C# - дибагер

На линку се налази упутство за покретање и коришћење дибагера унутар *Visual Studio Code*-а. Уколико у вашем пројекту немате *.vscode* фолдер са потребним фајловима за покретање дибагера, можете га генерисати на идући начин:

`<Ctrl Shift P>` па укуцајте `.NET: Generate assets for Build and Debug <Enter>`

Ова наредба ће генерисати *.vscode* фолдер са два фајла *launch.json* и *tasks.json*.

launch.json

launch.json – фајл са подешавањима која су потребна *Code*-у да извршава *.NET* апликације.

Генерисан фајл има нека подешавања која нам не одговарају приликом развоја, па је препорука да их измијените. То је: `"console": "internalConsole"` подешавање које говори *Code*-у да користи *Debug* прозор за испис и примање уноса од корисника. Ово не ради јер *Debug* прозор не може да прими унос од корисника, па се за апликације којима је потребан унос од корисника препоручује да користите `"console": "integratedTerminal"`.

Кад се користи интегрисани терминал *Code* опет отвара *Debug* прозор, а он ти често неће требати током развоја. Са `"internalConsoleOptions": "neverOpen"` можеш рећи *Code*-у да не отвара *Debug* прозор.

Све опције које можете подесити су пописане у документацији.

tasks.json

tasks.json – садржи задатке (енгл. *tasks*) који се често користе и које можеш покренути користећи *Code*. Ови задаци се покрећу са:

`<Ctrl Shift P> Task: Run task <Enter>` унеси назив такса који желиш да покренеш `<Enter>`

Генерисан фајл садржи 3 задатка:

- *build* – компајлира апликацију, увеже њене зависности и направи извршиве фајлове
- *publish* – компајлира апликацију и даје нам фајлове који су спремни за *deployment* (покретање апликације у продукцији)
- *watch* –сваки пут кад се сачува измјена у фајл који је одабран поново извршава апликацију– корисно за развој и тестирање

Можеш и сам да правиш ове задатке и врло су згодни кад временом уочиш да се неке ствари понављају и желиш да их аутоматизујеш. У документацији можеш пронаћи више информација.

1. Са канваса скини *TechJobsConsole.zip*. Користи дибагер и пречице да пронађеш багове и ријешиш их. Познати багови су:
    a. Претрага колоне по задатој вриједности не врати тачне податке. Помоћу дибагера провјери ову функцију и исправи грешку.
    b. Нема исписа за претрагу из тачке а, па прошири тај задатак да исписује резултате.
    c. Апликација има проблема са перформансама. Нека функција се непотребно позива више пута. Можеш ли пронаћи ту функцију и ријешити баг?

Пречице

Пречице за *Code* за *Widows*, *Linux*, *MacOS*. Уколико користиш неко од *IDE*-а *Rider*, *Visual Studio*. Подразумијевана подешавања нису увијек оптимална, зато је савијет да пречице прилагодиш себи и својим потребама. Акције које користиш чешће постави да ти буду близу и да буду кратке.

Програмери већи проценат свог времена троше на кретање кроз код у односу на куцање кода. Током ових вјежби што више покушај да користиш пречице за навигацију и кретање кроз фајлове како би убрзао развој и ријешавање задатака.

Пречице које бих издвојио да су битне да их научиш:

- <Ctrl Shift P> - Quick open, Go to file
- <Ctrl G> - Go to line
- <Ctrl Shift O> - Go to symbol
- <F8> – Go to next error/warning
- <Shift F8> - Go to previous error/warning
- <Ctrl Alt -> - Go back
- <Ctrl Shift- ->Go forward
- <Ctrl Space> Trigger suggestion
- <Ctrl Shift-space> Trigger parameter suggestion
- <Alt up/down arrow> - Move line up/down
- <Ctrl F>- find
- <Ctrl H> - replace
- <Ctrl Tab> - Next tab
- <Ctrl Shift Tab> - Previous tab

Наведене пречице су за *Code*.

2. Вјежбај дибагер и искористи што више пречица тако што ћеш се упознати са идућим пројектом: https://github.com/ZacharyPatten/dotnet-console-games. За почетак узми неку од једноставнијих игрица (рецимо *TicTacToe*) да се упознаш са начином имплементације. Након тога пређи на *RolePlayingGame* и помоћу дибагера истражуј игрицу и покушај да откријеш зашто кад држиш неко дугме за кретање (стрелице или *WSAD*), играч заустави на кратко, па тек онда настави кретање?