Gebruik van Try-Catch-statements in C#

D.K.

24 mei 2023

Inleiding

- Foutafhandeling is een essentieel onderdeel van softwareontwikkeling.
- C# ‘try-catch‘-statement - afhandelen van uitzonderingen.
- In deze presentatie zullen we de basisprincipes van ‘try-catch‘-statements in C# verkennen en hoe ze kunnen worden gebruikt om fouten effectief te beheren.

Hoe werkt een Try-Catch-verklaring?

- Een ‘try-catch‘-verklaring bestaat uit twee hoofdblokken: ‘try‘ en ‘catch‘.
- Code die mogelijk uitzonderingen kan veroorzaken, wordt geplaatst in het ‘try‘-blok.
- Als er een uitzondering optreedt in het ‘try‘-blok, wordt de uitvoering onmiddellijk overgebracht naar het bijbehorende ‘catch‘-blok.
- Het ‘catch‘-blok bevat de code die wordt uitgevoerd om de uitzondering af te handelen en te herstellen.

Syntax

De syntax van een ‘try-catch‘-verklaring in C# is als volgt:

```
try
{
    // Code die mogelijk uitzonderingen veroorzaakt
}
catch (ExceptionType exception)
{
    // Code om de uitzondering af te handelen
}
```

Voorbeeld

Laten we eens kijken naar een eenvoudig voorbeeld van het gebruik van een ‘try-catch‘-verklaring:

```
try
{
    int x = 10;
    int y = 0;
    int result = x / y;
}
catch (DivideByZeroException ex)
{
    Console.WriteLine("Fout: Kan niet delen door nul.");
}
```

Soorten uitzonderingen

- C# biedt verschillende uitzonderingstypen die kunnen worden afgehandeld met ‘catch‘-blokken.
- Enkele veelvoorkomende uitzonderingstypen zijn:
  - `DivideByZeroException`: Gegooid wanneer een deling door nul wordt geprobeerd.
  - `ArgumentNullException`: Gegooid wanneer een argument een `null`-waarde heeft, terwijl dit niet is toegestaan.
    item `FileNotFoundException`: Gegooid wanneer een bestand niet kan worden gevonden.
  - `FormatException`: Gegooid wanneer er een onjuiste indeling is bij het converteren van gegevens.
- Het is ook mogelijk om aangepaste uitzonderingstypen te maken.

Meerdere catch-blokken

- U kunt meerdere ‘catch‘-blokken gebruiken om verschillende soorten uitzonderingen af te handelen.
- De ‘catch‘-blokken worden in volgorde geëvalueerd en het overeenkomstige blok voor het uitzonderingstype wordt uitgevoerd.
- U kunt de basisuitzonderingsklasse ‘Exception‘ gebruiken om alle typen uitzonderingen af te vangen die niet worden afgehandeld door specifieke ‘catch‘-blokken.

Voorbeeld met meerdere catch-blokken

```
try
{
    // Code die mogelijk uitzonderingen veroorzaakt
}
catch (DivideByZeroException ex)
{
    Console.WriteLine("Fout: Kan niet delen door nul.");
}
catch (ArgumentNullException ex)
{
    Console.WriteLine("Fout: Een argument heeft een null-wa
}
catch (Exception ex)
{
    Console.WriteLine("Fout: Er is een uitzondering opgetre
}
```

Het finally-blok

- Naast het ‘try‘- en ‘catch‘-blok kan een ‘try-catch‘-verklaring ook een ‘finally‘-blok bevatten.
- Het ‘finally‘-blok wordt altijd uitgevoerd, ongeacht of er een uitzondering is opgetreden of niet.
- Het wordt vaak gebruikt om opruimingscode te schrijven, zoals het sluiten van geopende bestanden of het vrijgeven van bronnen.

Voorbeeld met finally-blok

```
try
{
    // Code die mogelijk uitzonderingen veroorzaakt
}
catch (Exception ex)
{
    Console.WriteLine("Fout: Er is een uitzondering opgetre
}
finally
{
    // Code die altijd wordt uitgevoerd
    Console.WriteLine("De try-catch-finally-verklaring is v
}
```

Conclusie

- De ‘try-catch‘-verklaring in C# biedt een mechanisme om uitzonderingen af te vangen en ermee om te gaan.
- Met behulp van de ‘catch‘-blokken kunt u specifieke uitzonderingstypen afhandelen en foutmeldingen weergeven of herstelacties uitvoeren.
- Het ‘finally‘-blok wordt gebruikt voor het schrijven van opr uimingscode en wordt altijd uitgevoerd.
- Het begrijpen van het gebruik van ‘try-catch‘-statements helpt bij het schrijven van robuuste en fouttolerante code.

Vragen?

Bedankt voor uw aandacht! Zijn er vragen over het gebruik van ‘try-catch‘-statements in C#?