Clases

❑ DateTime

https://www.php.net/manual/es/class.datetime.php

La clase DateTime es la usada para manejar las fechas y horas de forma robusta en php.

El método constructor es el siguiente: public __construct(string $datetime = "now", ?DateTimeZone $timezone = null)

Otro método importante es el **format**(), que te permite poner el formato deseado. A consultar: https://www.php.net/manual/es/datetime.format.php

Otro método destacado es **getTimestamp**(), que te devuelve la diferencia en segundos de la fecha DateTime con respecto al 1 de enero de 1970

https://www.php.net/manual/es/datetime.format.php

Por último, el método **diff**() te calcula la diferencia entre dos fechas. Te devuelve un objeto DateInterval

https://www.php.net/manual/es/datetime.diff.php

❑ PDO

https://www.php.net/manual/es/class.pdo.php

Es la clase que representa una conexión entre PHP y un servidor de bases de datos

El constructor es: public __construct(
  string $dsn,
  ?string $username = null,
  #[\SensitiveParameter] ?string $password = null,
  ?array $options = null)

El método **query**() te permite ejecutar una consulta sobre la base de datos

Ejemplo: $ConexionBD->query('select * from tabla')

https://www.php.net/manual/es/pdo.query.php

El método **prepare**() es parecido al query, pero sin ejecutarse inmediatamente. Permite al código a prepararse para la consulta y que tarde menos en ejecutarse. Además, ayuda a la seguridad. Para ejecutar dicha consulta usar **execute**()

Ejemplo : $consulta = $ConexionBD->prepare('select * from tabla')

https://www.php.net/manual/es/pdo.prepare.php

El método **exec**() se usa para ejecutar consultas sql devolviendo el número de filas afectadas por la consulta.

```
$ ConexionBD ->exec("DELETE FROM fruit");
```

https://www.php.net/manual/es/pdo.exec.php

El método **beginTransaction()** es usado cuando tienes varias consultas y quieres, o que se ejecuten todas, o que no se ejecute ninguna. Esto desactiva el autocommit, así que para que se guarden los cambios se debe usar el método **commit**() si todo ha ido bien o el método **rollback**() si algo ha salido mal y no queremos que se guarden los cambios.

Ejemplo de codigo real:

```
try{
        $miDB->beginTransaction();
        foreach ($aInsert as $registro){
           $insertql="...";
           $miDB->query($insertql);
        }
        $miDB->commit();
        echo 'HA FUNCIONADO';
    } catch (PDOException $miExceptionPDO){
        $miDB->rollBack();
        echo'Error: '.$miExceptionPDO->getMessage();
        echo '<br>';
        echo'Código de error: '.$miExceptionPDO->getCode();
    } finally {
        unset($miDB);
    }
```

https://www.php.net/manual/es/pdo.begintransaction.php

❑ PDOStatement

https://www.php.net/manual/es/class.pdostatement.php

Representa tanto como una consulta preparada como su conjunto de resultados una vez ejecutada la consulta.

Con el método **execute**() se ejecutan las consultas preparadas

$ConexiónBD ->execute([entre estos corchetes se puede añadir opcionalmente los parametros. Ej: ‘:fila’ => 3]).

Con el método fetchObject() obtenemos la siguiente fila del conjunto de resultados y nos devuelve un objeto

Ej.: $obj = $ ConexiónBD ->fetchObject();

❑ PDOException

https://www.php.net/manual/es/class.pdoexception.php

Esta clase representa un error emitido por la clase PDO.

Se suele usar en la estructura de un try catch, concretamente en la parte de catch

Ejemplo: try{

Codigo …

}catch(PDOException $e){

}

Dos métodos que son bastante parecidos, getCode() y getMessage(), que nos devuelve el número de código del error y el mensaje de error respectivamente

❑ DOM Document

https://www.php.net/manual/es/class.domdocument.php

Representa un documento HTML o XML completo; será la raíz del árbol del documento.

Con esta clase se puede manipular por completo un archivo HTML o XML. Crear elementos, añadir atributos, modificarlos,etc…

Funciones

❑ Is_null

https://www.php.net/manual/es/function.is-null.php

Es una función que sirve para saber si una variable tiene un valor = null. La función nos devuelve true o false.

Su uso es sencillo, simplemente is_null($variable)

❑ Empty

Es una función que sirve para saber si una variable esta vacia. La función nos devuelve true o false.

Su uso es sencillo, simplemente empty($variable)

Los casos en los que una variable se considera vacía: si vale 0; si esta literalmente vacia; si no esta definida

Ej: $var=0; o $var='';

❑ Isset

Es una función que sirve para saber si una variable esta declarada y es diferente de null. La función nos devuelve true o false.

Su uso es sencillo, simplemente isset($variable)

Ejemplo de uso: Si tenemos un botón y queremos que haga algo cuando se pulse, podemos usar el método isset($boton) dentro de un if(). Esto es asi porque hasta que se pulse, la variable $boton se considerara null, y la función nos devolvería falso. Sin embargo, una vez pulse el botón la variable dejará de ser null y se ejecutará el código dentro del if()

❑ Header

Envia una cabecera HTTP sin procesar. Puede cargar un formulario de login por defecto:

```
header('WWW-Authenticate: Basic realm="Contenido restringido"');
header('HTTP/1.0 401 Unauthorized');
```

O podemos cargar una página a nuestra eleccion con el parametro Location:

```
header('Location: http://www.example.com/');
```