

Aula 9

Professor:

Inês Dutra

Consultas a bancos de dados e
entrada de dados do usuário

PHP: Consultas a bancos de dados

- Objetivos deste módulo:
 - ▶ Utilização de funções MySQL para consultar dados em PHP
 - ▶ Manipulação de erros em MySQL
 - ▶ Formatação de dados usando `<table>`
 - ▶ Utilização de uma abordagem de consulta a bancos de dados baseada em 5 passos
 - Abrir conexão ao banco de dados
 - Consulta
 - Recuperação de linhas de tabelas
 - Processamento de atributos
 - Fechar conexão ao banco de dados

PHP: Consultas a bancos de dados

- Abrindo conexão com o banco de dados:
 - ▶ `$conexão = mysql_connect(hostname, username, password)`
- Fechando a conexão:
 - ▶ `mysql_close($conexão)`
- Consulta ao banco de dados:
 - ▶ `mysql_query(<query>, $conexão)`
- Recuperação de linhas de tabelas:
 - ▶ `mysql_fetch_row`
- Processamento de atributos: Utilização de
`mysql_num_fields`

PHP: Exemplo

```

<head>
<title> Exemplo de acesso a BDs </title>
</head>
<body><pre>
<?php

    echo "Neste exemplo, quando você clicou no botão <abrir navegador>, uma tabela
          chamada cliente foi criada, com atributos: id, nome e endereço. Para repetir
          este exemplo, fora do contexto da aula, é necessário criar esta tabela manualmente
          no servidor mysql.\n\n";

    // (1) abrir conexão com mysql no servidor remoto ao PHP
    $con = mysql_connect($_SERVER["REMOTE_ADDR"], "aluno", "aluno");
    // selecionar banco de dados
    mysql_select_db("prog2", $con);
    // (2) consulta ao bd prog2 sobre tabela cliente
    $res = mysql_query("SELECT * FROM cliente", $con);
    // (3) recupera linhas da tabela
    while ($row = mysql_fetch_row($res))
    {
        // (4) imprime valores de atributos
        for ($i=0; $i < mysql_num_fields($res); $i++)
            echo $row[$i]." ";
        echo "\n";
    }
    // (5) fechar conexão
    mysql_close($con);
?>
</pre></body>
</html>

```

[Abrir editor](#)
[Abrir navegador](#)

PHP: acessando bancos de dados

- `mysql_connect`: requer três parâmetros, hostname, username e password. Hostname pode também especificar a porta utilizada por mysql, que normalmente é a 3306
- Se a conexão for bem sucedida, retorna o descritor da conexão, que deve ser usado em outras chamadas de funções mysql subsequentes
- Se a conexão falhar, exibe mensagem de erro.
Para suprimir mensagem, utilizar `@mysql_connect`
- Username e password, que, no nosso caso, são, respectivamente: `aluno` e `aluno`

PHP: acessando bancos de dados

- **mysql_select_db**: abre e utiliza o banco de dados associado com a última conexão aberta recentemente. Atenção: segundo parâmetro pode ser omitido, porém é boa prática sempre utilizá-lo para evitar erros.
- Se nenhuma conexão foi aberta recentemente, uma tentativa de executar **mysql_connect** é feita, sem parâmetros, o que também pode gerar efeitos indesejáveis

PHP: acessando bancos de dados

- **mysql_query**: utiliza 2 parâmetros, a consulta e a conexão associada ao banco de dados a ser consultado
- O segundo parâmetro pode ser omitido, porém é boa prática sempre utilizá-lo para evitar erros
- Se o segundo parâmetro for omitido, qualquer conexão já aberta é utilizada, ou, no caso de não haver conexões abertas, uma chamada a **mysql_connect**, sem argumentos, é feita, o que também pode gerar erros

PHP: acessando bancos de dados

- `mysql_fetch_row`: retorna um array contendo linha a linha de uma tabela do banco de dados associado à consulta
- O parâmetro de entrada é o resultado de uma chamada a `mysql_query`
- Esta função retorna falso quando todas as linhas da tabela foram recuperadas num comando de iteração

PHP: acessando bancos de dados

- `mysql_num_fields`: retorna o número de atributos de uma tabela, ou seja, o número de colunas
- O parâmetro de entrada é o conjunto de dados recuperado com a função `mysql_query`

PHP: acessando bancos de dados

- `mysql_close`: fecha a conexão aberta com `mysql_connect`
- Se o parâmetro for omitido, a última conexão aberta é fechada

PHP: acessando bancos de dados

- Outras funções estão disponíveis em PHP para manipulação de bancos
- `mysql_data_seek`: permite que o usuário acesse qualquer linha de tabela diretamente
- Exemplo: `mysql_data_seek($res, 10)` retorna um array contendo todos os valores de atributos da linha `10` do conjunto de dados retornado na variável `$res`
- Útil para reduzir o número de comunicações entre o servidor e o cliente

PHP: acessando bancos de dados

- **mysql_fetch_array**: da mesma forma que **mysql_fetch_row**, retorna um array contendo os valores de atributos de uma linha de tabela do banco de dados
- Diferença: array pode ser acessado através do nome do atributo
- Exemplo:

```
$row = mysql_fetch_array($res);  
echo $row["nome"];
```

[Abrir editor](#)[Abrir navegador](#)

- Utiliza parâmetro opcional que indica o tipo de retorno do array: associativo (**MYSQL_ASSOC**), numérico (**MYSQL_NUM**) ou ambos (**MYSQL_BOTH**)

PHP: acessando bancos de dados

- **Atenção:** `mysql_fetch_array`
 - ▶ Um atributo pode ser referenciado como `cliente.nome` em um comando `SELECT`, porém deve ser acessado como `$row["nome"]`
 - ▶ Agregados , por exemplo, `SUM(cost)` , podem ser utilizados no array da seguinte forma: `$row["SUM(cost)"]`
 - ▶ Valores nulos (`NULL`) são ignorados quando utilizados em arrays associativos, mas fazem diferença na ordem de acesso dos valores, quando a indexação do array é feita de forma numérica

PHP: acessando bancos de dados

- `mysql_fetch_object`: mais uma alternativa para recuperação de resultados de uma consulta
- Atributos podem ser acessados pelo nome, da seguinte forma:

```
$objeto = mysql_fetch_object($res);  
echo $objeto->nome;
```

- Também funciona:
- `echo $objeto->1;`
- Segundo parâmetro, opcional, retorna o tipo de acesso do array: associativo (`MYSQL_ASSOC`), numérico (`MYSQL_NUM`) ou ambos (`MYSQL_BOTH`)

PHP: acessando bancos de dados

- `mysql_free_result`: utilizada para liberar espaço em memória quando o programa utiliza muitas consultas seguidas

PHP: acessando bancos de dados

- `mysql_num_rows`: retorna o número de colunas de uma tabela
- Funciona apenas com consultas que envolvem SELECT
- Se apenas o número de colunas for importante, mas não os dados, é mais eficiente:

```
num_rows = mysql_query("SELECT count(*) FROM table", $con);
```

Nota:

Nesta transparência, onde eu falo "coluna", por favor, entendam "linha".
Gostaria de ressaltar que `mysql_num_rows()` devolve o número de linhas da tabela e não o número de colunas.

Desculpem a inconveniência. Se tiverem qualquer dúvida, por favor me contactem através do email ines@cos.ufrj.br.

PHP: acessando bancos de dados

- **mysql_pconnect**: funciona da mesma forma que **mysql_connect**, porém mantém a conexão ligada (persistente)
 - Necessária uma única vez no programa
 - Conexão será re-utilizada por outros programas desde que usando os mesmos parâmetros
 - Evita o *overhead* de abrir e fechar conexões várias vezes
 - Conexões abertas com **mysql_pconnect** não podem ser fechadas com **mysql_close**
 - Conexões expiram automaticamente e tempo depende da configuração do servidor mysql
 - Ajustado com:

```
safe_mysqld --set-variable connect_timeout=<n segs>
```

PHP: acessando bancos de dados

- `mysql_unbuffered_query`: semelhante a `mysql_query`, porém não utiliza buffer para recuperar resultados
- Consulta termina antes dos resultados estarem completamente disponíveis no cliente
- Não consome recursos no lado do cliente PHP
- Desvantagem: `mysql_num_rows` não pode ser utilizado, pois o número de linhas do conjunto de dados é desconhecido no cliente

PHP: acessando bancos de dados

- `mysql_change_user`: muda o usuário que está acessando o servidor mysql
- Utiliza 2 parâmetros, `username` e `password` do novo usuário
- Possui 2 parâmetros opcionais: o banco de dados e o descritor da conexão
- Se omitidos, PHP assume o banco de dados corrente e a última conexão aberta
- Retorna falso se ocorrer uma falha, e mantém a conexão do usuário anterior

PHP: acessando bancos de dados

- `mysql_drop_db`: implementa a função drop de mysql
- Utiliza um parâmetro de entrada: o nome do banco de dados
- Utiliza um parâmetro opcional, descritor da conexão
- PHP assume a última conexão aberta, se este parâmetro for omitido

PHP: acessando bancos de dados

- **mysql_fetch_field**: retorna um objeto, cujo conteúdo é o metadado de cada atributo associado com o conjunto de dados obtido por uma consulta
- Parâmetro opcional indica o número do atributo para o qual o metadado associado será recuperado
- Exemplo: utilização de **mysql_fetch_field** para emular o comportamento de **SHOW COLUMNS**

PHP: acessando bancos de dados

```
<?php
    $con = mysql_connect($_SERVER["REMOTE_ADDR"],
                          "aluno", "aluno");
    mysql_select_db("prog2", $con);
    $res = mysql_query("SELECT * FROM cliente LIMIT 1",
                       $con);
    print str_pad("Field", 20) . str_pad("Type", 14) .
        str_pad("Null", 6) . str_pad("Key", 5) .
        str_pad("Extra", 12) . "\n";
    for ($i=0;$i<mysql_num_fields($res);$i++)
    {
        <PROCESSA>
    }
    mysql_close($con);
?>
```

[Abrir editor](#)[Abrir navegador](#)

PHP: acessando bancos de dados

```
// <PROCESSA>
for ($i=0;$i<mysql_num_fields($res);$i++)
{
    $info = mysql_fetch_field($res);
    print str_pad($info->name, 20);
    print str_pad($info->type, 6);
    print str_pad((".$info->max_length."),8);

    if ($info->not_null != 1) print "YES";
    else print str_pad("",6);

    if ($info->primary_key == 1) print "PRI";
    elseif ($info->multiple_key == 1) print "MUL";
    elseif ($info->unique_key == 1) print "UNI";

    if ($info->zerofill) print "Zero filled";
    print "\n";
}
```

[Abrir editor](#)[Abrir navegador](#)

PHP: acessando bancos de dados

Field	Type	Null	Key	Extra
id	int(11)		PRI	
nome	varchar(50)		MUL	
endereco	varchar(80)	YES		

PHP: acessando bancos de dados

- **mysql_list_tables**: retorna um conjunto de tabelas que pode ser utilizado pela função **mysql_tablename** para obter nomes de tabelas
- Exemplo:

```
$Tables = mysql_list_tables("prog2", $con);  
$TableName_10 = mysql_tablename($Tables, 10);
```

PHP: acessando bancos de dados

- Outras funções não utilizadas na prática:
 - ▶ `mysql_fetch_length`
 - ▶ `mysql_field_flags`
 - ▶ `mysql_field_name`
 - ▶ `mysql_field_len`
 - ▶ `mysql_field_table`
 - ▶ `mysql_field_type`
- São um conjunto completo de `mysql_fetch_field`

PHP: acessando bancos de dados

- `mysql_result`: versão mais lenta de `mysql_fetch_row` ou `mysql_fetch_array`
- `mysql_fetch_assoc`
- `mysql_field_seek`
- `mysql_db_query` (não usada em versões mais recentes de PHP)

PHP: acessando bancos de dados

- Manipulação de erros:

- ▶ `int mysql_errno($con)`
- ▶ `string mysql_error($con)`

PHP: acessando bancos de dados

```
<?php
function exibeErro()
{
    die("Erro ".mysql_errno()." : ".mysql_error());
}

if (! ($con = @mysql_connect($_SERVER["REMOTE_ADDR"],
                            "aluno", "aluno")))
    die("Conexão falhou!");
if (! (mysql_select_db("xxx", $con)))
    exibeErro();
?>
```

[Abrir editor](#)[Abrir navegador](#)

PHP: Entrada de dados

- Até o momento, utilizamos apenas operações em bancos de dados existentes, sem entrada de dados do usuário
- Para fazer consultas baseadas em entrada de dados do usuário, utilizaremos **forms**

PHP: Entrada de dados

- Objetivos deste módulo:
 - ▶ Como passar dados de um navegador web para um servidor web
 - ▶ Acessar dados de usuários usando scripts
 - ▶ Interagir de forma segura com o banco de dados
 - ▶ Consultar bancos de dados a partir de dados do usuário
 - ▶ Construir scripts que contêm HTML `<form>` e o código que produz resultados para as consultas (**combined script**)
 - ▶ Usar o processo de 5 passos visto anteriormente para produzir componentes para entrada do usuário

PHP: Entrada de dados

- Há três formas possíveis de manipulação de dados de entrada do usuário:
 - ▶ Entrada de dados manual através de uma URL contendo um script PHP com os parâmetros do usuário
 - ▶ Entrada através de forms
 - ▶ Entrada através de hyperlinks que apontam para um script PHP contendo parâmetros do usuário
- Formas mais comuns: forms e hyperlinks

PHP: Entrada de dados

- Na prática, dados são transferidos do navegador web para o servidor web através do protocolo HTTP
- Usando HTTP, dados são transferidos utilizando dois métodos: GET e POST
- Forms utilizam GET e POST
- Os outros dois métodos utilizam apenas GET

PHP: Entrada de dados

- Transferindo dados para um script utilizando URLs:
- `http://webserver/~user/pegaEntrada.php?assunto=suspense`
- No script PHP `pegaEntrada.php`, uma nova variável é criada com nome `$assunto`, cujo valor passa a ser `suspense`
 - Útil para passar para o script campos e valores de tabelas para consultas
 - É possível especificar mais de um parâmetro:

`http://webserver/~user/pegaEntrada.php?
assunto=suspense&ano=2006`

PHP: Entrada de dados

- Entrada de dados através de forms (mais comum):

```
<html>
<head> <title> Entrada de dados com forms </title>
<body>
  <form action="pegaEntrada.php" method="GET">
    <br> Entre com o assunto:
    <input type="text" name="assunto" value="Todos">
    (Digite Todos para ver todos os assuntos)
    <br>
    <input type="submit" value="Mostrar livros">
</body>
</html>
```

[Abrir editor](#)[Abrir navegador](#)

PHP: Entrada de dados

- Utilizando hyperlinks:

```
Procure todos os <a href="livros.php?  
assunto=Todos&amp;ano=2006"> livros </a>
```

- Clicando neste link, uma requisição HTTP é criada para a URL:

```
http://webserver/~user/livros.php?assunto=Todos&ano=2006
```

- Livros.php é um script PHP que cria as variáveis **\$assunto** e **\$ano** e faz o processamento adequado de acordo com as necessidades do usuário

PHP: Entrada de dados

- Algumas dicas de segurança
- Dados que são passados de um navegador para um servidor web devem ser seguros, ou seja, não se deve permitir que o usuário escreva qualquer entrada que possa comprometer a segurança do servidor e do cliente

PHP: Entrada de dados

- Exemplo: assegurar que os dados de entrada têm o tamanho correto e não possuem caracteres especiais normalmente utilizados para ataques a sistemas

```
$assunto = clean($assunto, 30);
function clean($input, $tamMaximo)
{
    $input = substr($input, 0, $tamMaximo);
    $input = EscapeShellCmd($input);
    return ($input);
}
```

- Expressões regulares também são muito utilizadas para validar entradas do usuário e evitar efeitos indesejáveis

PHP: Entrada de dados

- Outra forma de melhorar segurança: utilizar os arrays associativos `$HTTP_GET_VARS`, `$HTTP_POST_VARS` ou `$HTTP_COOKIE_VARS`

- Exemplo:

`http://webserver/test.php?path=/root`

- Acesso: `value = $HTTP_GET_VARS["path"];`
- Não manipula diretamente a variável de ambiente `$PATH` do sistema

PHP: Entrada de dados

- Mais informações sobre como garantir segurança mais tarde
- Consultar livros texto e web
- Técnicas para escrita de código seguro mudam de versão para versão
- Procurar utilizar versões mais novas de software

PHP: Entrada de dados

- Utilização de scripts combinados: Os exemplos mostrados separam o script de entrada do usuário do script que vai processar a entrada
- Podemos combinar os dois (combined script)

PHP: Entrada de dados

```
<?php
    $script = "slide_p42.php";
    if (empty($_GET["assunto"]))
    {
?>
    <form action="<?=$script; ?>" method="GET">
        <br> Entre com um assunto:
        <input type="text" name="assunto" value="Todos">
        (Digite Todos para ver todos os assuntos)
        </br> <input type="submit" value="Mostrar livros">
    </form><br>
<?php
    } // fim do if empty($assunto)
    <PROCESSA ALGUMA CONSULTA UTILIZANDO $assunto>
?>
```

[Abrir editor](#)[Abrir navegador](#)