

MÓDULO

PROGRAMAÇÃO PHP

UNIDADE

FUNÇÕES AVANÇADAS EM PHP

ÍNDICE

OBJETIVOS.....	3
INTRODUÇÃO.....	4
1. COOKIES	5
2. ENVIO DE E-MAILS COM PHP.....	10
3. TRABALHAR COM XML	13
4. TRABALHAR COM FICHEIROS DE LEITURA E ESCRITA.....	22
CONCLUSÃO	35
AUTOAVALIAÇÃO	37
SOLUÇÕES.....	41
PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO.....	42
BIBLIOGRAFIA	43

OBJETIVOS

Com esta unidade didática, pretende-se que desenvolva os seguintes objetivos de aprendizagem:

- Gerir cookies com PHP.
- Enviar e-mails com servidores externos.
- Trabalhar com ficheiros XML.
- Ler e guardar ficheiros.
- Criar formulários para enviar ficheiros para o servidor.

INTRODUÇÃO

Nesta unidade didática vão ser abordadas várias funcionalidades avançadas que o PHP permite, tais como: trabalhar com cookies, enviar e-mails de PHP (utilizando servidores de e-mail externos), trabalhar com XML (leitura e processamento), e ler e escrever ficheiros (incluindo fazer upload de ficheiros para o servidor).

1. COOKIES

Os cookies são um sistema aceite pelos navegadores (desde que esteja ativo), através do qual se armazenam certas informações que podem ser enviadas ao computador local, permitindo rastrear se o utilizador esteve na página web anteriormente.

Os cookies são parte dos cabeçalhos HTTP, ou seja, devem ser chamados antes de qualquer código HTML.

A linguagem PHP permite a gestão, criação e leitura de cookies; o sistema trata-os como variáveis globais e podem ser localizados dentro do array `$_COOKIE`.

O PHP utiliza a função **setcookie()** para definir o cookie para o navegador e recebe os seguintes parâmetros:

- **name**: o nome que o cookie terá.
- **value**: o valor que se pretender dar ao cookie.
- **expire**: o prazo do cookie, indicado em segundos, e a data Unix, ou seja, os segundos que passaram desde 1 de janeiro de 1970. Por exemplo, para passar um prazo de 30 dias, usar-se-ia o seguinte parâmetro:

```
time()+60*60*24*30 //time devolve a data unix atual
```

Se não for indicado qualquer valor para expirar, o cookie durará apenas enquanto o navegador estiver aberto.

- **path**: neste parâmetro será definida que parte da página web terá acesso ao cookie. Por exemplo, se se colocar `"/portfolio/"`, apenas os ficheiros PHP dentro dessa pasta terão acesso; o padrão é o diretório raiz.
- **domain**: o nome do domínio é indicado aqui. É importante referir que, se se indicar `www.domain.com`, o cookie será válido apenas para o domínio e não para subdomínios como, por exemplo, `dados.domain.com`.
- **secure**: se se indicar o valor `true` neste parâmetro, o cookie só estará acessível a partir de um servidor HTTPS seguro.
- **Httponly**: se se indicar o valor `true`, significa que só será acessível pelo protocolo HTTP e não por linguagens de script como JavaScript. Este parâmetro é útil para reduzir a possibilidade de ataques XSS.

Exemplo

Neste exemplo irá criar-se um formulário num website para enviar os dados para esse mesmo website. Graças ao uso de cookies, garantir-se-á que, se o utilizador enviar o formulário, sair da página e entrar novamente dentro de uma semana, os dados aparecerão.

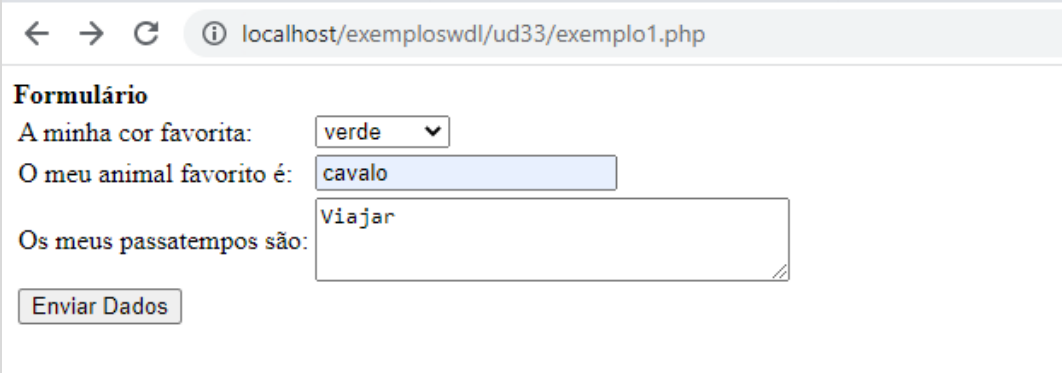
```
<?php
if(isset($_POST['enviar'])) {
    error_reporting(0);
    // guardar os cookies durante uma semana
    // hora atual + ( 7dias * 24horas * 60minutos * 60 segundos )
    $prazo = time() + (7 * 24 * 60 * 60);
    setcookie('corFavorita',$_POST['corFavorita'], $prazo);
    setcookie('animalFavorito',$_POST['animalFavorito'], $prazo);
    setcookie('passatempo',$_POST['passatempo'], $prazo);
    // mostrar os valores enviados
    echo 'Enviou o formulário! ';
    echo '<br>';
    echo 'Os dados enviados foram: ';
    echo '<ul><li>A minha cor favorita é
    '.$_POST['corFavorita'].'</li>';
```



```
echo '<li>O meu animal favorito é  
' . $_POST['animalFavorito'] . '</li>';  
  
echo '<li>Os meus passatempos são  
' . htmlentities($_POST['passatempo']) . '</li></ul>';  
  
} else {  
    // se não for nada enviado, será mostrado o formulário  
  
    echo '  
<strong>Formulário</strong>  
<br />  
<form action="" method="post">  
<table>  
<tr>  
<td>A minha cor favorita: </td>  
<td>  
<select name="corFavorita" id="corFavorita">  
<option value="laranja" ' . ($_COOKIE['corFavorita'] == "laranja" ?  
    "selected" : "") . ' >laranja</option>  
<option value="verde" ' . ($_COOKIE['corFavorita'] == "verde" ?  
    "selected" : "") . ' >verde</option>  
<option value="azul" ' . ($_COOKIE['corFavorita'] == "azul" ?  
    "selected" : "") . ' >azul</option>  
<option value="amarelo" ' . ($_COOKIE['corFavorita'] == "amarelo" ?  
    "selected" : "") . ' >amarelo</option>  
<option value="vermelho" ' . ($_COOKIE['corFavorita'] == "vermelho" ?  
    "selected" : "") . ' >vermelho</option>  
</select>  
</td>  
</tr>  
<tr>  
<td>O meu animal favorito é: </td>  
<td><input type="text" name="animalFavorito" id="animalFavorito"  
    value="' . htmlentities($_COOKIE['passatempo']) . '" size="20"  
    maxlength="50" />  
</td>  
</tr>
```

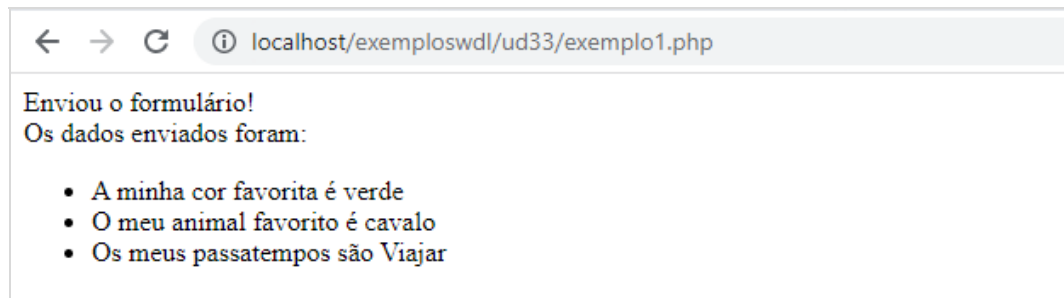
```
<tr>
<td>Os meus passatempos são:</td>
<td><textarea name="passatempo" id="passatempo" cols="35"
rows="3">'.htmlentities($_COOKIE['passatempo']).'</textarea></td>
</tr>
<tr>
<td colspan="2"><input type="submit" id="enviar" name="enviar"
value="Enviar Dados"></td></tr>
</table>
</form>';
}
?>
```

Na imagem abaixo pode ver-se o resultado inicial da página já com dados inseridos.



The screenshot shows a web browser window with the address bar displaying 'localhost/exemploswdl/ud33/exemplo1.php'. The page content is a form titled 'Formulário'. It contains three input fields: a dropdown menu for 'A minha cor favorita:' with 'verde' selected, a text input for 'O meu animal favorito é:' with 'cavalo' entered, and a text area for 'Os meus passatempos são:' with 'Viajar' entered. Below these fields is a button labeled 'Enviar Dados'.


Depois de se clicar no botão “Enviar Dados”, a visualização deverá ser similar à que se apresenta na imagem abaixo.



Para esvaziar um cookie é necessário sobrescrevê-lo, ou seja, se se pretender excluir um cookie, deverá sobrescrever-se o nome, indicando um valor expire menor do que a data atual.

2. ENVIO DE E-MAILS COM PHP

O PHP tem uma biblioteca/API gratuita que permite enviar e-mails, desde que tenha um servidor de e-mail.


+ Informações

Esta biblioteca chama-se **phpmailer** e encontra-se em: <https://github.com/PHPMailer/PHPMailer>.

Para o envio de e-mails serão utilizadas duas bibliotecas:

- `phpmailer.php`.
- `smtp.php`.

Estas já possuem as classes com as propriedades e métodos necessários para o envio de e-mails.

A maneira mais fácil de explicar como enviar um e-mail é com um exemplo prático em que são solicitados ao utilizador vários dados através de um formulário. Será utilizada uma conta do Google Gmail (gratuita) para enviar para essa mesma conta. Neste caso, no formulário será solicitado que o utilizador deixe os dados necessários para que seja possível contactá-lo.

Exemplo

```
<html>
<head>
</head>
<body>
<?php
if ($_GET["acc"] == "envioMail"){
//carregar as classes
require 'src/Exception.php';
require 'src/PHPMailer.php';
require 'src/SMTP.php';
//criar o objeto
$mail = new PHPMailer(true);
//passar os parametros
$mail->IsSMTP();
$mail->SMTPAuth = true;
$mail->SMTPSecure = "ssl";
$mail->Host = "smtp.gmail.com";
$mail->Port = 465;
$mail->Username = "xxxxxxx@gmail.com"; // a sua conta
$mail->Password = "senha"; // a sua senha
$mail->SetFrom('xxxxxxx@gmail.com', 'Nome');
//preencher o email com conteudo do formulário
$mail->Subject = "Contacto através do Formulario - WEB";
$conteudo = "Nome: ".$_POST["nome"]."<br/>";
$conteudo .= "Apelidos: ".$_POST["apelidos"]."<br/>";
$conteudo .= "Email: ".$_POST["email"]."<br/>";
$conteudo .= "Telemóvel: ".$_POST["telemovel"]."<br/>";
$conteudo .= "Mensagem: ".$_POST["mensagem"]."<br/>";
$mail->MsgHTML($conteudo);
$mail->AltBody = "Contacto desde Formulario\nxxxxx.";
```

```
//indicar a conta a enviar
$mail->AddAddress('xxxxxxx@gmail.com', 'Contacto');
//enviar verificando que funciona
if(!$mail->Send()) {
    echo "Error: " . $mail->ErrorInfo;
} else {
    echo "<br/><br/>Obrigado pelo contacto, terá uma resposta o mais
    brevemente possível.<br/>";
}
}else{
?>
<!--formulario de envio -->
<form name="envio" method="post" action="">
Nome* : <input type="text" name="nome" id="nome" /><br/>
Apelidos* :<input type="text" name="apelidos" id="apelidos"/><br/>
E-mail* : <input type="text" name="email" id="email" /><br/>
Telemóvel* : <input type="text" name="telemovel"
id="telemovel /><br/>
Mensagem : <br/>
<textarea cols="60" rows="8" name="mensagem" id="mensagem">
</textarea>
<input type="hidden" name="acc" value="envioMail"/>
<br/><input type="submit" name="bot" value="Enviar" />
</form>
<?php
}
?>
</body></html>
```

3. TRABALHAR COM XML

Em programação, é bastante comum ter de trabalhar com ficheiros XML. Estes podem conter informações de notícias, eventos, ou simplesmente ser a base de dados de um website.

XML significa Extensible Markup Language e é uma tecnologia simples que se baseia em tags e é utilizada para passar informações entre sistemas. Dada a sua grande padronização, as suas vantagens são não precisar de um motor como base de dados e ser muito fácil de criar. A sua principal desvantagem é tratar-se de um ficheiro plano, sem nenhum tipo de índice que permita percorrê-lo mais rapidamente.

Esta é uma linguagem amplamente utilizada no mundo da Internet. Por exemplo, os sitemaps (ou “mapas do site”), que são enviados para motores de busca como o Google ou o Yahoo são criados com esta linguagem.

A estrutura desses ficheiros está muito bem definida e baseia-se na criação de uma árvore hierárquica de níveis.

Segue-se um exemplo de um ficheiro XML:

```
<?xml version='1.0' standalone='yes'?>
<filmes>
  <filme>
    <titulo>Die Hard - Assalto ao Arranha-Céus</titulo>
```

```
<personagens>
<personagem>
<nome>John McClane</nome>
<ator> Bruce Willis </ator>
</personagem>
<personagem>
<nome> Hans Gruber </nome>
<ator> Alan Rickman </ator>
</personagem>
</personagens>
<argumento>
Um policia que fica preso num prédio que está a ser roubado e tem
de salvar a situação.
</argumento>
<grandes-linhas>
<linha>Filme policial com muita ação</linha>
</grandes-linhas>
<pontuacao tipo="polegares">8</pontuacao>
<pontuacao tipo="estrelas">5</pontuacao>
</filme>
</filmes>
```

Como pode verificar no exemplo, é uma estrutura em árvore fácil de seguir.

No PHP existe uma extensão chamada **SimpleXML**, que fornece ferramentas bastante simples para trabalhar com ficheiros XML.

Para carregar o conteúdo de um ficheiro XML externo num array, é utilizada a função **simplexml_load_file**, que permite carregar o ficheiro XML que é passado como um parâmetro.

Este exemplo carrega o ficheiro XML criado anteriormente e exhibe o seu conteúdo na página web.


```

<html>
<body>
<p>
<?php
// a função permite saber se o ficheiro existe ou não
if (file_exists('filmes.xml')) {
$xml = simplexml_load_file('filmes.xml');
// mostrar o conteúdo do array
print_r($xml);
} else {
exit('Erro ao abrir filmes.xml.');
```

No caso de ser utilizada uma variável string para passar o conteúdo do suposto XML, é utilizado o construtor da classe SimpleXMLElement:

```

<?php
$xmlstr = <<<XML
<?xml version='1.0' standalone='yes'?>
<filmes>
<filme>
<titulo>Die Hard - Assalto ao Arranha-Céus</titulo>
<personagens>
<personagem>
<nome>John McClane</nome>
<ator> Bruce Willis </ator>
</personagem>
<personagem>
```

```
<nome> Hans Gruber </nome>
<ator> Alan Rickman </ator>
</personagem>
</personagens>
<argumento>
Um policia que fica preso num prédio que está a ser roubado e tem
de salvar a situação.
</argumento>
<grandes-linhas>
<linha>Filme policial com muita ação</linha>
</grandes-linhas>
<pontuacao tipo="polegares">8</pontuacao>
<pontuacao tipo="estrelas">5</pontuacao>
</filme>
</filmes>
XML;
$xml = new SimpleXMLElement($xmlstr);
print_r($xml);
?>
```

O array obtido é um array PHP normal e é possível aceder aos seus valores como se fazia normalmente, ou seja, é possível percorrê-lo utilizando o foreach.

Por exemplo, se no caso anterior se pretendesse remover apenas as personagens do filme, seria:

```
<?php
foreach ($xml->filme->personagens->personagem as $personagem) {
echo $personagem->nome, ' interpretado por ', $personagem->ator;
}
?>
```

Ao olhar atentamente para o XML, é possível verificar que é utilizada uma iteração de elemento para fornecer dois tipos diferentes de pontuação e, para diferenciá-los, é criado um atributo para esse elemento filho do HTML. Esses atributos também são acessíveis a partir do array criado:

```
<?php
foreach ($xml->filme[0]->pontuacao as $pontuacao) {
    switch((string) $pontuacao['tipo']) {
        // considera os atributos como índices do elemento
        case 'polegares':
            echo $pontuacao, ' votos positivos';
            break;
        case 'estrelas':
            echo $pontuacao, ' estrelas';
            break;
    }
}
?>
```

Dentro da classe SimpleXML são encontradas algumas ferramentas que facilitam muito o trabalho a ser feito. Uma dessas ferramentas é a função **xpath**.

Esta função permite encontrar todos os elementos com o mesmo rótulo, mesmo que não se saiba onde é que estes se encontram na hierarquia da árvore:

```
<?php
foreach ($xml->xpath('//personagem') as $personagem) {
    echo $personagem->nome . ' interpretado por ' .
    $personagem->ator,;
}
?>
```

Os valores do array também podem ser modificados utilizando a atribuição (=).

```
<?php
$xml->filme[0]->personagens->personagem[0]->nome = 'Tenente
McClane;

?>
```

A classe SimpleXML, desde a versão 5.1 do PHP, também permite adicionar ramos e atributos à árvore. Para isso, são utilizadas duas funções: **addChild** para inserir novos ramos e **addAttribute** para inserir atributos no ramo.

Segue-se um exemplo de como poderiam ser inseridos novos ramos na árvore do código que tem sido desenvolvido neste ponto:

```
<?php
//mapeando o texto XML como no primeiro exemplo
$xml = new SimpleXMLElement($xmlstr);
// inserir um ramo nos personagens criando uma nova personagem
// note que foi atribuído o novo ramo a uma variável 'novoRamo'
$novoRamo = $xml->filme[0]->personagens->addChild
('personagem');
// criar em 'novoRamo' duas sub-ramificações com os valores
$novoRamo ->addChild('nome', ' Al Powell');
$novoRamo ->addChild('ator', ' Reginald VelJohnson');
//adicionar outro ramo, mas desta vez como uma pontuação com o
valor
//'Todos os públicos', voltar a criar um ramo de uma variável
$novaPontuacao = $xml->filme[0]->addChild
('pontuacao', 'Todos os públicos');
//no ramo criado anteriormente, adicionar um atributo chamado
'tipo' com o valor 'classificação'
$novaPontuacao ->addAttribute('tipo', 'classificacao');
//mostrar na página web
print_r($xml)

?>
```

Em alguns casos, os ficheiros XML podem ser externos ou ser tão extensos que é difícil verificar se estão corretos ou não. No PHP existe uma funcionalidade chamada **libxml** que permite controlar esses erros; para isso o serviço de erros é inicializado com a linha:

```
libxml_use_internal_errors(true);
```

Esta linha é suficiente para recolher o resultado do carregamento do ficheiro ou do texto XML numa variável e verificar se há algum erro.

Neste exemplo, iria ser carregado numa string que contém um texto XML mal formado e que criaria um erro quando se tentasse trabalhar com ele; segue-se, então, um exemplo de como se poderia detetar o problema antes que o mesmo acontecesse.

Exemplo

```
<html>
<body><p>
<?php
libxml_use_internal_errors(true);
$xml = simplexml_load_string( "<?xml version='1.0'><malformado>
<xml><isto não é assim></quebra>");
if (!$xml) {
echo "Erro ao carregar o XML<br/>";
foreach(libxml_get_errors() as $error) {
echo "<br/>&nbsp;-&nbsp;", $error->message;
}
}
?>
</p></body>
</html>
```

Neste exemplo, é utilizada uma nova forma de carregar dados XML num array; neste caso, foi utilizada a função **simplexml_load_string**, que permite carregar diretamente a string.

Outras funções da classe SimpleXML, ainda não referidas nesta unidade, são:

- **asXML**: devolve uma string no formato XML, se um ficheiro for especificado como parâmetro; as alterações serão guardadas no ficheiro (caso tenham ocorrido).

```
<?php
$xml = simplexml_load_file('filmes.xml');
// mostrar o conteúdo num array
$ramo = $xml->filme[0]->personagens->addChild('personagem');
$ramo->addChild('nome', 'Al Powel');
$ramo->addChild('ator', 'Dimitrick Vodchenco');
//criar um novo ficheiro filmes2.xml com a mudança
$xml->asXML('filmes2.xml');
?>
```

- **children**: procura e devolve um array com todos os filhos do ramo passados como parâmetro; se se deixar o parâmetro vazio, este será retirado da raiz.

```
<?php
foreach ($xml->children('personagem') as $vetor) {
}
?>
```

- **count**: devolve os elementos de um elemento num array (funciona apenas a partir da versão 5.3.0 do PHP).

```
<?php
$xml = <<<EOF
<carros>
<carro marca="ferrari">
```

```

<motor/>
<portas/>
<cavalos/>
</carro>
<carro marca="seat">
<motor/>
<portas/>
<cavalos/>
<cilindrada/>
<pneus/>
</carro>
</carros>
EOF;
$xml = new SimpleXMLElement($xml);
foreach ($xml as $carro) {
printf("%s tem %d características.\n", $carro['marca'], $carro-
>count());
}?>

```

- **getName:** devolve o nome do elemento. O exemplo a seguir mostra todos os nomes de itens do exemplo do filme utilizado anteriormente.

```

<?php
$sxe = new SimpleXMLElement($xmlstr);
echo $sxe->getName() . "<br/>";
foreach ($sxe->children() as $filho){
echo $filho->getName() . "<br/>";
}
?>

```

4. TRABALHAR COM FICHEIROS DE LEITURA E ESCRITA

A linguagem PHP permite aceder aos ficheiros no servidor graças a bibliotecas já integradas dentro da linguagem. Neste ponto da unidade, verá quais são as suas principais funções:

- **pathinfo**: devolve um array com as informações do caminho do ficheiro.

```
<?php
$partesCaminho = pathinfo('/www/php/filmes.xml');
echo $partesCaminho['dirname']. "<br/>"; // '/www/php
echo $partesCaminho['basename']. "<br/>"; // 'filmes.xml'
echo $partesCaminho['extension']. "<br/>"; // 'xml'
echo $partesCaminho['filename']. "<br/>"; // 'filmes'
?>
```

A visualização será:



- **copy**: copia um ficheiro para outro. Se o ficheiro de destino já existir, este será substituído.

```
<?php
$ficheiro = 'filmes.xml';
$novoFicheiro = 'filmes2.xml';
if (!copy($ficheiro, $novoFicheiro)) {
    echo "Erro ao copiar $ficheiro...\n";
}
?>
```

- **dirname**: devolve o diretório pai do caminho especificado.

```
<?php
echo dirname("/www/php"); //devolve /www
?>
```

- **file_exists**: verifica se o ficheiro ou diretório passado pelo parâmetro existe, devolvendo true se existir.

```
<?php
$ficheiro = 'filmes8.xml';
if (file_exists($ficheiro)) {
    echo "$nomeFicheiro existe";
} else {
    echo "$nomeFicheiro não existe";
}
?>
```

- **fopen**: abre um ficheiro ou URL que recebe como parâmetro. Também obtém um segundo parâmetro que informa como abrir o ficheiro. Os valores possíveis deste segundo parâmetro são:

Modo	Descrição
'r'	Abre no modo somente de leitura e posiciona-se no início do ficheiro.
'r+'	Abre para leitura e escrita e posiciona-se no início do ficheiro.
'w'	Abre o ficheiro somente para gravação, coloca-se no início do ficheiro e corta-o para o comprimento 0; se não existir, irá criá-lo.
'w+'	Abre o ficheiro para leitura e gravação, coloca-se no início do arquivo e corta-o para o comprimento 0; se não existir, irá criá-lo.
'a'	Abre somente para escrita e coloca o ponteiro no final do ficheiro.
'a+'	Abre em leitura e gravação e coloca o ponteiro no final do ficheiro.
'x'	Cria e abre o ficheiro na escrita apenas colocando o ponteiro no início do ficheiro; se já existir, irá criar uma mensagem de erro.
'x+'	Cria e abre o ficheiro no modo de leitura e gravação; se já existir, comporta-se como "x".
'c'	Abre o ficheiro somente como leitura; se o ficheiro não existir, é criado e o ponteiro é posicionado no início.
'c+'	Abre o ficheiro para leitura e escrita; se não existir, comporta-se como "c".
'e'	Define o sinalizador "close-on-exec" no descritor de ficheiro aberto. Disponível apenas em PHP compilado em sistemas em conformidade com POSIX.1-2008. Opção adicionada em PHP7.

```
<?php
$ficheiro = fopen("/www/php/filmes.txt", "r");
?>
```

- **fclose**: fecha o ponteiro para o ficheiro aberto. Para isso, é passada a variável que se encontra designada para conter o ficheiro.

```
<?php
$ficheiro = fopen("/www/php/filmes.txt", "r");
fclose($ficheiro);
?>
```

- **file**: transfere um ficheiro passado como parâmetro para um array.

```
<?php
$linhas = file("filmes.xml");

// percorrer o array, exibir o texto como tal e mostrar os números
das linhas também

foreach ($linhas as $numLinha => $linha) {
    echo "Linha #<b>{$numLinha}</b> : " . $linha . "<br />\n";
}
?>
```

- **filesize**: devolve o tamanho do ficheiro passado como parâmetro em bytes ou false, se ocorrer um erro.

```
<?php
echo filesize("filmes.xml");
?>
```

- **filetype**: devolve o tipo do ficheiro passado pelo parâmetro.

```
<?php
echo '../php/ é '.filetype('../php/').'<br/>'; // file
echo 'filmes.xml é '.filetype('filmes.xml'); // dir
?>
```

- **fread**: lê o ficheiro em modo binário. Recebe dois parâmetros: o identificador do ficheiro aberto e o comprimento dos bytes a serem lidos.

```
<?php
// colocar o conteúdo de um ficheiro numa string
$nomeFicheiro = "filmes.xml";
$gestor = fopen($nomeFicheiro, "r");
$conteudo = fread($gestor, filesize($nomeFicheiro));
echo $conteudo;
fclose($gestor);
```

```
?>
```

- **fwrite**: escreve no ficheiro em modo binário, passando como parâmetro o identificador do ficheiro aberto e a string a ser escrita.

```
<?php
$ficheiro = 'filmes2.xml';
$conteudo = " Muitos filmes ficam para ver ";
// verificar se existe e pode ser escrito
if (is_writable($ficheiro)) {
    // O ponteiro está no final do ficheiro ao abri-lo com 'a'
    if (!$ficheiroID = fopen($ficheiro, 'a')) {
        echo " Não é possível abrir o ficheiro($ficheiro)";
    }
    // Escrever $conteudo no ficheiro aberto.
    if (fwrite($ficheiroID, $conteudo) === FALSE) {
        echo "Não é possível escrever no ficheiro ($ficheiro)";
    }
    echo " Sucesso, ($conteudo) foi escrito no ficheiro ($ficheiro)";
    fclose($ficheiroID);
} else {
    echo "O ficheiro $ficheiro não é válido";
}
?>
```

- **mkdir**: cria um diretório passado pelo parâmetro, desde que tenha permissão para fazê-lo.

```
<?php
mkdir("diretorio", 0700);
?>
```

- **move_uploaded_file**: move o ficheiro enviado pela página web para o caminho especificado. O exemplo a seguir recolhe um ficheiro enviado de um formulário:

```
<form action="<?=$PHP_SELF?>" method="post"
enctype="multipart/form-data" name="form1">

<p align="center">Ficheiro

<input name="ficheiro" type="file" id="ficheiro">

</p>

<p align="center">

<input name="botao" type="submit" id="botao" value="Enviar">

</p>

</form>

<?php
if($botao) {
$uploadDiretorio = 'diretorio';
$nomeTmp = $_FILES["ficheiro"]["tmp_name"];
$nome = $_FILES["ficheiro"]["name"];
move_uploaded_file($nomeTmp, "$uploadDiretorio/$nome");
}
?>
```

A variável global `$_FILES` contém um array com os dados dos ficheiros enviados pelo formulário. As chaves de acesso são:

- `[name]`: nome do ficheiro.
- `[type]`: tipo de imagem.
- `[tmp_name]`: caminho e nome temporário do ficheiro carregado.
- `[error]`: indica o código de erro, se houver.
- `[size]`: tamanho em bytes.

- **rename**: renomeia um ficheiro ou diretório passando-o como parâmetro; primeiro o nome antigo e depois o novo.

```
<?php
rename("diretorio", "imagens");
// renomear diretório
?>
```

- **unlink**: apaga o ficheiro passado pelo parâmetro.

```
<?php
$fh = fopen('filmes.txt', 'a');
fwrite($fh, 'teste');
fclose($fh);
unlink('filmes.txt'); // apaga o ficheiro
?>
```

- **fgets**: obtém a linha onde se encontra o ponteiro; é passado pelo parâmetro o identificador do ficheiro e, opcionalmente, o comprimento a ler. Se não for especificado, irá ler o ficheiro linha a linha até que seja lido por completo.

```
<?php
$ficheiroID = @fopen("filmes.xml", "r");
if ($ficheiroID) {
    // ler a cada 4096 bytes
    while (($buffer = fgets($ficheiroID, 4096)) !== false) {
        echo $buffer;
    }
    if (!feof($ficheiroID)) {
        echo "Erro: falha inesperada de fgets ()\n";
    }
    fclose($ficheiroID);
}
```

```
?>
```

- **feof**: informa quando o final do ficheiro for alcançado com o ponteiro; é utilizado o `fgets` ao mesmo tempo para que não haja nenhum problema.

```
<?php
while(!feof($f)) {
    $linha = fgets($f);
    echo $linha;
}
?>
```

Seguem-se alguns exemplos completos do uso dessas funções:

Exemplo

Neste exemplo pretende-se guardar os dados enviados do formulário para um ficheiro; neste aparecerá um link (`exemplo2.php`) que lerá o ficheiro criado no PHP anterior.

- `exemplo.php`:

```
<html>
<body><p>
<?php
if(isset($_POST['enviar'])) {
    // criar o ficheiro e salvar os dados
    $f = fopen('dadosFormulario.txt','w+');
    // escrever e adicionar \r \n para a quebra de linha
    fwrite($f, $_POST['corFavorita']."\r\n");
    fwrite($f, $_POST['animalFavorito']."\r\n");
    fwrite($f, htmlentities($_POST['passatempo']));
    // fechar o ficheiro
    fclose($f);
```

```
// mostrar os valores enviados
echo 'Enviou o formulário! ';
echo '<br>';
echo 'Os dados enviados foram: ';
echo '<ul><li>A minha cor favorita é
'$_POST['corFavorita'].'</li>';
echo '<li>O meu animal favorito é
'$_POST['animalFavorito'].'</li>';
echo '<li>Os meus passatempos são
'.htmlentities($_POST['passatempo']).'</li></ul>';
} else {
// se não for nada enviado, será mostrado o formulário
echo '
<strong>Formulário</strong>
<br />
<form action="" method="post">
<table>
<tr>
<td>A minha cor favorita: </td>
<td>
<select name="corFavorita" id="corFavorita">
<option value="laranja" ' .($_COOKIE['corFavorita'] == "laranja" ?
"selected" : "").' >laranja</option>
<option value="verde" ' .($_COOKIE['corFavorita'] == "verde" ?
"selected" : "").' >verde</option>
<option value="azul" ' .($_COOKIE['corFavorita'] == "azul" ?
"selected" : "").' >azul</option>
<option value="amarelo" ' .($_COOKIE['corFavorita'] == "amarelo" ?
"selected" : "").' >amarelo</option>
<option value="vermelho" ' .($_COOKIE['corFavorita'] == "vermelho" ?
"selected" : "").' >vermelho</option>
</select>
</td>
</tr>
<tr>
```



```

<td>O meu animal favorito é: </td>
<td><input type="text" name="animalFavorito" id="animalFavorito"
value="'.htmlentities($_COOKIE['passatempo']).'" size="20"
maxlength="50" />
</td>
</tr>
<tr>
<td>Os meus passatempos são:</td>
<td><textarea name="passatempo" id="passatempo" cols="35"
rows="3">'.htmlentities($_COOKIE['passatempo']).'</textarea></td>
</tr>
<tr>
<td colspan="2"><input type="submit" id="enviar" name="enviar"
value="Enviar Dados"></td></tr>
</table>
</form>';
}
?>
</p></body>
</html>

```

■ exemplo2.php:

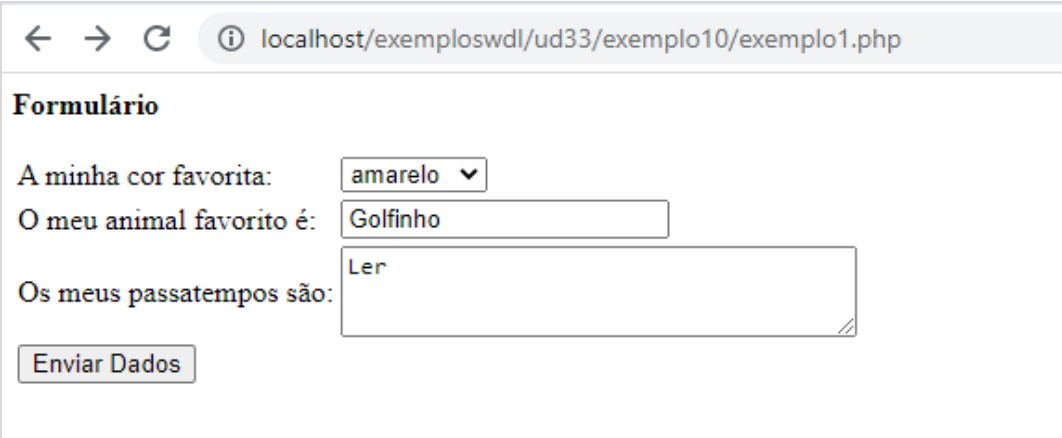
```

<html>
<body><p>
<?php
if(file_exists('dadosFormulario.txt'))
{
echo 'Leitura do ficheiro utilizando file_get_contents()<br />';
echo '<pre>';
echo file_get_contents('dadosFormulario.txt');
echo '</pre>';
echo '<hr />';
echo 'Leitura do fichero utilizando fgets()<br />';

```

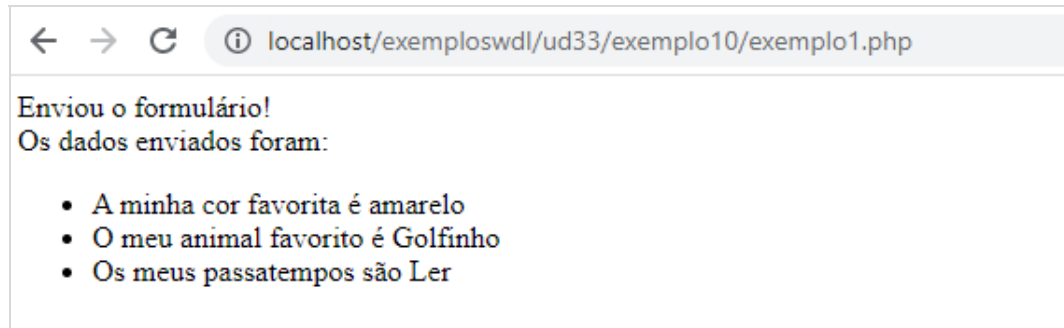
```
$f = fopen('dadosFormulario.txt','r');
if(!$f) { die('Não foi possível abrir o ficheiro'); }
echo '<pre>';
// feof detecta o fim do ficheiro
while(!feof($f)) {
$linha = fgets($f);
echo $linha;
}
echo '</pre>';
fclose($f);
} else {
echo 'O ficheiro não foi criado. Vá para <a href=
"exemplo.php">exemplo.php</a> para criá-lo';
}
?>
</p></body>
</html>
```

A visualização da página será (já com os dados preenchidos):



The screenshot shows a web browser window with the address bar displaying 'localhost/exemploswdl/ud33/exemplo10/exemplo1.php'. The page content is a form titled 'Formulário'. It contains three input fields: a dropdown menu for 'A minha cor favorita:' with 'amarelo' selected, a text box for 'O meu animal favorito é:' containing 'Golfinho', and a text area for 'Os meus passatempos são:' containing 'Ler'. Below these fields is a button labeled 'Enviar Dados'.

A visualização da página depois de clicar no botão “Enviar Dados” será:



Exemplo

Neste exemplo irá criar-se uma função que percorre o diretório indicado, mostrando os ficheiros que contém. Este caso é muito útil ao fazer galerias de fotografias.

```
<html>
<body><p>
<?php
function getDir($dir='.', $recursivo = false) {
    $files = array();
    if (is_dir($dir)) {
        $fh = opendir($dir);
        while (($file = readdir($fh)) !== false) {
            if(!in_array($file, array('.', '..'))) {
                $filepath = $dir . '/' . $file;
                if ( is_dir($filepath) && $recursivo ) {
                    array_push($files, $filepath);
                    $files = array_merge(
                        $files,
                        getDir($filepath, $recursivo)
                    );
                }
            }
        }
    }
}
```

```
    } else  
        array_push($files, $filepath);  
    }  
}  
closedir($fh);  
} else {  
    return false;  
}  
return $files;  
}  
$files = getDir('.', true);  
print_r($files);  
?>  
</p></body>  
</html>
```

CONCLUSÃO

Os recursos avançados que o PHP permite são muito úteis para o desenvolvimento das páginas web. Deverá sempre tentar tirar o máximo proveito das ferramentas do PHP e, assim, criar um website muito profissional com o mínimo de código.

Fazer upload de ficheiros para o servidor, navegar por pastas, enviar e-mails, etc., ou seja, todas as funcionalidades avançadas vistas estão em uso contínuo em praticamente todas as páginas web que transportam programação.

AUTOAVALIAÇÃO

1. **O que são cookies?**
 - a) Objetos HTML em formato de cookie.
 - b) Um sistema de armazenamento de dados de navegação no servidor.
 - c) Um sistema de armazenamento de dados de navegação no cliente.
 - d) Um array de armazenamento de dados do utilizador.

2. **Qual é o nome da variável global que contém o array com os cookies?**
 - a) \$_COOKIES.
 - b) \$_COOKIE.
 - c) \$COOKIES.
 - d) \$COOKIE.

3. **Na função setcookie, qual é o parâmetro que indica o prazo do cookie?**
 - a) caduc.
 - b) _expire.
 - c) \$expire.
 - d) expire.

4. Se se pretender que um cookie seja acessível apenas a partir de um subdomínio de dados de um domínio, o que deve utilizar-se?
- a) domain:dados.
 - b) domain:_dados.meudominio.com.
 - c) domain:dados.meudominio.com.
 - d) domain:exc www.meudominio.com.
5. Qual é a classe utilizada para enviar e-mails do PHP?
- a) phpmailer.
 - b) phpmail.
 - c) jmail.
 - d) mailPhp.
6. O que é XML?
- a) Uma linguagem de programação de script.
 - b) Um sistema de transferência de informações.
 - c) Um mecanismo de base de dados.
 - d) Uma biblioteca PHP.
7. O que se utiliza para carregar o conteúdo de um ficheiro XML diretamente num array?
- a) load_xml_file.
 - b) simppler_xmlload.
 - c) simple_xml_load.
 - d) simplexml_load_file.

- 8. O que é SimpleXMLElement?**
- a) Um método da classe SimpleXML.
 - b) Uma propriedade da classe SimpleXML.
 - c) O nome da classe SimpleXML.
 - d) Uma função da classe SimpleXML.
- 9. O que é que a função xpath da biblioteca SimpleXML permite?**
- a) Encontrar ficheiros XML de referência fora do servidor.
 - b) Encontrar todos os elementos do mesmo rótulo.
 - c) Encontrar o pai do rótulo passado pelo parâmetro.
 - d) Encontrar o diretório do ficheiro.
- 10. O que se utiliza para adicionar uma nova ramificação à árvore hierárquica de um XML?**
- a) addChild.
 - b) appendChild.
 - c) chilAppend.
 - d) chilAdd.

SOLUÇÕES

1.	c	2.	b	3.	d	4.	c	5.	a
6.	b	7.	d	8.	c	9.	b	10.	a

PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO

Para saber mais sobre XML, visite o seguinte website (em inglês):

- <http://www.w3.org/XML/>

Para conhecer todas as funções de gravação de ficheiros, visite a página web oficial do PHP:

- http://www.php.net/manual/pt_BR/ref.filesystem.php

BIBLIOGRAFIA

- Cabezas, L. M. (2010), *Php 5*. Madrid: Anaya Multimedia.
- The PHP Group (2001), “SimpleXML”. Disponível em:
https://www.php.net/manual/pt_BR/book.simplexml.php
Consultado a 25 de março de 2021.

