



ERROS E EXCEÇÕES EM PHP

Prof. Me. Hélio Esperidião

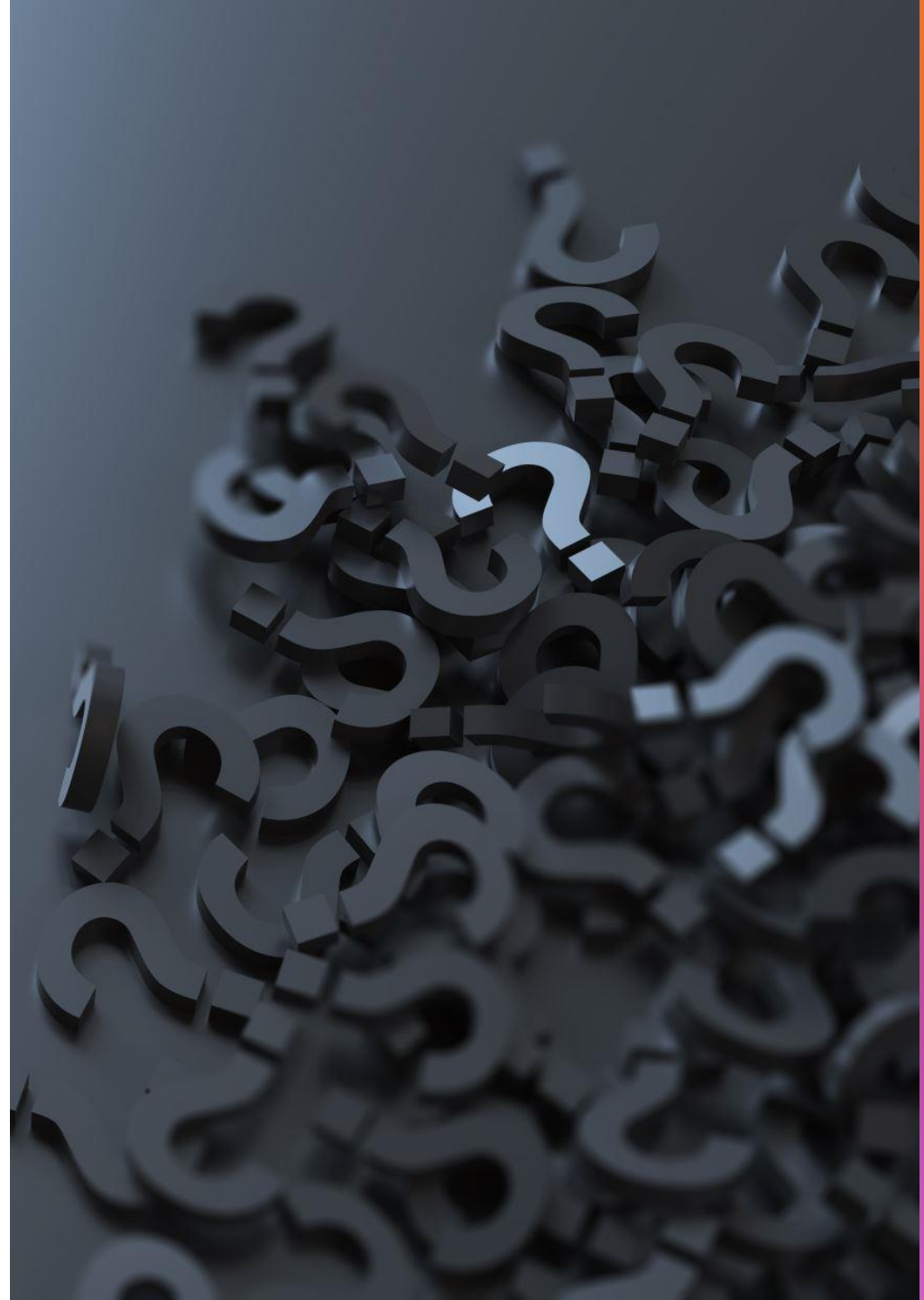
O que é uma Exceção?

- Definição:
 - Uma exceção é um evento indesejado ou inesperado durante a execução de um programa.
- Consequência:
 - Interrompe o fluxo normal de instruções.
- Causas:
 - Pode ser causada por diversos tipos de erros.



exceções e os erros

- Em PHP, as exceções e os erros são gerenciados de maneira diferente, especialmente a partir do PHP 7, quando a hierarquia de erros foi reorganizada.
- Entender a diferença entre Exception e Error ajuda a saber quando usar cada um



Diferença entre Exception e Error

- **Exception:**

- Classe base para todas as exceções. Usada para representar erros que podem ser previstos e tratados durante a execução do script.

- **Error:**

- Introduzida no PHP 7, é a classe base para todos os erros de programação que não são esperados e geralmente resultam em falha de execução. Inclui erros como TypeError, ParseError, ArithmeticError, DivisionByZeroError.

Quando usar Exception vs. Error

- **Exception:**

- Para capturar e tratar exceções lançadas explicitamente no seu código ou por funções/bibliotecas que usam exceções.

- **Error:**

- Para capturar erros fatais do PHP que não são lançados como exceções.
- Isso inclui erros de tipo, aritmética, parsing, etc.

Sintaxe

```
<?php
```

```
try {
```

```
    //tenta realizar uma determinada tarefa
```

```
} catch (Error $e) {
```

```
    //caso aconteça algum erro esse bloco é executado
```

```
}
```

```
catch (Exception $e) {
```

```
    //caso aconteça uma exceção
```

```
}
```

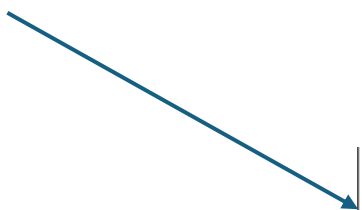
Não tratar

O que aconteceu?

```
<?php
```

```
    $x = 10 / 0;
```

```
?>
```



Fatal error: Uncaught DivisionByZeroError: Division by zero in C:\xampp\htdocs\exemplo.php:2 Stack trace: #0 {main} thrown in **C:\xampp\htdocs\exemplo.php** on line 2

Como tratar? 😊

Os erros geralmente resultam na interrupção imediata do script PHP
Exemplos de erros em PHP incluem ParseError, FatalError, TypeError, entre outros.

```
try {  
    $x = 10 / 0;  
} catch (Error $e) {  
    $objResposta = new stdClass();  
    $objResposta->erro = $e->getMessage();  
    echo json_encode($objResposta);  
}
```



```
function dividir($a, $b) {  
    $a = 10;  
    $b = 0;  
    if ($b == 0) {  
        throw new Exception('Não é possível dividir por zero');  
    }  
    return $a / $b;  
}  
  
//chamada do método dividir:  
try {  
    echo dividir(5, 0);  
} catch (Exception $e) {  
  
    $objResposta = new stdClass();  
    $objResposta->erro = $e->getMessage();  
    echo json_encode($objResposta);  
}
```

Uma exceção em PHP é uma condição que indica uma anomalia durante a execução do código, mas que pode ser tratada.

As exceções permitem que você escreva código para lidar com o problema de forma graciosa, seja corrigindo-o ou lidando com ele de alguma outra maneira

Aplicando a classe Banco

```
private static function conectar() {  
    // Desativar exibição de erros temporariamente  
    error_reporting(0);  
    try {  
        // Tenta estabelecer uma nova conexão utilizando as informações fornecidas  
        Banco::$CONEXAO = new mysqli(Banco::$HOST, Banco::$USER, Banco::$PWD, Banco::$DB, Banco::$PORT);  
        if (Banco::$CONEXAO->connect_error) {  
            throw new Exception("Erro ao conectar ao banco de dados: " . Banco::$CONEXAO->connect_error);  
        }  
    } catch (Exception $e) {  
        // Em caso de qualquer outra exceção, trata normalmente  
        $objResposta = new stdClass();  
        $objResposta->cod = 1;  
        $objResposta->erro = $e->getMessage();  
  
        die(json_encode($objResposta));  
    } catch (Error $e) {  
        $objResposta = new stdClass();  
        $objResposta->cod = 2;  
        $objResposta->erro = $e->getMessage();  
        echo json_encode($objResposta);  
        die(json_encode($objResposta));  
    }  
}
```

O print apresenta uma situação onde o servidor de banco de dados está offline

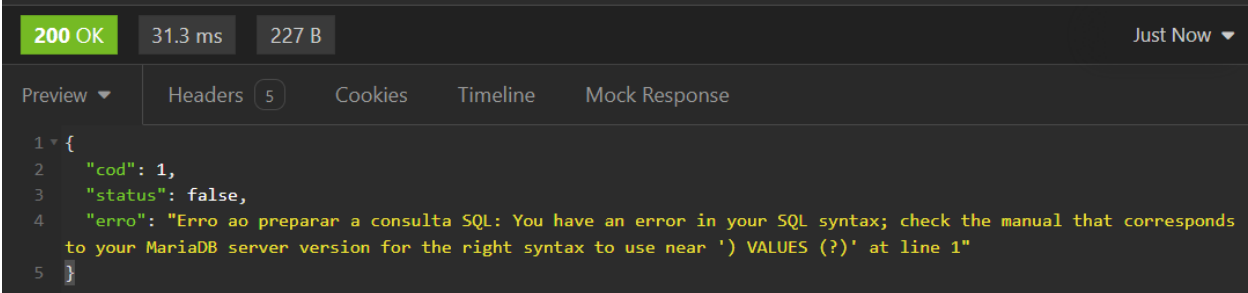
```
Preview ▾ Headers 5 Cookies Timeline Mock Response  
1 {  
2   "cod": 1,  
3   "erro": "Erro ao conectar ao banco de dados: Nenhuma conexão pôde ser feita porque a máquina de destino as  
4   recusou ativamente"
```

Aplicando aos métodos das classes

```
public function create(){
    try {
        // Define a consulta SQL para inserir um novo cargo
        $SQL = "INSERT INTO cargo (nomeCargo,) VALUES (?);";
        $prepareSQL = Banco::getConexao()->prepare($SQL); // Prepara a consulta
        // Verifica se a preparação da consulta foi bem-sucedida
        if (!$prepareSQL) {
            throw new Exception("Erro ao preparar a consulta SQL: " . $conexao->error);
        }
        // Define o parâmetro da consulta com o nome do cargo
        $prepareSQL->bind_param("s", $this->nomeCargo);
        $executou = $prepareSQL->execute(); // Executa a consulta
        // Verifica se a execução da consulta foi bem-sucedida
        if (!$executou) {
            throw new Exception("Erro ao executar a consulta SQL: " . $prepareSQL->error);
        }
        $idCadastrado = $conexao->insert_id; // Obtém o ID do cargo inserido
        // Define o ID do cargo na instância atual da classe
        $this->setIdCargo($idCadastrado);
    } catch (Exception $e) {
        // Trata a exceção
        $objResposta = new stdClass();
        $objResposta->cod = 1;
        $objResposta->status = false;
        $objResposta->erro = $e->getMessage();
        //echo json_encode($objResposta);
        die(json_encode($objResposta));
    }
    // Retorna se a operação foi executada com sucesso
    return $executou;
}
```

Verifique que é possível tratar dentro de cada operação de CRUD

Observe que há um erro de sintaxe no SQL



The screenshot shows a web browser's developer console. At the top, a green status bar indicates '200 OK' with a response time of '31.3 ms' and a size of '227 B'. Below this, the 'Preview' tab is active, displaying a JSON response. The response is an object with three properties: 'cod' (1), 'status' (false), and 'erro' (a detailed error message about SQL syntax). The error message states: 'Erro ao preparar a consulta SQL: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ') VALUES (?)' at line 1'. The JSON is formatted with line numbers 1 through 5 on the left.

```
1 {
2   "cod": 1,
3   "status": false,
4   "erro": "Erro ao preparar a consulta SQL: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ') VALUES (?)' at line 1"
5 }
```

Também é possível tratar no controle.

modelo

```
public function create(){
    dados
    $conexao = Banco::getConexao(); // Obtém a conexão com o banco de
    // Define a consulta SQL para inserir um novo cargo
    $SQL = "INSERT INTO cargo (nomeCargo,) VALUES (?);";
    $prepareSQL = $conexao->prepare($SQL); // Prepara a consulta
    // Verifica se a preparação da consulta foi bem-sucedida
    if (!$prepareSQL) {
        throw new Exception("Erro ao preparar a consulta SQL: " .
        $conexao->error);
    }
    // Define o parâmetro da consulta com o nome do cargo
    $prepareSQL->bind_param("s", $this->nomeCargo);
    $executou = $prepareSQL->execute(); // Executa a consulta
    // Verifica se a execução da consulta foi bem-sucedida
    if (!$executou) {
        throw new Exception("Erro ao executar a consulta SQL: " .
        $prepareSQL->error);
    }
    $idCadastrado = $conexao->insert_id; // Obtém o ID do cargo inserido
    // Define o ID do cargo na instância atual da classe
    $this->setIdCargo($idCadastrado);
    // Retorna se a operação foi executada com sucesso
    return $executou;
}
```

controle

```
try {
    // Verifica se a criação do novo cargo foi bem-sucedida
    $funcionario->create();
    $objResposta->cod = 4;
    $objResposta->status = true;
    $objResposta->msg = "cadastrado com sucesso";
    $objResposta->novoCargo = $funcionario;
} catch (Exception $e) {
    // Trata a exceção
    $objResposta = new stdClass();
    $objResposta->cod = 1;
    $objResposta->status = false;
    $objResposta->erro = $e->getMessage();
    //echo json_encode($objResposta);
    die(json_encode($objResposta));
}
```

Outras aplicações:

- Para soluções elegantes é possível substituir a cadeia de ifs que fazem a validação das regras de negócio por lançamentos de exceções.

controle

```
<?php
require_once ("modelo/Cargo.php");
header("Content-Type: application/json");
$objResposta = new stdClass(); // Cria um novo objeto para armazenar a resposta
$cargo = new Cargo(); // Cria um novo objeto da classe Cargo
$textoRecebido = file_get_contents("php://input"); // Obtém os dados enviados por meio de uma requisição POST em formato JSON
try {
    // Decodifica os dados JSON recebidos em um objeto PHP ou interrompe o script se o formato estiver incorreto
    $objJson = json_decode($textoRecebido);
    if ($objJson === null && json_last_error() !== JSON_ERROR_NONE) {
        throw new Exception('Formato JSON inválido');
    }
    $cargo->setNomeCargo($objJson->cargo->nomeCargo); // Define o nome do cargo recebido do JSON no objeto Cargo
    if ($cargo->getNomeCargo() == "") { // Verifica se o nome do cargo está vazio
        throw new Exception("O nome não pode ser vazio: ");
    }
    if (strlen($cargo->getNomeCargo()) < 3) {
        throw new Exception("o nome não pode ser menor do que 3 caracteres");
    }
    if (strlen($cargo->getNomeCargo()) > 32) {
        throw new Exception("o nome não pode ser maior do que 32 caracteres");
    }
    if ($cargo->isCargo() == true) {
        throw new Exception("Já existe um cargo cadastrado com o nome: " . $cargo->getNomeCargo());
    }
    $cargo->create(); // tenta criar um novo cargo
    $objResposta->status = true;
    $objResposta->msg = "cadastrado com sucesso";
    $objResposta->novoCargo = $cargo;
    header("HTTP/1.1 201");
} catch (Exception $e) {
    $objResposta->status = false;
    $objResposta->msg = $e->getMessage();
}
// Converte o objeto resposta em JSON e o imprime na saída
echo json_encode($objResposta);
```

