1. Cookies

São variáveis gravadas no cliente (*browser*) por um determinado *site*. Somente o site que gravou o cookie pode ler a informação contida nele. Este recurso é muito útil para que determinadas informações sejam fornecidas pelo usuário apenas uma vez. Exemplos de utilização de cookies são sites que informam a quantidade de vezes que você já visitou, ou alguma informação fornecida numa visita anterior. Existem cookies persistentes e cookies de sessão. Os persistentes são aqueles gravados em arquivo e que permanecem após o browser ser fechado, possuindo data e hora de expiração. Os cookies de sessão não são armazenados em disco e permanecem ativos apenas enquanto a sessão do browser não for encerrada.

Atenção I: o uso de cookies não é recomendado quando se trata de informações sigilosas. Os dados dos cookies são armazenados no diretório de arquivos temporários do visitante, sendo facilmente visualizado por pessoas "mal intencionadas". Há também a opção "aceitar cookies" que pode ser desativada a qualquer momento pelo visitante. Para uma transmissão de dados segura é recomendável o uso de sessões.

Atenção II: o trabalho com cookies pode encontrar alguns problemas como: o usuário apagar os cookies do navegador, o computador ser formatado, o usuário bloquear a utilização de cookies pelo navegador.

2. Para gravar cookies

Para gravar cookies no cliente, deve ser utilizada a função setcookie.

Sua sintaxe é a seguinte:

setcookie([NOME DO COOKIE], [VALOR DO COOKIE], [TEMPO DE VIDA]);

Sendo que:

> Nome do Cookie: É o nome que vamos usar para criar e acessar as informações do cookie.
>
> Valor do Cookie: É o valor que queremos gravar no cookie. Pode ser um valor fixo entre aspas ou até mesmo uma variável do PHP.
>
> Tempo de Vida: É o tempo em que o cookies vai ficar armazenado no computador do usuário. Este tempo e sempre indicado em segundos.

Aqui poderia ser um campo lido de um formulário ou campo de uma tabela no BD

Exemplo (pag1.php):

```
<?php
        $value = 'Estou no IFSP';
        setcookie("CookieTeste", $value, time()+3600);  /* expira em 1 hora */
?>
```

Obs.: Podemos criar vários cookies e armazenar valores, como de banco de dados. Podemos executar uma instrução mysqli com select, pegar os valores das colunas do comando SQL e guardar em cookies para que possam ser usados em outras páginas.

2. Lendo cookies gravados

Para ler o cookie podemos usar a instrução $_COOKIE.

Exemplo (pag2.php):

```
<?php
        // Mostra um cookie individual
        echo $_COOKIE["CookieTeste"];
?>
```

Exemplo prático que podem usar para implementar: contador de acesso à página *Web*

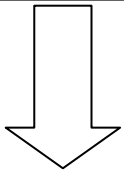
Extraído do link: https://pt.stackoverflow.com/questions/209856/salvar-cookie-da-visita-na-p%C3%A1gina

Olá estou usando um código php mysql que conta as visitas de cada página, porém sempre que a página é atualizada ele adiciona mais uma visita, gostaria de uma ajuda com meu código para que ele salve o cookie por um determinado tempo antes do contar uma nova visita.

PS: cada página está salva no BD e tem visualizações distintas

Código das paginas:

```
<?php
//Aqui pegamos o id da página
$idDaPagina = $explode[1];

//Busca na tabela o numero de vezes que a página ja foi visitada
$busca = "SELECT * FROM contador WHERE idPagina = '$idDaPagina'";
$exe = mysqli_query($conectar,$busca);

$resultado_vist = (mysqli_fetch_array($exe));

//Pega o numero de visistas que consta na tabela, adiciona mais um e atualiza
$visitantes = $resultado_vist['visitas'] + 1;
$altera = "UPDATE contador SET visitas = '$visitantes' WHERE idPagina = '$idDaPagina'";
$exe1 = mysqli_query($conectar,$altera);

//Faz uma nova busca e retorna o numero de visitas depois da atualização
$exe = mysqli_query($conectar,$busca);
$total = (mysqli_fetch_array($exe));
$visitas = $total['visitas'];

?>

No espaço do conteudo dou um <?php echo $visitas; ?>
```



Você pode trabalhar com cookies, defina o tempo necessário dele e quando expirar, você adiciona +1 ao contador. Seria o seguinte código:

```
// Define em horas quanto tempo vai durar cada cookie
$hours = 1;

// Comando para setar o cookie
setcookie("contador", true, time() + (3600 * $hours));

// Verificação do cookie
if (isset($_COOKIE["contador"]) && $_COOKIE['contador'] == true ) {
    // Faz nada...
} else {
    // Logica para adicionar mais 1 a visitas de páginas
}
```

Curiosidade:

```
<?php
        setcookie("cookie", "teste", time() + 3600*24*365);
?>
```

Esse cookie tem validade de um ano, considerando a conta:

3600 segundos = 1 hora
3600 segundos vezes 24 horas
24 horas vezes 365 dias

Para excluir cookie podemos usar o mesmo comando para criação, porém com apenas uma diferença. Veja exemplo.

```
<?php
	setcookie("cookie");
?>
```

Se usarmos o comando setcookie apenas com o nome do cookies que tínhamos criado, ou seja, sem definir nenhum valor, o PHP interpreta que o cookie será apagado.

OBS.: Podemos usar cookies para login do usuário em uma página PHP, porém quando ele dá logout na página, podemos usar a instrução de excluir o cookie para segurança. Assim o usuário não conseguirá mais navegar nas opções dentro do login, tendo que entrar com login e senha para ativar os cookies novamente. Veja o exemplo abaixo para logout:

```
<?php
	setcookie(“usuario”);
	setcookie(“senha”);
	header(“Location: login.html”);
?>
```

Conforme explicado anteriormente, o exemplo acima, exclui os cookies nome e senha e redireciona o navegador para a página de login. A partir desse momento, o usuário só poderá acessar o site se fornecer sua identificação novamente. Supondo que este é um exemplo que foi gravado no arquivo logout.php. Assim, podemos colocá-lo em formato de um link em todas as páginas do site, na qual o usuário poderá efetuar o logout a qualquer momento. Exemplo: Logout

Tecnicamente falando, um cookie é uma pequena quantidade de informação persistida temporariamente pelo navegador. Os navegadores normalmente limitam o tamanho dos cookies em até 4KB, e apagam cookies com a data de “validade vencida”.

2. Sessões

Uma sessão (session) é um período de tempo durante o qual uma pessoa navega num site. Uma sessão é gravada no servidor, e o tempo de duração de uma sessão pode ser até o usuário fechar a página.

Quando o usuário entra no site criamos uma sessão e ela recebe um número identificador único, chamado de session_id, para uma página ter o acesso aos dados da sessão ela tem que ter esse número identificador.

As sessões geralmente são usadas pra criar lojas virtuais e sistemas de login, onde o usuário entrará com usuário e senha em um formulário, buscará no banco de dados e se achar algum usuário gravará na sessão uma identificação que dirá que ele já foi logado. Isso somente durará até o fechamento do browser.

Session_start() – é o comando para se utilizar uma sessão. Este comando deve ser utilizado no início do código e antes de qualquer saída html.

Exemplo 1 (pag3.php):

```
<?php
        session_start();
        $_SESSION['nome'] = 'Michele C B';
        $_SESSION['usuario'] = 'mimica';
        $_SESSION['senha'] = '123456';
        // Mostra uma frase na tela
        echo 'Buenos dias ' . $_SESSION['nome'];
?>
```

As variáveis de sessão poderiam estar recebendo variáveis associadas aos campos de uma tabela no BD.

Também podemos usar os valores armazenados em sessões em outras páginas do site.

Exemplo 2:

```
<?php
        session_start();
        echo 'Olá ' . $_SESSION['nome'];
?>
```

Observe que o comando **session_start** se encontra nas duas páginas, pois este comando e o que inicializa as sessões, portanto seu uso é obrigatório.

O exemplo abaixo faz um contador apresentando quantas vezes o usuário acessou a página.

Exemplo 3 (pag4.php):

```
<?php
        session_start();
        echo $_SESSION['nome'] . "<br>";
        if (!isset($_SESSION['visitas']))
                $_SESSION['visitas'] = 1;
        else
                $_SESSION['visitas']++;
        echo "Visita do usuario de numero " . $_SESSION['visitas'];
?>
```

No exemplo 3, o PHP cria uma sessão e busca o nome que foi armazenado em sessão no exemplo 1. Após, o comando if irá verificar se existe a outra sessão chamada de visitas. Se não existir a variável sessão visitas (!isset), o PHP cria a variável sessão armazenando 1, já que tudo indica que é a primeira vez que o usuário está acessando a página. Senão, o else será executado, incrementando +1 na variável de sessão visitas Após, faz a impressão do valor da variável sessão visitas, indicando o total de vezes em que o usuário acessou aquela página.

Sendo assim, criando-se as sessões, os valores poderão ser usados em qualquer página do site, até que as mesmas sejam destruídas do computador.

Agora vamos analisar o exemplo 4. Nos exemplos anteriores sobre sessão, não fizemos a verificação se as mesmas existem. Então faremos isso usando o if.

Exemplo 4 (pag5.php):

```
<?php
	$usuario=$_GET["usuario"];
	session_start();
	$_SESSION['data']=date('d/m/y', time());
	if(!isset( $_SESSION['usuario']) && !isset($_SESSION['senha'])) {
		exit('Usuário não está logado');
	}
	else if(empty($_SESSION['usuario']) && empty($_SESSION['senha'])) {
		exit('Sessão terminada, faça login novamente');
	}
	else if ( $_SESSION['usuario'] != $usuario) {
		exit('Você não é o usuário correto.');
	}
	else {
		echo 'Olá ' . $_SESSION['nome'];
		echo '<p>Você logou em ' . $_SESSION['data'];
	}
?>
```

OBS.: A diferença entre os comandos: isset verifica se a variável ou outro elemento **existe;** empty verifica se **contém algum valor** na variável, ou seja, se ela esta vazia ou não.

O código acima está verificando se as variáveis de sessão existem, se elas não estão vazias e se o usuário é a “mimica", conforme sessão criada no exemplo 1, o PHP apresentará o nome do usuário (variável sessão definida no exemplo 1) e a data do último acesso do usuário, conforme sessão criada neste exemplo, onde recebe dia, mês e ano do dia atual que se encontra na máquina.

Para testar o usuário, até mesmo para verificar se é o mesmo da sessão do exemplo 1, foi criado um formulário para que o usuário pudesse entrar com o usuário para ver se é o mesmo da sessão. Sendo assim, foi criado o arquivo pag6.html.

Para destruir a seção usamos session_destroy(), conforme o exemplo abaixo. Não podemos esquecer de iniciar a sessão para que depois ela possa ser destruída. Depois de executar o exemplo 5 e tentarmos executar o exemplo 4, vamos observar que dará a mensagem “'Usuário não está logado”, já que as variáveis de sessão foram excluídas.

Exemplo 5 (pag.7.php):

```
<?php
        // Inicia a sessão
        session_start();
        session_destroy();
?>
```

As sessões têm um princípio similar *aos* cookies*, só* que o armazenamento do estado é feito pelo servidor *web*, e não pelo navegador.

Maiores informações sobre cookie e session, poderão acessar o link: https://klauslaube.com.br/2012/04/05/entendendo-os-cookies-e-sessoes.html.