**ZEND PHP Certification**

**CAPITULO 1. BASES DE PHP**

* Lenguaje interpretado.
* Embebido en texto plano (HTML, Javascript).
* <?php ?>
  + Modo HTML <script language=”PHP”> </script>
  + Modo ASP <% %>
  + Modo abreviado <? ?>
  + <?= ?> (Todos los resultados generan un texto de salida, sin necesidad de echo, pero es propenso a que se inyecte código malicioso, por lo tanto es poco seguro al igual que el **register\_globals**)

Los 3 anteriores deben ser configurados en el archivo php.ini

* Objetivo: procesar los contenidos que da el usuario y regresar una salida HTML.
* include (“nombre.php”) – busca primero en la ruta local y luego en la ruta configurada para las librerías. Si no lo encuentra manda un warning pero sigue con la ejecución del programa.
* require(“nombre.php”) – hace la búsqueda igual que el anterior, pero si no se encuentra manda un error y termina la ejecución del programa.
* include\_once(“nombre.php”)
* PHP no es *fuertemente* tipificado, es decir, puede sumar o concatenar datos de diferentes tipos.
* **Tipos de datos**
  + *Numéricos*
  + 1 2 3 (enteros)
  + 0.5 0.001 (reales)
  + 0x68 (hexadecimal) -> 104
  + 0150 (octal) -> 104
  + *Cadenas*
  + Delimitadores: “, ‘ (con la comilla doble tengo cadenas dinámicas, ejemplo: “Hola $a” = Hola Juan; y con la comilla sencilla tengo una cadena estática, ejemplo: ‘Hola $a’ = Hola $a)
  + Para escapar caracteres se usa la \ (checar página 26 del manual)
  + Checar sintaxis *heredoc* (página 27 del manual) – todo lo que esté dentro se toma como cadena; es similar a lo que hace la comilla sencilla
  + *Booleano*
  + True / False (**false** - 0 o cualquier numero negativo, null; **true** – 1 o cualquier numero positivo, “ “, num, objeto)
  + *Arreglos*
  + $arr[] = valor (puede tener enteros, booleanos, etc)
  + $arr = (índice1 => “valor1”, indice2 => “valor2”)
  + Si no se le indica una posición, automáticamente se hace referencia a la última posición.
  + *Null*
  + Para asignar un valor vacío
* Identificadores válidos de variables:
  + $\_mivariable
  + $MiVariable
* Identificadores NO válidos de variables:
  + $1objeto
* $variable = “entero1”;

$entero1 = 10;

$$variable += 1 -> 11

($entero1)

Útil cuando se trabajan datos en serie, por ejemplo calificaciones.

* **Constantes**
* define(“identificador”, valor) el valor puede ser cualquier tipo de variable, excepto arreglos y objetos. Ejemplo : define(“MAX\_USERS”,30);
* La diferencia con las variables es que éstas no llevan el signo de $
* **Operadores**
* *Asignación* – puede ser por valor o por referencia
* Ejemplo paso por valor:

$a = 10

$b = $a

$b = 9 :. $a = 10 (son 2 localidades de memoria diferentes, por lo tanto **NO** se modifica el valor de $a)

* Ejemplo paso por referencia:
* $a = 10
* $b =& $a
* $b = 9 :. $a = 9 (ya que hace referencia a la misma localidad en memoria, por lo tanto se modifica el valor de $a)
* *Aritméticos*: + - \* / %
* *Bitwise o Bit a bit*: & | ^ ~ << >>

Ejemplo de corrimiento a la izquierda: 16 << 1 -> 00010000 -> 00100000

* *Control de errores*: @

@session\_start() – si la sesión ya se inició evita que envíe un warning o un error

* *Cadena*: concatenación (.)
* *Comparación***:** igualdad(==), idéntico tanto en tipo como en valor (===), diferencia o negación booleana (!=), <, >, <=, >=
* *Lógicos*: AND (&&), OR(||), XOR(xor), NOT(!)
* **Combinación de operadores**
  + +=, -=, .=
* **Ciclos**
  + Posibles valores falsos:

False

0

0.0

“0”

[]

{}

NULL

* *IF*
  + if(condicion){

---

}

else{

---

}

* + if(condicion):

---

else:

---

endif

* + Modo abreviado: condicion ? verdadero : falso;
* *CASE*
  + switch($valor){

case “valor1”:

---

break;

default:

---

break;

}

* *WHILE y DO WHILE*
  + while(condicion){

---

}

* + do{

---

}

while(condicion);

La diferencia es que éste se ejecuta por lo menos una vez.

* *FOR*
  + for(inicio; fin; incremento){

---

}

* + for($i=0, $c=2; $i<=0; $i++, $c+=2){

$arr[$i] = $c;

}

0 = 2

1= 4

… 10 = 22

* **Funciones**
  + function identificador (parametros){

---

return $x;

}

A partir de PHP 5 se puede hacer paso de valor por referencia.

* + Definición de valores por omisión:

function date($date = “Y-m-d H:i:s”){

---

return $valor;

}

* + function semanas(){

$GLOBALS[‘edad’] = 10;

return $GLOBALS[‘edad’] \* 52;

}

* + *func\_num\_args()* – regresa el número de parámetros que se le pasaron a una función.

*func\_get\_arg($num\_arg)* – regresa un parámetro en particular de la posición dada.

*func\_get\_args* – regresa un arreglo que contiene toda la lista de parámetros.

Ejemplo:

function promedio(){

$args = func\_num\_args();

$sum = 0;

for($i=0;$i<$args;$i++){

$sum += func\_get\_arg($i);

}

return $sum/$args;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CAPITULO 2. PHP ORIENTADO A OBJETOS**

* Sintaxis

class alumno{

atributos

métodos

}

*Atributos*:

Tienen la sintaxis de una variable de PHP

*Métodos*:

Tienen la sintaxis de una función de PHP

*Instancia*: definición única de un objeto que pertenece a determinada clase.

$a = new alumno();

Si la clase tiene un constructor se deben poner forzosamente los paréntesis, de lo contrario se pueden omitir.

*Herencia*: debe existir una clase base (aquélla que tiene toda la información genérica para todas las subclases).

**Funciones mágicas:**

Permiten determinar qué parte del objeto se va a meter dentro de una codificación especial.

Con la serialización se logra que un objeto se pueda mantener entre formularios. Las diferentes formas de hacerlo son:

* Campos ocultos entre formularios
* Variables de sesión (a partir de la 5.2 permite almacenar objetos)
* Almacenamiento en BD

Las funciones mágicas nos permiten serializar y des-serializar para poder guardar dichos objetos.

**Clases anidadas**

class1 - class2

$obj2 = new class2();

$obj1 = new class1();

$obj1 -> obj2 =& $obj2;

$obj1 -> obj2 -> contador = 8;

$obj2 -> contador = 10;

$obj1 -> obj2 -> contador++;

**? $obj1 -> obj2 -> contador**

**Respuesta = 11**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**CAPITULO 3. PHP y la Web**

* Métodos básicos del protocolo HTTPRequest:
  + GET – se envían las variables a través de la URL (Uniform Resource Locator)
  + POST – las variables se empaquetan y se envían posteriormente de manera transparente, es decir, no se pueden ver en la URL, por lo tanto se considera más seguro.
* **Formularios**
  + Es una forma de enviar datos al servidor.
  + Su etiqueta HTML es <form></form>
  + Pueden contener:

<input>

<select>

<textarea>

<file>

* El atributo para determinar a qué documento se pasaran las variables es denominado “action”, el atributo para determinar por cuál método se enviarán se llama “method”, y el atributo para uso interno del formulario es “name”.
* Otra forma de enviar un formulario es a través de javascript con la sentencia: document.form1.submit();
* NOTA: cuando se quieren enviar archivos por un formulario, se debe agregar al form el atributo *enctype* con el valor: application/x-www-form-urlencoded (para archivos binarios), multipart/form-data (para archivos de texto), o el valor por defecto text/plain (para archivos de texto plano).
* Para poder recuperar las variables de un formulario desde php con el mismo nombre o identificador, debe estar activa la variable register\_globlas.

**\*\*\* NOTA DE SINTAXIS PHP \*\*\***

<?= ?> -> el signo de = después de la etiqueta de apertura de php es un auto echo, es decir, no necesitamos poner la instrucción echo para mostrar la cadena que se escriba dentro de las mismas.

* **Sesiones**

session\_start() – inicia sesión

session\_register() – registra una variable de sesión

session\_unregister – elimina una variable de sesión

session\_unset() – reinicia la sesión

session\_destroy – elimina la sesión

* + Para acceder a las variables de sesión ocupamos el arreglo asociativo: $\_SESSION[‘var’]
  + @session\_start() – con la @ evitamos que se muestre un error en caso de que el session\_autostart esté activo
  + **Serialización** – representar un objeto php en su forma de cadena

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CAPITULO 5. CADENAS Y EXPRSIONES REGULARES**

**Expresiones regulares**

Conjunto de herramientas basadas en un lenguaje bien definido que permiten hacer procesos de búsqueda y reemplazo de información de una manera muy sencilla.

Los lenguajes basados en expresiones regulares permiten un procesamiento d información que tienen un contenido definido.

**Búsqueda:**

[Expresión] *match* [Cadena] -> si/no

preg\_match($exp,$objetivo);

preg\_match\_all($exp,$objetivo,$arreglo de matchs),

Para poder hacer una búsqueda, reemplazo o mapeo global, se utiliza la función preg\_match\_all().

**Reemplazo:**

[Expresión - Reemplazo] *match* [Cadena] -> nueva cadena

preg\_replace($exp,$reemplazo,$objetivo); (regresa una cadena)

**Mapeo:**

[Expresión con estructura] map [Cadena] -> arreglo

preg\_split($exp,$objetivo); (regresa un arreglo)

A diferencia de javascript y java (/*expresión*/), PHP maneja las expresiones regulares como cadena.

Ejemplos:

$exp = “/(\S+)@(\S+)/”;

$cadena = preg\_replace($exp,”\1 en \2”,”yo@server.com”);

$cadena = “yo en server.com”;

$str = preg\_replace(“/(\d{2})\/(\d{2})\/(\d{4})/”,”\3-\1-\2”,$strArchivo);

$correo = “\w{1,}@\w{1,}(\.\w{1,}){1,}”;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CAPITULO 6. MANEJO DE ARCHIVOS**

**Mecanismos de seguridad**

* Token
* Sesiones / cookies
* Conexiones seguras

**Manejo de archivos**

Si queremos que la información de un archivo persevere se abre en modo “a” o append y si no es importante conservarla se abre en modo “w”.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**CAPITULO 7. ADMINISTRANDO FECHAS Y HORAS**

PHP no tiene una variable de tipo fecha, pero existen funciones para calcularla.

**Entero:**

time() – numero de milisegundos en formato (UTC)

**Arreglo:**

gettimeofday - Segundos, milisegundos, minutos que han transcurrido o si está activo el horario de verano (UTC)

getdate() – minutos, segundos, mes, día, etc (LT)

localtime() – igual que el anterior pero en vez de arreglo asociativo devuelve un arreglo ordenado (LT)

**Cadena:**

MySql – ‘AAAA-MM-DD’

date() – si no le pasas ningún parámetro devuelve: ddd, dd mmmm aaaa hh:mm:ss (GMT)

strftime() – permite recibir cualquier cadena que se quiera usar. Ejemplos:

strftime(‘%a’) -> ‘Sáb’

Es igual a la función date pero sus modificadores toman en cuenta la configuración del sistema, es decir podremos ver la fecha en el idioma que hayamos configurado.

mktime(hr, min, seg, mes, dia, año, horario verano) – regresa el timestamp de una fecha determinada.

Nota: si se pone una fecha no válida, por ejemplo mes 14 del año 2009, lo interpreta como mes 2 del año 2010. Por lo tanto ésta función permite avanzar en el tiempo y modificarlo.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CAPÍTULO 8. ADMINISTRANDO CORREO**

Un correo electrónico es un archivo de texto plano, PHP se encarga de crear los mecanismos para poder enviarlo.

Se divide en 3 secciones principales:

* Destinatario
* Cuerpo
* Cabeceras

**MTA –Mail Transport Agent**

Se encarga de enviar y recibir correos y las diferentes notificaciones que se van generando durante el proceso.

Algunos ejemplos son:

* sendmail (servidor dedicado, se deben crear las cuentas de usuario)
* posfix (permite una administración un poco más compleja y permite la intercomunicación entre varios servidores, los usuarios de sistema pueden utilizar la misma cuenta)
* qmail (portable, los usuarios de sistema pueden utilizar la misma cuenta)
* Microsoft Exchange(plataforma Windows, permite conexiones hacia un mecanismo de autenticación llamado Active Directory)
* Lotus Notes

**SMTP – Simple Mail Transport Protocol**

Define cómo se manejan los destinatarios, cuerpo y cabecera de un correo.

Utiliza el puerto 25 para el envío y recepción de correo electrónico.

**MUA – Mail User Agent**

* Outlook (Windows)
* Mail (MAC)
* Pine (Linux)

Permite revisar el correo, enviar, recibir, etc.

Para enviar correos con PHP se requiere un servidor con un MTA y PHP y se pueden enviar con la función mail($destino,$cuerpo,[$encabezados]).

El destino debe ser una cadena con un formato de correo electrónico válido. El MTA se encarga de pegarle el nombre del dominio que tenga asociado, por lo tanto si se quiere enviar a un correo con el mismo dominio solo basta el nombre del usuario.

El cuerpo contiene texto plano o texto con formato con HTML.

En las cabeceras van las definiciones de C.C, C.C.O, Adress\_response, tipo, attaches, en formato de cadena y todas las líneas terminan con \r\n y el primer elemento que deben tener es la definición de lo que están enviando.

Ejemplo:

$strDestino = [mimail@hotmail.com](mailto:mimail@hotmail.com);

$strCuerpo = “Hola \r\n”;

$strCabecera = “Cc: [mimama@hotmail.com](mailto:mimama@hotmail.com) \r\n” .

“Cco: [otradireccion@hotmail.com](mailto:otradireccion@hotmail.com) \r\n” .

“MIME-version 1.0 \r\n” .

“Content-Type: text/html; charset ‘iso\_8859\_1’ \r\n” .

“Content-Transference-Encoding: 7bit \r\n”;

Cuando se envían adjuntos PHP recomienda usar un cuarto parámetro:

$adjunto = “MIME-version 1.0 \r\n” .

“Content-Type: multipart/mixed; boundary=\”php-12345\” \r\n”;

*Boundary* - Es un delimitador para saber hasta dónde llega el adjunto, y debe ser una cadena única que no exista dentro del adjunto.

El contenido de un archivo adjunto va dentro del cuerpo de la siguiente manera:

“mensaje” . “\r\n” . “Content-Type: text/plain \r\n” . “Mi otro texto \r\n” . “-php-12345 \r\n”;

Para adjuntar imágenes se utiliza:

“Content-Type: imp/gif; name=\”archivo.gif\” \r\n” .

“Content-Transference-Encoding = base 64 \r\n” .

“Content-Location: URL \r\n” .

chunk\_split(base 64\_encode(file\_get\_contents(\”archive.gif”\))) . “\r\n” . “-php-12345”;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CAPÍTULO 9. PHP y BASES DE DATOS**

En las bases de datos se distinguen dos aspectos principales: la estructura y el contenido (datos).

SQL -> Standar Query Language

CREATE DATABASE <nombre>

CREATE TABLE <nombre>(

<campo> <tipo>,

<campo> <tipo>,

)

INDEX ON <campo(s)> TO table

**Llave primaria**: identificador único para cada registro de la BD. Consideraciones:

* Única
* No character
* No fecha

**Integridad Referencial**: identifica las llaves duplicadas dentro de la BD.

**Llave compuesta**: cuando se juntan varios campos para crear el índice.

**Llave foránea**: cuando se asigna la llave primaria de una primera tabla a una segunda, siempre y cuando estén relacionadas.

SELECT <campo(s)|\*> FROM tabla

WHERE <condición(es)>

ORDER BY <columna|criterio>

GROUP BY <campo(s)>

INSERT INTO tabla [(campo1,campo2,…, campoN)] VALUES (valor1, valor2, …, valorN)

INNER JOIN tabla1.campo (PK), tabla2.campo (FK)

UPDATE tabla SET campo = valor, campo2 = valor2 WHERE <criterio>

DROP – Elimina la estructura

DROP INDEX PRIMARY | DROP TABLE table | DROP DATABASE baseDatos

DELETE – Elimina el contenido

DELETE [FROM] tabla WHERE <criterio>

**Transacción**: Conjunto de acciones que tienen como característica que o bien se ejecuta una o no se ejecuta ninguna.

Sintaxis: BEGIN TRANSACTION

Sentencias SQL

COMMIT TRANSACTION | ROLLBACK TRANSACTION

**Funciones Agregadas**

*AVG()* – Promedio

*COUNT()* – Total de columnas correspondientes a algún criterio

*SUM()* – Suma el contenido de los campos

*MAX() | MIN()* – Valor máximo o mínimo de una columna específica

ORDER BY – Puede ser ascendente (0-1 | simbolos, A-Z, a-z, 0-1) o descendente.

Si no se define el criterio por default lo hace ascendente.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CAPITULO 10. STREAMING Y PROGRAMACIÓN DE RED**

**Streaming**

* Información enlazada
* Tiene un inicio y el final depende del tiempo
* Permite manipular segmentos de datos sin que necesariamente pase desde el inicio y hasta el final.
* Tiene como ventaja que se trabaja en tiempo real

PHP tiene algunos mecanismos basados en stream pero son para el manejo de archivos.

**File wrappers (Envolturas de arvhivos)**

Permiten acceder a un recurso a través de un conjunto de mecanismos que permiten recuperar la información.

Se definen como: scheme://ruta/archivo

Ejemplos:

[*http://sitio/pagina.php*](http://sitio/pagina.php) - accede al protocolo HTTPD e interpreta el código

[*file://sitio/pagina.php*](file://sitio/pagina.php) - accede al file system y muestra el archivo como tal (texto/binario/local)

[*ftp://sitio/pagina/*](ftp://sitio/pagina/) - despliega los mismos resultados que el anterior pero llega a ellos de manera diferente (texto/binario/red)

*/var/www/ (UNIX) | C:\ (windows)*

*xmlns://*

*php://* - permite acceso a los archivos PHP, ya sea el código o ya interpretado dependiendo de la función que se le pase (se especifica una ruta local, no desde navegador).

compress.zlib:// - permite leer, escribir o crear archivos comprimidos zip, bzz, gz

compress.bzip2:// - permite leer, escribir o crear archivos comprimidos zip, bzz, gz

Pueden ser predefinidas por PHP o definidas por el usuario. Las predefinidas por PHP a su vez se dividen en Locales y de Red.

Para que funcionen el http/https y ftp/ftps se requiere habilitar el parámetro allow\_url\_fopen en el php.ini

Todas las envolturas anteriores permiten leer, excepto php:// y todos permiten escribir excepto http/https.

Todas las envolturas permiten borrar, renombrar y crear un directorio (puede haber unas limitantes en Windows con PHP 4).

**Streams**

No se puede utilizar el fread (trae todo el archivo a una variable), en su lugar se ocupa el fget (lee renglón por renglón).

La información está disponible mientras la conexión a su vez esté disponible.

Se recomienda que en un lugar de usar una apertura de archivo convencional se use un socket.

**Socket**

Se tiene un archivo asociado a una conexión el cual tiene por lo general la extensión socket. Dichos archivos se conectan a un puerto y ejecutan la acción que se le haya establecido.

Otras envolturas que soporta PHP son:

* TCP
* UDP
* SSL
* TLS
* UNIX
* UDG

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**CAPITULO 11. SEGURIDAD**

Existen diversos mecanismos de seguridad para cada capa del modelo OSI, sin embargo PHP tiene identificados ciertos mecanismos que son los siguientes:

**Register\_globals = on / off**

Recomendaciones:

* No se debe ver el listado de directorios. (Se pone un index en la carpeta o se configura el htaccess de apache).
* Las librerías deben estar en una carpeta aparte y sin acceso web, si lo tuviera, poner un índice que no permita visualizarlos.

**Inyección de SQL**

Recomendaciones:

* Debido a que en PHP las consultas pasan como cadena se puede utilizar la función mysql\_escape\_string() que cancela las funciones de delimitador.

**Inyección de comandos**

Recomendaciones:

* Tratar de no poner comandos en formularios.

**Inyección de scripts**

Recomendaciones:

* Utilizar las funciones htmlentities() o htmlspecialchar() para que interprete todo como texto plano. Una solución desde html es el tag <pre></pre>

**Hosting compartido**

Recomendaciones:

* Habilitar en el php.ini el safe\_mode para no poder acceder a las carpetas a las que no se tenga permiso.
* Crear una jaula (cage) la cual crea un entorno virtual de los archivos propios y se configura desde el servidor.
* Usar el parámetro open\_basedir, para establecer a partir de qué ruta se puede obtener información. Esta se complemente con includes\_dir.

**Identidad**

Recomendaciones:

* Usar variables de sesión
* Usar la variable http\_referer

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*