Integração do PHP com banco de dados

Você vai compreender os conceitos de banco de dados, os principais aspectos da linguagem de programação PHP, o paradigma de programação orientado a objetos (POO) e a integração do PHP com sistemas gerenciadores de banco de dados.

Prof. Alexandre de Oliveira Paixão

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

- Definir a Classe PDO e sua utilização com MySQL e PostgreSQL.
- Descrever os principais métodos da Classe PDO.
- Construir uma aplicação contendo formulário HTML, tabela HTML e scripts PHP para inserção e listagem de dados em/de um SGDB.

Introdução

Neste vídeo, veremos os principais assuntos que serão abordados ao longo do conteúdo, destacando o conceito de banco de dados e como ocorre a integração da linguagem de programação PHP com sistemas gerenciadores de banco de dados.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Banco de dados e linguagem PHP

É comum aplicações precisarem armazenar dados de alguma forma. Diversas aplicações utilizadas no nosso cotidiano ilustram essa afirmativa, como os sistemas da faculdade, as redes sociais, o bankline, entre tantas outras. Toda vez que acessamos o bankline, informamos nossas credenciais (login e senha), as quais são enviadas ao servidor responsável por hospedar a aplicação.

O servidor, por sua vez, analisa as credenciais recebidas. e, uma vez corretas, interage com o banco de dados e, posteriormente, envia a página de resposta contendo os dados bancários do cliente. Podemos utilizar PHP para desenvolver as aplicações que, entre outras tarefas, possibilita interagir com bancos de dados.

Neste vídeo, vamos demonstrar como podemos utilizar a linguagem PHP para desenvolver as aplicações que, entre outras tarefas, possibilita interagir com bancos de dados, de fundamental importância para acessar bankline, redes sociais etc.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

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.

Linguagem de programação PHP

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

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.

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.

Notebook em cima de uma mesa com o símbolo do MySQL exibido na tela.

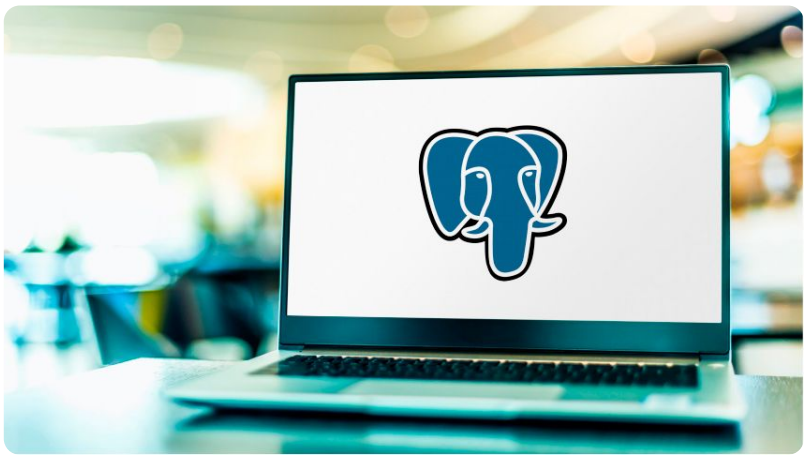
Em PHP, estão disponíveis vários métodos e funções, através da extensão que leva o mesmo nome do SGBD, para conexão e execução de instruções SQL no MySQL.

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.


Notebook em cima de uma mesa com o símbolo do PHP exibido na tela.

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

A exemplo do que fizemos com o MySQL, vejamos um código que utiliza funções PHP para conectar com o *PostgreSQL* 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.

Atividade 1

Analise a sentença e, em seguida, marque a alternativa correta.

Uma empresa de comércio eletrônico está desenvolvendo novas versões dos seus sistemas computacionais. Desse modo, é fundamental a utilização de banco de dados para armazenar registros de compras, entre outros. Concluímos que:

A

a linguagem PHP pode ser utilizada, pois oferece recursos de integração com vários bancos de dados.

B

podemos utilizar o banco de dados PHP para armazenar os dados em bancos relacionais.

C

podemos utilizar o banco de dados PHP para armazenar os dados em bancos não relacionais.

D

podemos utilizar o framework PHP para a construção das interfaces de usuário, ou seja, as páginas web.

E

PHP só pode ser utilizada como solução de backend para aplicações que utilizem computação em nuvem.

A alternativa A está correta.

PHP oferece recursos para integração com vários bancos de dados, sejam eles relacionais ou não. Além disso, PHP não é um framework de construção de interfaces de usuário e não é exclusividade de aplicações na nuvem.

A Classe PDO

Existem diversas linguagens de programação, assim como vários sistemas de gerenciamento de banco de dados. Consequentemente, na nossa trajetória profissional, podemos lidar com projetos que sejam desenvolvidos utilizando diferentes abordagens.

Independentemente das tecnologias consideradas, é importante conhecer as características essenciais para que uma aplicação se conecte ao banco de dados, permitindo a realização das operações CRUD (create, read, update, delete), ou seja, operações de: **criação** de um novo registro no banco de dados, **leitura**, **atualizaç ão** e **remoção** de algum dado já existente.

Portanto, com PDO, podemos estabelecer a conexão de aplicações com bancos de dados.

Neste vídeo, vamos entender a importância de se conhecer os processos de integração de aplicações com bancos de dados, de modo que se possa realizar as operações CRUD (create, read, update, delete).

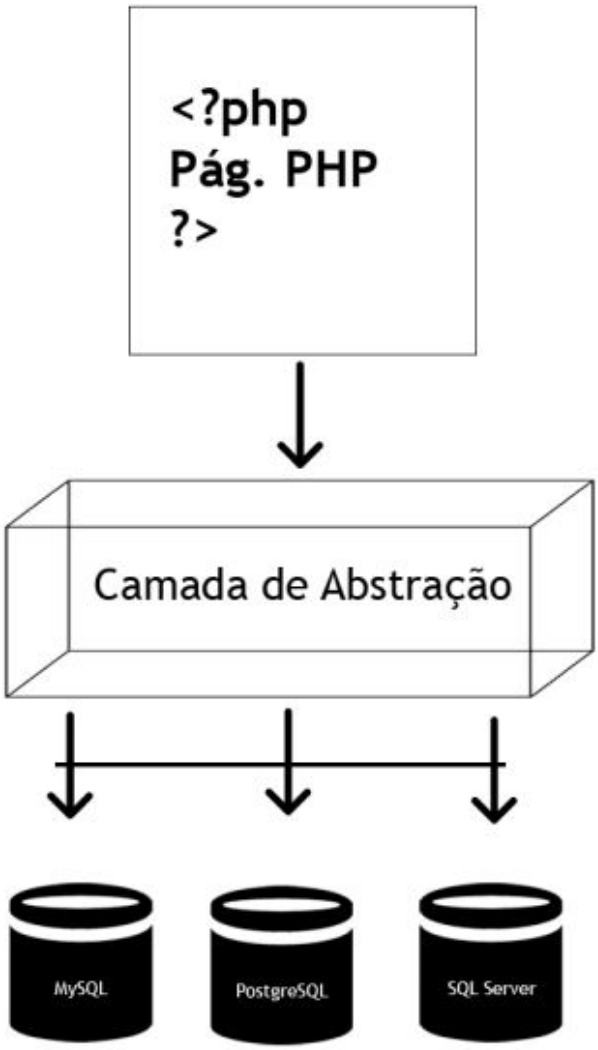
Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

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.



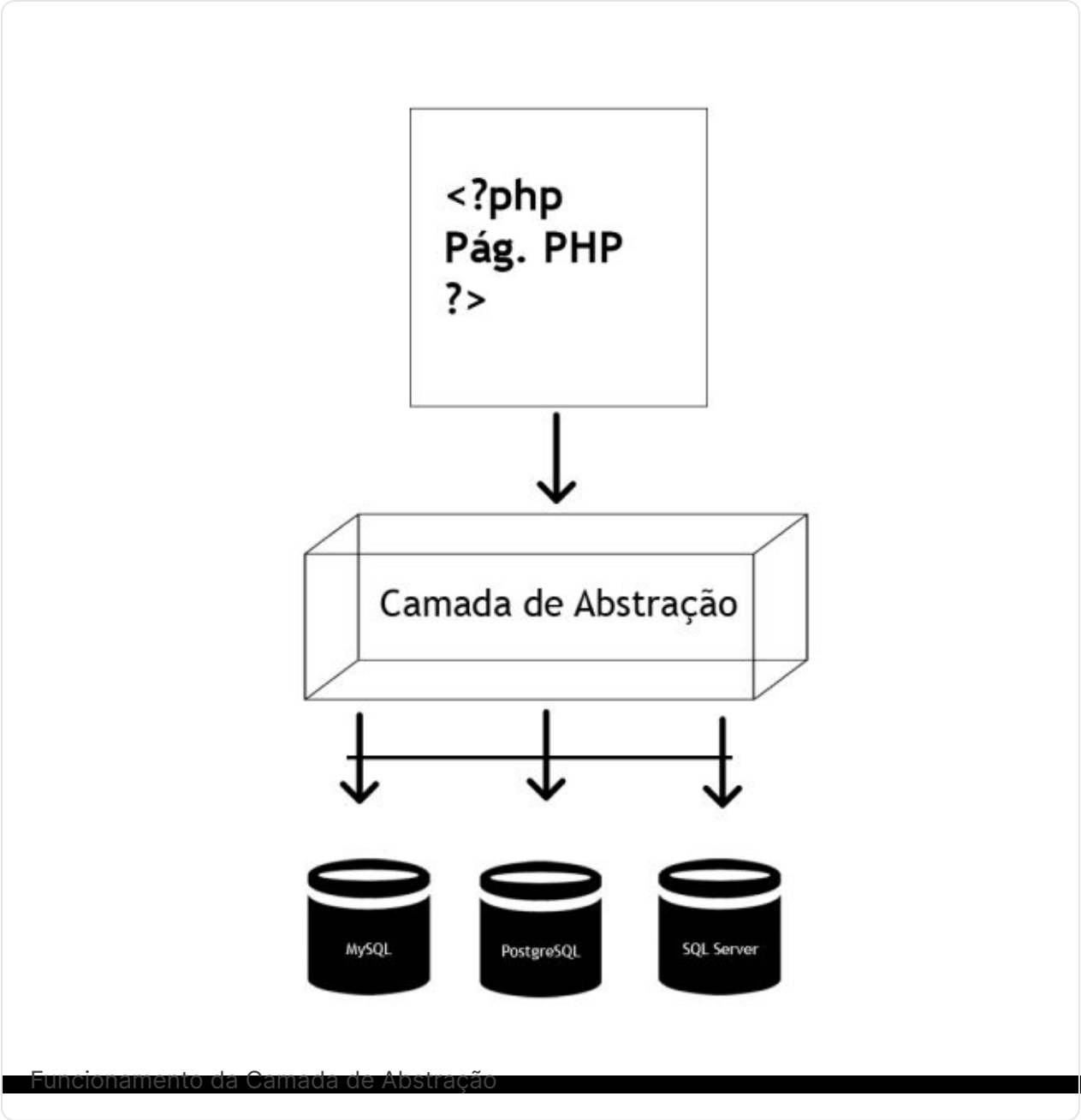
Funcionamento da Camada de Abstração.

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!

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
$dsn = new PDO("mysql:host=localhost;dbname=test", $user, $pass);

$dsn = new PDO("pgsql:host=localhost;dbname=test", $user, $pass);
```

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.

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.

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.

Atividade 2

Analise as sentenças e, em seguida, marque a alternativa correta em relação ao PDO.

A

Com PDO, podemos utilizar os recursos de HTML, CSS e JavaScript externamente, ou seja, as aplicações desenvolvidas com PHP têm recursos para referenciar recursos de frontend externamente.

B

A linguagem PHP oferece recursos como PDO, que, por sua vez, nos possibilita conectar aplicações a bancos de dados, uma etapa fundamental para a realização das operações CRUD.

C

Com PDO, podemos utilizar os recursos de Java externamente, ou seja, as aplicações desenvolvidas com PHP têm recursos para referenciar recursos de backend externamente.

D

PDO é uma linguagem de programação com funcionalidades que permitem a integração com PHP.

E

PHP ficou obsoleto e, com o avanço tecnológico, PDO substituiu o PHP.

A alternativa B está correta.

PDO é um recurso de PHP que possibilita a conexão de aplicações com bancos de dados, que é a base para a realização das operações de criação de um novo registro no banco de dados (create), a leitura (read), atualização (update) e remoção (delete) de algum dado já existente.

Exceções em PHP

Considere que uma aplicação crítica parou de funcionar de forma inesperada. Os prejuízos podem ser enormes. Portanto, deve-se evitar ao máximo esse tipo de situação. Uma alternativa muito importante é a denominada **tratamento de exceções**.

Mas, o que são as exceções?

Podemos considerar exceções os comportamentos atípicos dos programas, algo que fuja do esperado. Então, se uma exceção ocorrer, o programa deve detectá-la e executar outras linhas de código para contornar o problema, possibilitando a continuação do funcionamento.

PHP, assim como várias outras linguagens, tem recursos para lidar com o tratamento de exceções.

Neste vídeo, vamos entender a importância de considerar o tratamento de exceções no desenvolvimento de aplicações, tendo em vista que esta abordagem contribui para a robustez dos sistemas, evitando paradas de modo intempestivo.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

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

getMessage();
}
```

Atividade 3

A

Tratamento de exceções é o recurso oferecido por muitas linguagens de programação para armazenamento de conteúdos em variáveis maiores que os valores típicos dos tipos de dados.

B

Os tratamentos de exceções são muito importantes para garantir maior robustez no funcionamento de programas, mas a linguagem PHP não oferece suporte a isso, de modo que é trabalho do desenvolvedor evitar as exceções.

C

Os tratamentos de exceções são muito importantes para garantir maior robustez no funcionamento de programas e PHP oferece recursos para lidar com isso.

D

O recurso de tratamento de exceções disponibilizado por PHP refere-se ao CSS inline, ou seja, ao estilo definido na própria linha de código HTML.

E

O recurso de tratamento de exceções disponibilizado por PHP refere-se ao JavaScript inline, ou seja, ao estilo definido na própria linha de código HTML.

A alternativa C está correta.

Com tratamento de exceções, evitamos que os programas parem de forma inesperada, conferindo maior robustez às aplicações. E PHP, assim como várias linguagens, oferece recursos para lidarmos com as exceções, ou seja, são definidas as linhas de código capazes de gerar exceção e, caso ocorram, definem-se as linhas a serem executadas.

Encerramento de conexões e conexões persistentes

É comum liberarmos recursos que não são mais utilizados nos programas como, por exemplo, espaços de memórias de variáveis as quais não serão usadas novamente, assim como conexões a bancos de dados após o encerramento das operações.

Considere o usuário de um sistema que o acessou para realizar uma consulta aos seus dados e, após isso, ele finaliza sua aplicação. Consequentemente, não há necessidade de a conexão com o banco de dados permanecer ativa. Por outro lado, se o mesmo usuário desejar consultar outros dados em seguida, é interessante manter a conexão ativa. Com PHP, podemos alcançar esses dois objetivos.

Neste vídeo, vamos mostrar como, com PHP, é possível atingir os objetivos de desativar a conexão com o banco de dados, quando ela for desnecessária e, em seguida, se ela for necessária para consulta de novos dados, manter a conexão ativa.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

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

//...
$dsn = null;
```

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

 true)) ;

} catch (PDOException $e) {
        echo 'A conexão falhou e retornou a seguinte mensagem de erro: ' .$e-
>getMessage();
}
```

Atividade 4

Analise as sentenças e, em seguida, marque a alternativa correta em relação ao encerramento de conexões com bancos de dados com PHP.

A

Não se deve encerrar as conexões de aplicações com bancos de dados, mesmo após os usuários encerrá-las, porque isso degrada os bancos de dados.

B

Com PHP, as conexões com os bancos de dados são encerradas assim que os usuários fizerem as consultas aos bancos de dados. Ou seja, toda vez que quiser fazer uma consulta, deve-se estabelecer a conexão novamente.

C

As aplicações desenvolvidas com PHP não possibilitam o estabelecimento de conexões com bancos de dados. Para isso, deve-se utilizar recursos de JavaScript como AJAX.

D

Com PHP, pode-se desenvolver aplicações para encerrar as conexões com bancos de dados sempre que se desejar ou mantê-las ativas, mesmo após a finalização de um código.

E

Com PHP, só se pode estabelecer conexões persistentes, ou seja, conexões que ficam ativas mesmo após a finalização do programa ou após a troca de usuários em um sistema.

A alternativa D está correta.

É possível encerrar as conexões de aplicações desenvolvidas com PHP por meio de código, ou seja, o desenvolvedor define quando encerrá-las. Por outro lado, PHP também oferece recursos para conexões persistentes, ou seja, aquelas que permanecem ativas após o encerramento de um script.

Conectando o PHP ao PostgreSQL

As aplicações podem ser desenvolvidas em várias linguagens diferentes como, por exemplo, Java, C#, .NET, PHP, entre outras. E, também, há vários softwares de bancos de dados, como *PostgreSQL*, MySQL, MariaDB, entre outros. Mas, independentemente das tecnologias utilizadas, é preciso que as aplicações se conectem aos bancos de dados para realizarem as operações CRUD.

Neste vídeo, vamos ver como conectar o PHP ao *PostgreSQL* para realizar as operações CRUD e os recursos para facilitar a conexão.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Embora cada linguagem e cada banco tenham suas particularidades, normalmente, há recursos prontos para facilitar a conexão, de modo que os desenvolvedores podem utilizá-los para facilitar o trabalho.

A conexão entre PHP e *PostgreSQL* não é diferente. Há métodos prontos para se realizar a conexão e, geralmente, precisamos informar alguns parâmetros, como:

- Endereço da máquina em que está o banco de dados, podendo ser a mesma máquina em que a aplicação é executada (localmente) ou em máquina distinta (remotamente).
- Nome do banco de dados que se deseja conectar.
- Login de acesso ao banco de dados.
- Senha de acesso ao banco de dados.

Com a classe PDO, podemos instanciar um objeto e, durante esse processo, passar os dados citados da seguinte forma:

$dsn = new PDO("pgsql:host=". HOST . ";port=".PORT.";dbname=" . DBNAME .";user=" . USER . ";password=" . PASSWORD);

Roteiro na prática

Observe o trecho de código mostrado a seguir. Trata-se da conexão a um banco de dados *PostgreSQL* por meio de PHP.

```
php

getMessage();
}

$dsn = null;

?>
```

Da linha 4 até a 8, são definidos os parâmetros utilizados na conexão com o banco de dados, a saber:

- Linha 4: Local onde está o banco. Nesse exemplo hipotético, está na máquina com endereço IP igual a 127.0.0.1. Se estivesse na mesma máquina em que está a aplicação, seria localhost.
- Linha 5: Porta de acesso ao banco.
- Linha 6: Nome do banco.
- Linha 7: Login de acesso ao banco.
- Linha 8: Senha de acesso ao banco.

O código de acesso ao banco (linha 14) está dentro do bloco try, ou seja, se a conexão com o banco falhar por alguma razão, o bloco catch (linha 17) será executado e será mostrada a mensagem correspondente à execução do método getMessage().

Analisando novamente a linha de conexão (linha 14), instancia-se um objeto de PDO, denominado $dsn, que está relacionado à conexão com o banco. E, durante a instanciação desse objeto, passam-se os parâmetros de conexão definidos nas linhas 4 a 8, de modo que seus conteúdos serão concatenados com as demais strings de caracteres presentes nesse trecho de código, necessárias para o estabelecimento da conexão.

Atividade 5

Um desenvolvedor de uma fábrica de software está trabalhando no código de uma aplicação de e-commerce que utiliza PHP. Essa aplicação deve interagir com um banco de dados implementado com PostgreSQL, que armazena os dados de produtos vendidos e clientes, entre outros. O desenvolvedor está depurando o código e executando certas linhas PHP uma a uma. Durante essa depuração, ele se depara com a seguinte linha de código:

$dsn = new PDO("host=". HOST . ";port=".PORT.";dbname=" . DBNAME .";user=" . USER . ";password=" . PASSWORD);

O desenvolvedor então percebe que a linha de código citada contém um erro.

O que esse erro causaria na execução da aplicação e como ele pode ser corrigido?

A

O código causaria uma falha de conexão ao banco de dados, faltou informar o drive/SGDB “pgsql:” no construtor da classe PDO. Para corrigir o erro, a linha deverá incluir “pgsql:” no início da string parâmetro do construtor.

B

A linha de código mostrada corresponde à instanciação de objeto de PDO com os parâmetros de acesso ao banco, possibilitando a conexão de uma aplicação desenvolvida com PHP, mas com banco de dados MySQL ao invés do banco de dados PostgreSQL. O erro deverá ser corrigido substituindo a referência ao MySQL pela referência ao PostgreSQL.

C

O código causaria um erro porque o método PDO não suporta a conexão com PostgreSQL. A correção seria utilizar a função pg_connect() para se conectar ao banco. Após corrigida, o código usaria pg_connect() para a conexão.

D

O código causaria uma falha de conexão ao banco de dados porque o nome do banco de dados (dbname=" . DBNAME) deveria ser passado antes do host (host=". HOST). A correção seria alterar a ordem dos parâmetros dbname e host.

E

O código causaria um erro de sintaxe porque o PHP não aceita a concatenação de strings dentro do construtor da classe PDO. A correção seria fazer a concatenação dos parâmetros do construtor, atribuí-lo a uma variável e usar essa variável no construtor de PDO no lugar da concatenação.

A alternativa A está correta.

De fato, pode-se estabelecer a conexão com banco de dados *PostgreSQL* a partir da instanciação de objeto de PDO, passando os parâmetros de acesso ao banco, como localização, porta, nome do banco, além de login e senha de acesso. Porém, a linha de código apresentada possui um erro relacionado à falta do prefixo do driver de banco de dados "pgsql:" no início da string do construtor PDO. Sem esse prefixo, a aplicação não saberá que está tentando se conectar a um banco de dados *PostgreSQL*, resultando em uma falha de conexão. Para corrigir o erro, deve-se incluir "pgsql:" no início da string do parâmetro do construtor. A linha corrigida ficaria da seguinte forma:

$dsn = new PDO("pgsql:host=". HOST . ";port=".PORT.";dbname=" . DBNAME .";user=" . USER . ";password=" . PASSWORD);

Após corrigida, a linha de código criará uma conexão com o banco de dados *PostgreSQL* utilizando os parâmetros fornecidos na instanciação do objeto PDO, permitindo que a aplicação PHP se conecte com o banco de dados.

2. Métodos da Classe PDO

Programação orientada a objetos em PHP

O paradigma de orientação a objetos (OO) é muito utilizado em diversos sistemas computacionais, pois tira proveito de uma habilidade que temos desde crianças: a categorização. A partir de certa idade, sabemos identificar cachorros, gatos, bolas, carrinhos, bonecas, entre outros. Vamos considerar a categoria cachorros. Independentemente de tamanhos, cores, raças etc., a criança sabe identificar que se trata de cachorros. Então, nos referimos à classe de cachorros, e cada cachorro representa um objeto dessa classe.

Sim, em OO, objetos podem ser seres vivos também. Com PHP, podemos implementar aplicações sob o paradigma de OO.

Neste vídeo, vamos demonstrar como podemos implementar aplicações com o paradigma de orientação a objetos com PHP, sendo empregadas categorizações.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

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:

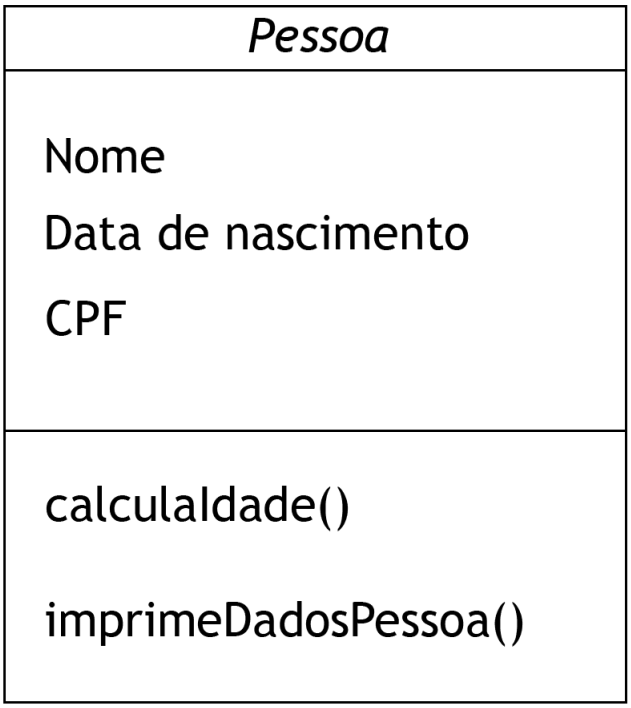


Diagrama de Classes – Classe Pessoa.

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:

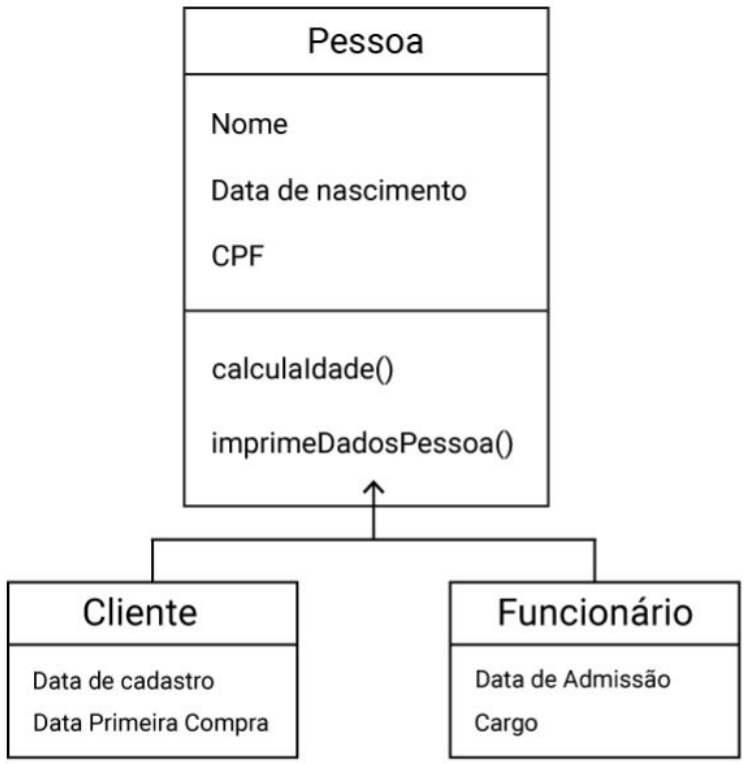


Diagrama de classes – exemplo de herança entre classes.

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
nome = $nome;
}
public function getNome()
{
return $this->nome;
}
public function setTipo($tipo)
{
$this->tipo = $tipo;
}
public function getTipo()
{
return $this->tipo;
}
}
```

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
// instanciando um objeto da classe Fruta
$fruta1 = new Fruta();
// utilizando o objeto criado para chamar o método “setNome”
$fruta1->setNome("morango");
// utilizando o objeto criado para chamar o método “setTipo”
$fruta1->setTipo("vermelha");
// utilizando o comando “echo” para imprimir o resultado da chamada ao método “getNome”
echo $fruta1->getNome();
// utilizando o comando “echo” para imprimir o resultado da chamada ao método “getTipo”
echo $fruta1->getTipo();
```

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”.

Conteúdo interativo

esse a versão digital para executar o código.

Atividade 1

Considere que você e seus colegas de trabalho estão conversando sobre computação em geral. Em determinado momento, comentaram sobre PHP e os paradigmas de programação. A título de exemplo, considere que cada uma das opções a seguir representa a opinião de cada um dos colegas. Qual deles está correto?

A

É possível desenvolver aplicações em PHP que utilizem o paradigma de orientação a objetos, pois ele compreende classes e objetos. As classes são compostas por atributos e métodos. Já os objetos são instâncias das classes. A orientação a objetos tem alguns pilares como o conceito de herança, no qual as classes filhas herdam recursos implementados nas classes pai (ou superclasse).

B

Não é possível desenvolver aplicações em PHP que utilizem o paradigma de orientação a objetos, apenas as que utilizem os demais paradigmas: procedural (estruturado), imperativo e declarativo.

C

É possível desenvolver aplicações em PHP que utilizem o paradigma procedural ou estruturado, pois ele compreende classes e objetos. As classes são compostas por atributos e métodos. Já os objetos são instâncias das classes. A orientação a objetos tem alguns pilares como o conceito de herança, no qual as classes filhas herdam recursos implementados nas classes pai (ou superclasse).

D

É possível desenvolver aplicações em PHP que utilizem o paradigma funcional, pois ele compreende classes e objetos. As classes são compostas por atributos e métodos. Já os objetos são instâncias das classes. A orientação a objetos tem alguns pilares como o conceito de herança, no qual as classes filhas herdam recursos implementados nas classes pai (ou superclasse).

E

É possível desenvolver aplicações com PHP que utilizem o paradigma de orientação a objetos, pois PHP é um framework e cada objeto é referenciado pelo ID da tag HTML que compõe as páginas web.

A alternativa A está correta.

É possível desenvolver aplicações em PHP que possam usufruir dos recursos estabelecidos pelo paradigma de orientação a objetos, com classes, objetos, conceito de herança, entre outros.

Os métodos da Classe PDO

A classe PDO tem recursos que possibilitam a conexão com bancos de dados. Isso é fundamental para que, posteriormente, seja possível realizar as operações CRUD (create, read, update, delete).

Não basta que as aplicações possam se conectar com os bancos de dados. É necessário também interagir com eles de alguma forma, para a inserção de novos registros, leitura, atualização e remoção de registros existentes.

Mas, como realizar essas operações? Os desenvolvedores terão que implementar os métodos para isso?

Não! Há recursos prontos para facilitar o trabalho de desenvolvimento por meio dos métodos da classe PDO.

Neste vídeo, vamos discutir a não necessidade de os desenvolvedores implementarem métodos para realizar as operações CRUD, havendo recursos prontos que facilitam esse trabalho de desenvolvimento por meio dos métodos da classe PDO.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

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
PDO {
      public __construct ( string $dsn [, string $username [, string $passwd [, array
$options ]]] )
      public beginTransaction ( void ) : bool
      public commit ( void ) : bool
      public errorCode ( void ) : string
      public errorInfo ( void ) : array
      public exec ( string $statement ) : int
      public getAttribute ( int $attribute ) : mixed
      public static getAvailableDrivers ( void ) : array
      public inTransaction ( void ) : bool
      public lastInsertId ([ string $name = NULL ] ) : string
      public prepare ( string $statement [, array $driver_options = array() ] ) :
PDOStatement
      public query ( string $statement ) : PDOStatement
      public quote ( string $string [, int $parameter_type = PDO::PARAM_STR ] ) : string
      public rollBack ( void ) : bool
      public setAttribute ( int $attribute , mixed $value ) : bool
}
```

```php
PDOStatement implements Traversable {
      /* Propriedades */
      readonly string $queryString;

      /* Métodos */
      public bindColumn ( mixed $column , mixed &$param [, int $type [, int $maxlen [,
mixed $driverdata ]]] ) : bool
      public bindParam ( mixed $parameter , mixed &$variable [, int $data_type =
PDO::PARAM_STR [, int $length [, mixed $driver_options ]]] ) : bool
      public bindValue ( mixed $parameter , mixed $value [, int $data_type =
PDO::PARAM_STR ] ) : bool
      public closeCursor ( void ) : bool
      public columnCount ( void ) : int
      public debugDumpParams ( void ) : void
      public errorCode ( void ) : string
      public errorInfo ( void ) : array
      public execute ([ array $input_parameters = NULL ] ) : bool
      public fetch ([ int $fetch_style [, int $cursor_orientation = PDO::FETCH_ORI_NEXT
[, int       $cursor_offset = 0 ]]] ) : mixed
      public fetchAll ([ int $fetch_style [, mixed $fetch_argument [, array $ctor_args =
array() ]]] ) : array
      public fetchColumn ([ int $column_number = 0 ] ) : mixed
      public fetchObject ([ string $class_name = "stdClass" [, array $ctor_args ]] ) :
mixed
      public getAttribute ( int $attribute ) : mixed
      public getColumnMeta ( int $column ) : array
      public nextRowset ( void ) : bool
      public rowCount ( void ) : int
      public setAttribute ( int $attribute , mixed $value ) : bool
      public setFetchMode ( int $mode ) : bool
```

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

exec("Delete From Cliente Where codigo_cliente = 1");
echo "Quantidade de linhas afetadas: " . $qtdeLinhasAfetadas
```

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<.< /p>

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

query($sql);
while ($row = $resultSet->fetch()) {
  echo $row['nome'] . "\t";
  echo $row['cpf'] . "\t";
  echo $row['telefone'] . "\n";
}
```

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

fetch(PDO::FETCH_ASSOC)) {
  //...
}
```

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.

Atividade 2

Analise as sentenças e, em seguida, marque a alternativa correta em relação aos recursos do PHP.

A

A classe PDO do PHP é um recurso com o objetivo de contornar uma limitação do PHP no que se refere à análise de dados quando se trabalha com ciência de dados, ou seja, tem recursos estatísticos para facilitar esse tipo de trabalho.

B

A classe PDO do PHP é um recurso com o objetivo de contornar uma limitação do PHP no que se refere ao tratamento de exceções, ou seja, para aumentar a robustez dos sistemas.

C

A classe PDO do PHP tem recursos que possibilitam a conexão com bancos de dados e a interação com eles.

D

PHP interage com o banco de dados denominado PDO, próprio do PHP, objetivando reduzir o tempo de acesso aos dados, ou seja, PHP utiliza seu próprio banco de dados.

E

PHP não tem recursos para acessar os dados armazenados em bancos de dados e, dessa forma, só deve ser usado para otimizar os estilos das páginas web.

A alternativa C está correta.

PHP tem recursos para acessar os dados em um banco de dados, entre eles, a classe PDO, que possibilita a conexão com bancos de dados e a interação entre eles e uma aplicação PHP.

O Ataque SQL injection

A segurança cibernética é uma área muito importante em computação. Além disso, temos a cultura DevSecOps, que incorpora conceitos de segurança durante o processo de desenvolvimento de software. Essa abordagem é importante, pois a computação está cada vez mais espalhada em diversos segmentos e, consequentemente, as preocupações com segurança devem acompanhar esta evolução.

Considerando uma aplicação web, que pode ser acessada por grande número de pessoas, uma tarefa muito importante é a validação de dados, desde a entrada nas páginas web por meio de formulários, para se evitar problemas como a inserção de códigos SQL maliciosos, denominados injeção SQL.

Neste vídeo, vamos entender a importância de evitar a injeção SQL, por meio da validação de dados, no contexto das preocupações com a segurança durante desenvolvimento de software.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

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

getMessage();
}

//Realizando uma consulta no BD através do login e senha recebidos por POST
$login = $_POST['login'];
$pswd = $_POST['pswd'];

$stmt = $dsn->prepare("Select * From Usuario Where login = :login And password
=:password");
$stmt->execute([':login' => $login, ':password' => $pswd]);

/*Aqui entraria o código para tratar o resultado da instrução SQL acima*/


//Destruindo o objecto statement e fechando a conexão
$stmt = null;
$dsn = null;
```

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

prepare("Select * From Usuario Where login = ? And password = ?");
$stmt->execute([$login, $pswd]);

//...
```

Atividade 3

A preocupação com a segurança cibernética deve fazer parte do desenvolvimento de software desde sua concepção. Há diversos problemas conhecidos e com soluções documentadas que continuam afetando muitas aplicações. Considere que as sentenças a seguir foram postagens em um fórum de tecnologia. Qual delas está correta?

A

O problema de injeção SQL é tipicamente resolvido inserindo-se login e senha de acesso ao banco de dados, bastando esse nível de proteção para impedir que esse tipo de ataque malicioso ocorra.

B

Durante o desenvolvimento de software, não se deve preocupar com a segurança cibernética. Qualquer problema de segurança é resolvido com firewall e antivírus.

C

A validação dos dados é muito importante para se evitar problemas de segurança cibernética, como injeção SQL. Portanto, desde o frontend, deve-se validar os dados que são inseridos em algum sistema computacional.

D

Injeção SQL é o nome dado ao recurso da classe PDO para se fazer a inserção de um novo registro no banco de dados.

E

Com PHP, não há problema de injeção SQL porque se trata de uma linguagem que possibilita o tratamento de exceções.

A alternativa C está correta.

Um meio de mitigar a possibilidade da ocorrência do ataque chamado injeção SQL é cuidar da validação dos dados tanto no lado frontend quanto no lado backend, visando evitar a ocorrência de inserção de códigos SQL maliciosos.

Demonstrações de ataques SQL injection

É muito importante fazer a validação dos dados desde o frontend para evitar problemas de injeção de SQL, ou seja, a inserção de códigos SQL maliciosos em formulários de páginas web que, se não forem detectados a priori, podem acabar sendo executados no banco de dados, trazendo consequências desastrosas.

Os problemas de violação de dados representam perdas financeiras significativas para as organizações, além de diversos outros problemas como a exposição de dados pessoais, entre outros. Por isso, é recomendado

tratar os dados sempre que possível, tanto no frontend quanto no backend, e levar em conta os preceitos de segurança cibernética.

Neste vídeo, vamos demonstrar de forma prática a importância de fazer a validação dos dados desde o frontend para se evitar problemas de injeção de SQL.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Roteiro de prática

Observe este trecho de código:

```
php
```

Os dados de login e senha, provenientes de formulários de uma aplicação, são obtidos por meio do método POST (linhas 4 e 5) e são passados diretamente para comporem a instrução SQL da linha 6.

Mas, e se alguém inserisse o código a seguir no campo de login do formulário?

login = ‘ ‘ OR true = true;

E, se no campo de senha, fosse inserido o seguinte código?

'*/--'

Consequentemente, a instrução SQL da linha 6 ficaria da seguinte forma:

$instrucaoSQL = "Select * From Usuario Where login = '' OR true = true;/* And password = '*/--'";

Resultando no acesso a alguém que pode não ter as credenciais adequadas para esse fim e, dessa forma, representando vulnerabilidade importante na segurança do sistema.

Atividade 4

Um desenvolvedor de uma fábrica de software está trabalhando no código de uma aplicação de uma seguradora que utiliza PHP. Essa aplicação deve interagir com um banco de dados implementado com *Postgre SQL*, responsável por armazenar os dados dos clientes, entre outros. O desenvolvedor está depurando o código e executando certas linhas PHP, uma a uma. Qual será o resultado se as linhas de código PHP a seguir forem executadas?

```
php
```

A

Trata-se de código que recebe os dados de login e senha, provenientes de banco de dados, por meio do método GET.

B

Trata-se da implementação do método read das operações CRUD, que faz a leitura de dados de um bando de dados, ou seja, códigos de geração do banco de dados PostgreSQL.

C

Trata-se de código que recebe os dados de login e senha por meio do método POST e constrói a linha de código de leitura do banco a partir desses dados. Porém, este código é vulnerável a problemas de injeção SQL, pois não faz a validação desses dados.

D

Trata-se de código que recebe os dados de login e senha por meio do método POST e constrói a linha de código de leitura do banco a partir desses dados. Além disso, este código é imune a problemas de injeção SQL, pois faz a validação desses dados.

E

Trata-se das linhas de código de construção de formulário de login e senha em uma página web.

A alternativa C está correta.

De fato, o código mostra a obtenção dos dados de login e senha pelo método POST e a construção do código SQL de acesso ao banco de dados a partir deles. Entretanto, os dados são passados diretamente ao código SQL sem qualquer validação, o que representa um risco caso haja códigos maliciosos sendo inseridos.

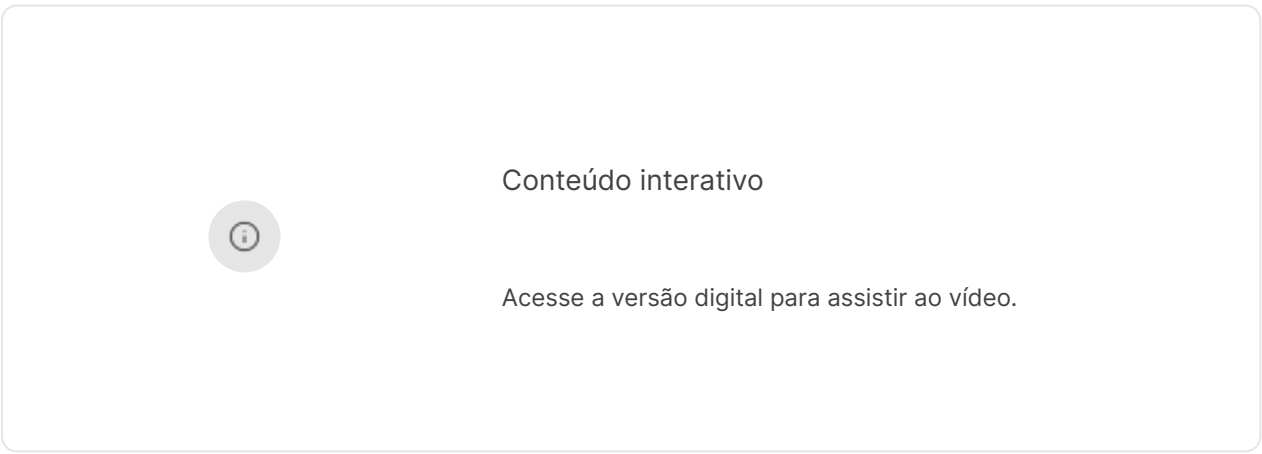
3. Construir uma aplicação com banco de dados

Primeiros passos na construção da aplicação

É comum que aplicações precisem acessar bancos de dados. Mas, como fazer isso na prática? PHP possibilita o acesso a bancos de dados? Precisamos de recursos adicionais? As respostas a essas perguntas já foram mostradas, mas vamos reavivar na nossa memória alguns conceitos importantes.

Inicialmente, é preciso ter páginas web que possibilitem a interação com o banco por trás da interface de usuário. Além disso, é necessário ter o banco de dados de modo efetivo. Inicialmente, é necessário trabalhar na sua modelagem, especificando suas características.

Neste vídeo, vamos ver como é possível fazer a integração de páginas web com bancos de dados por meio de PHP.



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.

Atividade 1

Quando a web surgiu, as páginas eram essencialmente estáticas, de modo que não havia a geração de conteúdo dinâmico e personalizado para cada usuário. Com o passar do tempo, a demanda por conteúdo dinâmico ficou evidente e, para implementar aplicações capazes de atender a essa necessidade, seria preciso utilizar repositórios de dados. Diante desse contexto, qual das alternativas está correta?

A

Para ter uma aplicação que acesse bancos de dados com PHP, devemos implementar o esquema de banco de dados em PHP que, por sua vez, fará a conversão para banco de dados não relacional, sendo a única forma de acesso a banco de dados com PHP.

B

Não é possível fazer páginas web com HTML que possibilitem interagir com usuários e acessem bancos de dados. Neste caso, deve-se substituir HTML por PHP.

C

Não é possível fazer páginas web com PHP que possibilitem interagir com usuários e acessem bancos de dados. Neste caso, deve-se substituir PHP por JavaScript.

D

Páginas web, desenvolvidas com HTML, podem interagir com banco de dados desenvolvido em PHP, que é a versão mais recente do SQL.

E

Podemos desenvolver páginas web, contendo formulários, as quais possam acessar bancos de dados. Para isso, podemos desenvolver aplicações com PHP que realizem a conexão com os bancos de dados.

A alternativa E está correta.

Podemos utilizar PHP para fazer aplicações que possibilitem a utilização dos dados (inseridos pelos usuários em formulários de páginas web) por bancos de dados, os quais foram previamente modelados e implementados.

O formulário HTML e o script PHP

HTML tem recursos para criação de formulários, amplamente utilizados em diversas aplicações, sejam aquelas que envolvem login e senha de acesso, ou campos de endereço para uma compra on-line.

É importante validar os dados antes do envio efetivo. Após o envio dos dados, pode-se utilizar códigos em PHP que os recebam e também façam a devida validação, aumentando a segurança das aplicações. Além disso, é possível utilizar PHP para a conexão com os bancos de dados e realizar as operações CRUD necessárias.

Neste vídeo, vamos entender a importância de se implementar páginas web com formulários em que os dados inseridos sejam extraídos por scripts feitos em PHP.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

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

```sql
CREATE TABLE cliente (
     id_cliente serial NOT NULL,
     nome_cliente varchar(255),
     cpf_cliente varchar(11),
     email_cliente varchar(255),
     data_nascimento_cliente timestamp,
     PRIMARY KEY (id_cliente)
);
```

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.

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

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
function validarFormulario(formulario){
  if(formulario.nome_cliente.value === "" || formulario.nome_cliente.value === null) {
    alert("O campo Nome não pode ficar vazio.");
    formulario.nome_cliente.focus();
    return false;
  }
  if(formulario.cpf_cliente.value.length != 11) {
    alert("O campo CPF precisa ter 11 caracteres.");
    formulario.cpf_cliente.focus();
    return false;
  }
  //o campo e-mail precisa ser válido, ou seja, deve : "@" e "."
  if(formulario.email_cliente.value.indexOf("@") == -1 ||
formulario.email_cliente.value.indexOf(".") == -1) {
    alert("O campo E-mail não é válido.");
    formulario.email_cliente.focus();
    return false;
  }
  if(formulario.data_nascimento_cliente.value === "" ||
formulario.data_nascimento_cliente.value === null) {
    alert("O campo Data de Nascimento não pode ficar vazio.");
    formulario.data_nascimento_cliente.focus();
    return false;
  }
}
```

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.

Atividade 2

Grande parte das aplicações web utilizadas considera a abordagem de geração de conteúdo dinamicamente, a partir de credenciais inseridas em páginas web. Como exemplo, podemos citar o acesso a bankline, redes sociais, entre outros. Analise as sentenças e, sem seguida, marque a alternativa correta.

A

É possível implementar páginas web com HTML contendo formulários e, dessa forma, os usuários podem inserir dados os quais serão enviados ao servidor e processados por códigos em PHP que interagem com bancos de dados.

B

Não é possível implementar páginas web com HTML que tenham formulários e os dados possam ser processados por scripts em PHP.

C

É possível implementar páginas web com CSS contendo formulários e, dessa forma, os usuários podem inserir dados os quais serão enviados ao servidor e processados por códigos em HTML que interagem com bancos de dados.

D

É possível implementar páginas web com PHP contendo formulários e, dessa forma, os usuários podem inserir dados os quais serão enviados ao servidor e processados por códigos em HTML que interagem com bancos de dados.

E

Os dados inseridos em formulários de páginas web devem ser validados e processados no próprio frontend com JavaScript, de modo que não é possível enviá-los ao servidor.

A alternativa A está correta.

A implementação de páginas web que possibilitem a inserção de dados como login e senha é de fundamental importância em diversas aplicações que interagem com bancos de dados.

Recuperação e exibição da informação em página HTML

Grande parte das aplicações são dinâmicas, ou seja, geram conteúdo de acordo com interações com os usuários e seus dados. Como exemplo, podemos citar os sistemas de bankline, redes sociais, sistemas acadêmicos, entre tantos outros. Os usuários entram com suas credenciais de login e senha e o sistema acessa algum banco de dados e exibe-os na tela como resposta do servidor. Portanto, o uso de formulários em páginas web é fundamental para aplicações dessa natureza.

Por outro lado, as aplicações dos servidores devem estar aptas a receber as solicitações, obter os dados enviados pelos usuários, processá-los, acessar bancos de dados e devolver os códigos das páginas de

resposta ao usuário. HTML tem tag de criação de formulários e seus campos, assim como PHP tem recursos para extrair os dados enviados em uma requisição e processá-los adequadamente.

Neste vídeo, vamos reforçar a importância de se implementar aplicações que possam recuperar dados armazenados em bancos de dados e exibi-los em páginas web.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Roteiro de prática

Inicialmente, vamos pensar em um banco de dados o qual deve ser modelado. Em seguida, implementam-se as tabelas do banco, utilizando códigos (como exemplificado a seguir), em que é criada uma tabela denominada cliente e tem os seguintes campos: id_cliente, nome_cliente, cpf_cliente, email_cliente, data_nascimento_cliente.

```
php

CREATE TABLE cliente (
     id_cliente serial NOT NULL,
     nome_cliente varchar(255),
     cpf_cliente varchar(11),
     email_cliente varchar(255),
     data_nascimento_cliente timestamp,
     PRIMARY KEY (id_cliente)
);
```

Depois, acessa-se o banco de dados com linhas de código semelhantes às mostradas a seguir:

```
php

getMessage();
}
/***PREPARAÇÃO E INSERÇÃO NO BANCO DE DADOS ***/
$instrucaoSQL = "Select id_cliente, nome_cliente, cpf_cliente,
email_cliente,data_nascimento_cliente From cliente";
$resultSet = $dsn->query($instrucaoSQL);
?>
```

Depois que as tabelas foram criadas e inseridos os registros, pode-se utilizar códigos para fazer a leitura desses dados e exibi-los em páginas web. Existem abordagens diferentes, mas pode-se utilizar linhas de código (como a mostrada a seguir) para exibir na tela os dados lidos em uma consulta.

```
php

fetch(PDO::FETCH_ASSOC){
        echo $row['id_cliente'];
        echo $row['nome_cliente'];
        echo preg_replace("/(\d{3})(\d{3})(\d{3})(\d{2})/", "\$1.\$2.\$3-\$4",
$row['cpf_cliente']);


echo $row['email_cliente'];
        echo date('d/m/Y', strtotime($row['data_nascimento_cliente']));
        }
?>
```

O método *$resultSet→fetch(PDO::FETCH_ASSOC)* em PHP é usado para obter uma linha de dados do resultado de uma consulta SQL usando a extensão PDO (PHP Data Objects). Na sequência, para cada linha, são lidos os dados de cada coluna e exibidos na tela.

Atividade 3

Um desenvolvedor de uma fábrica de software está trabalhando no código de uma aplicação bancária que utiliza PHP. Essa aplicação deve interagir com um banco de dados implementado com *PostgreSQL*, responsável por armazenar os dados dos clientes, entre outros. O desenvolvedor está depurando o código e executando certas linhas PHP, uma a uma. Qual será o resultado se as linhas de código PHP a seguir forem executadas?

```
php

query($instrucaoSQL);
        //...
        $row = $resultSet->fetch(PDO::FETCH_ASSOC);
        //...

?>
```

A

Podemos concluir que se trata do processo de consulta em um banco de dados a partir da conexão exemplificada por $dsn. O resultado da consulta é atribuído a $resultSet e, partir do método fetch(PDO::FETCH_ASSOC), podemos extrair o conteúdo lido e exibi-lo posteriormente em uma página web.

B

Trata-se do estabelecimento da conexão com um banco de dados denominado PDO::FETCH_ASSOC.

C

Podemos concluir que se trata do processo de inserção de novo registro no banco de dados denominado $dsn.

D

Trata-se da exclusão de um registro no banco de dados denominado resultSet.

E

Podemos concluir que se trata da atualização de um registro no banco de dados denominado fetch a partir do método denominado query.

A alternativa A está correta.

O método $resultSet = $dsn→query($instrucaoSQL) realiza uma consulta ao banco de dados a partir da conexão definida por $dsn e do código dado por $instrucaoSQL. O resultado da consulta é atribuído a $resultSet.

4. Conclusão

Considerações finais

- Mais detalhes sobre a integração de banco de dados com a linguagem PHP.
- A classe PDO.
- Como lidar com tratamento de exceções em PHP.
- Como estabelecer o encerramento de conexões com banco de dados.
- Como estabelecer conexões de aplicações PHP com banco de dados.
- Como reconhecer o ataque de injeção SQL e como preveni-lo.

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.