Sessões no PHP

Sessão nada mais é como o próprio nome já diz sessão, ou seja, onde pode-se aguardar e ler informações independente da página onde esteja. Uma forma de mandar informações de uma página para outra, mas não necessariamente precisa ser só isso, por exemplo, página de login, a pessoa digitou email e senha, foi para página redirecionada, validou o login, deu tudo certo, e ai guarda a identificação desse usuário que está logado. Essa informação fica salva na sessão, a partir daí todas as páginas no sistema que esse usuário tiver permissão, se sabe que está logado através da sessão com essas informações. É parecido com cookies, que é uma informação que fica salva no computador da pessoa que acessa o site. A diferença que uma sessão não fica salva no computador do usuário, fica no servidor. Então quando o usuário acessa ele identifica que a sessão pertence a esse usuário e aí pega as informações.

Como usamos na prática uma sessão ?

Primeiro, todas as página que forem usar uma sessão tem que terem logo no início um comando chamado **session_start().** Isso significa se não existir na página uma sessão ele vai criar e se existir vai recuperar essa sessão. Por exemplo, a página index.php vai ter então colocamos no início dela e do recebe.php também então coloca-se esse comando no início dela.

Arquivo index.php

```
<?php
   session_start();
?>
<form method="POST" action="recebe.php">
<label>
        Nome:
        </br>
        <input type="text" name="nome"/>
</label>
</br>
</br>
<label>
        E-mail:
```

```
        </br>
        <input type="text" name="email"/>
</label>
</br>
</br>
<label>
        Idade:
        </br>
        <input type="text" name="idade"/>
</label>
</br>
</br>
<input type="submit" value="Enviar"/>
</form>
```

Arquivo recebe.php

```
<?php
session_start();
$nome = filter_input(INPUT_POST,'nome');
$email = filter_input(INPUT_POST,'email',FILTER_VALIDATE_EMAIL);
$idade = filter_input(INPUT_POST,'idade',FILTER_SANITIZE_NUMBER_INT);
if($nome && $email && $idade){
   echo "Nome: ".$nome."</br>";
   echo "E-mail: ".$email."<br>";
   echo "Idade: ".$idade;
}else{
```

```
    header("Location: index.php");
   exit;
}
?>
```

Então, agora vamos fazer no arquivo recebe.php, com que salve informações na sessão e quando voltar para a página index.php ler essas informações.

Como fizemos para salvar essas informações na sessão?

Para isso vamos usar uma variável do php que é um array chamada de $_SESSION[], sempre escrita em maiúscula, senão não funciona. E dentro dela como string colocamos um nome específico para o aviso que queremos salvar. Por exemplo, queremos salvar um aviso, especificamos assim $_SESSION['aviso'] e recebendo uma string ou um número, pode ser um array, por exemplo:

```
$_SESSION['aviso'] = 'Preencha os itens corretamente';
```

Com isso salvamos uma informação na sessão e quando for para o index.php já vai ter essa informação.

Veja essa aplicação no arquivo **recebe.php**:

```
<?php
session_start();
$nome = filter_input(INPUT_POST,'nome');
$email = filter_input(INPUT_POST,'email',FILTER_VALIDATE_EMAIL);
$idade = filter_input(INPUT_POST,'idade',FILTER_SANITIZE_NUMBER_INT);
if($nome && $email && $idade){
    echo "Nome: ".$nome."</br>";
    echo "E-mail: ".$email."<br>";
    echo "Idade: ".$idade;
}else{
     //Aqui se der errado vem pra cá, salva e redireciona para o index.php
    $_SESSION['aviso'] = 'Preencha os itens corretamente';
    header("Location: index.php");
```

```
  exit;
}
?>
```

Agora, depois de salvar na variável $_SESSION['aviso'] essa informação vamos no index.php e colocamos uma verificação se essa sessão existe, então mostre essa informação.

Como fizemos ?

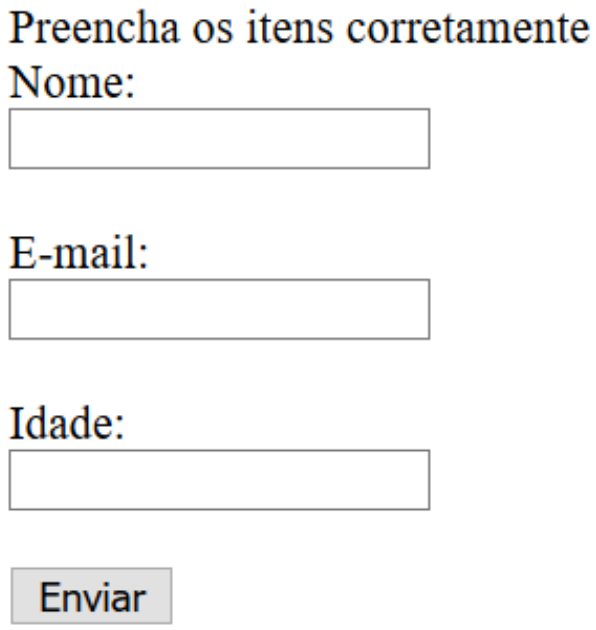
Veja essa aplicação de verificação no arquivo **index.php**, logo abaixo do código session_start:

Arquivo index.php

```
<?php
   session_start();
   if($_SESSION['aviso']){  //Faz a verificação se existe essa sessão $_SESSION['aviso']
                            //se existe mostra a informação...
        echo $_SESSION['aviso'];
   }

?>
<form method="POST" action="recebe.php">
<label>
        Nome:
        </br>
        <input type="text" name="nome"/>
</label>
</br>
</br>
<label>
        E-mail:
        </br>
        <input type="text" name="email"/>
```

```
</label>
</br>
</br>
<label>
        Idade:
        </br>
        <input type="text" name="idade"/>
</label>
</br>
</br>
<input type="submit" value="Enviar"/>
</form>
```

Na página index.php, por exemplo, não foi preenchido os campos e ao clicar no botão enviar, mostra essa mensagem da variável $_SESSION[], ficando assim:



Mas percebem que sempre que atualizar a página já mostra essa mensagem "Preencha os itens corretamente", para isso não ocorrer, vamos alterar no código para que uma vez mostrado essa

mensagem, tira esse aviso na sessão. Ou seja, mostra somente uma vez, perceba isso atualizando novamente a página, a mensagem não aparece.

Observem como isso é feito no arquivo index.php:

Arquivo index.php

```
<?php
   session_start();
   if($_SESSION['aviso']){
        echo $_SESSION['aviso'];
        $_SESSION['aviso'] = ''; //aqui após mostrar zera, limpa
   }

?>
<form method="POST" action="recebe.php">
<label>
        Nome:
        </br>
        <input type="text" name="nome"/>
</label>
</br>
</br>
<label>
        E-mail:
        </br>
        <input type="text" name="email"/>
</label>
</br>
</br>
<label>
```

```
        Idade:
        </br>
        <input type="text" name="idade"/>
</label>
</br>
</br>
<input type="submit" value="Enviar"/>
</form>
```

Cookies no PHP

Muito similar a uma sessão, a diferença que um cookie fica salvo no navegador do usuário. Vamos trabalhar com três coisas:

Primeiro, como setar, definir um cookie;

Segundo, como acessa esse cookie;

Terceiro, como deletar esse cookie.

Por exemplo, o nome que o usuário digitar no campo nome do formulário no arquivo index.php vai salvar no cookie, pra poder ser lido em qualquer lugar. Vamos fazer com que leia no cabeçalho da página esse nome digitado. Ou seja, digita o nome e vai para página recebe.php que seta no cookie esse nome, e vai ser lido no cabeçalho ou seja header.php.

Primeiro vamos ao arquivo recebe.php e entrando no if, significa que foi enviado um nome. E vamos setar um coookie dentro do if. Mas ATENÇÃO: Só tem como setar um cookie antes de exibir qualquer tipo de conteúdo.

Para setar um cookie, usamos uma função chamada setcookie(), ela tem vários parâmetros. O primeiro parâmetro é o nome do cookie. O segundo é o valor que vai ficar salvo nesse cookie. O terceiro quando é que esse cookie expira, ou seja, o prazo de validade, para isso usamos a função time().

Lembrando que a função time() retorna milissegundos do exato momento, então temos que colocar um tempo maior que o atual.

Arquivo recebe.php

```
<?php
```

```
$nome = filter_input(INPUT_POST,'nome');

$email = filter_input(INPUT_POST,'email',FILTER_VALIDATE_EMAIL);

$idade = filter_input(INPUT_POST,'idade',FILTER_SANITIZE_NUMBER_INT);

if($nome && $email && $idade){

//Aqui dentro do if vamos setar o cookie. Obs.: que até aqui não foi exibido nada ainda, //só a partir da próxima linha de código. Então só posso definir aqui um cookie, e não //depois de exibir algo na tela...

//Obs.: que esse coookie vai ter validade da data de agora mais 30 dias para frente, depois //para de ser lido.

//86400 = um dia

        $expiracao = time() + (86400 * 30);

        setcookie('nome', $nome, $expiracao);


        echo "Nome: ".$nome."</br>";

         echo "E-mail: ".$email."<br>";

        echo "Idade: ".$idade;

}else{

        header("Location: index.php");

         exit;

}
```

Depois disso, vamos para o arquivo header.php para ler esse cookie. Como lê esse cookie? Com uma variável específica chamada $_COOKIE['nome'], dentro vai o nome desse cookie.

Observe como fica no arquivo esse código:

Arquivo **header.php**

```
<h2> Cabecalho</h2>

<?php

//Obs.: a função isset() para verificar se está setado...
```

```
        if(isset($_COOKIE['nome'])){
                $nome = $_COOKIE['nome'];
                echo '<h2>'.$nome.'</h2>';
        }
?>
<hr>
```

Agora é só ir no index.php e preencher os campos e clicar no botão Enviar e aparecerá os dados do campos enviados para o recebe.php. Mas para verem o resultado da informação no index.php, após essa execução, tem que dar um voltar na página que estará no cabeçalho o nome que foi enviado para o cookie.

Na figura a seguir, mostra a informação enviado para o cookie, a palavra " Sandra " preenchida no campo nome.

Quando clicado o botão enviar, e após isso clicar no botão  do navegador, como podem ver estará no cabeçalho essa informação armazenada no cookie. Então se clicarmos no atualizar sempre estará mostrando essa informação no cabeçalho durante 30 dias conforme no time(). Mas se preencher com outro nome no campo do formulário ai sim vai ser substituído por esse novo nome.

Cabecalho

Sandra

Nome:

E-mail:

Idade:

Enviar

Agora que já aprenderam a setar e pegar as informações no cookie, vamos agora fazer outra página para limpar esse cookie, ou seja, como deletar.

Antes de criar esse arquivo vamos na página index.php e criamos um link para quando clicarmos nele, apagar o cookie:

 Apagar Cookie

Que fique tipo assim:

Vamos agora fazer a página **apagar.php**, que vai deletar o cookie e mandar, redirecionar de volta para a página index.php.

Como apagamos cookie ?

Temos que setar ele com um tempo de expiração no passado. Seja aplicação no arquivo a seguir.

Arquivo apagar.php

```
?php

    setcookie('nome', '', time() - 3600); // 3600 = um minuto a menos

    //no segundo parâmetro significa que pode estar vazio, não interessa...

    header("Location: index.php");

    exit;
```

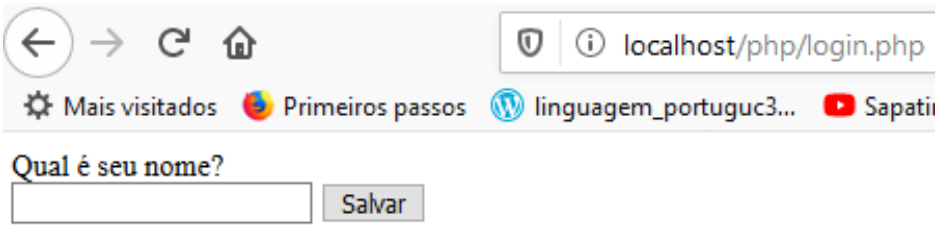
?>

Agora é só ir no arquivo index.php e clicar no link e ver o resultado.

Com esses exemplos puderam ver a diferença de SESSION para COOKIE. A sessão funciona enquanto o navegador estiver aberto, o momento que fechou é destruída. O cookie tem um tempo de validade, e permanece no computador mesmo desligando permanece durante o tempo especificado para validade.

Exercício Prático 1

Fazer um arquivo index.php, quando entrar nele, vai verificar se existe uma sessão chamada name preenchida. Caso não exista volta para página login.php. Na página login.php vamos ter o campo nome e o botão salvar, como mostra a figura a seguir:



Nessa página login.php, quando o usuário digitar um nome, clicar em Salvar, vai salvar esse nome numa sessão, e vai para página index.php, que deve mostrar a tela do index.php, por exemplo, a figura a seguir:

Nessa página index.php, para sair dela só quando o usuário clicar no botão Sair, que vai destruir, apaga essa sessão e redirecionar para a página de login. Observem quando redirecionar para página login, mesmo se colocarem na URL o endereço da página index.php não pode entrar nela sem passar pelo login novamente. Observem que ao todo são 3 páginas que devem ser feitas: login, index e sair.

Lendo Arquivos

Agora será visto como ler arquivos externos utilizando o php. Existe uma função muito simples para isso e serve tanto para arquivos externos como para internos que é a função **file_get_contents()**. Dentro dela como parâmetro se coloca o nome do arquivo ou servidor em que esse arquivo a ser lido, a URL completa.

Arquivo index.php

```
<?php

        $texto = file_get_contents('texto.txt');

        echo $texto

?>
```

Ainda podemos manipular esse texto como quisermos, por exemplo:

```
$texto = file_get_contents('texto.txt');

$texto = explode("\n",$texto);

echo "Linhas: ".count($texto);
```

Escrevendo em Aquivos

Agora como escrever em arquivos externos usando php. No php para escrever em um arquivo uma das funções mais utilizada é chamada de file_put_constents(). Essa função faz duas coisas, se o arquivo não existir ele cria esse arquivo, caso já tenha substitui esse arquivo. A função tem dois parâmetros: o primeiro é o nome do arquivo e o segundo é o conteúdo.

Arquivo index.php

```
<?php

//o texto que vou escrever em um arquivo

    $texto = 'Sandra Hedler de Amorim';

     file_put_contents('nome.txt',$texto);

    echo 'Arquivo criado com sucesso';

?>
```

No exemplo a seguir vamos ler o texto no arquivo texto.txt, depois adicionar uma nova linha a esse texto e salvar esse conteúdo nesse arquivo. Ou seja, vamos adicionar algo a mais no conteúdo desse arquivo.

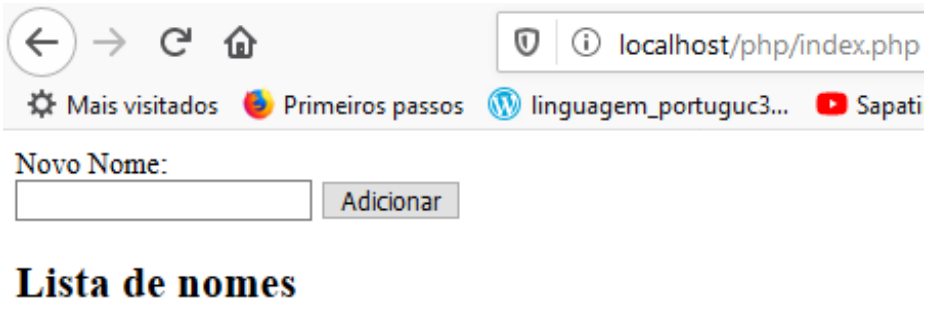
Arquivo index.php

```
<?php
    $texto = file_get_contents('texto.txt');
    $texto .= "\n Sandra Amorim"; //adicionando
    file_put_contents('texto.txt',$texto); //salvando...
    echo 'Conteudo adicionado com sucesso!';
?>
```

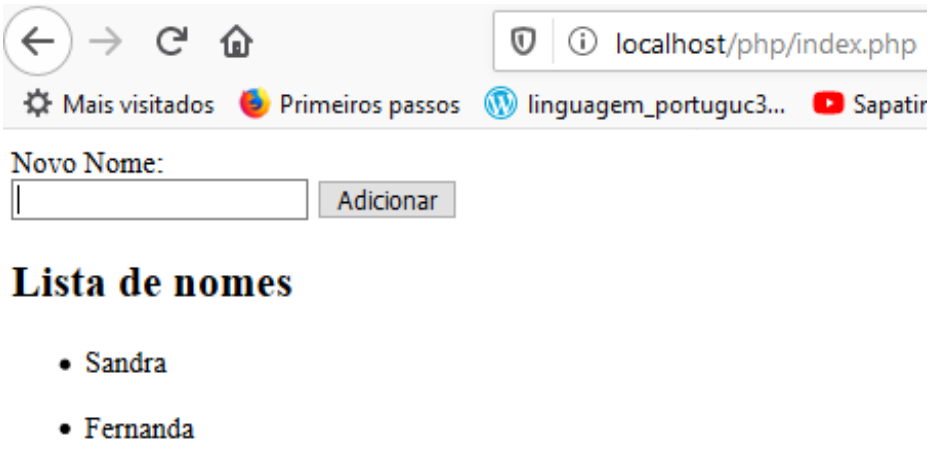
Observem no arquivo texto.txt que foi adicionado ao texto desse arquivo.

Exercício Prático 2 (Cadastro com txt)

Com o conhecimento de ler e escrever em arquivos, fazer uma lista de nome e salvar essa lista em um txt, onde cada nome em uma linha. E no index.php ler essa lista e exibir. Então, adicionar em algum lugar esses nome, validar (quero dizer, não enviado se o campo estiver em branco...) e mostrado no arquivo index. Na página index.php quando abrir ela fica assim conforme mostra na figura abaixo:



Conforme o usuário for adicionando nomes, deve ficar conforme figura abaixo:



Excluindo Arquivos

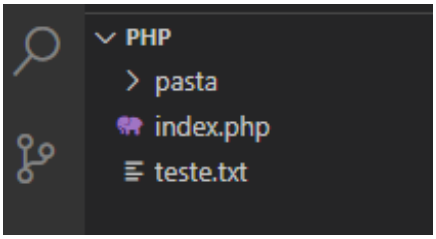
Para excluir aquivos, no php tem a função unlink() e dentro dela vai como parâmetro o nome do arquivo. Essa função exclui qualquer formato de arquivo. Mas exclui somente arquivos e não pastas.

Arquivo index.php

```
<?php
    unlink('lista.txt');
    echo "Arquivo excluido com sucesso!";
?>
```

Movendo Arquivos

Vamos ver como mover arquivos de um lugar para outro e também como renomear arquivos. Para isso vamos criar dentro da pasta PHP, outra pasta chamada **pasta**, e dentro da pasta **PHP** um arquivo chamado de **teste.txt**. Vai ficar assim conforme podem ver na figura a seguir:

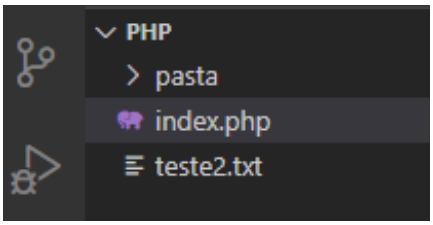


Dentro do arquivo teste, colocamos qualquer coisa para ter algum conteúdo, uma frase ou números a critério de cada um.

No arquivo **index.php** vamos fazer o seguinte código, ou seja, vamos começar com comando para renomear. O comando rename(), recebe dois parâmetros: o primeiro é o caminho até o arquivo que queremos renomear. E o segundo parâmetro é o novo nome.

```
<?php

    //Como o arquivo teste.txt está na mesma pasta do index.php, coloco direto o nome do arquivo

    rename('teste.txt','teste2.txt');

?>
```

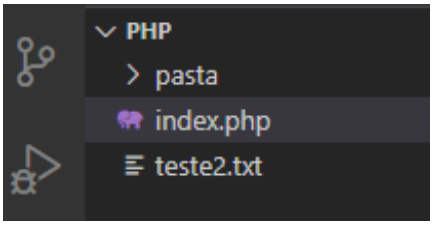
Executando o arquivo index.php, observem que trocou o nome do arquivo. Ou seja, ficou como mostra a figura a seguir:



A mesma função, usamos para mover um arquivo se quisermos. Vamos mover esse arquivo teste2.txt para a pasta chamada pasta:

```
<?php

    //mover o arquivo teste2 para pasta

    rename('teste2.txt','pasta/teste2.txt');

?>
```

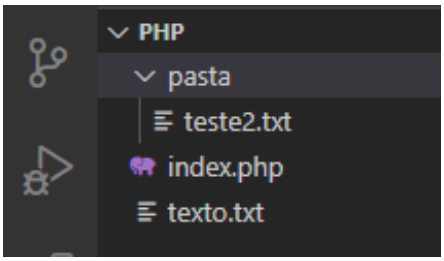
Ao executar o index.php o arquivo teste2.txt é movido para a pasta como mostra a figura a seguir:



Se quisermos fazer uma cópia do arquivo teste2.txt, usamos o comando copy(), veja exemplo a seguir:

```
<?php

        copy('pasta/teste2.txt','texto.txt');

?>
```

Observem ao executar o arquivo index.php o comando faz uma cópia do arquivo teste2.txt. Se entrarnos no arquivo texto.txt, podemos ver que o conteúdo do teste2 está nesse arquivo.



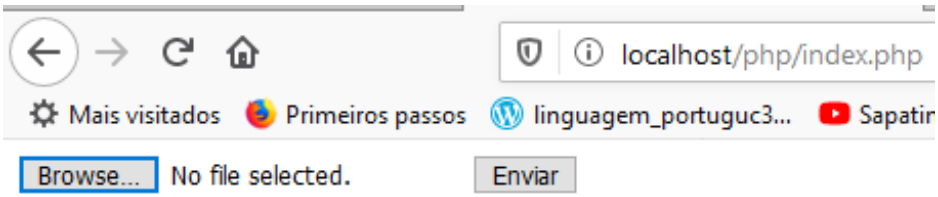
Upload de Arquivos

Em qualquer momento podemos estar fazendo upload de arquivos, seja qualquer tipo de arquivos, de imagens, etc, que queira fazer. Observem a seguir no código do formulário, quando faz upload de arquivos usamos o método de envio o POST. Pois não tem como enviar via URL, tem que ser pelo corpo da requisição. Nele temos o atributo action, que está o arquivo recebe.php, que vai receber essa requisição. Temos também o atributo enctype com os valores multipart/form-data, que possibilita o formulário agregar o conteúdo para fazer o envio desses arquivos. Nesse mesmo formulário, se quisermos adicionar campos para envio de dados, tipo campo nome, endereço não tem problema que envia normal, esse atributo enctype não vai interferir nesses campos.

Arquivo index.php

```
<form method="POST" action="recebe.php" enctype="multipart/form-data">

   <input type="file" name="arquivo"/>

   <input type="submit" value="Enviar">

</form>
```

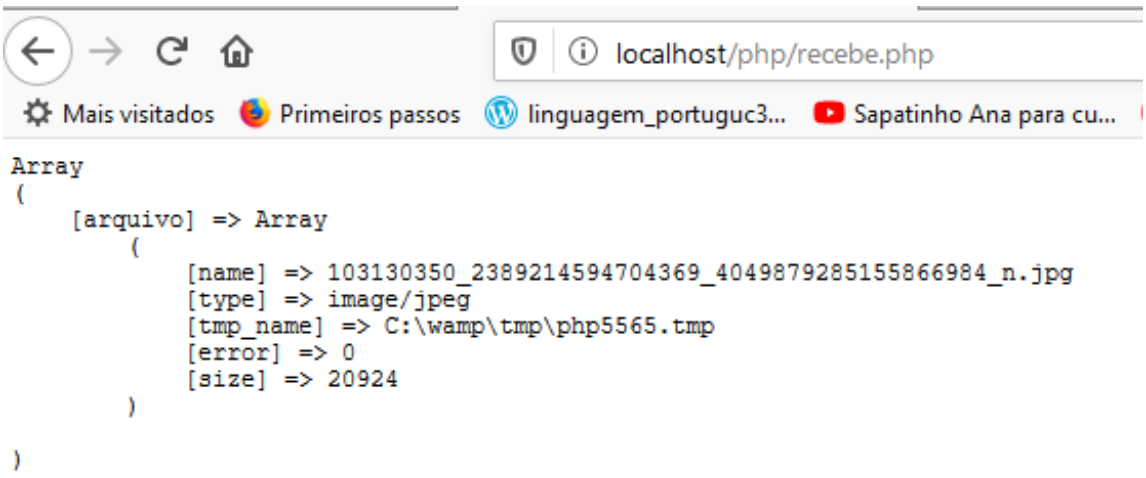
Esse arquivo index.php, vai ter esse formato conforme a figura a seguir:



Quando adicionar, anexar o arquivo, e clicar no botão Enviar o formulário vai enviar para a página recebe.php, que vai ter o código para fazer o que tem que ser feito com o arquivo. Veja a seguir o código para o arquivo **recebe.php**:

```
<?php

//Mostra na tela

echo '<pre>';

print_r($_FILES);

?>
```

Observem o resultado do envio do arquivo anexo na página index.php para o recebe.php:



Observem que tem um array, que é gerado com a variável global $_FILES, tem um campo chamado arquivo - [arquivo] => Array, que é o nome do campo file no formulário. Nesse array tem informações essenciais para entenderem como funciona: A primeira delas e o [tmp_name] => C:\wamp\tmp\php5565.tmp. Nesse tmp_name, por exemplo, quando fizemos um upload de um arquivo para um servidor o funcionamento é o seguinte, ele recebe esse arquivo e armazena temporariamente esse arquivo em algum local lá do servidor. Nessa linha, mostra onde ele temporariamente armazenou. Observem nessa linha, quem tem o XAMPP instalado no seu computador, no lugar de wamp aparecerá xampp. Outra informação importante o [error], se tiver 0 é que não teve erro nenhum. Se tiver outro valor, quer dizer que foi enviado, mas enviado com erro. [size] é o tamanho do arquivo em BYTES. [name] é o nome originário do arquivo que o

usuário enviou. Mais adiante vamos ver que não necessariamente precisa ser o nome original que o usuário enviar. Por exemplo, se enviar e já estiver um arquivo com mesmo nome não queremos que substitua. O [type] é uma propriedade chamada manetype, todo arquivo de qualquer sistema operacional tem, que diz o tipo raiz do arquivo, ou seja, a gente vê o tipo do arquivo pelo type.

Então onde está esse arquivo que fizemos upload ? Está na pasta temporária. Se não usar esse arquivo que fizemos upload, quando a requisição acabar o arquivo vai ser deletado automaticamente, dependendo da configuração. Por exemplo, esse arquivo que falamos se refere a [tmp_name] => C:\wamp\tmp\php5565.tmp

Observem, que podemos em tempo de requisição ainda aceitar esse arquivo. Para isso movemos esse arquivo da área de temporário para dentro do sistema onde queremos que faça esse processo. Existe no php uma função que faz esse processo que se chama **move_uploaded_file()**. Ela recebe dois parâmetros: O primeiro parâmetro onde está o arquivo na pasta temporária. E o segundo parâmetro , onde quer que o arquivo fique.

Veja aplicação no arquivo recebe.php:

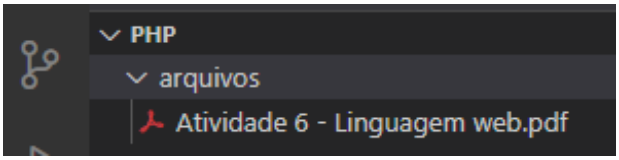
```
<?php

//Mostra na tela

        echo '<pre>';

        print_r($_FILES);

        $nome = $_FILES['arquivo']['name'];

//no segundo parâmetro coloquei para armazenar na pasta arquivos

//e o nome fica o mesmo do enviado...

        move_uploaded_file($_FILES['arquivo']['tmp_name'], 'arquivos/'.$nome);

?>
```

Após salvar essa alterações voltamos no index.php e vamos testar anexando novamente um arquivo e verificamos na pasta arquivos se foi enviado para essa pasta o arquivo anexado.

Na figura a seguir mostra a página do recebe.php

```
Array
(
    [arquivo] => Array
        (
            [name] => Atividade 6 - Linguagem web.pdf
            [type] => application/pdf
            [tmp_name] => C:\wamp\tmp\php791F.tmp
            [error] => 0
            [size] => 301311
        )

)
```

Abaixo figura da pasta arquivos

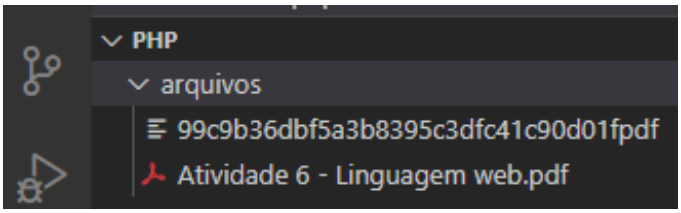


O problema de enviar assim, conforme código do index.php é se resolvermos reenviar esse mesmo arquivo, ele vai substituir na pasta, devido ter o mesmo nome o arquivo reenviado. Então não é muito apropriado fazer dessa forma.

Vamos agora fazer de forma, para resolver esse problema de substituição do arquivo com o mesmo nome. A melhor solução é gerar o próprio nome, ou seja, gerar um nome aleatório para o arquivo. Para isso vamos gerar com o comando md5(), um nome em hexadecimal, criptografado, e com o rand() um número aleatório, por exemplo de 0 a 1000. Veja a alteração no arquivo recebe.php:

```
<?php

//Mostra na tela

        echo '<pre>';

        print_r($_FILES);

//obs.: que por último foi colocado por padrão a extensão pdf

        $nome = md5(time().rand(0,100)).'pdf';

        move_uploaded_file($_FILES['arquivo']['tmp_name'], 'arquivos/'.$nome);

?>
```

Agora vamos reenviar esse mesmo arquivo e observem na pasta arquivos como ficou:



Gerou um arquivo com outro nome. Se fizermos mais de uma vez, cada uma desses novos reenvio vai gerar com nomes diferentes, não substituindo o outro arquivo. Esse é o segredo de receber arquivos dos usuários com qualquer nome é não se preocupar em substituir.

Agora vamos supor que estamos fazendo um formulário que só aceite imagens. Como fizemos? Primeiro temos que conferir se o type desse arquivo está dentro da lista de type que queremos. Para isso usaremos uma função chamada **in_array()**. No primeiro parâmetro vamos colocar o

type do arquivo e por segundo onde vamos procurar numa lista de array. Observem essa aplicação no recebe.php:

```
<?php

//Mostra na tela

echo '<pre>';

print_r($_FILES);

if(in_array($_FILES['arquivo']['type'], array('image/jpeg', 'image/jpg', 'image/png'))){

    //Sendo uma dessas extensões então vai gerar nome, salvar na pasta...

    $nome = md5(time().rand(0,100)).'pdf';

    move_uploaded_file($_FILES['arquivo']['tmp_name'], 'arquivos/'.$nome);

    echo "Arquivo salvo com sucesso!";

} else {

    echo "Arquivo não permitido!";

}

?>
```

Agora, basta ir à página index.php, enviar e observar na pasta arquivos e também testar com formatos diferentes desses 3 formatos especificados no código e ver que não envia, não aceita outra extensão de arquivo. Então é assim que fizemos um filtro nos envios de arquivos.