



C# Essentials

# Documentatie & Naamconventie

Sander De Puydt

**DE HOGESCHOOL  
MET HET NETWERK**

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt  
[www.pxl.be](http://www.pxl.be) - [www.pxl.be/facebook](https://www.pxl.be/facebook)

# Korte inhoud

- Richtlijnen
- XML
  - <summary>
  - <para>
  - <returns>
  - <param>
- Naamconventies



# Richtlijnen

- Documentatie dient voor verduidelijking van de code.
  1. Gebruik commentaar om complexe/verwarrende applicatie logica te verduidelijken.
  2. Gebruik XML tags in C# commentaar waar het kan.
  3. Indien de commentaar bij code groter is dan het blok code, dan kan je best de inhoud wat verkorten.
  4. Vraag je af wat een andere programmeur zich zou afvragen wanneer hij/haar je code voor het eerst ziet.



# Richtlijnen

- Traditioneel zou er commentaar zijn per bestand waarin wordt bijgehouden **wanneer** het bestand was **aangemaakt**, **wie** er aan **geschreven** heeft en een **korte uitleg**.

```
/// -----  
/// Author:          Sander De Puydt  
/// Create Date:     20/02/2002  
/// Description:      Documentatie demo project  
/// -----  
public partial class MainWindow : Window  
{  
    public MainWindow()  
    {  
        InitializeComponent();  
    }  
}
```



# Richtlijnen

- Wanneer je met versiebeheer werkt, dan wordt deze bestandsinformatie bewaart door de repository ( = opslagplaats van alle versies van het project).
  - Versiebeheer systemen:
    - Git
    - SVN



# Richtlijnen

- **Belangrijk:**

Voeg niet overal documentatie aan toe!

Wanneer de naam van de methode/klasse/variabele al genoeg uitleg geeft, dan is het niet nodig om je document te vullen met zinloze commentaar.



# Richtlijnen

- **Belangrijk:**

Voeg niet overal documentatie aan toe!

Het is **beter** om een **duidelijke** (en misschien langere) **naam** te gebruiken en geen commentaar, dan een korte, cryptische naam én commentaar.



# XML

- In C# kan XML structuur brengen in commentaar.
- XML comments worden getoond wanneer je op het element hoovert in VS.
- Documentatie commentaar wordt niet ondersteund op namespaces.

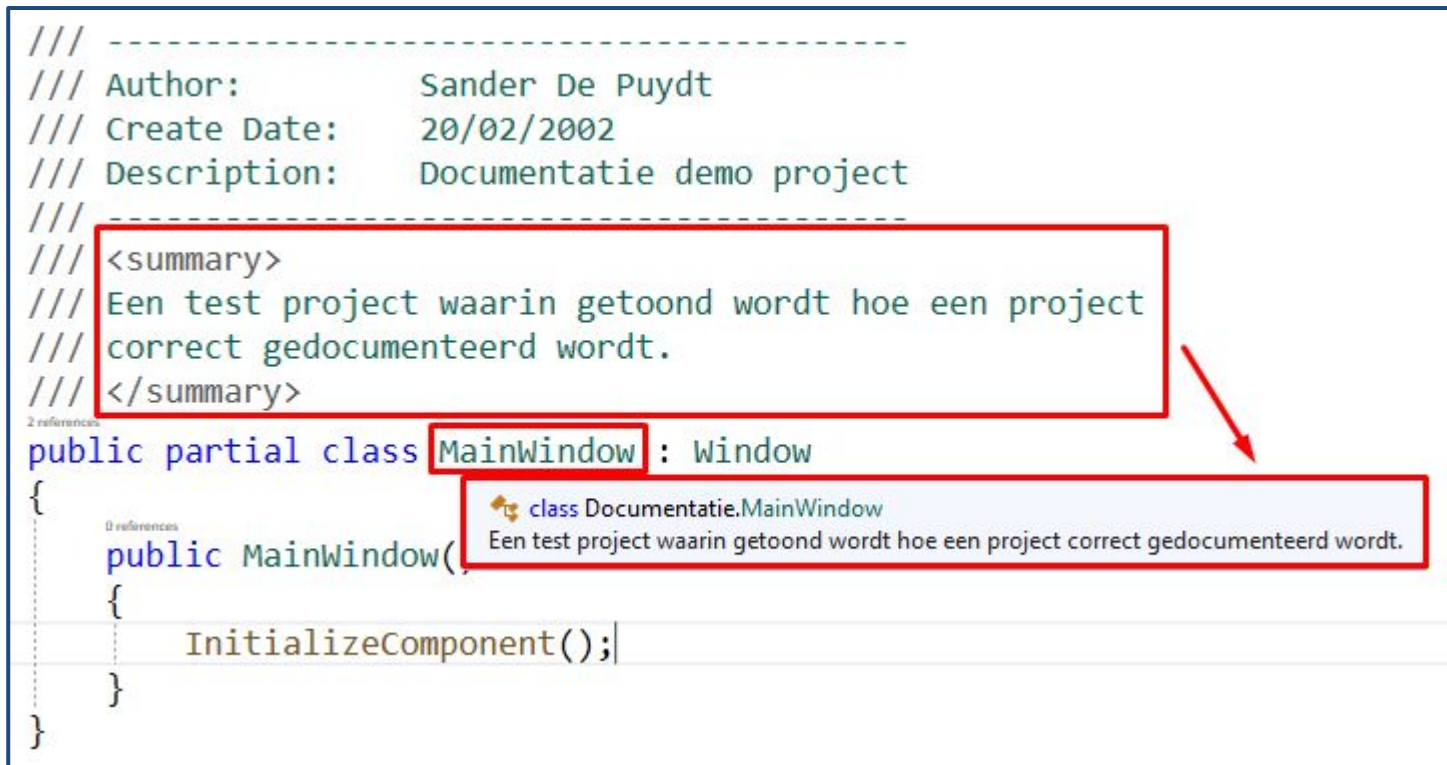




# <summary>

- De summary tag voegt een uitleg toe aan een element.

```
/// -----  
/// Author:      Sander De Puydt  
/// Create Date: 20/02/2002  
/// Description: Documentatie demo project  
/// -----  
/// <summary>  
/// Een test project waarin getoond wordt hoe een project  
/// correct gedocumenteerd wordt.  
/// </summary>  
2 references  
public partial class MainWindow : Window  
{  
    0 references  
    public MainWindow()  
    {  
        InitializeComponent();  
    }  
}
```



# <summary>

- De summary tag voegt een uitleg toe aan een element.
- Je kan de summary tag doen verschijnen door drie “/” te typen (///).



```
/// <summary>  
/// De DispatcherTimer controleert iedere minuut of  
/// de gebruiker geen hacks gebruikt om de applicatie  
/// te misbruiken.  
/// </summary>  
private DispatcherTimer timer;  
  
0 references  
public MainWindow()  
    (field) DispatcherTimer MainWindow.timer  
    De DispatcherTimer controleert iedere minuut of de gebruiker
```

# <para>

- De para tag laat je toe om paragrafen te schrijven in je commentaar.

```
/// <summary>
/// <para>Een test project waarin getoond wordt hoe een project
/// correct gedocumenteerd wordt.</para>
/// <para>Er wordt gefocust op XML documentatie om structuur te
/// brengen in commentaar.</para>
/// </summary>
```

2 references

```
public partial class MainWindow : Window
```

```
{
```

```
    /// <summary>
```

```
    /// <para>De Dispatcher
```

```
    /// de gebruiker geen tasks gebruikt om de applicatie
```

 class Documentatie.MainWindow

Een test project waarin getoond wordt hoe een project correct gedocumenteerd wordt.

Er wordt gefocust op XML documentatie om structuur te brengen in commentaar.



# <return>

- Bij het schrijven van een methode kan je de verwachte return value beschrijven met de <return> tag.

```
/// <summary>
/// Geeft een proces een threat score. Hoe hoger
/// de score is, hoe gevaarlijker het process is
/// voor de applicatie.
/// </summary>
/// <returns>Een threat socre tussen 0 (ongevaarlijk)
/// en 10 (gevaarlijk).</returns>
private double GenerateThreatScore(string processNaam)
{
    if (processM
    {
        return 1
    }
    Random rand = new Random();
    return rand.Next(0,11);
}
```

1 reference

**double MainWindow.GenerateThreatScore(string processNaam)**  
Geeft een proces een threat score. Hoe hoger de score is, hoe gevaarlijker het process is voor

Returns:  
Een threat socre tussen 0 (ongevaarlijk) en 10 (gevaarlijk).

# <param>

- <param name="parameterNaam"> Uitleg parameter </param>
- Geeft uitleg over de argumenten van een methode.

```
/// <summary>  
/// Geeft een proces een threat score. Hoe hoger  
/// de score is, hoe gevaarlijker het process is  
/// voor de applicatie.  
/// </summary>  
/// <param name="procesNaam">Naam van een actief  
/// proces op hetzelfde systeem als de applicatie</param>  
/// <returns>Een threat socre tussen 0 (ongevaarlijk)  
/// en 10 (gevaarlijk).</returns>
```

1 reference  
`private double GenerateThreatScore(string procesNaam)`

```
{  
    if (procesNaam.Equals("Hack"))  
    {  
        return 10;  
    }  
}
```

[0] (parameter) string procesNaam  
Naam van een actief proces op hetzelfde systeem als de applicatie

# Naamconventies

- Camel Case (camelCase): Eerste letter van het woord is klein en volgende woorden starten met een hoofdletter.
  - `parameterNaamVolgensCamelCase`
- Pascal Case (PascalCase): Eerste letter van elk woord start met een hoofdletter.
  - `ParameterNaamVolgensPascalCase`



# Naamconventies

- Zie syllabus bijlage 1 voor overzicht

Method	Pascal	<b>Berekening()</b> {}
Property	Pascal	public string <b>Naam</b> {get; set;}
Event	Pascal	public event EventHandler <b>Click</b>
public variable	Pascal	public int <b>Salaris</b>
private variable	Camel	private string <b>naam</b>
parameter	Camel	private int Converteer(string <b>waarde</b> ){}