Kratak sadržaj

POGLAVLJE 1
Zdravo C#! Dobrodošao .NET! 1

POGLAVLJE 2
Govorite jezikom C# 47

POGLAVLJE 3
Kontrola toka, konvertovanje tipova i rukovanje izuzecima. 95

POGLAVLJE 4
Pisanje funkcija, ispravljanje grešaka i testiranje funkcija 131

POGLAVLJE 5
Izgradnja sopstvenih tipova pomoću objektno-orijentisanog programiranja 177

POGLAVLJE 6
Implementiranje interfejsa i nasleđivanje klasa 219

POGLAVLJE 7
Pakovanje i distribucija .NET tipova 273

POGLAVLJE 8
Upotreba uobičajenih .NET tipova 317

POGLAVLJE 9
Upotreba fajlova, tokova i serijalizacije 369

POGLAVLJE 10

Upotreba baza podataka pomoću radnog okvira Entity Framework Core 407

POGLAVLJE 11

Slanje upita i manipulisanje podacima upotrebom LINQ upita 467

POGLAVLJE 12

Poboljšanje performansi i skalabilnosti upotrebom višeprogramskog rada 505

POGLAVLJE 13

Praktična primena jezika C# i .NET platforme 541

POGLAVLJE 14

Izgradnja veb sajtova upotrebom radnog okvira ASP.NET Core Razor Pages 567

POGLAVLJE 15

Izgradnja veb sajtova upotrebom obrasca Model-View-Controller 615

POGLAVLJE 16

Izgradnja i upotreba veb servisa 667

POGLAVLJE 17

Izgradnja korisničkog interfejsa pomoću radnog okvira Blazor 719

EPILOG 771

DODATAK

Odgovori na pitanja za testiranje znanja 775

INDEKS 775

Sadržaj

Uvod XXIX

POGLAVLJE 1

Zdravo C#! Dobrodošao .NET! 1

Podešavanje razvojnog okruženja 2
Izbor odgovarajućeg alata i tipa aplikacije za učenje 3
Prednosti i nedostaci ekstenzije .NET Interactive Notebooks 3
Upotreba uređivača koda Visual Studio Code za međuplatformski razvoj 4
Upotreba razvojnog okruženja GitHub Codespaces za razvoj u cloud platformi 4
Upotreba razvojnog okruženja Visual Studio for Mac za opšti razvoj 4
Upotreba razvojnog okruženja Visual Studio for Windows za opšti razvoj 5
Šta sam ja upotrebio? 5
Međuplatformsko raspoređivanje 6
Preuzimanje i instaliranje razvojnog okruženja Visual Studio 2022 for Windows 6
Prečice na tastaturi za Microsoft Visual Studio for Windows 7
Preuzimanje i instaliranje uređivača koda Visual Studio Code 7
Instaliranje drugih ekstenzija 8
Razumevanje verzija uređivača koda Microsoft Visual Studio Code 9
Prečice na tastaturi za uređivač koda Microsoft Visual Studio Code 9
Razumevanje platforme .NET 10
Razumevanje razvojne platforme .NET Framework 10
Razumevanje Mono, Xamarin i Unity projekata 10
Razumevanje radnog okvira .NET Core 11
Razumevanje budućih verzija platforme .NET 11
Razumevanje podrške za platformu .NET 12
Razumevanje verzija alata .NET Runtime i .NET SDK 13
Uklanjanje starih verzija .NET platforme 14
Šta je drugačije u modernoj .NET platformi? 14
Windows razvoj 14
Veb razvoj 15
Razvoj baze podataka 15

Teme moderne .NET platforme 15
Razumevanje specifikacije .NET Standard 15
.NET platforme i alatke upotrebljene u izdanjima ove knjige 16
Razumevanje posredničkog jezika 17
Upoređivanje .NET tehnologija 17
Izgradnja konzolnih aplikacija pomoću razvojnog okruženja Visual Studio 2022 18
Vođenje više projekata upotrebom razvojnog okruženja Visual Studio 2022 18
Pisanje koda korišćenjem razvojnog okruženja Visual Studio 2022 18
Kompajliranje i pokretanje koda pomoću razvojnog okruženja Visual Studio 20
Razumevanje direktorijuma i fajlova koje generiše kompajler 21
Pisanje programa najvišeg nivoa 21
Dodavanje drugog projekta pomoću razvojnog okruženja Visual Studio 2022 22
Implicitno importovani imenski prostori 22
Izgradnja konzolnih aplikacija pomoću uređivača koda Visual Studio Code 24
Vođenje više projekata upotrebom uređivača koda Visual Studio Code 24
Pisanje koda pomoću uređivača koda Visual Studio Code 24
Kompajliranje i pokretanje koda pomoću dotnet CLI alatke 27
Dodavanje drugog projekta korišćenjem uređivača koda Visual Studio Code 27
Korišćenje više fajlova pomoću uređivača koda Visual Studio Code 29
Istraživanje koda pomoću ekstenzije .NET Interactive Notebooks 29
Kreiranje beležnice 30
Pisanje i pokretanje koda u beležnici 31
Snimanje beležnice 32
Dodavanje jezika Markdown i specijalnih komandi u beležnicu 32
Izvršavanje koda u više ćelija 33
Korišćenje .NET interaktivnih beležnica za kod u ovoj knjizi 34
Pregledanje direktorijuma i fajlova za projekte 34
Razumevanje uobičajenih direktorijuma i fajlova 35
Razumevanje koda rešenja u GitHub skladištu 36
Dobra upotreba GitHub skladišta za ovu knjigu 36
Pitanja u vezi sa knjigom 36
Povratne informacije 37
Preuzimanje koda rešenja iz GitHub skladišta 37
Upotreba Git skladišta pomoću uređivača koda Visual Studio Code i komandne linije 38
Kloniranje skladišta koda rešenja ove knjige 38
Potražite pomoć 39
Čitanje Microsoft dokumentacije 39
Dobijanje pomoći za dotnet alatku 39
Dobijanje definicija tipova i njihovih članova 40
Traženje odgovora na veb sajtu Stack Overflow 42
Pretraživanje odgovora upotrebom pretraživača Google 43
Pretplata na zvaničan .NET blog 43
Pregled video snimaka Scotta Hanselmana 43

Vežbanje i istraživanje....43
Vežba 1.1 – Testirajte svoje znanje....43
Vežba 1.2 – Vežbajte C# svuda....44
Vežba 1.3 – Istražite teme....44
Rezime....45

POGLAVLJE 2

Govorite jezikom C# 47

Predstavljanje C# jezika....47
Razumevanje verzija jezika i funkcija....48
C# 1.0....48
C# 2.0....48
C# 3.0....48
C# 4.0....48
C# 5.0....49
C# 6.0....49
C# 7.0....49
C# 7.1....49
C# 7.2....50
C# 7.3....50
C# 8.0....50
C# 9....50
C# 10....50
Razumevanje standarda C# jezika....51
Otkrivanje verzija C# kompajlera....51
Ispis SDK verzije....52
Omogućavanje kompajlera specifične verzije jezika....52
Razumevanje gramatike i rečnika C# jezika....53
Prikazivanje verzije kompajlera....53
Razumevanje gramatike C# jezika....55
Iskazi....55
Komentari....55
Blokovi....56
Primeri iskaza i blokova....56
Razumevanje C# rečnika....57
Poređenje programskih jezika sa ljudskim jezicima....57
Promena kolorne šeme za C# sintaksu....57
Pomoć za pisanje tačnog koda....58
Importovanje imenskih prostora....59
Implicitno i globalno importovanje imenskih prostora....59
Glagoli su metodi....62
Imenice su tipovi, promenljive, polja i svojstva....62
Otkrivanje obima C# rečnika....63
Upotreba promenljivih....65
Imenovanje i dodela vrednosti....66
Vrednosti literala....66
Skladištenje teksta....66

Razumevanje doslovnih znakovnih nizova....67
Skladištenje brojeva....68
Skladištenje celih brojeva....68
Istraživanje celih brojeva....69
Skladištenje realnih brojeva....70
Pisanje koda za istraživanje veličine brojeva....70
Poređenje double i decimal tipova....71
Skladištenje logičkih vrednosti (Boolean)....73
Skladištenje bilo kog tipa objekta....73
Skladištenje dynamic tipova....74
Deklarisanje lokalnih promenljivih....76
Specifikovanje tipa lokalne promenljive....76
Izvođenje tipa lokalne promenljive....76
Upotreba target-typed new za instanciranje objekata....78
Preuzimanje i podešavanje podrazumevanih vrednosti za tipove....78
Skladištenje više vrednosti u niz....79
Dalje istraživanje konzolnih aplikacija....80
Prikazivanje rezultata korisniku....81
Formatiranje pomoću numerisanih pozicionih argumenata....81
Formatiranje upotrebom interpoliranih znakovnih nizova....82
Razumevanje znakovnih nizova za formatiranje....82
Dobijanje tekstualnog unosa od korisnika....84
Pojednostavljivanje upotrebe konzole....84
Dobijanje unosa tastera od korisnika....85
Prosleđivanje argumenata u konzolnu aplikaciju....86
Podešavanje opcija pomoću argumenata....88
Rukovanje platformama koje ne podržavaju API....90
Vežbanje i istraživanje....91
Vežba 2.1 – Testirajte svoje znanje....91
Vežba 2.2 – Testirajte svoje znanje o numeričkim tipovima....92
Vežba 2.3 – Vežbajte veličine brojeva i raspone....92
Vežba 2.4 – Istražite teme....93
Rezime....93

POGLAVLJE 3

Kontrola toka, konvertovanje tipova i rukovanje izuzecima. 95

Operacije u promenljivim....95
Unarni operatori....96
Binarni aritmetički operatori....97
Operatori dodele....98
Logički operatori....98
Uslovni logički operatori....100
Operatori nad bitovima i binarni operatori pomeranja....101
Razni operatori....103
Razumevanje iskaza selekcije....103
Grananje pomoću iskaza if....104
Zašto bi uvek trebalo da upotrebite velike zagrade u iskazima if?....105

Podudaranje obrazaca pomoću iskaza if ... 105
Grananje pomoću iskaza switch ... 106
Podudaranje obrazaca pomoću iskaza switch ... 108
Pojednostavljivanje switch iskaza pomoću switch izraza ... 109
Razumevanje iterativnih iskaza ... 110
Ponavljanje u petlji pomoću iskaza while ... 110
Ponavljanje u petlji pomoću iskaza do ... 111
Ponavljanje u petlji pomoću iskaza for ... 112
Ponavljanje u petlji pomoću iskaza foreach ... 112
Razumevanje kako iskaz foreach funkcioniše interno ... 113
Eksplicitna konverzija i konvertovanje između tipova ... 113
Implicitna i eksplicitna konverzija brojeva ... 114
Konvertovanje pomoću System.Convert tipa ... 115
Zaokruživanje brojeva ... 116
Razumevanje podrazumevanih pravila zaokruživanja ... 116
Preuzimanje kontrole nad pravilom zaokruživanja ... 117
Konvertovanje iz bilo kog tipa u znakovni niz ... 117
Konvertovanje iz binarnog objekta u znakovni niz ... 118
Raščlanjavanje iz znakovnih nizova u brojeve ili datume i vreme ... 119
Problemi pri korišćenju metoda Parse ... 120
Izbegavanje izuzetaka upotrebom metoda TryParse ... 120
Obrada izuzetaka ... 121
Umetanje koda sklonog greškama u blok try ... 121
Hvatanje svih izuzetaka ... 123
Hvatanje specifičnih izuzetaka ... 123
Hvatanje pomoću filtera ... 125
Provera prekoračenja ... 125
Podizanje izuzetka prekoračenja pomoću iskaza checked ... 125
Onemogućavanje kompajlera za proveru prekoračenja pomoću iskaza unchecked ... 127
Vežbanje i istraživanje ... 128
Vežba 3.1 – Testirajte svoje znanje ... 128
Vežba 3.2 – Istražite petlje i prekoračenja ... 129
Vežba 3.4 – Vežbajte obradu izuzetaka ... 130
Vežba 3.5 - Testirajte znanje o operatorima ... 130
Rezime ... 130

POGLAVLJE 4

Pisanje funkcija, ispravljanje grešaka i testiranje funkcija 131

Pisanje funkcija ... 131
Primer tablice množenja ... 132
Pisanje funkcije za tablicu množenja ... 132
Pisanje funkcije koja vraća vrednost ... 134
Konvertovanje brojeva iz rednog u osnovni broj ... 136
Izračunavanje faktorijala rekurzijom ... 137
Dokumentovanje funkcija pomoću XML komentara ... 140
Upotreba lambda izraza u implementacijama funkcije ... 141
Ispravljanje grešaka tokom razvoja ... 144

Kreiranje koda sa namernom greškom....144
Postavljanje tačke prekida i početak ispravljanja greške....145
Upotreba razvojnog okruženja Visual Studio 2022....145
Upotreba uređivača koda Visual Studio Code....146
Navigacija pomoću palete alatki za ispravljanje grešaka....148
Prozori za ispravljanje grešaka....149
Prolazak kroz kod....150
Prilagođavanje tačaka prekida....151
Evidentiranje tokom razvoja i izvršenja....153
Razumevanje opcija evidentiranja....153
Instrumentacija pomoću tipova Debug i Trace....154
Pisanje u standardni osluškivač ispisa....154
Konfigurisanje osluškivača ispisa....155
Menjanje nivoa ispisa....157
Dodavanje paketa u projekat u uređivaču koda Visual Studio Code....157
Dodavanje paketa u projekat u razvojnom okruženju Visual Studio 2022....158
Pregled paketa projekta....158
Jedinično testiranje....162
Tipovi testiranja....162
Kreiranje biblioteke klase koja zahteva testiranje....162
Pisanje jediničnih testova....164
Pokretanje jediničnih testova korišćenjem razvojnog okruženja Visual Studio....166
Ispravljanje greške....166
Generisanje i hvatanje izuzetaka u funkcijama....167
Razumevanje grešaka korišćenja i grešaka izvršenja....167
Uobičajeno generisani izuzeci u funkcijama....167
Razumevanje steka poziva....168
Gde uhvatiti izuzetke....171
Ponovno generisanje izuzetaka....171
Implementacija obrasca tester-doer....173
Problemi obrasca tester-doer....173
Vežbanje i istraživanje....174
Vežba 4.1 – Testirajte svoje znanje....174
Vežba 4.2 – Vežbajte pisanje funkcija, koristeći ispravljanje grešaka i jedinično testiranje....174
Vežba 4.3 – Istražite teme....175
Rezime....175

POGLAVLJE 5

Izgradnja sopstvenih tipova pomoću objektno-orijentisanog programiranja....177

Objektno-orijentisano programiranje....177
Izgradnja biblioteka klase....178
Kreiranje biblioteke klase....178
Definisanje klase u imenskom prostoru....179
Pojednostavljivanje deklaracija imenskog prostora....180
Razumevanje članova....181
Instanciranje klase....181

Referenciranje programskog sklopa ... 182
Importovanje imenskog prostora za upotrebu tipa ... 182
Razumevanje objekata ... 183
Nasleđivanje iz klase System.Object ... 184
Skladištenje podataka unutar polja ... 184
Definisanje polja ... 184
Razumevanje modifikatora pristupa ... 185
Zadavanje i ispisivanje vrednosti polja ... 186
Skladištenje vrednosti pomoću tipa enum ... 187
Skladištenje više vrednosti upotrebom tipa enum ... 188
Skladištenje više vrednosti pomoću kolekcija ... 189
Razumevanje generičkih kolekcija ... 190
Učinite polje statičkim ... 191
Učinite polje konstantnim ... 192
Kreirajte read-only polje ... 193
Inicijalizacija polja pomoću konstruktora ... 194
Definisanje više konstruktora ... 195
Pisanje i pozivanje metoda ... 195
Vraćanje vrednosti iz metoda ... 195
Kombinovanje više vraćenih vrednosti pomoću torki ... 196
Podrška jezika za torke ... 197
Imenovanje polja torke ... 198
Izvođenje naziva torke ... 198
Dekonstruisanje torki ... 198
Dekonstruisanje tipova ... 199
Definisanje i prosleđivanje parametara u metode ... 200
Preklapanje metoda ... 201
Prosleđivanje opcionih i imenovanih parametara ... 201
Imenovanje vrednosti parametra tokom pozivanja metoda ... 203
Kontrolisanje načina prosleđivanja parametara ... 203
Pojednostavljen parametar out ... 204
Razumevanje vraćenih vrednosti parametra ref ... 205
Rastavljanje klasa pomoću ključne reči partial ... 205
Kontrolisanje pristupa pomoću svojstava i indeksera ... 206
Definisanje read-only svojstava ... 206
Definisanje podesivih svojstava ... 207
Zahtevanje podešavanja svojstava tokom instanciranja ... 209
Definisanje indeksera ... 209
Podudaranje obrazaca pomoću objekata ... 210
Kreiranje i referenciranje .NET 6 biblioteke klase ... 210
Definisanje putnika leta ... 211
Poboljšanja u podudaranju obrazaca u verziji C# 9 ili novijim ... 212
Upotreba zapisa ... 213
Init-only svojstva ... 213
Razumevanje zapisa ... 214
Pozicioni članovi podataka u zapisima ... 215
Pojednostavljivanje članova podataka u zapisima ... 215

Vežbanje i istraživanje 216
Vežba 5.1 – Testirajte svoje znanje 217
Vežba 5.2 – Istražite teme 217
Rezime 217

POGLAVLJE 6

Implementiranje interfejsa i nasleđivanje klasa 219

Podešavanje biblioteke klase i konzolne aplikacije 220
Više o metodima 221
Implementiranje funkcionalnosti pomoću metoda 221
Implementiranje funkcionalnosti pomoću operatora 223
Implementiranje funkcionalnosti upotrebom lokalnih funkcija 224
Generisanje i obrada događaja 225
Pozivanje metoda pomoću delegata 226
Definisanje i obrada delegata 227
Definisanje i obrada događaja 229
Bezbedno kreiranje ponovo upotrebljivih tipova pomoću generičkih tipova 230
Upotreba negeneričkih tipova 230
Upotreba generičkih tipova 231
Implementiranje interfejsa 232
Uobičajeni interfejsi 232
Upoređivanje objekata tokom sortiranja 233
Poređenje objekata pomoću posebne klase 235
Implicitne i eksplicitne implementacije interfejsa 236
Definisanje interfejsa pomoću podrazumevanih implementacija 237
Upravljanje memorijom pomoću referentnog i vrednosnog tipa 239
Definisanje referentnih i vrednosnih tipova 239
Skladištenje referentnih i vrednosnih tipova u memoriji 240
Jednakost tipova 241
Definisanje tipova struct 242
Upotreba record struct tipova 243
Otpuštanje neupravljanih resursa 244
Uverite se da je pozvan metod Dispose 246
Korišćenje vrednosti null 246
Vrednosni tip koji prihvata vrednost null 246
Razumevanje referentnih tipova koji prihvataju vrednost null 247
Omogućavanje referentnih tipova koji prihvataju i koji ne prihvataju null vrednost 248
Deklarisanje promenljivih i parametara koji ne prihvataju vrednost null 248
Provera vrednosti null 250
Provera vrednosti null u parametrima metoda 251
Nasleđivanje iz klasa 252
Proširivanje klasa za dodavanje funkcionalnosti 252
Skrivanje članova 253
Zamena vrednosti članova 254
Nasleđivanje iz apstraktnih klasa 255
Sprečavanje nasleđivanja i promene vrednosti 256

Razumevanje polimorfizma ... 257
Konverzija unutar hijerarhije nasleđivanja ... 259
Implicitna konverzija ... 259
Eksplicitna konverzija ... 259
Izbegavanje izuzetaka konverzije ... 260
Nasleđivanje i proširenje .NET tipova ... 261
Nasleđivanje izuzetaka ... 261
Proširenje tipova kada se ne može izvršiti nasleđivanje ... 263
Primena statičkih metoda za ponovnu upotrebu funkcionalnosti ... 263
Korišćenje proširenih metoda za ponovnu upotrebu funkcionalnosti ... 264
Upotreba analizatora za pisanje boljeg koda ... 265
Obustavljanje upozorenja ... 267
Ispravljanje koda ... 268
Razumevanje uobičajenih StyleCop preporuka ... 270
Vežbanje i istraživanje ... 271
Vežba 6.1 – Testirajte svoje znanje ... 271
Vežba 6.2 – Vežbajte kreiranje hijerarhije nasleđivanja ... 271
Vežba 6.3 – Istražite teme ... 272
Rezime ... 272

POGLAVLJE 7

Pakovanje i distribucija .NET tipova ... 273

Uvod u .NET 6 ... 273
.NET Core 1.0 ... 274
.NET Core 1.1 ... 274
.NET Core 2.0 ... 275
.NET Core 2.1 ... 275
.NET Core 2.2 ... 275
.NET Core 3.0 ... 275
.NET Core 3.1 ... 276
.NET 5.0 ... 276
.NET 6.0 ... 276
Poboljšanje performansi od verzije .NET Core 2.0 do verzije .NET 5 ... 277
Provera ažuriranja za .NET SDK ... 277
Razumevanje .NET komponenata ... 277
Razumevanje programskih sklopova, NuGet paketa i imenskih prostora ... 278
Šta je imenski prostor? ... 278
Razumevanje zavisnih programskih sklopova ... 278
Razumevanje SDK paketa Microsoft .NET projekta ... 278
Razumevanje imenskih prostora i tipova u programskim sklopovima ... 279
Razumevanje NuGet paketa ... 280
Razumevanje radnih okvira ... 280
Importovanje imenskog prostora za upotrebu tipa ... 281
Povezivanje C# ključnih reči sa .NET tipovima ... 281
Mapiranje C# alijasa u .NET tipove ... 282
Razumevanje celih brojeva izvorne veličine ... 283
Otkrivanje lokacije tipa ... 283

Deljenje koda sa zastarelim platformama pomoću .NET Standard biblioteka klase ... 284
Podrazumevana podešavanja za biblioteke klase različitim SDK alatkama ... 284
Kreiranje .NET Standard biblioteke klase ... 285
Kontrola .NET SDK alata ... 286
Publikovanje koda za raspoređivanje ... 287
Kreiranje konzolne aplikacije za publikovanje ... 288
Razumevanje dotnet komandi ... 289
Kreiranje novih projekata ... 289
Dobijanje informacija o .NET platformi i njenom okruženju ... 290
Upravljanje projektima ... 291
Publikovanje samostalne aplikacije ... 292
Publikovanje aplikacije koja sadrži jedan fajl ... 293
Smanjivanje veličine aplikacije pomoću skraćivanja aplikacije ... 295
Omogućavanje skraćivanja na nivou programskog sklopa ... 295
Omogućavanje skraćivanja na nivou tipa i na nivou člana ... 295
Dekompajliranje .NET programskih sklopova ... 296
Dekompajliranje upotrebom ekstenzije ILSpy za razvojno okruženje Visual Studio 2022 ... 296
Dekompajliranje upotrebom ekstenzije ILSpy za uređivač koda Visual Studio Code ... 297
Ne, ne možete tehnički da sprečite dekompajliranje ... 301
Pakovanje biblioteka za NuGet distribuciju ... 302
Referenciranje NuGet paketa ... 302
Ispravljanje zavisnosti ... 303
Pakovanje biblioteke za NuGet ... 304
Publikovanje paketa na javni NuGet izvor ... 306
Publikovanje paketa na privatnom NuGet izvoru ... 307
Istraživanje NuGet paketa pomoću alata ... 307
Testiranje paketa biblioteke klase ... 308
Prelazak sa radnog okvira .NET Framework na modernu .NET platformu ... 309
Da li možete da prenesete aplikaciju? ... 309
Da li bi trebalo da prenesete aplikaciju? ... 310
Razlike između radnog okvira .NET Framework i moderne .NET platforme ... 311
Razumevanje .NET Portability Analyzer alatke ... 311
Razumevanje .NET Upgrade Assistant alatke ... 311
Upotreba biblioteka koje nisu deo .NET Standard specifikacije ... 312
Upotreba funkcija za pregled ... 313
Zahtevanje funkcija za pregled ... 314
Omogućavanje funkcija za pregled ... 314
Generička matematika ... 315
Vežbanje i istraživanje ... 315
Vežba 7.1 – Testirajte svoje znanje ... 315
Vežba 7.2 – Istražite teme ... 316
Vežba 7.3 - Istražite PowerShell ... 316
Rezime ... 316

POGLAVLJE 8

Upotreba uobičajenih .NET tipova 317

Upotreba brojeva . . . 318
Upotreba velikih celih brojeva . . . 318
Upotreba kompleksnih brojeva . . . 319
Razumevanje kvaterniona . . . 320
Upotreba teksta . . . 320
Preuzimanje dužine znakovnog niza . . . 320
Preuzimanje karaktera znakovnog niza . . . 321
Razdvanjanje znakovnog niza . . . 321
Preuzimanje dela znakovnog niza . . . 322
Provera sadržaja znakovnog niza . . . 323
Spajanje, formatiranje i drugi članovi znakovnog niza . . . 323
Efikasna izgradnja znakovnih nizova . . . 324
Upotreba datuma i vremena . . . 325
Specifikovanje vrednosti za datum i vreme . . . 325
Globalizacija pomoću datuma i vremena . . . 327
Upotreba samo datuma ili samo vremena . . . 329
Poklapanje obrazaca pomoću regularnih izraza . . . 330
Provera cifara unetih kao tekst . . . 330
Poboljšanje performansi regularnog izraza . . . 331
Razumevanje sintakse regularnog izraza . . . 332
Primeri regularnih izraza . . . 332
Razdvajanje kompleksnog znakovnog niza razdvojenog zarezima . . . 333
Skladištenje više objekata u kolekcije . . . 334
Zajedničke funkcije za sve kolekcije . . . 335
Poboljšanje performansi obezbeđivanjem kapaciteta kolekcije . . . 336
Razumevanje izbora kolekcije . . . 337
Liste . . . 337
Rečnici . . . 338
Stekovi . . . 339
Redovi čekanja . . . 339
Skupovi . . . 340
Rezime metoda kolekcije . . . 340
Upotreba lista . . . 340
Upotreba rečnika . . . 342
Upotreba redova čekanja . . . 344
Sortiranje kolekcija . . . 346
Upotreba specijalizovanih kolekcija . . . 347
Upotreba kompaktnog niza vrednosti bita . . . 347
Upotreba efikasnih listi . . . 347
Upotreba nepromenljivih kolekcija . . . 347
Dobra praksa za kolekcije . . . 348
Upotreba raspona, indeksa i opsega . . . 349
Efikasnija upotreba memorije pomoću raspona . . . 349
Identifikovanje pozicija pomoću tipa Index . . . 349
Identifikovanje opsega pomoću tipa Range . . . 350

Upotreba indeksa, opsega i raspona ... 350
Upotreba mrežnih resursa ... 351
Upotreba URI-a, DNS-a i IP adresa ... 352
Pingovanje servera ... 353
Upotreba refleksije i atributa ... 354
Verzionisanje programskih sklopova ... 355
Čitanje metapodataka programskog sklopa ... 355
Kreiranje prilagođenih atributa ... 358
Izvršavanje više akcija pomoću refleksije ... 360
Upotreba slika ... 360
Internacionalizacija koda ... 362
Detektovanje i promena aktuelne kulture ... 363
Vežbanje i istraživanje ... 365
Vežba 8.1 – Testirajte svoje znanje ... 365
Vežba 8.2 – Vežbajte regularne izraze ... 366
Vežba 8.3 – Vežbajte pisanje proširenih metoda ... 366
Vežba 8.4 – Istražite teme ... 366
Rezime ... 367

POGLAVLJE 9

Upotreba fajlova, tokova i serijalizacije ... 369

Upravljanje fajl sistemom ... 369
Rukovanje međuplatformskim okruženjima i fajl sistemima ... 369
Upravljanje drajvovima ... 371
Upravljanje direktorijumima ... 372
Upravljanje fajlovima ... 374
Upravljanje putanjama ... 375
Preuzimanje informacija o fajlu ... 376
Kontrolisanje načina upotrebe fajlova ... 377
Čitanje i pisanje pomoću tokova ... 378
Razumevanje apstraktnih i konkretnih tokova ... 378
Razumevanje tokova skladišta ... 379
Razumevanje tokova funkcije ... 379
Razumevanje pomoćnih tipova toka ... 379
Pisanje u tekstualne tokove ... 380
Pisanje u XML tokove ... 381
Uklanjanje resursa fajla ... 383
Pojednostavljenje odbacivanja upotrebom iskaza using ... 385
Kompresovanje tokova ... 386
Kompresovanje pomoću Brotli algoritma ... 388
Kodiranje i dekodiranje teksta ... 390
Kodiranje znakovnih nizova kao nizova bajtova ... 391
Kodiranje i dekodiranje teksta u fajlovima ... 393
Serijalizacija grafova objekta ... 394
Serijalizacija upotrebom XML formata ... 394
Generisanje kompaktnog XML fajla ... 397

Deserijalizacija XML fajlova ... 398
Serijalizacija pomoću JSON formata ... 399
Obrada JSON fajlova visokih performansi ... 400
Kontrolisanje obrade JSON formata ... 401
Novi JSON prošireni metodi za upotrebu u HTTP odgovorima ... 404
Prelaz iz biblioteke Newtonsoft u novi JSON ... 404
Vežbanje i istraživanje ... 405
Vežba 9.1 – Testirajte svoje znanje ... 405
Vežba 9.2 – Vežbajte serijalizaciju pomoću XML formata ... 405
Vežba 9.3 – Istražite teme ... 406
Rezime ... 406

POGLAVLJE 10

Upotreba baza podataka pomoću radnog okvira Entity Framework Core ... 407

Razumevanje modernih baza podataka ... 407
Razumevanje zastarelog radnog okvira Entity Framework ... 408
Upotreba zastarele verzije radnog okvira Entity Framework 6.3 ili novijih ... 408
Razumevanje radnog okvira Entity Framework Core ... 408
Kreiranje konzolne aplikacije za upotrebu radnog okvira EF Core ... 409
Upotreba primera relacione baze podataka ... 409
Upotreba Microsoft SQL Servera za Windows ... 410
Preuzimanje i instaliranje SQL Servera ... 411
Kreiranje Northwind primera baze podataka za SQL Server ... 412
Upravljanje primerom Northwind baze podataka pomoću alatke Server Explorer ... 413
Upotreba baze podataka SQLite ... 414
Podešavanje SQLite baze podataka za macOS ... 414
Podešavanje SQLite baze podataka za Windows ... 414
Postavljanje SQLite baze podataka za druge operativne sisteme ... 414
Kreiranje Northwind primera baze podataka za SQLite ... 415
Upravljanje Northwind primerom baze podataka pomoću alatke SQLiteStudio ... 415
Podešavanje radnog okvira EF Core ... 417
Biranje provajdera baze podataka za EF Core ... 417
Povezivanje sa bazom podataka ... 417
Definisanje kontekstne klase Northwind baze podataka ... 418
Definisanje EF Core modela ... 420
Upotreba EF Core konvencija za definisanje modela ... 421
Upotreba EF Core atributa anotacije za definisanje modela ... 421
Upotreba EF Core Fluent API-ja za definisanje modela ... 423
Razumevanje zadavanja početnih vrednosti podataka korišćenjem Fluent API-ja ... 423
Izgradnja EF Core modela za Northwind tabele ... 423
Definisanje klasa entiteta Category i Product ... 424
Dodavanje tabela u kontekstnu klasu Northwind baze podataka ... 426
Podešavanje alatke dotnet-ef ... 427
Scaffolding modela korišćenjem postojeće baze podataka ... 428
Konfigurisanje predkonvencijskih modela ... 432
Slanje upita za EF Core model ... 433

Filtriranje uključenih entiteta ... 435
Unicode karakteri u Windows konzoli ... 436
Filtriranje i sortiranje proizvoda ... 437
Preuzimanje generisanog SQL iskaza ... 438
Evidentiranje radnog okvira EF Core pomoću prilagođenog provajdera za evidentiranje ... 439
Filtriranje evidencije po vrednostima specifičnim za provajdera ... 442
Evidentiranje pomoću oznaka upita ... 443
Podudaranje obrazaca pomoću Like iskaza ... 444
Definisanje globalnih filtera ... 445
Obrasci učitavanja u radnom okviru EF Core ... 446
Entiteti učitavanja u jednom koraku ... 446
Omogućavanje odloženog učitavanja ... 447
Entiteti eksplicitnog učitavanja ... 448
Manipulisanje podacima pomoću radnog okvira EF Core ... 450
Umetanje entiteta ... 450
Ažuriranje entiteta ... 452
Brisanje entiteta ... 453
Udruživanje konteksta baze podataka ... 454
Upotreba transakcija ... 454
Kontrolisanje transakcija pomoću nivoa izolovanosti ... 455
Definisanje eksplicitne transakcije ... 455
Code First EF Core modeli ... 456
Razumevanje migracija ... 463
Vežbanje i istraživanje ... 464
Vežba 10.1 – Testirajte svoje znanje ... 464
Vežba 10.2 – Vežbajte eksportovanje podataka pomoću različitih formata serijalizacije ... 464
Vežba 10.3 – Istražite teme ... 465
Vežba 10.4 – Istražite NoSQL baze podataka ... 465
Rezime ... 465

POGLAVLJE 11

Slanje upita i manipulisanje podacima upotrebom LINQ upita 467

Pisanje LINQ izraza ... 467
Šta čini LINQ? ... 468
Izgradnja LINQ izraza pomoću klase Enumerable ... 468
Razumevanje odloženog izvršavanja ... 470
Filtriranje entiteta pomoću metoda Where ... 471
Upotreba imenovanog metoda ... 473
Pojednostavljenje koda uklanjanjem eksplicitnog instanciranja delegata ... 474
Upotreba lambda izraza ... 474
Sortiranje entiteta ... 475
Sortiranje po jednom svojstvu pomoću metoda OrderBy ... 475
Sortiranje po sledećem svojstvu pomoću metoda ThenBy ... 475
Deklarisanje upita korišćenjem tipa var ili specifikovanog tipa ... 476
Filtriranje po tipu ... 476
Upotreba skupova i multiseta u LINQ upitu ... 478
Upotreba LINQ upita u radnom okviru EF Core ... 480

Izgradnja EF Core modela ... 480
Filtriranje i sortiranje sekvenci ... 483
Projektovanje sekvenci u nove tipove ... 485
Spajanje i grupisanje sekvenci ... 486
Spajanje sekvenci ... 487
Spajanje sekvenci grupisanjem ... 488
Agregacija sekvenci ... 490
Zaslađivanje LINQ sintakse pomoću sintaksnog slatkiša ... 491
Upotreba višestrukih programskih niti pomoću funkcije Parallel LINQ ... 492
Kreiranje aplikacije koja koristi više programskih niti ... 492
Upotreba Windows-a ... 494
Upotreba macOS sistema ... 494
Za sve ostale operativne sisteme ... 494
Kreiranje sopstvenih LINQ proširenih metoda ... 495
Isprobavanje proširenog metoda koji može da se ulančа ... 498
Isprobavanje metoda Mode i Median ... 498
Upotreba LINQ to XML provajdera ... 499
Generisanje XML formata pomoću provajdera LINQ to XML ... 499
Čitanje XML formata pomoću provajdera LINQ to XML ... 500
Vežbanje i istraživanje ... 501
Vežba 11.1 – Testirajte svoje znanje ... 501
Vežba 11.2 – Vežbajte slanje upita pomoću LINQ sintakse ... 502
Vežba 11.3 – Istražite teme ... 503
Rezime ... 503

POGLAVLJE 12

Poboljšanje performansi i skalabilnosti upotrebom višeprogramskog rada ... 505

Razumevanje procesa, programskih niti i zadataka ... 505
Praćenje performansi i upotrebe resursa ... 506
Procena efikasnosti tipova ... 506
Praćenje performansi i memorije pomoću dijagnostike ... 507
Korisni članovi tipova Stopwatch i Process ... 508
Implementiranje klase Recorder ... 508
Merenje efikasnosti obrade znakovnih nizova ... 510
Nadgledanje performansi i memorije pomoću biblioteke Benchmark.NET ... 512
Asinhrono pokretanje zadataka ... 516
Sinhrono pokretanje više akcija ... 516
Asinhrono pokretanje više akcija pomoću zadataka ... 518
Pokretanje zadataka ... 518
Čekanje zadataka ... 519
Upotreba wait metoda u zadacima ... 519
Drugi zadatak - nastavak ... 520
Ugnežđeni zadaci i zadaci potomci ... 522
Omotavanje zadataka oko drugih objekata ... 523
Sinhronizovanje pristupa deljenim resursima ... 524

Pristupanje resursu iz više programskih niti ... 525
Primena međusobno isključive blokade za školjku ... 526
Razumevanje iskaza lock ... 527
Izbegavanje kružne blokade ... 528
Sinhronizovanje događaja ... 529
Učinite CPU operacije atomskim ... 530
Primena drugih tipova sinhronizacije ... 531
Razumevanje ključnih reči async i await ... 532
Poboljšanje prilagodljivosti za konzolne aplikacije ... 532
Poboljšanje prilagodljivosti za GUI aplikacije ... 533
Poboljšanje skalabilnosti za veb aplikacije i veb servise ... 537
Uobičajeni tipovi koji podržavaju višeprogramski rad ... 537
Upotreba ključne reči await u blokovima catch ... 537
Upotreba ključne reči async u tokovima ... 538
Vežbanje i istraživanje ... 539
Vežba 12.1 – Testirajte svoje znanje ... 539
Vežba 12.2 – Istražite teme ... 539
Rezime ... 540

POGLAVLJE 13

Praktična primena jezika C# i .NET platforme ... 541

Razumevanje modela aplikacija za C# i .NET ... 542
Izgradnja veb sajtova upotrebom sistema za upravljanje sadržajem ... 542
Izgradnja veb aplikacija korišćenjem SPA radnih okvira ... 543
Izgradnja veb servisa i drugih servisa ... 544
Izgradnja mobilnih i desktop aplikacija ... 545
Alternative za .NET MAUI ... 545
Razumevanje Uno platforme ... 545
Razumevanje Avalonia platforme ... 546
Nove funkcije u radnom okviru ASP.NET Core ... 546
ASP.NET Core 1.0 ... 546
ASP.NET Core 1.1 ... 546
ASP.NET Core 2.0 ... 546
ASP.NET Core 2.1 ... 547
ASP.NET Core 2.2 ... 547
ASP.NET Core 3.0 ... 548
ASP.NET Core 3.1 ... 548
Blazor WebAssembly 3.2 ... 548
ASP.NET Core 5.0 ... 548
ASP.NET Core 6.0 ... 548
Izgradnja desktop aplikacija samo za Windows sisteme ... 549
Razumevanje zastarelih platformi za Windows aplikacije ... 549
Razumevanje podrške moderne .NET platforme za zastarele Windows platforme ... 550
Strukturiranje projekata ... 550
Strukturiranje projekata u rešenju ili radnom prostoru ... 551
Upotreba drugih projektnih šablona ... 552
Instaliranje dodatnih paketa šablona ... 552

Izgradnja modela podataka entiteta za Northwind bazu podataka 553
Kreiranje biblioteke klase za modele entiteta korišćenjem SQLite baze podataka 554
Poboljšanje mapiranja klase-u-tabelu 555
Kreiranje biblioteke klase za kontekst Northwind baze podataka 559
Kreiranje biblioteke klase za modele entiteta korišćenjem SQL Server baze podataka 562
Vežbanje i istraživanje 565
Vežba 13.1 - Testirajte svoje znanje 565
Vežba 13.2 - Istražite teme 565
Rezime 565

POGLAVLJE 14

Izgradnja veb sajtova upotrebom radnog okvira ASP.NET Core Razor Pages 567

Razumevanje veb razvoja 567
Razumevanje HTTP protokola 568
Razumevanje komponenti URL adrese 568
Dodela brojeva portova za projekte u ovoj knjizi 569
Upotreba pretraživača Google Chrome za izvršavanje HTTP zahteva 569
Razumevanje tehnologija veb razvoja na strani klijenta 572
Razumevanje radnog okvira ASP.NET Core 572
Klasičan ASP.NET, nasuprot modernog radnog okvira ASP.NET Core 573
Kreiranje praznog ASP.NET Core projekta 574
Testiranje i obezbeđivanje veb sajta 576
Omogućavanje bolje bezbednosti i preusmeravanje na bezbednu konekciju 579
Kontrolisanje hosting okruženja 580
Razdvajanje konfiguracije za servise i pipeline 582
Omogućavanje veb sajtu da prikazuje statički sadržaj 584
Kreiranje direktorijuma za statičke fajlove i veb stranicu 584
Omogućavanje statičkih i podrazumevanih fajlova 585
Istraživanje ASP.NET Core Razor Pages funkcije 586
Omogućavanje Razor Pages funkcije 586
Dodavanje koda u Razor Pages stranicu 587
Upotreba deljenih rasporeda pomoću Razor Pages funkcije 588
Upotreba fajlova s pozadinskim kodom u Razor Pages funkciji 591
Upotreba radnih okvira Entity Framework Core i ASP.NET Core 593
Konfigurisanje Entity Framework Core radnog okvira kao servisa 593
Manipulisanje podacima upotrebom funkcije Razor Pages 596
Omogućavanje modelu da umeće entitete 596
Definisanje obrasca za unos novog dobavljača 597
Injektiranje servisa zavisnosti u Razor Page 597
Upotreba biblioteka klase Razor 598
Kreiranje biblioteke klase Razor 598
Onemogućavanje kompaktnih direktorijuma za Visual Studio Code 599
Implementiranje funkcije employees korišćenjem radnog okvira EF Core 600
Implementiranje delimičnog prikaza za prikaz jednog zaposlenog 602
Upotreba i testiranje biblioteke klase Razor 603
Konfigurisanje servisa i kanala za podatke HTTP zahteva 604

Razumevanje rutiranja krajnje tačke 604
Konfigurisanje rutiranja krajnje tačke 605
Pregled konfiguracije rutiranja krajnje tačke u projektu 605
Registracija servisa u metodu ConfigureServices 606
Podešavanje kanala za podatke HTTP zahteva u Configure metodu 608
Rezimiranje ključnih posredničkih proširenih metoda 609
Vizuelizacija HTTP pipeline-a 610
Implementiranje anonimnog umetnutog delegata kao posredničkog 610
Vežbanje i istraživanje 612
Vežba 14.1 – Testirajte svoje znanje 612
Vežbe 14.2 - Vežbajte izgradnju veb stranice vođene podacima 613
Vežba 14.3 - Vežbajte izgradnju veb stranica za konzolnu aplikaciju 613
Vežba 14.4 – Istražite teme 613
Rezime 613

POGLAVLJE 15

Izgradnja veb sajtova upotrebom obrasca Model-View-Controller 615

Podešavanje ASP.NET Core MVC veb sajta 615
Kreiranje ASP.NET Core MVC veb sajta 616
Kreiranje baze podataka za autentikaciju za SQL Server LocalDB 617
Razumevanje registracije posetioca 619
Pregled strukture projekta MVC veb sajta 620
Pregled ASP.NET Core Identity baze podataka 622
Istraživanje ASP.NET Core MVC veb sajta 622
Razumevanje ASP.NET Core MVC inicijalizacije 622
Razumevanje podrazumevane MVC rute 625
Razumevanje kontrolera i akcija 626
Razumevanje klase ControllerBase 626
Razumevanje klase Controller 627
Razumevanje odgovornosti kontrolera 628
Razumevanje konvencije putanje pretrage prikaza 629
Razumevanje evidentiranja 630
Razumevanje filtera 631
Upotreba filtera za obezbeđivanje metoda akcije 631
Omogućavanje programskog upravljanja ulogama i kreiranja uloge 632
Upotreba filtera za keširanje odgovora 635
Upotreba filtera za definisanje prilagođene rute 636
Razumevanje entiteta i modela prikaza 637
Razumevanje prikaza 640
Prilagođavanje ASP.NET Core MVC veb sajta 643
Definisanje prilagođenih stilova 643
Podešavanje slika za kategoriju 643
Razumevanje Razor sintakse 643
Definisanje tipiziranog prikaza 644
Pregled prilagođene Home stranice 647
Prosleđivanje parametara pomoću vrednosti rute 648

Razumevanje povezivača modela ... 650
Rešavanje dvosmislenosti metoda akcije ... 652
Prosleđivanje parametra rute ... 654
Prosleđivanje parametra obrasca ... 654
Validacija modela ... 654
Razumevanje pomoćnih metoda prikaza ... 657
Slanje upita za bazu podataka i upotreba šablona prikaza ... 659
Poboljšanje skalabilnosti upotrebom asinhronih zadataka ... 662
Učinite metode akcije kontrolera asinhronim ... 662
Vežbanje i istraživanje ... 663
Vežba 15.1 - Testirajte svoje znanje ... 663
Vežba 15.2 - Vežbajte implementiranje MVC obrasca implementiranjem stranice o detaljima kategorije ... 664
Vežba 15.3 – Vežbajte poboljšanje skalabilnosti razumevanjem i implementiranjem asinhronih metoda akcije ... 664
Vežba 15.4 – Vežbajte jedinično testiranje MVC kontrolera ... 665
Vežba 15.5 - Istražite teme ... 665
Rezime ... 665

POGLAVLJE 16

Izgradnja i upotreba veb servisa ... 667

Izgradnja veb servisa pomoću ASP.NET Core Web API-ja ... 667
Razumevanje akronima veb servisa ... 668
Radni okvir Windows Communication Foundation (WCF) ... 668
Alternativa za WCF ... 668
Razumevanje HTTP zahteva i odgovora za Web API-je ... 669
Kreiranje ASP.NET Core Web API projekta ... 671
Pregled funkcionalnosti veb servisa ... 674
Kreiranje veb servisa za Northwind bazu podataka ... 675
Kreiranje skladišta podataka za entitete ... 677
Implementiranje Web API kontrolera ... 681
Razumevanje vraćenih tipova metoda akcije ... 681
Konfigurisanje skladišta kupaca i Web API kontrolera ... 683
Specifikovanje detalja o problemu ... 687
Kontrolisanje XML serijalizacije ... 688
Dokumentovanje i testiranje veb servisa ... 688
Testiranje GET zahteva pomoću pretraživača ... 688
Testiranje HTTP zahteva pomoću REST Client ekstenzije ... 690
Kreiranje GET zahteva korišćenjem ekstenzije REST Client ... 690
Kreiranje drugih zahteva korišćenjem ekstenzije REST Client ... 692
Razumevanje Swagger alata ... 693
Testiranje zahteva pomoću Swagger UI funkcije ... 694
Omogućavanje HTTP evidentiranja ... 700
Upotreba servisa pomoću HTTP klijenata ... 702
Razumevanje klase HttpClient ... 702
Konfigurisanje HTTP klijenata upotrebom klase HttpClientFactory ... 702
Preuzimanje podataka o kupcima u JSON formatu u kontroleru ... 703

Omogućavanje Cross-Origin Resource Sharing (CORS) standarda 705
Implementiranje naprednih funkcija za veb servise 707
Implementiranje Health Check API-ja 708
Implementiranje Open API analizatora i konvencija 709
Implementiranje privremenog rukovanja greškama 709
Dodavanje bezbednosnih HTTP zaglavlja 710
Izgradnja veb servisa korišćenjem minimalnih API-ja 711
Izgradnja servisa za vremensku prognozu korišćenjem minimalnih API-ja 712
Testiranje minimalnog servisa za vremensku prognozu 714
Dodavanje vremenske prognoze na početnu stranicu Northwind veb sajta 714
Vežbanje i istraživanje 717
Vežba 16.1 – Testirajte svoje znanje 717
Vežba 16.2 - Vežbajte kreiranje i brisanje kupaca pomoću klase HttpClient 717
Vežba 16.3 - Istražite teme 717
Rezime 718

POGLAVLJE 17

Izgradnja korisničkog interfejsa pomoću radnog okvira Blazor 719

Razumevanje radnog okvira Blazor 719
JavaScript i prijatelji 720
Silverlight - C# i .NET koriste plagin 720
WebAssembly - cilj za Blazor 720
Razumevanje Blazor hosting modela 720
Razumevanje Blazor komponenti 721
Koja je razlika između radnih okvira Blazor i Razor? 722
Poređenje projektnih šablona radnog okvira Blazor 723
Pregled projektnih šablona Blazor Servera 723
Razumevanje CSS i JavaScript izolacije 729
Razumevanje Blazor rutiranja do komponenata stranice 729
Definisanje rutabilne komponente stranice 729
Kako se kretati Blazor rutama 729
Kako proslediti parametre rute 730
Razumevanje osnovnih klasa komponente 730
Korišćenje komponente navigacionog linka sa rutama 732
Pokretanje Blazor Server projektnog šablona 732
Pregled Blazor WebAssembly projektnog šablona 733
Izgradnja komponenti pomoću Blazor Server komponente 737
Definisanje i testiranje jednostavne komponente 737
Pretvaranje komponente u rutabilnu komponentu stranice 738
Ubacivanje entiteta u komponentu 739
Apstrahovanje servisa za Blazor komponentu 742
Definisanje obrazaca pomoću komponente EditForm 745
Izgradnja i korišćenje komponente obrasca za kupca 746
Testiranje komponente obrasca za kupce 749
Izgradnja komponenti pomoću BlazoraWebAssembly šablona 750
Konfigurisanje servera za Blazor WebAssembly 751
Konfigurisanje klijenta za Blazor WebAssembly 754

Testiranje Blazor WebAssembly komponenti i servisa ... 757
Poboljšanje Blazor WebAssembly aplikacija ... 758
Omogućavanje Blazor WebAssembly AOT kompajlera ... 759
Istraživanje podrške za progresivne veb aplikacije ... 760
Implementacija oflajn podrške za PWA ... 762
Razumevanje analizatora kompatibilnosti pretraživača za Blazor WebAssembly ... 762
Deljenje Blazor komponenti u biblioteci klasa ... 763
Interoperabilnost sa JavaScript bibliotekama ... 765
Biblioteke Blazor komponenti ... 767
Vežbanje i istraživanje ... 767
Vežba 17.1 – Proverite svoje znanje ... 768
Vežba 17.2 – Kreiranje komponente za tablicu množenja ... 768
Vežba 17.3 – Kreiranje navigacione stavke za državu ... 768
Vežba 17.4 – Istražite teme ... 769
Rezime ... 769

Epilog ... 771

Sledeći koraci na vašem putu učenja jezika C# i .NET platforme ... 771
Poboljšajte veštine smernicama za dizajn ... 771
Knjige za dalje učenje ... 772
.NET MAUI odložen ... 773
Sledeće izdanje izlazi u novembru 2022. godine ... 773
Srećno! ... 773
Podelite svoje mišljenje ... 773

DODATAK

Odgovori na pitanja za testiranje znanja ... 775

Poglavlje 1 – Zdravo, C#! Dobrodošao, .NET! ... 775
Poglavlje 2 – Govorite C# jezikom ... 777
Vežba 2.1 – Testirajte svoje znanje ... 777
Vežba 2.2 – Testirajte svoje znanje o numeričkim tipovima ... 778
Poglavlje 3 – Kontrola toka, konvertovanje tipova i rukovanje izuzecima ... 779
Vežba 3.1 – Proverite svoje znanje ... 779
Vežba 3.2 – Istražite petlje i prekoračenja ... 780
Vežba 3.5 – Proverite znanje o operatorima ... 780
Poglavlje 4 – Pisanje funkcija, ispravljanje grešaka i testiranje funkcija ... 781
Poglavlje 5 – Izgradnja sopstvenih tipova pomoću objektno-orijentisanog programiranja ... 782
Poglavlje 6 – Implementiranje interfejsa i nasleđivanje klasa ... 783
Poglavlje 7 – Pakovanje i distribucija .NET tipova ... 785
Poglavlje 8 – Upotreba uobičajenih .NET tipova ... 786
Poglavlje 9 – Upotreba fajlova, tokova i serijalizacije ... 787
Poglavlje 10 – Upotreba baza podataka pomoću radnog okvira EntityFramework Core ... 789
Poglavlje 11 – Slanje upita i manipulisanje podacima upotrebom LINQ upita ... 791
Poglavlje 12 – Poboljšanje performansi i skalabilnosti upotrebom višeprogramskog rada ... 793
Poglavlje 13 – Praktična primena jezika C# i .NET platforme ... 794

Poglavlje 14 – Izgradnja veb sajtova upotrebom radnog okvira
ASP.NET Core Razor Pages 794
Poglavlje 15 – Izgradnja veb sajtova upotrebom Model-View-Controller obrasca 797
Poglavlje 16 – Izgradnja i upotreba veb servisa 800
Poglavlje 17 – Izgradnja korisničkog interfejsa pomoću radnog okvira Blazor 802

INDEKS 805

Uvod

Postoji mnoštvo knjiga za programiranje koje su napisane da budu sveobuhvatne reference za C# programski jezik, .NET biblioteke i modele aplikacija, kao što su veb sajtovi, servisi i aplikacije za desktop i mobilne uređaje.

Ova knjiga je drugačija. Ona je koncizna, žustra i zabavna za čitanje i sadrži praktične vežbe koje će vas voditi kroz svaku temu. Širina sveobuhvatnog narativa ima cenu dubine, ali ćete pronaći mnoge putokaze za dalje istraživanje, ako želite.

Ova knjiga je istovremeno vodič, korak-po-korak, za učenje modernih, dokazanih tehnika jezika C# upotrebom međuplatformskog radnog okvira .NET, a takođe predstavlja kratak uvod u glavne tipove aplikacija koje možemo da gradimo. Ova knjiga je najbolja za početnike i za programere koji su već koristili jezik C# u prošlosti, ali smatraju da zaostaju u upotrebi novih funkcija zbog promena u proteklih nekoliko godina.

Ako ste već koristili starije verzije jezika C#, u prvom odeljku poglavlja 2, „Govorite C# jezikom“, možete da pregledate tabele novih funkcija jezika i da pređete direktno na njih.

Ako već imate iskustva u upotrebi starijih verzija .NET biblioteka, u prvom odeljku poglavlja 7, „Pakovanje i distribucija .NET tipova“, možete da pregledate tabele novih funkcija biblioteke i da pređete direktno na njih.

Istaći ću sve interesantne delove jezika C# i .NET platforme da biste mogli da impresionirate svoje kolege i brzo postanete produktivni. Da ne bih dosađivao nekim čitaocima objašnjavanjem svake sitnice, pretpostaviću da ste dovoljno pametni da potražite u pretraživaču Google objašnjenje za teme koje su povezane, ali nisu obavezno uključene u vodič za početnike ili programere sa malo iskustva, zbog ograničenog prostora u štampanoj knjizi.

GDE DA PRONAĐETE REŠENJA KODA

Možete da preuzmete rešenja za zadatke, vođene korak po korak, kao i vežbe iz sledećeg GitHub skladišta: https://bit.ly/34WALKH

Ako ne znate kako da to uradite, pogledajte instrukcije na kraju poglavlja 1, „Zdravo C#! Dobrodošao .NET!".

GDE DA PRONAĐETE ONLINE POGLAVLJA

Tri dodatna online poglavlja i dodatak pripadaju ovom novom izdanju, a možete da ih pročitate na adresi https://bit.ly/3se1s6r. Takođe možete da ih nađete na adresi https://bit.ly/35qaG6E.

ŠTA OBUHVATA OVA KNJIGA

Poglavlje 1, „Zdravo C#! Dobrodošao .NET!", posvećeno je podešavanju razvojnog okruženja i upotrebi uređivača koda Visual Studio Code za kreiranje najjednostavnijih mogućih aplikacija pomoću jezika C# i radnog okvira .NET. Za pojednostavljene konzolne aplikacije, videćete programske funkcije najvišeg nivoa, koje su predstavljene u verziji C# 9. Da biste naučili da pišete jednostavne jezičke konstrukcije i funkcije biblioteke, upotrebićete dodatni modul .NET Interactive Notebooks. Takođe ćete naučiti koja su najbolja mesta da potražite pomoć i načine da kontaktirate sa mnom, da bih vam pomogao da rešite problem, ili da mi pošaljete povratne informacije kako bih mogao da poboljšam knjigu i buduća izdanja pomoću njenog GitHub skladišta.

U *poglavlju 2, „Govorite C# jezikom"*, predstavićemo verzije jezika C# i tabele u kojima su prikazane verzije koje uključuju nove funkcije, a zatim ćemo objasniti gramatiku i rečnik koje ćete koristiti svakodnevno za pisanje izvornog koda za vaše aplikacije. Konkretno, naučićete kako da deklarišete i upotrebite različite tipove promenljivih.

U *poglavlju 3, „Kontrola toka, konvertovanje tipova i rukovanje izuzecima"*, biće reči o upotrebi operatora za izvršavanje jednostavnih akcija u promenljivim, uključujući poređenja, pisanje koda za donošenje odluka, podudaranje obrazaca od verzije C# 7 do verzije C# 10, ponavljanje bloka iskaza i konvertovanje između tipova. Takođe ćete upoznati odbrambeno pisanje koda za obradu izuzetaka kada se oni neizbežno javljaju.

Poglavlje 4, „Pisanje, otklanjanje grešaka i testiranje funkcija", posvećeno je praćenju principa Don't Repeat Yourself (DRY) pisanjem funkcija za višekratnu upotrebu korišćenjem i imperativnog stila i stilova funkcionalne implementacije. Naučićete kako da upotrebite alatke za otklanjanje grešaka da biste pratili i otklanjali programske greške, da biste nadgledali kod dok se izvršava, da biste dijagnostifikovali probleme i da biste strogo testirali kod i otklonili programske greške i obezbedili stabilnost i pouzdanost pre nego što je kod raspoređen u proizvodnju.

Poglavlje 5, „Izgradnja sopstvenih tipova upotrebom objektno-orijentisanog programiranja“, posvećeno je različitim kategorijama članova koje tipovi mogu imati, uključujući polja za čuvanje podataka i metode za izvršavanje akcija. Upotrebićete koncepte **objektno-orijentisanog programiranja (OOP)**, kao što su agregacija i kapsuliranje. Učićete o funkcijama jezika, kao što su podrška za sintaksu torke i promenljive out, podrazumevani literali i izvedeni nazivi torke, a takođe ćete naučiti kako da definišete i koristite nepromenljive tipove pomoću ključne reči record, init-only svojstva i with izraza, koji su predstavljeni u verziji C# 9.

U *poglavlju 6, „Implementiranje interfejsa i nasleđivanje klasa“*, govorićemo o izvođenju novih tipova iz postojećih, upotrebom objektno-orijentisanog programiranja (OOP). Naučićete da definišete operatore i lokalne funkcije, delegate i događaje, da implementirate interfejse o osnovnim i izvedenim klasama, da promenite član tipa, da upotrebite polimorfizam, da kreirate proširene metode, da konvertujete između klasa u hijerarhiji nasleđivanja, a učićete i o velikoj promeni u verziji C# 8, u kojoj su predstavljeni referentni tipovi koji prihvataju null.

U *poglavlju 7, „Pakovanje i distribucija .NET tipova“*, predstavićemo verzije .NET radnog okvira i uključićemo tabele u kojima su navedene verzije u koje su dodate nove funkcije, a zatim ćemo predstaviti .NET tipove koji su usklađeni sa specifikacijom .NET Standard i načine na koji su oni povezani sa C# jezikom. Naučićete da pišete i kompajlirate kod u bilo kom podržanom operativnom sistemu: Windows, macOS i Linux. Naučićete da pakujete, raspoređujete i distribuirate sopstvene aplikacije i biblioteke.

Poglavlje 8, „Upotreba uobičajenih .NET tipova“, posvećeno je tipovima koji omogućavaju kodu da izvrši uobičajene praktične zadatke, kao što su manipulisanje brojevima i tekstom, datumom i vremenom, skladištenje stavki u kolekcijama, upotreba mreže i manipulisanje slikama i implementiranje internacionalizacije.

U *poglavlju 9, „Upotreba fajlova, nizova i serijalizacije“*, govorićemo o interakciji sa fajl sistemom, čitanju fajlova i nizova i pisanju u njih, šifrovanju teksta i o formatima serijalizacije, kao što su JSON i XML, uključujući i poboljšanu funkcionalnost i performanse System.Text.Json klasa.

U *poglavlju 10, „Upotreba podataka pomoću radnog okvira Entity Framework Core“*, objasnićemo čitanje iz relacione baze podatakai pisanje u relacionu bazu podataka, kao što su Microsoft SQL Server i SQLite, korišćenjem tehnologije **objektno-relacionog mapiranja (ORM)** pod nazivom **Entity Framework Core (EF Core)**. Naučićete da definišete modele entiteta koji se mapiraju u postojeće tabele u bazi podataka i da definišete Code First modele koji mogu da kreiraju tabele i bazu podataka tokom izvršenja.

U *poglavlju 11, „Slanje upita i manipulisanje podacima upotrebom komponente LINQ“*, biće reči o **Language INtegrated Queries (LINQ)** – ekstenzijama jezika koje dodaju mogućnost upotrebe sekvenci stavki i filtriranja, sortiranja i projektovanja stavki u različite ispise. Učićete o specijalnim mogućnostima komponenata **Parallel LINQ (PLINQ)** i LINQ to XML.

U *poglavlju 12, „Poboljšavanje performansi i skalabilnosti korišćenjem višeprogramskog rada“*, objasnićemo omogućavanje izvršenja više akcija istovremeno radi poboljšanja performansi, skalabilnosti i produktivnosti korisnika. Učićete o funkciji async Main i o upotrebi tipova u imenskom prostoru System.Diagnostics za nadgledanje koda i merenje performansi i efikasnosti.

U *poglavlju 13, „Praktična primena jezika C# i radnog okvira .NET“*, predstavićemo tipove međuplatformskih aplikacija koje možete da izgradite pomoću jezika C# i radnog okvira .NET. Takođe ćete izgraditi EF Core model za predstavljanje Northwind baze podataka, koju ćete koristiti u ostalim poglavljima knjige.

U *poglavlju 14, „Izgradnja veb sajtova upotrebom radnog okvira ASP.NET Core Razor Pages“*, upoznaćete osnove izgradnje veb sajtova pomoću moderne HTTP arhitekture na strani servera korišćenjem radnog okvira ASP.NET Core. Naučićete da implementirate ASP.NET Core funkciju, poznatu kao Razor Pages, koja pojednostavljuje kreiranje dinamičkih veb stranica za male veb sajtove. Takođe ćete učiti o izgradnji protočne obrade (eng. pipeline) HTTP zahteva i odgovora.

U *poglavlju 15, „Kreiranje veb sajtova upotrebom Model-View-Controller obrasca“*, naučićete da izgradite velike, složene veb sajtove tako što će jedinično testiranje i upravljanje timovima programera biti olakšano upotrebom radnog okvira ASP.NET Core MVC. Učićete o početnoj konfiguraciji, autentikaciji, rutama, modelima, prikazima i kontrolerima.

U *poglavlju 16, „Kreiranje veb servisa i njihova upotreba“*, objasnićemo izgradnju veb servisa backend REST arhitekture korišćenjem radnog okvira ASP.NET Core Web API i opisaćemo kako da ih pravilno koristite pomoću fabrički instanciranih HTTP klijenata.

U *poglavlju 17, „Izgradnja korisničkih interfejsa pomoću radnog okvira Blazor“*, predstavljamo kako da izgradite komponente korisničkog veb interfejsa korišćenjem radnog okvira Blazor, koji može da se izvrši ili na strani servera ili unutar veb pretraživača na strani klijenta. Videćete razlike između Blazor Server i Blazor Web Assembly radnih okvira i objasnićemo kako da izgradite komponente koje je lakše menjati između dva hosting modela.

Tri dodatna pdf poglavlja na engleskom jeziku, kompletiraju ovo novo izdanje. Možete da ih preuzmete sa: https://bit.ly/3qhHupJ.

U *poglavlju 18, „Izgradnja i upotreba specijalizovanih servisa“*, predstavićemo izgradnju servisa korišćenjem radnog okvira gRPC, implementiranje komunikacije u realnom vremenu između servera i klijenta korišćenjem biblioteke SignalR, izlaganje EF Core modela korišćenjem OData protokola i hostovanje funkcija u cloud-u, koje odgovaraju na pokretače, pomoću servisa Azure Functions.

U *poglavlju 19, „Izgradnja mobilnih i desktop aplikacija korišćenjem radnog okvira .NET MAUI“*, predstavićemo izgradnju međuplatformskih mobilnih i desktop aplikacija za Android, iOS, macOS i Windows. Učićete o osnovama jezika XAML, koji možete da koristite za definisanje korisničkog interfejsa za grafičku aplikaciju.

U *poglavlju 20, „Zaštita podataka i aplikacija“*, biće reči o zaštiti podataka, odnosno kako da upotrebom enkripcije sprečite zlonamerne korisnike da ih pregledaju i kako pomoću heširanja i potpisivanja da ih sprečite da oštete podatke i manipulišu njima. Takođe ćete učiti o autentikaciji i autorizaciji da biste zaštitili aplikacije od neautorizovanih korisnika.

Dodatak „Odgovori na pitanja za testiranje znanja“ sadrži odgovore na test pitanja koja se nalaze na kraju svakog poglavlja.

ŠTA VAM JE POTREBNO ZA OVU KNJIGU?

Možete da razvijate i raspoređujete C# i .NET Core aplikacije upotrebom uređivača koda Visual Studio Code na mnogim platformama, uključujući Windows, macOS i mnoge različite Linux distribucije.

Sve što vam je potrebno da završite vežbe iz svih poglavlja, osim jednog, je operativni sistem koji podržava Visual Studio Code i internet konekcija.

Ako želite, možete da koristite Visual Studio za Windows ili macOS, ili nezavisnu alatku, kao što je JetBrains Rider.

Potreban vam je macOS da biste izgradili iOS aplikaciju iz *poglavlja 19, „Izgradnja mobilnih i desktop aplikacija pomoću radnog okvira .NET MAUI“*, jer je potrebno da imate macOS i Xcode za kompajliranje iOS aplikacija.

Preuzimanje kolornih slika za ovu knjigu

Takođe smo vam obezbedili PDF fajl koji sadrži kolorne slike ekrana/dijagrama koji su upotrebljeni u ovoj knjizi. Te slike će vam pomoći da bolje razumete promene u ispisu.

Taj fajl možete da preuzmete sa adrese

https://bit.ly/3BNqNaf.

Konvencije

U ovoj knjizi pronaći ćete više različitih stilova za tekst koje sam upotrebio za različite vrste informacija. Evo nekih primera ovih stilova i objašnjenja njihovog značenja.

Reči koda u tekstu, nazivi tabela baze podataka, nazivi direktorijuma, nazivi fajlova, ekstenzije fajla, nazivi putanja, kratki URL-ovi, korisnički unos i Twitter statusi su prikazani na sledeći način: „Direktorijumi `Controllers`, `Models` i `Views` sadrže ASP.NET `Core` klase i `.cshtml` fajlove za izvršenje na serveru“. Blok koda je postavljen na sledeći način:

```
// storing items at index positions
names[0] = "Kate";
names[1] = "Jack";
names[2] = "Rebecca";
names[3] = "Tom";
```

Da bismo usmerili vašu pažnju na određeni deo bloka koda, relevantne linije ili stavke su istaknute na sledeći način:

```
// storing items at index positions
names[0] = "Kate";
names[1] = "Jack";
names[2] = "Rebecca";
names[3] = "Tom";
```

Svi unosi ili ispisi komandne linije napisani su na sledeći način:

```
dotnet new console
```

Novi termini i važne reči su napisani **podebljanim** slovima. Reči koje vidite na ekranu - na primer, u menijima ili okvirima za dijalog, biće prikazane u tekstu na sledeći način: „Kliknite na dugme **Next** da biste se prebacili na sledeći ekran“.\



Važne napomene: linkovi ka eksternim izvorima za dalje čitanje biće prikazani u okviru kao što je ovaj.



Dobra praksa: Preporuke kako da programirate kao stručnjak prikazane su ovako.

STUPITE U KONTAKT

Povratne informacije naših čitalaca su uvek dobrodošle.

Štamparske greške: Iako smo preduzeli sve mere da bismo obezbedili tačnost sadržaja, greške su moguće. Ako pronađete neku grešku u ovoj knjizi, bili bismo zahvalni ako biste nam to javili. Otvorite stranicu http://www.packtpub.com/support/errata, selektujte knjigu, kliknite na link Errata Submission Form i unesite detalje o grešci.

1

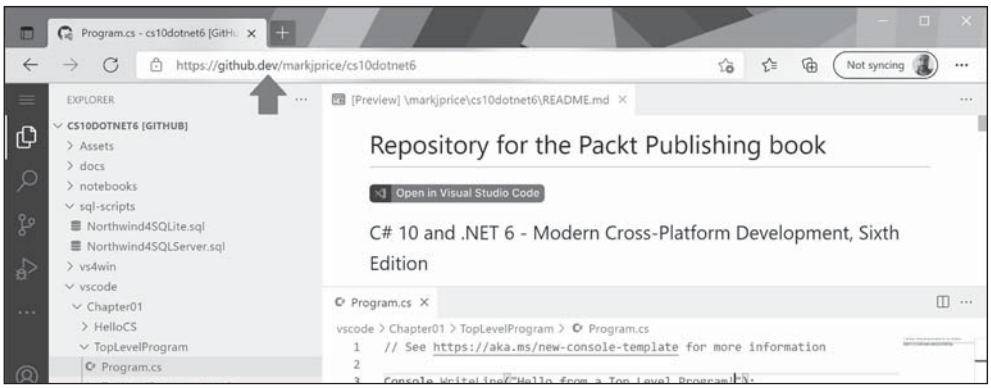
Zdravo C#! Dobrodošao .NET!

Ciljevi poglavlja 1 su podešavanje razvojnog okruženja, razumevanje sličnosti i razlika između radnih okvira .NET, .NET Core, .NET Framework, Mono, Xamarin i .NET Standard, a zatim kreiranje najjednostavnije aplikacije pomoću jezika C# 10 i radnog okvira .NET 6 korišćenjem različitih uređivača koda i otkrivanje dobrih mesta na kojima možete da potražite pomoć.

GitHub skladište za ovu knjigu sadrži rešenja koja koriste kompletne projekte aplikacije za sve zadatke vezane za kod i beleške, kada god je to moguće:

https://github.com/markjprice/cs10dotnet6

Jednostavno pritisnite taster . (tačka) ili promenite `.com` na `.dev` u prikazanom linku da biste promenili GitHub skladište u živi uređivač, korišćenjem uređivača Visual Studio Code for the Web, kao što je prikazano na *slici 1.1*:



Slika 1.1 Visual Studio Code for the Web - editovanje uživo GitHub skladišta knjige

Dobra ideja je pokrenuti Visual Studio Code for the Web zajedno sa izabranim uređivačem koda dok izvršavate zadatke kodiranja u ovoj knjizi. Možete da uporedite kod sa kodom rešenja i jednostavno da kopirate i pejstujete delove, ukoliko je potrebno.

U ovoj knjizi koristim termin **moderan .NET** za .NET 6 i njegove prethodne verzije, kao što je .NET 5, koji dolaze iz radnog okvira .NET Core. Takođe koristim termin **zastareli .NET** za .NET Framework, Mono, Xamarin i .NET Standard. Moderan .NET je objedinjavanje tih zastarelih platformi i standarda.

Nakon ovog poglavlja, knjiga može da se podeli na tri dela: prvi deo sadrži gramatiku i rečnik C# jezika, drugi deo sadrži tipove dostupne u radnom okviru .NET Core za izgradnju funkcija aplikacije, a treći deo sadrži primere uobičajenih međuplatformskih aplikacija koje možete da izgradite pomoću jezika C# i platforme .NET.

Većina korisnika uči komplikovane teme imitacijom i ponavljanjem, umesto da pročita detaljno objašnjenje teorije; prema tome, ja neću objašnjavati detaljno svaki korak u ovoj knjizi. Cilj je da vas naučim da napišete kod i da vidite da se on pokreće.

Nije potrebno da odmah znate sve detalje. To je nešto što ćete saznavati vremenom dok budete gradili svoje aplikacije i budete izašli izvan okvira onoga što bilo koja knjiga može da vas nauči.

Sudeći po knjizi „English dictionary iz 1755“, čiji je autor Samuel Johnson, verovatno sam napravio nekoliko kardinalnih grešaka i napisao nekoliko apsurdnih izraza, od kojih ni jedan posao ovakvog opsega nije oslobođen. Preuzimam potpunu odgovornost za greške i nadam se da ćete ceniti izazov mog pokušaja da „podignem prašinu“ pisanjem ove knjige o tehnologijama koje se veoma brzo razvijaju, kao što su C# i .NET Core, i o aplikacijama koje pomoću njih možete da izgradite.

Ovo poglavlje obuhvata sledeće teme:

- podešavanje razvojnog okruženja
- razumevanje .NET platforme
- izgradnja konzolnih aplikacija upotrebom razvojnog okruženja Visual Studio 2022
- izgradnja konzonih aplikacija upotrebom uređivača koda Visual Studio Code
- istraživanje koda pomoću ekstenzije .NET Interactive Notebooks
- pregled direktorijuma i fajlova za projekte
- dobra upotreba GitHub skladišta za ovu knjigu
- traženje pomoći

PODEŠAVANJE RAZVOJNOG OKRUŽENJA

Pre nego što počnete da programirate, potreban vam je editor koda za jezik C#. Microsoft ima familiju uređivača koda i integrisanih razvojnih okruženja (**Integrated Development Environment - IDE**), koja uključuje:

- Visual Studio 2022 for Windows
- Visual Studio 2022 for Mac

- Visual Studio Code for Windows, Mac, ili Linux
- GitHub Codespaces

Nezavisni proizvođači su kreirali sopstvene uređivače koda za jezik C#, na primer, JetBrains Rider.

Izbor odgovarajućeg alata i tipa aplikacije za učenje

Koji je najbolji alat i tip aplikacije za učenje jezika C# i .NET platforme?

Kada učite, najbolji alat je onaj koji vam pomaže da napišete kod i konfiguraciju, ali ne krije šta se zaista dešava. IDE obezbeđuje grafičke korisničke interfejse koji su jednostavni za korišćenje, ali šta oni zapravo rade za vas? Dok učite, mnogo je bolji osnovniji uređivač koda, koji je bliži akciji, dok pruža pomoć pri pisanju koda.

Prema tome, možete da iznesete argument da je najbolji alat onaj koji vam je već poznat ili koji ćete vi ili vaš tim koristiti kao svakodnevni razvojni alat. Iz tog razloga, slobodno izaberite bilo koji uređivač koda za jezik C# ili IDE, da biste dovršili zadatke kodiranja u ovoj knjizi, uključujući Visual Studio Code, Visual Studio for Windows, Visual Studio for Mac, ili čak JetBrains Rider.

U trećem izdanju ove knjige dao sam detaljna uputstva, korak po korak, i za Visual Studio for Windows i za Visual Studio Code, za sve zadatke kodiranja. Nažalost, to je brzo postalo neuredno i zbunjujuće. U ovom šestom izdanju dajem detaljna uputstva, korak po korak, o tome kako da kreirate više projekata u razvojnom okruženju Visual Studio 2022 for Windows i u uređivaču koda Visual Studio Code, samo u *poglavlju 1*. Nakon toga, dajem nazive projekata i opšta uputstva koja funkcionišu sa svim alatima, tako da možete da koristite alat koji želite.

Najbolji tip aplikacije za učenje konstrukcija jezika C# i mnogih .NET biblioteka je onaj koji ne odvlači pažnju nepotrebnim kodom aplikacije. Na primer, nema potrebe da kreirate celu Windows desktop aplikaciju ili veb sajt samo da biste naučili da napišete iskaz `switch`.

Iz tog razloga, verujem da je izgradnja konzolnih aplikacija najbolji metod za učenje tema za jezik C# i .NET platformu u *poglavljima 1 -12*. Zatim ćete u *poglavljima 13-19* graditi veb sajtove, servise i grafičke aplikacije za desktop i mobilne uređaje.

Prednosti i nedostaci ekstenzije .NET Interactive Notebooks

Još jedna prednost uređivača koda Visual Studio Code je .NET Interactive Notebooks ekstenzija. Ta ekstenzija pruža jednostavno i bezbedno mesto za pisanje jednostavnih isečaka koda. Omogućava vam da kreirate jedan fajl beležnice koji meša „ćelije“ Markdown (bogato formatiranog teksta) i koda korišćenjem jezika C# i drugih srodnih jezika, kao što su PowerShell, F# i SQL (za baze podataka).

Međutim, ekstenzija .NET Interactive Notebooks ima neka ograničenja:

- Ne može da čita unos korisnika, na primer, ne možete da koristite `ReadLine` ili `ReadKey`.
- Ne možete da joj prosledite argument.
- Ne dozvoljava da definišete sopstvene imenske prostore.

- Nema alate za otklanjanje grešaka (ali će biti dodati u budućnosti).

Upotreba uređivača koda Visual Studio Code za međuplatformski razvoj

Najmoderniji i najjednostavniji uređivač koda koji možete da izaberete i jedini iz Microsofta koji je međuplatformski je Microsoft Visual Studio Code. On može da se pokreće na svim uobičajenim operativnim sistemima, uključujući Windows, macOS i mnoge distribucije Linux operativnog sistema, uključujući Red Hat Enterprise Linux (RHEL) i Ubuntu.

Visual Studio Code je dobar izbor za moderan međuplatformski razvoj jer ima obiman i rastući skup ekstenzija za podršku mnogih jezika, pored jezika C#.

Pošto je međuplatformski i jednostavan, može da se instalira na svim platformama na kojima će aplikacije biti raspoređene, za brzo otklanjanje programskih grešaka i izvršavanje nekih drugih zadataka.Upotreba uređivača koda Visual Studio Code podrazumeva da programer može da koristi međuplatformski uređivač koda za razvoj međuplatformskih aplikacija.

Visual Studio Code ima jaku podršku za veb razvoj, iako trenutno ima slabu podršku za razvoj aplikacija za desktop i mobilne uređaje.

Visual Studio Code je podržan na ARM procesorima tako da možete da razvijate aplikacije na računarima Apple Silicon i Raspberry Pi.

Prema anketi Stack Overflow 2021, Visual Studio Code je najpopularnije integrisano razvojno okruženje i izbor je više od 70% profesionalnih programera.

Upotreba razvojnog okruženja GitHub Codespaces za razvoj u cloud platformi

GitHub Codespaces je potpuno konfigurisano razvojno okruženje zasnovano na razvojnom okruženju Visual Studio Code, koje može da se pokreće u okruženju hostovanom u cloud platformi, a pristupa mu se putem bilo kog veb pretraživača. Podržava Git skladišta, ekstenzije i ugrađen interfejs komandne linije, pa možete da editujete, pokrećete i testirate kod sa bilo kog uređaja.

Upotreba razvojnog okruženja Visual Studio for Mac za opšti razvoj

Korišćenjem razvojnog okruženja Microsoft Visual Studio 2022 for Mac možete da kreirate većinu tipova aplikacija, uključujući konzolne aplikacije, veb sajtove, veb servise i aplikacije za desktop i mobilne uređaje.

Da biste kompajlirali aplikacije za Apple operativne sisteme, kao što je iOS, za pokretanje na uređajima kao što su iPhone i iPad, potrebno je da imate Xcode, koji se pokreće samo na macOS operativnom sistemu.

Upotreba razvojnog okruženja Visual Studio for Windows za opšti razvoj

Korišćenjem razvojnog okruženja Microsoft Visual Studio 2022 for Windows možete da kreirate većinu tipova aplikacija, uključujući konzolne aplikacije, veb sajtove, veb servise i aplikacije za desktop i mobilne uređaje. Iako možete da koristite Visual Studio 2022 for Windows sa njegovom ekstenzijom Xamarin za pisanje međuplatformskih aplikacija za mobilne uređaje i dalje vam je potreban macOS i Xcode da biste kompalirali te aplikacije.

Visual Studio 2022 for Windows se pokreće samo na Windows verziji 7 SP1, ili novijoj. Potrebno je da ga pokrenete na verziji Windows 10 ili Windows 11 da biste kreirali **Universal Windows Platform (UWP)** aplikacije, koje se instaliraju iz aplikacije Microsoft Store i pokreću se u izolovanom režimu sandbox radi zaštite računara.

Šta sam ja upotrebio?

Za pisanje i testiranje koda za ovu knjigu ja sam upotrebio sledeći hardver:

- HP Spectre (Intel) laptop
- Apple Silicon Mac mini (M1) dekstop
- Raspberry Pi 400 (ARM v8) desktop

A koristio sam sledeći softver:

- Visual Studio Code na:
- macOS sistemu - Apple Silicon Mac mini (M1) desktopu
- Windows 10 sistemu - HP Spectre (Intel) laptopu
- Ubuntu 64 sistemu - Raspberry Pi 400
- Visual Studio 2022 for Windows na:
- Windows 10 sistemu - HP Spectre (Intel) laptopu
- Visual Studio 2022 for Mac na:
- macOS sistemu - Apple Silicon Mac mini (M1) desktopu

Nadam se da i vi imate pristup različitom hardveru i softveru, jer ako vidite razlike u platformama to će pomoći da bolje razumete izazove razvoja, mada je bilo koja kombinacija navedenih stavki dovoljna da naučite osnove jezika C# i .NET platforme i da izgradite praktične aplikacije i veb sajtove.



Više informacija: Čitanjem dodatnog članka možete da naučite da pišete kod korišćenjem jezika C# i .NET platforme pomoću Raspberry Pi 400 sistema sa instaliranim Ubuntu Desktop 64-bitnim operativnim sistemom, a pronaći ćete ga na sledećem linku: https://github.com/markjprice/cs9dotnet5-extras/blob/main/raspberry-pi-ubuntu64/README.md.

Međuplatformsko raspoređivanje

Vaš izbor uređivača koda i operativnog sistema za razvoj neće ograničiti mesto gde će kod biti raspoređen.

.NET 6 podržava sledeće platforme za raspoređivanje:

- **Windows:** Windows 7 SP1, ili noviju, Windows 10 verziju 1607, ili noviju, uključujući Windows 11. Windows Server 2012 R2 SP1, ili noviji, Nano Server verziju 1809, ili noviju.
- **Mac:** macOS Mojave (verzija 10.14), ili novija.
- **Linux:** Alpine Linux 3.13, ili noviji. CentOS 7, ili noviju verziju. Debian 10, ili noviju. Fedora 32, ili noviju. openSUSE 15, ili noviju. Red Hat Enterprise Linux (RHEL) 7, ili noviju. SUSE Enterprise Linux 12 SP2, ili noviju. Ubuntu 16.04, 18.4, 20.04, ili noviju.
- **Android:** API 21, ili noviju.
- **iOS:** 10, ili noviju verziju.

Windows ARM64 podrška u .NET 5 i novijoj verziji podrazumeva da razvijate aplikacije na Windows ARM uređajima, kao što je Microsoft Surface Pro X, i da raspoređujete aplikacije na njima. Ali razvoj na Apple M1 Mac sistemu korišćenjem alata Parallels i Windows 10 ARM virtuelne mašine, navodno je dvostruko brži!

Preuzimanje i instaliranje razvojnog okruženja Visual Studio 2022 for Windows

Mnogi profesionalni Microsoft programeri koriste Visual Studio 2022 for Windows u svom svakodnevnom radu. Čak i ako odlučite da koristite Visual Studio Code za izradu zadataka kodiranja u ovoj knjizi, možda ćete želeti da upoznate i razvojno okruženje Visual Studio 2022 for Windows.

Ako nemate Windows računar, onda možete da preskočite ovaj odeljak i da pređete na sledeći odeljak gde ćete preuzeti i instalirati Visual Studio Code na macOS ili Linux.

Od oktobra 2014. godine, Microsoft je učinio besplatno dostupnim izdanje razvojnog okruženja Visual Studio for Windows, profesionalnog kvaliteta, studentima, saradnicima otvorenog koda i pojedincima. To izdanje je pod nazivom Community Edition. Bilo koje izdanje je pogodno za izradu zadataka iz ove knjige. Ako ga još niste instalirali, uradimo to sada:

1. Preuzmite Microsoft Visual Studio 2022 verziju 17.0, ili noviju, za Windows sa sledećeg linka: https ://visualstudio.microsoft.com/downloads/.
2. Pokrenite instalacioni program.
3. Na kartici **Workloads** izaberite sledeće:
    - **ASP.NET and web development**
    - **Azure development**
    - **.NET desktop development**

- **Desktop development with C++**
- **Universal Windows Platform development**
- **Mobile development with .NET**

4. Na kartici **Individual components**, u odeljku **Code tools**, izaberite sledeće:
   - **Class Designer**
   - **Git for Windows**
   - **PreEmptive Protection - Dotfuscator**
5. Kliknite **Install** i sačekajte da instalacioni program preuzme izabrani softver i instalira ga.
6. Kada se instalacija završi, kliknite **Launch**.
7. Kada prvi put pokrenete Visual Studio od vas će biti zatraženo da se prijavite. Ako imate Microsoft nalog, možete da koristite taj nalog. Ako ga nemate, onda se registrujte za nov nalog na sledećem linku: https ://signup.live.com/.
8. Kada prvi put pokrenete Visual Studio, od vas će biti zatraženo da konfigurišete okruženje. Za **Development Settings** izaberite **Visual C#**. Za kolornu temu, ja sam odabrao **Blue**, ali možete odabrati šta god želite.
9. Ako želite da prilagodite prečice na tastaturi, kliknite na **Tools** | **Options…** , a zatim izaberite odeljak **Keyboard.**

Prečice na tastaturi za Microsoft Visual Studio for Windows

U ovoj knjizi izbegavaću prikazivanje prečica na tastaturi jer su one često prilagođene. Pokušaću da ih pokažem u slučajevima gde su one konzistentne u svim uređivačima koda i gde se često koriste. Ako želite da identifikujete i prilagodite svoje prečice na tastaturi, možete to da uradite, kao što je prikazano na sledećem linku: https://docs.microsoft.com/en-us/visualstudio/ide/identifying-and-customizing-keyboard-shortcuts-in-visual-studio.

Preuzimanje i instaliranje uređivača koda Visual Studio Code

Visual Studio Code se brzo poboljšavao tokom poslednjih nekoliko godina i prijatno je iznenadio Microsoft svojom popularnošću. Ako ste hrabri i volite da živite „na ivici“, upotrebite Insiders izdanje, koji je dnevna izrada sledeće verzije.

Čak i ako planirate da koristite samo Visual Studio 2022 for Windows za razvoj, preporučujem da preuzmete i instalirate uređivač koda Visual Studio Code i pomoću njega da izvršite zadatke kodiranja iz ovog poglavlja, a zatim odlučite da li želite da koristite samo Visual Studio 2022 u ostalim poglavljima knjige.

Sada ćemo preuzeti i instalirati Visual Studio Code, .NET SDK i C# i .NET Interactive Notebooks ekstenzije:

1. Preuzmite i instalirajte Stable verziju ili Insiders izdanje uređivača koda Visual Studio Code sa sledećeg linka: https://code.visualstudio.com/.



Više informacija: Ako vam je potrebna pomoć za instaliranje uređivača koda Visual Studio Code, možete da pročitate zvaničan vodič na sledećem linku: https://code.visualstudio.com/docs/setup/setup-overview.

2. Preuzmite i instalirajte .NET Core SDK za verzije 3.1, 5.0 i 6.0 sa sledećeg linka: https://www.microsoft.com/net/download.



Da biste naučili kako da kontrolišete .NET SDK, potrebno je da imate više instaliranih verzija. Tri trenutno podržane verzije su .NET Core 3.1, .NET 5.0 i .NET 6.0. Možete slobodno da instalirate više verzija, jednu pored druge. Uz ovu knjigu naučićete da ciljate verziju koju želite.

3. Da biste instalirali C# ekstenziju, potrebno je da prvo pokrenete Visual Studio Code aplikaciju.
4. U uređivaču koda Visual Studio Code kliknite na **Extensions** ikonicu ili na **View | Extensions**.
5. C# je jedna od najpopularnijih dostupnih ekstenzija, pa bi trebalo da je vidite na vrhu liste, ili možete da unesete `C#` u polje za pretragu.
6. Kliknite na **Install** i sačekajte da paketi podrške budu preuzeti i instalirani.
7. Unesite `.NET Interactive` u polje za pretragu da biste pronašli ekstenziju **.NET Interactive Notebooks**.
8. Kliknite na **Install** i sačekajte da ekstenzija bude instalirana.

Instaliranje drugih ekstenzija

U narednim poglavljima ove knjige upotrebljavaćemo više ekstenzija. Ako želite da ih instalirate sada, sve ekstenzije koje ćemo upotrebljavati prikazane su u sledećoj tabeli:

NAZIV I IDENTIFIKATOR EKSTENZIJE	OPIS
C# for Visual Studio Code (powered by OmniSharp) `ms-dotnettools.csharp`	Podrška za C# editovanje, uključujući isticanje sintakse, IntelliSense, Go to Definition, Find All References, podršku za otklanjanje grešaka za .NET i podršku za `csproj` projekte na Windows, macOS i Linux sistemima.
.NET Interactive Notebooks `ms-dotnettools.dotnet-interactive-vscode`	Ova ekstenzija dodaje podršku za upotrebu .NET Interactive alata u Visual Studio Code beležnici. Ima zavisnost od Jupyter ekstenzije (`ms-toolsai.jupyter`).
MSBuild project tools `tinytoy.msbuild-project-tools`	Obezbeđuje IntelliSense za MSBuild projektne fajlove, uključujući i automatsko završavanje za `<PackageReference>` elemente.

NAZIV I IDENTIFIKATOR EKSTENZIJE	OPIS
REST Client `humao.rest-client`	Šalje HTTP zahtev i prikazuje odgovor direktno u uređivaču koda Visual Studio Code.
ILSpy .NET Decompiler `icsharpcode.ilspy-vscode`	Dekompajlira MSIL sklopove - podrška za moderan .NET, .NET Framework, .NET Core i .NET Standard.
Azure Functions for Visual Studio Code ms-azuretools.vscode-azurefunctions	Kreira, otklanja greške, upravlja i raspoređuje bezserverske aplikacije direktno iz uređivača koda VS Code. Ima zavisnost od Azure Account (`ms-vscode.azure-account`) i Azure Resources (`ms-azuretools.vscode-azureresourcegroups`) ekstenzija.
GitHub Repositories `github.remotehub`	Pretražuje, uređuje i potvrđuje sva udaljena GitHub skladišta direktno iz uređivača koda Visual Studio Code.
SQL Server (mssql) for Visual Studio Code `ms-mssql.mssql`	Za razvoj Microsoft SQL Server, Azure SQL Database i SQL Data Warehouse baza podataka bilo gde, sa bogatim skupom funkcionalnosti.
Protobuf 3 support for Visual Studio Code `zxh404.vscode-proto3`	Isticanje sintakse, validacija sintakse, isečci koda, završavanje koda, formatiranje koda, podudaranje zagrada i dodavanje komentara u liniju i blok.

Razumevanje verzija uređivača koda Microsoft Visual Studio Code

Microsoft objavljuje novu funkcionalnu verziju uređivača koda Visual Studio Code (skoro) svakog meseca, a ispravke grešaka još češće. Na primer:

- Verzija 1.59, izdanje funkcije u avgustu 2021
- Verzija 1.59.1, izdanje ispravke greške u avgustu 2021

Verzija koja se koristi u ovoj knjizi je 1.59, ali verzija uređivača koda Microsoft Visual Studio Code je manje važna od verzije C# ekstenzije za Visual Studio Code koju ste instalirali.

Iako C# ekstenzija nije potrebna, ona pruža IntelliSense dok kucate, navigaciju kodom i funkcije za otklanjanje grešaka i prema tome, to je nešto veoma korisno, pa preporučujemo da je instalirate i ažurirate da biste imali podršku za najnovije funkcije C# jezika.

Prečice na tastaturi za uređivač koda Microsoft Visual Studio Code

U ovoj knjizi izbegavaću da prikazujem prečice na tastaturi koje se koriste za zadatke kao što je kreiranje novog fajla, jer se često razlikuju na različitim operativnim sistemima. Situacije u kojima ću prikazati prečice na tastaturi su kada je potrebno više puta da pritiskate taster, na primer, tokom otklanjanja grešaka. Takođe je veća verovatnoća da će one biti dosledne u svim operativnim sistemima.

Ako želite da prilagodite prečice na tastaturi za Visual Studio Code, možete to da uradite kao što je prikazano na sledećem linku: https://code.visualstudio.com/docs/getstarted/keybindings.

Preporučujem da preuzmete PDF prečica na tastaturi za vaš operativni sistem sa sledeće liste:

- **Windows**: https://code.visualstudio.com/shortcuts/keyboard-shortcuts-windows.pdf

- **macOS**: https://code.visualstudio.com/shortcuts/keyboard-shortcuts-macos.pdf
- **Linux**: https://code.visualstudio.com/shortcuts/keyboard-shortcuts-linux.pdf

RAZUMEVANJE PLATFORME .NET

.NET 6, .NET Core, .NET Frameworki Xamarin su povezane i preklapajuće platforme koje programeri koriste za razvoj aplikacija i servisa. U ovom odeljku ćemo predstaviti svaki od ovih .NET koncepata.

Razumevanje razvojne platforme .NET Framework

.NET Framework je razvojna platforma koja uključuje **Common Language Runtime (CLR)** za upravljanje izvršenjem koda i **Base Class Library (BCL)**, bogatu biblioteku klasa za izgradnju aplikacija.

Microsoft je prvobitno dizajnirao .NET Framework tako da ima mogućnost da bude međuplatformski, ali se Microsoft potrudio i da ova platforma najbolje funkcioniše na Windows sistemima.

Od verzije .NET Framework 4.5.2, ona je zvanična komponenta Windows operativnog sistema. Komponente imaju istu podršku kao i njihovi roditeljski proizvodi, pa zato verzija 4.5.2, i novije, prati pravilo životnog ciklusa Windows operativnog sistema na kom je instalirana. .NET Framework je instaliran na više od milijardu računara, pa se mora menjati što je moguće manje. Ne ažurira se često, jer čak i ispravke grešaka mogu da izazovu probleme.

Za verziju .NET Framework 4.0, ili novije, sve aplikacije na računaru napisane za .NET Framework dele istu verziju komponente CLR i biblioteka sačuvanih u kešu **Global Assembly Cache (GAC)**, što može da dovede do problema ako neki od njih zahteva specifičnu verziju za kompatibilnost.



Dobra praksa: Praktično, .NET Framework je zastarela platforma samo za Windows sisteme. Nemojte kreirati nove aplikacije upotrebom ove platforme.

Razumevanje Mono, Xamarin i Unity projekata

Nezavisni programeri su razvili .NET Framework implementaciju pod nazivom **Mono** projekat. Mono je međuplatformski projekat, ali znatno zaostaje za zvaničnom implementacijom razvojnog okruženja .NET Framework.

Mono projekat je odlična osnova za **Xamarin** mobilnu platformu i za platforme za razvoj međuplatformskih igara, kao što je **Unity**.

Microsoft je kupio Xamarin 2016. godine i sada, uz Visual Studio razvojno okruženje besplatno nudi ono što je nekada bila skupa Xamarin ekstenzija. Microsoft je promenio naziv razvojne alatke Xamarin Studio, koja je mogla da se koristi samo za razvoj mobilnih aplikacija, u Visual Studio for Mac i omogućio je kreiranje drugih tipova projekata, kao što su konzolne aplikacije i veb servisi. Korišćenjem razvojnog okruženja Visual Studio 2022 for Mac, Microsoft je zamenio delove Xamarin Studio editora delovima iz razvojnog okruženja Visual Studio for Windows da bi obezbedio slično iskustvo i performanse. Visual Studio 2022 for Mac takođe je prepisan tako da bude originalna macOS UI aplikacija za poboljšanje pouzdanosti i korišćenje ugrađenih pomoćnih tehnologija macOS operativnog sistema.

Razumevanje radnog okvira .NET Core

Mi živimo danas u pravom međuplatformskom svetu u kojem je zbog razvoja modernih mobilnih i cloud platformi Windows mnogo manje važan operativni sistem. Zbog toga se Microsoft trudio da razdvoji usku vezu platforme .NET od Windows operativnog sistema. Dok je menjao .NET Framework tako da bude stvarno međuplatformski, Microsoft je iskoristio mogućnost da ga refaktoriše i ukloni glavne delove koji se više ne smatraju osnovnim.

Ovaj novi proizvod je nazvan .NET Core, a uključuje mađuplatformsku implementaciju komponente CLR, poznatiju kao CoreCLR, i modernizovanu biblioteku klasa (BCL), koja je poznata kao CoreFX.

Scott Hunter, Microsoft Partner Director Program Manager za .NET, ističe: „Četrdeset procenata naših .NET Core kupaca su novi programeri na ovoj platformi, što upravo i želimo za .NET Core. Želimo da uvedemo nove ljude".

NET Core se brzo razvija i pošto može da bude raspoređen „rame uz rame" sa aplikacijom, može da se menja često, a te promene neće uticati na druge .NET Core aplikacije na istoj mašini. Većina poboljšanja koja Microsoft izvršava u razvojnoj platformi .NET Core ne mogu jednostavno da se dodaju u .NET Framework.

Razumevanje budućih verzija platforme .NET

Na Microsoft Build programerskoj konferenciji, održanoj u maju 2020. godine, .NET tim je objavio da su njihovi planovi za objedinjavanje platforme .NET odloženi. Rekli su da će .NET 5 biti izdat 10. Novembra 2020. godine i da će objediniti sve različite .NET platforme, osim mobilne. Mobilna platforma će biti podržana u objedinjenoj platformi .NET novembra 2021. godine.

.NET Core je preimenovan u .NET, a glavni broj verzije je izostavio broj 4 da bi se izbegla konfuzija sa radnim okvirom .NET Framework 4.x. Microsoft planira godišnja izdavanja glavne verzije svakog novembra, a ne kao što Apple izdaje glavne verzije za iOS svakog septembra.

U sledećoj tabeli prikazano je kada su izdate ključne verzije moderne platforme .NET, za kada Microsoft planira sledeća izdanja i koja verzija je upotrebljena za različita izdanja ove knjige:

VERZIJA	DATUM IZDANJA	IZDANJE KNJIGE	PUBLIKOVANA
.NET Core RC1	novembar 2015. godine	prvo	u martu 2016. godine
.NET Core 1.0	jun 2016. godine		
.NET Core 1.1	novembar 2016. godine		
.NET Core 1.0.4 and .NET Core 1.1.1	mart 2017. godine	drugo	u martu 2017. godine
.NET Core 2.0	avgust 2017. godine		
.NET Core for UWP in Windows 10 Fall Creators Update	oktobar 2017. godine	treće	u novembru 2017. godine
.NET Core 2.1 (LTS)	maj 2018. godine		
.NET Core 2.2 (Current)	decembar 2018. godine		
.NET Core 3.0 (Current)	septembar 2019. godine	četvrto	u oktobru 2019. godine
.NET Core 3.1 (LTS)	decembar 2019. godine		
Blazor WebAssembly 3.2 (Current)	maj 2020. godine		
.NET 5.0 (Current)	novembar 2020. godine	peto	u novembru 2020. godine
.NET 6.0 (LTS)	novembar 2021. godine	šesto	u novembru 2021. godine
.NET 7.0 (Current)	novembar 2022. godine	sedmo	u novembru 2022. godine
.NET 8.0 (LTS)	novembar 2023. godine	osmo	u novembru 2023. godine

U .NET Core 3.1 verziju je uključen Blazor Server za izgradnju veb komponenti. Microsoft je takođe planirao da uključi Blazor WebAssembly u to izdanje, ali je bilo odloženo. Blazor WebAssembly je kasnije izdat kao opcioni dodatni modul za .NET Core 3.1. Uključio sam ga u prethodnu tabelu jer ima verziju 3.2 da bi se isključio iz LTS izdanja verzije .NET Core 3.1.

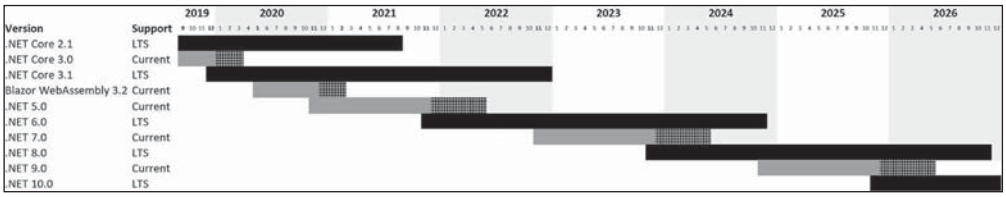
Razumevanje podrške za platformu .NET

.NET verzije su ili **Long-Term Support (LTS)** (imaju dugotrajnu podršku) ili **Current** (aktuelne), kao što je opisano u sledećoj listi:

- **LTS** izdanja su stabilna i zahtevaju manje ažuriranja u svom „životnom veku". To je dobar izbor za aplikacije koje ne nameravate da ažurirate često. LTS izdanja će biti podržana tri godine nakon što aplikacije postanu dostupne, ili jednu godinu nakon sledećeg LTS izdanja, koje god je duže.
- **Current** izdanja uključuju funkcije koje mogu da se menjaju na osnovu povratnih informacija. Ta izdanja su dobar izbor za aplikacije koje trenutno razvijate, jer obezbeđuju pristup najnovijim poboljšanjima. Nakon šestomesečnog održavanja, ili 18 meseci nakon što je aplikacija postala dostupna, prethodna manja verzija više neće uključivati podršku.

Oba izdanja se ažuriraju važnim ispravkama u svom „životnom veku“ zbog bezbednosti i pouzdanosti. Morate da imate najnovije zakrpe da biste dobili podršku. Na primer, ako sistem pokreće 1.0 verziju, a izdata je verzija 1.0.1, da biste dobili podršku potrebno je da bude instalirana verzija 1.0.1.

Da biste bolje razumeli vaš izbor Current i LTS izdanja, korisno je da ih vidite vizuelno, sa crnim trakama dugim 3 godine za LTS izdanja i sivim trakama promenljive dužine za Current izdanja, koje se završavaju unakrsnim šrafiranjem tokom 6 meseci nakon novog većeg ili manjeg izdanja za koje zadržavaju podršku, kao što je prikazano na *slici 1.2* :



Slika 1.2: Podrška za različite verzije

Na primer, ako ste kreirali projekat korišćenjem radnog okvira .NET Core 3.0, onda, kada je Microsoft objavio .NET Core 3.1 u decembru 2019. godine, morali ste da nadogradite svoj projekat na .NET Core 3.1 do marta 2020. godine. (Pre verzije .NET 5, period održavanja za Current izdanja bio je samo tri meseca.)

Ako vam je potrebna dugoročna podrška od Microsoft-a, izaberite .NET 6.0 danas i koristite ga do verzije .NET 8.0, čak i kada Microsoft objavi .NET 7.0, jer će .NET 7.0 biti Current izdanje i stoga će izgubiti podršku pre verzije .NET 6.0. Samo zapamtite da čak i sa LTS izdanjima morate da nadograđujete na izdanja za ispravku grešaka, kao što je 6.0.1.

Sve verzije platformi .NET Core i moderne .NET platforme su dostigle kraj života osim onih koje su prikazane na sledećoj listi:

- .NET 5.0 će dostići kraj životnog veka u maju 2022. godine.
- .NET Core 3.1 će dostići kraj životnog veka 3. decembra 2022. godine.
- .NET 6.0 će dostići kraj životnog veka u novembru 2024. godine.

Razumevanje verzija alata .NET Runtime i .NET SDK

Versionisanje alata .NET Runtime prati semantičko verzionisanje, odnosno, povećanje glavne verzije ukazuje na velike promene, povećanje manje verzije ukazuju na nove funkcije, a povećanja zakrpa ukazuju na ispravke grešaka.

.NET SDK verzionisanje ne prati semantičko verzionisanje. Brojevi glavne i male verzije su vezani za verziju izvršenja sa kojom se podudara. Broj zakrpe sledi konvenciju koja označava glavne i manje verzije skupa alata SDK.

Primer toga možete da vidite u sledećoj tabeli:

PROMENA	RUNTIME	SDK
Prvobitno izdanje	6.0.0	6.0.100
Ispravka greške za SDK	6.0.0	6.0.101
Ispravka grešaka za Runtime i SDK	6.0.1	6.0.102
Nova funkcija alata SDK	6.0.1	6.0.200

Uklanjanje starih verzija .NET platforme

.NET Runtime ažuriranja su kompatibilna sa glavnom verzijom, kao što je 6.x, a ažurirana izdanja alata .NET SDK održavaju mogućnost izgradnje aplikacija koje ciljaju prethodne verzije izvršenja, što omogućava bezbedno uklanjanje starijih verzija.

Možete da vidite koji alati SDK i Runtime su trenutno instalirani, pomoću sledećih komandi:

- `dotnet --list-sdks`
- `dotnet --list-runtimes`

U operativnom sistemu Windows koristite odeljak **App & features** da biste uklonili .NET SDK. Na macOS ili Windows sistemu koristite alatku `dotnet-core-uninstall`. Ova alatka nije podrazumevano instalirana.

Na primer, dok sam pisao četvrto izdanje, koristio sam sledeću komandu svakog meseca:

```
dotnet-core-uninstall remove --all-previews-but-latest --sdk
```

Šta je drugačije u modernoj .NET platformi?

Moderan .NET je modularizovan u poređenju sa starim radnim okvirom .NET Framework, koji je monolitan. On je otvorenog koda i Microsoft javno donosi odluke o poboljšanjima i promenama. Microsoft je uložio posebne napore da poboljša performanse moderne .NET platforme.

.NET platforma je mnogo manja od poslednje verzije radnog okvira .NET Framework, zbog činjenice da su uklonjene zastarele tehnologije i tehnologije koje nisu međuplatformske. Na primer, radna opterećenja, kao što su Windows Forms i **Windows Presentation Foundation (WPF)**, mogu da se upotrebe za izgradnju aplikacija **grafičkog korisničkog interfejsa (GUI)**, ali su u uskoj vezi sa Windows ekosistemom, pa zato nisu uključena u .NET Core na macOS i Linux sistemima.

Windows razvoj

Jedna od funkcija moderne .NET platforme je podrška za pokretanje starih Windows Forms i WPF aplikacija upotrebom paketa alata Windows Desktop Pack, koji je uključen u Windows verziju radnog okvira .NET Core 3.1, ili noviju, zbog čega je i veći od alata SDK za macOS i Linux sisteme. Možete da izvršite neke male promene u starim Windows aplikacijama ako je potrebno, a zatim da ih ponovo izgradite za .NET 6 da biste iskoristili nove funkcije i poboljšanja performansi.

Veb razvoj

ASP.NET Web Forms i Windows Communication Foundation (WCF) su stare tehnologije za veb aplikacije i servise, koje sve manje programera koristi u novim razvojnim projektima, pa su takođe uklonjene iz moderne .NET platforme. Umesto toga, programeri radije koriste ASP.NET MVC, ASP.NET Web API, SignalR i gRPC. Ove tehnologije su refaktorisane i kombinovane u platformu koja se pokreće na modernoj .NET platformi, pod nazivom ASP.NET Core. Učićete o ovim tehnologijama u *poglavlju 14, „Izgradnja veb sajtova pomoću radnog okvira ASP.NET Core Razor Pages“*, u *poglavlju 15 „Izgradnja veb sajtova upotrebom model-view-controller obrasca“*, u *poglavlju 16, „Izgradnja veb servisa i njihova upotreba“* i u *poglavlju 18, „Izgradnja i upotreba specijalizovanih servisa“* (poglavlje 18 je dostupno na adresi: https://github.com/markjprice/cs10dotnet6/blob/main/9781801077361_Bonus_Content.pdf).



Više informacija: Neki .NET Framework programeri su uznemireni što ASP.NET Web Forms, WCF i Windows Workflow (WF) nedostaju iz moderne .NET platforme i voleli bi kada bi se Microsoft predomislio. Postoje projekti otvorenog koda koji omogućavaju uključivanje WCF i WF tehnologije u modernu .NET platformu. Možete da pročitate više o ovome na sledećem linku: https://devblogs.microsoft.com/dotnet/supporting-the-community-with-wf-and-wcf-oss-projects/. Postoji projekat otvorenog koda za Blazor Web Forms komponente na sledećem linku: https://github.com/FritzAndFriends/BlazorWebFormsComponents.

Razvoj baze podataka

Entity Framework (EF) 6 je tehnologija objektno-relacionog mapiranja koja je projektovana za upotrebu podataka sačuvanih u relacionim bazama podataka, kao što su Oracle i Microsoft SQL Server. Tokom godina je ova tehnologija „dobila na težini“, pa je, stoga, međuplatformski API skraćen i dobija podršku za nerelacione baze podataka, kao što je Microsoft Azure Cosmos DB, i nosi naziv Entity Framework Core. O ovoj verziji ćete naučiti više u *poglavlju 10, „Upotreba baza podataka pomoću radnog okvira Entity Framework Core“*.

Ako imate postojeće aplikacije koje koriste stari EF, znate da je verzija 6.3 podržana u radnom okviru .NET Core 3.0 i novijim.

Teme moderne .NET platforme

Microsoft je kreirao veb sajt korišćenjem alata Blazor, koji prikazuje glavne teme moderne .NET platforme: https://themesof.net/.

Razumevanje specifikacije .NET Standard

Godine 2019., u .NET platformi su postojale tri grane koje je kontrolisao Microsoft, kao što je prikazano na sledećoj listi:

- **.NET Core:** za međuplatformske i nove aplikacije
- **.NET Framework:** za zastarele aplikacije
- **Xamarin:** za aplikacije za mobilne uređaje

Sve tri platforme imaju svoje vrline i mane, jer su dizajnirane za različite scenarije. To je dovelo do problema da je programer morao da nauči tri platforme, od kojih svaka ima neke dosadne „začkoljice“ i ograničenja.

Zbog toga, Microsoft je definisao .NET Standard - specifikaciju za skup interfejsa za programiranje aplikacija (API) koje svaka .NET platforma može da implementira da bi ukazala koji nivo kompatibilnosti ima. Na primer, osnovna podrška ukazuje da je platforma usklađena sa specifikacijom .NET Standard 1.4.

Izdavanjem specifikacije .NET Standard 2.0 i novijih, Microsoft je učinio da sve tri platforme konvergiraju ka modernom minimalnom standardu, što je olakšalo programerima deljenje koda između bilo kojih verzija .NET platforme.

Za .NET Core 2.0 i novije verzije to je dodalo većinu nedostajućih API-ja koji su programerima potrebni za prenos starog koda, napisanog za .NET Framework, u međuplatformski .NET Core. Međutim, neki API-ji su implementirani ali generišu izuzetak, što ukazuje programeru da ne bi trebalo da budu upotrebljeni! To se dešava obično zbog razlika u operativnom sistemu na kom pokrećete .NET Core. U *poglavlju 2, „Govorite C# jezikom“*, naučićete kako da rukujete ovim izuzecima.

Važno je da razumete da je .NET Standard samo standard. Ne možete da ga instalirate isto kao što ne možete da instalirate HTML5. Da biste upotrebili HTML5, potrebno je da instalirate veb pretraživač koji implemenitira HTML5 standard.

Da biste upotrebili .NET Standard, potrebno je da instalirate .NET platformu koja implementira .NET Standard specifikaciju. Najnoviji .NET Standard, verzija 2.1 je implementiran u projektima .NET Core 3.0, Mono i Xamarin. Neke funkcije jezika C# 8.0 zahtevaju .NET Standard 2.1, koji nije implementiran u .NET Framework 4.8, pa bi trebalo da tretiramo .NET Framework kao zastareo.

Izdanjem platforme .NET 6, novembra 2021. godine, znatno je smanjena potreba za specifikacijom .NET Standard jer sada postoji jedna .NET platforma za sve platforme, uključujući i mobilnu. .NET 6 ima jednu BCL biblioteku i dve CLR komponente: CoreCLR je optimizovana za server ili desktop scenarije, kao što su veb sajtovi i Windows desktop aplikacije, a Mono izvršenje je optimizovano za mobilne aplikacije i veb aplikacije koje imaju ograničene resurse.

Čak i sada, potrebno je da aplikacije i veb sajtovi kreirani za .NET Framework imaju podršku, pa je zato važno da razumete da možete da kreirate .NET Standard 2.0 biblioteke klase koje su kompatibilne sa starijim .NET platformama.

.NET platforme i alatke upotrebljene u izdanjima ove knjige

Za prvo izdanje ove knjige, koje je pisano u martu 2016. godine, fokusirao sam se na .NET Core funkcionalnosti, ali sam upotrebio .NET Framework kada u .NET Core još nisu bile implementirane važne i korisne funkcije, jer je to bilo pre finalnog izdanja .NET Core 1.0 verzije. Visual Studio 2015 je upotrebljen za većinu primera, a Visual Studio Code je samo ukratko predstavljen.

Drugo izdanje je bilo skoro potpuno očišćeno od svih primera .NET Framework koda, pa su čitaoci mogli da se fokusiraju na .NET Core primere koji se zaista pokreću međuplatformski.

Treće izdanje, u kojem je završeno prebacivanje, ponovo je napisano tako da je ceo kod čist .NET Core. Međutim, instrukcije korak po korak za Visual Studio Code i za Visual Studio 2017 za sve zadatke dodale su nepotrebnu složenost ovom izdanju.

U četvrtom izdanju nastavljamo trend prikazivanjem primera kodiranja samo upotrebom uređivača koda Visual Studio Code za skoro sva poglavlja knjige (osim za poslednja dva). U *poglavlju 20, „Izgradnja Windows desktop aplikacija“*, korišćen je Visual Studio pokrenut na Windows 10 sistemu, a u *poglavlju 21, „Izgradnja međuplatformskih aplikacija za mobilne uređaje“*, korišćen je Visual Studio for Mac.

U petom izdanju, *poglavlje 20, „Izgradnja Windows desktop aplikacija“* je prebačeno u Dodatak B, da bismo oslobodili prostor za novo *poglavlje 20, „Izgradnja veb korisničkih interfejsa pomoću alata Blazor“*. Blazor projekti mogu da budu kreirani upotrebom uređivača koda Visual Studio Code.

U šestom izdanju, *poglavlje 19, „Izgradnja mobilnih i desktop aplikacija pomoću radnog okvira .NET MAUI“*, bilo je ažurirano da bismo prikazali da mobilne i desktop međuplatformske aplikacije mogu da budu kreirane upotrebom razvojnog okruženja Visual Studio 2022 i **.NET MAUI (Multi-platform App UI)**. Možete da ga pronađete na adresi: https://github.com/markjprice/cs10dotnet6/blob/main/9781801077361_Bonus_Content.pdf.

Do sedmog izdanja i izdanja radnog okvira .NET 7, Visual Studio Code će imati ekstenziju za podršku razvojnog okruženja .NET MAUI. Tada će čitaoci moći da upotrebe Visual Studio Code za sve primere iz knjige.

Razumevanje posredničkog jezika

C# kompajler (pod nazivom **Roslyn**), koji koristi CLI alatka `dotnet`, konvertuje C# izvorni kod u kod posredničkog jezika (**intermediate language - IL**) i skladišti IL u programskom sklopu (**assembly**) (DLL ili EXE fajlu). Iskazi IL koda su kao instrukcije asemblerskog jezika, koje izvršava virtuelna mašina .NET platforme, poznata kao CoreCLR.

U vreme izvršenja CoreCLR učitava IL kod iz programskog sklopa, **just-in-time (JIT)** kompajler ga kompajlira u izvorne CPU instrukcije, a zatim ga CPU izvršava na vašoj mašini.

Prednost tog procesa kompajliranja, koji se sastoji od dva koraka, proističe od činjenice da Microsoft može da kreira CLR komponente za Linux i macOS, kao i za Windows sisteme. Isti IL kod se pokreće svuda zbog drugog procesa kompajliranja, koji generiše kod za izvorni operativni sistem i skup CPU instrukcija.

Bez obzira na to na kom jeziku je napisan izvorni kod - na primer, na jezicima C#, Visual Basic ili F#, sve .NET aplikacije koriste IL kod za svoje instrukcije sačuvane u programskom sklopu. Microsoft i drugi proizvođači obezbeđuju disasembler alatke koje mogu da otvore programski sklop i otkriju IL kod, kao što je ekstenzija ILSpy .NET Decompiler.

Upoređivanje .NET tehnologija

Možemo da rezimiramo i uporedimo .NET tehnologije, što je prikazano u sledećoj tabeli:

TEHNOLOGIJA	OPIS	HOST OPERATIVNI SISTEMI
Moderan .NET	moderan skup funkcija, potpuna podrška za jezik C# 8, 9 i 10, služi za prenos postojećih aplikacija i za kreiranje novih desktop, mobilnih i veb aplikacija i servisa	Windows, macOS, Linux, Android, iOS
.NET Framework	stari skup funkcija, ograničena podrška za jezik C# 8, nema podršku za verzije jezika C# 9 ili 10, služi samo za održavanje postojećih aplikacija	Samo Windows

TEHNOLOGIJA	OPIS	HOST OPERATIVNI SISTEMI
Xamarin	samo za mobilne i desktop aplikacije	Android, iOS, macOS

IZGRADNJA KONZOLNIH APLIKACIJA POMOĆU RAZVOJNOG OKRUŽENJA VISUAL STUDIO 2022

Naš cilj u ovom odeljku je da pokažemo kako da izgradite konzolnu aplikaciju upotrebom razvojnog okruženja Visual Studio 2022 for Windows.

Ako nemate Windows računar ili želite da upotrebite Visual Studio Code, onda možete da preskočite ovaj odeljak, jer će kod biti isti, samo koristite drugi alat.

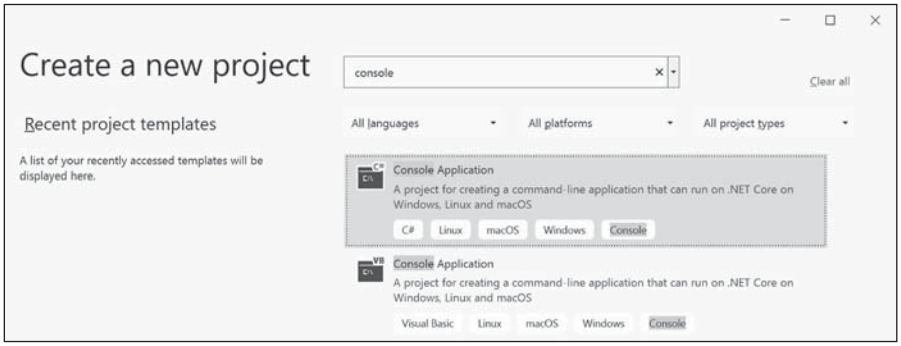
Vođenje više projekata upotrebom razvojnog okruženja Visual Studio 2022

Visual Studio 2022 ima koncept pod nazivom **rešenje** (solution) koje vam omogućava da otvorite i vodite više projekata istovremeno. Mi ćemo koristiti rešenje za vođenje dva projekta koja ćemo kreirati u ovom poglavlju.

Pisanje koda korišćenjem razvojnog okruženja Visual Studio 2022

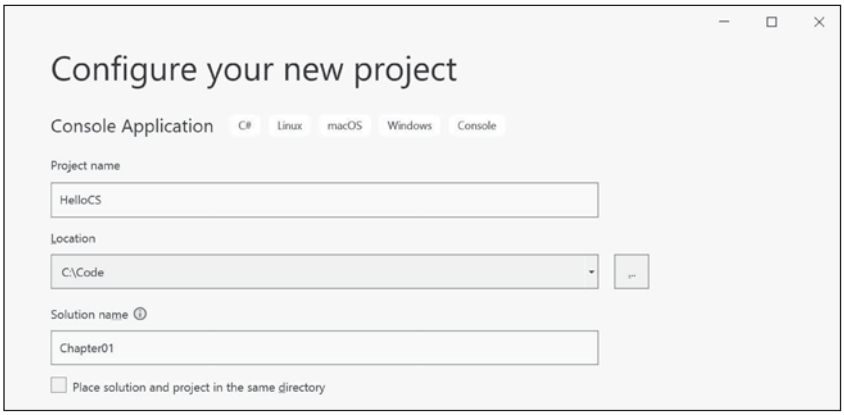
Započnite sada pisanje koda!

1. Pokrenite Visual Studio 2022.
2. U prozoru Start kliknite **Create a new project**.
3. U **Create a new project** okviru za dijalog unesite `console` u polje **Search for templates**, i selektujte **Console Application**, a ujedno potvrdite da ste za šablon projekta izabrali C# a ne neki drugi jezik, kao što su F# ili Visual Basic, kao što je prikazano na *slici 1.3*:



Slika 1.3: Selekcija projektnog šablona Console Application

4. Kliknite **Next**.
5. U **Configure your new project** okviru za dijalog, unesite `HelloCS` za naziv projekta, unesite `C:\Code` za lokaciju i unesite `Chapter01` za naziv rešenja, kao što je prikazano na *slici 1.4*:



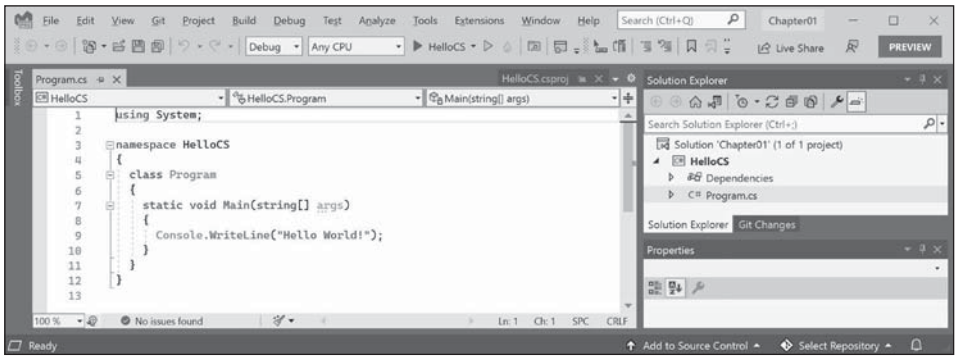
Slika 1.4: Konfigurisanje naziva i lokacija za nov projekat

6. Kliknite **Next**.



Namerno ćemo koristiti stariji projektni šablon za .NET 5.0 da bismo videli kako izgleda cela konzolna aplikacija. U sledećem odeljku, kreiraćemo konzolnu aplikaciju korišćenjem .NET 6.0 platforme i videćemo šta se promenilo.

7. U **Additional information** okviru za dijalog, u padajućoj listi **Target Framework**, zabeležite izbore za Current i LTS verzije .NET platforme, a zatim selektujte **.NET 5.0 (Current)** i kliknite na **Create**.
8. U **Solution Explorer** prozoru dvostruko kliknite na fajl pod nazivom `Program.cs` da biste ga otvorili i vidite da **Solution Explorer** prikazuje **HelloCS** projekat, kao što je prikazano na *slici 1.5*:



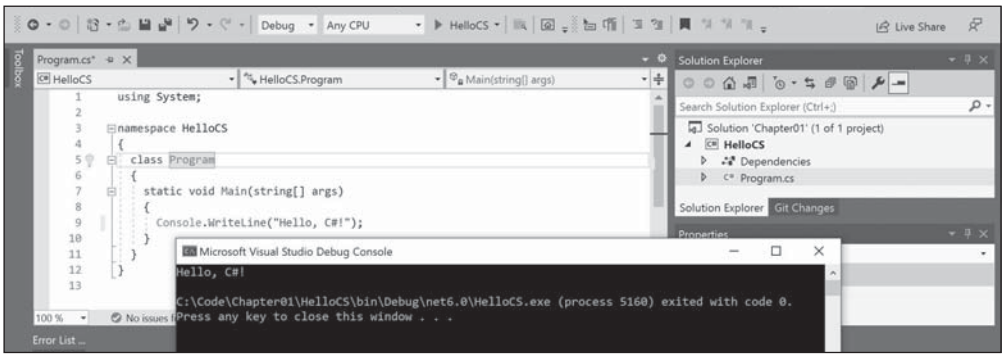
Slika 1.5: Uređivanje Program.cs fajla u razvojnom okruženju Visual Studio 2022

9. U fajlu `Program.cs` modifikujte liniju 9 tako da se u konzoli ispisuje tekst `Hello, C# !`

Kompajliranje i pokretanje koda pomoću razvojnog okruženja Visual Studio

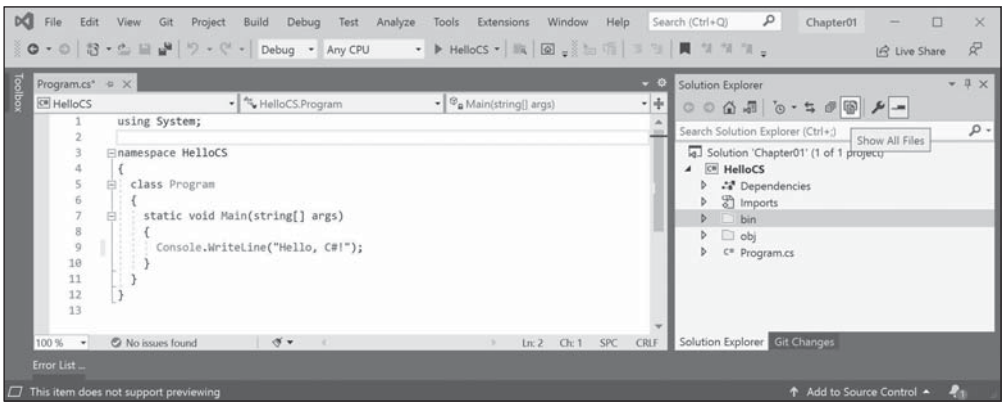
Sledeći zadatak je da kompajlirate i pokrenete kod.

1. U razvojnom okruženju Visual Studio kliknite **Debug | Start Without Debugging**.
2. Ispis u prozoru konzole će prikazati rezultat pokretanja aplikacije, kao što je prikazano na *slici 1.6*:



Slika 1.6: Pokretanje konzolne aplikacije na Windows sistemu

3. Pritisnite bilo koji taster da biste zatvorili prozor konzole i vratili se u razvojno okruženje Visual Studio.
4. Selektujte **HelloCS** projekat, a zatim u liniji alata **Solution Explorer** prozora uključite dugme **Show All Files** i videćete da su vidljivi direktorijumi `bin` i `obj` koje je generisao kompajler, kao što je prikazano na *slici 1.7*:



Slika 1.7: Prikaz direktorijuma i fajlova koje je generisao kompajler.

Razumevanje direktorijuma i fajlova koje generiše kompajler

Kompajler je generisao dva direktorijuma, `obj` i `bin`. Nije još potrebno da gledate u ove direktorijume ili da razumete njihove fajlove. Samo imajte na umu da kompajler kreira privremene direktorijume i fajlove da bi obavio svoj posao. Možete da izbrišete ove direktorijume i njihove fajlove, a kasnije oni ponovo mogu da budu kreirani. Programeri to često rade da bi „očistili" projekat. Visual Studio čak ima komandu u meniju **Build** pod nazivom **Clean Solution** koja briše neke od ovih privremenih fajlova umesto vas. Ekvivalentna komanda za Visual Studio Code je `dotnet clean`.

- Direktorijum `obj` sadrži jedan kompajliran *object* fajl za svaki fajl izvornog koda. Ti objekti još uvek nisu povezani u konačni izvršni fajl.
- Direktorijum `bin` sadrži *binarni* izvršni fajl za aplikaciju ili biblioteku klasa. To ćemo detaljnije opisati u *poglavlju 7* , *„Pakovanje i distribucija .NET tipova"*.

Pisanje programa najvišeg nivoa

Možda mislite da je bilo potrebno puno koda samo da biste ispisali `Hello, C#!`.

Iako je projektni šablon napisao šablonski kod za vas, da li postoji jednostavniji način?

U jeziku C# 9 ili novijem, postoji jednostavniji način, a poznat je kao **program najvišeg nivoa (top-level program)**.

Uporedimo konzolnu aplikaciju koju je kreirao šablon projekta, kao što je prikazano u sledećem kodu:

```
using System;

namespace HelloCS
{
  class
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Hello World!");
    }
  }
}
```

sa novom minimalnom konzolnom aplikacijom programa najvišeg nivoa, kao što je prikazano u sledećem kodu:

```
using System;

Console.WriteLine("Hello World!");
```

To je mnogo jednostavnije, zar ne? Ako biste morali da počnete rad praznim fajlom i sami da napišete sve iskaze, ovo je bolje rešenje. Ali kako to funkcioniše?

Tokom kompajliranja, sav šablonski kod za definisanje imenskog prostora, klase `Program` i njenog `Main` metoda, generiše se i omotava oko iskaza koje pišete.

Ključne tačke koje je potrebno da zapamtite o programima najvišeg nivoa uključuju sledeću listu:

- Svi iskazi `using` i dalje moraju da budu na vrhu fajla.
- U projektu može da postoji samo jedan ovakav fajl.

Iskaz `using System;` na vrhu fajla importuje imenski prostor `System`. To omogućava da funkcioniše iskaz `Console.WriteLine`. U sledećem poglavlju ćete naučiti više o imenskim prostorima.

Dodavanje drugog projekta pomoću razvojnog okruženja Visual Studio 2022

Dodaćemo drugi projekat našem rešenju za istraživanje programa najvišeg nivoa:

1. U razvojnom okruženju Visual Studio kliknite na **File** | **Add** | **New Project**.
2. U okviru za dijalog **Add a new project**, u listi **Recent project templates**, izaberite **Console Application [C#]**, a zatim kliknite **Next**.
3. U okviru za dijalog **Configure your new project,** za naziv **Project name** unesite `TopLevelProgram`, za lokaciju ostavite `C:\Code\Chapter01`, a zatim kliknite **Next**.
4. U okviru za dijalog **Additional information** izaberite **.NET 6.0 (Long-term support)**, a zatim kliknite **Create**.
5. U prozoru **Solution Explorer**, u projektu `TopLevelProgram`, dvostruko kliknite na fajl `Program.cs` da biste ga otvorili.
6. U fajlu `Program.cs` vidite da se kod sastoji samo od komentara i jednog iskaza jer koristi funkciju programa najvišeg nivoa, koja je predstavljena u jeziku C# 9, kao što je prikazano u sledećem kodu:

```
// See https://aka.ms/new-console-template for more information
   Console.WriteLine("Hello, World!");
```

Ali kada sam ranije uveo koncept programa najvišeg nivoa, bio nam je potreban iskaz `using System;`. Zašto nam ovde nije potreban?

Implicitno importovani imenski prostori

Trik je u tome što još uvek moramo da importujemo `System` imenski prostor, ali to je sada urađeno za nas pomoću funkcije predstavljene u jeziku C# 10. Pogledajmo kako:

1. U **Solution Explorer** prozoru izaberite projekat `TopLevelProgram` i uključite dugme **Show All Files** i vidite da su vidljivi direktorijumi `bin` i `obj` koje generiše kompajler.
2. Proširite direktorijum `obj`, proširite direktorijum `Debug`, proširite direktorijum `net 6.0` i otvorite fajl pod nazivom `TopLevelProgram.GlobalUsings.g.cs`.
3. Imajte na umu da ovaj fajl automatski kreira kompajler za projekte koji ciljaju .NET 6 platformu i da koristi funkciju predstavljenu u jeziku C# 10 pod nazivom **global imports** koja importuje neke najčešće korišćene imenske prostore, kao što je `System`, za upotrebu u svim fajlovima koda, kao što je prikazano u sledećem kodu:

```
// <autogenerated />
global using global::System;
global using global::System.Collections.Generic;
global using global::System.IO;
global using global::System.Linq;
global using global::System.Net.Http;
global using global::System.Threading;
global using global::System.Threading.Tasks;
```

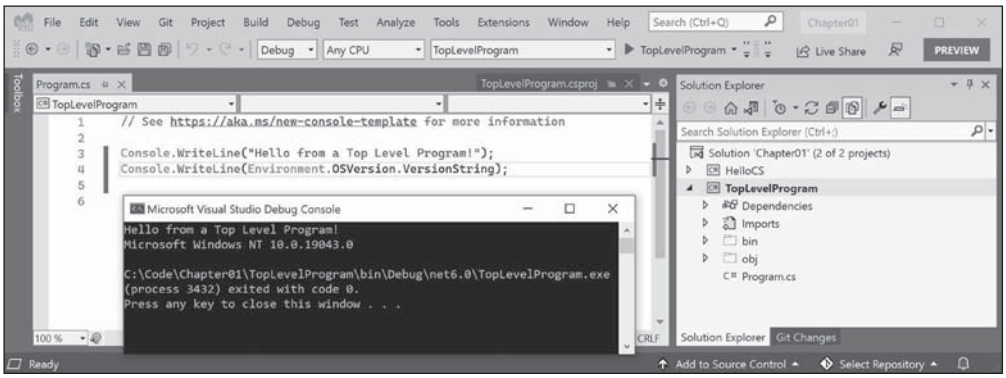


Govoriću više o ovoj funkciji u sledećem poglavlju. Za sada, samo imajte na umu da je značajna promena između verzija .NET 5 i .NET 6 ta što mnogi projektni šabloni, kao što je šablon konzolne aplikacije, koriste nove funkcije jezika za skrivanje onoga što se zaista dešava.

4. U projektu `TopLevelProgram`, u fajlu `Program.cs`, modifikujte iskaz tako da ispisuje drugu poruku i verziju operativnog sistema, kao što je prikazano u sledećem kodu:

```
Console.WriteLine("Hello from a Top Level Program!");
Console.WriteLine(Environment.OSVersion.VersionString);
```

5. U **Solution Explorer** prozoru, kliknite desnim tasterom miša na rešenje **Chapter01**, izaberite opciju **Set Startup Projects…**, podesite **Current selection**, a zatim kliknite **OK**.
6. U **Solution Explorer** prozoru kliknite na projekat **TopLevelProgram** (ili bilo koji fajl ili direktorijum u njemu) i vidite da Visual Studio ukazuje da je **TopLevelProgram** sada početni projekat, tako što je naziv projekta podebljan.
7. Kliknite na **Debug | Start Without Debugging** da biste pokrenuli projekat **TopLevelProgram** i vidite rezultat, kao što je prikazan na *slici 1.8*:



Slika 1.8: Pokretanje programa najvišeg nivoa u rešenju razvojnog okruženja Visual Studio, sa dva projekta na Windows sistemu

IZGRADNJA KONZOLNIH APLIKACIJA POMOĆU UREĐIVAČA KODA VISUAL STUDIO CODE

Naš cilj u ovom odeljku je da pokažemo kako da izgradite konzolnu aplikaciju upotrebom uređivača koda Visual Studio Code.

Ako ne želite da isprobate Visual Studio Code ili .NET Interactive Notebooks, slobodno preskočite ovaj odeljak i sledeći, a zatim nastavite čitanje od odeljka *Pregled direktorijuma i fajlova za projekte*.

I instrukcije i snimci ekrana u ovom odeljku su za Windows sistem, ali iste akcije će funkcionisati i za Visual Studio Code na macOS sistemu i na različitim Linux varijantama.

Glavne razlike će biti originalne akcije komandne linije, kao što je brisanje fajla: komanda i putanja će verovatno biti drugačije na Windows sistemu ili macOS i Linux sistemima. Srećom, `dotnet` alatka komandne linije će biti identična na svim platformama.

Vođenje više projekata upotrebom uređivača koda Visual Studio Code

Visual Studio Code ima koncept pod nazivom radni prostor (**workspace**) koji vam omogućava da otvorite i vodite više projekata istovremeno. Mi ćemo koristiti radni prostor za vođenje dva projekta, koje ćemo kreirati u ovom poglavlju.

Pisanje koda pomoću uređivača koda Visual Studio Code

Počnimo sada pisanje koda:

1. Pokrenite Visual Studio Code.
2. Uverite se da nemate otvorenih fajlova, direktorijuma ili radnih prostora.
3. Kliknite na **File | Save Workspace As…**.
4. U okviru za dijalog otvorite vaš korisnički direktorijum na macOS sistemu (moj direktorijum je `markjprice`), na Windows sistemu selektujte direktorijum `Documents` ili bilo koji direktorijum ili drajv na koji želite da snimite projekte.
5. Kliknite na dugme **New Folder** i direktorijumu dodelite naziv `Code`. (Ako ste uradili vežbu za Visual Studio 2022, onda ovaj direktorijum već postoji).
6. U direktorijumu `Code` kreirajte nov direktorijum, pod nazivom `Chapter01-vscode`.
7. U direktorijumu `Chapter01-vscode` snimite radni prostor kao `Chapter01.code-workspace`.
8. Kliknite na **File | Add Folder to Workspace…** ili kliknite na dugme **Add Folder**.
9. U direktorijumu `Chapter01-vscode` kreirajte nov direktorijum pod nazivom `HelloCS`.

10. Selektujte `HelloCS` direktorijum i kliknite na dugme **Add**.
11. Kliknite na **View | Terminal**.

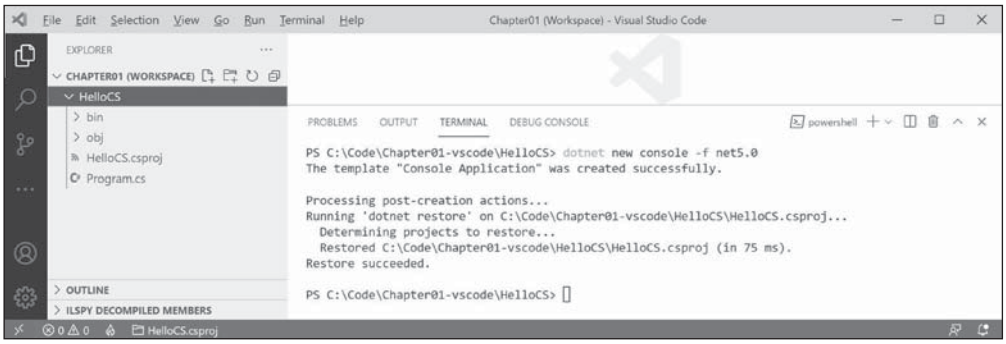


Namerno koristimo stari projektni šablon za .NET 5.0 da bismo videli kako izgleda kompletna konzolna aplikacija. U sledećem odeljku ćete kreirati konzolnu aplikaciju korišćenjem razvojnog okruženja .NET 6 i videćete šta se promenilo.

12. U **TERMINAL** prozoru potvrdite da se nalazite u direktorijumu `HelloCS`, a zatim upotrebite `dotnet` alatku komandne linije da biste kreirali novu konzolnu aplikaciju za .NET 5.0, kao što je prikazano u sledećoj komandi:

```
dotnet new console -f net5.0
```

13. Videćete da alatka komandne linije `dotnet` kreira nov projekat **Console Application** u aktuelnom direktorijumu, a prozor **EXPLORER** prikazuje dva kreirana fajla `HelloCS.csproj` i `Program.cs` i direktorijum `obj`, kao što je prikazano na *slici 1.9*:



Slika 1.9: Prozor EXPLORER će prikazati da su kreirana dva fajla i direktorijum

14. U prozoru **EXPLORER** kliknite na fajl pod nazivom `Program.cs` da biste ga otvorili u prozoru uređivača. Kada to prvi put budete radili, možda će biti potrebno da Visual Studio Code preuzme i instalira C# zavisnosti, kao što su OmniSharp, .NET Core Debugger i Razor Language Server, ako to nije uradio kada ste instalirali C# ekstenziju ili ako im je potrebno ažuriranje. Visual Studio Code će prikazati napredak u prozoru **Output**, a možda će prikazati i poruku `Finished`, kao što je prikazano u sledećem izlazu:

```
Installing C# dependencies...
Platform: win32, x86_64

Downloading package 'OmniSharp for Windows (.NET 4.6 / x64)' (36150
KB)..................... Done!
```

```
Validating download...
Integrity Check succeeded.
Installing package 'OmniSharp for Windows (.NET 4.6 / x64)'

Downloading package '.NET Core Debugger (Windows / x64)' (45048
KB)..................... Done!
Validating download...
Integrity Check succeeded.
Installing package '.NET Core Debugger (Windows / x64)'

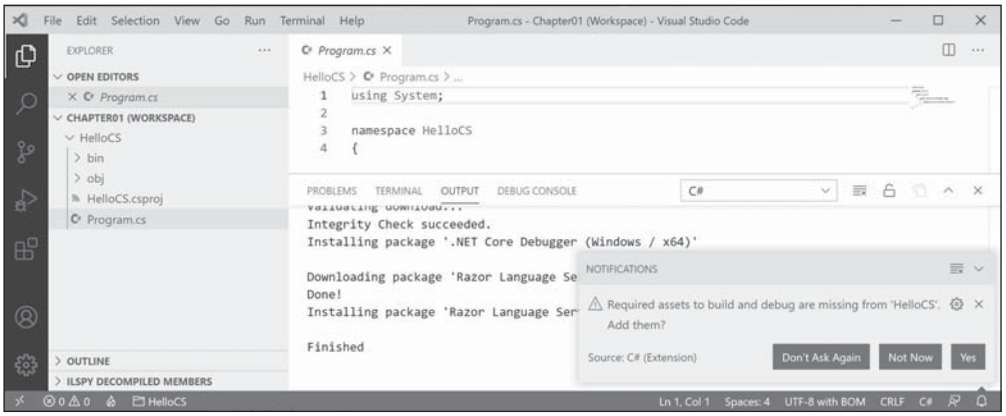
Downloading package 'Razor Language Server (Windows / x64)' (52344
KB)..................... Done!
Installing package 'Razor Language Server (Windows / x64)'

Finished
```



Prethodni izlaz je iz uređivača koda Visual Studio Code na Windows sistemu. Kada je pokrenut na macOS ili Linux sistemu, izlaz će izgledati malo drugačije, ali će biti preuzete i instalirane ekvivalentne komponente za vaš operativni sistem.

15. Direktorijumi `obj` i `bin` će biti kreirani, a kada vidite upozorenje da nedostaju potrebni elementi, kliknite na **Yes**, kao što je prikazano na *slici 1.10*:



Slika 1.10: Poruka upozorenja za dodavanje potrebne verzije i elemenata za otklanjanje grešaka

16. Ukoliko upozorenje nestane pre nego što izvršite akciju, onda možete da kliknete na ikonicu zvona u desnom uglu statusne linije da biste ga ponovo prikazali.
17. Nakon nekoliko sekundi, biće kreiran još jedan direktorijum pod nazivom `.vscode` koji sadrži iste fajlove koje je Visual Studio Code upotrebio za obezbeđivanje funkcija, kao što je IntelliSense tokom ispravljanja greške, o kom ćete učiti u *poglavlju 4, „Pisanje, ispravljanje grešaka i testiranje funkcija“*.

18. U fajlu `Program.cs` modifikujte liniju 9 tako da tekst napisan u konzoli glasi `Hello, C#!`



Dobra praksa: Kliknite na **File | Auto Save**. Ova opcija će vam pomoći, jer ne morate da pamtite da je potrebno da snimate fajl svaki put pre nego što ponovo izgradite aplikaciju.

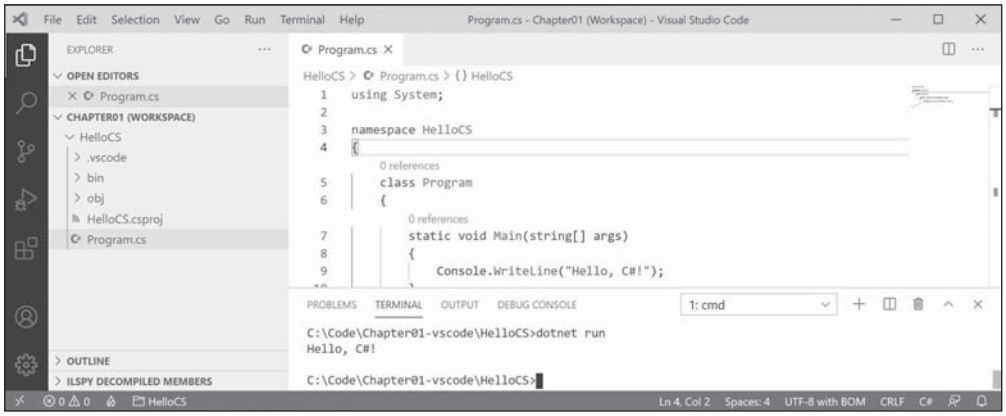
Kompajliranje i pokretanje koda pomoću dotnet CLI alatke

Sledeći zadatak je da kompajlirate i pokrenete kod:

1. Kliknite na **View | Terminal** i unesite sledeću komandu:

```
dotnet run
```

2. Rezultat u prozoru **TERMINAL** će prikazati rezultat pokretanja aplikacije, kao što je prikazano na *slici 1.11*:



Slika 1.11: Rezultat pokretanja prve konzolne aplikacije

Dodavanje drugog projekta korišćenjem uređivača koda Visual Studio Code

Dodajmo sada drugi projekat u radni prostor da bismo istražili programe najvišeg nivoa:

1. U uređivaču koda Visual Studio Code, kliknite na **File | Add Folder to Workspace…**.
2. U direktorijumu `Chapter01-vscode`, kliknite dugme **New Folder** da biste kreirali nov direktorijum pod nazivom`TopLevelProgram`, selektujte ga i kliknite na **Add**.

3. Kliknite na **Terminal** | **NewTerminal**, a u padajućoj listi koja će se pojaviti izaberite **TopLevelProgram**. Alternativno, u prozoru **EXPLORER**, kliknite desnim tasterom miša na direktorijum `TopLevelProgram`, a zatim izaberite opciju **Open in Integrated Terminal**.
4. U **TERMINAL** prozoru potvrdite da je otvoren direktorijum `TopLevelProgram`, a zatim unesite komandu za kreiranje nove konzolne aplikacije, kao što je prikazano u sledećoj komandi:

```
dotnet new console
```



Dobra praksa: Kada koristite radne prostore, budite pažljivi kada unosite komande u TERMINAL. Uverite se da je otvoren ispravan direktorijum pre nego što unesete potencijalno destruktivne komande! Zato sam želeo da kreirate nov terminal za direktorijum TopLevelProgram pre izdavanja komande za kreiranje nove konzolne aplikacije.

5. Kliknite na **View** | **Command Palette**.
6. Unesite `omni`, a zatim u padajućoj listi koja će se pojaviti izaberite **OmniSharp: Select Project**.
7. U padajućoj listi za dva projekta izaberite **TopLevelProgram** projekat, a kada se to od vas zatraži, kliknite**Yes** da dodate potrebne elemente za otklanjanje grešaka.



Dobra praksa: Da biste omogućili otklanjanje grešaka i druge korisne funkcije, kao što su formatiranje koda i Go to Definition, morate da specifikujete funkciji OmniSharp na kom projektu aktivno radite u uređivaču koda Visual Studio Code. Možete brzo da menjate aktivne projekte tako što ćete kliknuti na projekat/direktorijum desno od ikone plamena na levoj strani statusne linije.

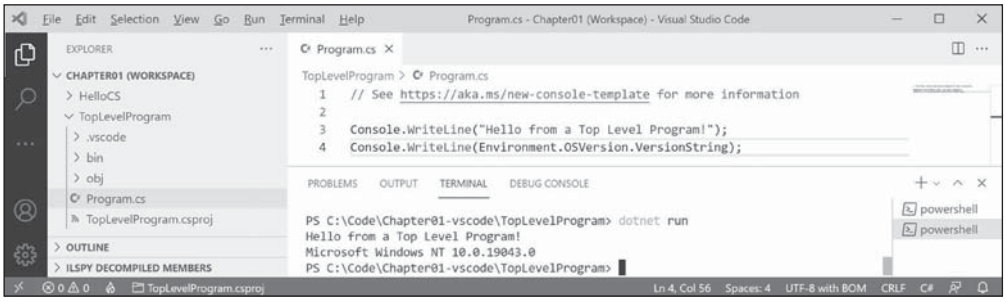
8. U prozoru **EXPLORER**, u `TopLevelProgram` direktorijumu, izaberite fajl `Program.cs`, a zatim promenite postojeći iskaz tako da ispisuje drugu poruku i da prikazuje znakovni niz verzije operativnog sistema, kao što je prikazano u sledećem kodu:

```
Console.WriteLine("Hello from a Top Level Program!");
Console.WriteLine(Environment.OSVersion.VersionString);
```

9. U **TERMINAL** unesite komandu za pokretanje programa, kao što je prikazano u sledećoj komandi:

```
dotnet run
```

10. Vidite izlaz u prozoru **TERMINAL**, a prikazan je i na *slici 1.12*:



Slika 1.12: Pokretanje programa najvišeg nivoa u radnom prostoru Visual Studio Code sa dva projekta na Windows sistemu

Ako biste pokrenuli program na macOS Big Sur sistemu, operativni sistem okruženja bi bio drugačiji, kao što je prikazano u sledećem izlazu:

```
Hello from a Top Level Program!
Unix 11.2.3
```

Korišćenje više fajlova pomoću uređivača koda Visual Studio Code

Ako imate više fajlova koje želite da koristite istovremeno, možete da ih postavite jedan pored drugog dok ih uređujete:

1. U prozoru **EXPLORER** proširite dva projekta.
2. Otvorite oba `Program.cs` fajla iz dva projekta.
3. Kliknite, držite i prevucite karticu prozora za uređivanje za jedan od otvorenih fajlova da biste ih rasporedili tako da možete da vidite oba fajla istovremeno.

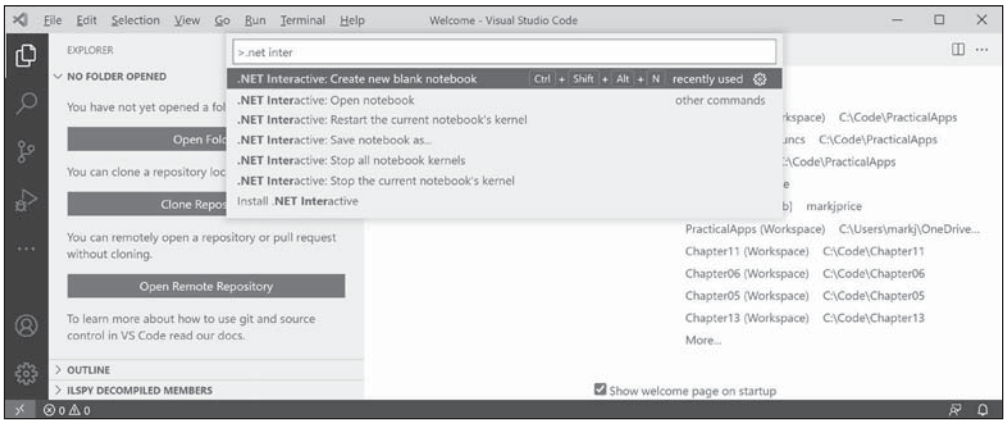
ISTRAŽIVANJE KODA POMOĆU EKSTENZIJE .NET INTERACTIVE NOTEBOOKS

.NET Interactive Notebooks još više olakšava pisanje koda od programa najvišeg nivoa. Zahteva Visual Studio Code, pa ako ga niste instalirali, instalirajte ga sada.

Kreiranje beležnice

Potrebno je da prvo kreiramo beležnicu:

1. U uređivaču koda Visual Studio Code zatvorite sve otvorene radne prostore ili direktorijume.
2. Kliknite na **View | Command Palette**.
3. Unesite `.net inter`, a zatim selektujte opciju **.NET Interactive: Create new blank notebook**, kao što je prikazano na *slici 1.13*:



Slika 1.13: Kreiranje nove prazne .NET beležnice

4. Kada se od vas zatraži da izaberete ekstenziju fajla, izaberite **Create as ‚.dib'**.



`.dib` je eksperimentalni format fajla koji je definisao Microsoft da bi se izbegla zabuna i problemi sa kompatibilnošću sa `.ipynb` formatom koji koriste Python interaktivne beležnice. Ekstenzija fajla je istorijski bila samo za Jupyter beležnice koje mogu da sadrže interaktivnu (I) mešavinu podataka, Python kod (PI) i izlaz u fajl beležnice (NB). Za .NET Interactive Notebooks, koncept se proširio kako bi omogućio mešavinu C#, F#, SQL, HTML, JavaScript, Markdown i drugih jezika. `.dib` je poliglot, što znači da podržava mešovite jezike. Podržana je konverzija između `.dib` i `.ipynb` formata fajlova.

5. Izaberite **C#** za podrazumevani jezik za ćelije koda u beležnici.
6. Ako je dostupna novija verzija ekstenzije .NET Interactive, možda ćete morati da sačekate deinstalaciju stare verzije i instalaciju nove. Kliknite na **View | Output**, a u padajućoj listi izaberite opciju **.NET Interactive : diagnostics**. Budite strpljivi. Može proći nekoliko minuta da se beležnica pojavi jer mora da pokrene host okruženje .NET. Ako se ništa ne dogodi nakon nekoliko minuta, zatvorite Visual Studio Code i ponovo ga pokrenite.

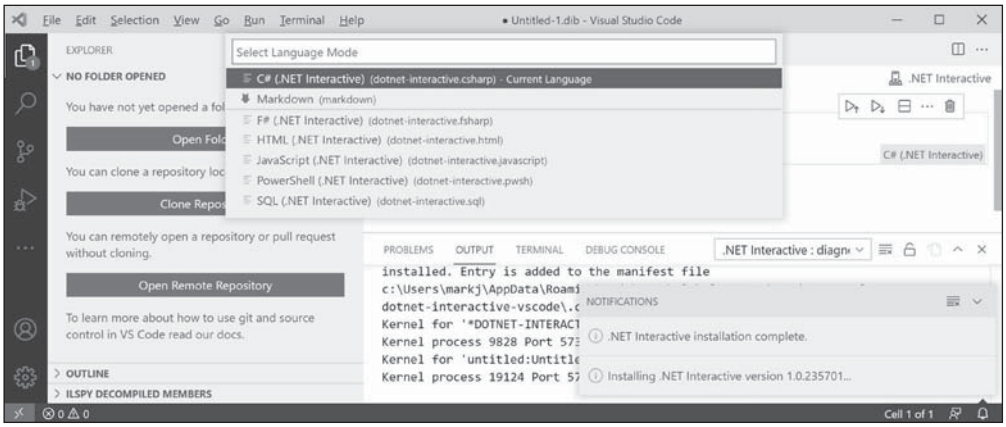
7. Kada je ekstenzija .NET Interactive Notebooks preuzeta i instalirana, dijagnostika prozora **OUTPUT** će prikazati da je proces kernela započet (proces i broj porta će se razlikovati od ovde prikazanog izlaza), kao što je prikazano u sledećem izlazu, koji je izmenjen radi uštede prostora:

```
Extension started for VS Code Stable.
...
Kernel process 12516 Port 59565 is using tunnel uri http://
localhost:59565/
```

Pisanje i pokretanje koda u beležnici

Zatim možemo da pišemo kod u ćelije beležnice:

1. Prva ćelija bi već trebalo da bude podešena na **C# (.NET Interactive)**, ali ako je podešena na bilo šta drugo, kliknite na birač jezika u donjem desnom uglu ćelije koda, a zatim izaberite **C# (.NET Interactive)** kao režim jezika za datu ćeliju i označite druge izbore jezika za ćeliju koda, kao što je prikazano na *slici 1.14*:

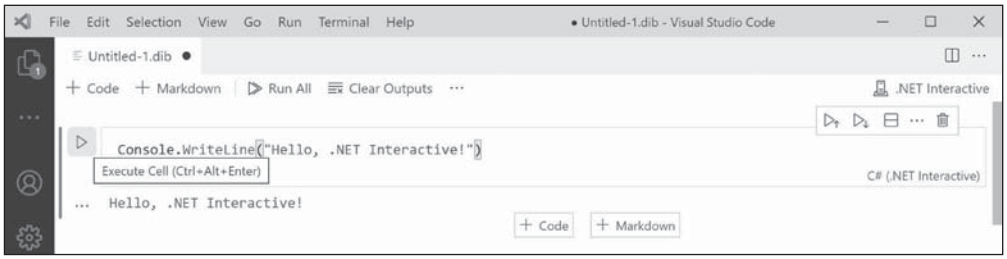


Slika 1.14: Promena jezika za ćeliju koda u .NET Interactive beležnici

2. Unutar **C# (.NET Interactive)** ćelije koda, unesite iskaz za ispis poruke u konzoli, ali imajte na umu da ne morate da završavate iskaz tačka-zarezom, kao što biste inače uradili u punoj aplikaciji, kao što je prikazano u sledećem kodu:

```
Console.WriteLine("Hello, .NET Interactive!")
```

3. Kliknite na dugme **Execute Cell**, levo od ćelije koda i vidite izlaz koji se pojavljuje u sivom polju ispod ćelije koda, kao što je prikazano na *slici 1.15*:



Slika 1.15: Pokretanje koda u beležnici i pregled rezultata ispod

Snimanje beležnice

Kao i svaki drugi fajl, trebalo bi da sačuvamo beležnicu pre nego što nastavimo rad:

1. Kliknite na **File** | **Save As...**.
2. Prebacite se na `Chapter01-vscode` direktorijum i sačuvajte beležnicu kao `Chapter01.dib`.
3. Zatvorite `Chapter01.dib` karticu uređivača.

Dodavanje jezika Markdown i specijalnih komandi u beležnicu

Možemo da mešamo i uparujemo ćelije koje sadrže jezik Markdown i kod sa specijalnim komandama:

1. Kliknite na **File** | **Open File...**, i selektujte fajl `Chapter01.dib`.
2. Ako vidite poruku `Do you trust the authors of these files?`, kliknite **Open**.
3. Zadržite kursor miša iznad bloka koda i kliknite **+ Markup** da biste dodali Markdown ćeliju.
4. Unesite naslov nivoa 1, kao što je prikazano u sledećem Markdown kodu:

```
# Chapter 1 - Hello, C#! Welcome, .NET!
Mixing *rich* **text** and code is cool!
```

5. Kliknite na kvačicu u gornjem desnom uglu ćelije da biste zaustavili uređivanje ćelije i prikazali obrađen Markdown.

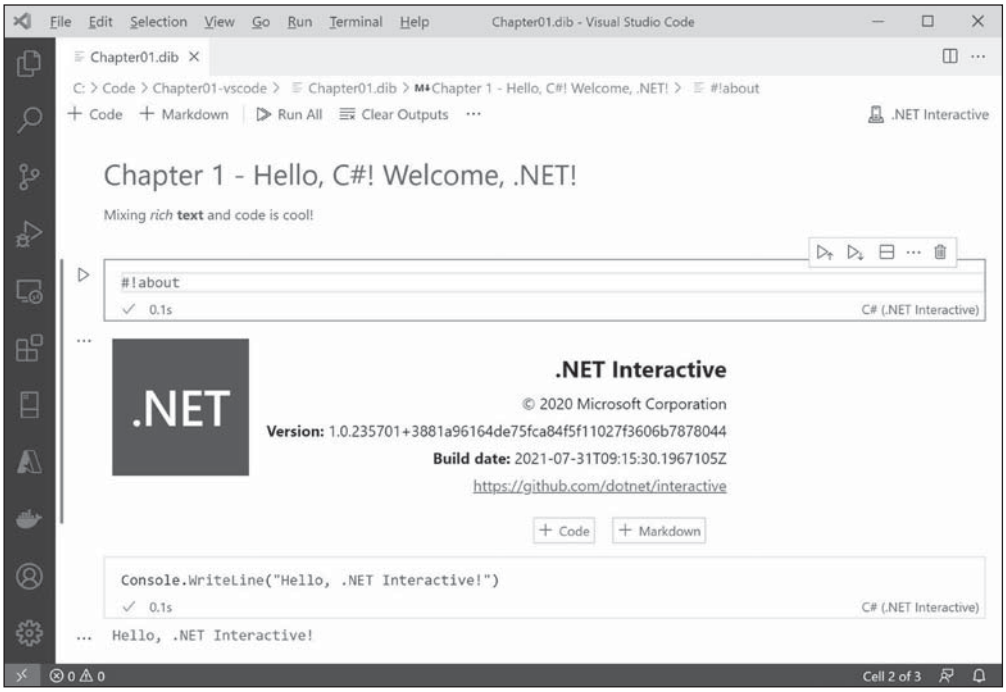


Ako su ćelije u pogrešnom redosledu, možete da prevučete i otpustite ćelije da biste ih preuredili.

6. Zadržite kursor između ćelije Markdown i ćelije koda i kliknite na **+ Code**.
7. Unesite specijalnu komandu za ispis informacija o verziji ekstenzije .NET Interactive, kao što je prikazano u sledećem kodu:

```
#!about
```

8. Kliknite dugme **Execute Cell** i videćete izlaz kao na *slici 1.16*:



Slika 1.16: Mešanje jezika Markdown, koda i specijalnih komandi u .NET Interactive beležnici

Izvršavanje koda u više ćelija

Kada imate više ćelija koda u beležnici, potrebno je da izvršite prethodne ćelije koda pre nego što njihov kontekst postane dostupan u sledećim ćelijama koda:

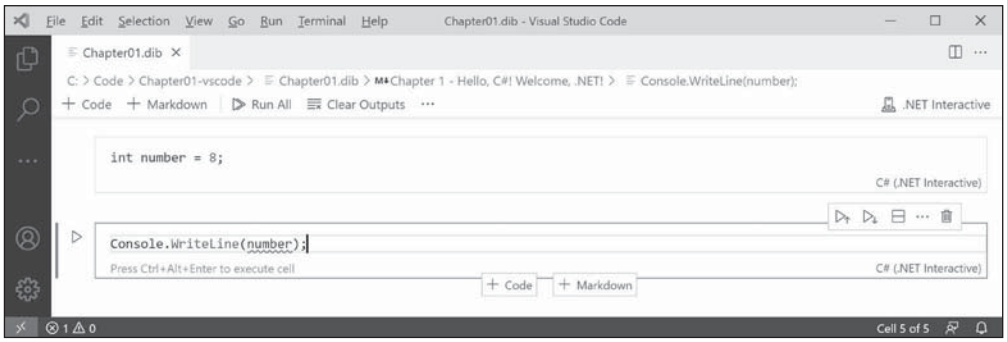
1. Na dno beležnice dodajte novu ćeliju koda, a zatim unesite iskaz za deklarisanje promenljive i dodelu celobrojne vrednost, kao što je prikazano u sledećem kodu:

```
int number = 8;
```

2. Na dno beležnice, dodajte novu ćeliju koda, a zatim unesite iskaz za ispis promenljive `number`, kao što je prikazano u sledećem kodu:

```
Console.WriteLine(number);
```

3. Imajte na umu da druga ćelija koda ne zna za `number` promenljivu jer je definisana i dodeljena u drugoj ćeliji koda, odnosno kontekstu, kao što je prikazano na *slici 1.17*:



Slika 1.17: Promenljiva number ne postoji u aktuelnoj ćeliji ili kontekstu

4. U prvoj ćeliji kliknite dugme **Execute Cell** da biste deklarisali i dodelili vrednost promenljivoj, a zatim u drugoj ćeliji kliknite dugme **Execute Cell** da biste ispisali promenljivu `number` i da biste videli da to funkcioniše. (Alternativno, u prvoj ćeliji možete da kliknete na dugme **Execute Cell and Below**.)



Dobra praksa: Ako imate povezan kod podeljen između dve ćelije, ne zaboravite da izvršite prethodnu ćeliju pre izvršavanja sledeće ćelije. Na vrhu beležnice nalaze se sledeća dugmad –**Clear Outputs i Run All**. Veoma su korisna jer možete da kliknete na jedno pa drugo da biste bili sigurni da se sve ćelije koda izvršavaju ispravno, sve dok su u ispravnom redosledu.

Korišćenje .NET interaktivnih beležnica za kod u ovoj knjizi

U ostalim poglavljima neću davati eksplicitna uputstva za korišćenje beležnica, ali GitHub skladište za knjigu sadrži beležnice sa rešenjima. Očekujem da će mnogi čitaoci želeti da pokreću moje unapred kreirane beležnice za funkcije jezika i biblioteke koje su obuhvaćene u *poglavljima 2-12*, koje žele da vide u akciji i da uče o njima bez potrebe da pišu kompletnu aplikaciju, čak i ako je to samo konzolna aplikacija:

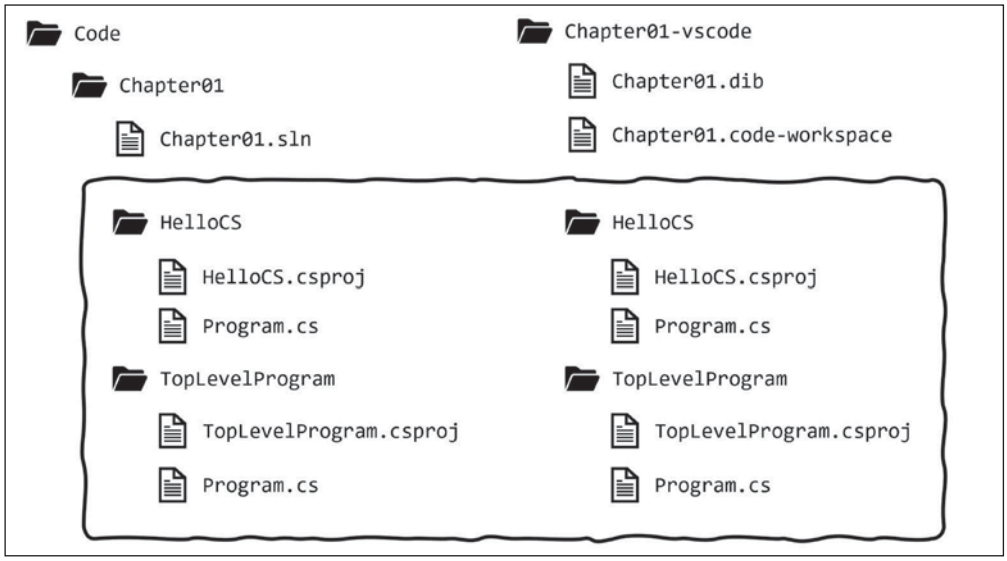
https://github.com/markjprice/cs10dotnet6/tree/main/notebooks

PREGLEDANJE DIREKTORIJUMA I FAJLOVA ZA PROJEKTE

U ovom poglavlju kreirali ste dva projekta pod nazivima `HelloCS` i `TopLevelProgram`.

Visual Studio Code koristi fajl radnog prostora za vođenje više projekata. Visual Studio 2022 koristi fajl rešenja za vođenje više projekata. Takođe ste kreirali .NET Interactive beležnicu.

Rezultat je struktura direktorijuma i fajlova koja će se ponavljati u narednim poglavljima, ali sa više od dva projekta, kao što je prikazano na *slici 1.18*:



Slika 1.18: Struktura direktorijuma i fajlovi za dva projekta u ovom poglavlju

Razumevanje uobičajenih direktorijuma i fajlova

Iako se fajlovi `.code-workspace` i `.sln` razlikuju, direktorijumi i fajlovi projekta, kao što su `HelloCS` i `TopLevelProgram` su identični za Visual Studio 2022 i Visual Studio Code. To znači da možete da mešate i uparujete oba uređivača koda, ako želite:

- U razvojnom okruženju Visual Studio 2022, sa otvorenim rešenjem, kliknite na **File | Add Existing Project…** da biste dodali fajl projekta kreiran drugim alatom.
- U uređivaču koda Visual Studio Code, sa otvorenim radnim prostorom, kliknite na **File | Add Folder to Workspace…** da biste dodali direktorijum projekta koji je kreirao drugi alat.



Dobra praksa: Iako je izvorni kod, kao što su fajlovi `.csproj` i `.cs`, identičan, direktorijumi `bin` i `obj`, koje automatski generiše kompajler mogu da imaju neusklađene verzije fajlova koji dovode do greške. Ako želite da otvorite isti projekat u razvojnom okruženju Visual Studio 2022 i u uređivaču koda Visual Studio Code, izbrišite privremene direktorijume `bin` i `obj` pre otvaranja projekta u drugom uređivaču koda. Zbog toga sam vas zamolio da kreirate drugi direktorijum za rešenja Visual Studio Code u ovom poglavlju.

Razumevanje koda rešenja u GitHub skladištu

Kod rešenja u GitHub skladištu za ovu knjigu sadrži odvojene direktorijume za fajlove uređivača koda Visual Studio Code, razvojnog okruženja Visual Studio 2022 i .NET Interactive beležnica, kao što je prikazano na sledećoj listi:

- Visual Studio 2022 rešenja:
  https://github.com/markjprice/cs10dotnet6/tree/main/vs4win
- Visual Studio Code rešenja:
  https://github.com/markjprice/cs10dotnet6/tree/main/vscode
- .NET Interactive Notebook rešenja:
  https://github.com/markjprice/cs10dotnet6/tree/main/notebooks



Dobra praksa: Ako je potrebno, vratite se na ovo poglavlje da biste se podsetili kako da kreirate i vodite više projekata u uređivaču koda po vašem izboru. GitHub skladište sadrži instrukcije korak-po-korak za četiri uređivača koda (Visual Studio 2022 for Windows,Visual Studio Code, Visual Studio 2022 for Mac i JetBrainsRider), zajedno sa dodatnim snimcima ekrana: https://github.com/markjprice/cs10dotnet6/blob/main/docs/code-editors/.

DOBRA UPOTREBA GITHUB SKLADIŠTA ZA OVU KNJIGU

Git je sistem za upravljanje izvornim kodom koji se najčešće koristi. GitHub je kompanija, veb sajt i desktop aplikacija koja olakšava upravljanje Git skladištem. Microsoft je kupio GitHub 2018. godine, pa će biti nastavljena bliža integracija sa Microsoft alatkama.

Ja sam upotrebio GitHub za ovu knjigu za sledeće:

- za skladištenje koda rešenja za knjigu, koji može da se održava i nakon datuma publikovanja.
- za obezbeđivanje dodatnog materijala koji proširuje knjigu, kao što su ispravke štamparskih grešaka, mala poboljšanja, liste korisnih linkova i duži članci koji ne mogu da se objave u štampanoj knjizi.
- za obezbeđivanje prostora čitaocima da bi mogli da kontaktiraju sa mnom ukoliko imaju problema u vezi sa knjigom.

Pitanja u vezi sa knjigom

Ukoliko se javi problem tokom praćenja instrukcija u ovoj knjizi, ili ako pronađete grešku u tekstu ili kodu u rešenjima, molim vas da me o tome obavestite na GitHub skladištu:

1. Upotrebite vaš omiljen pretraživač da biste otvorili sledeći link: https://github.com/markjprice/cs10dotnet6/issues.

2. Kliknite **New Issue**.
3. Unesite što je moguće više detalja koji će mi pomoći da dijagnostikujem problem. Na primer:

1. Vaš operativni sistem, na primer, Windows 11 64-bitni, ili macOS Big Sur verzija 11.2.3.
2. Vaš hardver, na primer, Intel, Apple Silicon ili ARM CPU.
3. Vaš uređivač koda, na primer, Visual Studio 2022, Visual Studio Code ili neki drugi, uključujući i broj verzije.
4. Onoliko koda i konfiguracije koliko smatrate da je relevantno i potrebno.
5. Opis očekivanog ponašanja i ponašanje koje ste iskusili.
6. Snimci ekrana (ako je moguće).

Pisanje ove knjige je moj sporedan posao. Imam posao sa punim radnim vremenom, pa obično na knjizi radim vikendom. To znači da ne mogu uvek odmah da odgovorim na pitanja. Ali moja želja je da svi čitaoci budu uspešni uz moju knjigu, pa ako mogu da vam pomognem bez previše muke, to ću rado uraditi.

Povratne informacije

Ako želite da mi pošaljete opšte povratne informacije o knjizi, upotrebite ankete na `README.md` stranici GitHub skladišta. Možete anonimno da pošaljete povratne informacije, ili ako želite da vam odgovorim, možete da unesete svoju email adresu. Tu adresu ću koristiti samo da vam odgovorim na pitanje.

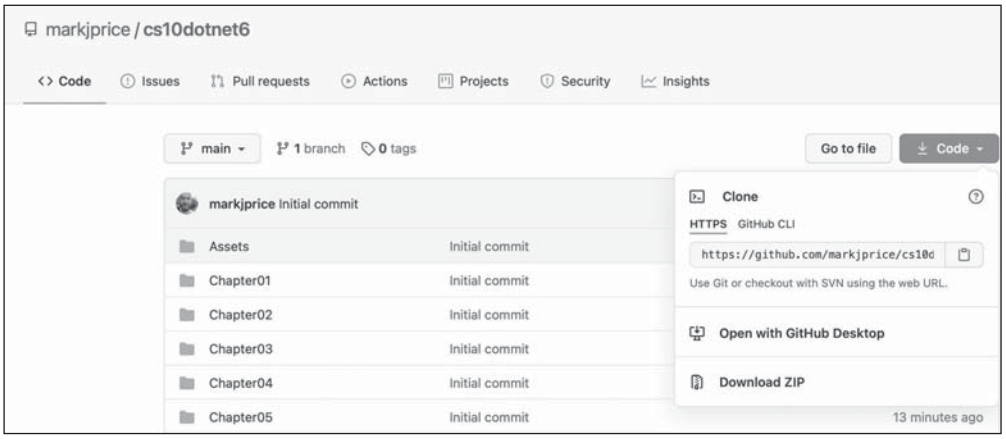
Volim da čujem mišljenje čitaoca, da mi kažu šta se njima dopalo u mojoj knjizi, kao i predloge za poboljšanje i načine na koji oni koriste jezik C# i .NET platformu, pa, prema tome, nemojte se stideti. Kontaktirajte sa mnom.

Unapred se zahvaljujem za vašu promišljenu i konstruktivnu povratnu informaciju.

Preuzimanje koda rešenja iz GitHub skladišta

Ja koristim GitHub za skladištenje rešenja za sve praktične primere kodiranja iz poglavlja i za praktične vežbe koje se nalaze na kraju svakog poglavlja. Skladište ćete pronaći na sledećem linku: https://github.com/markjprice/cs10dotnet6.

Ako samo želite da preuzmete fajlove rešenja, bez upotrebe Git skladišta, kliknite na zeleno dugme **Code** a zatim selektujte opciju **Download ZIP**, kao što je prikazano na *slici 1.19*:



Slika 1.19: Preuzimanje skladišta kao ZIP fajla

Preporučujem da dodate prethodni link u omiljene obeleživače jer ga koristim za GitHub skladište za ovu knjigu i za objavljivanje štamparskih grešaka (ispravki) i drugih korisnih linkova.

Upotreba Git skladišta pomoću uređivača koda Visual Studio Code i komandne linije

Visual Studio Code ima podršku za Git, ali će upotrebiti Git instalaciju vašeg operativnog sistema, pa prvo morate da instalirate Git 2.0 ili noviju verziju, pre nego što dobijete ove funkcije.

Možete da instalirate Git sa sledećeg linka: https://git-scm.com/download.

Ako želite da koristite GUI, možete da preuzmete GitHub Desktop sa sledećeg linka: https://desktop.github.com.

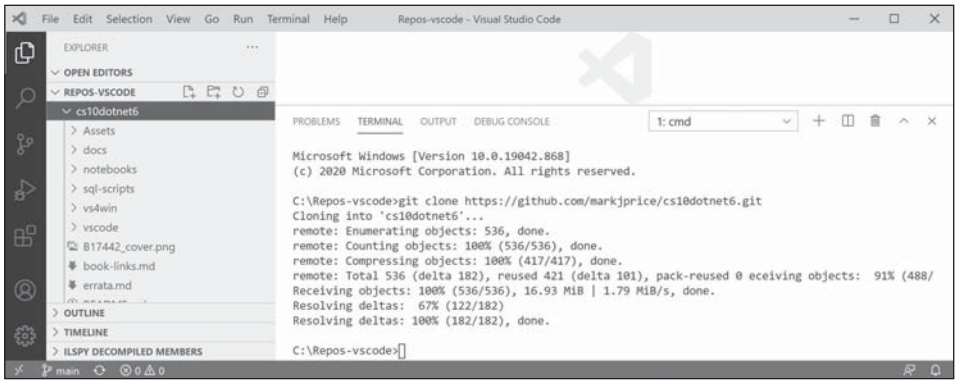
Kloniranje skladišta koda rešenja ove knjige

Sada ćemo klonirati skladište koda rešenja ove knjige. U koracima koji slede koristićete Visual Studio Code terminal, ali možete da unesete komande u bilo koji komandni odzivnik ili prozor terminala:

1. Kreirajte direktorijum, pod nazivom `Repos-vscode`, u vašem korisničkom direktorijumu, ili u direktorijumu `Documents`, ili u bilo kom direktorijumu u kom želite da sačuvate Git skladišta.
2. U uređivaču koda Visual Studio Code otvorite direktorijum `Repos-vscode`.
3. Kliknite na **View | Terminal** i unesite sledeću komandu:

```
git clone https://github.com/markjprice/cs10dotnet6.git
```

4. Kloniranje svih rešenja za sva poglavlja će potrajati nekoliko minuta, kao što je prikazano na *slici 1.20*:



Slika 1.20: Kloniranje koda rešenja za ovu knjigu korišćenjem uređivača koda Visual Studio Code

POTRAŽITE POMOĆ

U ovom odeljku pronaći ćete savete kako da pronađete kvalitetne informacije o programiranju na vebu.

Čitanje Microsoft dokumentacije

Definitivni resurs za dobijanje pomoći za Microsoft programerske alatke i platforme je Microsoft Docs, koji možete da nađete na linku https://docs.microsoft.com/.

Dobijanje pomoći za dotnet alatku

U komandnoj liniji možete da zatražite pomoć za opis komandi `dotnet` alatke.

1. Da biste otvorili zvaničnu dokumentaciju u prozoru pretraživača za `dotnet new` komandu, unesite sledeće u komandnu liniju ili u Visual Studio Code Terminal:

```
dotnet help new
```

2. Da biste dobili pomoćni ispis u komandnoj liniji, upotrebite `-h` ili `--help` oznaku, kao što je prikazano u sledećoj komandi:

```
dotnet new console -h
```

3. Videćete sledeći delimičan ispis:

```
Console Application (C#)
Author: Microsoft
Description: A project for creating a command-line application that can
run on .NET Core on Windows, Linux and macOS
```

```
Options:
  -f|--framework. The target framework for the project.
                      net6.0           - Target net6.0
                      net5.0           - Target net5.0
                      netcoreapp3.1.   - Target netcoreapp3.1
                      netcoreapp3.0.   - Target netcoreapp3.0
                    Default: net6.0

--langVersion    Sets langVersion in the created project file text
– Optional
```

Dobijanje definicija tipova i njihovih članova

Jedna od najkorisnijih funkcija u uređivaču koda je **Go To Definition**. Ona je dostupna u uređivačima koda Visual Studio Code i Visual Studio 2022. Funkcija prikazuje, čitanjem metapodataka u kompajliranom programskom sklopu, kako izgleda javna definicija tipa ili člana.

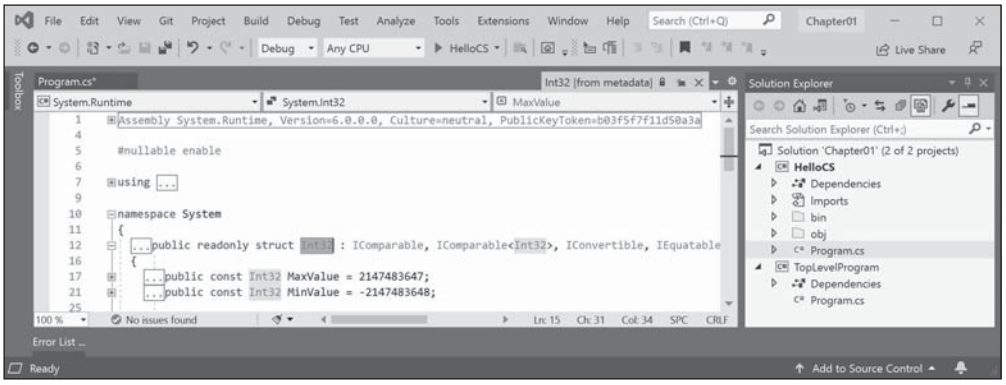
Neke alatke, kao što je ILSpy .NET Decompiler, izvršiće za vas čak i povratnu analizu iz metapodataka i IL koda nazad u C#.

U sledećim koracima opisaćemo kako da upotrebite funkciju **Go To Definition**:

1. U uređivaču koda Visual Studio 2022 ili Visual Studio Code otvorite rešenje / radni prostor pod nazivom `Chapter01`.
2. Da biste deklarisali promenljivu celog broja pod nazivom `z`, u fajl `Program.cs`, unutar metoda `Main` projekta `HelloCS`, unesite sledeći iskaz:

   ```
   int z;
   ```

3. Kliknite unutar deklaracije `int`, a zatim kliknite desnim tasterom miša i izaberite **Go To Definition**.
4. U prozoru koda koji će biti prikazan možete da vidite kako je definisan tip podataka `int`, kao što je prikazano na *slici 1.21*:



Slika 1.21: Metapodaci tipa podataka int

Možete da vidite da je `int`:

- definisan upotrebom ključne reči `struct`
- da se nalazi u programskom sklopu `System.Runtime`
- da se nalazi u imenskom prostoru `System`
- da mu je naziv `Int32`
- da je, prema tome, alijas za `System.Int32` tip
- da implementira interfejse, kao što je `IComparable`
- da ima konstantne maksimalnu i minimalnu vrednost
- da ima metode, kao što je `Parse`

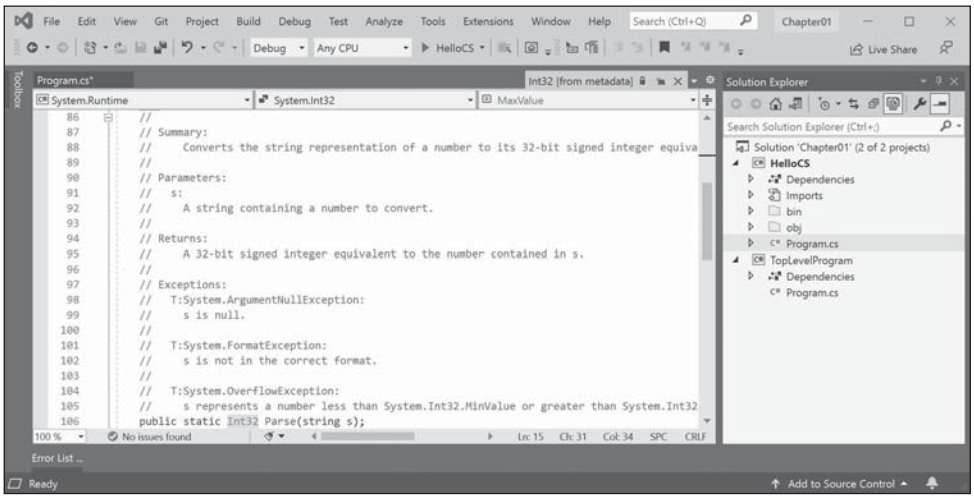


Dobra praksa: Kada pokušate da upotrebite opciju **Go To Definition** u uređivaču koda Visual Studio Code, ponekad ćete videti grešku **No definition found**. Razlog je činjenica da C# ekstenzija ne poznaje aktuelni projekat. Da biste ispravili ovu grešku kliknite na **View | Command Palette**, unesite omni, selektujte **OmniSharp: Select Project**, a zatim selektujte odgovarajući projekat koji želite da upotrebite.

Sada funkcija **Go To Definition** nije mnogo korisna, jer još ne znate šta sve ove informacije znače.

Do kraja prvog dela ove knjige, *poglavlja 2-6*, u kom ćete učiti o C# jeziku, ova funkcija će vam postati veoma korisna.

5. U prozoru uređivača koda skrolujte nadole da biste pronašli `Parse` metod sa jednim `string` parametrom u liniji 106, i komentare koji ga dokumentuju, u linijama od 86 do 105, kao što je prikazano na *slici 1.22*:



Slika 1.22: Komentari za Parse metod sa string parametrom

U komentarima ćete videti da je Microsoft dokumentovao sledeće:

- rezime koji opisuje metod
- parametre kao što je `string` vrednost, koji mogu da se proslede u metod
- vraćenu vrednost metoda, uključujući i njen tip podataka
- tri izuzetka koji se mogu javiti ako pozovete ovaj metod, uključujući `ArgumentNullException`, `FormatException` i `OverflowException`. Sada znate da možete da obuhvatite poziv za ovaj metod u `try` iskaz i koje izuzetke je potrebno da hvatate.

Pretpostavljam da ste nestrpljivi da naučite šta sve ovo znači!

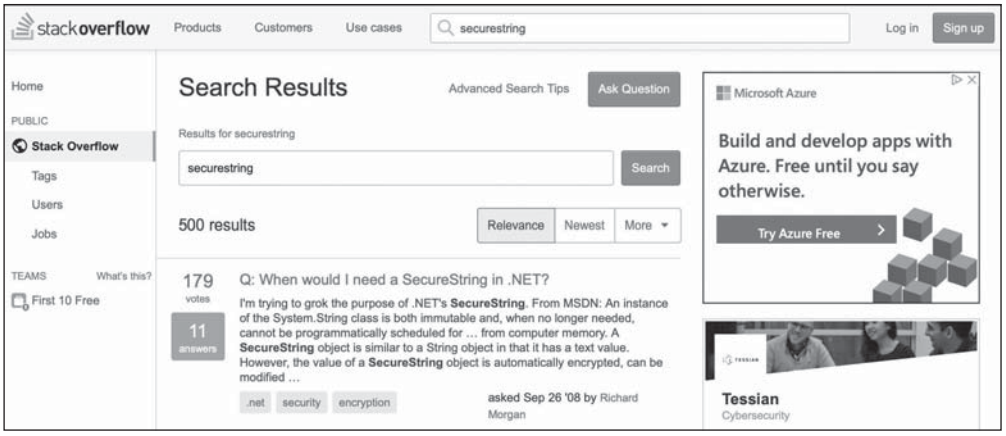
Budite još malo strpljivi. Skoro ste na kraju ovog poglavlja, a u sledećem ćete detaljnije upoznati C# jezik. Međutim, prvo pogledajte gde još možete da potražite pomoć.

Traženje odgovora na veb sajtu Stack Overflow

Stack Overflow je najpopularniji nezavisni veb sajt za dobijanje odgovora na teška programerska pitanja. Toliko je popularan da pretraživači, kao što je DuckDuckGo, imaju specijalan način za pisanje upita za njegovu pretragu.

1. Otvorite vaš omiljeni veb pretraživač.
2. Otvorite stranicu `DuckDuckGo.com`, unesite sledeći upit i vidite rezultate pretrage, koji su takođe prikazani na *slici 1.23*:

```
!so securestring
```



Slika 1.23: Rezultati pretrage za securestring na veb sajtu Stack Overflow

Pretraživanje odgovora upotrebom pretraživača Google

Možete da pretražujete Google, koristeći napredne opcije pretrage, da biste povećali verovatnoću pronalaženja onoga što tražite.

1. Otvorite Google.
2. Potražite informacije za `garbage collection`, koristeći jednostavan Google upit, i videćete verovatno mnogo reklama za garbage collection servise u vašem lokalnom okruženju, pre nego što vidite definiciju na Wikipedia sajtu za garbage collection u računarskoj nauci.
3. Poboljšajte pretragu tako što ćete je ograničiti na koristan sajt, kao što je Stack Overflow, i tako što ćete ukloniti jezike koji vas ne zanimaju, kao što su C++, Rust i Python, ili dodati eksplicitni jezik C# i .NET platformu, kao što je prikazano u sledećem upitu:

```
garbage collection site:stackoverflow.com +C# -Java
```

Pretplata na zvaničan .NET blog

Da biste pratili zbivanja u vezi sa .NET platformom, odličan blog na koji možete da se prijavite je zvaničan .NET Blog, koji su pisali timovi .NET inženjera, a možete da ga nađete na sledećem linku: https://devblogs.microsoft.com/dotnet/.

Pregled video snimaka Scotta Hanselmana

Scott Hanselman iz Microsoft-a ima odličan YouTube kanal o računarskim temama o kojima vas ne podučavaju: https://computerstufftheydidntteachyou.com/.

Preporučujem ovaj kanal svima koji koriste računar.

VEŽBANJE I ISTRAŽIVANJE

Testirajte svoje znanje tako što ćete odgovoriti na neka pitanja, uraditi praktične vežbe i istražiti detaljno teme koje su opisane u ovom poglavlju.

Vežba 1.1 – Testirajte svoje znanje

Pokušajte da odgovorite na sledeća pitanja (većinu odgovora možete da pronađete u ovom poglavlju, a za druge odgovore će biti potrebno istraživanje na Internetu ili pisanje koda):

1. Da li je Visual Studio 2022 bolji od uređivača koda Visual Studio Code?
2. Da li je .NET 6 bolji od radnog okvira .NET Framework?
3. Šta je .NET Standard i zašto je i dalje važan?

4. Zašto programer može da upotrebi različite jezike (na primer, C# i F#) za pisanje aplikacija koje se pokreću na .NET platformi?
5. Koji je naziv metoda za unos .NET konzolne aplikacije i kako bi trebalo da bude deklarisan?
6. Šta je program najvišeg nivoa i kako pristupate bilo kom argumentu komandne linije?
7. Šta je potrebno da ukucate u odzivnik da biste izgradili i izvršili C# izvorni kod?
8. Koje su neke od prednosti upotrebe ekstenzije .NET Interactive Notebooks za pisanje C# koda?
9. Gde ćete potražiti pomoć o C# ključnim rečima?
10. Gde ćete potražiti rešenja za uobičajene programerske probleme?



Dodatak, „Odgovori na pitanja za testiranje znanja“, dostupan je za preuzimanje sa linka u README fajlu na GitHub skladištu: https://github.com/markjprice/cs10dotnet6.

Vežba 1.2 – Vežbajte C# svuda

Nisu vam potrebni Visual Studio Code, ili, čak, Visual Studio 2022 za Windows ili za Mac da biste pisali C# kod. Možete da otvorite stranicu .NET Fiddle - https://dotnetfiddle.net/ - i da počnete kodiranje online.

Vežba 1.3 – Istražite teme

Pokušao sam da pronađem odgovarajuću ravnotežu među temama koje sam uključio u štampanu knjigu. Ostali sadržaj koji sam napisao možete pronaći u GitHub skladištu za ovu knjigu.

Verujem da ova knjiga obuhvata osnovno znanje i veštine koje bi programer, koji koristi jezik C# i .NET platformu, trebalo da ima. Neki duži primeri su uključeni kao linkovi ka Microsoft dokumentaciji ili ka člancima nezavisnih autora.

Upotrebite linkove na sledećoj stranici da biste naučili više o temama koje su opisane u ovom poglavlju:

https://github.com/markjprice/cs10dotnet6/blob/main/book-links.md#chapter-1---hello-c-welcome-net