COMPONENTES PEAR
PEAR::DB

DIOGO CEZAR TEIXEIRA BATISTA
http://inf.cp.utfpr.edu.br/diogo
diogo@diogocezar.com

A BIBLIOTECA PEAR

- a construção do PEAR se iniciou no PHP4 de modo a fornecer uma biblioteca padrão de classes para as operações mais comuns em aplicativos PHP;
- ela é mantida como um projeto autônomo dentro do PHP, com documentação e etc à parte, mas é fornecida em cada novo release do PHP;
- especifica convenções que devem ser seguidas por quaisquer classes reutilizáveis no PHP;
- fornece um mecanismo padrão de tratamento de erros para as classes da biblioteca.

COMPONENTES FORNECIDOS PELO PEAR

- acesso a bancos de dados
- email
- opções de linha de comando
- conexões TCP e HTTP
- autenticação
- templates HTML
- XML
- documentação e testes automatizados

CLASSES DB DO PEAR

- fornecem uma API padrão para todos os bancos de dados suportados pelo PHP: não é mais preciso reescrever o código quando se muda de banco;
- necessitam dos módulos específicos para cada banco compilados / carregados no PHP, ou seja, o acesso ainda é nativo;
- a conexão a um banco é feita por um "método fábrica" que fornece um objeto especializado para o banco sendo utilizado no momento;

PEAR::DB

- DB: representa uma conexão a um banco de dados qualquer, fornecendo métodos para execução de comandos SQL e geração de seqüência;
- DB_Result: representa o resultado de um comando SELECT ou procedimento armazenado (ou seja, representa um cursor). Fornece métodos para a iteração pelas linhas / registros retornados e obtenção de metadados sobre o resultado.

Uma URL (ou DSN) PEAR DB

- protocolo://login:senha@host:porta/banco
  - protocolo: identifica o banco. Ex: mysql, pgsq, oci8, mssql, ...
  - login:senha: nome do usuário e senha para conexão ao banco;
  - host:porta: nome ou endereço IP do servidor, opcionalmente seguido da porta TCP;
  - banco: apenas para servidores que gerenciam múltiplos bancos.

CONECTANDO AO BANCO

- para abrir uma conexão com um banco de dados deve-se utilizar a função estática DB::connect() que possui a seguinte sintaxe:

```
DB::connect(string $dns);
```

- se a conexão funcionar, essa função retorna um objeto de conexão.
- para verificar se houve algum erro na conexão, utilizamos o método estático:

```
DB::isError($db)
```

$db é a conexão gerada anteriormente pelo DB::connect();

CONECTANDO AO BANCO

- uma sintaxe simples de conexão com o banco de dados:

```
include('DB.php');
$usuario  = "root";
$senha    = "****";
$servidor = "localhost";
$banco    = "teste";
$dns = "mysql://$usuario:$senha@$servidor/$banco";
$db = DB::connect($dns);
if(DB::isError($db)){
      die($db->getMessage());
}
else{
...
}
```

EXECUTANDO CONSULTAS

- podemos utilizar a função query para executar os nossos comandos SQL;
- em caso de sucesso na execução de um comando SQL SELECT, essa função retorna um objeto de resultado (DB_Result);
- para outras operações é retornado o valor constante DB_OK;
- em caso de falha é retornado um objeto de erro (DB_Error);
- por isso é sempre recomendado testar a ocorrência de algum erro com a função DB::isError().

EXEMPLO DE CONSULTA

```
...
$sql = "SELECT * FROM TABELA";
$resultado = $db->query($sql);
if(DB::isError($resultado)){
      die($resultado->getMessage());
}
else{
...
}
```

Depois da execução dessa consulta, os registros resultantes poderão ser acessados através do objeto retornado pela variável $resultado;

PERCORRENDO OS DADOS

- o objeto do resultado nos disponibiliza duas funções para obter as linhas resultantes:
  - fetchRow() : retorna uma linha do resultado e não tem parâmetros obrigatórios;

```
while($linha = $resultado->fetchRow()){
        $campo = $linha[0];
        echo $campo;
}
```

Ambas em um laço, percorrem os registros automaticamente;

  - fetchInto() : recebe um array como parâmetro que receberá a linha do resultado;

```
while($resultado->fetchInto($linha)){
        $campo = $linha[0];
        echo $campo;
}
```

PERCORRENDO OS DADOS

- existem ainda funções para obter informações sobre a consulta como por exemplo:
  - número de linhas resultantes: $resultado->numRows();
  - número de colunas resultantes: $resultado->numCols();
  - número de linhas afetadas (INSERT, UPDATE e DELETE): $db->affectedRows();

ENCERRANDO O PROCESSO

- ao terminar o processo é recomendável liberar o espaço da memória alocado para os registros, para isso utiliza-se a função free():

```
$resultado = $db->query("SELECT * FROM TABELA")
while($linha = $resultado->fetchRow()){
      $campo = $linha[0];
      echo $campo;
}
$resultado->free();
```

MODOS DE BUSCA DO RESULTADO

- entre os modos disponíveis para indexar os registros do resultado de uma consulta, temos:
  - DB_FECHMODE_ORDERED - é o valor default. Retorna um array ordenado, utilizando a ordem das colunas especificadas no comando SELECT;
  - DB_FETCHMODE_ASSOC - retorna um array associativo (equivalente ao fecth_array) utilizando os nomes dos campos como chaves;
  - DB_FETCHMODE_OBJECT - retorna um objeto de linha (DB_Row) utilizando os nomes das colunas como atributos;

```
Essas constantes podem ser passadas como
 parâmetro para a função fetchRow(), no
momento da busca pelas linhas desejadas.
```

EXEMPLOS

DB_FETCHMODE_ORDERED

```
$res = $db->query("SELECT codigo, nome FROM clientes");
$linha = $res->fetchRow(DB_FETCHMODE_ORDERED);
$codigo = $linha[0];
$nome = $linha[1];
```

DB_FETCHMODE_ASSOC

```
$res = $db->query("SELECT codigo, nome FROM clientes");
$linha = $res->fetchRow(DB_FETCHMODE_ASSOC);
$codigo = $linha['codigo'];
$nome = $linha['nome'];
```

DB_FETCHMODE_OBJECT

```
$res = $db->query("SELECT codigo, nome FROM clientes");
$linha = $res->fetchRow(DB_FETCHMODE_OBJECT);
$codigo = $linha->codigo;
$nome = $linha->nome;
```

OUTRAS FUNÇÕES ÚTEIS PARA RETORNO DE DADOS

- getOne() -> Retorna a primeira coluna da primeira linha do resultado da consulta, utilizando quando realizamos consultas que retornam um único valor;

```
$num = $db->getOne("SELECT COUNT(*) FROM clientes");
```

- getRow() -> Obtem a primeira linha do resultado e a retorna em um array;

```
$linha = $db->getRow("SELECT * FROM CLIENTES LIMIT 0,1");
```

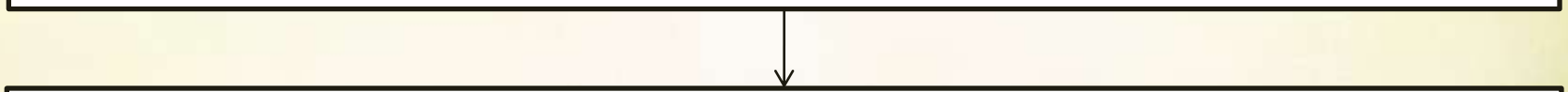
OUTRAS FUNÇÕES ÚTEIS PARA RETORNO DE DADOS

- getCol() -> Retorna um array contendo os dados da coluna selecionada;

```
$array_nomes = $db->getCol("SELECT nome FROM clientes");
```

- getAssoc() -> Retorna um array associativo contendo todos os registros resultantes de uma consulta SQL utilizando a primeira coluna como chave, por exemplo:

```
$dados = getAssoc("SELECT nome, cidade, estado FROM clientes");
```

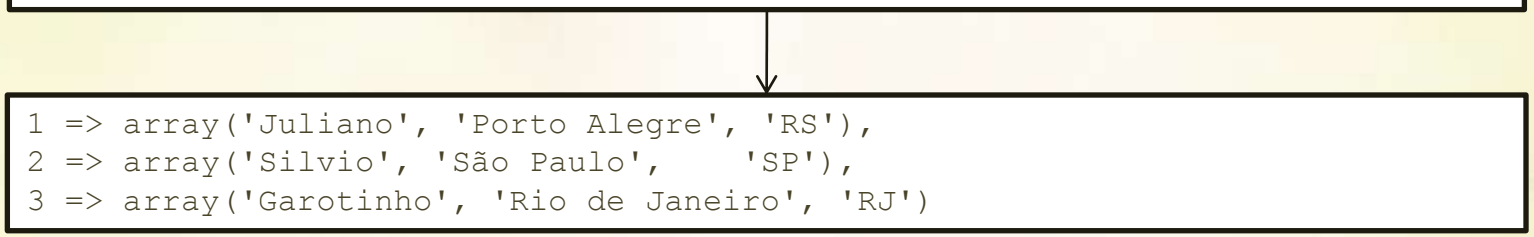


```
'Juliano' => array('Porto Alegre', 'RS'),
'Silvio'  => array('São Paulo',    'SP'),
'Garotinho' => array('Rio de Janeiro', 'RJ')
```

OUTRAS FUNÇÕES ÚTEIS PARA RETORNO DE DADOS

- getAll() - retorna um array contendo todos os registros resultantes de uma consulta sql, por exemplo:

```
$dados = getAll("SELECT nome, cidade estado FROM clientes");
```



```
1 => array('Juliano', 'Porto Alegre', 'RS'),
2 => array('Silvio', 'São Paulo',    'SP'),
3 => array('Garotinho', 'Rio de Janeiro', 'RJ')
```

ENCERRANDO A CONEXÃO

- após abrir a conexão e realizar as manipulações necessárias a conexão deve ser encerrada com o comando disconnect();

```
$db->disconnect();
```