Utilidades em PHP

Leonardo W. Sommariva

Strings

Leonardo W. Sommariva

Criando uma String

- *String* é um tipo de dado muito usado na programação de computadores. Ela é o principal tipo de dado utilizado para armazenar uma sequencia de caracteres em memória.

- Há duas formas distintas de se criar uma *string*.
  - Utilizando aspas simples:

```
$string = 'Olá Mundo';
```

  - Utilizando aspas duplas:

```
$string = "Olá Mundo";
```

Criando uma String

- A diferença entre criar uma *string* com aspas simples ou duplas são poucas.
  - Uma vantagem em criar uma *string* com aspas simples, é se caso o programador desejar inseris aspas duplas como caractere desta *string*, poderá faze-lo normalmente.

  - Quando um programador criar uma *string* com aspas duplas, para inserir outras aspas duplas terá que utilizar um caractere de escape (“Olá \”Mundo\””).
  - Uma outra vantagem em se utilizar *strings* com aspas duplas, é que se pode inserir variáveis dentro da própria string, e seu valor será computado. Por exemplo:

```
$idade = 30;
$str = “Jose tem $idade anos”;
```

Funções PHP para Strings

- strlen($str) : Determina o comprimento de uma *string*;
- strcmp($str1, $str2) : Compara duas *strings* diferenciando letras maiúsculas de minúsculas;
- strcasecmp($str1, $str2) : Compara duas *strings* sem diferenciar maiúsculas de minúsculas;
- strtolower($str) : Converte todos os caracteres de uma *string* para minúsculo;
- strtoupper($str) : Converte todos os caracteres de uma *string* para maiúsculo;
- ucfirts($str) : Converte o primeiro caractere de uma *string* para maiúsculo;
- ucwords($str) : Converte o primeiro caractere de cada palavra de uma *string* para maiúsculo;

Funções PHP para Strings

- htmlentities($str) : Converte caracteres especiais aos seus equivalentes HTML;
- explode($str, $tokens) : Divide uma *string* baseado em um delimitador predefinido.
- strpos($str, $substr) : Encontra a posição da primeira ocorrência de $subtr dentro da *string*.
- strrpos($str, $substr) : Encontra a posição da última ocorrência de $subtr dentro da *string*.
- substr($str, $start [, $length]) : Corta parte definida pelo programador de uma *string*;

Data e Hora

Manipulação de data

- O PHP fornece a função *date()* para a criação de variáveis com data e hora. Por padrão essa função retorna a data e hora atual do sistema. Isso se os parâmetros da função ferem especificados corretamente.

- A função *date()* requer que pelo menos um argumento seja passado em sua chamada, especificando o formato da data e hora.

Manipulação de data

- Para buscar a data atual no formato dd/mm/aaaa podemos fazer da seguinte maneira:

```
$data = date(“d/m/Y”);
```

- Se uma data no formato americano (aaaa-mm-dd), podemos faze-lo assim:

```
$data = date(“Y-m-d”);
```

- Ou ainda, se quisermos além da data, a horário no formato 24h:

```
$data = date(“d/m/Y H:i:s”);
```

Alguns parâmetros de formato para a função date()

Parâmetro	Descrição	Exemplo
d	Dia do mês começando do zero	01 a 31
m	Representação numérica do mês, com zero	01 a 12
Y	Representação do ano com quatro dígitos	1901 a 2038
h	Formato de 12 horas com zero	01 a 12
H	Formato de 24 horas com zero	00 a 23
i	Minutos com zero	01 a 60
s	Segundos com zero	01 a 60
A	Sigla antes e depois do meio dia com letra maiúscula.	AM e PM

Operações com date()

- A função *date()* ainda permite que seja feito operações com datas. Por exemplo, se querermos adicionar 50 dias a data atual:

```
$data_futura = strtotime(“50 days”);
$data = date(“d/m/Y”, $data_futura);
```

- Podemos fazer o mesmo para anos, meses, semanas, horas, etc:

```
$data_futura = strtotime(“2 years 3 months 2 weeks”);
$data = date(“d/m/Y”, $data_futura);
```

Operações com date()

- A função *strtodate()* converte tempo escrito por extenso para *timestamp,* permitindo assim que a operação seja realizada.

- Também é possível realizar operações matemáticas e comparações tradicionais em datas. Por exemplo:

```
echo “12/08/2015” – “07/08/2015”; //Saída: 5
```

- Ou

```
echo “12/08/2015” > “07/08/2015”; //Saída: 1 (Ou seja, true)
```

Autenticação de Usuários

Autenticação de Usuários

- Autenticar usuários é uma prática comum em aplicações web atualmente.
- Não somente por uma questão de segurança, mas também por uma questão de personalização para cada usuário.
- As variáveis de autenticação do PHP são:
  - $_SERVER[“PHP_AUTH_USER”]
  - $_SERVER[“PHP_AUTH_PW”]

Autenticação de Usuários

- Existem três metodologias de autenticação:
  - Autenticação Hard-coded;
  - Autenticação baseada em arquivo;
  - Autenticação baseada em banco de dados;

Hard-coded

- A metodologia Hard-Coded tem como objetivo que o nome de usuário e senha estejam configurados diretamente no script PHP. Por exemplo:

```
if ($_SERVER[“PHP_AUTH_USER”] != “joao”
|| $_SERVER[“PHP_AUTH_PW”]) != “joao123”)
{
    echo “Você não forneceu as credenciais corretas”;
}
```

Baseada em arquivo

- Nesta metodologia já possível que cada usuário tenha um login e senha exclusivo.
- Desta maneira é possível ter um maior controle sobre as ações de usuário.
- É comum arquivos de autenticação terem a seguinte estrutura:

```
joao:<senha_criptografada>
jose:<senha_criptografada>
maria:<senha_criptografada>
```

- Esta forma de autenticação requer um pouco mais de esforço, pois terá que abrir o arquivo e percorre-lo até encontrar a combinação de usuário e senha.

Baseada em banco de dados

- Esta é considerada a metodologia de autenticação mais poderosa e mais segura. Além de ser uma opção mais escalável, permite ser integrada com base de dados maiores.
- Uma estrutura básica de banco de dados para armazenamento de usuários e senhas seria:

id	Login	senha
1	joao	<senha_criptografada>
2	maria	<senha_criptografada>

- Esta opção irá requerer um pouco mais de esforço, pois será necessário possuir uma conexão com o banco de dados e criar os scripts de consultas.

Gerenciamento de Upload de Arquivos

- O PHP permite ao programador realizar a tarefa de levantar arquivos do cliente para o servidor.

- O PHP fornece o *array* superglobal $_FILES para auxiliar no upload de arquivos.

- Para realizar o upload de arquivos, é necessários se atentar alguns detalhes na implementação do lado cliente tanto como no lado do servidor.

Gerenciamento de Upload de Arquivos

- No lado do cliente há dois detalhes importantes que precisam ser satisfeitos:
  - Input para arquivo: No input é preciso especificar que ele será um input para arquivos através de seu *type:*
    - `<input type="file" name="arquivo" />`
  - Formulário que de suporte a dados *multipart* com a devida configuração *enctype:*
    - `<form action="" enctype="multipart/form-data">`

Gerenciamento de Upload de Arquivos

- No lado do servidor, o *array* superglobal $_FILES terá um grande papel, pois através dele é possível obter informações sobre o arquivo que está sendo levantado para o servidor. Como:
  - Nome do arquivo;
  - Tamanho do arquivo;
  - Tipo do arquivo;
  - Nome temporário do arquivo;

Gerenciamento de Upload de Arquivos

- O PHP também fornece algumas funções para a manipulação de arquivos que estão sendo levantados:
  - *move_uploaded_file($nome_arq, $destino)* : Move o arquivo levantado para o diretório desejado;
  - *is_uploaded_file($nome_arq)* : Verifica se o arquivo é um arquivo levantado;
  - *copy($nome_arq, $destino)* : Copia o arquivo levantado para o diretório desejado;

Conexão com MySQL

Leonardo W. Sommariva

Conexão com MySQL

- Existem duas alternativas para realizar a conexão com o banco de dados MySQL (E o mesmo se aplica para outros banco de dados).
  - A primeira é utilizando funções nativas do PHP;
  - A segunda é utilizando a biblioteca PDO;

Conexão com MySQL

- Utilizando as funções nativas do PHP:

```
$conecta = mysql_connect(“host”, “login”, “senha”)
                or die(mysql_error);

mysql_select_db(“nome_db”, $conecta);
$sql = “select * from dual”;
$result = mysql_query($sql);

while ($consulta = mysql_fetch_array($result))
{
        //Ações com os dados
}

mysql_close($conecta);
```

Conexão com MySQL

- Conexão com PDO (PHP Data Object):

```
try{
   $con = new PDO(“mysql:dbname=$db_name;host=$host,
                    $usuario, $senha”);

   $sql = “select * from pessoas where nome = :nome”;

   $result = $con->prepare($sql);
   $result->bindParam(“:nome”, “Maria”, PDO::PARAM_STR);

   foreach($result as $linha)
   {
            echo $linha[“nome”]; //Imprime: Maria
   }
   $con = null;
} catch(PDOException $e) {
    echo $e->getMessage();
}
```

Referências bibliográficas

- GILMORE, Jason W. **Dominando PHP e MySQL.** 2011. Terceira Edição. Alta Books Editora. Rio de Janeiro.

- The PHP Group. **Manual do PHP.** 2015. Disponível em <http://php.net/manual/pt_BR/configuration.changes.php>. Acessado em 3 ago. 2015.