UNIPAC
Barbacena

Centro Universitário Presidente Antônio Carlos
Programação para Internet

Manipulação de Arquivos

Felipe Roncalli de Paula Carneiro
felipecarneiro@unipac.br

O que vamos aprender nessa aula

- Introdução;
- Leitura de Arquivos;
- Escrita em Arquivos;
- fopen();
- Abertura e fechamento de Arquivos;
- Manipulação de Diretórios;

Introdução

Manipulação de arquivos em programação web não é apenas uma tarefa técnica, mas uma habilidade estratégica que potencializa a capacidade dos desenvolvedores de criar aplicações robustas e dinâmicas. Ao explorarmos as funcionalidades, estaremos equipando-nos com ferramentas poderosas para enfrentar os desafios do desenvolvimento web moderno.

Introdução

Ao compreender e dominar a manipulação de arquivos em PHP, os desenvolvedores ampliam suas capacidades para criar aplicações web mais flexíveis, dinâmicas e adaptáveis, proporcionando uma base sólida para o desenvolvimento de sistemas robustos e eficientes.

Leitura de Arquivos

A leitura de arquivos em PHP é uma operação fundamental para muitas aplicações web. PHP oferece diversas maneiras de realizar a leitura de arquivos, e vamos abordar mais especificamente 3.

- file()
- fgets()
- file_get_contents()

Leitura de Arquivos - file

A função file() simplifica o processo de leitura de todo o conteúdo de um arquivo para um array. Cada elemento do array representa uma linha do arquivo.

```php
<?php
// Exemplo de leitura de arquivo com file()
$nomeArquivo = 'exemplo.txt';

// Lê o conteúdo do arquivo e armazena em um array
$linhasDoArquivo = file($nomeArquivo);

// Exibe o conteúdo do array (cada elemento representa uma linha do arquivo)
foreach ($linhasDoArquivo as $linha) {
    echo $linha . "<br>";
}
?>
```

Leitura de Arquivos - fgets

Para processar um arquivo linha a linha, você pode usar a função fgets() em conjunto com feof() para verificar o final do arquivo.

```php
<?php
// Exemplo de leitura de arquivo linha a linha
$nomeArquivo = 'exemplo.txt';
// Abre o arquivo para leitura
$arquivo = fopen($nomeArquivo, 'r');
// Lê o arquivo linha a linha até o final
while (!feof($arquivo)) {
    $linha = fgets($arquivo);
    echo $linha . "<br>";
}
// Fecha o arquivo
fclose($arquivo);
?>
```

Leitura de Arquivo - file_get_contents

Se a intenção é obter todo o conteúdo do arquivo como uma string, a função file_get_contents() é uma opção eficiente.

```php
<?php
// Exemplo de leitura do conteúdo completo de um arquivo
$nomeArquivo = 'exemplo.txt';

// Lê o conteúdo completo do arquivo
$conteudo = file_get_contents($nomeArquivo);

// Exibe o conteúdo
echo $conteudo;
?>
```

Leitura de Arquivo

É importante mencionar que ao trabalhar com a leitura de arquivos, é crucial considerar a segurança. Validar e filtrar os dados provenientes de arquivos é essencial para prevenir ataques maliciosos, especialmente ao lidar com uploads de usuários.

Lembre-se também de verificar se o arquivo que está sendo acessado existe e se o script PHP tem as permissões necessárias para ler o arquivo.

Escrita em Arquivo

A escrita em arquivos é uma operação comum ao desenvolver aplicações web, permitindo que você armazene dados gerados dinamicamente, configure configurações do sistema e muito mais. Aqui estão algumas maneiras de realizar a escrita em arquivos:

- file_put_contents()
- fopen(), fwrite(), e fclose()

Escrita em Arquivo

A escrita em arquivos é uma operação comum ao desenvolver aplicações web, permitindo que você armazene dados gerados dinamicamente, configure configurações do sistema e muito mais. Aqui estão algumas maneiras de realizar a escrita em arquivos:

- file_put_contents()
- fopen(), fwrite(), e fclose()

Escrita em Arquivo - file_put_contents()

A função file_put_contents() é uma maneira concisa de escrever dados em um arquivo. Se o arquivo não existir, a função cria um novo; se existir, ela sobrescreve o conteúdo.

```php
<?php
// Exemplo de escrita em arquivo com file_put_contents()
$nomeArquivo = 'exemplo.txt';

// Dados a serem escritos no arquivo
$dados = "Conteúdo a ser escrito no arquivo.";

// Escreve os dados no arquivo
file_put_contents($nomeArquivo, $dados);

echo "Escrita concluída com sucesso.";
?>
```

Escrita em Arquivo - fopen(), fwrite(), e fclose()

Se precisar de mais controle sobre o processo de escrita, você pode utilizar as funções fopen(), fwrite(), e fclose().

```php
<?php
// Exemplo de escrita em arquivo com fopen(), fwrite() e fclose()
$nomeArquivo = 'exemplo.txt';
// Abre o arquivo para escrita (o parâmetro 'w' indica escrita)
$arquivo = fopen($nomeArquivo, 'w');
// Dados a serem escritos no arquivo
$dados = "Conteúdo a ser escrito no arquivo.";
// Escreve os dados no arquivo
fwrite($arquivo, $dados);
// Fecha o arquivo
fclose($arquivo);
echo "Escrita concluída com sucesso.";
?>
```

Escrita em Arquivo - fopen(), fwrite(), e fclose()

Se você deseja adicionar conteúdo ao final de um arquivo sem apagar o conteúdo existente, pode utilizar a opção 'a' ao abrir o arquivo.

```php
<?php
// Exemplo de adição de conteúdo ao final do arquivo
$nomeArquivo = 'exemplo.txt';
// Abre o arquivo para adicionar conteúdo ao final (o parâmetro 'a' indica append)
$arquivo = fopen($nomeArquivo, 'a');
// Dados a serem adicionados ao final do arquivo
$dados = "Conteúdo a ser adicionado ao final do arquivo.";
// Escreve os dados no final do arquivo
fwrite($arquivo, $dados);
// Fecha o arquivo
fclose($arquivo);
echo "Adição de conteúdo concluída com sucesso.";
?>
```

Sobre a Função fopen()

A função fopen() em PHP é utilizada para abrir um arquivo e retorna um identificador de arquivo (resource) que é necessário para operações subsequentes no arquivo. Essa função tem a seguinte assinatura:

```php
<?php
fopen(string $nome_do_arquivo, string $modo[, bool $uso_incluir_caminho = false[, resource $contexto]]);
?>
```

Sobre a Função fopen()

$nome_do_arquivo (obrigatório):

- Uma string que representa o caminho e o nome do arquivo a ser aberto. Pode ser um caminho absoluto ou relativo.

$modo (obrigatório):

- Uma string que define o modo como o arquivo será aberto. Os modos mais comuns incluem:

  'r': Abre o arquivo para leitura. O ponteiro do arquivo é colocado no início do arquivo.

  'w': Abre o arquivo para escrita. Se o arquivo já existir, seu conteúdo é apagado. Se não existir, um novo arquivo é criado.

  'a': Abre o arquivo para escrita, adicionando o conteúdo ao final do arquivo. Se o arquivo não existir, um novo arquivo é criado.

  'x': Cria e abre o arquivo para escrita. Se o arquivo já existir, a operação falhará.

Sobre a Função fopen()

$uso_incluir_caminho (opcional, padrão = false):

- Um parâmetro booleano que indica se o caminho do arquivo fornecido em $nome_do_arquivo deve ser incluído no identificador de arquivo retornado. Se true, o caminho será incluído; se false, apenas o nome do arquivo será incluído.

$contexto (opcional):

- Um contexto pode ser especificado para personalizar o comportamento da função, como configurações de contexto criadas com stream_context_create().

Abertura de Arquivos

A função fopen() é usada para abrir um arquivo e retorna um identificador de arquivo (resource) que é necessário para operações subsequentes no arquivo.

```php
<?php
$nomeArquivo = 'exemplo.txt';
// Abre o arquivo para leitura
$arquivo = fopen($nomeArquivo, 'r');
// Verifica se a abertura do arquivo foi bem-sucedida
if ($arquivo) {
    echo "Arquivo aberto com sucesso!";
    // Realize operações no arquivo aqui...
    // Não se esqueça de fechar o arquivo após as operações
    fclose($arquivo);
} else {
    echo "Falha ao abrir o arquivo.";
}
?>
```

Fechamento de Arquivos

A função fclose() é usada para fechar o identificador de arquivo aberto anteriormente com fopen(). Isso é importante para liberar os recursos associados ao arquivo.

```php
<?php
$nomeArquivo = 'exemplo.txt';
// Abre o arquivo para leitura
$arquivo = fopen($nomeArquivo, 'r');
// Verifica se a abertura do arquivo foi bem-sucedida
if ($arquivo) {
    echo "Arquivo aberto com sucesso!";
    // Realize operações no arquivo aqui...
    // Fecha o arquivo após as operações
    fclose($arquivo);
    echo "Arquivo fechado com sucesso!";
} else {
    echo "Falha ao abrir o arquivo.";
}
?>
```

Boas Práticas

- Verificação de Abertura:
  - Sempre verifique se a abertura do arquivo foi bem-sucedida antes de prosseguir com operações no arquivo.
- Fechamento Adequado:
  - Certifique-se de fechar o arquivo usando fclose() assim que as operações no arquivo forem concluídas. Isso libera os recursos associados ao arquivo.

Boas Práticas

- Controle de Erros:
  - Utilize estruturas de controle de erro (como try, catch para exceções) para lidar com falhas durante a abertura do arquivo.
- Contexto Opcional:
  - Considere o uso de um contexto opcional ao abrir o arquivo para personalizar o comportamento, especialmente em operações mais avançadas.
  -

Criando um Diretório

A função mkdir() é usada para criar um novo diretório. Ela aceita o nome do diretório a ser criado e, opcionalmente, as permissões do diretório.

```php
<?php
$nomeDiretorio = 'novo_diretorio';

// Cria um novo diretório
if (mkdir($nomeDiretorio, 0755)) {
    echo "Diretório criado com sucesso!";
} else {
    echo "Falha ao criar o diretório.";
}
?>
```

Removendo um Diretório

A função rmdir() é usada para remover (excluir) um diretório vazio.

```php
<?php
$nomeDiretorio = 'diretorio_a_ser_removido';
// Remove o diretório
if (rmdir($nomeDiretorio)) {
    echo "Diretório removido com sucesso!";
} else {
    echo "Falha ao remover o diretório.";
}
?>
```

Listando Arquivos em um Diretório

A função scandir() é usada para obter uma lista de arquivos e diretórios em um diretório.

```php
<?php
$nomeDiretorio = 'meu_diretorio';
// Lista todos os arquivos e diretórios no diretório
$conteudo = scandir($nomeDiretorio);
// Exibe o conteúdo
foreach ($conteudo as $item) {
    echo $item . "<br>";
}
?>
```

Renomeando um Diretório

A função rename() é usada para renomear um diretório.

```php
<?php
$antigoNomeDiretorio = 'antigo_nome';
$novoNomeDiretorio = 'novo_nome';
// Renomeia o diretório
if (rename($antigoNomeDiretorio, $novoNomeDiretorio)) {
    echo "Diretório renomeado com sucesso!";
} else {
    echo "Falha ao renomear o diretório.";
}
?>
```

Verificando se um Diretório Existe

A função is_dir() é usada para verificar se um determinado caminho é um diretório.

```php
<?php
$nomeDiretorio = 'meu_diretorio';
// Verifica se é um diretório
if (is_dir($nomeDiretorio)) {
    echo "É um diretório!";
} else {
    echo "Não é um diretório.";
}
?>
```

Resumo

Estas são apenas algumas das funções básicas para manipulação de diretórios em PHP. Ao trabalhar com manipulação de arquivos e diretórios, é essencial garantir que o script tenha as permissões adequadas para executar as operações desejadas e realizar verificações para evitar erros inesperados.

Dúvidas??