Ferramentas de Desenvolvimento Web

Aula 5

Prof. Dr. Silvio Bortoleto

PHP

PHP

- PHP (Hypertext Preprocessor, ou Pré-processador de Hipertexto) é uma linguagem de programação open source mundialmente utilizada, principalmente no ambiente web. Em ambientes desktop, é utilizada a PHP–GTK. (...)

- Possui uma grande capacidade de se misturar com HTML, CSS e JavaScript, tornando mais fácil a geração de páginas web dinâmicas, atraentes e funcionais
- Assim como o JavaScript e jQuery, o PHP é **case sensitive**

- PHP é utilizada para criação de páginas web dinâmicas, bem como para a ligação com banco de dados e servidores web

PHP — Sintaxe Básica

- O código PHP poderá ficar embutido no documento HTML — o compilador e o browser irão identificar e interpretar o código PHP devido à presença dos delimitadores <? ... ?>

- Para fazer impressões na tela, poderão ser utilizadas as funções **echo** e **print**

PHP — Variáveis

- Assim como toda linguagem de programação, o PHP também possui suas variáveis. A declaração de variáveis segue o seguinte padrão:

  $nome da variável = "dado";

PHP — Data Types

- Uma variável poderá conter:
  - valores inteiros: Integer ou Long
  - pontos flutuantes: Float ou Double
  - Strings
  - Arrays

PHP – Funções

- Qualquer código PHP válido pode estar contido em funções. Como a checagem de tipos é dinâmica, o tipo de retorno não deve ser declarado, (...)

(...) sendo necessário que estejamos atentos para que a função retorne o tipo desejado

```
function nome_da_funcao([argumento1, argumento2]){
    comandos;
    [return <valor de retorno>];
}
```

- Existem 2 maneiras de fazer com que a função tenha parâmetros passados por referência:

- indicando isso na declaração da função, fazendo com que a passagem de parâmetros seja sempre assim
- na própria chamada da função
  - ✓ Nos dois casos se utiliza o modificador "&"

- Em PHP, é possível ter valores-padrão para argumentos de funções, valores que serão assumidos em caso de nada ser passado no lugar do argumento. (...)

(...) Quando algum parâmetro é declarado dessa maneira, a passagem dele na chamada da função torna-se opcional

- A forma mais simples de campos de entradas é a marcação *text*. Este campo permite a digitação de uma única palavra ou linha de texto, e possui uma largura padrão de 20 caracteres

- Opção **name**: opção obrigatória, é por meio dela que os scripts vão se basear para serem executados
- Opção **value**: valor pré-definido do elemento, aparecerá quando a página for carregada

- Opção **size**: valor opcional, especifica o tamanho do campo ao ser exibido no form
- Opção **maxlength**: o tamanho máximo do texto contido no elemento, em caracteres

- Tipo de campo semelhante ao anterior, com a diferença de que neste os dados digitados são substituídos por ****. (...)

(...) É importante lembrar que nenhuma criptografia é utilizada, apenas uma máscara visual, fazendo com que o que está sendo digitado não apareça na tela

PHP — Formulários — GET

- Por meio da definição da instrução HTML *method="get"* na tag <form>, (...)

(...) todas as informações transmitidas pelo form para o servidor são ativadas ao final da URL ativa — a maioria dos documentos HTML são recuperados a partir da requisição de uma única URL ao servidor

- Outro fato importante é informarmos ao <form> para onde esses dados estão indo, por meio da instrução *action="..."* colocada na tag <form>, onde a URL apontará para um script PHP que receberá e decodificará os dados enviados

PHP — Formulários — POST

- Por meio da definição da instrução HTML *method="post"* na tag <form>, todas as informações transmitidas pelo form para o servidor são imediatamente enviadas após a URL, (...)

(...) ou seja, quando o servidor recebe uma ativação de um formulário que está utilizando o método POST, ele reconhece que precisa continuar "ouvindo" para obter a informação

PHP — Banco de Dados

- Com a breve revisão sobre <form>, vamos ver agora a relação com banco de dados — conexão com o banco, inserção, consulta, alterações e exclusão de registros

- Utilizando o servidor MySQL, o trabalho com a linguagem SQL através de PHP se torna fácil.
- O comando responsável por efetuar a conexão com o servidor MySQL de banco de dados é:

```
mysql_connect(string host [:porta], string login, string senha);
```

- Onde:
  - **string host** indica o endereço do servidor MySQL
  - **[:porta]** indica uma porta opcional na qual o servidor está operando

- **string login** indica o nome de usuário do qual será realizada a conexão
- **string senha** indica a senha do usuário que está se conectando ao servidor

- O PHP nos permite armazenar valores em variáveis, podendo assim armazenar toda a string de conexão dentro de uma:

```
$nome = mysql_connect(string host [:porta], string login, string senha);
```

- Assim, se a conexão for bem-sucedida, ela ficará armazenada dentro da variável, e poderemos solicitá-la no momento em que desejarmos:

```php
<?php
    $conexao = mysql_connect("localhost", "root", "")
    or die("Não foi possível conectar");
?>
```

- Uma vez conectados ao servidor MySQL, precisamos selecionar um banco de dados no qual vamos trabalhar, por meio do comando:

```
mysql_select_db(string nome_do_banco, string conexao);
```

- Se tudo correr bem, o valor do retorno será 1, caso ocorra algum erro será 0. O nome da base de dados para seleção deverá ser sempre o 1º parâmetro fornecido, seguido do identificador da conexão

- Após a execução do comando anterior, qualquer consulta executada para aquela conexão utilizará a base de dados selecionada

- Ao executar uma query SQL SELECT por meio do comando mysql_query, o identificador do resultado deve ser armazenado em uma variável que pode ser tratada de diversas formas

- A utilização do comando mysql_result, ou o comando mysql_fetch_row, respectivamente, são maneiras interessantes para fazer isso

```
mysql_result(variável de resultado, linha, campo);

mysql_fetch_row(variável de resultado);
```

PHP — Orientação a Objetos

- PHP não foi criada para ser uma linguagem orientada a objetos, mas a partir da sua 3ª versão passou a suportar tal função

- Antes disso, os programadores utilizavam a programação estruturada e/ou a orientação a funções. Basicamente, o método organizava as funções mais utilizadas em arquivos específicos (da mesma maneira que é feito com JavaScript). (...)

(...) Funções como *insert, update* e *delete* eram colocadas em arquivos onde estavam contidas as funções direcionadas para banco de dados. Depois, esse arquivo era incluído no local onde seriam utilizadas as funções

- O primeiro conceito básico da orientação a objetos é a classe, onde são feitas abstrações do software de objetos similares — um template de onde os objetos serão criados

- A sintaxe básica para a criação de classes em PHP é:

```
<?php
        class nome_da_classe{ }
?>
```

- Vamos definir uma classe, *Usuario*, determinando alguns atributos:

```php
<?php
 class Usuario{
   public $nome;
   public $cpf;
   public $endereco;

   //construtor da classe
   function Usuario(){
     $this->preparaUsuario();
   }

   function preparaUsuario(){
     $this->nome = "João da Silva";
     $this->cpf = "99999999999";
     $this->endereco = "Rua das Goiabas";
   }
 }
?>
```

- *Construtor da classe* será o método executado quando a classe *Usuario* for chamada — nesse método são colocados os passos para a inicialização da classe

```php
<?php
 require_once 'usuario.php';

 $usuario = new Usuario();

 echo $usuario->nome;
 echo "<br>";
 echo $usuario->cpf;
 echo "<br>";
 echo $usuario->endereco;
?>
```

- Com esse código, estamos testando a classe, imprimindo os dados armazenados

- Assim como em linguagens orientadas a objetos, como Java, precisamos realizar o *import* do arquivo que contém a classe para instanciar a classe desejada. Após a instanciação, é só acessar os métodos

- A visibilidade, outro conceito fundamental em orientação a objetos. A visibilidade de um método ou atributo pode ser definida fixando as palavras-chave *public, private* ou *protected*

- Por padrão, se não houver a prefixação, o PHP interpretará como *public*

```
<?php
 class Usuario{
   public $nome;
   protected $cpf;
   private $endereco;

   //construtor da classe
   function Usuario(){
     $this->preparaUsuario();
   }

   private function preparaUsuario(){
     $this->nome = "João da Silva";
     $this->cpf = "99999999999";
     $this->endereco = "Rua das Goiabas";
   }

   public function getCpf (){
     return $this->cpf;
   }

   public function getNome(){
     return $this->nome;
   }

   function getEndereco(){
     return $this->endereco;
   }
 }
?>
```

- Para acessarmos os *get*'s da classe Usuario, utilizamos o código:

```
require_once 'usuario.php';

class AcessaUsuario{
  function imprimeUsuario(){
    $usuario = new Usuario;
    echo $usuario->nome;
    echo $usuario->getCpf();
    echo $usuario->getEndereco();
  }
}
```

PHP — Configurando

- Para seus arquivos .php serem executados em ambiente local, você deve possuir instalado em seu computador um servidor local, pois os browsers não conseguem interpretar os scripts em PHP assim como fazem com HTML

- Costumeiramente, utilizamos o WampServer, que automaticamente instala tudo o que você precisa para começar a desenvolver aplicações web, sendo muito intuitivo em seu uso

- O prefixo WAMP deriva de Windows, Apache (servidor), MySQL (banco de dados) e PHP-Perl-Python, pois o programa instala e oferece suporte a todas essas ferramentas, e ainda integra itens como o PHPMyAdmin

- Após instalar o WampServer, será criado um diretório *"wamp"*, onde estarão localizadas as preferências e configurações do software. Dentro do diretório *wamp*, haverá uma pasta chamada *"www"*, que é onde você irá colocar seus projetos

- A partir daí, você já poderá rodar seus projetos em .php em ambiente local!

PHP — Exercícios

- <..\..\exercicios\php.html>.
- <http://www.devmedia.com.br/introducao-a-orientacao-a-objetos-em-php/26762>.
- <http://www.techtudo.com.br/tudo-sobre/wampserver.html>.