SI401

Programação para a Web

2º Semestre - 2024

Programação *Back-End*: PHP
Unidade 17

Prof. Guilherme Palermo Coelho
guilherme@ft.unicamp.br

Roteiro

- Acesso a Banco de Dados via PHP;
- Referências.

Acesso a Banco de Dados

Acesso a Banco de Dados

- PHP possui uma série de funções nativas que permitem o acesso e a manipulação de dados em diferentes ***Sistemas Gerenciadores de Bancos de Dados*** (SGBDs);
- Aqui serão tratados apenas os mecanismos gerais de interação entre PHP e bases de dados relacionais (usando MySQL/MariaDB como exemplo);
  - **Não** serão discutidos detalhes sobre sistemas de bancos de dados e sobre a linguagem SQL (*Structured Query Language*);
  - Para maiores informações sobre o MySQL: http://dev.mysql.com/
  - Para maiores informações sobre a linguagem SQL: http://www.w3schools.com/sql/default.asp

Acesso a Banco de Dados

- A partir da versão 5, PHP passou a poder trabalhar com bases de dados de duas formas:
  - Via extensões, ***específicas*** para cada SGBD;
    - Oferecem APIs orientadas a objetos e procedurais;
  - Via **PDO** (*PHP Data Objects*);
    - Funciona com vários *SGBDs* diferentes → facilita mudanças no sistema;
    - Utilizam orientação a objetos.
- Neste tópico será tratada a utilização de PDO para acesso a Banco de Dados MySQL (ou MariaDB);

Acesso a Banco de Dados

- PHP *Data Objects* (PDO):
  - É uma extensão que define uma interface leve e consistente para acesso e manipulação de bancos de dados;
  - PDO provê uma camada de *abstração de dados*:
    - Permite que as mesmas funções possam ser usadas para execução de *queries,* independentemente do SGBD em uso;
    - Isso facilita eventuais migrações para outros SGBDs (não é necessário reescrever toda a aplicação)
  - É necessário que se tenha o *driver* associado ao SGBD a ser usado instalado na máquina:
    - Mais informações: http://php.net/manual/en/pdo.drivers.php

Conectando-se a um Banco de Dados

- O primeiro passo que deve ser tomado antes de se acessar um banco de dados para realização de operações é ***abrir uma conexão com o banco***;
- Em PDO, isto é feito **criando-se uma *instância*** da classe-base PDO:
  - O construtor recebe, como parâmetros:
    - Uma **string** com referência ao *servidor* e ao *nome da base de dados*;
    - Uma **string** com o *username* para autenticação (opcional);
    - Uma **string** com a senha do usuário para autenticação (opcional).

Conectando-se a um Banco de Dados

- Caso ocorra algum erro, uma *PDOException* é lançada;
- Mais informações:
  - http://php.net/manual/en/pdo.connections.php;

```php
<?php
    $sname = "localhost";
    $uname = "root";
    $pwd = "";

    try {
        $conn = new PDO("mysql:host=$sname;dbname=myDB", $uname, $pwd);
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }
    catch(PDOException $e){
        echo "Connection failed: " . $e->getMessage();
    }
?>
```

Esta mudança de parâmetro define que todo erro **deve** lançar uma *PDOException*.

Conectando-se a um Banco de Dados

- Por definição, a conexão com o BD permanece ativa enquanto o objeto PDO existir (até o final da execução do *script*).
  - Caso seja necessário fechar a conexão antes, deve-se eliminar o objeto PDO → atribuindo-se *null* à variável que o referencia.

```
<?php
    ...

    try {
        $conn = new PDO("mysql:host=$sname;dbname=myDB", $uname, $pwd);
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        ...
        $conn = null;
    }
    catch(PDOException $e){
        echo "Connection failed: " . $e->getMessage();
    }
?>
```

Executando comandos SQL

- Depois de estabelecida a conexão com o BD, é possível executar comandos SQL no banco de dados através do método ***exec***() do objeto da classe PDO;
- A sintaxe deste método é:
  - **$objetoPDO->***exec*(**string** *cmd*);
    - A variável **$objetoPDO** contém a referência ao objeto PDO criado;
    - O parâmetro *cmd* é uma *string* que contém o **comando SQL** a ser enviado ao BD (**não** deve terminar em ";");
- Também é possível realizar ***transações em batelada*** via PDO, mas isto não será tratado aqui.

Executando comandos SQL

```
<?php
    try {
        $conn = new PDO("mysql:host=localhost;dbname=myDB", "root", "");

        $sql = "CREATE TABLE produtos (codigo_produto smallint NOT NULL,
                PRIMARY KEY(codigo_produto),
                nome_produto varchar(80) NOT NULL,
                descricao_produto text,
                preco float NOT NULL,
                peso float,
                cod_categoria smallint NOT NULL,
                cod_subcategoria smallint NOT NULL,
                adicionais text)";

        $conn->exec($sql);
    }
    catch(PDOException $e){
        echo "Ocorreu um erro: " . $e->getMessage();
    }
?>
```

Executando comandos SQL

```php
<?php
    try {
        $conn = new PDO("mysql:host=localhost;dbname=myDB", "root", "");

        $sql = "INSERT INTO produtos VALUES (
                1,
                'Camiseta do Palmeiras',
                'Camiseta nas cores verde e branco',
                89.95,
                1.5,
                5,
                2,
                'Disponivel nos tamanhos: 3, 5, 9 e 10')";

        $conn->exec($sql);
    }
    catch(PDOException $e){
        echo "Ocorreu um erro: " . $e->getMessage();
    }
?>
```

Executando comandos SQL

```
<?php
    //…

    $conn = new PDO("mysql:host=localhost;dbname=myDB", "root", "");
    $stmt = $conn->query("SELECT * FROM produtos");

    //…
?>
```

- No exemplo acima, como executamos uma *query* que retorna resultados, utilizamos o método ***query***() do objeto PDO;
- Este método retorna um objeto da classe **PDOStatement**, que representa o conjunto de resultados retornado pela *query*;
- Para exibir estes resultados, é necessário explorar os métodos que tal classe provê.

Exibindo o resultado de comandos SQL

- A classe **PDOStatement** disponibiliza uma série de métodos que permitem tratar as informações retornadas pelos comandos SQL executados:

Método	Descrição
PDOStatement::rowCount	Retorna o número de linhas alteradas pelo comando SQL executado
PDOStatement::columnCount	Retorna o número de colunas no conjunto de resultados
PDOStatement::fetch	Obtém a próxima linha do conjunto de resultados;
PDOStatement::fetchAll	Retorna um vetor com todas as linhas do conjunto de resultados
...	...

- Mais informações: http://php.net/manual/en/class.pdostatement.php

Exibindo o resultado de comandos SQL

- Para ilustrar, vamos supor que queiramos exibir os resultados de uma consulta:
  - Para isso, vamos usar o método ***fetch***(), da classe *PDOStatement,* que retorna, a cada chamada, uma linha dos resultados*;*
  - Utilizaremos o parâmetro **PDO::FETCH_ASSOC** para o método ***fetch*()**, que coloca a linha retornada em um ***array* associativo** (nome do campo na tabela é o "índice" do vetor);
  - A cada execução de *fetch*() o ponteiro que indica a linha a ser processada no objeto *PDOStatement* é avançado para a posição seguinte.
- Mais informações sobre o método *fetch()*:

http://php.net/manual/en/pdostatement.fetch.php

Exibindo o resultado de comandos SQL

```php
<?php
   try {
         $conn = new PDO("mysql:host=localhost;dbname=myDB", "root", "");

         $stmt = $conn->query("SELECT * FROM produtos");

         while($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
                echo "<p>Nome do produto: ".$row["nome_produto"]."</p>";
                echo "<p>Descrição: ".$row["descricao_produto"]."</p>";
         }
   }
   catch(PDOException $e) {
         echo "Ocorreu um erro: " . $e->getMessage();
   }
?>
```

```
Nome do produto: Camiseta do Palmeiras
Descrição: Camiseta nas cores verde e branco
```

EXERCÍCIO SUGERIDO 3

Exercício Sugerido 3

- Crie um site que contenha um *menu* lateral (exibido via SSI) que contenha opções de acesso a duas páginas:
  - A primeira delas deve conter um formulário de cadastro de produtos para uma loja de artigos esportivos. Este formulário deve permitir o cadastro do ***código*** do produto, do ***nome***, da sua ***descrição***, do seu ***preço*** e do seu ***peso***. Ao clicar no botão "Cadastrar", o produto deve ser cadastrado em uma tabela de um banco de dados (use MySQL ou MariaDB).
  - A segunda página, ao ser carregada, deve consultar e listar para o usuário todos os produtos cadastrados no banco de dados.

Referências

Referências

[1] Niederauer, J. “Desenvolvendo Websites com PHP”, 2ª ed. Novatec, 2011.

[2] W3Schools PHP Tutorial:

http://w3schools.com/php/default.asp

[3] Manual do PHP:

http://php.net/manual/pt_BR/index.php

[4] Hudson, P. “Hacking with PHP”:

http://www.hackingwithphp.com/

[5] Prettyman, S. "Learn PHP 7“, Apress, 2016:

https://link.springer.com/book/10.1007/978-1-4842-1730-6