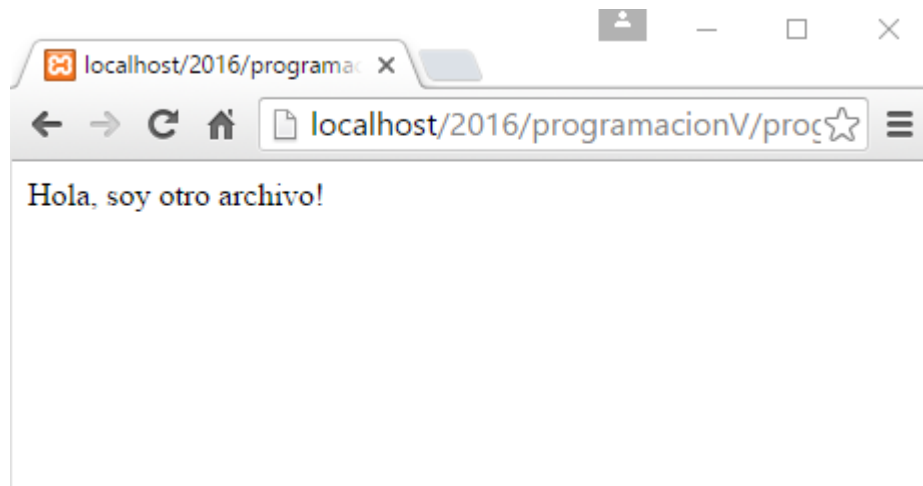
index.php

```
<?php
    include 'mensaje.php';
?>
```

mensaje.php

```
<?php
    echo 'Hola, soy otro archivo!';
?>
```

Resultado en el navegador:



De esta manera, estamos llamando a un archivo dentro de nuestro servidor que contiene un mensaje.

La sentencia **include\_once**, es muy parecida a la función **include**, nada más que esta, incluye el archivo sólo una vez, es decir si el archivo ya fue incluido y es vuelto a ser llamado, no lo volverá a incluir. Como su nombre lo indica, lo incluye una vez.

**include\_once** se puede utilizar en casos donde el mismo fichero podría ser incluido y evaluado más de una vez durante una ejecución particular de un script, así que en este caso, puede ser de ayuda para evitar problemas como la redefinición de funciones, reasignación de valores de variables, etc.

Y por último la sentencia **require**, es también similar a la función **include**, nada más que en caso de producirse una excepción o fallo producirá un error fatal, es decir se detiene el script mientras que el **include** sólo emitirá una advertencia. Por ejemplo si cargamos un archivo de conexión a base de datos y por alguna razón este archivo no

es encontrado, el script no se ejecutará, se detendrá, mostrará un mensaje de **error fatal**.

Con **require\_once** pasa exactamente lo mismo que con `include_once`, incluye el archivo una sola vez, pero si este no está disponible nos daría un error del tipo fatal; es decir no se ejecutaría el script.

### ¿Cuántos **include** podemos utilizar por programa?

La respuesta es simple: Todos los que queramos y donde queramos. Dentro de una página PHP podemos hacer múltiples llamadas a distintos archivos a través de **include**, incluso, podemos llamar a un mismo archivo varias veces en distintos momentos de la ejecución.

El lugar de la ejecución es simplemente donde necesitemos hacer la inclusión. Si necesitamos incluir al principio, lo hacemos al principio, si es al medio o dentro de un ciclo o dentro de una iteración condicional, lo hacemos ahí. No hay límites con respecto a eso. Además, los **include** pueden ser anidados, es decir, podemos hacer una llamada a una página que a su vez realice una llamada a otra página.

Los **include** son muy útiles, no solamente para guardar funciones en los mismos, sino también para maquetar una página web, con la ayuda de estilos **CSS** (recordemos que el archivo tiene que tener la extensión .php).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Incluir archivos</title>
</head>
<body>
    <?php
        include 'inc/head.php';
        include 'inc/detail.php';
        include 'inc/footer.php'
    ?>
</body>
</html>
```

Generalmente estos tres documentos .php, son los que comúnmente se repiten en una página web. La cabecera, la parte del medio y el pie de página. Y como podemos ver en el código de ejemplo anterior están guardados en un directorio llamado **inc**. De esta manera no tenemos que repetir cada página cada vez que lo necesitamos, sino que simplemente hacemos el llamado a ese archivo.

## Sesiones

Muchas veces nos sucede que necesitamos pasar una variable entre varias páginas, para hacer esto debemos crear sesiones, o mejor dicho variables de sesión. Hasta ahora hemos visto que creando un formulario en HTML, el usuario ingresaba un valor y ese valor podía ser procesado en el servidor, pero ¿qué pasaría si yo quiero guardar ese valor y utilizarlo más adelante en otra página sin utilizar base de datos? La respuesta es utilizando **sesiones**.

Una sesión es una variable que se crea en el servidor y esta variable puede ejecutarse sin que el usuario de la Web tenga conocimiento alguno de ello y la misma puede utilizarse en cualquier momento y lugar de nuestro sitio web. El ejemplo clásico de sesiones, es el carrito de compras de una página web.

En una tienda de libros, un usuario inicia sesión (puede ser con usuario y contraseña), busca en la web varios libros y a medida que va buscando elige algunos para posteriormente adquirirlos, para esto el usuario, pasa por varias páginas, “novelas”, “drama”, “tecnología” entre otras y va eligiendo en cada una de ellas lo que quiere comprar. Al finalizar su búsqueda, este usuario eligió comprar 7 libros, se dirige a la sección de compras de la página web, la misma le realiza la suma del total a pagar, ingresa sus datos de envío del producto y finaliza la operación.

Para nosotros los programadores, este usuario del ejemplo anterior, cada libro que elige es guardado en una variable de sesión, para nosotros no son libros, sino variables que va generando el usuario a medida que navega por nuestra página web, y que vamos pasando de página en página hasta que el usuario decide finalizar la sesión y comprar o no el producto. Es lógico que cada sesión tenga una duración si la página web esta inactiva, esta sesión se destruye sola y el usuario si quiere comprar va a tener que loguearse nuevamente para comenzar el proceso de compra.

Pero ¿en qué se diferencian las sesiones de los métodos **POST** y **GET**? Los métodos **POST** y **GET** permiten que los usuarios asignen valores a variables, y también permiten que la propia página tome valores internos, por ejemplo de una base de datos, y opere con ellos. Los valores emitidos por **POST** y **GET** pueden tener un origen conocido para el usuario y también su destino puede ser. De cualquier forma, el usuario podrá tener conocimiento de las variables que se envían y de sus valores. Pues bien, con las sesiones podemos hacer lo mismo, pero con una diferencia, la variable de sesión será recuperable en cualquier parte del sitio Web sin tener que crear enlaces de pasos de variable o formularios con métodos **GET** o **POST**, por lo que el usuario no sabrá ni cuando se crea (aunque lo pueda intuir) ni donde o cuando se recupera y ejecuta la variable. Simplemente cuando pasas de una página a otra mediante un enlace normal y corriente, tendrás la variable disponible para usarla.

Veamos un ejemplo práctico para entender un poco más el tema:

*Un usuario ingresa su nombre, luego pasa a otra página donde nos muestra su nombre y elige una bebida entre varias, para finalizar pasa a otra página donde se muestra el nombre, y la bebida que eligió.*

Para este ejemplo voy a utilizar 3 archivos **.php**, el primer archivo podría ser **.html**, eso como más les guste, **pero si o si los siguientes 2 deben tener extensión .php**.

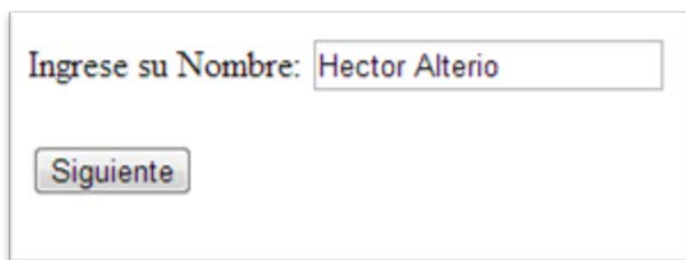
**index.php o index.html**

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Encuesta</title>
</head>
    <body>
        <form action="2.php" method="POST">
            <label>Ingrese su Nombre: </label>
            <input type="text"
name="nombre"><br><br>
            <input type="submit"
value="Siguiente">
        </form>
    </body>
</html>

```

Hasta ahora es todo conocido, un formulario con un **label** y un input del tipo **text**, acompañados de un **botón** "Siguiente":



**1.php**

```

<?php
    SESSION_START( );

    $nombre = $_POST['nombre'];
    echo "Mi nombre por POST es:
".$nombre."<br><br>";
    $_SESSION['nombre'] = $nombre;
    echo "Mi nombre por SESSION es:
".$_SESSION['nombre']."<br><br>";

    echo '
<form action="3.php" method="POST" >
    <select name="bebidas">
        <label>Seleccione la bebida: </label>
        <option>Gaseosa</option>

```

```
        <option>Vino</option>
        <option>Cerveza</option>
    </select>
    <input type="submit" value="Siguiente">
</form><br>
<a href="1.php">volver</a>
';
?>
```

En este archivo tenemos que tener en cuenta varias cosas... para comenzar a utilizar variables de sesión, siempre delante de todo en la cabecera del archivo debemos escribir:

```
SESSION_START( );
```

Esto hace que el intérprete, sepa que vamos a utilizar variables de sesión. Sin esa función por más que agreguemos variables de sesión el sistema no las va a reconocer como tales. Esta función deberá estar al comienzo de todas las páginas que manejemos variables de sesión.

Luego para guardar una variable de sesión hacemos lo siguiente:

```
$_SESSION['nombre'] = $nombre;
```

Es decir al contenido de la variable \$nombre, lo guardo como variable de sesión con **\$\_SESSION[' ']**. De esta forma guardamos datos en una variable de sesión y la misma va a estar guardada todo lo que dure la sesión.

Luego de mostrarle el nombre ingresado al usuario por POST y por SESSION le doy opciones para elegir una bebida, agrego un botón siguiente y uno para volver atrás:



The screenshot shows a web form with the following content:

- Mi nombre por POST es: Hector Alterio
- Mi nombre por SESSION es: Hector Alterio
- A dropdown menu with "Cerveza" selected.
- A "Siguiente" button.
- A purple underlined link labeled "volver".

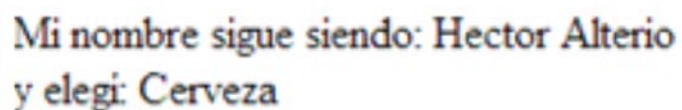
**3.php**

```
<?php
    SESSION_START( );

    $_SESSION['bebida'] = $_POST['bebidas'];
    echo "Mi nombre sigue siendo:
".$_SESSION['nombre']. "<br> y elegí:
".$_SESSION['bebida'] ;

    SESSION_DESTROY( );
?>
```

Nuestro último archivo, comienza con **SESSION\_START()**, ya que como dije anteriormente seguimos utilizando variables de sesión, guardo en una nueva variable de sesión la bebida que el usuario eligió en la página anterior, y por ultimo muestro su nombre y la bebida elegida. Y destruyo la sesión con **SESSION\_DESTROY()**.



Mi nombre sigue siendo: Hector Alterio  
y elegi: Cerveza

Resumiendo, si utilizo variables de sesión siempre en el encabezado del archivo PHP en donde va a pasar la variable debo comenzar la sesión con **SESSION\_START()**, y cuando termino la sesión (en el último archivo que la utilice) debo finalizarla con **SESSION\_DESTROY()**.

Para crear una variable de sesión utilizo **\$\_SESSION[' ']**, entre comillas el nombre de la variable.

## Actividades

1. Realizar un sitio web en donde el usuario ingrese un número y me muestre en otra página la tabla de multiplicar del mismo de 0 a 10. Se deberá utilizar una función y deberá ser incluida en otro archivo.
2. Realizar un sitio web que me permita calcular el iva (21%) de un monto ingresado por el usuario. El iva deberá ser calculado con una función y deberá estar en otro archivo que deberá ser incluido.
3. Realizar una encuesta que tenga 4 pantallas. La primera que le pida al usuario sus datos personales (Nombre, Apellido, Dirección, Mail, Teléfono, Sexo, Edad), una vez que lleno sus datos personales la segunda pantalla deberá pedirle los estudios cursados (Primario, Secundario, Terciario, Universitario), la tercer pantalla deberá preguntarle si practica algún deporte (fútbol, tenis, vóley basquet), que libros lee (Acción, Novelas, Dramas, Técnicos) y que tipo de comida le gusta (China, Vegetariana, Pastas). Por último la cuarta pantalla deberá mostrar todos los datos ingresados por el usuario en todas las pantallas anteriores. "Ud es: [Datos personales], tiene estudios de: [Estudios cursados], le gusta practicar [Deporte], le gusta leer [Libros] y le gusta la comida: [Comida]."
4. Agregarle al punto anterior la posibilidad de que el usuario ingrese al sistema a través de un login (sin base de datos). Si usuario es igual a [USUARIO] y contraseña es igual a [CONTRASEÑA], que pueda ingresar.

## Autoevaluación

1. ¿En qué nos beneficia el incluir un archivo?
2. ¿Qué diferencia hay entre include y require?
3. ¿Qué diferencia hay entre include\_once e include?
4. ¿Para qué se utilizan las variables de sesión?
5. Explique la función **SESSION\_START()**.