

LABORATÓRIO	
ESSENCIAL CONHECIMENTO PHP	
Classe: 12 ^a	Telefone: (+244) 929 379 578
Curso: Informática	Site: www.dsprosystem.co.ao
Eng.MSc Divava Salakiaku	Email: Divava@dsprosystem.co.ao

Introdução

PHP possibilita você inserir funcionalidades avançadas no seu site.

A finalidade deste Laboratório é fazer uma introdução clara e precisa do PHP. Iniciaremos do zero, mas é necessário que você possua um bom conhecimento da linguagem HTML. Caso você não conheça HTML recomendamos começar com a leitura do nosso Laboratório HTML.

PHP pode ser usado em vários contextos. - fóruns de discussão, enquetes, lojas on-line, pontes SMS, listas de discussão, etc. A única limitação para o uso do PHP é a sua imaginação. Não é difícil aprender PHP, mas esteja ciente de que PHP é mais sofisticado e exige mais do que o aprendizado da HTML. Assim, lembre-se que a paciência no processo de aprendizado é uma virtude.

Obviamente este Laboratório não vai lhe ensinar tudo sobre PHP. Será necessário seu engajamento efetivo e realização de experimentos com os códigos mostrados. Se você necessitar de ajuda durante o processo de aprendizado recomendamos ligar para o Mestre.

O que é necessário?

Supõe-se que você tem acesso a um editor de texto e sabe como usá-lo.

Você precisa ter acesso a um computador ou a um servidor capaz de processar PHP. Ao contrário do que ocorre com a HTML e as CSS, para PHP não faz a menor diferença o navegador que o usuário está usando, mas o importante é o tipo de servidor na qual sua página está hospedada. Isto porque PHP é uma tecnologia processada no lado do servidor.

Nas lições seguintes você aprenderá como o PHP funciona e como configurar seu computador para processar PHP. Em seguida você aprenderá sobre funções e métodos do PHP.

Ao término deste Laboratório você estará em condições de desenvolver scripts PHP e em consequência poderá usar as ilimitadas funcionalidades da linguagem para adicionar interatividade às suas páginas web.

Divirta-se!

Lição 1: O que é PHP

Quando se começa o estudo de PHP as primeiras perguntas são: O que é PHP? Como ele funciona?

Nesta lição daremos a resposta a estas duas perguntas. É essencial que você conheça as respostas antes de começar a desenvolver com PHP. O exato conhecimento do que é e de como funciona o PHP servirá de base para acelerar de forma significativa o seu processo de aprendizado.

Bem, vamos começar!

O que é PHP?

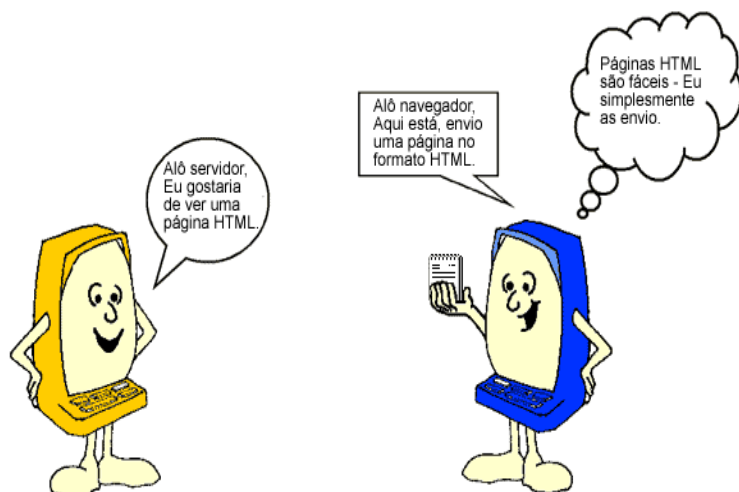
Inicialmente PHP foi um acrônimo para **P**ersonal **H**ome **P**ages, mas posteriormente assumiu o significado de **PHP**: **H**ypertext **P**reprocessor.

PHP foi criado por Rasmus Lerdorf nascido na Groelândia - Dinamarca e posteriormente começou a ser desenvolvido como código livre. PHP não é um Padrão Web - é uma tecnologia de código aberto. PHP não é uma linguagem de programação no sentido estrito da palavra, mas sim uma tecnologia que permite a inserção de scripts nos seus documentos.

A descrição sumária do que seja uma página PHP é que trata-se de um arquivo gravado com a extensão **.php** contendo tags HTML e scripts que são executados em um servidor web.

Como funciona o PHP?

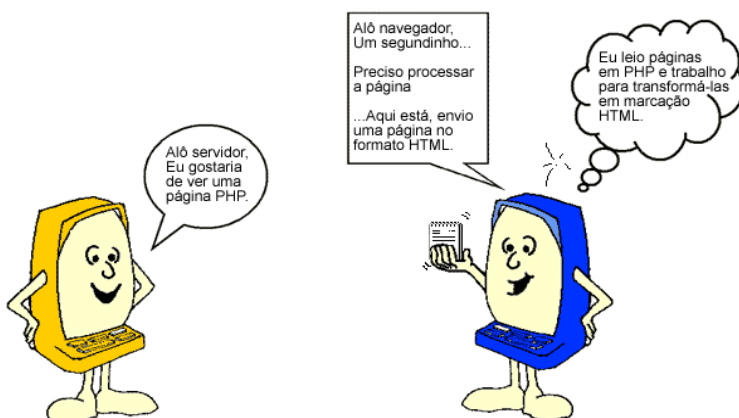
A melhor maneira de explicar como o PHP funciona é comparando-o com a HTML. Suponha que você digite o endereço de um documento HTML (por exemplo: <http://www.meusite.com/page.htm>) na barra de endereços do navegador. Esta ação desencadeia a **requisição** de uma página HTML. A requisição é ilustrada conforme a figura a seguir:



inicia seu trabalho de processamento:

Como você pode observar, o servidor simplesmente envia uma página HTML ao cliente. Suponha agora que você digita **http://www.meusite.com/page.php**

requisitando uma **página PHP** - o servidor



inspecionando o código fonte da página. Você terá que aprender PHP de outras maneiras, por exemplo: lendo nosso Laboratório.

O servidor lê cuidadosamente o arquivo PHP procurando por tarefas a serem por ele executadas. Somente depois de executar eventuais tarefas o resultado é enviado ao cliente. É importante ressaltar que o cliente recebe e vê somente o **resultado** do trabalho do servidor e não as instruções para executar o trabalho.

Se você clicar a funcionalidade do navegador para inspecionar o código fonte de uma página PHP você não verá código PHP - verá somente as tags HTML e seus conteúdos. Assim, não é possível visualizar os scripts PHP inseridos em uma página

Lição 2: Servidores

PHP é uma tecnologia que funciona no *lado do servidor*. Então, você precisa de um servidor para processar PHP. Você não desembolsará um tostão para dispor de um servidor e existem várias opções para obter um.

Opção 1: Servidor remoto de hospedagem

A primeira escolha é hospedar seu site em um servidor de hospedagem que suporte PHP.

- Teste para verificar se o servidor suporta PHP
- Se você ainda não dispõe de um servidor remoto pode criar uma conta gratuita 000webhost.com que suporta PHP.

Opção 2: Instalação de PHP no seu computador

Não existe uma receita de bolo para instalar PHP em um computador. Esta opção é recomendada para usuários com experiência, mas usuário comum não está impedido de escolhê-la. Basta seguir as instruções para download e instalação (em inglês) contidas nos links a seguir:

Opção 3: XAMPP

XAMPP é um programa que habilita PHP no computador sem necessidade de instalação complicada pelo usuário. É indicado para usuários comuns.

Lição 3: Sua primeira página PHP

Nas lições 1 e 2 aprendemos o que é PHP e instalamos um servidor ou vamos usar um servidor remoto para seguir nossas lições. Assim, estamos em condições de criar nossa primeira página PHP. Seguiremos um caminho simples e fácil, contudo ao chegar ao final da lição você entenderá muito mais de PHP e saberá o que pode fazer com ele.

Basicamente um arquivo PHP é um arquivo de texto com a extensão **.php** consistindo de:

- Texto
- Tags HTML
- Scripts PHP

Você já sabe o que são textos e tags HTML. Então vamos examinar os scripts PHP.

Exemplo: Hello World!

Comece gerando um documento HTML e grave-o na raiz do site com o nome *page.php*. Se você usa XAMPP, lição 2, o caminho no seu computador é: "c:\xampp\htdocs\page.php" (ali seu computador é um servidor).

O código HTML deve ser como mostrado a seguir:

```
<html>
<head>
<title>Minha primeira página PHP</title>
</head>
<body>
</body>
</html>
```

Na lição 1 ensinamos que PHP destina-se a **escrever comandos para o servidor**.. Vamos escrever um comando para o servidor.

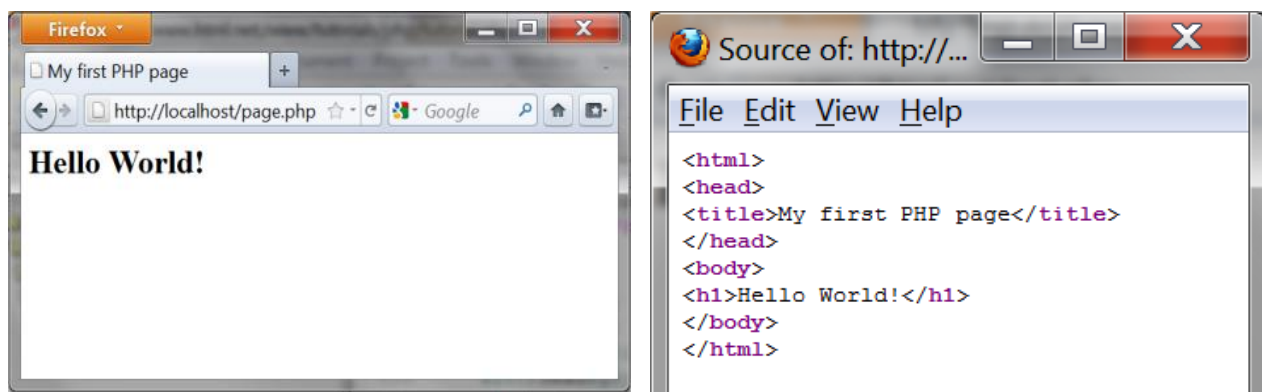
Primeiramente devemos dizer ao servidor onde PHP **começa** e onde **termina**. Para isso usamos as tags **<?php** e **?>** respectivamente, para marcar o início e o fim do código a ser executado pelo servidor (na maioria dos servidores é suficiente escrever **<?** para tag de início, contudo **<?php** é a maneira mais correta e recomendada de se escrever a tag de início do PHP).

Acrescente o seguinte código na sua marcação HTML:

```
<html>
<head>
<title>Minha primeira página PHP</title>
</head>
<body>

<?php

echo "<h1>Hello World!</h1>";
?>
</body>
</html>
```



Se você abrir o documento em um navegador deverá ver algo parecido com o seguinte:

Observe o que acontece quando você examina o código fonte do documento (selecione "view source"):

O código PHP sumiu! Como foi dito na lição 1, apenas o servidor "vê" o código PHP - **o cliente (no caso o navegador) "vê" somente o resultado!**

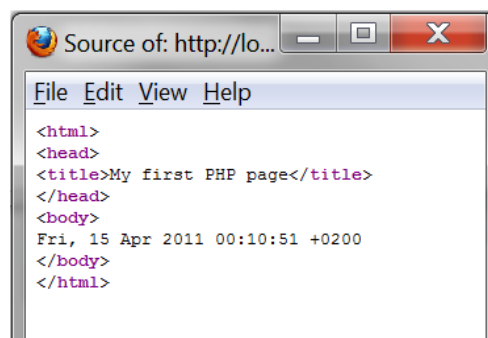
Vamos examinar o que acontece. Pedimos ao servidor para escrever `<h1> Hello World!</h1>`. Em linguagem técnica podemos dizer que usamos a função `echo` para instruir o servidor a escrever uma string para o cliente. O ponto e vírgula termina o comando. Não se apavore! Tentaremos manter o uso de linguagem técnica a um mínimo possível.

Este primeiro exemplo, sem dúvida, não é nada empolgante. Espere um pouco! Daqui para frente as coisas se tornarão cada vez mais empolgantes. Vejamos outro exemplo.

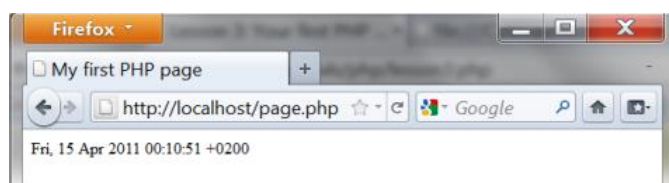
Exemplo: Agora!

Vamos fazer o servidor escrever mais. Podemos, por exemplo, pedir ao servidor para escrever a data e hora atual:

```
<html>
<head>
<title>Minha primeira página PHP</title>
</head>
<body>
<?php
echo date("r");
?>
</body>
</html>
```



O resultado em um navegador é algo como mostrado a seguir:



E o código HTML correspondente é: Está ficando empolgante. Certo?

Fizemos o servidor mostrar a data e hora na página renderizada. Notar que se você recarregar a página uma nova data e hora será mostrada. O servidor escreve a data e hora toda vez que a página é enviada ao cliente. É importante notar que o código HTML renderizado contém apenas a data - não é mostrado o script PHP.

Chamamos a atenção novamente para o ponto e vírgula no final de uma linha de código PHP. Este ponto e vírgula é obrigatório ao final de um comando. Se você esquecer o script não funcionará.

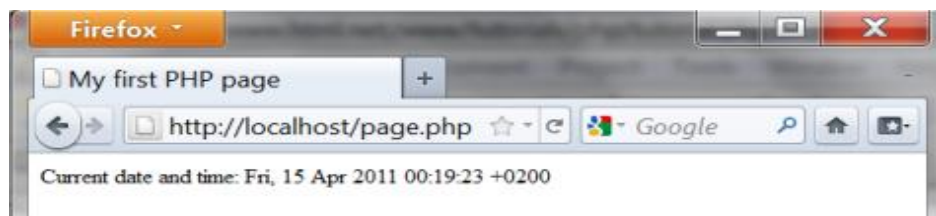
No exemplo nós usamos `date`, que é uma função destinada a retornar a data e hora atuais no servidor.

Vamos ampliar o exemplo escrevendo uma *string* e uma *function* - separadas por um "." (ponto) - observe a seguir:

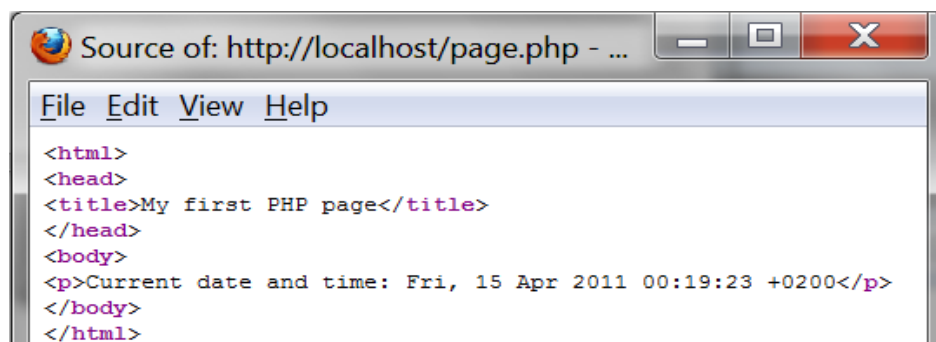
```
<html>
<head>
<title>Minha primeira página PHP</title>
</head>
<body>

<?php
echo "<p>O grupo data/hora atual é: " . date("r") . "</p>";
?>
</body>
</html>
```

O resultado em um navegador é algo como mostrado a seguir:



E o código HTML correspondente é:



Na próxima lição estudaremos com detalhes a função `date` e os diferentes formatos do grupo data/hora.

Lição 4: Trabalhando com datas e horas

Nesta lição você aprenderá as diferentes maneiras de trabalhar com datas e horas usando o PHP. Começaremos mostrando exemplos bem simples com a finalidade de mostrar o que PHP é capaz de fazer. Nesta lição estudaremos com detalhes a função `date` do PHP.


Funções para data e hora

PHP nos fornece uma série de funções relacionadas a data e hora. Nesta lição veremos a mais importante destas funções; a função `date`.

Usando-se os diferentes parâmetros previstos para a função `date` ela retornará o grupo data/hora em diferentes formatos. Os parâmetros mais usuais são:

`date("y")` -> Retorna o ano atual - para o dia de hoje o valor retornado é: **14**
`date("m")` -> Retorna o mês atual - para o dia de hoje o valor retornado é: **03**
`date("F")` -> Retorna o nome do mês atual - para o dia de hoje o valor retornado é: **March**
`date("d")` -> Retorna o dia do mês atual - para o dia de hoje o valor retornado é: **09**
`date("l")` -> Retorna o nome do dia da semana atual - para o dia de hoje o valor retornado é: **Sunday**

date("w")->Retorna o número correspondente ao dia da semana atual - para o dia de hoje o valor retornado é: **0**
date("H")->Retorna a hora atual - para o dia de hoje o valor retornado é: **19**
date("i")->Retorna o minuto atual - para o dia de hoje o valor retornado é: **38**
date("s")->Retorna o segundo atual - para o dia de hoje o valor retornado é: **51**

O exemplo a seguir ilustra o uso da função  [date](#)

```
<html>
<head>
<title>Grupo data/hora</title>


</head>
<body>

<?php

echo "<p>Hoje é " . date("l") . "</p>";

?>

</body>
</html>
```

 [Ver o resultado do exemplo](#)


A hora é: 1394390331

Ops... aqui parece coisa para nerds. A função  [time\(\)](#) retorna a hora atual em número de segundos contados desde 1 de janeiro de 1970 às 12:00 PM, GMT.



```
<html>
<head>
<title>Grupo data/hora</title>
</head>
<body>

<?php

echo "<p>São decorridos exatamente " . time() . " segundos desde 1 de
janeiro de 1970, 12:00 PM, GMT </ p> ";
?>
</body>
</html>
```

 [Ver o resultado do exemplo](#)

A representação da hora em segundos, contados desde 1 de janeiro de 1970 às 12:00 PM, GMT é chamada "timestamp" (UNIX timestamp) que é uma representação muito útil quando se trabalha com datas/horas passadas e futuras.

Por padrão, a função  [date](#) usa o timestamp atual (isto é, o valor corrente de  [time\(\)](#)). Mas, usando um parâmetro você pode especificar o retorno da função em um diferente time stamp e assim trabalhar com datas passadas e futuras. No exemplo a seguir nós definimos o timestamp para 0 segundos a partir de 1 de janeiro de 1970 às 12:00 PM, GMT. Fazendo assim podemos verificar qual foi o dia da semana de 1 de janeiro de 1970.

```
<html>
<head>
<title>Grupo data/hora</title>
</head>
<body>

<?php
```

```


        echo "<p>O dia da semana em 1 de janeiro de 1970 foi " . date("l",0) .
"</p>";


    ?>

</body>
</html>

```

➡ [Ver o resultado do exemplo](#)

A menos que você seja um gênio da matemática é muito complicado calcular o número de segundos passados desde de 1 de janeiro de 1970 para uma determinada data. Felizmente existe uma função chamada  [mktime](#) que realiza o cálculo.

A sintaxe para a função  [mktime](#) é (*hora, minuto, segundo, mês, dia, ano*). O exemplo a seguir faz a conversão para a data da chegada do homem à lua (21 de julho de 1969 às 02:56):

```


<html>
<head>
<title>Grupo data/hora</title>
</head>
<body>

<?php
echo mktime (2,56,0,7,21,1969);
?>
</body>
</html>

```

➡ [Ver o resultado do exemplo](#)

Notar que o retorno foi um número negativo. Isto indica que a data é anterior a 1 de janeiro de 1970.

Agora já podemos no valer do que até aqui aprendemos e usar a função  [date](#) para saber qual foi o dia da semana em que ocorreu o evento histórico da chegada do homem à lua.

```

<html>
<head>
<title>Grupo data/hora</title>
</head>
<body>

<?php


echo date("l", mktime(2,56,0,7,21,1969));

?>
</body>
</html>

```

➡ [Ver o resultado do exemplo](#)

Mas afinal qual a utilidade prática?

Até agora tudo parece muito teórico. Afinal de que maneira posso usar uma do tipo  [time\(\)](#) de uma forma prática? Ao aprender algo novo é importante saber se você poderá usar o que aprendeu em uma página web.

Considere que o que você aprende neste Laboratório é a construção de blocos de códigos - o que você pode fazer com eles depende somente da sua criatividade e está limitado apenas pela sua imaginação! Eu ousaria afirmar que você já aprendeu muito mais do que está pensando. Por exemplo:

você acha que é capaz de criar um background para uma página que mude a cada dia da semana e funcione em todos os navegadores?

Claro que você é capaz! Observe o exemplo a seguir:

```
<html>
<head>
<title>Grupo data/hora</title>
</head>

<body background="background <?php echo date("w"); ?>.png">

</body>
</html>
```

➡ [Ver o resultado do exemplo](#)

O exemplo mostrado, que usa uma imagem de fundo dinâmica, requer que você construa apenas sete imagens e as nomeie como background_1.png, background_2.png, background_3.png, etc.

Se o usuário entra no seu site na terça feira a imagem de fundo será background_2.png e no dia seguinte será background_3.png. Fácil e simples!


Na próxima lição estudaremos os blocos de código destinados a construção de loops e aqueles destinados a repetição de código.

PHP é legal. Você não acha?

Lição 5: Loops

Na linguagem PHP existem diferentes estruturas de controle destinadas a gerenciar a execução de scripts. Nesta lição nós estudaremos os loops. Loops são usados para executar repetidamente uma parte de um script em um determinado número de vezes ou até que seja encontrada uma determinada condição.

Loop "while"

A sintaxe para um loop  **while** é conforme mostrada a seguir:

```
while (condição) {
    Comandos PHP de execução
}
```

Esta sintaxe pode ser traduzida para linguagem corrente como: *execute comandos PHP **enquanto** (**while**) a condição for satisfeita.*

Vejam os um exemplo bem simples:

```
<html>
<head>
<title>Loops</title>

</head>
<body>

<?php

$x = 1;
```



```

while ($x <= 50) {
    echo "<p>Este texto se repete 50 vezes</p>";
    $x = $x + 1;
}
?>

</body>

</html>


```

➡ [Ver o resultado do exemplo](#)

No exemplo mostrado usamos uma *variável* denominada "\$x". Como você pode notar, nomes de variáveis em PHP sempre começam com o caractere "\$". No início é fácil esquecer esta sintaxe, contudo é absolutamente necessário o símbolo "\$" para iniciar o nome de variáveis, pois do contrário o script não funciona.

Além disto o restante do script é auto-explicável. No começo a variável \$x recebe o valor 1. A seguir o loop pede para o servidor executar o comando de escrever um texto enquanto a variável for menor do que 50. A cada execução a variável é incrementada de 1 unidade.

Loop "for"

Outra maneira de executar um loop é com uso de  `for` conforme mostrado a seguir:

```

for (Inicialização; Condição; passo) {
    Comandos PHP de execução
}

```

Os comandos PHP de execução se repetem com a 'Inicialização' + 'passo' enquanto a 'Condição é satisfeita'. Se isso não faz sentido para você o exemplo a seguir esclarece melhor:

```

<html>
<head>

<title>Loops</title>
</head>
<body>

<?php

for ($x=0; $x<=50; $x=$x+5) {
    echo '<p>A variável $x agora tem o valor igual a ' . $x . '</p>';
}
?>

</body>
</html>

```

➡ [Ver o resultado do exemplo](#)

No exemplo mostrado \$x é incrementado de 5 unidades em cada loop. O loop continua sua execução enquanto \$x for menor ou igual a 50. Notar que o valor de \$x é usado como parte do texto a ser escrito pelo script.

Observe outro exemplo:

```

<html>
<head>

<title>Loops</title>
</head>

```

```

<body>

<?php

for ($x=1; $x<=6; $x=$x+1) {
    echo "<h" . $x . "> Cabeçalho nível</h" . $x . ">";
}
?>

</body>
</html>

```

➡ [Ver o resultado do exemplo](#)

Você entendeu? começamos definindo o valor de \$x igual a 1. A seguir em cada loop escrevemos um cabeçalho nível \$x (h1, h2, h3, etc.) até \$x alcançar o valor seis.

Loops dentro de loops

A princípio não existe limite para a quantidade de loops a usar. Você pode facilmente aninhar loops e criar **muitas** repetições.

Mas, cuidado! A execução de PHP torna-se lenta quando desenvolvemos scripts extensos e complicados. Por exemplo: observe a seguir o script com três loops capaz de escrever 16 milhões de cores!

Com a finalidade de não tornar a página de carregamento muito lento nós reduzimos drasticamente o número de repetições para passos de 30 limitando o número de cores escritas para 512.

```

<html>

<head>
<title>Loops </title>
</head>
<body>

<?php

for ($intRed=0; $intRed<=255; $intRed=$intRed+30) {

    for ($intGreen=0; $intGreen<=255; $intGreen=$intGreen+30) {

        for ($intBlue=0; $intBlue<=255; $intBlue=$intBlue+30) {

            $StrColor = "rgb(" . $intRed . "," . $intGreen . "," . $intBlue
. ")";

            echo "<span style='color:" . $StrColor . "'>" . $StrColor .
"</span>";

        }
    }
}
?>

</body>
</html>

```

➡ [Ver o resultado do exemplo](#)


No exemplo mostrado cada uma das três cores primárias (red, green e blue) pode ter um valor entre 0 e 255. Qualquer combinação de três valores resulta em uma cor do tipo rgb(255,255,255). O código da cor é escrito em um elemento que por sua vez é estilizado com a respectiva cor.

Loops tornam-se mais interessantes e práticos depois que você aprende mais algumas funcionalidades do PHP. Assim que você entender o princípio de funcionamento do loop passe para a lição seguinte na qual estudaremos as condicionais.

Lição 6: Condicionais

Condicionais são usadas para executar um bloco de script sempre que determinada condição seja satisfeita. Por exemplo: uma condição pode estabelecer que uma data seja posterior a 1 de janeiro de 2012 ou que uma variável seja maior do que 7.

If...

A primeira condicional que estudaremos é o  `if` cuja sintaxe é mostrada a seguir:

```
if (condition) {  
    Comandos PHP  
}
```



Mais uma vez a sintaxe se parece com a linguagem corrente: **If (Se)** se a condição é satisfeita, execute alguma coisa. Vejamos um exemplo simples:

```
<html>  
<head>  
<title>Loops </title>  
</head>  
<body>  
  
<?php  
  
$x = 2;  
  
if ($x > 1) {  
    echo "<p>A variável $x é maior que 1 </p>";  
}  
  
?>  
</body>  
</html>
```

if ... else ...

Vejamos agora a condicional  `else` cuja sintaxe é mostrada a seguir:

Mais uma vez a sintaxe se parece com a linguagem corrente: **if (se)** a condição é satisfeita, execute alguma coisa **else (se não)** execute outra coisa.

Na [Lição 4](#) mostramos como encontrar o número representativo do mês. No exemplo a seguir usaremos este número em uma condicional  `if`  `else` para encontrar em qual das estações do ano estamos:

```
<html>  
<head>  
<title>Condicionais</title>  
</head>  
<body>  
  
<?php  
  
if (date ("m") == 3) {
```

```

        echo "<p>Estamos no outono!</p> ";
    }
    else {
        echo "<p>Eu não sei em que estação estamos!</p> ";
    }

    ?>
</body>
</html>

```

➔ Ver o resultado do exemplo

Notar que se trata de uma condicional não muito inteligente - ela só funciona para o Mês de Março!

Contudo, existem várias maneiras de aperfeiçoar a condicional tornando-a mais precisa. Observe a seguir alguns operadores de comparação que podemos usar na condicional mostrada:

== Igual, < Menor que, > Maior que <= Menor ou igual a, >= Maior ou igual a, != Diferente

Existem também operadores lógicos: && e || ou ! não

Operadores se destinam a criar condicionais mais precisas e com seu uso podemos melhorar o exemplo mostrado anteriormente fazendo com que o retorno do script seja a estação primavera em todos os meses que ela ocorre e não somente em março:

```

<html>
<head>
<title>Condicionais</title>

</head>
<body>

<?php

    if (date("m") >= 3 && date("m") <= 5) {
        echo "<p> Estamos no outono!</p> ";
    }
    else {
        echo "<p> A estação atual é primavera, verão ou inverno!</p> ";
    }

    ?>
</body>
</html>

```

Vamos examinar estas novas condicionais:

```
date("m") >= 3 && date("m") <= 5
```


Pode ser traduzida como:

Se o número que representa o mês for maior ou igual a 3 e menor ou igual a 5

Legal não é? Operadores são largamente usados em diferentes blocos de script do PHP.

Contudo nosso exemplo só funciona para os meses de março, abril e maio, Os demais meses não são contemplados pela condicional. Assim, vamos aperfeiçoá-lo mais:

if ... elseif ... else...

Usando  elseif podemos expandir a condicional e fazer com que nosso script funcione para todos os meses do ano:

```
<html>
<head>
<title>Condicionais</title>

</head>
<body>

<?php

if (date("m") >= 3 && date("m") <= 5) {
    echo "<p>Estamos no outono!</p>";
}


elseif (date("m") >= 6 && date("m") <= 8) {
    echo "<p>Estamos no inverno!</p>";
}

elseif (date("m") >= 9 && date("m") <= 11) {
    echo "<p>Estamos na primavera!</p>";
}

else {
    echo "<p>Estamos no verão!</p>";
}

?>

</body>
</html>
```

 [Ver o resultado do exemplo](#)

Escrever condicionais é uma questão de lógica e método. O exemplo mostrado é bem esclarecedor, porém o uso de condicionais pode tornar-se bem mais complexo.

switch ... case

Outra maneira de se escrever condicionais é com uso do método  switch:

```
switch (expressão) {

case 1:
    comandos PHP
    break;
case 2:
    comandos PHP
    break;
default:
    comandos PHP
    break;
}
```

Este método toma por base uma **expressão** e a seguir relaciona uma série de "respostas" ou "valores" e respectivos **comandos PHP**. A maneira mais fácil de entender esta condicional é observando um exemplo.

Na lição 4 nós vimos que a função `date("w")` retorna um número representando dia da semana. Vamos mostrar um exemplo para escrever o nome do dia da semana em lugar do número que o representa:


```
<html>
<head>
<title>Condicionais</title>
</head>
<body>

<?php

switch(date("w")) {

case 1:
    echo "Hoje é segunda-feira";
    break;
case 2:
    echo "Hoje é terça-feira";
    break;
case 3:
    echo "Hoje é quarta-feira";
    break;
case 4:
    echo "Hoje é quinta-feira";
    break;
case 5:
    echo "Hoje é sexta-feira";
    break;
case 6:
    echo "Hoje é sábado";
    break;
default:
    echo "Hoje é domingo";
    break;
}

?>
</body>
</html>
```

 [Ver o resultado do exemplo](#)

Lição 7: Comentando seus scripts

Por que é importante comentar os scripts?

Quando codificamos, estamos escrevendo comandos para um servidor/computador usando uma linguagem estritamente formal que pode não refletir claramente o que você estava pensando quando criou o script. No mundo dos negócios é comum as Companhias exigirem que os scripts e programas sejam comentados, pois é considerado um alto risco aceitar um sistema no qual a identificação e correção de erros se torna uma tarefa difícil pela falta de comentários.

Como inserir comentários?

É fácil inserir um comentário. Você simplesmente começa um texto de comentário escrevendo duas barras: `"/`.

Observe um exemplo mostrado na lição 5 no qual inserimos comentários: Então! Não se esqueça de comentar seus scripts.

Lição 8: Arrays

Nesta lição veremos o que array, como usá-los e o que podemos fazer com eles.

Entender arrays pode ser um pouco difícil no início. Mas, não desanime, vamos tentar... nós iremos tornar o processo de aprendizado o mais fácil possível.

O que é array?

Array é uma coleção (ou conjunto) de elementos indexados na qual cada um dos elementos tem um número identificador único.

Parece confuso? Creia, não é tão complicado.

Imagine uma lista de palavras separadas por vírgula como a mostrada a seguir:

```
maçãs, peras, bananas, laranjas, limões
```

Agora imagine dividir a lista tomando como separador cada vírgula. Em seguida atribua a cada divisão um número identificador único:

maçãs	peras	bananas	laranjas	limões
0	1	2	3	4

O que você acabou de ver é um array. Podemos dar um nome para o array como, por exemplo, "frutas". A ideia é que possamos acessar o array usando um número identificador e com ele consultar o valor correspondente usando uma sintaxe como mostrada a seguir:

```
frutas(0) = maçãs  
frutas(1) = peras  
frutas(2) = bananas  
frutas(3) = laranjas  
frutas(4) = limões
```

esta é a ideia por trás de arrays. Vamos ver um exemplo prático.

Como usar um array?

Continuaremos com o array de frutas. Passo a passo mostraremos como fazer para a lista funcionar como um array. Primeiro vamos criar uma variável para conter a lista como mostrado a seguir:

```
<?php  
$listadefrutas = "maçãs, peras, bananas, laranjas, limões";  
?>
```

A seguir vamos usar a função  `explode` para dividir a lista pelas vírgulas:

```
<?php  
$listadefrutas = "maçãs, peras, bananas, laranjas, limões";  
  
$arrFrutas = explode(",", $listadefrutas);  
?>
```

Viva! "\$arrFrutas" agora é um array!

Observe que chamamos a função  `explode` com dois argumentos:

1. a lista a ser dividida
2. e o delimitador - isto é, o caractere usado para separar os itens da lista (no caso do exemplo a vírgula) colocado entre aspas ",".

Usamos a vírgula como delimitador, mas podemos usar qualquer caractere e até mesmo uma palavra.

Vamos comentar o script e colocá-lo em uma página PHP:

```
<html>
<head>
<title>Array</title>
</head>
<body>

<?php


// Lista separada por vírgula
$listadefrutas = "maças, peras, bananas, laranjas, limões";

// Cria um array seprando os itens da lista (tendo a vírgula como
delimitador)
$arrFrutas = explode(",", $listadefrutas);

// Escreve os valores do array
echo "<p>Lista de frutas:</p>";

echo "<ul>";
echo "<li>" . $arrFrutas[0] . "</li>";
echo "<li>" . $arrFrutas[1] . "</li>";
echo "<li>" . $arrFrutas[2] . "</li>";
echo "<li>" . $arrFrutas[3] . "</li>";
echo "<li>" . $arrFrutas[4] . "</li>";
echo "</ul>";

?>
</body>
</html>
```

 [Ver o resultado do exemplo](#)

este exemplo é muito simples e na verdade não há vantagem alguma em usar um array para realizar este tipo de tarefa. Mas, um momento... arrays podem ser usadas de maneira muito mais vantajosa.

Loop por um array

Na [lição 5](#) estudamos loops. A seguir veremos como realizar um loop por um array.

Quando você conhece o número de itens de um array não encontra problemas para definir um loop por ele. Você começa em 0 e desenvolve o loop até atingir o número de itens do array. No exemplo das frutas um loop pelo array seria como mostrado a seguir:

```
<html>
<head>
<title>Array</title>
</head>
<body>
<?php

// Lista separada por vírgula
```



```

    $listadefrutas = "maçãs, peras, bananas, laranjas, limões";

    // Cria um array separando os itens da lista (tendo a vírgula como
delimitador)
    $arrFrutas = explode(",", $listadefrutas);

    echo "<p>Lista de frutas:</p>";
    echo "<ul>";

    // Loop pelo array $arrFrutas
    for ($x=0; $x<=4; $x++) {
        echo "<li>" . $arrFrutas[$x] . "</li>";
    }
    echo "</ul>";
?>
</body>
</html>

```


➡ [Ver o resultado do exemplo](#)

Como você pode notar a variável `$x` (que cresce de 0 a 4 no loop) foi usada para chamar o array.

Como encontrar o tamanho de um array

Mas, o que acontecerá se uma outra fruta for adicionada à lista? Nosso array passará a ter mais um item cujo número identificador será **5**. Já deu para ver o problema? Precisamos alterar o loop para que ele funcione de 0 a 5, caso contrário não serão incluídos todos os elementos do array.

Não seria maravilhoso se pudéssemos saber automaticamente quantos itens um array tem?

É isso exatamente o que faremos a seguir com uso da função  [foreach](#). Agora podemos criar um loop que funciona em todo array independentemente do número de itens nele contido:

```

<?php
    foreach ($arrFrutas as $x) {
        echo $x;
    }
?>

```

este loop funciona independentemente do número de itens contido no array.

Outro exemplo

A seguir mostramos um exemplo, usando array, para escrever os nomes dos meses:

```

<html>
<head>
<title>Array</title>

</head>
<body>

<?php
    // Cria um array dos meses.
    // Cria um array com os meses. Notar a vírgula antes do mês de janeiro.
Isto é necessário //porque não existe mês representado pelo número
    $arrMes =
array("", "Janeiro", "Fevereiro", "Março", "Abril", "Maio", "Junho", "Julho", "Agosto", "
Setembro", "Outubro", "Novembro", "Dezembro");

    // Chama o array com o número do mês - escreve no navegador do usuário

```

```
echo $arrMes[date("n")];  
?>  
</body>  
</html>
```

➡ [Ver o resultado do exemplo](#)

Notar que usamos a função `array` e não `explode` para criar o array.

Ok. Isto é tudo sobre arrays! Na próxima lição veremos como escrever nossas próprias funções.

Lição 9: Funções

Nas lições anteriores você aprendeu a usar funções, tais como, `date()` e `array()`. Nesta lição você aprenderá a criar suas próprias funções usando a funcionalidade do PHP chamada `function`.

O que é função?

Uma função se destina a processar *inputs (entradas)* e retornar um *output (saída)*. Isto pode ser muito útil se, por exemplo, você tiver que processar uma grande quantidade de dados ou se tiver que realizar cálculos ou rotinas que devam ser executadas muitas vezes.

A sintaxe geral para uma função é mostrada a seguir:

```
Nome da função(lista de parâmetros) {  
    Comandos PHP  
}
```

O exemplo a seguir esclarece a sintaxe para uma função bem simples destinada a adicionar 1 unidade a um número:

```
function AddOne($x) {  
    $x = $x + 1;  
    echo $x;  
}
```

O nome escolhido para nossa função foi **AddOne** e deve ser chamada com um parâmetro que é o número a somar - por exemplo: 34....

```
echo AddOne(34);
```

... o retorno da função será (pasmem!) 35.

No exemplo mostrado processou-se um número, contudo as funções podem processar textos, datas e qualquer outro tipo de dado. Você pode, até mesmo, criar funções para serem chamadas por parâmetros. Nesta lição estudaremos diferentes tipos de funções.

Exemplo 1: Função com vários parâmetros

Como dito anteriormente é possível criar funções com vários parâmetros. No exemplo a seguir criaremos uma função que é chamada com 3 números como parâmetros e retorna a soma deles:

```
<html>  
<head>  
<title>Funções</title>  
  
</head>  
<body>
```

```

<?php

function AddAll($number1,$number2,$number3) {
    $plus = $number1 + $number2 + $number3;
    return $plus;
}

echo "123 + 654 + 9 é igual a " . AddAll(123,654,9);

?>

</body>
</html>

```

[!\[\]\(1d3a1175dd4902218e694b9c098adb83_img.jpg\) Ver o resultado do exemplo](#)

Ok. Isto foi bastante simples! Mas, o objetivo foi apenas mostrar que uma função pode ser chamada com vários parâmetros.

Exemplo 2: Data e hora em inglês

Vamos criar uma função um pouquinho mais complexa. Uma função a ser chamada com date() e time() e retornar a data e hora no formato: **Wednesday, 15 February, 2012, 10:00:00 AM**

```

<html>
<head>
<title>Funções</title>
</head>
<body>

<?php

function EnglishDateTime($date) {

    // Array com o nome em português dos dias e semanas
    $arrDay = array("segunda-feira","terça-feira","quarta-feira","quinta-
feira","sexta-feira","sábado","domingo");

    // Array with the English names of the months
    $arrMonth =
array("", "janeiro", "fevereiro", "março", "abril", "maio", "junho", "julho", "agosto", "
setembro", "outubro", "novembro", "dezembro");

    // Montar a data
    $EnglishDateTime = $arrDay[(date("w",$date))] . ", " .
date("d",$date);
    $EnglishDateTime = $EnglishDateTime . " " .
    $arrMonth[date("n",$date)] . " " . date("Y",$date);
    $EnglishDateTime = $EnglishDateTime . ", " . date("H",$date) . ":" .
date("i",$date);

    return $EnglishDateTime;

}

// Função de teste
echo EnglishDateTime(time());

?>

</body>
</html>

```

 [Ver o resultado do exemplo](#)

Notar que a definição de '\$arrMonth' e '\$EnglishDateTime' constam de diversas linhas do script. Fizemos assim para que usuários com baixa resolução de tela tenham uma melhor visão do exemplo. Isto não tem nenhum efeito no funcionamento do código.

A função mostrada funciona em qualquer servidor independentemente do idioma. Isto significa que você pode usar esta função no seu site mesmo que ele esteja hospedado em um servidor na França e você quiser mostrar suas datas no formato inglês.

Por ora ficaremos por aqui sem nos aprofundar mais, pois agora você já sabe alguma coisa sobre como as funções trabalham.

Lição 10: Passando variáveis no URL

Quando se trabalha com PHP é comum a necessidade de se passar variáveis de uma página para outra. Nesta lição veremos como passar variáveis em um URL.

Como isto funciona?

Talvez você já tenha visto e ficado intrigado com URLs no formato parecido com o mostrado a seguir:

```
http://html.net/page.php?id=1254
```

O que significa aquele sinal de interrogação depois do nome da página?

A resposta é: os caracteres depois do sinal de interrogação são uma **HTTP query string**. Uma HTTP query string contém variáveis e seus valores. No exemplo mostrado a HTTP query string contém uma variável chamada "id", e seu respectivo valor igual a "1254".

Observe outro exemplo: `http://html.net/page.php?name=vandersá`

Aqui você tem uma variável ("name") com o valor ("vandersá").

Como recuperar o valor de uma variável com PHP?

Suponha uma página PHP chamada **people.php**. Suponha, também, que você "chame" esta página usando o seguinte URL: `people.php?name= vandersá`

A sintaxe PHP para recuperar o valor da variável 'name' passada no URL é mostrada a seguir: `$_GET["name"]`

Use  `$_GET` para recuperar o valor de uma variável. Vejamos outro exemplo:

```
<html>
<head>
<title>Query string</title>
</head>
<body>

<?php

// Extrai o valor da variável name
echo "<h1>Olá " . $_GET["name"] . "</h1>";
?>
</body>
</html>
```

➡ Ver o resultado do exemplo (observe o URL)

Ao abrir a página que demonstra este exemplo faça a seguinte experiência: substitua na barra de endereços do navegador o nome "Joe" pelo seu nome e recarregue a página! Legal, não é?

Várias variáveis no mesmo URL

Você não está limitado a passar uma só variável no URL. Para passar mais de uma variável use o sinal **&** (e comercial) para separar as variáveis, como mostrado a seguir:

```
people.php?name=Joe&age=24
```

Este URL contém duas variáveis: name (nome) e age (idade). Tal como foi explicado anteriormente para recuperar as duas variáveis a sintaxe é mostrada a seguir:

```
$ _GET["name"]  
$ _GET["age"]
```

Vamos usar mais uma variável no nosso exemplo:

```
<html>  
<head>  
<title>Query string </title>  
</head>  
<body>  
  
<?php  
  
// Extrai o valor da variável name  
echo "<h1>Olá " . $ _GET["name"] . "</h1>";  
  
// Extrai o valor da variável age  
echo "<h1>Você tem ". $ _GET["age"] . " anos de idade</h1>";  
  
?>  
  
</body>  
</html>
```

➡ Ver o resultado do exemplo (keep an eye on the URL)

Bem, agora você já sabe como passar variáveis em um URL. Na próxima lição veremos mais um método de passagem de variáveis: o método usando formulários.

Lição 11: Passando variáveis com uso de formulários

Sites interativos se baseiam em entrada de dados pelos usuários. Uma das formas mais comuns de coletar dados e com uso de formulários

Nesta lição estudaremos como construir formulários e processar seus dados no servidor.


<form>

Para o elemento form destinado a marcar um formulário estão previstos, entre outros, dois atributos de suma importância, são eles: **action** e **method**.

action

Destina-se a definir o URL para o qual os dados do formulário serão enviados para processamento. Neste URL encontra-se o arquivo com o script de processamento dos dados.

method

Este atributo admite os valores "post" e "get" que são dois métodos deferentes de passagem de dados. Por enquanto você não precisa saber as diferenças entre estes métodos, basta saber que com "get" os dados são passados pelo URL e com post "post" são enviados em forma de um bloco de dados via um mecanismo denominado STDIN. Na lição anterior estudamos como recuperar dados passados no URL com uso de  `$ _GET`. Nesta lição veremos como recuperar dados passados com uso do método "post".

Formulário em uma página HTML

A página contendo o formulário não precisa, necessariamente, ser uma página PHP. Não precisa nem mesmo estar no mesmo site que recebe seus dados para processamento.

O nosso primeiro exemplo mostra uma página contendo um formulário bem simples contendo apenas um campo de entrada de dados:


```
<html>
<head>
<title>Formulário</title>
</head>
<body>

<h1>Informe seu nome</h1>

<form method="post" action="handler.php">
<input type="text" name="username">
<input type="submit">
</form>
</body>
</html>
```

A renderização no navegador é conforme mostrada a seguir:

Agora vem a parte divertida: receber e manipular dados com PHP

Para requisitar dados enviados por um formulário usamos  `$ _POST`:

```
$ _POST["fieldname"];
```

Retorna o valor de um campo do formulário. Vejamos um exemplo.

Crie uma página contendo o formulário mostrado anteriormente. Crie outra página chamada "handler.php" (notar que este nome é o mesmo daquele constante como valor do atributo action da tag <form> no nosso formulário).

O arquivo "handler.php" é como mostrado a seguir:

```
<html>
<head>
<title>Formulário</title>
</head>
<body>
<?php

echo "<h1>Olá " . $_POST["username"] . "</h1>";

?>
</body>
</html>
```

➡ [Ver o resultado do exemplo](#)

Dados do usuário e condicionais

No exemplo a seguir manipularemos os dados do usuário com condicionais. Nosso formulário será com mostrado a seguir:

```
<html>
<head>
<title>Formulário</title>
</head>
<body>

<form method="post" action="handler.php">

<p>Qual é o seu nome:</p>
<input type="text" name="username"></p>

<p>Qual é a sua cor favorita:
<input type="radio" name="favoritecolor" value="r" /> Vermelha
<input type="radio" name="favoritecolor" value="g" /> Verde
<input type="radio" name="favoritecolor" value="b" /> Azul </p>

<input type="submit" value="Enviar" />
</form>
</body>
</html>
```

A renderização no navegador é conforme mostrada a seguir:

A seguir mostraremos um script que usa o dado relativo a preferência de cor do usuário para mudar a cor do fundo da página conforme sua preferência. Faremos isso criando uma condicional (ver [lição 6](#)) que usa a cor escolhida pelo usuário no formulário.

```
<?php
$strHeading = "<h1>Olá " . $_POST["username"] . "</h1>";

switch ($_POST["favoritecolor"]) {
case "r":
    $strBackgroundColor = "rgb(255,0,0)";
    break;
case "g":
    $strBackgroundColor = "rgb(0,255,0)";
    break;
case "b":
    $strBackgroundColor = "rgb(0,0,255)";
    break;
default:
    $strBackgroundColor = "rgb(255,255,255)";
    break;
}
?>
<html>
<head>
<title>Formulário</title>
</head>
<body style="background: <?php echo $strBackgroundColor; ?>;">

<? echo $strHeading; ?>
</body>
</html>
```

O fundo será branco se o usuário não escolher uma cor. Isto é feito com o uso da declaração **default** que define a ação a tomar se nenhuma das condições for satisfeita.

Mas, o que acontece se o usuário não fornece seu nome? O resultado será somente "Hello". Vamos usar mais uma condicional para resolver esta questão.

```
<?php

$strUsername = $_POST["username"];

If ($strUsername != "") {
    $strHeading = "<h1>Olá " . $_POST["username"] . "</h1>";
}
else {
    $strHeading = "<h1>Olá visitante!</h1> ";
}

switch ($_POST["favorite color"]) {
case "r":
    $strBackgroundColor = "rgb(255,0,0)";
    break;
case "g":
    $strBackgroundColor = "rgb(0,255,0)";
    break;
case "b":
    $strBackgroundColor = "rgb(0,0,255)";
    break;
default:
    $strBackgroundColor = "rgb(255,255,255)";
    break;
}

?>


<html>

<head>

<title>Formulário</title>
</head>
<body style="background: <?php echo $strBackgroundColor; ?>;">

<? echo $strHeading; ?>

</body>
</html>
```

 [Ver o resultado do exemplo.](#)

Lição 12: Sessão

Quando você visita um site você realiza uma série de ações. Navega de uma página a outra. Talvez você preencha um formulário ou compre um produto. Como desenvolvedor essas informações são de grande valia para criação de um site de sucesso. Suponha que você necessita criar um site no qual algumas páginas serão de acesso restrito requerendo login e senha. Para que a proteção ao acesso seja efetiva as páginas restritas devem possuir mecanismos capazes de detectar se o usuário está logado. Em outras palavras, quando o usuário requisita uma nova página do site é preciso "lembrar" o que ele fez anteriormente.

É exatamente para isto que as sessões se prestam - como usar a funcionalidade sessions do PHP para armazenar e recuperar informações sobre um usuário em visita ao site.

Session

As session do PHP permitem gerenciar informações colhidas durante uma sessão (estada no site) do usuário.

Exemplo de uso de sessions

```
<html>
  <head>
    <title>Imprimir session</title>
  </head>
  <body>
    <?php
      //Inicializando session
      session_start();
      $id = session_id();
      print("Session Id: ".$id);
    ?>
  </body>
</html>
```

Vamos examinar outro exemplo de aplicação para sessão: uso de senha.

Sistema de login com uso de sessão

No exemplo a seguir iremos criar um sistema simples de login. Usaremos muitos dos conceitos que aprendemos até aqui.

Precisamos de um formulário para coletar login e senha do usuário. Observe a seguir:

```
<html>
<head>
<title>Login</title>

</head>
<body>
<form method="post" action="login.php">
<p>Usuário: <input type="text" name="username" /></p>
<p>Senha: <input type="text" name="password" /></p>

<p><input type="submit" value="Login" /></p>

</form>
</body>
</html>
```

A seguir criamos um arquivo denominado: login.php.

Neste arquivo o script verifica se o login e senha fornecidos no formulário estão corretos. Se estiverem, iniciamos uma sessão informando que o usuário está logado com um login e senha correta.

```
<html>

<head>
<title>Login</title>
```

```

</head>
<body>
<?php

// Verifica se usuário e senha conferem
if ($_POST["username"] == "php" && $_POST["password"] == "php") {

// Se usuário e senha conferir definimos session para YES
    session_start();
    $_SESSION["Login"] = "YES";
    echo "<h1>Você está logado</h1>";
    echo "<p><a href='document.php'>Link para o arquivo restrito</a><p/>";

}
else {

// Se usuário e senha conferir definimos session para NO
    session_start();
    $_SESSION["Login"] = "NO";
    echo "<h1>Você NÃO está logado</h1>";
    echo "<p><a href='document.php'>Link para o arquivo restrito</a><p/>";

}
?>

</body>
</html>

```

Nas páginas de acesso restrito precisamos verificar se o usuário que as requisitou está logado corretamente. Se não estiver ele é enviado para a página de login. Observe como é feito o script de proteção das páginas:

```


<?php
// Iniciar Session PHP
session_start();

// Se o usuário não estiver logado manda ele para o formulário de login
if ($_SESSION["Login"] != "YES") {
    header("Location: form.php");
}
?>
<html>
<head>
<title>Login</title>
</head>

<body>
<h1>Este é um documento de acesso restrito</h1>

<p>Acesso permitido somente para usuários logados.</p>
</body>
</html>

```

 [Click aqui para testar o sistema de login](#)

Para Destroi a sessao de um acesso

```

<?php
session_destroy();
?>

```

A seguir está outro exemplo desta função, aqui temos duas páginas do mesmo aplicativo na mesma sessão.

session_page1.htm

```
<?php
    if(isset($_POST['SubmitButton'])) {
        //Starting the session
        $id = session_create_id();
        session_id($id);
        print("\n"."Id: ".$id);
        session_start();
        $_SESSION['name'] = $_POST['name'];
        $_SESSION['age'] = $_POST['age'];
        session_commit();
    }
?>
<html>
    <body>
        <form action="#" method="post">
            <label for="fname">Entra com valores apos click Submitter e click para
Next</label>
            <br><br><label for="fname">Name:</label>
            <input type="text" id="name" name="name"><br><br>
            <label for="lname">Age:</label>
            <input type="text" id="age" name="age"><br><br>
            <input type="submit" name="SubmitButton"/>
            <?php
                echo '<br><br /><a href="session_page2.htm">Next</a>';
            ?>
        </form>
    </body>
</html>
```

Ao clicar em **Avançar**, o seguinte arquivo é executado.

session_page2.htm

```
<html>
    <head>
        <title>Second Page</title>
    </head>
    <body>
        <?php
            //Session started
            session_start();
            print("Values from the session with id: ".session_id());
            echo "<br>";
            print($_SESSION['name']);
            echo "<br>";
            print($_SESSION['age']);
        ?>
    </body>
</html>
```

Lição 13: Sistema de arquivos

Usando PHP você pode acessar o sistema de arquivos no servidor. Esta funcionalidade permite que você manipule diretórios e arquivos de texto com scripts PHP.

Você pode usar PHP para ler um arquivo de texto ou mesmo nele escrever. Ou ainda, você pode inspecionar quais são os arquivos existentes em um determinado diretório no servidor. As possibilidades são muitas e o PHP pode facilitar bastante o seu trabalho.

filemtime

Retorna a hora em que o conteúdo de um arquivo foi editado pela última vez (no formato UNIX timestamp - ver lição 4)).

fileatime

Retorna a hora em que o conteúdo de um arquivo foi acessado (aberto) pela última vez (no formato UNIX timestamp - ver lição 4)).

filesize

Retorna o tamanho do arquivo em bytes.

Vamos verificar as três propriedades descritas para o arquivo que você está lendo agora: `/Laboratórios/php/lesson14.php`


```
<html>
<head>
<title>Sistema de arquivos</title>
</head>
<body>

<?php

    // Pesquisar e escrever propriedades
    echo "<h1>Arquivo: lesson14.php</h1>";
    echo "<p>Editado pela última vez em: " . date("r",
filemtime("lesson14.php"));
    echo "<p>Aberto pela última vez em: " . date("r",
fileatime("lesson14.php"));
    echo "<p>Tamanho do arquivo: " . filesize("lesson14.php") . " bytes";

?>

</body>
</html>
```

 [Ver o resultado do exemplo](#)

Lição 14: Ler arquivos de texto

Na lição anterior estudamos como acessar o sistema e arquivos do servidor. Nesta lição aplicaremos o conhecimento adquirido para fazer a leitura de um arquivo de texto no servidor.

Arquivos de texto são uma maneira poderosa de armazenar vários tipos de dados. Eles não são tão flexíveis quanto um Banco de Dados, mas têm a vantagem de não requerer muita memória. Além disso por ser em formato de texto puro funcionam na grande maioria dos sistemas.

Abrir um arquivo de texto

Usamos a função  **fopen** para abrir um arquivo de texto. A sintaxe é mostrada a seguir:

```
fopen(filename, mode)
```

filename

Nome do arquivo a abrir.


mode

Define o modo, que pode ser: "r" (reading - ler), "w" (writing - escrever) ou "a" (appending - adicionar). Nesta lição nós usaremos o modo de leitura "r". Nas próximas lições estudaremos como escrever e adicionar texto em um arquivo de texto.

Nos exemplos a seguir usaremos um arquivo de texto denominado [unitednations.txt](#). trata-se de uma lista dos Programas e Fundações das Nações Unidas e seus respectivos domínios. Você fazer o download do arquivo ou criar seu próprio arquivo para testar os exemplos com ele.

Vamos começar abrindo o arquivo unitednations.txt:

Exemplo 1: Ler uma linha de um arquivo de texto

Usamos a função  `fgets` para ler uma linha do arquivo. Este método lê até o primeiro "break" de linha no arquivo (não inclui a linha do "break").

```
<html>
<head>
<title>Ler um arquivo de texto</title>
</head>
<body>


<?php

$f = fopen("unitednations.txt", "r");

// Lê uma linha e escreve no cliente
echo fgets($f);

fclose($f);

?>
</body>
</html>
```

 [Ver o resultado do exemplo](#)

Exemplo 2: Ler todas as linhas de um arquivo de texto

```
<html>
<head>
<title>Ler um arquivo de texto</title>
</head>
<body>


<?php


$f = fopen("unitednations.txt", "r");

// Lê cada uma das linhas do arquivo
while(!feof($f)) {
    echo fgets($f) . "<br />";
}

fclose($f);

?>
</body>
</html>
```

 [Ver o resultado do exemplo](#)


No exemplo anterior percorremos as linhas com uso de um loop e usamos a função  `feof` (for end-of-file) para verificar se chegamos ao fim do arquivo. Se não ("!" - ver lição 6), a linha é escrita.

Na próxima lição veremos como escrever em um arquivo de texto.

Lição 15: Escrever em arquivos de texto

Na lição anterior aprendemos a ler um arquivo de texto com PHP. Nesta lição veremos como escrever em um arquivo de texto com PHP.

Abrir um arquivo de texto para escrever

Tal como fizemos para ler um arquivo de texto usaremos a função  `fopen` para escrever, mas agora configurada para o modo "w" (writing - escrever) ou "a" (appending - adicionar).

A diferença entre os modos *writing* e *appending* é a posição do 'cursor' para escrever - se no começo ou no fim do arquivo respectivamente.

Nos exemplos desta lição usaremos um arquivo vazio denominado textfile.txt, mas se você preferir pode usar seu próprio arquivo.

Vamos começar abrindo o arquivo de texto:

Exemplo 1: Escrever uma linha no arquivo de texto

Para escrever uma linha usamos a função  `fwrite` como mostrado a seguir:

```
<html>
<head>
<title>Escrever em um arquivo de texto</title>
</head>
<body>

<?php

// Abre o arquivo de texto
$f = fopen("textfile.txt", "w");


// Escreve no arquivo de texto
fwrite($f, "PHP is fun!");


// Fecha o arquivo de texto
fclose($f);


// Abre o arquivo e lê a linha
$f = fopen("textfile.txt", "r");
echo fgets($f);


fclose($f);

?>
</body>
</html>
```

 [Ver o resultado do exemplo](#)

 [Desafio Prático>Escrever um texto num ficheiro e mandar para outro formulário](#)

 [CONSULTA LAB PROJECTO PHP, HTML,CSS, MYSQL](#)

 [FIM DO LAB.](#)