LINGUAGEM DE PROGRAMAÇÃO IV - INTERNET

ILP540

ILP540

PHP

Uma Linguagem de Programação para a camada CGI da arquitetura MVC.

João Maurício Hypólito
prof.joao.hypolito@gmail.com

Programa

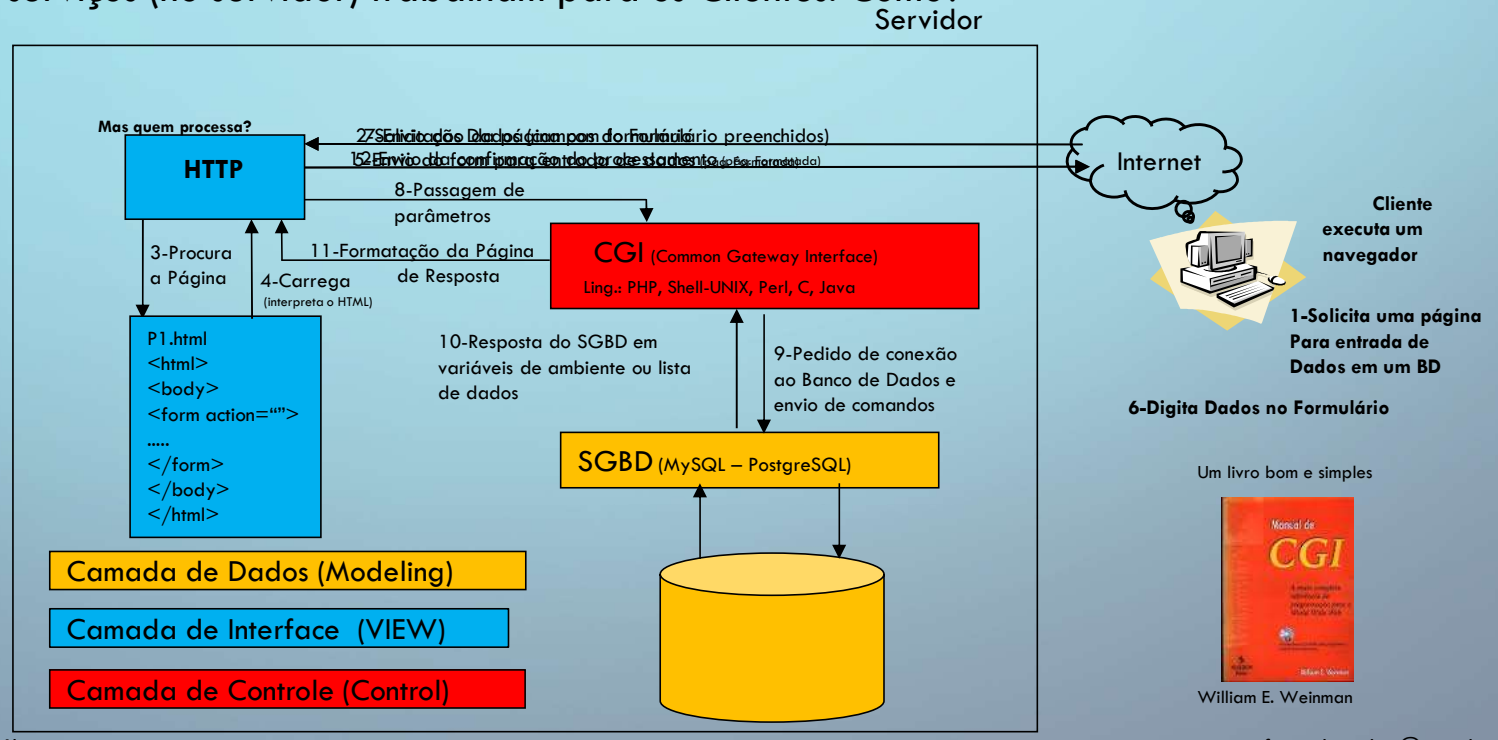
- Histórico
- A Camada CGI do MVC?
- Características Gerais
- Escrevendo um PA... Onde? Como?
- As fontes da linguagem PHP
- O primeiro PA... executando...
- Comentários no PA
- Variáveis e vetores no PHP
- O PA recebendo valores para processamento
- Conectando um PA a um Banco de Dados
- Executando um comando SQL no banco
- Tratando a resposta de um comando SQL
  - Algumas funções de ambiente do PHP.
- Comandos de Repetição
- Tarefa

PHP – Linguagem de Programação da Camada CGI

- 1994 Rasmus Lerdorf escreve um interpretador de comandos para montagem de homepages personalizadas (Personal HomePage).
- 1997 - 50.000 sites
- Outubro de 1998 - 100.000 sites
- 1999 -1.000.000 sites
- Zeev Suraski e Andi Gutmans, dois programadores israelenses que desenvolveram os analisadores de sintaxe PHP3 e PHP4.
- A empresa ZEND passa a coordenar o desenvolvimento da Linguagem PHP como linguagem de programação da Camada CGI na arquitetura MVC.
- Acabou o histórico.
- Mas... O que é Camada CGI na Arquitetura MVC?

PHP – Linguagem de Programação da Camada CGI

- Na década de 1970 desenvolve-se o conceito de redes de computadores
  - InterNet: BOM! os serviços podem ser especializados
  - Servidores de páginas, Servidores de Bancos de Dados, etc...
    - Bons no que fazem, mas só fazem BEM o que sabem fazer.
  - Os serviços (no servidor) trabalham para os Clientes. Como?

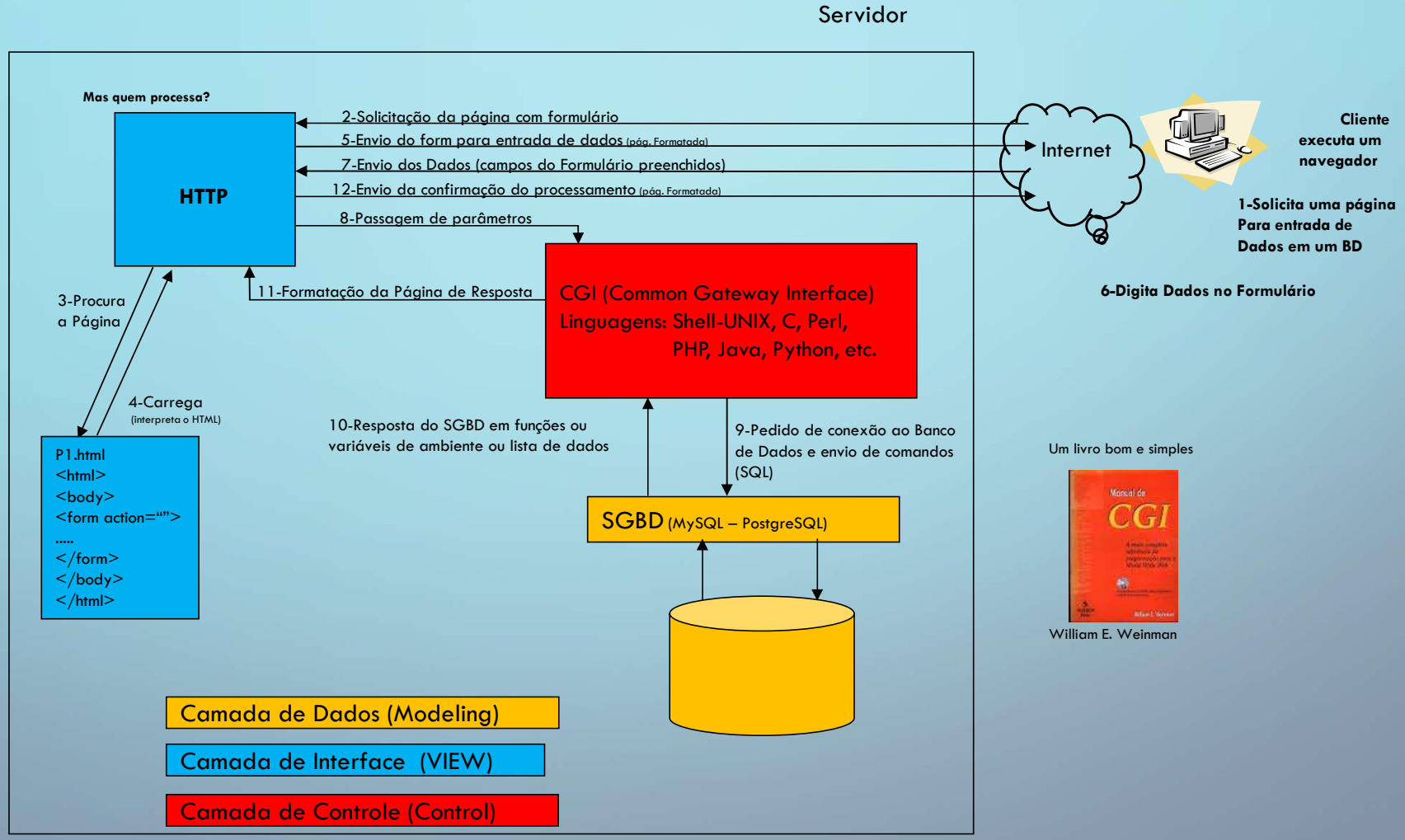


Um livro bom e simples

William E. Weinman

prof.joao.hypolito@gmail.com

PHP – Linguagem de Programação da Camada CGI



Um livro bom e simples

William E. Weinman

João Maurício Hypólito

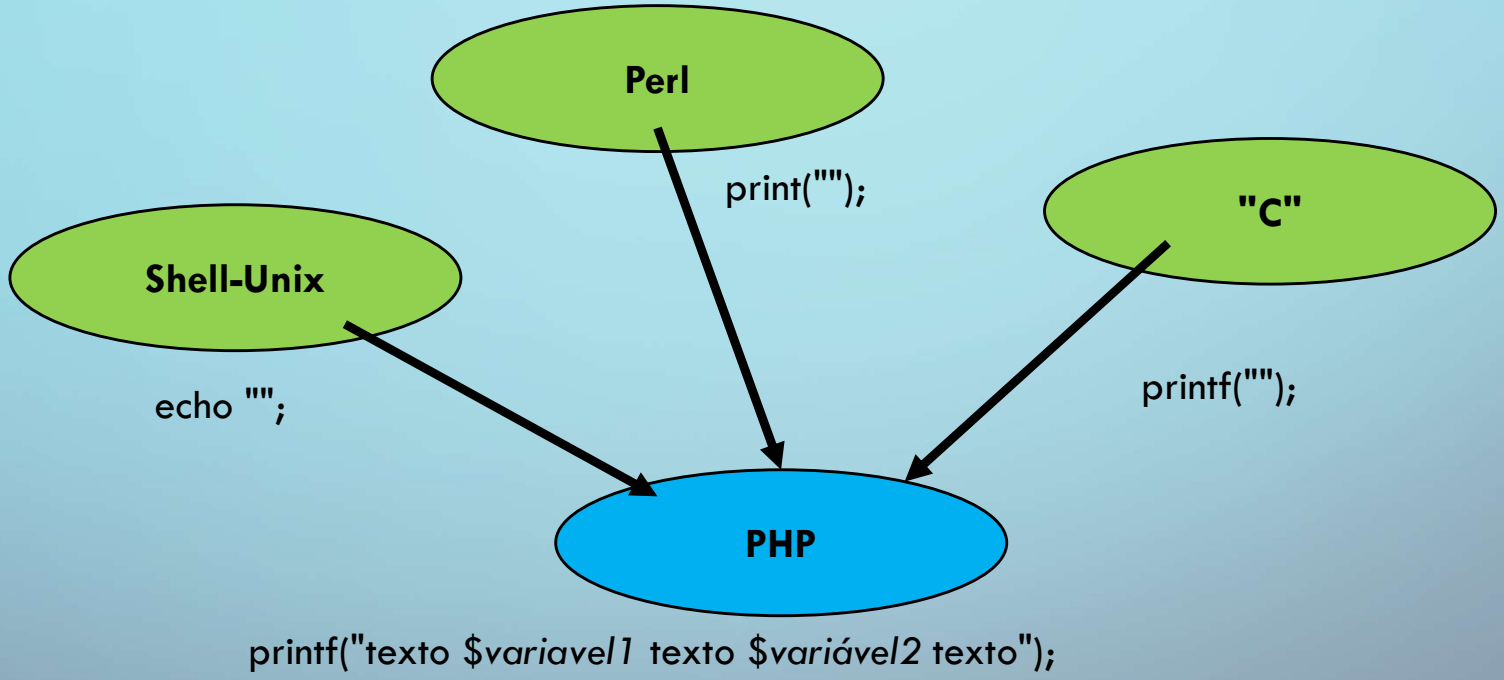
prof.joao.hypolito@gmail.com

PHP – Linguagem de Programação da Camada CGI

- Suporte a um grande número de bancos de dados e acessa uma diversidade de serviços (servidores HTTP, SNMP, POP3, etc.). Sendo, portanto, uma linguagem adequada ao desenvolvimento de sistemas na arquitetura MVC.
- É uma linguagem que permite criar 'sites dinâmicos' (páginas resultantes de acessos e processamento de dados – não ficam gravadas nos servidores).
- As páginas são “escritas” por scripts CGI, que têm a LÓGICA de uma regra de negócio (o usuário não acessa o fonte do script preservando a propriedade intelectual do autor).
- As páginas são “montadas” on-fly, ou seja em tempo de execução. As páginas não são armazenadas e sim escritas durante o processamento de um programa PHP.
- Do mesmo modo pode-se 'capturar' o resultado do processamento em outros ambientes de programação (um jogo, por exemplo). Sendo assim, um modo de um jogo atuar sobre bancos de dados.

PHP – Linguagem de Programação da Camada CGI

- Antes de sair escrevendo php...
- PHP tem três linguagens das quais 'herdou' muitos comandos:



PHP – Linguagem de Programação da Camada CGI

- Um pouco da arquitetura do ambiente
- PHP é um interpretador de comandos, isso significa que não gera binários para execução no ambiente do sistema operacional. O interpretador PHP executa as três fases da compilação (análise sintática, léxica e semântica) para cada comando escrito. Isso é herança do Shell-Unix.
- O interpretador dispõe de funções de ambiente dividas em dois categorias:
  - internas – construídas para processamentos de operações básicas, por exemplo exibir uma string: echo, print() ou printf() e
  - Externas – 'acopladas' ao ambiente por bibliotecas externas, por exemplo para tratar dados em um SGBD. Por exemplo ao trabalhar com MySQL o PHP tem funções começando com mysql_ ou mysqli_, para o SGBD PostgreSQL as funções começam com pg_.

PHP – Linguagem de Programação da Camada CGI

- Um pouco da arquitetura do ambiente
- O PHP implementa o mesmo conceito de função do usuário: segmento de código escrito pelo usuário (programador ou analista) que deve ser identificado para ser usado por um Programa Aplicativo (ou mais de um PA).
- O PHP controle de execução de PA invocando o controle de processos no Sistema Operacional e para cada processo iniciado, cria páginas de execução que mantém no setor de memória determinado pelo SO. Todo o gerenciamento de paginação de execução do PA é feito pelo interpretador PHP.

PHP – Linguagem de Programação da Camada CGI

- Um pouco da arquitetura do ambiente
- O acesso de um PA à um arquivo externo (PA ou arquivo qualquer) é feito por duas funções de ambiente: INCLUDE(); e REQUIRE();
  ambas recebem como parâmetro o caminho/nome do arquivo ser acessado.
  - O caminho necessariamente DEVE estar 'dentro' do sistema de arquivos do sistema operacional onde o interpretador está instalado. Ou seja, não dá para montar uma REDE de PA entre servidores diferentes.
- Ainda as duas funções podem ser complementadas com '_once()' para indicar que o arquivo deve ser lido uma vez e se for alterado nas vezes subsequentes.
- Com o include, se o PA 'secundário' tiver algum erro de execução o PA 'primário' não é interrompido, continuando sua execução.
- Com o require, se o PA 'secundário' tiver algum erro de execução o PA 'primário' INTERROMPE sua execução.

PHP – Linguagem de Programação da Camada CGI

- Um pouco da arquitetura do ambiente
- Ainda um pouco mais sobre funções de usuário...
- São declaradas com o comando:

  function <NomeDaFunção> ([parâmetros])

  {

  }
- E são 'executadas' simplesmente assim:

  <NomeDaFunção>();
- Notas:
  - As funções podem não dar ***retorno***. Existem linguagens onde o retorno é obrigatório.

PHP – Linguagem de Programação da Camada CGI

- Um pouco da arquitetura do ambiente
- Exemplificando...

```
<?php
function emite()
{
  printf("Só emite um texto");
}
printf("<html><head><meta charset='UTF-8'></head><body>Texto<br>");
emite();
printf("<br>Depois</body></html>");
?>
```

- Veremos no navegador:

Texto
Só emite um texto
Depois

PHP – Linguagem de Programação da Camada CGI

- Mas como se escreve um programa aplicativo (PA) em PHP?
  - Usa-se um editor de texto de programação (limpo) – Bloco de Notas, NoteTab, NotePad++ - ou um ambiente de programação (framework).
  - O arquivo com PA PHP DEVE ter a terminação .php para o SO e o Navegador 'saberem' que o interpretador PHP é o 'responsável' por controlar a execução do PA.
  - O PA PHP tem marcadores para informar ao interpretador onde 'começa' e onde 'termina' o segmento de código. O que estiver antes e depois das marcas é tratado pelo navegador.
    - As Marcas são
      INÍCIO: **<?php**
      e
      TÉRMINO: **?>**

PHP – Linguagem de Programação da Camada CGI

- Então, agora, podemos escrever um PA-PHP...

```
<?php
  $CorDaCasa="Marrom";
  $Tamanho="220m2";
  printf("Olá, mundo!\n");
  printf("Moro em uma casa $CorDaCasa e quem tem $Tamanho!");
?>
```

- O comando printf é IGUAL ao da linguagem C, mas também podemos escrever **print** (como em Perl) ou **echo** (como em shell-script).
- Variáveis tem seu NOME sempre começando com '$' (herança do Shell-Unix).
- NÃO é preciso declarar o TIPO DE DADO de variáveis.
- Repare que o printf(); é 'semelhante' ao C... Mas em C não dá para escrever o nome da variável 'dentro' da string de exibição. Em PHP pode...

PHP – Linguagem de Programação da Camada CGI

- Então, agora, copie os comandos em um arquivo e grave em um subdiretório com nome **ilp540**. De o nome do arquivo de **'pa1.php'**
- Para 'executar' seu PA, o Servidor de Página DEVE estar ativado. Por isso:
  - Vá para o diretório c:\xampp e execute o programa xampp-control.exe e ative os servidor Apache. Ative também seu Servidor de Banco de Dados. Clique nos botões "START" e ative estes dois serviços.
- Agora abra seu navegador e crie uma nova aba. No campo de url escreva: http://localhost/**ilp540/pa1.php**/
- Descreva o que você consegue visualizar na tela do seu navegador.
- Coloque o cursor do mouse sobre a área de visão da aba do navegador e clique o botão secundário. Escolha o item "ver código fonte" (ou algo semelhante). Clique na nova ABA aberta no navegador. Você vê o código da página como está sendo recebida pelo navegador (enviada pelo Apache). Veja a diferença no salto das linhas no texto recebido e no texto exibido. Por que será que isso acontece?

PHP – Linguagem de Programação da Camada CGI

- Bem, vamos fazer mais alguns comentários sobre os PA-PHP...
- Colocando comentários nos programas
  - Igual ao modo de comentários da linguagem C
  - Igual ao modo de comentários de Shell-Script
- Exemplo:

```
<?php
  printf("teste\n"); # isto é um teste
  printf("teste\n"); // este teste é similar ao anterior
  printf("teste\n");
  /* Isto é um comentário em bloco (ou em mais de uma
     linha. Pode ser usado para documentar programas
     de modo mais completo. */
  printf("teste\n");
  #---------------------------
  # Se bem que eu prefira
  # assim: dá para formar caixas de texto mais regulares.
  #---------------------------
?>
```

- Grave em um pa2.php e execute (ou rode) seu PA. Comente.

PHP – Linguagem de Programação da Camada CGI

- Variáveis
- Definindo...
  - É um 'pedaço' de memória RAM que pode ter seu 'valor' manipulado por comandos da linguagem.
- Todo nome de variável (e também vetor) deve começar com '$'
- As variáveis em PHP são muito flexíveis:
  - Não existe tipo de dado fixo. O tipo de dado é determinado no comando de atribuição de valor na variável.
  - $a="casa"; # $a é uma string
  - $a=10; # $a agora é um número
  - Isso se chama **coercividade**.
- Podemos exibir o conteúdo de variável usando o printf:

```
<?php
$a="casa";
printf("O Valor de ".'$a'." é: $a\n");
?>
```

  - Note: Dentro do printf usamos o operador '.' Este operador é o **concatenador** de strings. A string $a aparece entre aspas simples (' ') para ser mostrada sem o processamento do valor da variável.
  - Note: A terceira string do printf (que é " é: $a
\n" quando encontra o $a processa a exibição do conteúdo da variável.

PHP – Linguagem de Programação da Camada CGI

- Variáveis (lidando com vetor)
  - O conceito de vetor em PHP é o mesmo que C.
  - Diferença 1: As posições do vetor podem ser 'nomeadas'
  - Diferença 2: existe em PHP um comando para mostrar os vetores.

```
<?php
# o vetor é criado com duas posições
$vet1 = array('comida' => "pizza", 'fome' => true);
printf("$vet1[comida]<br>\n"); # vemos bar
printf("$vet1[fome]<br>\n");   # vemos 1 (TRUE)
printf("<pre>\n");
$vet2 = array("pizza", 2, 3, 4, 5); # vetor com 5 posições.
# Este comando exibe as posições e valores do vetor
print_r($vet1);
print_r($vet2);
printf("</pre>\n");
?>
```

- O mais completo guia de referencia do PHP é o site:
  - http://www.php.net ou https://www.php.net/manual/pt_BR/index.php

PHP – Linguagem de Programação da Camada CGI

- Ah! Está bem... Mas como acessar um PA passando um parâmetro para executar, por exemplo, um SE...ENTÃO...SENÃO?
- Um PA-PHP 'recebe' um parâmetro em uma "String de Transmissão de Dados – STD. A STD pode transmitir vários parâmetros e valores. Cada parâmetro recebe um valor em uma expressão na forma: par_n=$valor_n$. Esta expressão se designa Atribuição.
- A STD é, então, formada assim:
  " par_1=vlr_1&par_2=vlr_2&...&par_{n-1}=vlr_{n-1}&par_n=vlr_n"
- Mas como tratar isso no PHP?
- Em PHP existem 2 Vetores que recebem os vlr_1, vlr_2, ..., vlr_{n-1}, vlr_n, nas posições ' par_1', 'par_2', ..., 'par_{n-1}', 'par_n'. A partir da versão 5 do php existem: $_POST[], $_GET[] e $_REQUEST[].
- Já vimos isso na aula de Formulários HTML.

PHP – Linguagem de Programação da Camada CGI

- Já vimos isso na aula de Formulários HTML.
- Vamos então fazer a (RE)construção de um PA que monta um formulário (exform.php) e de outro PA que recebe e exibe os dados digitados nos campos do form feito no exform.php (vamos adotar o nome exleform.php)
- Depois apresentamos a estrutura de um programa recursivo (recursivo-estrut.php) e desenvolvemos um que monta um formulário e le&exibe os valores digitados nos campos do form (recur1.php).
- Os SO tem limites de arquivos abertos em um mesmo diretório. Criar muitos arquivos pequenos e gravá-los todos no mesmo diretório pode gerar uma sobrecarga no servidor quando executados por muitos usuários.
- Este é um dos motivos para implementar a recursividade.

PHP – Linguagem de Programação da Camada CGI

- Recursividade: Técnica de programação que permite desenvolver duas ou mais funcionalidades distintas em um mesmo arquivo de programa.
- Em PHP usam-se funções de ambiente e comandos de desvio condicionais para implementar a recursividade.
- Funções: ISSET() – recebe uma variável ou vetor e retorna 'TRUE' se a variável | vetor contém valor (no *set* de variáveis do PA), senão retorna 'FALSE'.
- Comandos de desvio condicional:
  - *if* () {...} *else* {...} ou o operador ternário () ? : ; e
  - *switch* () {case():{...} case():{...} case():{...}...}
- Analisaremos uma estrutura de um PA recursivo.

PHP – Linguagem de Programação da Camada CGI

- Analisaremos uma estrutura de um PA recursivo.

```
<?php
# determinando o bloco que deve ser executado.
$bloco=(ISSET($_REQUEST['bloco'])) ? $_REQUEST['bloco'] : 1;
# emitindo as TAGs HTML que iniciam a página
printf("<html>\n<body>\n");
# Comando de desvio de Fluxo Principal do PA. Dois blocos principais.
switch (TRUE)
{
  case ( $bloco==1 ):
  {
    printf("<form action='taxona.php' method='POST'>
          <input type='hidden' name='bloco' value='2'>
          <input type='radio' name='fruta' value='L'>Laranja ou
          <input type='radio' name='fruta' value='M'>Ma&ccedil;&atilde;
          <input type='submit' value='vai'>
          <form>\n");
    break;
  }
  case ( $bloco==2 ):
  {
    $fruta = $_POST['fruta']=='L' ? "Laranja" : "Ma&ccedil;&atilde;";
    $tipo = $fruta=="Laranja" ? "pera" : "fuji";
    $var = "Esta &eacute; uma ".$fruta." ".$tipo;
    printf("$var\n");
    break;
  }
} # Fim do Switch...case que divide os blocos principais do PA
# emitindo as TAGs HTML que terminam a página
printf("</body>\n</html>\n");
?>
```

PHP – Linguagem de Programação da Camada CGI

- Fazendo a conexão com uma base de dados de um servidor.
- Usa-se uma função do ambiente do PHP 'para' o SGBD que se crer acessar. Para o MariaDB e o MySQL usamos a função:
  mysqli_connect();. Esta função recebe 4 parâmetros:
  nome do servidor, nome do usuário, senha de acesso e nome da base a ser acessada.
  Isso vale para versão 5+.
- A função *mysqli_connect();* retorna um número de conexão (se a conexão der certo).
  Assim:
  $CON=mysqli_connect('localhost','root','','ilp540m20212');
  recebe um número de uma conexão SE tudo estiver certo com os parâmetros.
- A variável $CON passa a ser um parâmetro necessário em algumas outras funções.
- Certo... Agora como 'executar' o comando SQL na base que foi conectada?

PHP – Linguagem de Programação da Camada CGI

- Executando um comando na base de dados (no servidor conectado).
- Pode-se escrever um comando em uma variável:
  $cmdsql="select * from funcionarios";
- Para executar um comando SQL usa-se uma função de ambiente do PHP
  (mas quantas existem? Devem ser umas 60 funções. Nome e parâmetros de todas?
  http://www.php.net)
  mysqli_query(); - recebe 2 parâmetros e retorna um 'treco'. Este treco é um vetor com três partes: os nomes das tabelas 'operadas', os campos que foram operados e os endereços das linhas que foram operadas. Então para o comando: "select * from funcionarios" o retorno é:
  funcionarios | todos os campos | todos endereços das linhas (números).
- E se der alguma coisa errada no comando? A função retorna um 'status' de erro.
- Pode-se escrever:

```
$CON=mysqli_connect('localhost','root','','ilp540m20212');
$cmdsql="select * from funcionarios";
$execcmd=mysqli_query($CON,$cmdsql);
```

PHP – Linguagem de Programação da Camada CGI

- E o processamento destes comandos não EXIBE NADA! Mas... Por que?
- O 'vetor complexo' fica disponível para outras funções de ambiente (para contagem de registro, para exibição e outras.
- "Pegando" a primeira linha colocando num vetor: $reg=*mysqli_fetch_array($execcmd);* Daí pode-se usar o print_r($reg); para ver a linha.
- Assim:

```
<?php
$CON=mysqli_connect('localhost','root','','ilp540m20212');
$cmdsql="select * from funcionarios";
$execcmd= mysqli_query($CON,$cmdsql);
$reg=mysqli_fetch_array($execcmd);
printf("<pre>");
print_r($reg);
printf("</pre>");
?>
```

- Vai exibir a primeira linha da tabela funcionarios.

PHP – Linguagem de Programação da Camada CGI

- Mas... "Preciso ver todas as linhas de uma tabela! Como faço?".
- Precisamos 'repetir a montagem' do vetor (usando a função *mysqli_fetch_array()*) até que a lista de endereços (montada quando se executa o comando SQL com a função *mysqli_query()*) se esvazie.
- Cada vez que se executa o *mysqli_fetch_array()* o interpretador retorna um TRUE se o comando deu certo. Então, pode-se montar um comando de repetição montando o vetor com os dados de cada linha da tabela assim:

```
<?php
$CON=mysqli_connect('localhost','root','','ilp540m20212');
$cmdsql="select * from funcionarios";
$execcmd= mysqli_query($CON,$cmdsql);
while ( $reg=mysqli_fetch_array($execcmd) )
{
     printf("<pre>");
     print_r($reg);
     printf("</pre>");
}
?>
```

PHP – Linguagem de Programação da Camada CGI

- Podemos escolher 'mostrar' os dados de cada campo da tabela usando o *printf()* dentro do laço de repetição e melhorar o aspecto dos dados na tela do navegador.

```
<?php
$CON=mysqli_connect('localhost','root','','ilp540m20212');
$cmdsql="select * from funcionarios";
$execcmd= mysqli_query($CON,$cmdsql);
while ( $reg=mysqli_fetch_array($execcmd) )
{
    printf("$reg[cpfuncionario] -
            $reg[txnomefunc] -
            $reg[txsobrenomefunc] -
            $reg[nutelefone] -
            $reg[dtcontratacao] -
            $reg[dtcadfuncionario]<br>");
}
?>
```

- Nota: neste exemplo tiramos os comandos <pre> ... </pre> e usamos o printf() com
 que faz o salto de linha. Também não emiti os valores de todos os campos de funcionarios, por um efeito de entendimento do comando *while(){}*.

PHP – Linguagem de Programação da Camada CGI

- Então, agora, podemos escrever um PA-PHP para cada operação que queremos fazer na tabela?
  - Sim! Ou podemos ler na STD uma atribuição indicando qual operação queremos fazer e depois 'executar' esta operação.
- Mas com quais comandos podemos desviar o fluxo de execução de comandos?
- Pode-se usar:
  if () {} elseif {} else {}
  switch () { case(){}...case(){} default {} }
- Para laços de repetição o PHP tem:
  while () { };
  do { }while();
  for (ini;cond;passo){} e
  foreach(){}
- Estes comandos são muito semelhantes aos da linguagem "C", exceto o foreach().
- Aconselhamos a leitura de tutorial dos comandos em https://www.php.net/ no campo de pesquisa inicie com foreach e depois escolha os outros comandos.

PHP – Linguagem de Programação da Camada CGI

- if () {} elseif {} else {}

```
<?php
# execute este PA passando o parâmetro num como um número entre 1 e 20.
# http://localhost/if.php?num15
if ( $_GET['num'] < 5 )
{
    printf("Num é menor que 5");
}
elseif ( $_GET['num'] < 10 )
{
    printf("5 <= Num < 10");
}
elseif ( $_GET['num'] < 15 )
{
    printf("10 <= Num < 15");
}
else
{
    printf("15 <= Num < 20");
}
?>
```

PHP – Linguagem de Programação da Camada CGI

- switch () { case(){}...case(){} default {} }

```
<?php
# execute este PA passando o parâmetro num como um número entre 1 e 20.
# http://localhost/switch.php?num=15
switch (TRUE)
{
     case ( $_GET['num'] < 5 ):
     {
          printf("Num é menor que 5");
          break;
     }
     case ( $_GET['num'] < 10 ):
     {
          printf("5 <= Num < 10");
          break;
     }
     case ( $_GET['num'] < 15 ):
     {
          printf("10 <= Num < 15");
          break;
     }
     default:
     {
          printf("15 <= Num < 20");
     }
}
?>
```

PHP – Linguagem de Programação da Camada CGI

- while () { };

```
<?php
# execute este PA passando o parâmetro num como um número entre 1 e 20.
# http://localhost/while.php?num=15
$contador=10;
while ( $contador <= $_GET['num'] )
{
    printf("Contador = $contador e limite é $_GET[num]<br>");
    $contador=$contador+1;
}
?>
```

- do { }while();

```
<?php
# execute este PA passando o parâmetro num como um número entre 1 e 20.
# http://localhost/dowhile.php?num=15
$contador=10;
do
{
    printf("Contador = $contador e limite é $_GET[num]<br>");
    $contador=$contador+1;
} while ( $contador <= $_GET['num'] )
?>
```

PHP – Linguagem de Programação da Camada CGI

- for (ini;cond;passo){}

```
<?php
# execute este PA passando o parâmetro num como um número entre 1 e 20.
# http://localhost/for.php?num=15
for($contador=1;$contador <= $_GET['num'];$contador=$contador+1)
{
    printf("Contador = $contador e limite é $_GET[num]<br>");
}
?>
```

- foreach(){}

```
<?php
$arr = array(1, 2, 3, 4);
# Verificando os valores do vetor $arr
printf("Antes: <pre>");print_r($arr);printf("</pre>");
foreach ($arr as &$value) {
    $value = $value * 2;
}
# A passagem dos valores do vetor como ponteiro de referência para &$value
# faz com que as alterações em $value acontecãm nos valores correspondentes
# do vetor em cada ciclo do foreach()
# Verificando os valores do vetor $arr
printf("Depois:<pre>");print_r($arr);printf("</pre>");
?>
```

PHP – Linguagem de Programação da Camada CGI

- Agora à tarefa:
- Você deve escrever um PA-PHP em um diretório do seu computador designado com SEU NOME (como escrito no SiGA).
- Neste diretório crie um arquivo .php e escreva um programa que:
  - Recebe na chamada o valor de um parâmetro com nome *par1*;
  - Faz uma conexão com a base de dados de estudo do SQL;
  - Faz uma pesquisa em uma tabela de sua escolha (exceto a tabela *funcionarios*), comparando o valor da chave primária da tabela com *par1*;
  - Exibe o registro recuperado sem usar nenhum laço de repetição;
  - Executa uma segunda leitura em outra tabela com o mesmo valor de *par1* comparado com a chave primária da tabela; e
  - Exibe o registro recuperado usando um comando de laço de repetição (mesmo que seja somente para um registro.
- Teste seu PA sob localhost com diferentes valores de par1. Se em alguma situação uma linha não recuperada, então seu PA deve exibir a mensagem "Linha não existe"
- Grave seu arquivo, gere um compactado do diretório e envie na tarefa do Teams.

LINGUAGEM DE PROGRAMAÇÃO IV - INTERNET

ILP540