SESSION

Fonte:

https://blog.vilourenco.com.br/php-trabalhando-com-sessions/

O que é sessão?

A sessão ou session nos permite passar dados de uma página para outra, sem que haja perda de informação ou necessidade usar os métodos POST ou GET. Cada visitante ao acessar o site recebe um identificador único, a ser propagado via URL ou armazenado em um cookie na máquina do usuário. Podemos declarar inúmeras variáveis e associá-las a sessão do usuário, criando ambientes personalizados para cada usuário do nosso site. Para não dificultar a compreensão do conceito, vamos assumir que as variáveis de sessão ficarão armazenadas na sessão do usuário enquanto o seu navegador continuar aberto, ou enquanto a mesma não for encerrada (seja por inatividade, ou por intermédio de alguma função própria para esse fim). Uma vez encerrada, as variáveis de sessão não eliminadas.

Uma sessão é iniciada pela função session_start(), a ser colocada no topo da página, antes de qualquer outro código. Veja o exemplo abaixo:

```php
<?php
session_start();
//Resto do código.... <html>...</html>
?>
```

Salvo em alguns casos especiais, sempre usaremos a função session_start() no início da página e antes de qualquer outro código. O que essa função faz é habilitar o uso de todas as variáveis de sessão dentro da página onde foi chamada, ou seja, dizemos ao PHP que essa página poderá fazer uso de variáveis de sessão e funções para gerenciamento da mesma. Toda sessão pode expirar por inatividade, pois possui algo chamado "tempo de vida" definido em 180 minutos, por padrão. Se precisarmos alterar o tempo de vida de uma função podemos fazer uso de funções específicas como session_cache_expire(), a serem estudadas mais a frente.

Array $_SESSION

Ok! Já entendemos o que é uma sessão, como iniciá-la e para o que ela serve, agora vamos descobrir como armazenar dados na sessão. Todas as informações gravadas em sessão ficam armazenadas em uma variável superglobal definida pelo próprio PHP, chamada $_SESSION. Trata-se de um array onde, uma vez iniciada a sessão, temos acesso aos seus campos e podemos adicionar campos novos. Veja o exemplo abaixo:

login.php

```
1 <form action="validaLogin.php" method="post" id="form" name="form">
2 <label for="login">Login: <input type="text" name="login" id="login" /></label>
3 <label for="senha">Senha: <input type="password" name="senha" id="senha" /></label>
4 <button type="submit">LOGIN</button>
5 </form>
```

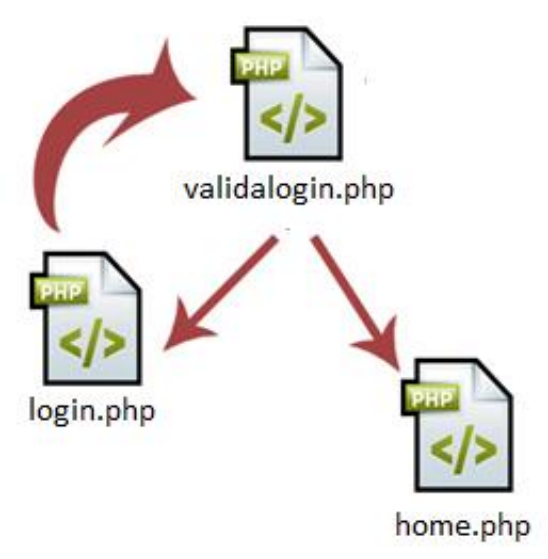
validalogin.php

```php
<?php
//INICIO A SESSÃO
session_start();

$login = array("user01", "user02", "user03", "user04", "user05");
$senha = array("senha01", "senha02", "senha03", "senha04", "senha05");

//Calculo o tamanho do array $login
$tamArray = count($login);
//Crio uma variável auxiliar
$msg = FALSE;
//Uso um loop para percorrer o array
for ($i = 0; $i < $tamArray; $i++) {
if ($_POST["login"] == $login[$i] && $_POST["senha"] == $senha[$i]) {
$msg = TRUE;
break;
}
}
//Verifico se a variável auxiliar $msg saiu do loop com o valor TRUE (indicando login efetuado com sucesso)
if ($msg) {
//Armazeno duas informações na sessão do usuário: se ele está logado, e o login de acesso. A partir desse momento, qu
$_SESSION["logado"] = TRUE;
$_SESSION["user"] = $_POST["login"];
//Uso a função header() para fazer o redirecionamento para a página principal do site, uma vez que o login foi execut
header ("Location: home.php");
}
else {
//Caso o login dê errado, devolvo o usuário para a página de login
header ("Location: login.php");
}
?>
```

A única novidade no script acima é o uso da função header() que, dentre outras coisas, permite o redirecionamento do usuário de uma página para outra.

Bem, vimos que esse script funciona em três tempos:

o usuário entra com os dados de login na página **login.php**, esses dados são recebidos e tratados em **validalogin.php** e, em caso de sucesso armazenamos dois dados em variáveis de SESSÃO, caso contrário devolvemos o usuário para a tela de login.



Já sabemos como armazenar dados na sessão do usuário, agora veremos como resgatá-los. Ao redirecionar o usuário para a página **home.php**, devemos verificar se ele de fato está logado no sistema. Sabemos disso através da variável de sessão $_SESSION[“logado”], se ela existir e seu valor for TRUE, é porque o usuário fez login e senha corretamente. Vamos ao código:

home.php

```php
<?php
//INICIO A SESSÃO
session_start();

//Verifico se o usuário está logado no sistema
if (!isset($_SESSION["logado"]) || $_SESSION["logado"] != TRUE) {
    header("Location: login.php");
}
else {
    echo "<h1>Seja bem-vindo, ".$_SESSION["user"]."</h1>";
}
?>
```

Já sabemos como armazenar dados na sessão do usuário, agora veremos como resgatá-los. Ao redirecionar o usuário para a página **home.php**, devemos verificar se ele de fato está logado no sistema. Sabemos disso através da variável de sessão $_SESSION[“logado”], se ela existir e seu valor for TRUE, é porque o usuário fez login e senha corretamente. Vamos ao código:

home.php

```php
<?php
//INICIO A SESSÃO
session_start();

//Verifico se o usuário está logado no sistema
if (!isset($_SESSION["logado"]) || $_SESSION["logado"] != TRUE) {
    header("Location: login.php");
}
else {
    echo "<h1>Seja bem-vindo, ".$_SESSION["user"]."</h1>";
}
?>
```

Include(“checarLogin.php”);

checarLogin.php

<?php ...
?>

Basta iniciar session_start() e todas as variáveis de sessão declaradas em qualquer lugar da aplicação, estarão disponíveis na página de chamada.

A novidade do script acima fica por conta da função isset(), que verifica se uma variável existe e, em caso afirmativo retorna TRUE, caso contrário retorna FALSE.

Principais **funções para gerenciamento de sessões em PHP**

session_start()

Essa função inicia a sessão do usuário, e disponibiliza para a página todas as variáveis de sessão desse usuário. Em resumo: somente iniciando a sessão, poderemos trabalhar com sessão em uma página. Ela sempre retorna TRUE. Outra coisa importante: sempre utilize a função session_start() no topo da página, acima de qualquer outro elemento, caso contrário ela pode não funcionar. Veja um exemplo de session_start() em funcionamento:

```php
<?php
//INICIO A SESSÃO
session_start();

$_SESSION["nome"] = "Diego";
echo $_SESSION["nome"];
?>
```

session_cache_expire()

Essa função modifica o tempo de expiração da sessão do usuário. Normalmente quando uma sessão fica inativa por mais de 180 minutos ela expira, apagando todas as variáveis gravadas nela. A função session_cache_expire() aumenta ou diminui o tempo de expiração de uma sessão. Recebe como parâmetro o tempo em minutos para expiração da sessão. Precisamos declarar essa função antes de session_start(). Veja um exemplo dessa função:

```php
<?php
//INICIO A SESSÃO COM UM TEMPO DE EXPIRAÇÃO DE 10 MINUTOS
session_cache_expire(10);
session_start();

$_SESSION["nome"] = "Diego";
echo $_SESSION["nome"];
?>
```

session_id()

Essa função retorna o ID da sessão, ou seja, um código identificador único para cada sessão de usuário.

Essa função retorna o ID da sessão de usuário.

Veja session_id() em funcionamento:

```php
<?php
//INICIO A SESSÃO
session_start();

$id = session_id();
echo $id;
?>
```

session_destroy()

Essa função destrói a sessão e todos os dados associados a ela. Retorna TRUE em caso de sucesso, FALSE em caso de erro.

Veja um exemplo da função em funcionamento:

```php
<?php
//INICIO A SESSÃO
session_start();

if (session_destroy()) {
    echo "Sessão destruída";
}
else {
    echo "Não foi possível destruir a sessão";
}
?>
```

Utilize na página **sair.php**