

Installatie en gebruik Visual studio 2019 en GIT

OPO: C# OO Programming (OGI01S)

Johan Donné, Kristien Van Assche, Katja Verbeeck

Inhoud

| | |
|--|----|
| Installatie Visual Studio | 1 |
| Inleiding | 2 |
| Installatieprocedure | 2 |
| Aanmaak en configuratie van GIT Repository | 4 |
| 1. Aanmaak repository op server | 4 |
| 2. Aanmaak lokale kopie van de repository | 5 |
| 3. De directory structuur aanmaken binnen je repository..... | 7 |
| 4. Een test-solution aanmaken binnen de 'Blok1' directory | 7 |
| 5. Configuratie Automatische Build & Test ('Continuous Integration') | 10 |
| 6. Test van automatische Build & Test | 11 |
| Een 'solution' aanmaken voor je oplossing binnen een blok..... | 14 |
| Een project toevoegen aan een solution | 15 |
| Gebruik van GIT in Visual Studio 2019 | 17 |
| Richtlijnen en tips | 17 |
| Gebruik van GIT-features tijdens het programmeren..... | 18 |
| Extra content of resources toevoegen aan je Solution | 19 |
| Documentatie over je oplossing..... | 19 |
| Resources voor je toepassing..... | 19 |
| Gebruik van unittests in Visual Studio 2019..... | 21 |
| Bronvermelding in je programmacode | 23 |

Installatie Visual Studio

Indien Visual Studio 2019 reeds op je pc staat, kan je onmiddellijk overgaan naar het deel 'Aanmaak en configuratie van GIT repository'. Als je reeds een oudere versie gebruikt (2017, 2015...) kan je best overstappen naar de nieuwe versie: die is op heel wat punten verbeterd en handiger in gebruik.

Inleiding

Visual Studio is de ontwikkelingsomgeving van Microsoft voor de ontwikkeling van toepassingen op de verschillende Windows-platforms.

De meest recente versie (Visual Studio 2019) bestaat in een aantal versies:

- Visual Studio professional: voor individuele ontwikkelaars of kleine teams.
Als student van onze opleiding gratis te verkrijgen op 'Azure For Students' (zie opmerking hieronder).
- Visual Studio Enterprise: voor professionele ontwikkelaars, grote teams met extra features.
Als student van onze opleiding gratis te verkrijgen op 'Azure For Students'.
- Visual Studio Community: gratis versie (identiek aan 'professional') voor privé gebruik en KMO's. Te downloaden op <https://www.visualstudio.com> of via 'Azure For Students'.

Aanverwante producten:

- Visual Studio 'Code': editor voor ontwikkelaars, beschikbaar op Linux, OS X en Windows
- Visual Studio Online: extra online services voor collaboration & 'Application Lifecycle Management'

Opmerking: 'Azure for students'

'Azure For Students' is een platform van Microsoft waar je als student van onze opleiding gratis (ook professionele) software kan afhalen en gebruik kan maken van de cloudservices van Microsoft.

Je account activeren kan je via <https://azure.microsoft.com/be-nl/free/students/> (gebruik daarbij je mailaccount van de hogeschool).

Daarna krijg je toegang tot de software via <https://azureforeducation.microsoft.com/devtools>.

Installatieprocedure

Voor de opgaven van het vak C# OO programmeren, gebruik je minimaal de 'Professional' of 'Community' versie. We raden de Enterprise versie aan vanwege de extra features.

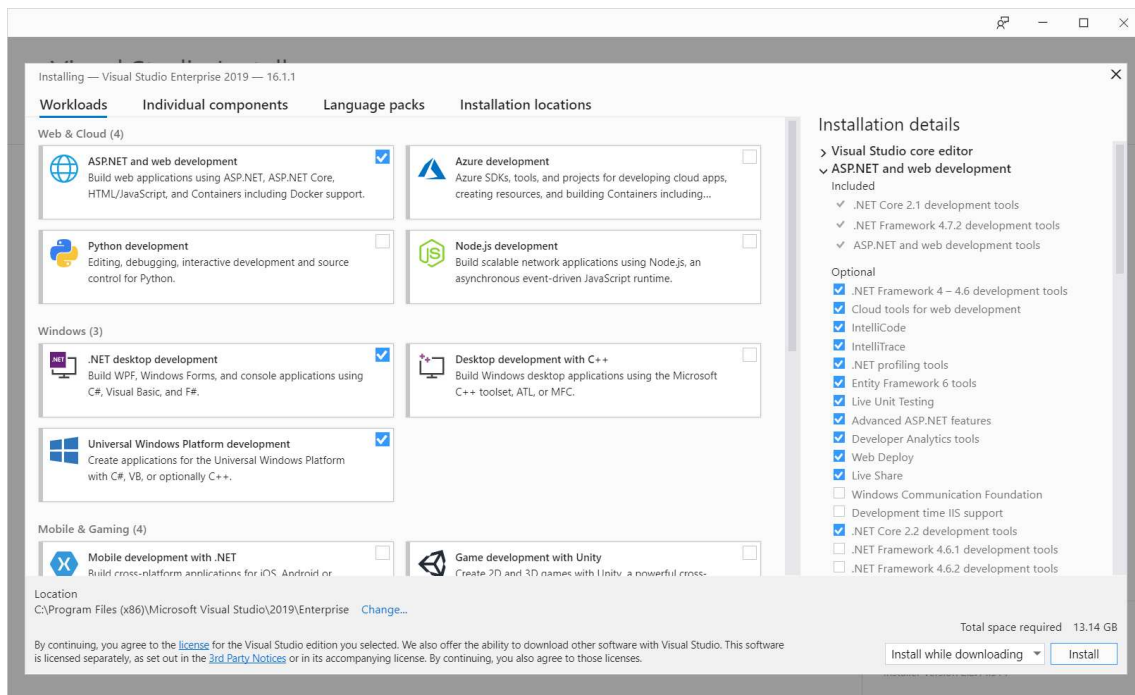
Opmerking:

Wanneer je Visual studio afhaalt van de 'Azure For Students' website, zal je ook een activatiekey krijgen **die je moet intikken na de installatie van het pakket**. Als je dit niet doet, werk je in 'Trial' mode en zal je na een tijd de melding krijgen dat je proefperiode afgelopen is.

Het is zo dat Visual Studio 2019 een licentie heeft per gebruiker (en niet per toestel). Je mag dit dus perfect legaal met dezelfde sleutel op meerdere toestellen installeren (bv. desktop en laptop) zolang dit enkel voor gebruik door jou persoonlijk bedoeld is.

Het bestand dat je downloadt is een 'installer' die dan achteraf de nodige componenten downloadt en installeert. Als je de installatie opstart, zal je kunnen kiezen welke features je wil installeren.

Zoals je op volgende pagina ziet kan je dat doen via 'Workloads' (scenario's voor een specifiek soort softwareontwikkeling) of door afzonderlijke componenten zelf te selecteren.



Je moet minimaal de workload ‘**.Net desktop development**’ selecteren.

Afhankelijk van je individuele situatie en voorkeur kunnen ook volgende workloads nuttig zijn:

- **.Net Core cross-platform development:** .Net Core is de nieuwe versie van het .Net Framework met enkele belangrijke voordelen. De code is efficiënter, sneller en kleiner. Het kan gebruikt worden voor toepassingen op Windows, Linux, IOS. Vanaf .Net core versie 3.0 kan je er ook WinForms- en WPF-toepassingen mee ontwikkelen (die dan enkel op Windows uitgevoerd kunnen worden).
- **Universal Windows Platform development:** voor het schrijven van de modernere ‘UWP’ toepassingen en o.a. ‘IOT’ toepassingen voor Windows 10 ‘IOT’ Core.
- **ASP.Net and web development:** voor het ontwikkelen van web-toepassingen op het .Net platform (ASP.Net, MVC...). Wordt gebruikt in het ICT keuzetraject Web & Mobile in de 3^e fase van de opleiding.
- **Python Development:** voor wie vanuit Visual Studio toepassingen wil schrijven in Python.
- **Data storage and processing:** voor wie toepassingen schrijft die gebruik maken van gegevensbanken. Dat zal voor alle ICT-studenten gebruikt worden in semester 2 van de 2^e fase.
- **Game development with Unity:** 'Unity' is een game engine voor de ontwikkeling van grafische spellen in C#.
- ...

Bij de individuele componenten moet je zeker het volgende selecteren (als dat nog niet vanzelf gebeurd is):

- **Git for Windows:** voor integratie met onze GIT-server (opgelet: dit is iets anders dan de ‘GitHub Extension for Visual Studio’. Die heb je voor dit vak niet nodig).
- **.Net Framework 4.7.2 SDK**
- **.Net Framework 4.7.2 targeting pack**

Volgende componenten kunnen ook nuttig zijn:

- Class designer
- Code Map
- GitHub extension for Visual Studio (als je eigen repositories op GitHub gebruikt).
- Live Share

Als je achteraf nog componenten of workloads wil toevoegen of verwijderen, kan je gewoon de installer opnieuw starten en kiezen voor 'Modify', waarna je hetzelfde selectievenster opnieuw te zien krijgt.

Als je .Net versie 4.8 of .Net Core 3.0 wil gebruiken (aangeraden!) moet je die zelf toevoegen door het 'developer pack' daarvoor te installeren vanaf:

- <https://dotnet.microsoft.com/download/dotnet-framework>
- <https://dotnet.microsoft.com/download/dotnet-core>

Als je Visual Studio 2019 de eerste keer opstart zal het pakket vragen of je je wil aanmelden met een Windows 'Live' account (bv. je Odissee-mailaccount). Dit is absoluut niet verplicht, maar het laat je toe om je instellingen voor Visual Studio te synchroniseren tussen alle toestellen waarop je aangemeld bent met dezelfde 'Live' account.

Verder zal je bovenaan het Visual studio venster ook altijd volgende icoon kunnen zien:



Door op dat icoontje te klikken kan je feedback geven aan Microsoft over Visual Studio (of eventuele bugs rapporteren).

Onderaan kan je soms ook volgend icoon zien:



Dit geeft aan dat er berichten zijn in verband met eventuele updates van onderdelen van Visual Studio. Het is een goede gewoonte om minstens de updates van Visual Studio zelf uit te voeren. Bij het schrijven van deze nota's is het versienummer '16.1.1'.

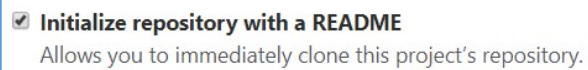
Aanmaak en configuratie van GIT Repository

Voor C# OO programming en de opdrachten daarin gebruiken je één GIT repository waarbinnen je per blok een afzonderlijke directory zal aanmaken.

1. Aanmaak repository op server

- Open de pagina <https://git.ikdoeict.be>
- Meld je aan met je schoolaccount

- Maak een nieuwe repository aan (op onze GIT-server een 'project' genoemd) met 'New project'.
- Als naam gebruik je '1920-CSharpOO-<voornaam><achternaam>' (bv. '1920-CSharpOO-PietPienter'). Je kiest voor een 'private' project.
Vink onderaan de optie aan om je repository te initialiseren:

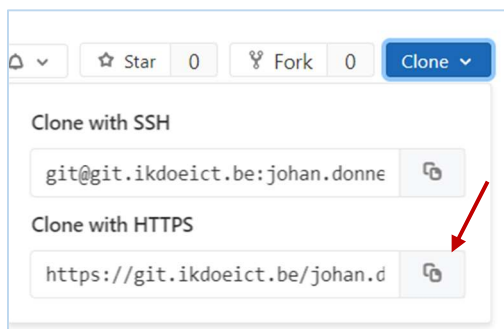


Klik op 'Create Project'

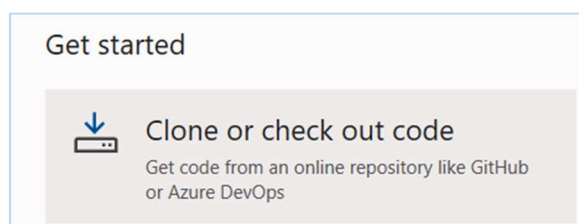
- Je selecteert in het menu links onder 'Settings' de optie 'members' en voegt daar onderstaande docenten toe met 'maintainer' als 'role permission':
johan.donne@odisee.be, kristien.vanassche@odisee.be, katja.verbeeck@odisee.be

2. Aanmaak lokale kopie van de repository

1. Op de projectpagina voor je repository open je de dropdown voor 'Clone' en selecteert daar de methode die je wil gebruiken om je lokale kopie te synchroniseren met de server. Daarvoor kopieer je de URL voor de gewenste methode naar je klemmenbord:



- https: de eenvoudigste methode (ze vraagt geen verdere configuratie, maar is iets minder efficiënt).
 - SSH: de meest efficiënte methode, maar die vraagt wat extra configuratie op je PC (aanmaken van een public/private keypair en registratie van de public key op de GIT-server).
2. Bij het opstarten van Visual Studio krijg je onder 'get started' de mogelijkheid om een 'clone' te maken van een GIT-repository:

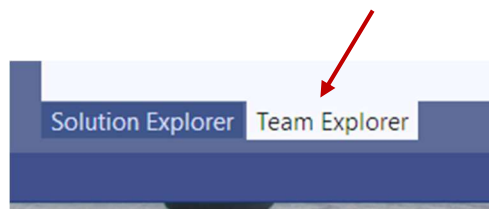


Geef vervolgens de URL van je server-repository in (bv. 'https://git.ikdoeict/...'), samen met een directory waar je de lokale kopie van je repository wil plaatsen (noem die bv. 'Git\1920-CsharpOO') en klik op 'Clone'.

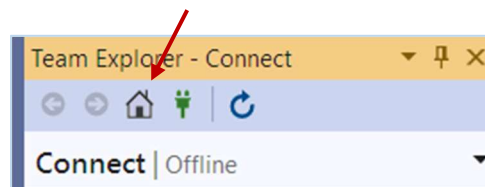


Je kan ook een clone maken als je Visual Studio al opgestart had. Dit doe je via 'Team Explorer'

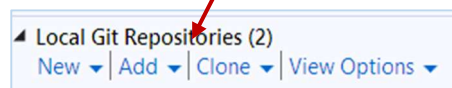
- Selecteer rechts onderaan 'Team Explorer' rechts onderaan



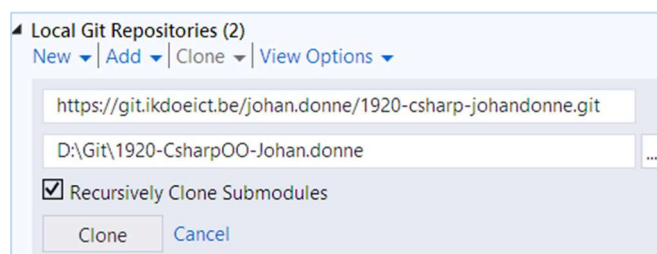
- Selecteer bovenaan in de 'Team Explorer' de connection manager:



- Klik onderaan bij 'Local GIT Repositories' op 'Clone'



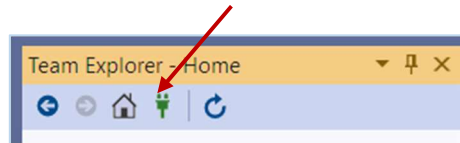
en geef vervolgens de URL van je server-repository in (bv. 'https://git.ikdoeict/...'), samen met een directory waar je de lokale kopie van je repository wil plaatsen (noem die bv. 'Git\1920-CsharpOO').



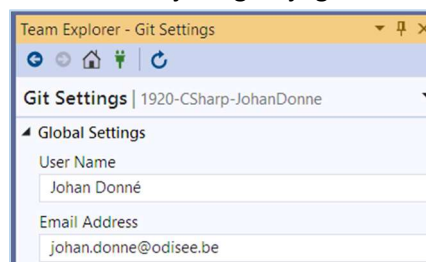
Klik dan op 'Clone' (Als je https gebruikt zal je de eerste keer je gebruikersnaam – **zonder '@student.odisee.be'** - en wachtwoord voor GIT moeten opgeven).

3. Configuratie van je GIT-identity in Visual Studio

Klik bovenaan in de 'Team Explorer' op de 'Home' button



Klik op 'Settings', geef bij de globale configuratie je naam in bij 'User Name' en je mailadres (van de school), de andere velden laat je ongewijzigd:

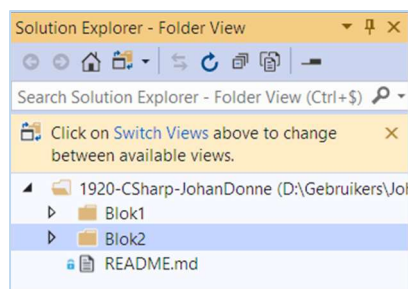


3. De directory structuur aanmaken binnen je repository

Binnen je repository zal je voor elk blok een directory moeten aanmaken:

Klik bovenaan in de 'Team Explorer' op de 'Home' button en klik onder 'Solutions' op 'Show Folder View'.

Klik met je rechtermuisknop op de naam van je repository, selecteer 'Add' – 'New Folder' en noem die map 'Blok1'. Doe hetzelfde voor 'Blok2'.



4. Een test-solution aanmaken binnen de 'Blok1' directory

Bij wijze van test zullen we een eenvoudige console-toepassing aanmaken binnen de directory voor blok 1:

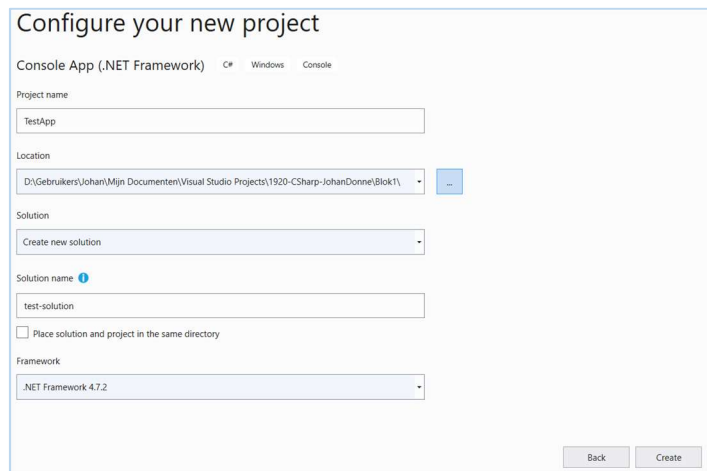
Selecteer in de 'connection manager' je nieuwe lokale repository door er onderaan op te dubbelklikken, en klik daarna op 'New'.

Je krijgt dan een 'Wizard' te zien voor de aanmaak van een nieuwe solution.

Je selecteert daar 'Console App (.NET Framework)' en klikt op 'Next'.

Je geeft een geschikte naam in (bv. 'Test'), **en selecteert de juiste plaats (de map 'Blok1') voor je test-solution via 'Browse'.**

Je stelt de versie van het framework waarvoor je een toepassing wil aanmaken in op '4.7.2'



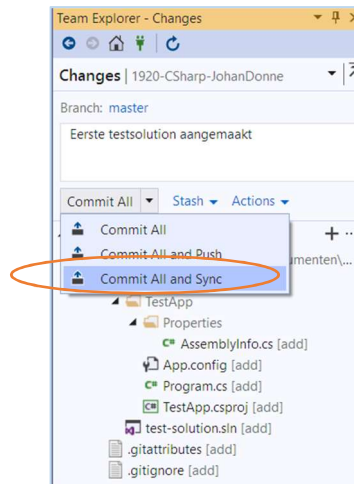
Wanneer je op 'Create' klikt wordt je solution aangemaakt en krijg je de (bijna lege) code te zien in 'Program.cs'.

Op de statusbalk onderaan krijg je de melding dat er wijzigingen zijn gebeurd in je repository (het getal 7 naast het potloodje):



Als je op dat icoontje klikt, kom je in het 'Changes'-menu van de team explorer en zie je daar welke bestanden gewijzigd zijn bij de aanmaak van je solution.

Om die wijzigingen te registreren in GIT en te synchroniseren geef je in het gele veld een 'commit message' in (bv. 'eerste testsolution aangemaakt') en selecteer je daaronder dan 'Commit All and Sync':



Je code zal nu lokaal in je GIT-repository geregistreerd worden en daarna ook naar de GIT-server gekopieerd worden.

- Opmerking: in dat drop down menu zie je drie mogelijkheden:
 - 'Commit All': registreert de wijzigingen enkel in je lokale repository (op jouw PC), maar synchroniseert ze niet naar de GIT-server.
 - 'Commit All and Push': registreert de wijzigingen naar je lokale repository en kopieert ze naar de GIT-server (eventuele wijzigingen die daar vanop een ander toestel zijn gebeurd, gaan verloren).
 - 'Commit All and Sync': registreert de wijzigingen naar je lokale repository en synchroniseert ze met de GIT-server. Lokale wijzigingen worden gekopieerd naar de server en eventuele wijzigingen die daar vanop een ander toestel zijn gebeurd, worden naar je lokale repository gekopieerd. In geval van conflicten worden die gemeld en kan je zelf beslissen welke versie je houdt.

Als je vanop meerdere toestellen aan hetzelfde project werkt (bv. desktop, laptop, op school) moet je wel op elk toestel een lokale repository maken door een 'clone' actie uit te voeren. Daarna zorg je gewoon regelmatig voor een 'Commit All and Sync' van op het toestel waar je wijzigingen doet, en op de andere toestellen voer je na het openen van je solution in Team Explorer een 'Sync' uit die ervoor zorgt dat je lokale versie gelijk wordt aan die op de server.

- Je kan verifiëren dat je testsolution ook op de GIT-server in je repository staat door via de webinterface op <https://git.ikdoeict.be> je repository (project) te openen. Je zal dan een link zien staan naar je bestanden en ook de commit message die je in Visual Studio ingegeven hebt:

Johan Donné > 1920-CSharp-JohanDonne > Details

1

1920-CSharp-JohanDonne

Project ID: 7346

🔒

☆ Star 0

🍴 Fork 0

📄 Clone

Add license

→ 2 Commits

🌿 1 Branch

🏷️ 0 Tags

📄 102 KB Files

Auto DevOps
 It will automatically build, test, and deploy your application based on a predefined CI/CD configuration.
[Learn more in the Auto DevOps documentation](#)

master

1920-csharp-johandonne /

+

+

History

🔍 Find file

Web IDE

🔗

Eerste testsolution aangemaakt
 Johan Donné authored just now

1536deeb

🔗

📖 README

📄 Add CHANGELOG

📄 Add CONTRIBUTING

📄 Add Kubernetes cluster

📄 Set up CI/CD

🔒 Security Dashboard

| Name | Last commit | Last update |
|-----------------------|----------------------------------|-------------|
| 📁 Blok1/test-solution | Eerste testsolution aangemaakt | just now |
| 📄 nitattributes | Eerste testenitration aangemaakt | just now |

Je configuratie is nu klaar om vanuit Visual Studio solutions en projecten aan te maken binnen je repository en geïntegreerd met GIT te werken.

5. Configuratie Automatische Build & Test ('Continuous Integration')

In praktijk gebeurt het soms dat een oplossing goed werkt op jouw computer, maar dat er problemen opduiken wanneer die op een andere computer uitgevoerd wordt. Dat zal bijvoorbeeld het geval zijn als je in je toepassing werkt met bestandspaden die verwijzen naar jouw computer. Of ook als je bibliotheken gebruikt die wel bij jou lokaal zijn geïnstalleerd, maar niet op de andere computer.

Het is nochtans zo dat de evaluatie van je oplossingen zal gebeuren op de computers van de docenten of op onze servers.

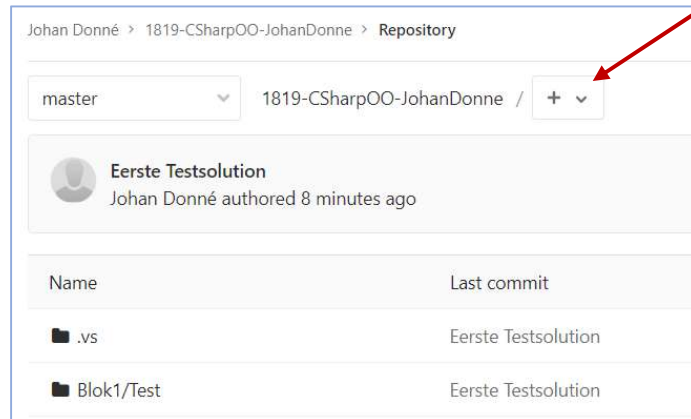
We willen vermijden dat je door dergelijke onverwachte problemen een slechte score krijgt. Daarom hebben we een mechanisme opgezet dat het volgende doet:

- Na elke 'Commit All and Sync' wordt je code op onze servers gecompileerd ('Build') en worden eventuele aanwezige unittests uitgevoerd.
- In de webinterface van de GIT-server kan je het resultaat van die Build & Test zien.
- Als daar de status 'Passed' verschijnt kon je oplossing zonder problemen gecompileerd worden op onze servers (dat is een vereiste om een score te krijgen).
- In het rapport dat voor die 'Build & Test' aangemaakt is, zie je het resultaat van eventuele unittests op onze servers en details van eventuele Build-errors.

Op die manier kan je eventuele problemen met je oplossing op tijd detecteren.

Er is echter wel nog een kleine configuratie nodig om die automatische 'Build & Test' mogelijk te maken:

- In de webinterface van de GIT-server ga je naar de bestanden van je project (door in het bovenstaande scherm op 'Files' te klikken).
- Je geeft aan dat je een bestand wil toevoegen (dat moet in de root van je repository gebeuren) door op de '+' te klikken:



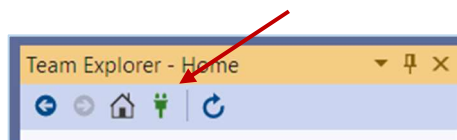
- Je selecteert 'Upload file' en selecteert het bestand '.gitlab-ci.yml' dat je op Toledo voor dit vak kan vinden (let er op dat het puntje vooraan in de bestandsnaam blijft staan. Chrome heeft de vervelende gewoonte dat punt te verwijderen. Voeg het eventueel zelf weer toe na een download met Chrome).
- Als commit message geef je bijvoorbeeld 'Add CI configuration file' in
- Je klikt onderaan op 'upload'.

Opmerking: dat bestand zal later ook in je lokale repository terecht komen. Gelieve dat niet te wijzigen, anders zal de automatische 'Build & Test' mogelijk niet meer werken.

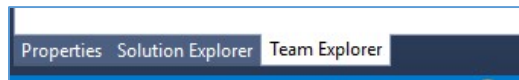
6. Test van automatische Build & Test

Na het doorlopen van voorgaande stappen zou je configuratie helemaal in orde moeten zijn. Je kan dit verifiëren op volgende manier:

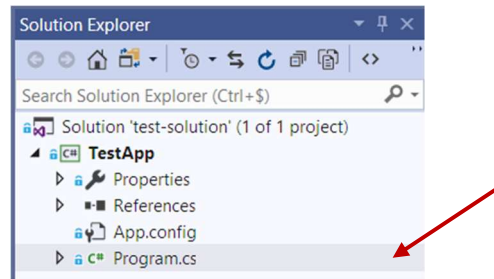
- Wijzig je testsolution in Visual Studio:
 - Start Visual Studio
 - Selecteer bovenaan in de 'Team Explorer' de connection manager:



- Dubbelklik op je repository.
- Dubbelklik op 'Test.sln'.
- Selecteer onderaan de 'Solution Explorer':



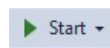
- Dubbelklik daarin op 'program.cs':



- Vul de (lege) 'Main'-methode aan naar volgende code:

```
static void Main(string[] args)
{
    Console.WriteLine("Hello World");
    Console.ReadLine();
}
```

- Klik bovenaan op 'start':



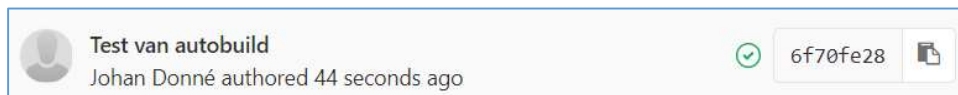
- Je toepassing wordt nu gecompileerd en uitgevoerd (als je geen tikfouten gemaakt hebt).
- Klik onderaan op het symbooltje voor de GIT-changes:



- Geef een commit-message in (bijvoorbeeld: 'test van Autobuild') en selecteer 'Commit All and Sync'. Bevestig – indien gevraagd - dat de gewijzigde bestanden opgeslagen moeten worden.

- Verifieer de build op de GIT-server.

- Open <https://git.ikdoeict.be>
- Selecteer de repository die je net aangemaakt hebt (mogelijk is er maar eentje).
- Daar zou nu een berichtje moeten staan over de 'commit' (soms ook een 'Merge') die je net gedaan hebt met daarbij een groene vinkje als alles goed ging:



- Als je op dat groene vinkje klikt, dat in het volgende scherm nog een keer doet (om het even welk vinkje) en tenslotte op 'job1', krijg je het resultaat van de build op onze servers te zien:

```
Running with gitlab-ci-multi-runner 9.5.0 (413da38) on CSharp-SharedBuildAndTest (4a462780)
Using Shell executor...
Running on WIN-NUJ6D85HP10...
Cloning repository...
Cloning into 'R://builds/4a462780/0/johan.donne/1920-CSharpOO-JohanDonne'...
Checking out 6f70fe28 as master...
Skipping Git submodules setup
$ CSharpOOTest

C# testrunner ver. 2.0.6597.17758 - 13/08/2018 23:14:29
Author: Johan Donne

Repository: 1920-CSharpOO-JohanDonne
userName: johan.donne

Solutions to process: Test.sln

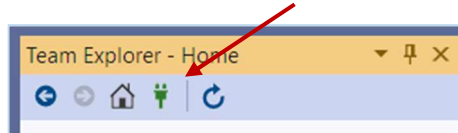
Restoring Nuget packages for Test.sln
MSBuild auto-detection: using msbuild version '15.7.179.6572' from 'C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\MSBuild\15.0\bin'.
Nothing to do. None of the projects in this solution specify any packages for NuGet to restore.

Building solution Test.sln
Microsoft (R) Build Engine version 15.7.179.6572 for .NET Framework
Copyright (C) Microsoft Corporation. All rights reserved.

    Test -> R:\builds\4a462780\0\johan.donne\1819-CSharpOO-JohanDonne\Blok1\Test\Test\bin\Debug\Test.exe
Build succeeded.
    0 Warning(s)
    0 Error(s)
Time Elapsed 00:00:00.66
Time taken for build: 3117msec
Executing tests for solution Test.sln
Testrunner finished without build-errors.
Job succeeded
```

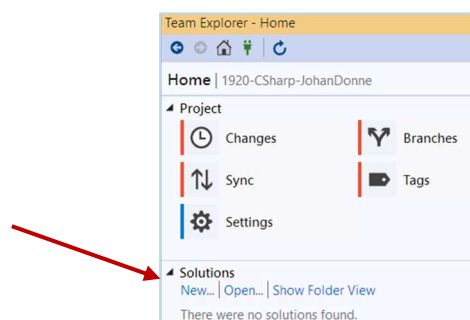
Een 'solution' aanmaken voor je oplossing binnen een blok

1. Start Visual Studio
2. Selecteer bovenaan in de 'Team Explorer' de connection manager

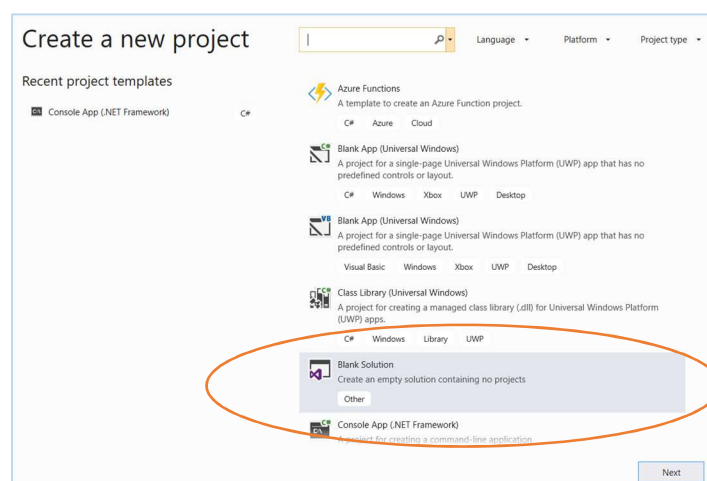


en dubbelklik op je repository.

3. Klik op 'New':



4. We zullen nu een solution aanmaken waarbinnen je meerdere projecten en documenten kunt onderbrengen. Selecteer als projecttype 'Blank Solution'.



5. Kies een geschikte naam (in dit voorbeeld: 'Oplossing Blok 1') en geef aan waar je solution moet komen (directory Blok1 onder je Repository):

Configure your new project

Blank Solution Other

Project name

Oplossing blok 1

Location

D:\Gebruikers\Johan\Mijn Documenten\Visual Studio Projects\1920-CSharp-JohanDonne\Blok1\

Solution

Create new solution

Solution name ⓘ

Oplossing blok 1

Back Create

Laat de andere velden ongewijzigd en bevestig je keuze met 'Create'.
Het is een goede gewoonte hierna al onmiddellijk een eerste 'Commit All & Sync' te doen.

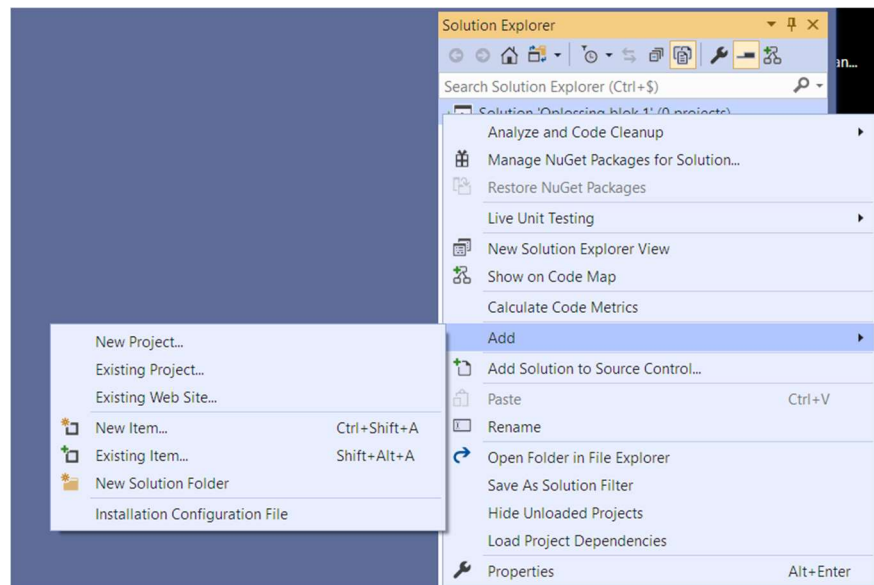
Een project toevoegen aan een solution

Aan de solution voor de oplossing voor je blokopgave, zal je minstens één project moeten toevoegen met de code van je oplossing. In praktijk is er echter een grote kans dat je binnen die solution meerdere projecten wil beheren. Zo zou je bijvoorbeeld een hoofdprogramma kunnen maken, met daarnaast nog een afzonderlijke 'Class library' waarin een aantal ondersteunende klassen gedeclareerd zijn voor gebruik vanuit meerdere programma's of andere bibliotheken.

Soms wil je ook verschillende versies van een toepassing (console, Windows Forms, WPF) samen in één solution.

Het is achteraf mogelijk om aan een bestaande solution een project toe te voegen:

1. Klik in de Solution Explorer met de rechtermuisknop op de naam van de **solution** (en selecteer 'Add' – 'New project...')



2. In het volgende venster selecteer je de geschikte template. Je kan geen locatie meer kiezen omdat het nieuwe project in de bestaande solution zal opgenomen worden.

Gebruik van GIT in Visual Studio 2019

Richtlijnen en tips

Het gebruik van GIT is helemaal geïntegreerd in de Visual studio omgeving. Hierbij toch enkele richtlijnen en tips.

- Als je Visual Studio opstart om toepassingen te schrijven voor je opleiding (C# OO Programming, Applied Programming, Project 1, C# Programming Techniques), ga je best naar de 'Team Explorer' en selecteert daar de 'connection manager'.

Daarin selecteer je onderaan de repository voor het vak waaraan je wil werken (bijvoorbeeld '1920-CSharpOO...'). Je krijgt dan een lijstje te zien van de verschillende solutions die binnen die repository bestaan. Je kiest uiteraard de solution waaraan je wil werken.

- Als je vanop verschillende toestellen aan dezelfde toepassing werkt (desktop, laptop, schoolcomputer) dan zorg je er best voor dat je allereerst je lokale repository synchroniseert met de remote versie op de GITLAB-server. Dit doe je door in '**Team Explorer**' onder '**Home**' '**Sync**' te selecteren en daar dan '**Sync**' te selecteren.

Tijdens het ontwikkelen kan je best op strategische momenten een 'Commit All and Sync' doen ('Team Explorer' – 'Home' - 'Changes' – 'Commit All and Sync').

Je moet daarbij een commentaartekst opgeven die de status van je project beschrijft.

Met die 'commit' registreert GIT alle wijzigingen in je lokale repository en zal die dan door kopiëren naar de centrale server. Als je in team werkt aan dezelfde toepassing (bv. in het vak projecten) kan het zijn dat iemand anders op een andere plaats in de toepassing veranderingen heeft aangebracht. Die zullen ook meteen gekopieerd worden naar je lokale versie.

- Je doet dit best zodra je een zinvolle en logisch samenhangende aanpassing aan je code afgewerkt hebt:
 - wanneer je de declaratie van een klasse hebt afgewerkt
 - wanneer je een methode hebt afgewerkt
 - je een fout gevonden en opgelost hebt
 - wanneer je een betekenisvolle aanpassing gedaan hebt aan de user interface (layout gewijzigd, componenten toegevoegd of weggelaten)
 - op elk moment waarvan je het nuttig vindt om achteraf terug te kunnen keren naar de toestand van dat moment.
- Een nuttige blogpost hierover: <https://www.freshconsulting.com/atomic-commits/>
- Je doet dit zeker voordat je Visual Studio afsluit!
- Zorg ervoor dat je enkel code commit en synchroniseert die zonder fouten kan builden!

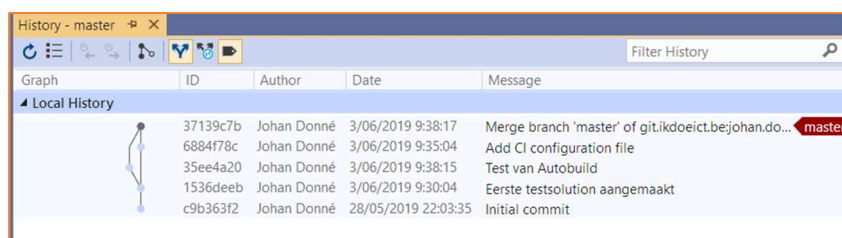
Gebruik van GIT-features tijdens het programmeren

Het gebruik van GIT levert een aantal voordelen op:

- Je kan op een betrouwbare manier van verschillende toestellen (en in team) aan een toepassing kan werken.
- Je hebt steeds een reservekopie op onze GITLAB-server staan.
- Een afzonderlijke upload naar Toledo is niet meer nodig (en kan je dus niet vergeten)

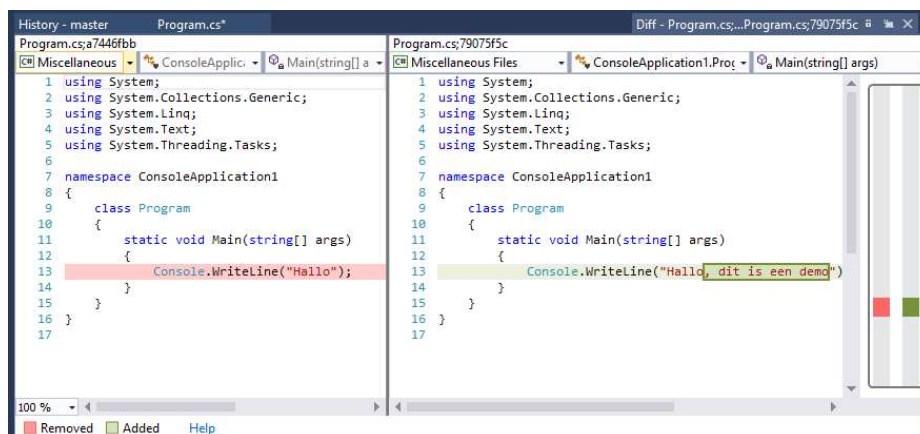
Daarnaast zijn er ook nog een aantal handige features die je kan gebruiken als je aan het programmeren bent:

- Je kan steeds de geschiedenis en details van vorige 'Commits' opvragen: In Team Explorer kies je binnen 'Changes' of 'Sync' onder 'Actions' de optie 'View History...':



Hier kan je zien wie de commit gedaan heeft, wanneer dat gebeurde en welk commentaar er meegegeven werd.

- Als je een specifieke Commit selecteert, kan je in Team Explorer onder 'Changes' zien welke veranderingen er in die Commit allemaal gebeurd zijn door te klikken op een gewijzigd bestand. Dan worden de vorige en nieuwe versie naast elkaar getoond met de wijzigingen:



- Ook in het code-venster zelf vind je met de rechtermuisknop het context menu 'Source Control'. Daarin kan je snel alle wijzigingen sinds de vorige commit ongedaan maken, de commit history opvragen, alle wijzigingen sinds de vorige commit tonen in de code, en snel een nieuwe commit doen.

Extra content of resources toevoegen aan je solution

Documentatie over je oplossing

Binnen je solution moet je de nodige documentatie opnemen over je oplossing:

- (voorstel van) opgave
- Zelfevaluatie en feedback door mentor
- Resultaten van codereview door andere studenten

Het is de bedoeling dat ook die bestanden in je GIT-repository opgenomen worden.

Dat doe je op volgende manier:

1. Klik in de Solution Explorer met de rechtermuisknop op de naam van de **solution** en selecteer 'Add' – 'New Item...'.
2. Kies het type document dat je wilt toevoegen (bv. 'General' – 'Text File') en geef de naam van het bestand in.
3. Als je op 'Add' klikt wordt er een map 'Solution Items' aangemaakt met daarin het nieuwe bestand.

Als je een reeds bestaand bestand wilt toevoegen (in plaats van een nieuw te creëren), doe je dat als volgt:

1. Klik in de Solution Explorer met de rechtermuisknop op 'Open folder in File Explorer'.
2. Kopieer het bestaande bestand naar de geopende map.
3. Klik in de Solution Explorer met de rechtermuisknop op de naam van de **solution** en selecteer 'Add' – 'Existing Item...'.
4. Dubbelklik op het gewenste bestand.

Vergeet niet om ook na het toevoegen (of wijzigen) van dergelijke bestanden een 'Commit All & Sync' uit te voeren, zodat ook die bestanden in je Git-repository terecht komen.

Resources voor je toepassing

Het zal af en toe eens gebeuren dat je bij de oplossing van een opgave naast je programmacode ook nog extra resources nodig hebt (zoals afbeeldingen, gegevensbestanden, geluidsfragmenten...).

Als je die bestanden vanuit je code wil gebruiken, stelt zich het probleem dat je dan moet weten waar die precies te vinden zijn op schijf.

Een voorbeeld codefragment waar zo een bestand gelezen wordt:

```
string path = @"D:\Gebruikers\PietPienter\Mijn Documenten\ImageA.jpg";  
demoPictureBox.Image = new Bitmap(Image.FromFile(path));
```

Als je een oplossing indient met dergelijke code, zal er een probleem ontstaan: je toepassing zal misschien goed werken op jouw computer, maar ze zal een fout geven op onze servers. Het pad dat je in je code gespecificeerd hebt, bestaat daar namelijk niet.

Los daarvan zal je gegevensbestand ook niet mee gekopieerd worden naar de GIT-server, want het maakt geen deel uit van je GIT-repository.

Gelukkig is er een vrij eenvoudige **oplossing** voor dit probleem:

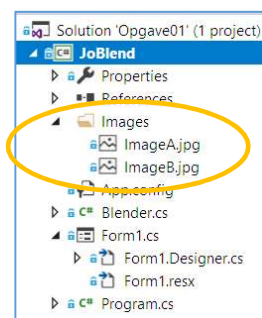
1. Maak in de solution explorer binnen het project met je oplossing een extra map aan voor je gegevensbestanden:

Klik met je rechtermuisknop op de naam van je **startproject**, selecteer daar 'Add' en 'New Folder' en geef die map een geschikte naam (bv. "Images", "Resources").

2. Kopieer vanuit de solution explorer het gegevensbestand naar die nieuwe map:

- Klik met de rechtermuisknop op die nieuwe map.
- Selecteer 'Add' en 'Existing Item'.
- Selecteer het bestaande gegevensbestand dat zich ergens op je schijf bevindt.

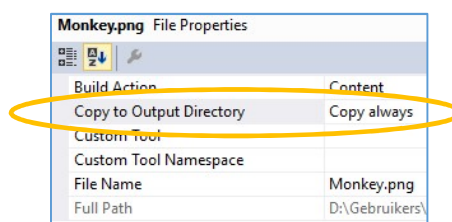
Het gegevensbestand wordt naar je extra map gekopieerd en verschijnt ook in de solution explorer:



Nu maakt die map met het gegevensbestand deel uit van je solution en zal dus ook opgenomen worden in je repository (en gesynchroniseerd worden naar de GIT-server).

3. Zorg ervoor dat bij het bouwen van je oplossing de extra map steeds gekopieerd wordt naar de directory waar je uitvoerbare code in terecht komt:

- Klik met de rechtermuisknop op het gegevensbestand
- Selecteer 'Properties'
- Zet de waarde van de property 'Build Action' op 'Content' (als dat nog niet zo is).
- Zet de waarde van de property 'Copy to Output Directory' op 'Copy always' of 'Copy if newer'.



Telkens als je toepassing gecompileerd wordt, zal de map met de extra bestanden nu ook gekopieerd worden naar de directory waarin je toepassing opgestart zal worden (normaal gezien is dat de 'bin\Debug\' of 'bin\Release\' directory binnen je project).

Vanuit je code kan je dat bestand dan inlezen via een relatief pad:

```
string path = @"Images\ImageA.jpg";  
demoPictureBox.Image = new Bitmap(Image.FromFile(path));
```

Ook op onze servers zal je code dan zonder problemen werken.

Opmerking:

Als je het voorgaande leest, zou je in de verleiding kunnen komen om je gegevensbestanden gewoon zelf in de juiste directory ('bin\Debug\' of 'bin\Release') te plaatsen, zonder de procedure via de solution explorer te volgen, en dan vanuit je code met een relatief pad te werken;

Ook dat zal op onze servers problemen geven: Omdat GIT een 'source control' systeem is, zal het bij de synchronisatie naar de GIT-server enkel bronbestanden kopiëren, en niet de (uitvoerbare) bestanden die in de 'bin\Debug\' of 'bin\Release\' directories staan. Dat zou ook verspilling van bandbreedte en schijfruimte zijn: die uitvoerbare bestanden kunnen achteraf gewoon opnieuw aangemaakt worden door een 'Build' te doen van de oplossing.

Maar het betekent dus ook dat een bestand dat je zelf manueel naar een van die directories kopieert niet opgenomen wordt in de GIT-repository en dus ook achteraf niet beschikbaar zal zijn op onze servers.

Gebruik van unittests in Visual Studio 2019

Unittests zijn korte stukjes code die als bedoeling hebben de goede werking van afzonderlijke stukken ('units') van je oplossing te testen. Zo hoeft je niet telkens heel je programma uit te voeren om bijvoorbeeld één afzonderlijke methode te testen.

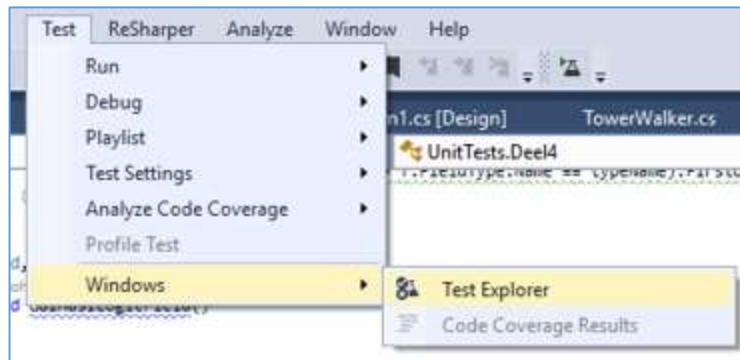
Ze zijn erg belangrijk, omdat ze ook snel zullen signaleren dat je bij wijzigingen aan je oplossing onbedoeld fouten introduceert.

Unittests kan je zelf schrijven. Binnen dit vak kan je leren hoe je dat doet.

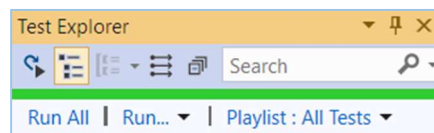
Binnen Visual Studio is een complete 'testing engine' geïntegreerd die je toelaat de unittests manueel of automatisch (bij elke build) te laten uitvoeren en de resultaten van de tests te bekijken.

Om de unit tests efficiënt te gebruiken maak je best de 'Test Explorer' zichtbaar:

- Klik bovenaan in het menu op 'Test' – 'Windows' – 'Test Explorer':





Je zal dan links de Test Explorer te zien krijgen:



Als je daar op 'Run All' klikt, worden alle beschikbare tests gedetecteerd en uitgevoerd.

Enkel nuttige tips:

- Met de  knop links bovenaan kan je een optie inschakelen waardoor na elke build automatisch alle testst uitgevoerd worden
- Met de  knop links bovenaan kan je de tests laten sorteren volgens een eventueel ingebouwde hiërarchie (aangeraden).
- Je kan ook individuele tests uitvoeren (rechtermuisknop op de test en 'Run Selected Tests').
- Als een test niet geslaagd is, krijg je onderaan extra informatie:



Meestal geeft de Message nuttige informatie over wat er misging. Je kan de source van de unittest bekijken (om uit te vissen hoe de test werkt) door te klikken op de link naast 'Source'. Daar kan je zien hoe de unittest jouw code gebruikt, wat er getest wordt en welk resultaat verwacht wordt.

Op basis daarvan kan je wellicht uitzoeken waar de fout in jouw code zit.

- Ook binnen unittests kan je breakpoints gebruiken om je code te debuggen. Je zal de test dan moeten starten met 'Debug Selected Tests' in plaats van 'Run Selected Tests'.

Na een 'Commit All en Sync' kan je in de Web-interface van de GIT-server controleren of de build van je toepassing op onze servers gelukt is (tenminste als je de configuratie op pagina 8 doorlopen hebt). Je kan daar bovenop in de logfile van die build ook zien wat het resultaat was van de unittests op onze server.

Dat kan nuttig zijn wanneer een toepassing wel correct werkt op je eigen computer, maar niet meer op onze servers (bijvoorbeeld door het gebruik van absolute paden naar bestanden).

Bronvermelding in je programmacode

Net zoals bij het schrijven van verslagen, bachelorproef, stageverslag, ... moet je ook bij het schrijven van programmeercode de bron waar je hulp of voorbeeldcode... gevonden hebt, vermelden. In plaats van een referentie op het einde van het document of als voetnoot voeg je je referentie toe als commentaar in je code op de plaats waar je het specifieke stuk code gebruikt. Op deze manier geef je niet alleen erkenning aan de personen die je direct of indirect geholpen hebben, maar kan dit later ook een grote hulp zijn bij het debuggen of begrijpen van je code.

Je citeert je bron als:

- Je "woordelijk" code hebt gekopieerd van een externe bron (website, forum, boek, ...). Dit geldt ook als je maar één lijn code hebt gekopieerd.
- Je je baseert op bestaande code (via een externe bron) en die aangepast hebt aan je specifieke noden.

Je citeert je bron als volgt:

- Voeg minstens de URL van de website waar je je informatie gehaald hebt toe, eventueel samen met de datum waarop je de website bezocht hebt. Je mag uiteraard altijd meer details toevoegen om de lezer te helpen bij het begrijpen van de code.
- Indien je code overgenomen is van voorbeeldcode uit een boek, dan geef je in commentaar minstens de namen van de auteurs, de titel van het boek, de paginanummers waarop je de code gevonden hebt, en het ISBN-nummer (indien aanwezig).

Doe dit ook als je de gevonden code aangepast hebt en als het dus geen letterlijke kopie is. Dan schrijf je in commentaar "Gebaseerd op code gevonden op" en vervolgens de URL of naam van het boek.

Voorbeeld:

```
// Connectie maken m.b.v. DbProviderFactory klasse.
//
// bron: "http://msdn.microsoft.com/en-us/library/dd0w4a2z(v=vs.110).aspx"
// geraadpleegd op 15 mei 2013.
//
DbProviderFactory factory = DbProviderFactories.GetFactory(providerName);
DbConnection connection = factory.CreateConnection();
```