Integração do PHP com banco de dados

Prof. Alexandre de Oliveira Paixão

Prof. Kleber de Aguiar

Descrição

Linguagem de Programação PHP. Linguagem PHP. Linguagem de script PHP. Linguagem de Programação Web PHP. Integração de PHP com banco de dados. Integração de PHP com MySQL. Integração de PHP com PostgreSQL. PDO. Classe PDO. Métodos da Classe PDO. Integração entre HTML, PHP e banco de dados./p>

Propósito

Apresentar as funcionalidades da linguagem de programação PHP para integração com banco de dados, fazendo uso da extensão PDO – PHP Data Objetcs.

Preparação

Para aplicação dos exemplos, serão necessários: um editor de texto com suporte à marcação PHP; um servidor web com suporte à linguagem PHP e com um SGBD instalado e configurado para ser utilizado com PHP; um aplicativo Cliente para acesso ao SGDB. Em relação ao editor, no Sistema Operacional Windows, é indicado o Notepad++. No Linux, o Nano Editor. Quanto ao servidor, recomenda-se o Apache. Sobre o SGDB deverá ser utilizado o MySQL ou o PostgreSQL. Por fim, quanto ao aplicativo Cliente para acesso ao BD, recomenda-se o DBeaver.

Objetivos

Módulo 1

A Classe PDO em banco de dados

Definir a Classe PDO e sua utilização com MySQL e PostgreSQL.

Módulo 2

Métodos da Classe PDO

Descrever os principais métodos da Classe PDO.

Módulo 3

Construir uma aplicação com banco de dados

Construir uma aplicação contendo formulário HTML, tabela HTML e scripts PHP para inserção e listagem de dados em/de um SGDB.

Introdução

Ao longo deste conteúdo, veremos como integrar o PHP com banco de dados fazendo uso da Classe PDO (PHP Data Objects), uma interface de acesso a databases.

Veremos sua instalação, conceitos básicos, formas de utilização com MySQL e PostgreSQL, bem como utilização de dois dos seus principais métodos. Tudo isso apoiados em exemplos práticos.

Ao final, desenvolveremos um formulário HTML para a inclusão de informações em um banco de dados.

Vamos começar nossa jornada acessando os códigos-fontes originais propostos para o aprendizado de PHP com PDO. Baixe o arquivo, descompactando-o em seu dispositivo. Assim, você poderá utilizar os códigos como material de apoio ao longo do tema.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

1 - A Classe PDO em banco de dados

Ao final deste módulo, você será capaz de definir a Classe PDO e sua utilização com MySQL e PostgreSQL.

Banco de dados e linguagem PHP

Os bancos de dados são o ponto central de muitos sites, sistemas e aplicativos. Tais estruturas permitem o armazenamento e a recuperação de qualquer tipo de informação, desde dados simples, como os posts de um blog pessoal, a dados altamente sensíveis, como os dados bancários ou financeiros, por exemplo. Nesse sentido, a linguagem de programação PHP oferece amplo suporte à integração com bancos de dados, sejam relacionais ou não, sejam suportados por diversos Sistemas Gerenciadores de Bancos de Dados (SGBD), como SQL Server, Oracle, MySQL e PostgreSQL – sendo esses dois últimos, normalmente, associados ao PHP.

Inicialmente, antes de tratarmos da Classe PDO e de sua utilização com dois dos principais SGBDs existentes, vamos observar um pouco mais esses sistemas e sua integração com PHP.

Linguagem de programação PHP

É uma linguagem interpretada livre, capaz de gerar conteúdo dinâmico na internet.

Sistemas Gerenciadores de Banco de Dados

MySQL

O MySQL é um sistema de gerenciamento de banco de dados criado em 1995. Inicialmente de código aberto, foi adquirido posteriormente pela Oracle, que, atualmente, mantém tanto uma versão GPL quanto uma versão comercial. Trata-se de

um dos SGBDs mais utilizados na web, sendo, normalmente, associado ao PHP, dada a facilidade de conexão entre ambos.

Em PHP, estão disponíveis vários métodos e funções, através da extensão que leva o mesmo nome do SGBD, para conexão e execução de instruções SQL no MySQL.

Para fins de comparação, o código abaixo demonstra a conexão com o MySQL e a execução de uma instrução de consulta SQL utilizando as funções PHP específicas para o SGBD em questão.

PHP

PostgreSQL

O PostgreSQL é um SGBD de código aberto, atualmente disponível sob a licença BSD, criado em 1996. Enquanto o MySQL é um SGBD puramente relacional, o PostgreSQL é considerado um SGBD objeto-relacional. Na prática, essa diferença pode ser vista nas funcionalidades que o PostgreSQL possui, como a herança de tabelas e a sobreposição de funções. Outra diferença importante é que o PostgreSQL adere de forma mais próxima aos padrões SQL.

A linguagem PHP também possui uma série de métodos e funções, habilitados através da extensão pgsql, para conexão com o Postgres.

A exemplo do que fizemos com o MySQL, vejamos um código que utiliza funções PHP para conectar com o PostgresSQL e realizar uma instrução SQL de consulta de dados.

PHP

Conceitos básicos da linguagem de programação PHP

A linguagem PHP é uma linguagem **server side**, ou seja, atua no lado servidor, sobre um servidor web como o Apache, Nginx ou IIS. Trata-se de uma linguagem de script e que é gratuita, o que faz dela uma linguagem amplamente utilizada no ambiente web.

Em relação à sua sintaxe, é possível combinar código PHP com tags HTML, ou utilizar apenas código PHP em um script. Tal código deverá estar entre a tag inicial " " – sendo essa última não obrigatória, quando houver apenas código PHP em um script.

Tomando como base os exemplos de código anteriores, temos alguns recursos da linguagem sendo utilizados. A seguir, esses recursos serão revistos:

Constantes

As constantes são identificadores ou nomes que contêm um valor único e imutável, ao contrário das variáveis que, conforme o nome indica, possuem valor variável ao longo de um programa.

No exemplo, foram declaradas constantes para armazenarem as credenciais de acesso ao SGBD. A sintaxe de utilização de constantes é composta pela palavra reservada DEFINE, seguida de parênteses, onde são incluídos o nome da constante e o seu valor.

Variáveis

As variáveis são objetos, ou espaços reservados na memória do computador, destinados a armazenarem e a exibirem os dados utilizados, e alterados, durante a execução de um programa.

Em PHP, as variáveis são representadas por um $ seguido pelo seu nome, que é case-sensitive, ou seja, uma letra maiúscula é diferente de uma letra minúscula ($var é diferente de $Var). Em relação à declaração e atribuição de valores, ambos os processos podem ocorrer ao mesmo tempo. Veja no último exemplo que a variável $stringConn foi declarada (inserida pela primeira vez no código) ao mesmo tempo que recebeu um valor (processo de atribuição). Por último, as variáveis em PHP não possuem tipo. Logo, não é necessário indicar o tipo de dado a ser armazenado por uma variável.

Estruturas de decisão e repetição

Nos exemplos, foi vista a estrutura de repetição while. Além dela, há outras estruturas de repetição disponíveis em PHP: do-while, for e foreach.

A respeito das estruturas de decisão, estão disponíveis em PHP: if, else, elseif e switch.

Arrays

Os arrays, ou vetores, são variáveis que guardam uma lista de itens relacionados. Tais variáveis são compostas pelo par índice/valor. A variável $row, nos exemplos, é um array associativo, ou seja, seus índices são compostos por strings – o nome de cada coluna selecionada do banco de dados. Há ainda os arrays numéricos e os híbridos (arrays com índices associativos e numéricos).

Funções

As funções são pedaços de código que podem ser chamados a partir de qualquer parte do programa e que executam tarefas específicas. Além das funções definidas pelo usuário, em PHP estão disponíveis inúmeras funções nativas. Em nossos exemplos, temos as funções define e echo, além das relacionadas à conexão e ao manuseio de dados, como mysql_connect ou pg_connect, mysql_query ou pg_query, entre outras.

Extensões, bibliotecas e classes

As extensões, bibliotecas e classes são um importante recurso presente na maioria das linguagens de programação. Através delas, é possível estender as funções básicas de uma linguagem, adicionando novos recursos para a execução de determinadas tarefas.

Em relação à definição, em PHP, temos as extensões, que são arquivos binários (no S.O. Windows, as *.dll*, em S.O.s *unix-like*, os *.so*) que contêm bibliotecas. Já as bibliotecas são um conjunto de funções e/ou classes. As classes, então, dentro do paradigma de orientação a objetos, são um conjunto de códigos compostos por atributos e métodos. São exemplos de extensões os drivers para os SGBDs MySQL e PostgreSQL, que, quando habilitados, permitem a sua utilização com o PHP.

Dica

Consulte o Manual do PHP para uma lista completa de todas as suas extensões.

A Classe PDO

PDO e sua conexão com MYSQL e POSTGRESQL

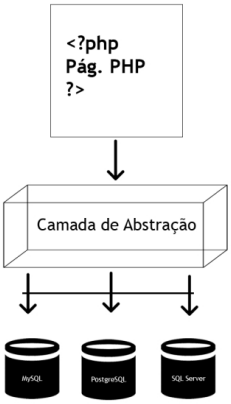
No vídeo a seguir, apresentamos a classe PDO e a conexão do PHP com MySQL e PostgreSQL, usando recursos de captura de telas.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

PDO

PDO, acrônimo para PHP Data Objects, ou Objetos de Dados PHP, e conforme definição vista anteriormente, é uma classe contida dentro de uma biblioteca e, por consequência, dentro de uma extensão. Ao referi-la, vamos chamá-la apenas de classe ao longo deste material.

Trata-se de uma interface leve para acesso a bancos de dados em PHP. Nesse sentido, cabe a cada banco de dados implementar a interface PDO a fim de expor os seus recursos e funções específicos que, então, ficarão disponíveis para serem utilizados através do código PHP.



Funcionamento da Camada de Abstração.

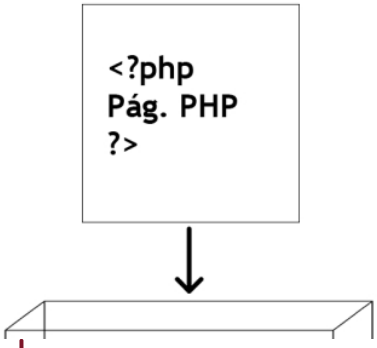
Para utilizarmos a Classe PDO, é necessário habilitar sua extensão no servidor web, ou seja, habilitar a extensão que contém o driver específico para cada banco de dados com os quais se deseja trabalhar.

Dica

Em relação à instalação e configuração da extensão PDO e drivers mencionados, há diversos tutoriais disponíveis na internet. Lembre-se de que esse processo é diferente de acordo com o sistema operacional utilizado. Na seção Explore +, separamos algumas referências base para esse processo.

Como mostrado na imagem Funcionamento da Camada de Abstração, assim como nos códigos de exemplo relacionados à conexão e execução de instruções próprios do MySQL e do PostgreSQL, vistos anteriormente, a principal vantagem em se utilizar PDO no lugar das funções nativas do PHP para cada SGBD é o fato dessa extensão fornecer uma camada de abstração de acesso a dados. Em outras palavras, isso significa que:

Funcionamento da Camada de Abstração



> É possível utilizar os métodos e as funções PDO independentemente do SGBD.

Na prática, e voltando aos códigos PHP para integração com cada SGBD, perceba que, se fôssemos trocar o banco de dados utilizado, precisaríamos também mudar algumas linhas de código: as de conexão com o banco de dados; as de execução de query, entre outras não vistas nos exemplos. Entretanto, fazendo uso do PDO, só precisaríamos mudar, nessa situação, o nome do driver a ser utilizado – conforme poderá ser visto adiante. No mais, todas as funções de execução de query, de tratamento de resultados, e demais, permaneceriam exatamente iguais.

Drivers PDO

Atualmente, estão disponíveis 12 drivers PDO, ou seja, é possível utilizá-lo com doze diferentes SGBDs. A tabela a seguir ilustra os drivers e os bancos de dados suportados. Vejamos!

Driver	SGBD Suportados
PDO_CUBRID	Cubrid

Driver	SGBD Suportados
PDO_DBLIB	FreeTDS / Microsoft SQL Server / Sybase
PDO_FIREBIRD	Firebird
PDO_IBM	IBM DB2
PDO_INFORMIX	IBM Informix Dynamic Server
PDO_MYSQL	MySQL 3.x/4.x/5.x
PDO_OCI	Oracle Call Interface
PDO_ODBC	ODBC v3 (IBM DB2, unixODBC e win32 ODBC)
PDO_PGSQL	PostgreSQL
PDO_SQLITE	SQLite 3 e SQLite 2
PDO_SQLSRV	Microsoft SQL Server / SQL Azure
PDO_4D	4D

Tabela: Drivers PDO.
PHP, 2020.

Conexão com o SGBD utilizando PDO

A primeira ação necessária ao trabalharmos com banco de dados é realizar a conexão com o próprio. Para isso, utilizamos o construtor da classe PDO, que é responsável por criar uma instância de conexão. O fragmento abaixo demonstra a conexão com MySQL e com o PostgreSQL.

PHP

Como pode ser visto, o que muda entre as duas instruções é o nome do drive/SGBD. Na primeira, temos “mysql” e na segunda “pgsql”. Além disso, outras opções/parâmetros podem ser passados no construtor, como a porta , por exemplo.

Reforçando o que foi dito, caso precisássemos alterar o SGBD usado em nosso projeto, só precisaríamos modificar essas informações acima e todo o restante do código, relacionado ao database, permaneceria inalterado.

Porta

Uma porta, em redes de computadores, tem a função de identificar aplicações e processos a fim de permitir que eles compartilhem uma única conexão com determinado endereço IP.

Nesse sentido, a porta padrão do Mysql é a 3306 e a do PostgreSQL a 5432.

Exceções em PHP

Tratamento de exceções

Uma boa prática, em qualquer linguagem de programação, é tratar as possíveis exceções que possam ocorrer no código. Cada linguagem possui uma sintaxe própria, sendo muito comum na maioria a utilização da instrução “try/catch”.

A maioria das linguagens de programação conta com instruções específicas para o tratamento de exceções. Em linhas gerais, temos uma exceção que pode ser lançada (*throw*) e capturada (*catch*), estando o respectivo código envolvido por um bloco try, cuja sintaxe deve contar com, ao menos, um bloco catch ou finally. Por fim, temos que o objeto lançado, ou seja, a exceção, precisa ser uma instância ou subclasse da classe Exception.

Em relação ao bloco catch, podemos ter vários relacionados a um bloco try, sendo cada um responsável pelo tratamento de um tipo de exceção. Já o conteúdo do bloco finally será sempre executado após o bloco try ou catch.

Saiba mais

Consulte por **Exceções** no Manual do PHP.

O tratamento de exceções é uma prática recomendadíssima em qualquer código produzido. Além de informar ao usuário ou mesmo ao desenvolvedor que um erro ocorreu e que, com isso, o programa não funcionará de acordo com o esperado, o tratamento de exceções possibilita também que outras partes do programa permaneçam funcionais ou que o programa não seja encerrado de forma inesperada, mesmo que um erro tenha ocorrido.

A partir de um dos exemplos anteriores, poderíamos utilizar o seguinte código, com tratamento de possíveis exceções, para realizar a conexão com o MySQL, por exemplo:

PHP

Encerramento de conexões e conexões persistentes

Encerramento de conexões

Para encerrar uma conexão aberta através de PDO, basta atribuir NULL à variável que contém a instância da classe. O mesmo deve ser feito com as variáveis que armazenam o resultado da execução dos métodos que executam instruções SQL, como Exec e Query – que serão vistos mais adiante.

A linha a seguir demonstra o encerramento da conexão aberta no código anterior:

PHP

Conexões persistentes

PDO oferece suporte a conexões persistentes – que, em linhas gerais, consiste em não encerrar uma conexão aberta com o SGBD ao final de execução de um script. Com isso, é possível fazer cache da conexão e reutilizá-la quando outros scripts requisitarem uma conexão semelhante (com as mesmas credenciais) a essa que ficou aberta. Para usar conexões persistentes em PDO, é necessário incluir um parâmetro a mais no momento de criação da instância. Veja como ficaria o nosso exemplo de conexão com o PostgreSQL:

PHP

Exemplo prático de conexão com o PostgreSQL

Confira a implementação de uma conexão de PHP com o banco de dados PostgreSQL.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Em relação às vantagens na utilização de uma Camada de Abstração de acesso a dados temos:

I. O uso de uma Camada de Abstração reduz a quantidade de trabalho, uma vez que fornece uma série de métodos e propriedades prontos para uso.

II. A Camada de Abstração unifica o processo de comunicação entre uma aplicação e um banco de dados.

III. A Camada de Abstração possibilita que o SGBD utilizado em uma aplicação seja trocado, sem que seja necessário reescrever todo o código responsável pela comunicação e transações com o banco de dados.

IV. A Camada de Abstração possui métodos prontos de acesso a SGBDs e execução de instruções. Com isso, sequer é necessário escrever as instruções SQL que se deseja executar, bastando configurar as credenciais de acesso ao banco de dados, como host, usuário e senha.

V. A Camada de Abstração facilita ao desenvolvedor a tarefa de estabelecer conexão entre a aplicação e o banco de dados.

Estão corretas:

A I, II, III e IV.

B III e IV.

C I, II, V.

D I, II, III e V.

E II, III e V.

Parabéns! A alternativa D está correta.

A Camada de Abstração, como PDO, facilita o trabalho de integração com bancos de dados. Com seu uso, é possível reduzir a escrita de código, sendo esse independente das especificidades de cada SGBD. Entretanto, além de configurar o acesso, é preciso escrever as instruções SQL que se deseja executar, além de tratar os seus resultados de acordo com a operação realizada.

Questão 2

Em relação à utilização da Classe PDO com a linguagem PHP, assinale a afirmativa correta.

A Para utilizar a Classe PDO, é preciso instalar os drivers de cada SGBD disponíveis para PHP.

B A principal desvantagem de se utilizar PDO é não contar com o suporte a conexões persistentes.

C Caso seja necessário alterar o SGBD utilizado, basta alterar o nome do driver na string de conexão – construtor da Classe PDO.

D O encerramento de conexões, quando utilizada a classe PDO, é feito de forma automática.

E Para utilizar a Classe PDO, é necessário apenas configurar a sua extensão no código da aplicação, sem a necessidade de habilitar nada no servidor web.

Parabéns! A alternativa C está correta.

Para utilizar a Classe PDO, é necessário instalar o driver PDO específico do SGBD com o qual se deseja conectar. Após realizada a conexão, que pode inclusive ser persistente, já que PDO tem suporte a esse recurso, o seu controle fica a cargo do programador. Logo, tanto o tratamento de exceções quanto o encerramento de conexões não são inerentes à classe, devendo ser codificados com recursos próprios da linguagem PHP, como o bloco try/catch, para as exceções, e a atribuição de null à variável que armazena a conexão, para o seu encerramento, por exemplo.

2 - Métodos da Classe PDO

Ao final deste módulo, você será capaz de descrever os principais métodos da Classe PDO.

Programação orientada a objetos em PHP

Orientação a objetos em PHP: classes, objetos e herança

Vamos iniciar apresentando exemplos práticos em PHP, da criação de classes, instanciação de objetos de uma classe e o mecanismo de herança.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Orientação a objetos em PHP

Como mencionado, uma classe, em linguagens de programação, é um conceito vinculado ao paradigma de orientação a objetos. Logo, antes de descrever a Classe PDO e seus métodos e, para melhor entendimento, serão descritos os conceitos básicos de OO (orientação a objeto) em PHP.

O paradigma de orientação a objetos, na engenharia de software, representa uma mudança na forma de se construir programas: no lugar de um conjunto de procedimentos e variáveis agrupados por determinado contexto ou objetivo, muitas vezes, não organizados adequadamente, na OO temos uma visão mais próxima ao mundo real representada por objetos e estruturas (classes) com as quais temos maior familiaridade.

Classes e objetos

Esses dois elementos são a base da programação orientada a objetos. As classes podem ser definidas como estruturas que definem um tipo de dados e podem ser compostas por atributos (variáveis) e por funções (métodos). Em alusão ao mundo real, são exemplos de classes: pessoas, produtos, veículos, imóveis etc. Assim como no mundo real, essas classes são compostas por características – os seus atributos. Por exemplo: uma pessoa possui nome, data de nascimento, CPF etc. Para acessar essas características, utilizamos métodos. Veja a representação dessa classe na imagem a seguir:

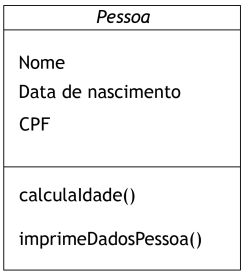


Diagrama de Classes – Classe Pessoa.

A partir dessa classe, poderíamos construir novas classes que compartilhassem os mesmos atributos e funcionalidades que a Classe Pessoa, além de poderem possuir propriedades e funções próprias. Veja, a seguir, o próximo exemplo:

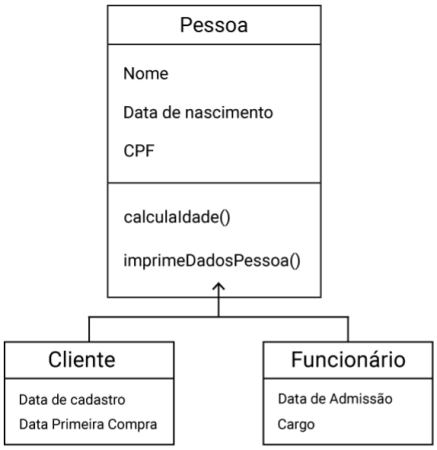


Diagrama de classes – exemplo de herança entre classes.

Esse compartilhamento de atributos e comportamentos, na orientação a objetos, recebe o nome de herança. Logo, dentro desse conceito, é possível criar classes, como as classes Cliente e Funcionário, que além de herdarem características de uma classe "pai", possuem ainda suas próprias características distintas – data de cadastro e data da primeira compra, em Cliente e data de admissão e cargo, em Funcionário.

Comentário

Há muito mais a ser visto sobre orientação a objetos. As considerações apresentadas anteriormente foram apresentadas de maneira resumida, a fim de iniciar esse assunto, ou mesmo revisá-lo, uma vez que utilizaremos alguns desses conceitos a seguir, quando abordarmos a Classe PDO, seus atributos e propriedades.

Por fim, a seguir, serão demonstrados exemplos práticos, em PHP, da criação de classes. Note os comentários inseridos no código, pois eles contêm algumas informações extras.

PHP

A partir dessa estrutura, a Classe “Fruta”, podemos declarar vários objetos “filhos” provenientes dela mesma. Esses objetos também são chamados de instâncias e contêm valores distintos, uma vez que a estrutura é dinâmica.

Veja a seguir um exemplo de criação de objeto oriundo da Classe “Fruta” e utilização dos métodos dessa classe através do objeto criado.

PHP

Como visto no código acima, a criação de novas “frutas”, ou de objetos da classe “Frutas”, é feita com a criação de uma variável que recebe o operador “new” seguido do nome da Classe. A partir desse ponto, essa variável em questão torna-se uma instância da classe. Na prática, essa instância possui os mesmos atributos e propriedades da classe da qual se originou.

Vamos praticar mais um pouco? Utilize o emulador abaixo, que contém o código da classe “Fruta” visto anteriormente. Após o código da classe, instancie novos objetos “Fruta” e utilize os seus métodos, como demonstrado no código anterior. Lembre-se de utilizar o comando “echo” para imprimir o valor de retorno dos métodos “getNome” e “getTipo”.

TUTORIAL COPIAR

Php

Input

Console

Executar

Os métodos da Classe PDO

A Classe PDO e seus métodos

A Classe PDO é dividida em duas classes: a própria PDO e a PDOStatement. Enquanto a primeira contém métodos para conexão com os SGBDs, para o controle de

transações e execução de queries; a segunda, PDOStatement, contém outros métodos relacionados às instruções executadas e ao seu retorno.

Os códigos a seguir mostram as estruturas dessas classes. Vejamos!

PHP

PHP

A seguir, conheça melhor os métodos:

Método EXEC

Este método, pertencente à Classe PDO, executa uma instrução SQL e retorna o número de linhas afetadas pela instrução. Sua sintaxe pode ser vista a seguir:

PHP

O código apresentado é funcional e complementar ao código demonstrado anteriormente, uma vez que o método Exec deve ser invocado a partir da instância para a Classe PDO (correspondente à variável $dsn em nosso código).

Em relação às linhas afetadas, tal informação é útil para confirmarmos se a operação foi executada com sucesso. Logo, podemos utilizar uma estrutura de decisão, como “if”, para verificar o conteúdo da variável $qtdeLinhasAfetadas. Caso nenhuma

linha tenha sido afetada, será retornado 0 (zero). Além disso, Exec pode retornar ainda o booleano FALSE ou então "" (vazio).

Por fim, cabe destacar que esse método não retorna à quantidade de linhas afetadas quando for executada uma instrução SELECT, o que pode ser feito através do próximo método que veremos, o Query<.

Método Query

O método Query, também pertencente à Classe PDO, tem função parecida com o método Exec. Entretanto, ele, além de executar uma instrução SQL, retorna – quando houver – um conjunto de resultados (result set) como um objeto PDOStatement. Em caso de falhas, é retornado o booleano FALSE. Vejamos um exemplo de sua utilização:

PHP

Perceba que, como o método retorna um objeto, é possível acessar seus índices na forma de índices de array. Para isso, foi utilizado o método fetch, um laço while que percorreu o result set retornado, imprimindo os dados selecionados.

A respeito do método fetch, que retorna um resultset no formato de array numérico e associativo, há ainda outros disponíveis: o fetchAll, fetchColumn e fetchObject. Além disso, esse método pode receber como parâmetro o tipo de retorno, ou seja, se deverá retornar um array associativo, numérico, associativo e numérico (que é o seu padrão por omissão), entre outros. Veja o exemplo de sua utilização retornando um array associativo:

PHP

Outros métodos importantes

Além desses dois métodos apresentados, as Classes PDO e PDOStatment possuem outros importantes métodos, como o Prepare e o Execute, por exemplo.

Considerando a sintaxe desses dois métodos, em comparação com o que vimos dos métodos Exec e Query, é boa prática fazer uso de ambos no lugar dos dois primeiros. Para embasar tal afirmação, veremos alguns conceitos extras, a começar pelo SQL Injection.

O Ataque SQL injection

SQL injection

O SQL Injection, ou Injeção de SQL, é um tipo de ataque baseado na manipulação e alteração de instruções SQL. Tal ataque é possível devido a vulnerabilidades encontradas em aplicações e sistemas que aceitam dados de entrada sem fazer o devido tratamento, além de executarem a conexão e as instruções SQL utilizando usuários com privilégios altos.

Vejamos um exemplo prático, normalmente, utilizado em formulários de login, onde poderia ser aplicada uma injeção de SQL:

PHP

Perceba que, no código, o conteúdo das variáveis POST 'login' e 'pswd', equivalentes aos campos input de um formulário HTML para login, são recebidos no PHP e atribuídos a novas variáveis. A seguir, o conteúdo dessas variáveis é utilizado em uma instrução SQL. Veja o que poderia acontecer se no formulário HTML fossem inseridos valores diferentes do previsto, por exemplo, os seguintes valores:

Exemplo de SQL injection em um formulário HTML.

Os valores inseridos no formulário mostrado na imagem anterior seriam recebidos no código PHP da seguinte maneira:

PHP

Veja que, no lugar dos valores esperados – login e senha –, seriam recebidos comandos que modificariam o sentido original da instrução SQL, no código PHP, permitindo assim o acesso ao sistema em questão.

Métodos Prepare e Execute

Para resolver os problemas acima, de SQL Injection, poderíamos escrever alguns códigos em PHP. Em linhas gerais, esses códigos conteriam instruções para tratar os dados recebidos antes de utilizá-los em instruções SQL. Esse trabalho poderia ser maçante, já que precisaria prever e tratar diversas formas de códigos maliciosos. Para facilitar e resolver tais questões, a Classe PDO possui um método chamado Prepare.

> **O método Prepare, como o nome indica, prepara uma instrução antes de ser executada, retornando um objeto do tipo statement, que será então executado através do método Execute (pertencente à classe PDOStatement). Ele faz uso de um recurso chamado bind. Tal recurso vincula a variável ou valor a ser utilizado na instrução SQL a uma outra variável (também pode ser utilizado o sinal "?"). Durante esse processo de preparação da instrução e bind dos valores, a Classe PDO realiza, de forma interna, ou seja, sem que precisemos nos preocupar com eles, uma série de tratamentos para evitar a SQL Injection.**

Vejamos um exemplo prático de utilização do método Prepare:

PHP

Fique atento à dica a seguir:

Dica

Repare no código que o método Prepare recebe como parâmetro a instrução SQL a ser executada e que, nos lugares onde seriam utilizadas as variáveis PHP com os valores provenientes do formulário HTML, são usadas outras variáveis, chamadas parâmetros nomeados (named parameters).

O método Execute faz o vínculo (*bind*) entre os *named parameters* e as variáveis PHP. O mesmo código apresentado anteriormente poderia ser executado tendo o sinal de interrogação "?" no lugar dos parâmetros nomeados. Veja, a seguir, como ficaria esse fragmento de código:

PHP

Teoria na prática

Para te ajudar na assimilação do conteúdo apresentado até aqui, chegou a hora de praticar! Todos os códigos anteriores, usados como exemplos, são funcionais. Logo, a começar pelo código responsável pela conexão com o SGBD (e tomando o cuidado de alterar as credenciais, nas constantes PHP, para refletirem os dados de seu ambiente), copie e execute cada um deles relacionados à Classe PDO.

Veja, por exemplo, as diferenças, na prática, entre os métodos Exec e Query; e entre esses dois métodos e os métodos Prepare e Execute. Tente realizar uma mesma consulta ou instrução SQL usando os dois e analise o resultado.

Ou seja, você deve:

- Incluir ao menos uma instrução para cada tipo possível: insert, update, delete e select.
- Fazer o tratamento de erros e exceções.
- Observar onde é possível melhorar os códigos utilizados. Você pode, por exemplo, incluir algumas verificações extras etc.

Mostrar solução

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Em relação ao método Exec, assinale a alternativa correta.

A O método Exec não retorna dados.

B O método Exec, por ser o método mais simples da Classe PDO, pode ser usado sem que uma instância de conexão com o banco de dados seja declarada.

C O método Exec possui uma estrutura de decisão própria, que pode ser representada por uma propriedade da Classe PDO, que permite tratar o seu retorno, havendo sucesso ou não em sua execução.

D O método Exec retorna o número de linhas afetadas pela instrução executada, exceto quando executada a instrução SQL SELECT.

E O método Exec pode retornar os booleanos FALSE ou TRUE.

Parabéns! A alternativa D está correta.

O método Exec é um método simples, que executa uma instrução SQL e retorna à quantidade de linhas afetadas, exceto quando executada a instrução SELECT. A partir do seu retorno, é necessário utilizar recursos da linguagem PHP, como

estruturas de decisão, para verificar se a operação executada obteve sucesso ou não.

Questão 2

Em relação ao método Query, assinale a alternativa correta.

A — O método Query retorna à quantidade de linhas afetadas pela instrução executada.

B — O método Query é idêntico ao método Exec, exceto pelo fato de retornar à quantidade de linhas afetas pela instrução SELECT.

C — O método Query retorna um objeto PDOStatement composto pelo conjunto de resultados da instrução executada ou FALSE em caso de falha.

D — O objeto $resultSet, nesse fragmento de código "$resultSet = $dsn->query($sql)", contém um result set em formato de array. Logo, ao executar a instrução "SELECT nome, cpf, telefone, ...", basta utilizar o índice correspondente à coluna para acessar seu valor. Por exemplo $resultSet['nome'].

E — O método Query pode retornar os booleanos FALSE ou TRUE.

Parabéns! A alternativa C está correta.

O método Query executa uma instrução SQL, retornando, quando houver, um conjunto de resultados como objeto PDOStatement. Com isso, para acessar os dados em uma instrução SELECT, por exemplo, é preciso fazer uso de um método da Classe PDOStatement, como o fetch, a partir do objeto que contém o conjunto de resultados.

3 - Construir uma aplicação com banco de dados

Ao final deste módulo, você será capaz de construir uma aplicação contendo formulário HTML, tabela HTML e scripts PHP para inserção e listagem de dados em/de um SGDB.

Primeiros passos

Resumo do módulo

Como este módulo é essencialmente prático, confira agora a execução dos códigos que serão apresentados ao longo do conteúdo.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Preparação

Este módulo terá caráter prático. Logo, todos os conceitos vistos nos módulos anteriores serão utilizados na confecção de um Formulário HTML, que enviará dados para um script PHP que os inserirá em um SGBD, nesse caso, no PostgreSQL, e de uma página para listagem desses mesmos dados. Cada etapa desse processo será apresentada em detalhes, desde a criação das tabelas no SGBD, passando pela criação do formulário HTML e chegando nos scripts PHP que farão a conexão e inserção dos dados através das classes PDO e PDOStatement.

Motivação

Deverá ser criado um formulário para armazenar os dados de clientes, contendo o seu nome completo (obrigatório), o seu CPF (obrigatório; apenas os números), um endereço de e-mail válido (obrigatório) e sua data de nascimento (obrigatório; dia, mês e ano). Além disso, deverá ser criada uma listagem para exibição dos clientes cadastrados.

Esquematizando o banco de dados

Os dados dos clientes deverão ser armazenados em uma tabela. Tal tabela deverá possuir um campo identificador, autoincremental e único.

O formulário HTML e o script PHP

A instrução SQL a seguir contém um exemplo de como a tabela cliente pode ser criada, observe:

SQL

A seguir, temos uma dica para você!

Dica

A sintaxe completa da instrução “Create Table”, para o PostgreSQL, pode ser vista em seu próprio manual.

Confeccionando o formulário HTML

O formulário HTML deverá ser escrito utilizando a HTML5. Além disso, os campos deverão ser validados de acordo com seu tipo (quando possível), tamanho e obrigatoriedade de preenchimento. Poderá ser utilizado um framework CSS, como o Bootstrap, para uma melhor apresentação do cadastro.

O código do formulário poderá ser visto ao final deste módulo, junto aos demais exemplos de código. Tal código poderá ser usado como template para desenvolvimentos similares.

Considerações sobre o formulário HTML

O formulário HTML possui elementos de ligação com o script PHP para o qual será submetido. A tabela a seguir apresenta esses elementos, seus atributos e uma breve descrição.

Elemento	Atributo	Obrigatório?	Des
form	*action*	Sim	Cor nor qua pas HT pro doc
form	*method*	Não. O padrão é POST	Def HT util trar dac PO
input	*name*	Sim	Cor car rec cor ass arra $_F $_F
button ou *input*	*type*	Sim	O v atri atri um car eve for

Tabela: Elementos de ligação entre o formulário HTML e o script PHP.
Alexandre de Oliveira Paixão

Em termos de boas práticas, é recomendado validar os dados a serem submetidos tanto no lado cliente, ou seja, no código HTML diretamente, com a utilização dos recursos introduzidos pela HTML5, ou através de Javascript, assim como no lado servidor. Com isso, caso a validação falhe, por algum motivo, do lado cliente, teremos a garantia de validação do lado servidor. Veremos a seguir como realizar a validação de nosso formulário nos dois ambientes – cliente e servidor.

Validação no lado cliente - HTML5

A partir da HTML5, é possível marcar um elemento como de preenchimento obrigatório fazendo uso do atributo required. No código do formulário, visto ao final deste módulo, os campos “Nome”, “CPF”, “Endereço de E-mail” e “Data de Nascimento” receberam esse atributo. Além disso, foram incluídas outras regras de validação: o campo CPF precisa ter exatos 11 caracteres (definidos com o uso dos atributos minlength e maxlength) e o endereço de e-mail precisa ser válido – tal validação é feita diretamente pela HTML, ao definirmos a tag <input> sendo do tipo (type) “email”.

Validação no lado cliente - Javascript

Confira agora um exemplo prático de criação de um formulário com validação JavaScript e a respectiva codificação PHP.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Para validar formulários, antes da HTML5, era necessário utilizar funções em códigos escritos na linguagem Javascript. Com isso, o formulário é verificado e, caso passe pelas validações, submetido ao servidor. Do contrário, a respectiva mensagem de falha é exibida, para as devidas correções. A validação do nosso formulário, utilizando Javascript, pode ser vista no código a seguir:

Javascript

Para utilizar a função Javascript acima, é necessário mudar um pouco o código do arquivo HTML que contém o formulário, removendo os atributos da HTML5, modificando o DocType para uma versão anterior da HTML, entre outros ajustes. Para facilitar, o código completo desse arquivo HTML foi adicionado ao final, junto aos demais códigos.

Codificando o script PHP que processará o formulário

O script PHP definido no atributo “action” da tag “form”, no arquivo HTML, é o responsável por receber os dados do formulário, validar os dados recebidos, conectar com o banco de dados e inserir as informações.

O atributo method, do formulário HTML, define o método HTTP utilizado para a transmissão dos dados – POST ou GET, sendo POST o valor padrão, caso esse atributo seja omitido no formulário. Para tratar os dados transmitidos por cada um desses métodos, há uma variável global pré-definida em PHP: $_POST e $_GET. Além disso, há também a variável $_REQUEST, que recebe os dados transmitidos por ambos os métodos.

Essas variáveis pré-definidas são, na verdade, um array associativo, cujos índices equivalem ao valor definido para o atributo name, em cada campo do formulário. Logo, se um input no formulário HTML for definido com “name=nome_cliente”, em PHP ele

poderá ser lido dessa forma: $_REQUEST['nome_cliente'] - ou através de $_POST['nome_cliente'] ou $_GET['nome_cliente'], dependendo do método utilizado.

Dica

Normalmente, em cenários reais, são utilizados padrões de projeto, como o MVC (Model-View-Controller), e paradigmas como a orientação a objetos, para realizar a integração entre a HTML, o PHP e o banco de dados. Tais questões foram deixadas de lado em nosso cenário, por estarem fora do escopo deste conteúdo. Entretanto, é recomendado estudar sobre esses padrões e paradigmas a fim de aplicá-los, garantindo assim maior qualidade ao código gerado, além de outros benefícios.

O script PHP que processa o formulário pode ser visto ao final deste módulo. Tal script realiza a conexão com o SGBD PostgreSQL, prepara a instrução SQL com o método Prepare e os insere no banco de dados através do método Execute. Mais adiante, é realizada uma verificação e, em caso de sucesso ou erro, exibida uma mensagem correspondente. Por fim, tanto a instância de conexão PDO quanto o objeto PDOStatement são encerrados.

Recuperação da informação

Listando os dados cadastrados

Embora o objetivo principal fosse o formulário de inserção de dados no SGBD, é interessante vermos também a recuperação de informações e exibição em uma página HTML. Como dito, embora funcional, o código utilizado para listagem das informações tem caráter apenas de estudo. Em situações reais, uma série de cuidados, tanto estéticos como de padrões de codificação, devem ser utilizados.

Nossos códigos de exemplo, que podem ser vistos ao final do módulo, são compostos por uma única página PHP contendo tanto códigos de programação quanto elementos HTML. Veja os códigos base nos exemplos.

Códigos: exemplos práticos

Código do formulário HTML para inserção de clientes

Veja o código de exemplo a seguir:

HTML

Código do formulário com validação em Javascript

Veja o código de exemplo a seguir:

Javascript

Script PHP para processamento do formulário HTML

Veja o código de exemplo a seguir:

PHP

Script PHP para listagem de clientes

Veja o código de exemplo a seguir:

PHP

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Para que um formulário HTML submeta dados para um script server side escrito em PHP, é necessário que alguns campos e atributos sejam definidos. Assinale a alternativa abaixo que corresponde a tais elementos.

A — Com a HTML5, não são necessários elementos específicos para que um formulário submeta dados para um script server side. Basta definir o atributo action.

B — Os atributos action e method, pertencentes à tag form são obrigatórios. O primeiro, para indicar qual script processará o formulário. O segundo, para definir o método HTTP de transmissão dos dados, pois, sem ele, o script PHP não é capaz de capturar as informações.

C — O atributo action, definido na tag form e o elemento input com "type=button", são os únicos elementos obrigatórios em um formulário para que seus dados sejam enviados para um script server side.

D — Os atributos action e name são obrigatórios. O primeiro, atribuído à tag form, indica qual script server side processará o formulário. O segundo, atribuído a cada elemento/campo pertencente ao formulário, define os nomes desses elementos e, consequentemente, o seu índice no array correspondente ao método HTTP utilizado. Além disso, o formulário deve conter um elemento button (ou um input) com "type=submit" para submeter o formulário.

E — O atributo method, definido na tag form e o elemento input com "type=button", são os únicos elementos obrigatórios em um formulário para que seus dados sejam enviados para um script server side.

Parabéns! A alternativa D está correta.

O formulário HTML possui elementos de ligação com o script PHP. Alguns desses elementos e/ou atributos são obrigatórios: o atributo action, da tag form; o atributo name, dos campos do formulário; e um elemento button ou um input com "type=submit". Há ainda um item que é opcional, o atributo method da tag form, que pode ser omitido, já que seu valor padrão é o método HTTP "post".

Questão 2

Em relação a um script server side, escrito em PHP, utilizado para processar os dados oriundos de um formulário HTML, assinale a alternativa correta.

A — O script server side responsável por processar o formulário HTML deve possuir a lógica necessária para receber os dados (através das variáveis globais relativas ao método HTTP usado no formulário), tratá-los (embora não obrigatório, é recomendado validar as informações. Além disso, ao usar a Classe PDO e seus métodos, é possível fornecer uma camada extra de segurança para evitar problemas como SQL Injection, por exemplo) e inseri-los no SGBD

(tarefa também facilitada pela Classe PDO, embora ainda seja necessário escrever as instruções SQL para cada operação a ser realizada).

B

É obrigatório que o script server side valide os dados recebidos do formulário, implementando as mesmas regras usadas para validação no lado cliente side.

C

O script server side precisa conter códigos específicos, escritos pelo programador, caso a caso, para tratar ameaças de SQL Injection.

D

Ao utilizarmos a Classe PDO para a conexão com o SGBD e inserção ou listagem de dados, não há necessidade de escrever as instruções SQL relacionadas. Ao invés disso, basta utilizar um dos métodos da Classe PDO, indicando qual a tabela que receberá os dados ou da qual os dados serão recuperados. A partir daí, o método em questão monta automaticamente a instrução SQL correspondente.

E

Os métodos POST ou GET são os métodos HTTP para transmissão de dados. No atributo method do formulário HTTP deve-se definir qual dos dois métodos será utilizado, através dos valores POST ou GET, sendo GET o valor padrão em caso de omissão do atributo method no formulário.

Parabéns! A alternativa A está correta.

O código de um script server side que processa os dados de um formulário pode ser otimizado com a utilização de recursos como a Classe PDO. Com isso, tarefas como tratamentos de segurança, conexão com o banco de dados, execução de instruções SQL e manuseio de dados recuperados são facilitados. Por outro lado, cabe ao programador utilizar corretamente esses recursos, além de escrever todo o código adicional necessário, como os de validação dos dados do formulário, por exemplo – tarefa essa que também pode ser facilitada com a utilização de técnicas como a orientação a objetos, onde métodos podem ser reaproveitados para a execução de tarefas repetitivas.

Considerações finais

Ao longo deste conteúdo, cujo propósito foi apresentar as funcionalidades da linguagem de programação PHP para integração com bancos de dados, demonstramos alguns recursos da linguagem. Alguns diretamente voltados para esse propósito, como as funções de conexão específicas com os SGBDs MySQL e PostgreSQL, além da classe PDO e alguns de seus métodos, outros relacionados à linguagem em si, como as constantes, as funções, as extensões, o tratamento de exceções, e o paradigma de orientação a objetos.

Ao final, cada um desses conceitos foi aplicado na construção de um formulário e de uma listagem HTML que insere e recupera informações de uma base de dados.

Podcast

Ouça agora um interessante bate-papo e fique ainda mais por dentro do assunto estudado!

Para ouvir o *áudio*, acesse a versão online deste conteúdo.

Explore +

Para saber mais sobre os assuntos explorados conteúdo, pesquise:

PHP. Mysql., no Manual do PHP.

PHP. PostgreSQL, no Manual do PHP.

PHP. Data Objects., no Manual do PHP.

O PHP possui algumas funções específicas para vários SGBDs, através das quais é possível realizar a conexão com o banco de dados e executar instruções SQL. Além disso, há funções próprias para a recuperação de dados, como as que transformam o conjunto de resultados em arrays numéricos ou associativos. O Manual do PHP contém a listagem dessas funções, assim como exemplos de sua utilização. Cabe ressaltar que, no lugar da utilização dessas funções específicas, deve-se dar preferência ao uso da Classe PDO como camada abstração para esse fim. Nesse mesmo manual, não deixe de ler também maiores informações sobre o método Fetch e equivalentes.

Referências

PHP. **Manual do PHP.** Publicado em: 5 ago. 2020. Consultado na internet em: 15 set. 2022.

Material para download

Clique no botão abaixo para fazer o download do conteúdo completo em formato PDF.

Download material

O que você achou do conteúdo?

Relatar problema