# exceções pré definidas

Anteriormente estudamos sobre exceções e vimos que elas são na verdade classes pré definidas, que implementam outras classes, e têm seus próprios parâmetros e métodos.

Portanto, agora iremos ver algumas dessas classes pré definidas, que servem para nos dar suporta ao tratamento de erros e exceções.

## Hierarquia de Exceções no PHP

O PHP possui um sistema estruturado de tratamento de erros e exceções. As exceções pré-definidas são classes que estendem diretamente ou indiretamente as classes principais:

* **Base das exceções**: Throwable é a interface implementada por todas as exceções e erros.

**O que a interface Throwable define?**

A interface **Throwable** define os seguintes métodos obrigatórios, que todas as suas implementações (como Exception e Error) devem conter:

1. **getMessage()**
2. **getCode()**
3. **getFile()**
4. **getLine()**
5. **getTrace()**
6. **getTraceAsString()**
7. **\_\_toString()**

## Organização de Classes: Herança e Subclasses

**O que são subclasses?**

**Subclasse**: Uma classe que **herda** de outra classe. A classe "pai" é chamada de **superclasse**, e a "filha" é a subclasse.

A subclasse **reutiliza** os métodos e propriedades da superclasse, mas pode **adicionar ou sobrescrever** comportamento específico.

Em PHP, a herança é declarada com a palavra-chave extends.

**1. Superclasse Principal: Throwable**

* Todas as classes de **exceção** e **erro** no PHP implementam a interface Throwable.

**2. Exception**

* É a classe base para exceções **controláveis**.
* **Subclasses de Exception:**
  + ErrorException
  + ClosedGeneratorException

**3. Error**

* É a classe base para **erros fatais** introduzida no PHP 7.
* Usada para representar problemas que normalmente causariam interrupções na execução.
* **Subclasses de Error:**
  + ArgumentCountError
  + ArithmeticError
    - **Subclasse de ArithmeticError:**
      * DivisionByZeroError
  + AssertionError
  + CompileError
    - **Subclasse de CompileError:**
      * ParseError
  + TypeError
  + ValueError
  + UnhandledMatchError
  + FiberError

Agora iremos ver os métodos e propriedades dessas classes.

Exception

### Propriedades

1. **$message** *(protected)*:  
   Armazena a mensagem de erro associada à exceção.
2. **$string** *(private)*:  
   Interno da classe. Não é acessível diretamente.
3. **$code** *(protected)*:  
   Código de erro associado à exceção.
4. **$file** *(protected)*:  
   O arquivo onde a exceção foi lançada.
5. **$line** *(protected)*:  
   A linha do arquivo onde a exceção foi lançada.
6. **$trace** *(private)*:  
   Um array que contém o rastreamento da pilha no momento em que a exceção foi lançada.
7. **$previous** *(private)*:  
   Uma referência à exceção anterior, útil para exceções encadeadas.

### métodos

**\_\_construct(string $message = "", int $code = 0, ?Throwable $previous = null)**   
*(public)*:  
Construtor para criar uma nova instância de Exception com mensagem, código e exceção anterior opcional.

**getMessage(): string** *(final public)*:  
Retorna a mensagem de erro da exceção.

**getPrevious(): ?Throwable** *(final public)*:  
Retorna a exceção anterior em caso de exceções encadeadas.

**getCode(): int** *(final public)*:  
Retorna o código de erro associado à exceção.

**getFile(): string** *(final public)*:  
Retorna o nome do arquivo onde a exceção foi lançada.

**getLine(): int** *(final public)*:  
Retorna o número da linha onde a exceção foi lançada.

**\_\_toString(): string** *(public)*:  
Retorna uma representação textual da exceção.

**\_\_clone(): void** *(private)*:  
Clonagem da exceção não é permitida, por isso é um método privado.

**getTrace(): array** *(final public)*:  
Retorna o rastreamento da pilha como um array.

**getTraceAsString(): string** *(final public)*:  
Retorna o rastreamento da pilha em forma de string.

ErrorException

### propriedades +

**$severity** *(protected)*:  
Nível de severidade do erro (por padrão, **E\_ERROR**). É usado para classificar o tipo de erro.

**+ :** propriedades herdades de Exception...

### métodos

**\_\_construct(  
string $message = "",   
int $code = 0,   
int $severity = E\_ERROR,   
?string $filename = null,   
?int $line = null,   
?Throwable $previous = null)** *(public)*:  
Constrói uma nova instância de ErrorException com parâmetros opcionais: mensagem, código, severidade, nome do arquivo, linha e exceção anterior.

**getSeverity(): int** *(final public)*:  
Retorna o nível de severidade do erro associado à exceção.

**+ :** métodos herdades de Exception...

ClosedGeneratorException

Métodos e propriedades todos herdados de Exception.

Eles foram adicionados a uma classe ClosedGeneratorException apenas para para identificar esse tipo de erro de maneira mais explícita, relacionados a operações em Generators fechados.

ERROR

### Propriedades

1. **$message** *(protected)*:  
   Mensagem descritiva do erro.
2. **$string** *(private)*:  
   Propriedade interna para representar o erro como string (uso interno do PHP).
3. **$code** *(protected)*:  
   Código numérico associado ao erro.
4. **$file** *(protected)*:  
   Nome do arquivo onde o erro foi gerado.
5. **$line** *(protected)*:  
   Número da linha no arquivo onde o erro foi gerado.
6. **$trace** *(private)*:  
   Array contendo o rastreamento da pilha (stack trace) no momento em que o erro ocorreu.
7. **$previous** *(private)*:  
   Referência para um erro ou exceção anterior, se houver encadeamento de erros/exceções.

### Métodos:

1. **\_\_construct(string $message = "", int $code = 0, ?Throwable $previous = null)** *(public)*:  
   Constrói uma nova instância de Error, permitindo a definição de uma mensagem, código e uma referência a um erro/exceção anterior.
2. **getMessage(): string** *(final public)*:  
   Retorna a mensagem associada ao erro.
3. **getPrevious(): ?Throwable** *(final public)*:  
   Retorna o erro ou exceção anterior, se houver encadeamento.
4. **getCode(): int** *(final public)*:  
   Retorna o código do erro.
5. **getFile(): string** *(final public)*:  
   Retorna o nome do arquivo onde o erro ocorreu.
6. **getLine(): int** *(final public)*:  
   Retorna o número da linha no arquivo onde o erro ocorreu.
7. **getTrace(): array** *(final public)*:  
   Retorna o rastreamento da pilha no momento do erro, em formato de array.
8. **getTraceAsString(): string** *(final public)*:  
   Retorna o rastreamento da pilha como uma string formatada.
9. **\_\_toString(): string** *(public)*:  
   Retorna uma representação textual detalhada do erro, incluindo mensagem, código, e rastreamento da pilha.
10. **\_\_clone(): void** *(private)*:  
    Impede que um objeto da classe Error seja clonado.

ArgumentCountError

### Propriedades e métodos

Todos as propriedades e métodos são herdados de Error e TypeError.

***Diferenciais da Classe ArgumentCountError:***

* Especializada para tratar erros relacionados ao número de argumentos fornecidos a funções e métodos.
* Ajuda a diferenciar erros de tipo (TypeError) de erros relacionados ao número de argumentos, proporcionando maior clareza e especificidade no tratamento de erros.
* É derivada de TypeError, que, por sua vez, herda de Error. Isso significa que ela também pode ser capturada em blocos catch que tratam TypeError ou Error.

ArithmecticError

### propriedades e métodos

Todos as propriedades e métodos são herdados de Error.

***Diferenciais da Classe ArithmeticError:***

* **Específica para erros aritméticos**:  
  A ArithmeticError é uma classe genérica para lidar com erros aritméticos. Subclasses como DivisionByZeroError fornecem maior especificidade.
* **Relacionada ao uso incorreto de cálculos**:  
  Exemplos incluem divisão por zero em inteiros ou operações que não suportam certos tipos de valores (ex.: números não finitos ou tipos não numéricos).

AssertionError

### propriedades e métodos

Todos as propriedades e métodos são herdados de Error.

***Diferenciais da Classe AssertionError:***

* **Ligada ao uso de assert**:  
  Esta classe é lançada automaticamente quando uma chamada para assert falha (se o modo de depuração estiver ativado). Por padrão, ela não precisa ser instanciada manualmente.
* **Exclusiva para validação interna**:  
  É útil para garantir que condições específicas do programa sejam atendidas durante a execução, especialmente em fases de depuração ou validação de lógica.

DivisionByZeroError

### propriedades e métodos

Todos as propriedades e métodos são herdados de Error.

***Diferenciais da Classe DivisionByZeroError:***

* **Especialização de Erros Matemáticos**:  
  Esta classe é usada especificamente para operações matemáticas inválidas que envolvem divisão por zero.
* **Automática pelo PHP**:  
  Não precisa ser instanciada manualmente em código comum. PHP automaticamente lança um DivisionByZeroError ao detectar uma divisão por zero, seja em inteiros ou floats.
* **Facilita o Tratamento Específico**:  
  Diferente de usar um ArithmeticError genérico, ela permite capturar e tratar divisões por zero de forma mais granular.

CompileError

### propriedades e métodos

Todos as propriedades e métodos são herdados de Error.

***Características da Classe CompileError:***

1. **Ocorrência Antes da Execução**:  
   Diferentemente de erros em tempo de execução, um CompileError é detectado antes que o código seja executado, geralmente devido a problemas de sintaxe ou erros estruturais no código.
2. **Não Pode Ser Capturado Diretamente**:  
   CompileError não pode ser tratado usando blocos try-catch, pois o script nem sequer será executado se um erro de compilação ocorrer.
3. **Erro Fatal**:  
   Sempre resulta na interrupção imediata do processamento do código.
4. **Relacionado ao Interpretador**:  
   É gerado diretamente pelo interpretador do PHP durante a análise do código.

ParseError

### propriedades e métodos

Todos as propriedades e métodos são herdados de CompileError.

***Características da Classe ParseError:***

1. **Erro de Sintaxe**:  
   ParseError é gerado quando o interpretador PHP detecta uma construção de código inválida, como uma falta de ponto e vírgula ou o uso incorreto de chaves.
2. **Ocorrência Antes da Execução**:  
   Assim como CompileError, os erros de análise são detectados antes de o script começar a ser executado. Isso significa que eles não podem ser capturados diretamente por blocos try-catch.
3. **Herdeira de CompileError**:  
   Como uma subclasse de CompileError, ela mantém todas as características e comportamentos dessa classe.
4. **Erro Fatal**:  
   Sempre resulta na interrupção imediata do script, impedindo que o código seja processado.

TypeError

### propriedades e métodos

Todos as propriedades e métodos são herdados de CompileError.

***Características da Classe TypeError:***

1. **Erro de Tipo**:  
   Lançado quando há uma incompatibilidade de tipo em:
   * Argumentos passados para funções ou métodos.
   * Tipos de retorno definidos.
   * Operações em variáveis tipadas.
2. **Exceção Fatal**:  
   Sempre interrompe a execução do script se não for tratada.
3. **Compatível com try-catch**:  
   Pode ser capturado e manipulado para evitar a interrupção do script.
4. **Parte do PHP 7+**:  
   Reflete o foco do PHP 7 e posteriores em tipos escaláveis e mais robustos.

ValueError

### propriedades e métodos

Todos as propriedades e métodos são herdados de Error.

***Características da Classe ValueError:***

1. **Foco em Valores Inválidos**:  
   Usada para indicar erros em que o valor fornecido não é aceitável para uma função ou operação, mesmo que o tipo esteja correto.
2. **Parte do PHP 8.0+**:  
   Introduzida para melhorar a diferenciação entre erros de tipo (TypeError) e erros lógicos de valor.
3. **Erros Não Tratáveis Automaticamente**:  
   Normalmente exige que o código verifique os valores antes de utilizá-los ou capture o erro em um bloco try-catch.
4. **Compatível com Tratamento de Exceções**:  
   Pode ser capturada com catch (ValueError $e).

UnhandledMatchError

### propriedades e métodos

Todos as propriedades e métodos são herdados de Error.

***Características da Classe UnhandledMatchError:***

1. **Relacionada ao match**:  
   Projetada exclusivamente para lidar com cenários em que um match não possui casos correspondentes e não há cláusula default.
2. **Parte do PHP 8.0+**:  
   Introduzida junto com o construtor match para melhorar o tratamento de erros e a semântica do código.
3. **Erro Não Tratável Automaticamente**:  
   Lançada automaticamente pelo PHP quando a situação de erro ocorre.
4. **Compatível com Tratamento de Exceções**:  
   Pode ser capturada em um bloco try-catch usando catch (UnhandledMatchError $e).

FiberError

### propriedades e métodos

Todos as propriedades e métodos são herdados de Error, mas com uma diferença no \_\_construtc.

**Métodos:**

1. **\_\_construct()** *(public)*:  
   Construtor da classe FiberError. Permite inicializar uma instância de FiberError.  
   **Nota**: Não aceita parâmetros diretamente.

***Características da Classe UnhandledMatchError:***

1. **Relacionada ao match**:  
   Projetada exclusivamente para lidar com cenários em que um match não possui casos correspondentes e não há cláusula default.
2. **Parte do PHP 8.0+**:  
   Introduzida junto com o construtor match para melhorar o tratamento de erros e a semântica do código.
3. **Erro Não Tratável Automaticamente**:  
   Lançada automaticamente pelo PHP quando a situação de erro ocorre.
4. **Compatível com Tratamento de Exceções**:  
   Pode ser capturada em um bloco try-catch usando catch (UnhandledMatchError $e).