## Budapesti Műszaki és Gazdaságtudományi Egyetem

## Témalaboratórium (BMEVIAUAL00)

# **Az open-source projekt kiadás alappilérei félév végi beszámoló.**

Hargitai Tamás (EM5O2R)

Bedő Bence (IS5WY7)

2018. december 01.

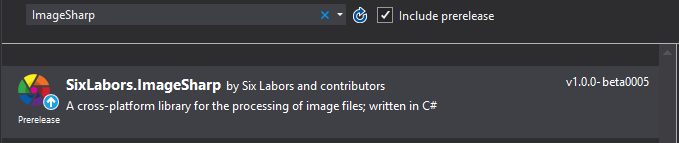
**SigStat - Dokumentáció csapat**

A félév során “az open-source projekt kiadás alappílérei” nevű témán dolgoztunk. A félév elején a témára jelentkezők csapatokra lettek osztva, majd megkaptuk a konkrét feladatokat, amelyeket aztán egymás között felosztottunk és azon dolgoztunk. A kitűzött feladatok a következők voltak: *Automatizált markdown dokumentálás megvalósítása*, *issue tracking, SEO-val való foglalkozás, GitHub nyitóoldal elkészítése, PDF dokumentáció.* Ezen témákat szeretnénk most pontról pontra bemutatni, részletesen kitérve a probléma eredetére és az általunk megvalósított megoldásokra.

Az *automatizált dokumentáció készítése* volt a legösszetettebb feladat a félév során, ezzel nagyon sokat dolgoztunk. Az alapötlet a következő volt: kell számunkra egy eszköz, amely a projektünkben található XML kommentek alapján automatizáltan dokumentációt készít, a kommentek módosulása esetén frissíti azt. A Visual Studio alapvetően elkészíti az XML dokumentációt, de az emberi olvasásra kevésbé alkalmas. Így az első feladatunk az volt, hogy ennek a megfelelő alternatíváját megtaláljuk. Vizsgáltunk más népszerű GitHub projekteket, és azokból merítettünk ötleteket.

Az összes megoldást elemezve **a dokumentáció helyének** szempontjából (README.md, docs mappában, külön oldalon, wiki oldalon) egyértelműen a GitHub oldalra integrált **Wiki-oldal** elkészítése volt a legmegfelelőbb. A GitHub-on Markdown jelölőnyelv (.md) a használandó, így egy olyan megoldást kellett találnunk, amely XML kódból Markdown kódot készít. Ehhez próbálkoztunk különböző nuGet package-k installálásával (pl.: Vsxmd), külső forrásokból való dokumentum generálással, de végül egy másik open-source, .NET-es projektet vettünk segítségül. A projekt neve **MarkdownGenerator**, és itt érhető el: <https://github.com/neuecc/MarkdownGenerator>

Fontos, hogy ezen program használatához fel kell telepíteni a **SixLabors.ImageSharp** nevű nuGet package-t, amelyet Visual Studio 2017-ben a Tools menüpont → NuGet Package Manager → Manage NuGet Packages for Solution → Browse résznél tehetünk meg.



A MarkdownGenerator projekt fordítása után előálló **MarkdownGenerator.exe** fájlt emeltük át a saját projektünkbe a hozzá szükséges .dll fájlokkal, illetve a program dokumentációját képező xml fájlokkal együtt a **docs\generator** mappába. Innentől kezdve csak ezekkel a fájlokkal dolgoztunk. Első lépésként az egyszerű működésre bírás volt a célunk, majd ezután tértünk át az **AppVeyor**, mint Continous Integrationt biztosító alkalmazás integrálására, illetve a wiki oldalra való automatikus feltöltésre.

A MarkdownGenerator a következő szintaxissal dolgozik:

<MarkdownGenerator.exe elérési útja> <A projektünk fordítása után előállt .dll fájl> <A generált dokumentum célmappája>

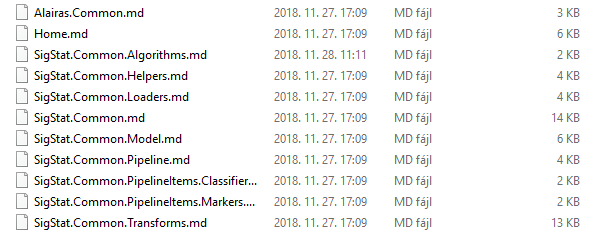
Először ezt lokálisan, a le pullolt projekten próbáltuk ki, még pedig úgy, hogy a projektünk post build event-jében valósítottuk meg a fenti szintaxist a következő módon:

"$(SolutionDir)docs\generator\MarkdownGenerator.exe" // az elérési út "$(SolutionDir)src\SigStat.Common\bin\Debug\

netstandard2.0\SigStat.Common.dll" // a dll fájl

"$(SolutionDir)docs\md") // a célmappa

Ezen build event lefutása után az *md* mappában megjelenik a projektünk kommentjeiből generált markdown kód.



A következő feladatunk az volt, hogy a dokumentum generálás **ne build-eléshez** legyen kötve, sokkal inkább használjuk ki az *AppVeyor* integrációs eszköz funkcionalitását. A SigStat projekthez az AppVeyor integráció már megvalósult, így csak a konfigurációért felelős **appveyor.yml** fájl megfelelő módosítása volt a feladatunk. Két részre osztottuk a feladatot, először műkődésre bírtuk az automatikus dokumentum generálást, majd utána a wiki-re való automatikus feltöltéssel foglalkoztunk.

Az elkészült elkészült yml fájl releváns sorait mutatjuk, illetve lentebb mellékeljük a magyarázatot:

(1) on\_success:

(2) - git config --global credential.helper store

- ps: Add-Content "$HOME\.git-credentials"

"https://$($env:access\_token):x-oauth-basic@github.com`n"

- git config --global user.email "hargitomi97@gmail.com"

- git config --global user.name "hargitomi97"

(3) - md C:\projects\wiki

- cd C:\projects\wiki

- git clone https://github.com/sigstat/sigstat.wiki.git

(4) - C:\projects\sigstat\docs\generator\MarkdownGenerator.exe C:\projects\sigstat\src\SigStat.Common\bin\Release\netstandard2.0\SigStat.Common.dll "C:\projects\wiki\SigStat.wiki"

(5) - echo All | xcopy C:\projects\sigstat\docs\md\\_Sidebar.md C:\projects\wiki\SigStat.wiki

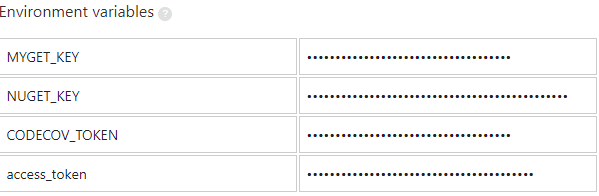
(6) - cd SigStat.wiki

- git add -A

- git diff-index --quiet HEAD || git commit -am "Commiting wiki changes"

- git push

Az on\_success (1) tag arra utal, hogy amennyiben sikeresen lefordult a projektünk, a lentebb látható lépéseket hajtsa végre az AppVeyor. Az első 4 sorban azonosítjuk magunkat (2), e nélkül nem tudjuk elérni a kívánt repository-t (sigstat). Első lépésként az AppVeyor webes felületén, a projektünk beállításai között meg kell adnunk egy általunk GitHub-on létrehozott **Personal Access Token**-t.



Miután ez megtörtént, az appveyor.yml fájlba már bele írhatjuk az első 4 sort, és így az azonosítás sikeresen meg fog történni, elérjük a SigStat repositoryját. (Részletekért lásd: <https://www.appveyor.com/docs/how-to/git-push/>)

Ezután készítünk egy wiki mappát, bele navigálunk, és leclone-ozzuk ide a projektünk jelenlegi wiki oldalát. (3)

Majd a fentebb ismertetett módon kell a MarkdownGeneratort műkődésbe hozni.

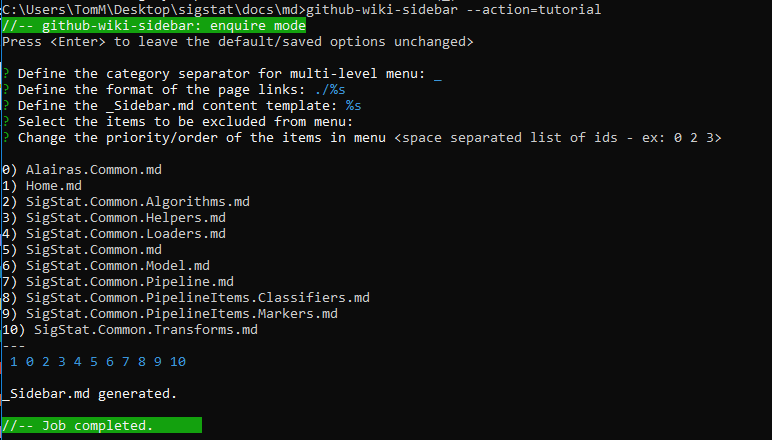
Ehhez megkeressük a MarkdownGenerator.exe fájlt, majd a lefordított projekt .dll fájlját (mivel korábban configuration = Release van beállítva a konfigurációs fájlban, így Debug helyett a Release mappában keresünk), majd megadjuk, hogy hova tegye az elkészült markdown fájlokat, természetesen a le clone-olt wikibe szeretnénk tenni. (4)

A következő sor a sidebar generáláshoz van, erre lentebb térünk ki. (5)

Az utolsó 4 sorban bele navigálunk a leclone-ozott wiki mappába, és a már ismert módon stagelünk minden változást, és ha történt változás (ennek ellenőrzésére szolgál a diff), akkor commitolunk és felpusholjuk a repositoryba a megfelelő fájlokat. (6)

Az automatizált dokumentációhoz tartozik még a wiki oldalon történő **Sidebar** generálása, amely lehetővé teszi, hogy oldalt is felsoroljuk az egyes fájlokat. Ez a sima Pages füllel jelenleg azonos funkcionalitással bír, de a jövőben továbbfejleszthető. Ehhez a **github-wiki-sidebar** nevű projektet használtuk fel, amely itt érthető el: <https://github.com/adriantanasa/github-wiki-sidebar>

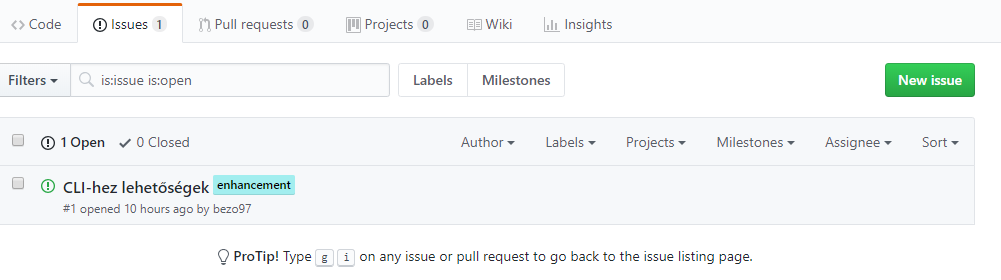
Telepítéséhez **Node JS** szükséges. Telepítés után a projekt docs mappájába ha kiadjuk parancssorból a *github-wiki-sidebar --action=tutorial* parancsot, akkor előállíthatjuk azt a *options.json* fájlt, amely tartalmazza, hogy mely projekt részekből készüljön el a sidebar.



Ezután az elkészült sidebart kell feltölteni a wikinkre, amelynek menetét az (5)-ös lépés tartalmazza: az elkészült **\_Sidebar.md** fájlt átmásoljuk a wiki könyvtárunkba. Az *echo All* azért szükséges, hogy mindig megtörténjen a felülírás.

A következő megvalósítandó feladat az *issue tracking* volt, amely tömören annak a platformnak a bevezetését jelenti, amelyen keresztül mi magunk, illetve a jövendőbeli contributorok tudnak hibákat feljegyezni. Ezen hibajegyeket megoldottnak lehet jelölni, lehet kommentelni alá, és szerkeszthetőek. Itt is vizsgáltunk más projekteket, vannak akik egy külső oldalt vonnak be ebbe, míg valakik GitHub-on lokálisan oldják meg ezt.

Mi is a **GitHub-os** megoldás mellett maradtunk. A projekt főoldalán a fenti *Issues* menüpontra kattintva jön be a megfelelő felület. Ehhez csupán annyi teendőnk volt, hogy a projektnél engedélyezni kellett az Issues menüpontot.



Az automatikusan generált dokumentáció készítése során fontos volt, hogy a wiki-re generált dokumentáción kívül mindig automatikusan létrehozzunk egy dokumentációs PDF-et is. Ennek a megvalósításához alapvetően két másik projekt segítségére és szintén az AppVeyor-ra volt szükségünk. Először is a Github Wikito Converter(<https://www.npmjs.com/package/github-wikito-converter>) segítségével készítünk egy egy oldalba ömlesztett oldalt az egész github-on lévő wikiből majd utána a node-html-pdf(<https://github.com/marcbachmann/node-html-pdf>)-el konvertálunk belőle egy PDF-et. Itt szintén az appveyor.yml konfigurációs fájl módosítására volt szükségünk. Először is telepítjük a szükséges komponenseket a before\_build résznél.



Ezután le klónozzuk a wikit



majd végrehajtjuk a generálást és konvertálást.



Továbbá a mi feladatunk volt, hogy a github projekthez készítsünk egy főoldalt. Ennek elkészítése során segítségünkre szolgált <https://dillinger.io/> amely szinte azonnal ki generálja az általunk írt főoldalt. Elsősorban nem az volt a feladatunk, hogy egy végleges oldalt készítsünk, inkább találjunk ki neki egy struktúrát amelyet a egyre jobban lehet fejleszteni a projekt fejlődése során. Ennek az elkészítéséhez különböző cikkekből illetve más projektekből(akkor éppen felkapott projektek Github-on) merítettünk ötleteket. Ezek alapján szinte minden főoldal nélkülözhetetlen része egy rövid bemutató szöveg, különböző funkciók ismertetése illetve milyen technológiai alapokra épül.

A félév során kisebb mértékben érintettük a SEO (Search Engine Optimization) megvalósítását is. Ezzel a főoldalhoz hasonlóan más projektek illetve cikkek alapján foglalkoztunk. Azonban ezek nagy része egy készülő weboldalhoz adtak tanácsokat, amely elkészítésével nem mi foglalkoztunk. Ezen felül a github főoldalt is próbáltuk ennek alapján elkészíteni illetve rengetegen kiemelték, hogy mennyi szerepe lehet a különböző szociális hálókon való aktív részvételnek, ennek megfelelően hagytuk benne a Latest News menüpontot a főoldalon. Természetesen ennek csak akkor lesz jelentősége amikor eljut egy olyan szintre a projekt, hogy tudjon megfelelő tartalmat közvetíteni, illetve lesz rá elegendő kapacitás.