Universidad Tecnológica de Coahuila

Unidad 3

Desarrollo de back-end

Saber 3 para evaluar el Saber de la materia de apps web

Técnico Superior Universitario
En

TI

Área infraestructura de redes y ciberseguridad.

Elaborado por:

Jesus Francisco Lucio Calvillo

Maestro:

Gabriel reyes

Ramos Arizpe, Coahuila

21/07/2024

Introducción

Para esta unidad enfocada en el back-end es necesario saber algunos conceptos mas profundos y complejos en la programación de aplicaciones web, uno de ellos y a su ve el primeo que se tratara en este saber serán las sintaxis básicas del lado del servidor, importantísimas bases de programación de back end con php, contemplando las variables, las expresiones las estructuras de control y las clases.

Otro punto muy importante son los métodos de los formularios en html, los cuales son la manera en la que se comunicara el liente con el servidor y viceversa, en este saber se presenta un cuadro comparativo de los métodos: POST y GET, sus características y sus ventajas y desventajas.

Ya se habló de los métodos de comunicación, pero para que sean útiles hay que estar conectados a la base de datos, aquí se presentan dos métodos de conexión de base de datos, también se describen los procesos de conexión, inserción, modificación y eliminación de la misma base.

Cuando se habla de interacción con un usuario lo más común y usado en la actualidad, es la capacidad de las paginas y aplicaciones web que permiten al usuario subir y descargar archivos, desde una imagen hasta aplicaciones completas, en este texto tiene su apartado con su sintaxis para su uso.

Y para terminar se presenta de manera clara y concisa un mapa mental en donde se habla de la importancia y funcionamiento de las sesiones en ambientes web.

sintaxis basica de PHP

variables

Las variables en PHP se definen con el símbolo $ seguido del nombre de la variable y se pueden asignar diferentes tipos de valores.

- **$variable = valor;**
- $nombre = "Juan"; // Cadena de texto
  $edad = 25; // Número entero

EXPRESIONES

Operadores aritméticos

Se utilizan para realizar operaciones matemáticas básicas

- **$suma = $a + $b; // Suma**
  $resta = $a - $b; // Resta
  $multiplicacion = $a * $b; // Multiplicación
  $division = $a / $b; // División

Operadores de comparación

Se utilizan para comparar valores y devuelven un valor booleano (verdadero o falso).

- **$igual = $a == $b; // Igual a**
  $diferente = $a != $b; // Diferente de
  $mayor = $a > $b; // Mayor que
  $menor = $a < $b; // Menor que

BY: JESUS FRANCISCO LUCIO CALVILLO
20/07/2024

sintaxis básica de

PHP

ESTRUCTURAS DE CONTROL

Operadores lógicos

Se utilizan para combinar expresiones lógicas y evaluar condiciones complejas.

```
$y = $a && $b; // Y lógico (AND)
$o = $a || $b; // O lógico (OR)
$no = !$a; // No lógico (NOT)
```

Condicionales (if, else, elseif)

Permiten ejecutar código condicionalmente basándose en si una expresión evaluada es verdadera o falsa.

```
if ($condicion) {
    // Código si $condicion es verdadera
} elseif ($otraCondicion) {
    // Código si $otraCondicion es verdadera
} else {
    // Código si ninguna condición es verdadera
}
```

Switch

Permite seleccionar entre múltiples bloques de código a ejecutar, basándose en el valor de una variable

```
switch ($variable) {
    case "valor1":
        // Código si $variable es igual a "valor1"
        break;
    case "valor2":
        // Código si $variable es igual a "valor2"
        break;
    default:
        // Código si ningún caso coincide
}
```

BY: JESUS FRANCISCO LUCIO CALVILLO
20/07/2024

BUCLES

sintaxis básica de

PHP

while

Ejecuta un bloque de código mientras una condición sea verdadera

```
while ($condicion) {
    // Código a ejecutar mientras $condicion es verdadera
}
```

do...while

Similar a while, pero garantiza que el bloque de código se ejecuta al menos una vez.

```
do {
    // Código a ejecutar
} while ($condicion);
```

FOR

Permite ejecutar un bloque de código un número específico de veces, útil para iteraciones controladas.

```
for ($i = 0; $i < 10; $i++) {
    // Código a ejecutar en cada iteración
}
```

BY: JESUS FRANCISCO LUCIO CALVILLO
20/07/2024

sintaxis básica de

PHP

foreach

Itera sobre cada elemento de un arreglo, asignando el valor del elemento a una variable temporal.

```
foreach ($array as $valor) {
    // Código a ejecutar para cada elemento en $array
}
```

CLASES

Definición de Clase

Las clases son plantillas para crear objetos, que pueden tener propiedades (variables) y métodos (funciones).

```
class NombreClase {
    // Propiedades
    public $propiedad;

    // Método Constructor: Se ejecuta al crear una instancia de la clase
    public function __construct($param) {
        $this->propiedad = $param;
    }

    // Métodos: Definen el comportamiento de los objetos de la clase
    public function metodo() {
        // Código del método
    }
}
```

Crear una Instancia

Para utilizar una clase, se crea una instancia (objeto) de ella

```
$objeto = new NombreClase("valor");
```

BY: JESUS FRANCISCO LUCIO CALVILLO
20/07/2024

sintaxis básica de

PHP

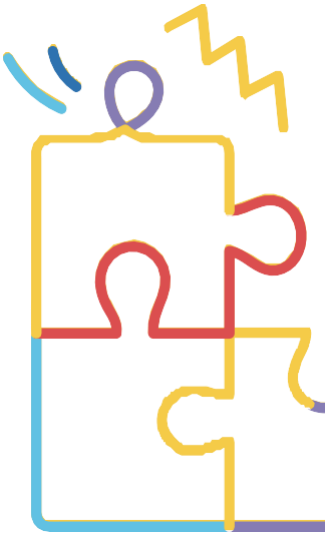
Acceso a Propiedades y Métodos

Se accede a las propiedades y métodos de un objeto usando el operador ->.

```
echo $objeto->propiedad;  // Acceso a la propiedad
$objeto->metodo();        // Llamada a un método
```

BY: JESUS FRANCISCO LUCIO CALVILLO
20/07/2024

GET O POST



CUADRO COMPARATIVO

ASPECTO A COMPARAR	POST	GET
METODO	Envía datos en el cuerpo de la solicitud	Envía datos en la URL del navegador
USO COMUN	Enviar datos al servidor (creación/actualización)	Recuperar datos del servidor (lectura)
LONGITUD DE DATOS	Ilimitada (depende del servidor)	Limitada por la URL (aprox. 2048 caracteres)
VISIBILIDAD	Datos ocultos en el cuerpo de la solicitud	Datos visibles en la URL
IDEMPOTENCIA	Sí, hacer la misma solicitud varias veces no cambia el estado del servidor	No necesariamente, puede cambiar el estado del servidor

BY: JESUS FRANCISCO LUCIO CALVILLO

GET O POST

CUADRO COMPARATIVO

ASPECTO A COMPARAR	POST	GET
CACHE DEL NAVEGADOR	No, generalmente no son almacenadas en caché	Sí, las solicitudes pueden ser almacenadas en caché
MARCADORES Y URLS COMPARTIBLES	No, no se pueden marcar ni compartir URLs	Sí, se pueden marcar y compartir fácilmente
SEGURIDAD	Más seguro, los datos no se exponen en la URL	Menos seguro, los datos se exponen en la URL
VENTAJAS	- Permite enviar grandes cantidades de datos -Mayor seguridad para datos sensibles - No tiene restricciones de longitud	- Fácil de implementar y usar - Permite marcar y compartir URLs - Permite cacheo
DESVENTAJAS	- No se puede marcar ni compartir fácilmente - No se almacena en caché - Puede ser más complejo de implementar	- Limitado a la longitud de la URL - Menos seguro para datos sensibles - No adecuado para operaciones que cambian el estado del servidor

BY: JESUS FRANCISCO LUCIO CALVILLO

métodos de conexión a base de datos

by Jesús francisco lucio calvillo

USANDO PDO

PDO es una extensión de PHP que proporciona una interfaz orientada a objetos para acceder a bases de datos de manera uniforme. Soporta múltiples bases de datos, lo que facilita el cambio de SGBD sin modificar el código.

Este bloque de código muestra cómo establecer una conexión a una base de datos MySQL utilizando PDO. Si la conexión falla, se lanza una excepción que puede ser capturada para manejar errores de conexión.

```
try {
    $dsn = 'mysql:host=localhost;dbname=testdb';
    $username = 'root';
    $password = 'password';

    // Crear una instancia de PDO
    $pdo = new PDO($dsn, $username,
$password);

    // Establecer el modo de error de PDO a
excepción
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

     echo "Conexión exitosa";
} catch (PDOException $e) {
    echo "Error en la conexión: " . $e-
>getMessage();
}
```

ejecutar consultas

```
// Consulta SELECT
$stmt = $pdo->query('SELECT * FROM users');
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
    echo $row['name'] . "<br>";
}

// Consulta INSERT
$sql = "INSERT INTO users (name, email) VALUES (:name, :email)";
$stmt = $pdo->prepare($sql);
$stmt->execute(['name' => 'Juan', 'email' => 'juan@example.com']);

// Consulta UPDATE
$sql = "UPDATE users SET email = :email WHERE name = :name";
$stmt = $pdo->prepare($sql);
$stmt->execute(['email' => 'nuevoemail@example.com', 'name' => 'Juan']);

// Consulta DELETE
$sql = "DELETE FROM users WHERE name = :name";
$stmt = $pdo->prepare($sql);
$stmt->execute(['name' => 'Juan']);
```

Este bloque muestra cómo realizar consultas de selección, inserción, actualización y eliminación utilizando PDO. Las consultas preparadas (prepared statements) son utilizadas para prevenir inyecciones SQL

Uso de la Clase de Conexión

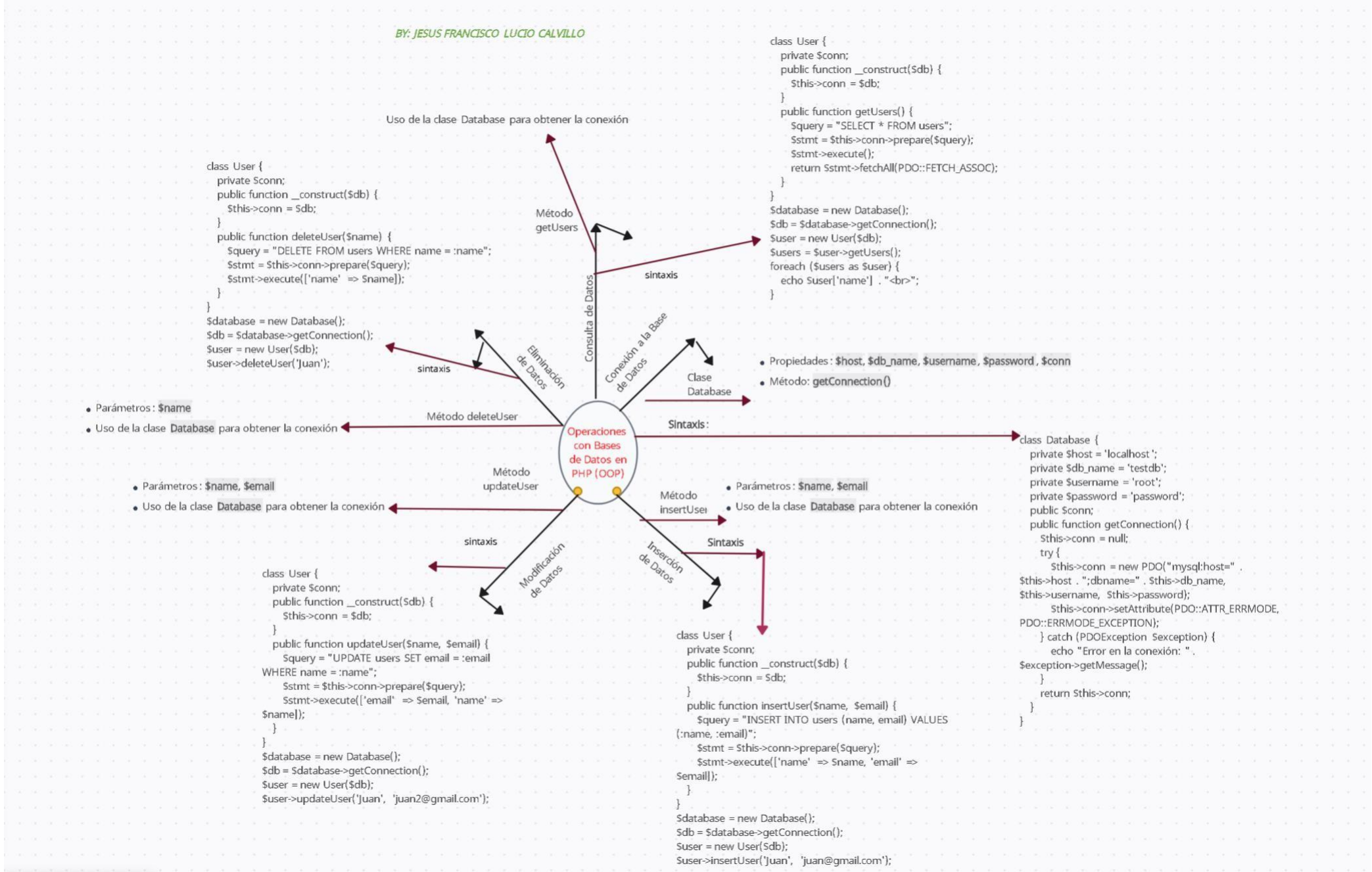
Crear clases personalizadas para manejar conexiones a bases de datos y operaciones CRUD es una práctica común en aplicaciones orientadas a objetos, promoviendo la reutilización del código y mejorando la mantenibilidad.

Esta clase Database encapsula los detalles de la conexión a la base de datos. El método getConnection establece una conexión a la base de datos utilizando PDO y maneja cualquier excepción que pueda ocurrir durante la conexión.

```
class Database {
    private $host = 'localhost';
    private $db_name = 'testdb';
    private $username = 'root';
    private $password = 'password';
    public $conn;

    // Método para obtener la conexión

    public function getConnection() {
        $this->conn = null;
        try {
            $this->conn = new PDO("mysql:host=" .
$this->host . ";dbname=" . $this->db_name,
$this->username, $this->password);
            $this->conn-
>setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
        } catch (PDOException $exception) {
            echo "Error en la conexión: " . $exception-
>getMessage();
        }
        return $this->conn;
    }
}
```

```
// Crear una instancia de la clase Database y obtener la conexión
$database = new Database();
$db = $database->getConnection();

// Consulta SELECT
$query = "SELECT * FROM users";
$stmt = $db->prepare($query);
$stmt->execute();
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
    echo $row['name'] . "<br>";
}

// Consulta INSERT
$query = "INSERT INTO users (name, email) VALUES (:name, :email)";
$stmt = $db->prepare($query);
$stmt->execute(['name' => 'Juan', 'email' => 'juan@example.com']);

// Consulta UPDATE
$query = "UPDATE users SET email = :email WHERE name = :name";
$stmt = $db->prepare($query);
$stmt->execute(['email' => 'nuevoemail@example.com', 'name' => 'Juan']);

// Consulta DELETE
$query = "DELETE FROM users WHERE name = :name";
$stmt = $db->prepare($query);
$stmt->execute(['name' => 'Juan']);
```

Este bloque muestra cómo utilizar la clase Database para establecer una conexión y ejecutar consultas SQL. Las consultas SELECT, INSERT, UPDATE y DELETE se realizan utilizando consultas preparadas para mayor seguridad.




```
class User {
  private $conn;
  public function __construct($db) {
    $this->conn = $db;
  }
  public function getUsers() {
    $query = "SELECT * FROM users";
    $stmt = $this->conn->prepare($query);
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
  }
}
$database = new Database();
$db = $database->getConnection();
$user = new User($db);
$users = $user->getUsers();
foreach ($users as $user) {
  echo $user['name'] . "<br>";
}
```

```
class User {
  private $conn;
  public function __construct($db) {
    $this->conn = $db;
  }
  public function deleteUser($name) {
    $query = "DELETE FROM users WHERE name = :name";
    $stmt = $this->conn->prepare($query);
    $stmt->execute(['name' => $name]);
  }
}
$database = new Database();
$db = $database->getConnection();
$user = new User($db);
$user->deleteUser('Juan');
```

```
class User {
  private $conn;
  public function __construct($db) {
    $this->conn = $db;
  }
  public function updateUser($name, $email) {
    $query = "UPDATE users SET email = :email
WHERE name = :name";
    $stmt = $this->conn->prepare($query);
    $stmt->execute(['email' => $email, 'name' =>
$name]);
  }
}
$database = new Database();
$db = $database->getConnection();
$user = new User($db);
$user->updateUser('Juan', 'juan2@gmail.com');
```

```
class User {
  private $conn;
  public function __construct($db) {
    $this->conn = $db;
  }
  public function insertUser($name, $email) {
    $query = "INSERT INTO users (name, email) VALUES
(:name, :email)";
    $stmt = $this->conn->prepare($query);
    $stmt->execute(['name' => $name, 'email' =>
$email]);
  }
}
$database = new Database();
$db = $database->getConnection();
$user = new User($db);
$user->insertUser('Juan', 'juan@gmail.com');
```

```
class Database {
  private $host = 'localhost';
  private $db_name = 'testdb';
  private $username = 'root';
  private $password = 'password';
  public $conn;
  public function getConnection() {
    $this->conn = null;
    try {
      $this->conn = new PDO("mysql:host=" .
$this->host . ";dbname=" . $this->db_name,
$this->username, $this->password);
      $this->conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    } catch (PDOException $exception) {
      echo "Error en la conexión: " .
$exception->getMessage();
    }
    return $this->conn;
  }
}
```

Carga y Descarga de Archivos en PHP (OOP)

BY: JESUS FRANCISCO LUCIO CALVIILLO

carga (upload)

Subir (uploadFile)

Clase `Upload`

```
private EtargetDir;
private Efile;
public function _construct(EtargetDir) {
Ethis->targetDir = EtargetDir;
}

public function uploadFile(Efile) {
Ethis->file = Efile;
EtargetFile = Ethis->targetDir.basename(Ethis->file["name"]);
EuploadOk = 1;
EimageFileType = strtolower(pathinfo(EtargetFile, PATHINFO_EXTENSION));

Echeck = getimagesize(Ethis->file["tmp_name"]);
if (Echeck !== false) {
echo "El archivo es una imagen- " . Echeck["mime"] . ".";
EuploadOk = 1;
} else {
echo "El archivo no es una imagen.";
EuploadOk = 0;
}

if (file_exists(EtargetFile))
{
echo "El archivo ya existe.";
EuploadOk = 0;
}

if (Ethis->file["size"] > 500000) {
echo "Lo siento, tu archivoes demasiado grande.";
EuploadOk = 0;
}

if (EimageFileType != "jpg" &&
EimageFileType != "png" &&
EimageFileType != "jpeg" &&
EimageFileType != "gif") {
echo "Lo siento, solo se permiten archivos JPG, JPEG, PNG y GIF.";
EuploadOk = 0;
}

if (EuploadOk == 0) {
echo "Lo siento, tu archivo no fue subido.";
} else {
if (move_uploaded_file(Ethis->file["tmp_name"], EtargetFile))
{
echo "El archivo " . basename(Ethis->file["name"]) . " ha sido subido.";
} else {
echo "Lo siento, hubo un error al subir tu archivo.";
}
}
}
}
```

descarga (download)

Descargar (downloadFile)

Clase `Download`

```
private 3fileDir;

public function __construct(3fileDir) {3this->fileDir = 3fileDir;
}

public function downloadFile(3fileName) {
3filePath = 3this->fileDir . fileName;
if (file_exists(3filePath)) {
header('Content-Description:File Transfer');
header('Content-Type: application/octet-stream');
header('Content-Disposition:attachment; filename="'.basename(3filePath).
header('Expires: 0');
header('Cache-Control: must-revalidate');
header('Pragma: public');
header('Content-Length: ' .
filesize(3filePath));
flush();
readfile(3filePath);
exit;
} else {
echo "El archivo no existe.";
}
}
```

BY:JESUS FANCISCO LUCIO CALVILLO

SESIONES EN AMBIENTES WEB

FUNCIONAMIENTO

INICIO DE SESIÓN

Autenticación: Verifica la identidad del usuario.

Validación de Credenciales: Comprueba que el usuario ha ingresado la información correcta.

ALMACENAMIENTO DE DATOS DEL USUARIO

Guarda información relevante mientras dure la sesión.

CIERRE DE SESIÓN

Finaliza la sesión y borra los datos almacenados.

SEGURIDAD EN SESIONES

PREVENCIÓN DE SECUESTRO DE SESIÓN

Implementación de medidas para evitar que un atacante tome el control de una sesión de usuario.

REGENERACIÓN DE ID DE SESIÓN

Cambia el ID de sesión regularmente para mitigar ataques.

USO DE COOKIES SEGURAS

Configura las cookies para que solo se transmitan a través de conexiones HTTPS, evitando la transmisión de cookies en texto claro.

MECANISMOS DE ALMACENAMIENTO

COOKIES

Pequeños archivos de texto almacenados en el navegador del usuario.

VARIABLES DE SESIÓN

Datos almacenados en el servidor que corresponden a una sesión específica.

BASES DE DATOS

Se pueden utilizar para almacenar información de sesión en una escala más grande y de manera más segura.

IMPORTANCIA

MEJORA LA EXPERIENCIA DEL USUARIO

Facilita una experiencia más fluida y personalizada.

Las sesiones son una técnica esencial para mantener el estado y la persistencia de datos entre diferentes páginas web.

PERSISTENCIA DE DATOS ENTRE PÁGINAS:

Permite que los datos persistan mientras el usuario navega por diferentes páginas de un sitio web.

CONCLUSION

Después de esta extensa compilación de información, recabada solo con el fin de poder entender el funcionamiento del back-end puedo decir que se usar sintaxis de variables, expresiones como sumas, y funciones matemáticas, estructuras de control y como se pueden indentar entre sí, y también pude aprender a como crear clases y darles atributos y métodos a dichas clases.

Por otro lado, pude diferenciar el método de comunicación entre cliente-servidor para elegir el necesario según sean los requerimientos del sitio, también ligado a esto pude entender como conectar una base de datos a mi pagina web usando la sintaxis tradicional y la clase ‘conexión’, gracias al mapa cognitivo pude insertar y modificar información, también entendiendo como eliminarla.

Lo que me llamo la atención por lo simple que es a comparación de como se oye, poder hacer que tu pagina web pueda recibir archivos de usuarios y que los mismos usuarios puedan descargar archivos no es tan complicado y fue algo que me llamo mucho la atencion. Y que de hecho es algo a implementar en el saber hacer.

Ya para concluir puedo decir que las sesiones en ambientes web son la base de personalización a gusto del usuario lo cual sube el rating del sitio y ayuda a que la experiencia sea mas personal y por lo tanto el usuario quiera regresara a la pagina

Fuentes

by: Mehdi Achour. (s. f.). PHP: Hypertext Preprocessor.

https://www.php.net/manual/en/

GET - HTTP | MDN. (2023, 10 abril). MDN Web Docs.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET

POST - HTTP | MDN. (2024, 18 julio). MDN Web Docs.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST

PHP: HypertextPreprocessor.(s. f.).

https://www.php.net/manual/en/mysqli.quickstart.php

PHP: Hypertext Preprocessor. (s. f.-b).

https://www.php.net/manual/en/book.pdo.php

Using HTTP cookies - HTTP | MDN. (2024, 8 julio). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies