Inhalt

[Async/Await 2](#_Toc32586066)

[Kein **Task.Wait** 2](#_Toc32586067)

[ConfigureAwait(false) 2](#_Toc32586068)

[Kein async void verwenden 2](#_Toc32586069)

[Tasks durchschleifen 2](#_Toc32586070)

# Async/Await

## Kein **Task.Wait**

Ein **Task.Wait** blockiert den Thread in der er aufgerufen wurde und läuft in einem neuen Thread. Bei einer Exception des Aufrufs kommt es zu einer **AggregateException** der die eigentliche Fehlermeldung beinhalted. Anstelle von **Task.Wait** soll man also **GetAwaiter().GetResult()** verwenden der die Originalexception abhandelt. Außerdem kann **Task.Wait** keine Rückgabewerte verarbeiten.

Dieses **GetAwaiter().GetResult()** soll aber nur sehr selten verwendet werden da das Problem mit zwei verwendeten Threads dennoch besteht.

|  |
| --- |
| //correct  \_clientId = Interaction.CallClassMethod("cmNT.Kassa", "GetOmanFirma").GetAwaiter().GetResult();  //wrong  Interaction.CallVoidClassMethod("cmNT.Kassa", "GetOmanFirma").Wait(); |

## ConfigureAwait(false)

Thread A ruft einen asynchronen Code mit dem Schlüsselwort await auf. Der asynchrone Code wird im freien Thread B verarbeitet, Thread A ist für andere Tätigkeiten offen. Ist der asynchrone Code fertig wird gewartet bis Thread A wieder frei ist, erst dann kann die Weiterverarbeitung weitergehen.

Ein await mit ConfigureAwait(false) wartet auch bis der asynchrone Code in Thread B fertig ist arbeitet in einem freien Thread weiter, kann wieder Thread A sein, kann auch Thread C sein.

Ein ConfigureAwaite(true) ist nicht nötig, das ist der Standard wenn ein Code asynchrone aufgerufen wird.

|  |
| --- |
| //correct  \_clientId = Interaction.CallClassMethod("cmNT.Kassa", "GetOmanFirma").GetAwaiter().GetResult();  //wrong  Interaction.CallVoidClassMethod("cmNT.Kassa", "GetOmanFirma").Wait(); |

## Kein async void verwenden

Async void Methoden sollen nie verwendet werden da nicht gewartet wird.

## Tasks durchschleifen

Eine asynchrone Methode, die nur eine weitere asynchrone Methode durchschleift soll ohne async und await auskommen, dann wird kein Kontextswitch gemacht.

|  |
| --- |
| //good  public static Task<string> GetServiceAreaId(string posId)  {  return Nt.Database.DB.Api.Pos.GetServiceAreaId(posId);  }  //bad  public static async Task<string> GetServiceAreaIdAsync(string posId)  {  return await Nt.Database.DB.Api.Pos.GetServiceAreaId(posId);  } |