

MANIPULAÇÃO DE ARQUIVOS

É a capacidade de ler, escrever, criar e apagar arquivos em um sistema.

Por que é importante?

Permite armazenamento de dados, configuração de sistemas, registro de logs, etc.

IMPORTÂNCIA EM MANIPULAÇÃO DE ARQUIVOS

Armazenamento de Dados:

- As aplicações web frequentemente precisam armazenar informações, como configurações, logs, uploads de usuários e dados temporários.
- A manipulação de arquivos permite salvar e recuperar esses dados de forma eficiente.

IMPORTÂNCIA EM MANIPULAÇÃO DE ARQUIVOS

Upload e Download de Arquivos:

- Os usuários podem enviar arquivos (como imagens, documentos ou vídeos) para a aplicação.
- A manipulação de arquivos possibilita o processamento desses uploads e o fornecimento de downloads para os usuários.

IMPORTÂNCIA EM MANIPULAÇÃO DE ARQUIVOS

Geração de Conteúdo Dinâmico:

- Muitas vezes, as aplicações geram conteúdo dinamicamente, como páginas HTML, PDFs ou relatórios.
- A manipulação de arquivos permite criar e atualizar esses conteúdos sob demanda.

IMPORTÂNCIA EM MANIPULAÇÃO DE ARQUIVOS

Cache e Otimização:

- Arquivos estáticos (como CSS, JavaScript e imagens) podem ser armazenados em cache para melhorar o desempenho.
- A manipulação de arquivos facilita o gerenciamento desse cache.

IMPORTÂNCIA EM MANIPULAÇÃO DE ARQUIVOS

Integração com APIs e Serviços Externos:

- Aplicações web frequentemente se comunicam com serviços externos ou APIs.
- A manipulação de arquivos permite a troca de dados com esses serviços.

SUPOORTE A FORMATOS DE ARQUIVO PHP

- Arquivo.txt
- Arquivo.log
- arquivo.xyz
- Arquivo.csv
- Arquivo.gif, arquivo.jpg etc.

FUNÇÃO FOPEN()

Abre um arquivo ou URL e retorna um ponteiro para o arquivo. É necessário especificar o modo de abertura.

FUNÇÃO FOPEN()

Modos de Abertura:

r: Abre para leitura; o ponteiro do arquivo começa no início do arquivo.

w: Abre para escrita; apaga o conteúdo do arquivo ou cria um novo arquivo se ele não existir.

a: Abre para escrita; o ponteiro do arquivo começa no final do arquivo. Cria o arquivo se ele não existir.

x: Cria e abre para escrita; retorna falso se o arquivo já existir.

r+, **w+**, **a+**, **x+**: Abrem o arquivo para leitura e escrita.

EXEMPLO

```
🐘 index.php
1  <?php
2  $file = fopen("exemplo.txt", "r");
3  if ($file) {
4      echo "Arquivo aberto com sucesso.";
5      fclose($file);
6  } else {
7      echo "Falha ao abrir o arquivo.";
8  }
```

FUNÇÃO FCLOSE()

Fecha um arquivo aberto.

Fecha o relacionamento com o arquivo e não fecha o arquivo aberto.

Visualmente não vai acontecer nada.

EXEMPLO

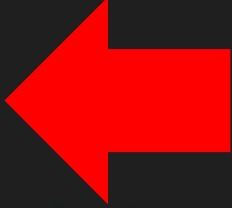

```
$file = fopen("exemplo.txt", "r");  
fclose($file);
```

FUNÇÃO FREAD()

lê o conteúdo de um arquivo aberto.

EXEMPLO

```
index.php
1  <?php
2  $file = fopen("exemplo.txt", "r");
3  ✓ if ($file) {
4      echo "Arquivo aberto com sucesso.<br>";
5      $abrir = fread($file, filesize('exemplo.txt'));
6      fclose($file);
7      echo $abrir;
8  ✓ } else {
9      echo "Falha ao abrir o arquivo.";
10 }
11
```



FUNÇÃO FWRITE()

A função `fwrite()` escreve dados em um arquivo aberto.

EXEMPLO FWRITE()

```
index.php
1  <?php
2  $file = fopen("exemplo.txt", "r");
3  if ($file) {
4      echo "Arquivo aberto com sucesso.<br>";
5      $abrir = fread($file, filesize('exemplo.txt'));
6      fclose($file);
7      echo $abrir;
8  } else {
9      echo "Falha ao abrir o arquivo.";
10 }
11 ?>
12 <?php
13 $file = fopen('exemplo.txt', 'r+');
14 fwrite($file, 'Escrevendo dados no arquivo.');
```

15 \$abrir=fread(\$file, filesize('exemplo.txt'));

16 fclose(\$file);

17 echo \$abrir;

18 ?>

19

FUNÇÃO FILE_GET_CONTENTS()

em PHP é uma maneira simples e eficiente de ler o conteúdo de um arquivo em uma única operação. Ela lê o arquivo inteiro e retorna seu conteúdo como uma string. Essa função é frequentemente usada quando se precisa carregar todo o conteúdo de um arquivo em uma variável para processamento posterior.

DIFERENÇA ENTRE `FILE_GET_CONTENTS()` E `FREAD()`

As funções `file_get_contents()` e `fread()` são utilizadas em PHP para ler arquivos, mas têm diferenças importantes em como são usadas e em que contextos são mais apropriadas:

FILE_GET_CONTENTS()

Uso: Esta função é utilizada para ler o conteúdo completo de um arquivo para uma string.

Simplicidade: É uma função simples de usar, especialmente útil quando se deseja obter todo o conteúdo de um arquivo de uma vez.

Parâmetros: Aceita um parâmetro obrigatório, que é o nome do arquivo a ser lido, e opcionalmente pode aceitar flags adicionais.

Retorno: Retorna o conteúdo do arquivo como uma string, ou false em caso de falha

FREAD()

Uso: É uma função mais flexível que permite ler um número específico de bytes de um arquivo.

Controle de leitura: Pode ser utilizado para ler partes específicas de um arquivo, útil em situações onde se deseja ler em blocos.

Parâmetros: Aceita como parâmetros um ponteiro para o arquivo (obtido geralmente com `fopen()`), e o número de bytes a serem lidos.

Retorno: Retorna os dados lidos como uma string, ou false quando ocorre um erro ou quando chega ao final do arquivo.

RESUMO ENTRE AS DIFERENÇAS FILE_GET_CONTENTS() E FREAD()

Contexto de uso: `file_get_contents()` é geralmente preferida para leituras simples e rápidas de arquivos pequenos e médios, onde a simplicidade é vantajosa.

`fread()` é mais flexível e adequada quando há necessidade de controle fino sobre a leitura ou quando se trabalha com arquivos grandes.

EXEMPLO FILE_GET_CONTENTS()

```
<?php  
$content = file_get_contents('exemplo.txt');  
echo $content;  
?>
```

FUNÇÃO FGETS()

A função `fgets()` lê uma linha de um arquivo.

A função `fgets()` retorna o valor do arquivo referente o tamanho em kb.

EXEMPLO FUNÇÃO FGETS()

```
<?php  
$file = fopen("exemplo.txt", "r");  
$arquivo = fgets($file, 35);  
echo $arquivo;  
?>
```


FUNÇÃO UNLINK()

A função `unlink()` exclui um arquivo.

EXEMPLO UNLINK()

```
<?php  
unlink('exemplo.txt');  
echo "<br>Arquivo deletado com sucesso";  
?>
```

EXEMPLO CRIANDO ARQUIVO DE LOG

log.php

```
1  <?php
2  // Função para registrar eventos em um arquivo de log
3  function logEvento($mensagem) {
4      // Caminho para o arquivo de log
5      $caminhoLog = 'logs/app.log';
6
7      // Formato da mensagem de log
8      //PHP_EOL: serve para colocar para a próxima linha
9      $registro = date('[Y-m-d]') . ' ' . $mensagem . PHP_EOL;
```

EXEMPLO CRIANDO ARQUIVO DE LOG

```
10
11 // Adiciona a mensagem ao arquivo de log
12 // Usamos FILE_APPEND para adicionar ao final do arquivo sem truncar
13 //FILE_APPEND é uma propriedade do file_put_contents e serve para colocar no
14 //final de um arquivo existente sem substituir seu conteúdo anterior.
15 file_put_contents($caminhoLog, $registro, FILE_APPEND);
16 }
17
18 // Exemplo de uso da função de log
19 logEvento('Eduardo Florence Batista.');
```

20

```
21 echo "Evento registrado com sucesso!";
22 ?>
```

EXEMPLO LER ARQUIVO DE LOG

```
1  <?php
2  // Função para ler e exibir o conteúdo do arquivo de log
3  //criou uma função que recebe em parametro o caminho do log
4  ✓ function lerLogs($caminhoArquivo) {
5      // Verifica se o arquivo existe
6  ✓   if (file_exists($caminhoArquivo)) {
7       // Lê o conteúdo do arquivo
8       $conteudo = file_get_contents($caminhoArquivo);
9
10      // Exibe o conteúdo do arquivo na tela
11      //nl2br: serve para quebrar linha - transforma /n em <br>
12      //htmlspecialchars mostrar as informações corretamente com acentuação
13      echo nl2br(htmlspecialchars($conteudo)); // Converte quebras de linha em <br> e escapa ca
14  ✓   } else {
15       echo "Arquivo de log não encontrado.";
16   }
17 }
18
```

EXEMPLO LER ARQUIVO DE LOG

```
19 // Exemplo de uso da função de leitura de logs
20 $arquivoLog = 'logs/app.log';
21 echo "<h2>Logs do Aplicativo</h2>";
22 echo "<pre>";
23 lerLogs($arquivoLog);
24 echo "</pre>";
25 ?>
```

CONCLUSÃO

Em resumo, a manipulação de arquivos é essencial para o funcionamento correto e eficiente das aplicações web, garantindo que elas possam armazenar, processar e fornecer informações de maneira confiável.