Mark J. Price

Istraživanje poboljšanja jezika C# 7.2 i 7.3

C# 7.1 i .NET Core 2.0

Moderno međuplatformsko programiranje

III izdanje

Kreirajte moćne aplikacije pomoću .NET Standarda 2.0, ASP.NET Corea 2.0 i Entity Framework Corea 2.0, koristeći Visual Studio 2017 ili Visual Studio Code

kompjuter biblioteka

Packt>

Mark J. Price

C# 7.1 i .NET Core 2.0

Moderno međuplatformsko programiranje

kompjuter biblioteka

Packt>

Izdavač:

kompjuter biblioteka

Obalskih radnika 4a, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:
Mihailo J. Šolajić

Autor: Mark J. Price

Prevod: Slavica Prudkov

Lektura: Miloš Jevtović

Slog: Zvonko Aleksić

Znak Kompjuter biblioteke:
Miloš Milosavljević

Štampa: „Svetlost“, Čačak

Tiraž: 500

Godina izdanja: 2018.

Broj knjige: 501

Izdanje: Prvo

ISBN: 978-86-7310-524-6

C# 7.1 and .NET Core 2.0 – Modern Cross-Platform Development
Mark J. Price

ISBN 978-1-78839-807-7

O AUTORU

Mark J. Price je stekao sertifikate za Microsoft Certified Solutions Developer (MCSD), Microsoft Specialist: Programming in C# i Episerver Certified Developer i ima više od 20 godina iskustva u edukaciji i programiranju.

Od 1993. godine položio je više od 80 „Microsoftovih" ispita za programiranje i specijalizovao se za pripremanje drugih korisnika za polaganje ispita. Njegovi učenici su i profesionalci koji imaju iza sebe decenije bogatog iskustva i mladi ljudi koji uopšte nemaju iskustva. On uspešno vodi sve njih kombinovanjem obrazovnih veština sa primerima iz stvarnog sveta, konsultujući se i razvijajući sisteme za preduzeća širom sveta. Između 2001. i 2003. godine Mark je bio zaposlen, sa punim radnim vremenom, za pisanje zvaničnog materijala za obuku za „Microsoft" u Redmondu (SAD). Njegov tim je napisao prvi materijal za obuku za C# dok je taj jezik još uvek bio u ranoj alfa verziji. Dok je radio u „Microsoftu", održao je kurs „obuči-instruktora" da bi obučio ostale MCT sertifikovane stručnjake za C# i .NET.

Trenutno, Mark piše i isporučuje materijale za obuke za „Episerverov" Digital Experience Cloud, najbolji .NET CMS za Digital Marketing i E-commerce. Stekao je sertifikate Episerver Certified Developer (ECD) na Episerver CMS-u.

Mark je 2010. godine stekao postdiplomski sertifikat u obrazovanju (PGCE). Predavao je GCSE i matematiku A-nivoa u dve srednje škole u Londonu. Ima Computer Science BSc (Hons) diplomu stečenu na Univerzitetu u Bristolu, u Velikoj Britaniji.

Zahvaljujem se mojim roditeljima Pameli i Ianu što su me vaspitali da budem ljubazan, vredan i radoznao. Zahvaljujem se mojim sestrama Emily i Juliet što me vole, bez obzira što sam njihov čudan stariji brat. Takođe izražavam zahvalnost mojim prijateljima i kolegama koji me inspirišu tehnički i kreativno. Na kraju, hvala učenicima koje sam podučavao tokom godina što su me podsticali da budem bolji profesor.

O RECENZENTIMA

Dustin Heffron je danju softverski inženjer, a noću nezavisni programer igrica. Ima više od 10 godina iskustva u programiranju u različitim jezicima i osam godina iskustva rada u C#-u i .NET-u. Trenutno, Dustin razvija alatke za automatizovanje i testiranje medicinskih instrumenata u Becton Dickinson instituciji. Takođe je suosnivač i CEO za SunFlake Studios.

Dustin već dugo vrši recenzije za „Packt“, uključujući knjige „XNA 4.0 Game Development by Example: Beginner‘s Guide“, „C# 6“ i „.NET Core 1.0: Moderan međuplatformski razvoj“ i seriju video uputstava XNA 3D Programming by Example. Takođe je koautor serije video uputstava XNA 3D Toolkit (drugi koautor je Larry Louisiana).

Efraim Kyriakidis je vešt softverski inženjer, sa više od 10 godina iskustva u razvoju i pružanju softverskih rešenja za različite klijente i projekte. On je dobro upućen u sve faze razvoja softvera. Prvi dodir sa računarima i programiranjem je imao kao dete, 1980-ih godina, na slavnom Commodore 64 računaru. Ima diplomu elektrotehničkog inženjera sa Aristotle univerziteta u Solunu iz Grčke. Uglavnom je radio na Microsoft tehnologijama, koristeći C# i .NET od verzije .NET 1.0. Trenutno radi za „Siemens AG“ u Nemačkoj kao programer.

Kratak sadržaj

Sadržaj

POGLAVLJE 1

POGLAVLJE 3

POGLAVLJE 4

POGLAVLJE 5

UVOD

Postoji mnoštvo knjiga za C# (neke od njih imaju i više od 1.000 strana), koje su autori „predodredili“ da budu sveobuhvatne reference za C# programski jezik i .NET Framework.

Ova knjiga je drugačija – ona je koncizna, jednostavna za čitanje i ima praktične vodiče. Napisao sam je u nameri da bude najbolji vodič, korak-po-korak, za modernu međuplatformsku C# i .NET praksu.

Istaći ću sve interesantne delove C#-a da biste mogli da impresionirate svoje kolege i zaposlene i brzo postanete produktivni. Da vam ne bih dosađivao i polako objašnjavao sve sitnice, pretpostavljam da, ako je neki od termina koje koristim za vas nov, znate kako da ga potražite odgovor korišćenjem Googlea.

Na kraju svakog poglavlja postoji odeljak „Vežbanje i istraživanje“, koji sadrži praktične vežbe i u kojem možete detaljnije sami da istražite temu, uz moju malu pomoć, tako što ću vas usmeriti u odgovarajućem smeru.

Možete da preuzmete rešenja za vežbe iz sledećeg GitHub skladišta. Obezbediću vam i instrukcije kako da to uradite, koristeći Visual Studio 2017 i Visual Studio Code, na kraju Poglavlja 1, „Zdravo C#!, dobrodošao .NET Core!“.

https://github.com/markjprice/cs7dotnetcore2.

ŠTA OBUHVATA OVA KNJIGA?

Poglavlje 1, „*Zdravo C#!, dobrodošao .NET Core!*“, posvećeno je podešavanju razvojnog okruženja i upotrebi različitih alata za kreiranje najjednostavnijih mogućih aplikacija pomoću C#-a. Naučićete kako da napišete i kompajlirate kod, koristeći Visual Studio 2017 na Windows sistemu, Visual Studio Code na Mac OS X, Linux i Windows sistemu ili Visual Studio for Mac n Mac OS X sistemu. Takođe ćete naučiti više o.NET tehnologijama .NET Framework, .NET Core, .NET Standard i .NET Native.

Deo 1 – C# 7.1

U Poglavlju 2, „*Govoriti C# jezikom*“, biće reči o gramatici i rečniku koji ćete koristiti svakog dana za pisanje izvornog koda za vaše aplikacije. Konkretno, naučićete kako da deklarišete i upotrebite različite vrste promenljivih.

Poglavlje 3, „*Kontrolisanje toka i konvertovanje tipova*“, posvećeno je pisanju koda koji donosi odluke, ponavlja blokove iskaza i konvertuje tipove i pisanju koda za obradu grešaka kada se one dese. Takođe ćete naučiti koja su najbolja mesta na kojima možete zatražiti pomoć.

Poglavlje 4, „*Pisanje, ispravljanje grašaka i testiranje funkcija*“, posvećeno je praćenju principa Ne Ponavljaj Se (Don’t repeat yourself – DRY) pisanjem ponovo upotrebljivih funkcija, a naučićete kako da alatke za ispravljanje grešaka upotrebite za praćenje i uklanjanje programskih grešaka, kontrolisanje koda dok se izvršava da biste dijagnostifikovali probleme i strogo testiranje koda da biste uklonili programske greške i obezbedili stabilnost i pouzdanost pre nego što je kod primenjen u proizvodnji.

U Poglavlju 5, „*Kreiranje sopstvenih tipova pomoću objektno-orijentisanog programiranja*“, tema su različite kategorije članova koje tipovi mogu imati, uključujući polja za čuvanje podataka i metode za izvršavanje akcija. Upotrebićete OOP koncepte, kao što su agregacija i kapsuliranje. Naučićete i koje su funkcije C# 7 jezika, kao što je podrška za sintaksu torke i izlazne promenljive, i funkcije C# 7.1 jezika, kao što su standardni literali i izvedeni nazivi torke.
U Poglavlju 6, „Implementiranje interfejsa i nasleđivanje klasa“, osnovna tema je isporučivanje novih tipova iz postojećih pomoću objektno-orijentisanog programiranja (OOP). Naučićete kako da definišete operatore i C# 7 lokalne funkcije, prosleđivanja i događaje, da implementirate interfejse o osnovnim i izvedenim klasama, da promenite vrednost člana tipa, da upotrebite polimorfizam, da kreirate metode ekstenzije i da se postavite između klasa u hijerarhiji nasleđivanja.

Deo 2 - .NET Core 2.0 i .NET Standard 2.0

U Poglavlju 7, „*Razumevanje i pakovanje .NET Standard tipova*“, predstavljeni su .NET Core 2.0 tipovi koji su deo .NET Standarda 2.0 i načini na koji su oni povezani sa C# jezikom. Naučićete kako da primenite i pakujete sopstvene aplikacije i biblioteke.

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

U Poglavlju 9, „*Upotreba fajlova, nizova i serijalizacije*“, biće reči o interakciji fajl sistema, čitanju i pisanju u fajlove i tokove, šifriranju teksta i serijalizaciji.
U Poglavlju 10, „Zaštita podataka i aplikacija“, saznaćete kako se podaci štite pomoću enkripcije od pregleda zlonamernih korisnika, a pomoću heširanja i označavanja od manipulisanja i oštećenja. Takođe ćete učiti o proveri identiteta i autorizaciji za zaštitu aplikacije od neautorizovane upotrebe.

Poglavlje 11, „*Upotreba baza podataka pomoću Entity Framework Corea*“, posvećeno je čitanju i pisanju u baze podataka, kao što su Microsoft SQL Server i SQLite, uz korišćenje objektno-relacione tehnologije mapiranja pod nazivom Entity Framework Core.

U Poglavlju 12, „*Slanje upita i manipulisanje podacima pomoću LINQ-a*“, tema su Language Integrated Queries (LINQ) – ekstenzije jezika koje dodaju mogućnost upotrebe sekvenci stavki, filtriranja, sortiranja i projektovanja u različite ispise.

U Poglavlju 13, „*Poboljšanje performanse i skalabilnosti koristeći višeprogramski rad*“, saznaćete kako se obezbeđuje istovremeno izvršenje više akcija radi poboljšanja performanse, skalabilnosti i produktivnosti korisnika. Učićete o C# 7.1 funkciji async Main i o načinu kako se upotrebljavaju tipovi u imenskom prostoru System.Diagnostics za kontrolisanje koda i merenje performansi i efikasnosti.

Deo 3 – Modeli aplikacije

Poglavlje 14, „*Izgradnja veb sajtova pomoću ASP.NET Core Razor Pagesa*“, posvećeno je učenju osnova izgradnje veb sajtova pomoću moderne HTTP arhitekture na strani servera upotrebom ASP.NET Core-a. Učićete o novoj funkciji ASP.NET Corea, koja je poznata kao Razor Pages i koja pojednostavljuje kreiranje veb stranica za male veb sajtove.

U Poglavlju 15, „*Kreiranje veb sajtova pomoću ASP.NET Core MVC-a*“, biće reči o izgradnji velikih, složenih veb sajtova na način koji je lako testirati i upravljati u tipovima programera, upotrebom ASP.NET Core-a. Učićete o početnoj konfiguraciji, proveri identiteta, rutama, modelima, prikazima i kontrolerima u ASP.NET Core MVC-u.

U Poglavlju 16, „*Kreiranje veb servisa i aplikacija pomoću ASP.NET Corea*“, opisana je izgradnja veb aplikacija pomoću kombinacije modernih čeonih tehnologija, kao što su Angular i React, i pozadinskog veb servisa REST arhitekture pomoću ASP.NET Core Web API-ja.

Poglavlje 17, „*Izgradnja Windows aplikacija pomoću XAML-a i Fluent Designa*“, posvećeno je osnovama XAML-a koji možete da upotrebite za definisanje korisničkog interfejsa za grafičke aplikacije za Universal Windows Platform (UWP) i primenite principe i funkcije Fluent Design-a. Ova aplikacija zatim može da se pokrene na bilo kom uređaju koji pokreće Windows 10, Xbox One i čak i na Mixed Reality uređajima kao što je HoloLens.

U Poglavlju 18, „*Izgradnja aplikacija za mobilne uređaje pomoću XAML-a i Xamarin. Formsa*“, opisan je prenos C# jezika na mobilne uređaje izgradnjom međuplatformske aplikacije za iOS i Android. Aplikacije za mobilne uređaje na strani klijenta će biti kreirane pomoću Visual Studioa za Mac i pomoću XAML-a i Xamarin.Formsa.

Dodatak „Odgovori za testiranje znanja“ sadrži odgovore na pitanja postavljena na kraju svakog poglavlja.

ŠTA VAM JE POTREBNO ZA OVU KNJIGU?

C# možete da koristite na mnogim platformama, uključujući Windows, Mac OS X i mnoge distribucije Linuxa. Za najbolje iskustvo u programiranju, i da bi kod bio dostupan na većini platformi, preporučujem da naučite osnove svih članova porodice Visual Studio: Visual Studio 2017, Visual Studio Code i Visual Studio for Mac.

Moja preporuka za kombinaciju operativnog sistema i razvojne alatke je sledeća:

- Visual Studio 2017 za Windows 10
- Visual Studio for Mac za Mac OS X
- Visual Studio Code za Windows 10 ili Mac OS X

Najbolja verzija Windowsa koju možete da upotrebite je Microsoft Windows 10 zato što vam je ova verzija potrebna da biste kreirali Universal Windows Platform aplikacije u Poglavlju 17, Izgradnja Windows aplikacija pomoću XAML-a i Fluent Designa. Starije verzije Windowsa, kao što je 7 ili 8.1, će biti dovoljne za sva ostala poglavlja.

Najbolja verzija Mac OS X-a koju možete da upotrebite je Sierra ili High Sierra jer će vam biti potreban Mac OS X za kreiranje iOS mobilne aplikacije u Poglavlju 18, Izgradnja aplikacija za mobilne uređaje pomoću XAML-a i Xamarin.Formsa. iako možete da upotrebite Visual Studio 2017 na Windows sistemu za pisanje koda za iOS i Android mobilne aplikacije, potrebno je da imate macOS i Xcode da biste ih kompajlirali.

ZA KOGA JE OVA KNJIGA?

Ako ste čuli da je C# popularan programski jezik za osnovnu namenu koji se koristi za kreiranje svakog tipa softvera, od veb aplikacija i servisa do poslovnih aplikacija i igara, onda je ova knjiga za vas.

Ako ste čuli da u programskom jeziku C# možete da kreirate softver koji se pokreće na različitim uređajima, od desktopa do servera, od mobilnih uređaja do sistema za igrice kao što je Xbox One, onda je ova knjiga za vas.

Ako ste čuli da je .NET Core Microsoftova međuplatformska alatka .NET budućnosti, optimizovana za veb razvoj na strani servera u oblaku, i za Augmented Reality (AR) ili Virtual Reality (VR) uređaje kao što je HoloLens, onda je ova knjiga za vas.

Ako ste čuli da Microsoft ima popularnu međuplatformsku razvojnu alatku pod nazivom Visual Studio Code, koja kreira ove međuplatformske aplikacije, i zainteresovani ste za nju, onda je ova knjiga za vas.

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";
```

Kada želimo da privučemo vašu pažnju na određeni deo bloka koda, relevantne linije ili stavke će biti ispisane podebljanim slovima:

```
// 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 napisane podebljanim slovima. Reči koje vidite na ekranu, na primer, u menijima ili okvirima za dijalog, prikazaće se u tekstu na sledeći način: „Kliknite na dugme Next da biste se prebacili na sledeći ekran.“

Upozorenja ili važne napomene će biti prikazivani u ovakvom okviru.

Dobra praksa
Preporuke kako da programirate kao stručnjak prikazne su ovako.

KORISNIČKA PODRŠKA

Sada, kada ste vlasnik ove knjige, mi imamo mnogo toga da vam ponudimo da bismo vam pomogli da dobijete maksimum iz svoje narudžbe.

PREUZIMANJE PRIMERA KODA

Možete da preuzmete fajlove sa primerima koda za ovu knjigu sa našeg sajta na adresi :

https://goo.gl/q4e6qE

ŠTAMPARSKE GREŠKE

Iako smo preduzeli sve mere da bismo obezbedili tačnost sadržaja, greške se dešavaju. Ako pronađete grešku u nekoj od naših knjiga – možda grešku u tekstu ili u kodu – bili bismo zahvalni ako biste nam to prijavili. Na taj način možete da poštedite čitaoce od frustracije i nama da pomognete da poboljšamo naredne verzije ove knjige. Ako pronađete neku štamparsku grešku, molimo vas da nas obavestite tako što ćete posetiti stranicu knjige:

https://goo.gl/r2CRVU

Na dnu stranice, kliknite:

Ostavite komentar

I unesite grešku koju ste pronašli.

Kada je greška verifikovana, vaša prijava će biti prihvaćena i greška će biti na veb stranici knjige.

PIRATERIJA

Piraterija autorskog materijala na internetu je aktuelan problem na svim medijima. U Packtu, mi zaštitu autorskih prava i licenci shvatamo veoma ozbiljno. Ako pronađete ilegalnu kopiju naših knjiga, u bilo kojoj formi, na internetu, molimo vas da nas o tome obavestite i pošaljete nam adresu lokacije ili naziv web sajta da bismo mogli da podnesemo tužbu.

Molimo vas, kontaktirajte nas na adresi informatori@kombib.rs i pošaljite nam link ka sumnjivom materijalu.

Zahvalni smo vam na pomoći u zaštiti naših autora i mogućnosti da vam pružimo vredan sadržaj.

PRIJAVLJIVANJE NA MEJLING LISTU

Ukoliko već niste na našoj listi, posetite veb stranu: https://goo.gl/2GQxcS

Zašto da budete na listi?

Redovno ćete dobijati informatore o knjiga koje su u pripremi, a na koje možete da se pretplatite. Knjige u pretplati su 40% jeftinije od knjiga koje su izašle iz štampe.

Ukoliko ste naš registrovan kupac i na mejling listi, jednom godišnje možete da kupite bilo koju našu knjigu sa 50% popusta na dan kada vam je rođendan.

Knjige sa specijalnim popustom mogu da kupuju samo kupci koji su na mejling listi.

Prijavljivanje na mejling listu

Potrebno je da popunite samo tri podatka:

- imejl adresu
- ime
- prezime

1

Zdravo C#!, dobrodošao .NET Core!

Ovo poglavlje je posvećeno podešavanju razvojnog okruženja, razumevanju sličnosti i razlika između .NET Corea, .NET Frameworka, .NET Standarda i .NET Nativea i upotrebi različitih alatki za kreiranje najjednostavnijih mogućih aplikacija pomoću C# jezika i .NET Corea.

Većina korisnika uči o komplikovanim temama imitacijom i ponavljanjem, umesto čitanjem detaljnih teoretskih objašnjenja. Dakle, neću objašnjavati svaku ključnu reč i svaki korak. Cilj je da vas naučim da napišete kod, da izgradite aplikaciju i da je pokrenete. Još ne treba da znate detalje kako sve to funkcioniše.

Samuel Johnson, autor knjige „English dictionary“ iz 1755. godine, napisao je: „Verovatno sam napravio nekoliko kardinalnih grešaka i napisao nekoliko apsurdnih izraza, od kojih ni jedan posao ovakvog opsega nije pošteđen“. Preuzimam potpunu odgovornost za greške i nadam se da ćete ceniti moj trud uložen u pisanje knjige o .NET Coreu i njegovim alatkama komandne linije tokom njegovog „rađanja“ u toku 2016. i 2017. godine.

Ovo poglavlje obuhvata sledeće teme:

- podešavanje razvojnog okruženja
- razumevanje .NET-a
- pisanje i kompajliranje koda pomoću .NET Core CLI alatke
- pisanje i kompajliranje koda pomoću Visual Studio 2017 alatke
- pisanje i kompajliranje koda pomoću Visual Studio Codea
- pisanje i kompajliranje koda pomoću Visual Studio for Mac alatke
- upravljanje izvornim kodom pomoću GitHuba

PODEŠAVANJE RAZVOJNOG OKRUŽENJA

Pre nego što započnete da programirate, potrebno je da izaberete Intgrated Development Environment (IDE), koji uključuje editor koda za C#. „Microsoft“ ima familiju IDE-ova:

- Visual Studio 2017
- Visual Studio for Mac
- Visual Studio Code

Najstariji i potpuno opremljeni IDE koji možete da izaberete je **Microsoft Visual Studio 2017**, ali se on pokreće samo na Windows operativnom sistemu.

Najsavremeniji i najjednostavniji IDE koji možete da izaberete i jedini iz „Microsofta“ koji je međuplatformski, je **Microsoft Visual Studio** `Code` - on se pokreće na svim operativnim sistemima, uključujući Windows, macOS i mnoge distribucije Linuxa, kao što su **Red Hat Enterprise Linux (RHEL)** i Ubuntu.

Kao pomoć u odluci da li je Visual Studio Code odgovarajući za vas, preporučujem da pogledate video Beginner's Guide to VS Code: Up and Running in Ten Minutes: https://channel9.msdn.com/Blogs/raw-tech/Beginners-Guide-to-VS-Code

IDE koji je najusklađeniji i koji možete da izaberete za razvoj aplikacija za mobilne uređaje je **Visual Studio for Mac**. Da biste kreirali aplikaciju za iOS (iPhone i iPad), tvOS, macOS i watchOS, treba da imate macOS i Xcode. Iako možete da upotrebite Visual Studio 2017 sa njegovim Xamarin ekstenzijama za *pisanje* međuplatformske aplikacije za mobilne uređaje, i dalje su vam potrebni macOS i Xcode da biste *kompajlirali* aplikaciju.

U sledećoj tabeli prikazano je koji IDE i operativni sistemi mogu ili moraju da budu upotrebljeni za svako od poglavlja ove knjige:

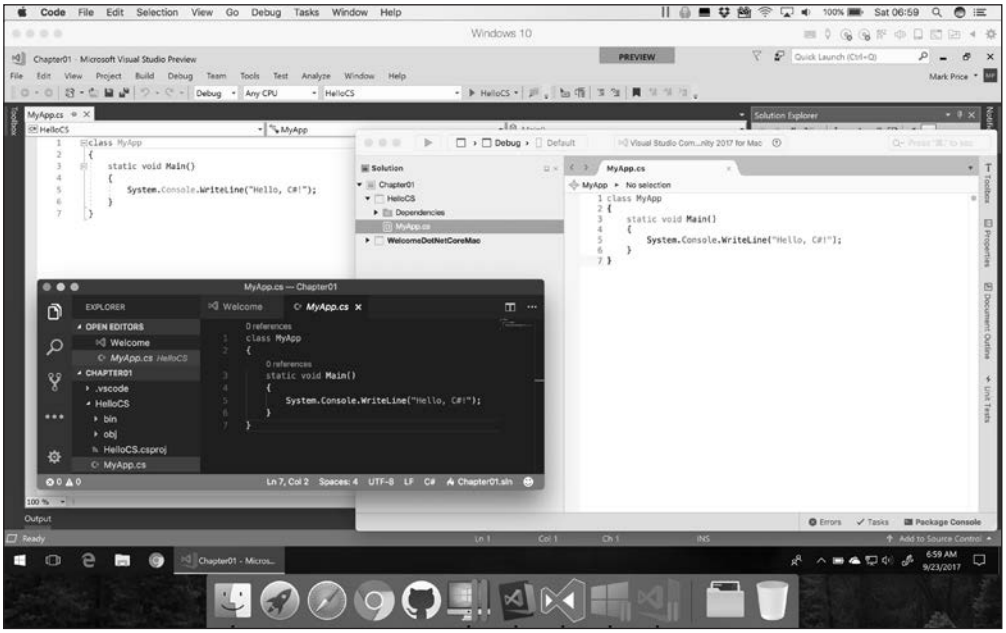
POGLAVLJA	IDE	OPERATIVNI SISTEMI
Poglavlja od 1 do 16	Visual Studio 2017	Windows 7 SP1 ili noviji
Poglavlja od 1 do 16	Visual Studio Code	Windows, macOS, Linux
Poglavlja od 1 do 16	Visual Studio for Mac	macOS
Poglavlje 17	Visual Studio 2017	Windows 10
Poglavlje 18	Visual Studio for Mac	macOS

Dobra praksa

Ako imate mogućnosti, preporučujem da izradite sve vežbe kodiranja pomoću IDE-a Visual Studio 2017 na Windowsu, Visual Studio Codea na macOS-u, Linuxu ili Windowsu i Visual Studio for Mac. Biće veoma korisno da isprobate C# i .NET Core na različitim operativnim sistemima i različitim razvojnim alatkama.

Za pisanje trećeg izdanja ove knjige ja sam upotrebio sledeći softver (što je prikazano i na sledećoj slici):

- Visual Studio 2017 na sistemu Windows 10 u virtuelnoj mašini
- Visual Studio for Mac na macOS sistemu
- Visual Studio Code na macOS sistemu
- Visual Studio Code na RHEL sistemu (nije prikazano na slici)



Upotreba alternativnih C# IDE-ova

Postoje i alternativni IDE-ovi za C# - na primer, MonoDevelop i JetBrains Rider. Možete da instalirate bilo koji od ova dva IDE-a praćenjem sledećih URL-ova:

- Za MonoDevelop IDE posetite stranicu http://www.monodevelop.com/.
- Za JetBrains Rider posetite stranicu https://www.jetbrains.com/rider/.

Cloud9 je IDE zasnovan na veb pretraživaču, pa je usklađeniji za sve plaforme od ostalih IDE-ova. Ovaj IDE je sve popularniji. Evo i linka: https://c9.io/web/sign-up/free.

Primena međuplatformskog razvoja

Vaš izbor IDE-a i operativnog sistema za razvoj neće ograničiti primenu koda. .NET Core 2.0 podržava sledeće platforme za primenu:

- Windows 7 SP1 ili noviji
- Windows Server 2008 R2 SP1 ili noviji
- Windows IoT 10 ili noviji
- macOS Sierra (version 10.12) ili noviji
- RHEL 7.3 ili noviji
- Ubuntu 14.04 ili noviji
- Fedora 25 ili noviji
- Debian 8.7 ili noviji
- openSUSE 42.2 ili noviji
- Tizen 4 ili noviji

Linux operativni sistemi su popularne platforme za hostovanje servera, jer su relativno mali i mnogo isplativiji, u poređenju sa platformama, kao što su Windows i macOS.

U sledećem odeljku ćete instalirati Microsoft Visual Studio 2017 za Windows. Ako želite radije da upotrebite Microsoft Visual Studio `Code`, pređite na odeljak „Instaliranje *Microsoft Visual Studio Codea za Windows, macOS ili Linux*“. Ako želite da upotrebite *Microsoft Visual Studio for Mac*, pređite na odeljak „I*nstaliranje Microsoft Visual Studioa for Mac*“.

Instaliranje IDE-a Microsoft Visual Studio 2017

Možete da upotrebite Windows 7 SP1 ili noviji da biste uradili vežbe iz većine poglavlja ove knjige, ali ćete steći mnogo bolje iskustvo ako upotrebite Windows 10 Fall Creators Update.

Od oktobra 2014. godine „Microsoft“ je kreirao izdanje profesionalnog kvaliteta IDE-a Visual Studio, koje je dostupno besplatno. To je izdanje **Community Edition**.

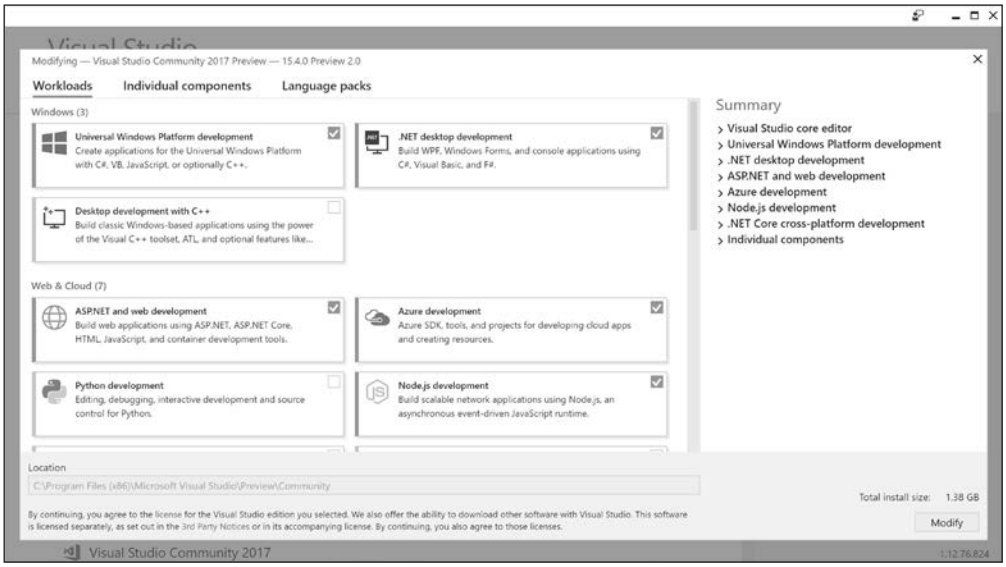
Preuzmite i instalirajte Microsoft Visual Studio 2017 verziju 15.4 ili noviju sa linka https://www.visualstudio.com/downloads/.

Treba da instalirate verziju 15.4 ili noviju Visual Studio 2017 da biste mogli da koristite .NET Core for UWP. Morate da instalirate verziju 15.3 ili noviju Visual Studio 2017 da biste mogli da koristite .NET Core 2.0. Starije verzije Visual Studio 2017 podržavaju samo .NET Core 1.0 i 1.1.

Biranje radnog opterećenja

Na kartici Workloads izaberite sledeće (kao što je delimično prikazano na sledećoj slici):

- Universal Windows Platform development
- .NET desktop development
- ASP.NET and web development
- Azure development
- Node.js development
- .NET Core cross-platform development



Biranje dodatnih komponenata

Na kartici **Individual components** izaberite sledeće dodatne komponente:

- Class Designer
- GitHub extension for Visual Studio
- PowerShell tools

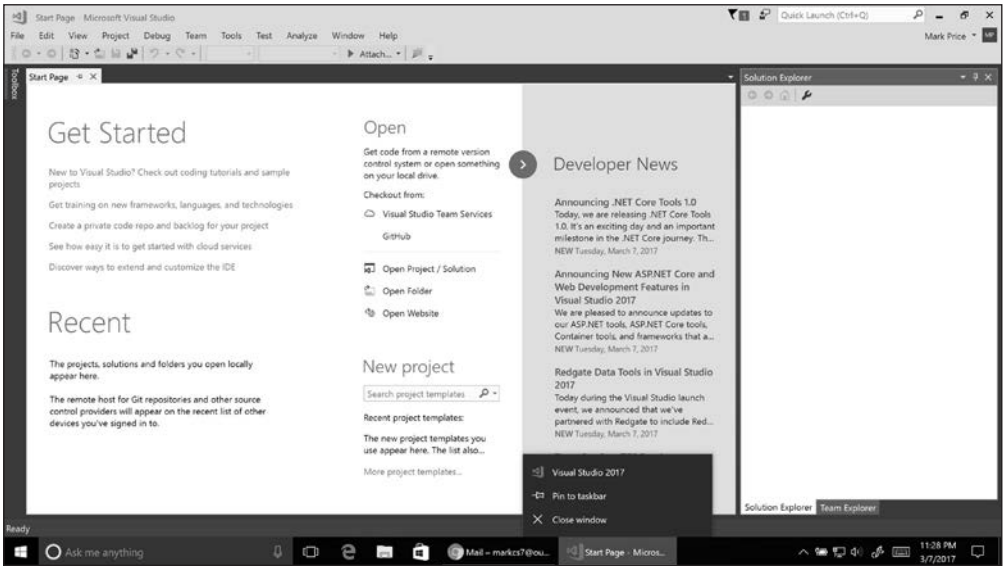
Kliknite na **Install** i čekajte dok instaler ne preuzme selektovani softver i instalira ga. Kada je instalacija završena, kliknite na **Launch**.

Dok čekate da se Visual Studio 2017 instalira, možete da nastavite čitanje odeljka „Razumevanje .NET-a“.

Prvi put kada pokrenete Visual Studio 2017 biće zatraženo da se prijavite. Ako imate Microsoft nalog, možete da ga upotrebite. Ako ga nemate, onda se registrujte na linku https://signup.live.com/.

Kada pokrenete Visual Studio 2017 prvi put, biće od vas zatraženo da konfigurišete okruženje. Za **Development Settings** izaberite **Visual C#**. Za kolornu temu ja sam izabrao **Blue**, ali možete da izaberete koju god želite.

Videćete Microsoft Visual Studio korisnički interfejs sa otvorenom početnom stranicom u centralnom području. Kao i većina Windows desktop aplikacija, Visual Studio ima liniju menija, liniju sa alatkama za uobičajene komande i statusnu liniju u podnožju stranice. Sa desne strane prozora nalazi se **Solution Explorer**, u kojem su izlistani otvoreni projekti:

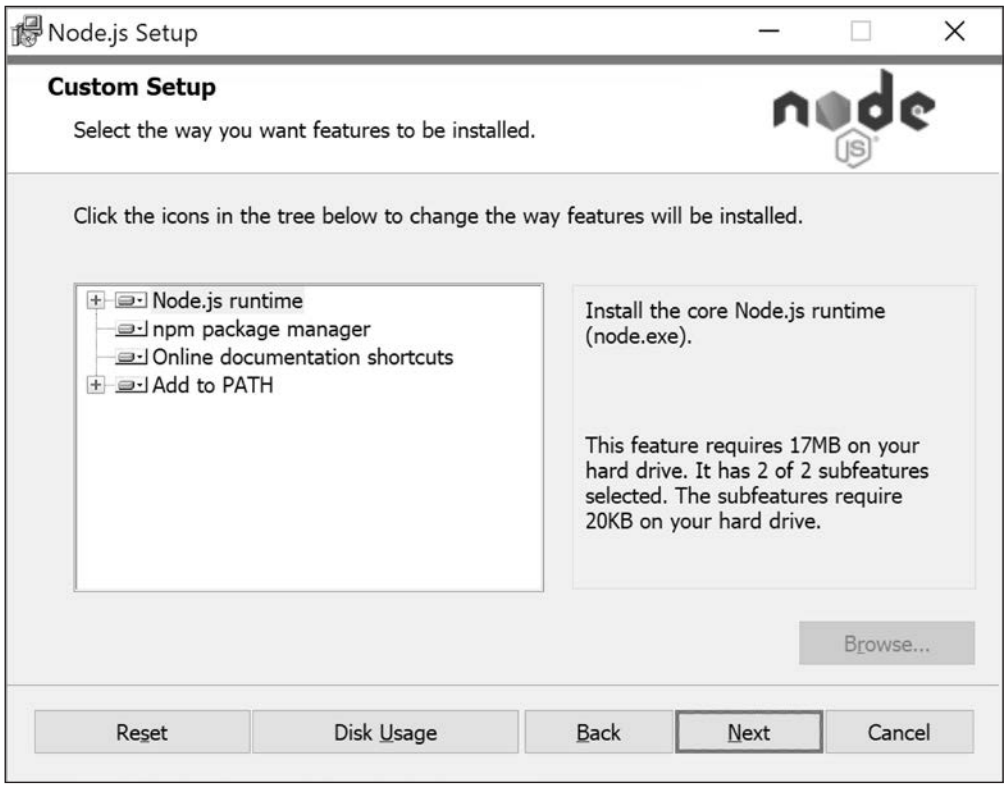


Da biste ubuduće imali brz pristup Visual Studio IDE-u, kliknite desnim tasterom miša na unos u Windows paleti poslova i selektujte opciju Pin this program to taskbar.

U Poglavlju 14, „Izgradnja veb sajtova pomoću ASP.NET Core Razor Pagesa“, Poglavlju 15, „Kreiranje veb sajtova pomoću ASP.NET Core MVC-a“, i Poglavlju 16, „Kreiranje veb servisa i aplikacija pomoću ASP.NET Corea“, potrebno je da imate instalirane Node.js i NPM.

Preuzmite Node.js instaler za Windows sa linka https://nodejs.org/en/download/.

Pokrenite Node.js instaler, kao što je prikazano na sledećoj slici:



Instaliranje Microsoft Visual Studio Codea

Između juna 2015. i septembra 2017. godine „Microsoft“ je izdavao novu verziju Visual Studio Codea skoro svakog meseca. Visual Studio `Code` se brzo poboljšavao i iznenadio svojom popularnošću. Čak i ako planirate da upotrebite Visual Studio 2017 ili Visual Studio for Mac kao glavnu razvojnu alatku, preporučujem da naučite kako da koristite Visual Studio `Code` i .NET Core alatku komandne linije.

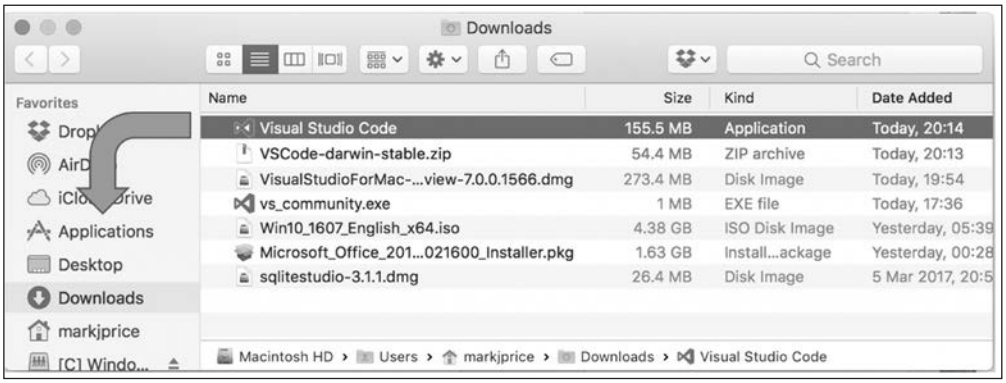
Možete da preuzmete Visual Studio `Code` sa linka https://code.visualstudio.com/.

Možete da pročitate više o „Microsoftovim“ planovima za Visual Studio Code u 2018. godini na adresi https://github.com/Microsoft/vscode/wiki/Roadmap.

Instaliranje Microsoft Visual Studio Codea za macOS

U ovoj knjizi ja ću vam pokazati primere i snimke ekrana Visual Studio Codea, koristeći verziju za macOS. Koraci za izradu tih primera pomoću Visual Studio Codea za Windows i različite distribucije Linuxa veoma su slični, pa neću ponavljati instrukcije za svaku platformu.

Nakon što preuzmete verziju Visual Studio `Code` for macOS, prevucite je i otpustite u direktorijum `Applications`, kao što je prikazano na sledećoj slici:



Sada treba da instalirate .NET Core SDK za macOS. Kompletne instrukcije, uključujući i video snimak, opisani su na sledećoj stranici, a osnovne korake sam uključio u ovu knjigu:

https://www.microsoft.com/net/core#macos

Prvi korak je da instalirate Homebrew (ako ga već niste instalirali).

Pokrenite **Terminal** aplikaciju macOS-a i u odzivnik unesite sledeću komandu:

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Terminal će zatražiti da pritisnete Enter da biste nastavili, a zatim će zatražiti lozinku.

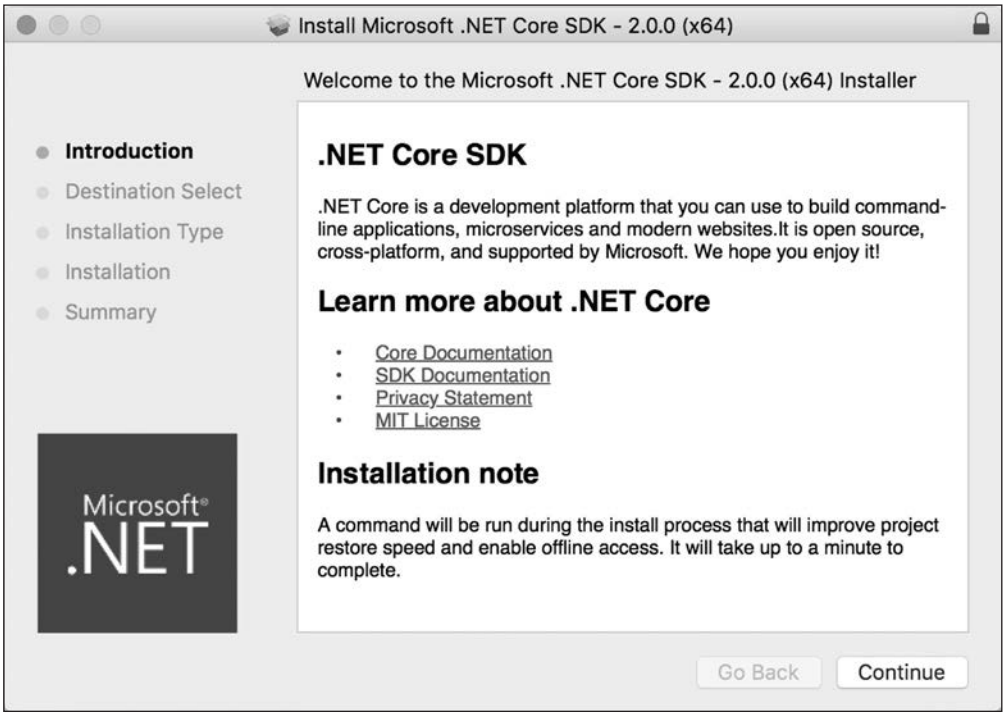
Ako koristite .NET Core 1.0 ili 1.1, sada treba da upotrebite Homebrew za instaliranje OpenSSL-a, koji je potreban za starije verzije .NET Corea na macOS sistemu.

Instaliranje .NET Core SDK-a za macOS

Sledeći korak je da preuzmete .**NET Core SDK** instaler za macOS (x64) sa adrese

https://www.microsoft.com/net/download/core

Pokrenite instalacioni paket `dotnet-sdk-2.0.0-sdk-osx-x64.pkg`, kao što je prikazano na sledećoj slici:



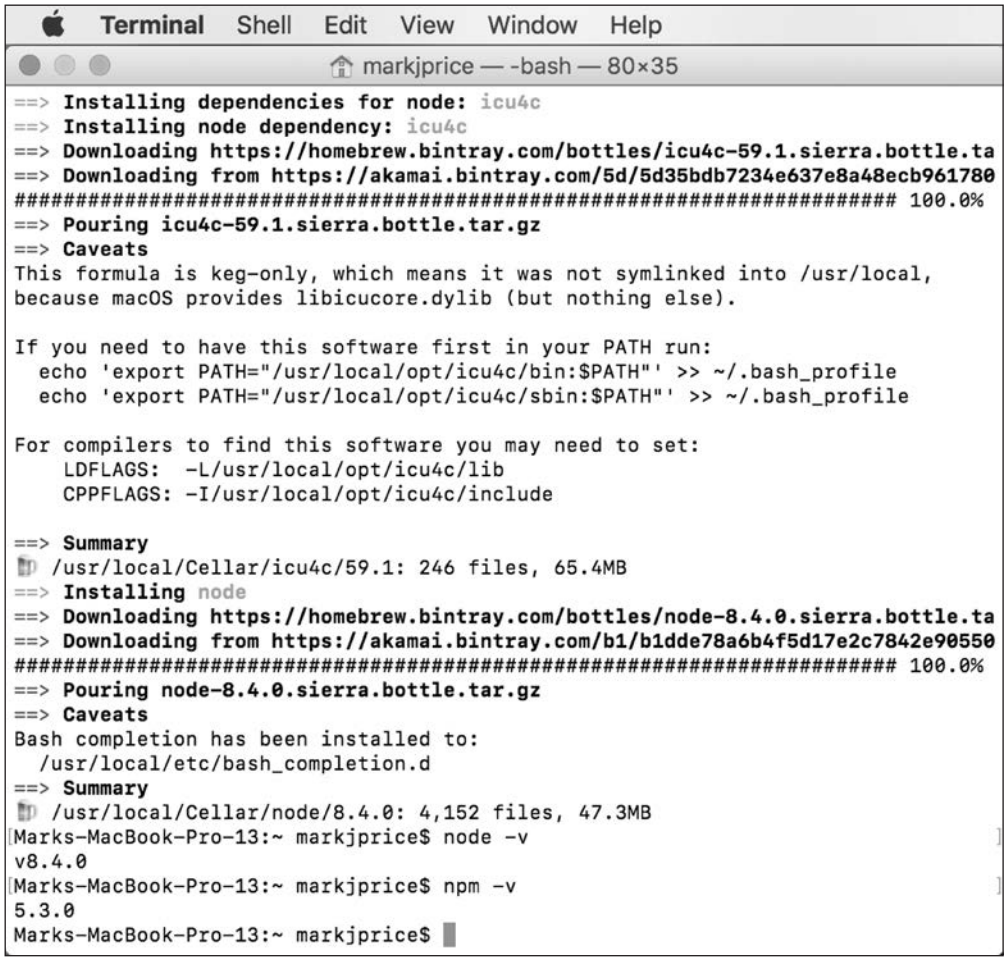
Kliknite na **Continue**, prihvatite ugovor o licenciranju, kliknite na **Install**, a kada je instalacija završena, kliknite na Close.

Instaliranje Node Package Managera za macOS

U Poglavlju 14, „Izgradnja veb sajtova pomoću ASP.NET Core Razor Pagesa“, Poglavlju 15, „Kreiranje veb sajtova pomoću ASP.NET Core MVC-a“ i Poglavlju 16, „Kreiranje veb servisa i aplikacija pomoću ASP.NET Corea“, potrebno je da imate instalirane Node.js i NPM.

U Terminal unesite komande za instaliranje Node.jsa i NPM-a, pa proverite njihove verzije, koje su u vreme pisanja ove knjige bile Node.js verzija 8.4 i NPM verzija 5.3, kao što je prikazano na sledećoj slici:

```
brew install node
node -v
npm -v
```

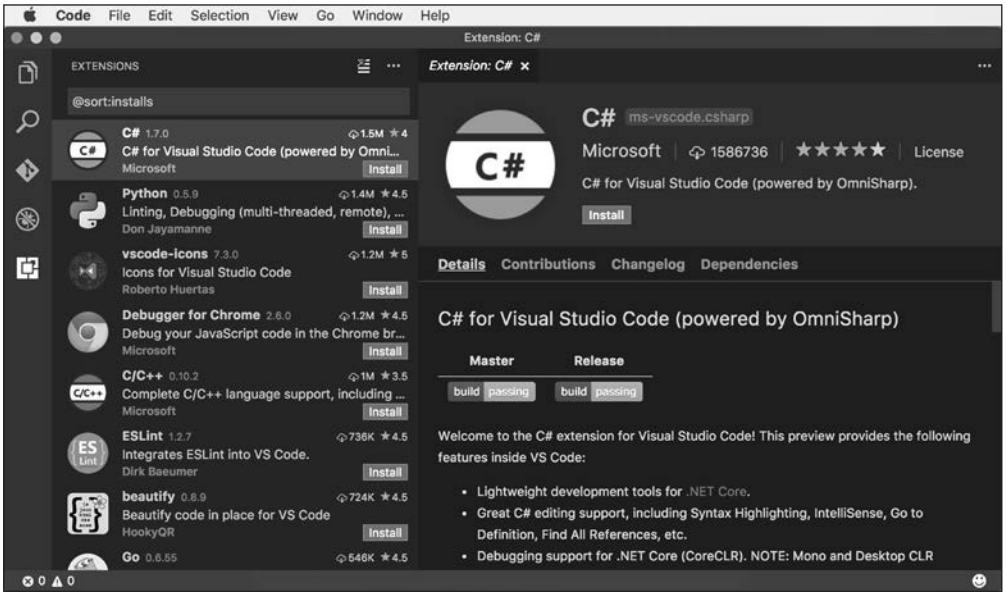


Instaliranje Visual Studio Code ekstenzije za C#

Visual Studio `Code` ekstenzija za C# nije obavezna, ali sadrži IntelliSense dok kucate, pa je veoma korisno da bude instalirana.

Pokrenite **Visual Studio** `Code` i kliknite na ikonicu **Extensions**, ili kliknite na **View** | **Extensions** ili pritisnite *Cmd* + *Shift* + *X*.

C# je najpopularnija ekstenzija, pa treba da je vidite na vrhu liste, kao na sledećoj slici:



Kliknite na **Install**, pa na **Reload**, da biste ponovo učitali prozor i aktivirali ekstenziju.

Instaliranje verzije Visual Studio for Mac

U novembru 2016. godine „Microsoft“ je izdao probnu verziju Visual Studio for Mac. Na početku je ta verzija mogla da se upotrebi samo za kreiranje Xamarin aplikacija za mobilne uređaje, jer je ona deo Xamarin Studio proizvoda. Finalna verzija izdanja, dostupna od maja 2017. godine, ima podršku za kreiranje biblioteka .NET Standard 2.0 klase, ASP.NET Core veb aplikacija i servisa i konzolskih aplikacija; možete da upotrebite ovu verziju za izradu skoro svih vežbi u ovoj knjizi.

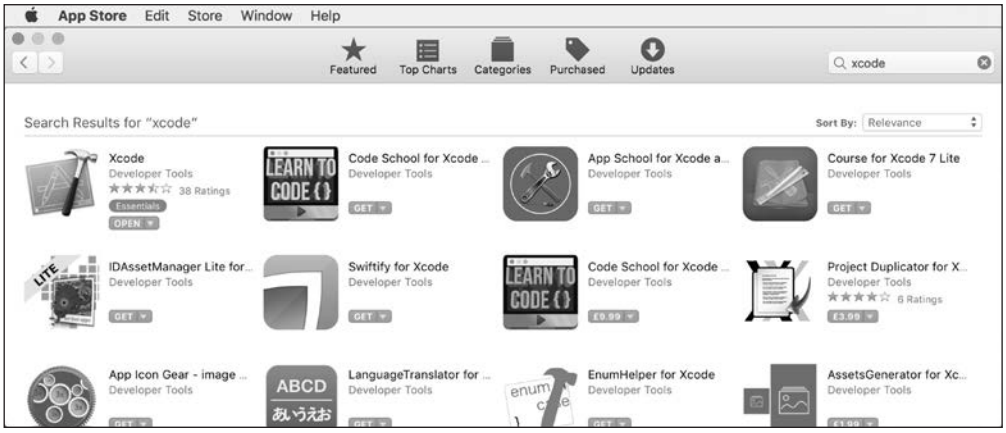
Iako Visual Studio 2017 na Windows sistemu može da se upotrebi za kreiranje aplikacija za mobilne uređaje za iOS i Android, samo Xcode koji je pokrenut na macOS ili OS X sistemu može da kompajlira iOS aplikacije, pa smatram da bi programeri mogli da upotrebe originalnu verziju Visual Studio for Mac za kreiranje aplikacija za mobilne uređaje.

Instaliranje Xcodea

Ako još niste instalirali Xcode na Mac sistem, instalirajte ga sada iz App Storea.

U meniju Apple izaberite **App Store**....

U prozoru **App Storea** unesite xcode u polje za pretragu i jedan od prvih rezultata će biti `Xcode`, kao što je prikazano na sledećoj slici:



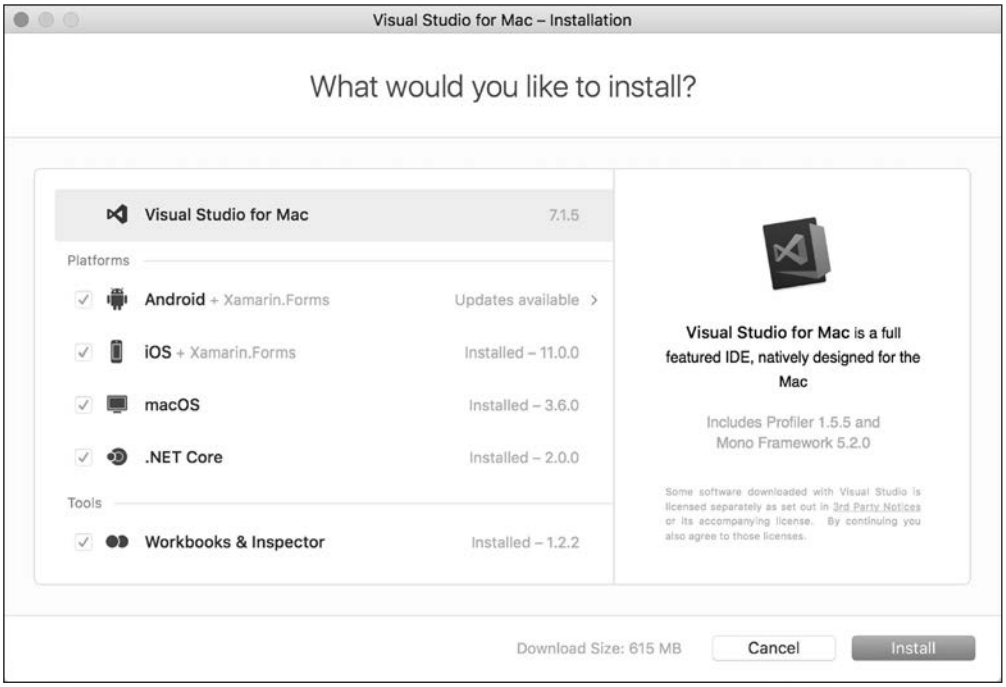
Kliknite na **Get** i sačekajte da se Xcode instalira.

Preuzimanje i instaliranje verzije Visual Studio for Mac

Možete da preuzmete i instalirate Visual Studio for Mac sa linka

https://www.visualstudio.com/vs/visual-studio-mac/.

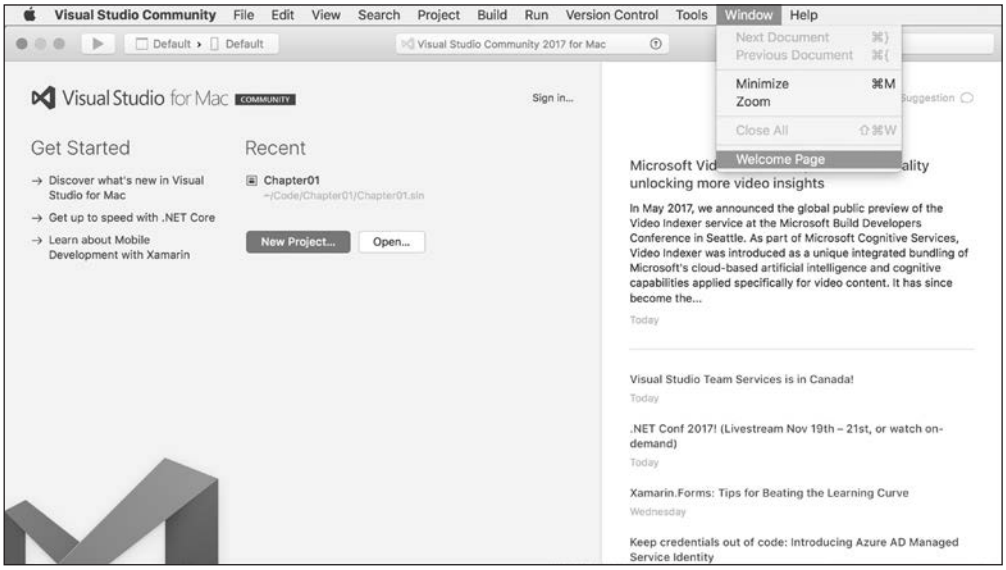
U prozoru **Visual Studio for Mac Instaler** prihvatite License Terms and the Privacy Statement, izaberite da instalirate sve komponente, a zatim kliknite na **Continue**, kao što je prikazano na sledećoj slici:



Kliknite na **Continue**, pa na **Install**.

Potvrdite polje za uslove licence za komponente, kao što je Android SDK, kliknite na **Continue** i sačekajte da se Visual Studio for Mac u potpunosti instalira.

Pokrenite **Visual Studio for Mac** da biste videli stranicu dobrodošlice, kao što je prikazano na sledećoj slici.



Ako se zatraži da ažurirate komponente, kliknite na **Restart and Install Updates**.

Sada, kada ste instalirali i podesili razvojno okruženje, treba da naučite malo više o .NET-u pre nego što počnete da pišete kod.

RAZUMEVANJE .NET-A

.NET Framework, .NET Core, .NET Standard i .NET Native su slične platforme koje programeri koriste za razvoj aplikacija i servisa.

Razumevanje .NET Framework platforme

„Microsoftova“ .NET Framework platforma je razvojna platforma koja uključuje **Common Language Runtime (CLR)** za upravljanje izvršenjem koda i obezbeđuje bogatu biblioteku klasa za izgradnju aplikacija.

„Microsoft“ je dizajnirao .NET Framework tako da može da bude međuplatformski, ali se takođe potrudio da ova platforma najbolje funkcioniše na Windows operativnom sistemu.

Praktično, .NET Framework je platforma samo za Windows operativne sisteme.

Razumevanje Mono i Xamarin projekata

Nezavisni programeri su razvili .NET implementaciju pod nazivom Mono projekat, o kojoj možete da pročitate više na linku

http://www.mono-project.com/.

Mono je međuplatformski projekat, ali se nalazi prilično daleko od zvanične implementacije .NET Frameworka i odlična je osnova za Xamarin mobilnu platformu.

„Microsoft“ je kupio Xamarin 2016.godine i sada nudi ono što je nekada bila skupa Xamarin estenzija besplatno sa Visual Studio 2017 softverom. „Microsoft“ je promenio naziv razvojne alatke Xamarin Studio u Visual Studio for Mac i omogućio kreiranje ASP.NET Core Web API servisa pomoću nje. Xamarin je namenjen za razvoj aplikacija za mobilne uređaje i izgradnju cloud servisa za podršku aplikacijama za mobilne uređaje.

Razumevanje .NET Corea

Mi živimo danas u pravom međuplatformskom svetu. Moderni razvoj mobilnih i cloud platformi je učinio da Windows bude mnogo manje važan operativni sistem. Dakle, „Microsoft“ se potrudio da razdvoji usku vezu .NET-a sa Windows operativnim sistemom.

Dok je menjao .NET, tako da bude stvarno međuplatformski, „Microsoft“ je iskoristio mogućnost da preradi .NET i ukloni glavne delove koji se više ne smatraju „jezgrom“.

Ovaj novi proizvod, nazvan .**NET Core**, uključuje međuplatformsku implementaciju CLR-a, koji je poznatiji kao **CoreCLR**, i modernizovanu biblioteku klasa, koja je poznata kao **CoreFX**.

Scott Hunter, direktor programa za Microsoft partnere za .NET, kaže: „Četrdeset procenata naših .NET Core kupaca su novi programeri za platformu, što upravo i želimo za .NET Core, a to je da uvedemo nove ljude.“

U sledećoj tabeli prikazano je kada su izdate važne verzije .NET Corea i za kada „Microsoft“ planira sledeća glavna izdanja:

VERZIJA	IZDATA
.NET Core RC1	u novembru 2015.
.NET Core 1.0	u junu 2016.
.NET Core 1.1	u novembru 2016.
.NET Core 1.0.4 and .NET Core 1.1.1	u martu 2017.
.NET Core 2.0	u avgustu 2017.
.NET Core for UWP in Windows 10 Fall Creators Update	u oktobru 2017.
.NET Core 2.1	planirana za prvi kvartal 2018.

Ako treba da koristite .NET Core 1.0 i 1.1, preporučujem da pročitate objavu za .NET Core 1.1, mada su informacije na sledećem URL-u korisne za sve .NET Core programere:
https://blogs.msdn.microsoft.com/dotnet/2016/11/16/announcing-net-core-1-1/

.NET Core je mnogo manji od aktuelne verzije .NET Frameworka, jer je mnogo štošta uklonjeno.

Na primer, **Windows Forms i Windows Presentation Foundation (WPF)** mogu da se upotrebe za izgradnju grafičkog korisničkog interfejsa (GUI) aplikacija, ali su usko povezani sa Windowsom, pa su uklonjeni iz .NET Corea. Najnovija tehnologija koja se koristi za izgradnju Windows aplikacija je **Universal Windows Platform (UWP)**, koji je građen na prilagođenoj verziji .NET Corea. Učićete više o tome u Poglavlju 17, „Izgradnja Windows aplikacija pomoću XAML-a i Fluent Designa“.

ASP.NET Web Forms i **Windows Communication Foundation (WCF)** su stare veb aplikacije i tehnologije servisa koje sve manje programera koristi u novim razvojnim projektima, pa su, takođe, uklonjene iz .NET Corea. Umesto njih, programeri koriste **ASP.NET MVC** i **ASP.NET Web API**. Ove dve tehnologije su prerađene i kombinovane u novi proizvod koji se pokreće na .NET Coreu, pod nazivom **ASP.NET Core**. Učićete o ASP.NET Core MVC-u u Poglavlju 15, „Kreiranje veb sajtova pomoću **ASP.NET Core MVC-a“**, o **ASP.NET Core Razor Pagesu** u Poglavlju 14, „Izgradnja veb sajtova pomoću **ASP.NET Core Razor Pagesa“**, i o **ASP.NET Core Web API-ju** i o **Single Page Applicationu (SPA)**, kao što su Angular i React, u Poglavlju 16, „Izgradnja veb servisa i aplikacija pomoću ASP.NET Corea“.

Entity Framework (EF) 6 je tehnologija mapiranja koja se odnosi na objekte za upotrebu podataka koji su sačuvani u relacionim bazama podataka, kao što su Oracle i Microsoft SQL Server. Tokom godina ova tehnologija je „dobila na težini“, pa je međuplatformska verzija skraćena i nosi naziv **Entity Framework Core**. O ovoj verziji ćete naučiti više u Poglavlju 11, „Upotreba baza podataka pomoću Entity Framework Corea“.

Osim što je uklonio velike delove iz .NET Frameworka za kreiranje .NET Corea, „Microsoft“ je upakovao .NET Core u NuGet pakete - male pakete funkcija koji mogu da budu primenjeni nezavisno.

„Microsoftov“ glavni cilj nije da učini .NET Core manjim od .NET Frameworka. Cilj je da sklopi .NET Core za podršku modernim tehnologijama koji ima manje zavisnosti, pa da primena zahteva samo one pakete koji su potrebni za određenu aplikaciju.

Razumevanje .NET Standarda

Postoje tri različite .NET platforme koje kontroliše „Microsoft“:

- .NET Framework
- .NET Core
- Xamarin

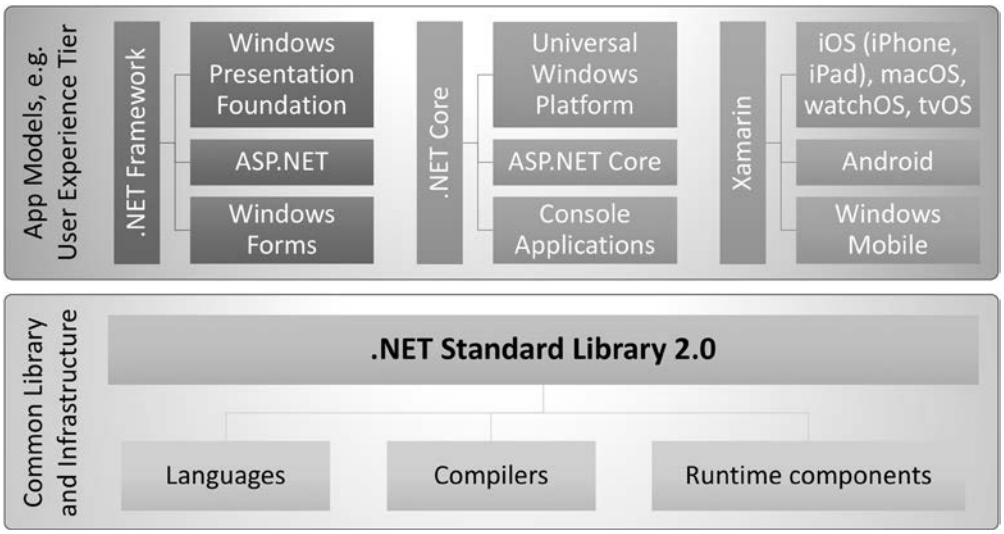
Svaka od ovih platformi ima svoje vrline i slabosti, jer su dizajnirane za različite scenarije. Zbog toga, programeri moraju da nauče tri platforme, od kojih svaka ima neke dosadne „začkoljice“ i ograničenja.

Dakle, „Microsoft“ je definisao .NET Standard 2.0 – to je specifikacija za skup API-ja koju svaka .NET platforma mora da implementira. Ne možete da instalirate .NET Standard 2.0 isto kao što ne možete da instalirate HTML5. Da biste upotrebili HTML5, treba da instalirate veb pretraživač koji implemenitira HTML5 specifikaciju. Da biste upotrebili .NET Standard 2.0, treba da instalirate .NET platformu koja implementira .NET Standard 2.0 specifikaciju.

.NET Standard 2.0 je implementiran najnovijom verzijom .NET Frameworka, .NET Corea i Xamarina. .NET Standard 2.0 olakšava programerima da dele kod između bilo kog tipa .NET-a.

Za .NET Core 2.0 ovo deljenje podrazumeva mnogo nedostajućih API-ja koji su programerima potrebni da bi preneli stari kod napisan za .NET Framework u međuplatformski .NET Core. Međutim, neki API-ji su implementirani, ali podižu izuzetak da bi ukazali programeru da, u stvari, ne bi trebalo da budu upotrebljeni. Razlog pojave ovih izuzetaka je razlika u operativnom sistemu na kojem je pokrenut .NET Core. Naučićete kako se obrađuju ovi izuzeci u Poglavlju 2, „Govoriti C# jezikom".

U sledećem dijagramu rezimirano je kako će tri varijante .NET-a (poznate i kao App Models) deliti zajednički .NET Standard 2.0 i infrastrukturu:



Prvo izdanje ove knjige fokusirano je na .NET Core, ali smo upotrebili .NET Framework kada važne i korisne funkcije još nisu bile implementirane u .NET Core. Visual Studio 2015 je upotrebljen za većinu primera, a Visual Studio `Code` je samo kratko predstavljen.

Drugo izdanje je bilo skoro potpuno očišćeno od svih primera .NET Framework koda.

Treće izdanje je napisano tako da je ceo kod čist .NET Core i može da bude napisan upotrebom verzija Visual Studio 2017, Visual Studio for Mac ili Visual Studio `Code` na bilo kom podržanom operativnom sistemu. Jedini izuzeci su poslednja dva poglavlja. U Poglavlju 17, „Izgradnja Windows aplikacija pomoću XAML-a i Fluent Designa", upotrebićete .NET Core for UWP, koji zahteva da je Visual Studio 2017 pokrenut na sistemu Windows 10, a u Poglavlju 18, „Izgradnja aplikacija za mobilne uređaje pomoću XAML-a i Xamarin. Formsa", upotrebićete Xamarin, umesto .NET Corea.

Razumevanje .NET Native platforme

Još jedna .NET inicijativa je .NET Native, koja kompajlira C# kod u izvorne CPU instrukcije pre vremena (**AoT**), umesto da koristi CLR za kompajliranje koda posrednog jezika (**IL**) tačno na vreme (**JIT**) u izvorni kod.

.NET Native poboljšava brzinu izvršenja i smanjuje upotrebu memorije za aplikacije. Podržava sledeće:

- UWP aplikacije za Windows 10, Windows 10 Mobile, Xbox One, HoloLens i **Internet of Things (IoT)** uređaje, kao što je Raspberry Pi
- veb razvoj na strani servera pomoću ASP.NET Corea
- aplikacije za konzole za upotrebu u komandnoj liniji

Upoređivanje .NET tehnologija

U sledećoj tabeli rezimirane su i upoređene .NET tehnologije:

TEHNOLOGIJA	SKUP FUNKCIJA	KOMPAJLIRA SE U	HOST OPERATIVNI SISTEMI
.NET Framework	iz starih verzija i iz moderne verzije	IL kod	samo Windows
Xamarin	samo za aplikacije za mobilne uređaje	IL kod	iOS, Android, Windows Mo bile
.NET Core	samo za aplikacije za mobilne uređaje	IL kod	Windows, macOS, Linux
.NET Native	samo za aplikacije za mobilne uređaje	izvorni kod	Windows, macOS, Linux

PISANJE I KOMPAJLIRANJE KODA POMOĆU ALATKE .NET CORE CLI

Kada instalirate Visual Studio 2017, Visual Studio for Mac ili .NET Core SDK, instalirani su alatka interfejsa komandne linije (**CLI**) pod nazivom `dotnet` i .NET Core izvršavanje.

Pre nego što upotrebimo CLI alatke, kao što je `dotnet`, potrebno je da napišemo neki kod.

Pisanje koda pomoću jednostavnog editora za tekst

Ako koristite Windows, pokrenite Notepad.

Ako koristite macOS, pokrenite TextEdit. Kliknite na **TextEdit | Preferences**, isključite polje za potvrđivanje **Smart quotes**, a zatim zatvorite okvir za dijalog. Kliknite na **Format | Make Plain Text**.

Alternativno, pokrenite omiljeni editor za tekst.

Unesite sledeći kod:

```
class MyApp { static void Main() {
 System.Console.WriteLine("Hello, C#!"); } }
```

Jezik C# razlikuje velika i mala slova, što znači da morate da kucate karaktere velikih slova i malih slova tačno onako kako je prikazano u prethodnom kodu. C# ne razlikuje razmak, što znači da nije važno da li koristite tabulator, razmak ili znak za početak novog reda da biste rasporedili kod onako kako želite.

Možete da kucate kod u jednu liniju ili da ga proširite u više linija i upotrebite uvlačenje koda. Na primer, sledeći kod će, takođe, biti kompajliran i imaće isti rezultat:

```
class
      MyApp    {
 static                    void
Main          (){System.       Console.
    WriteLine(       "Hello, C#!");         }       }
```

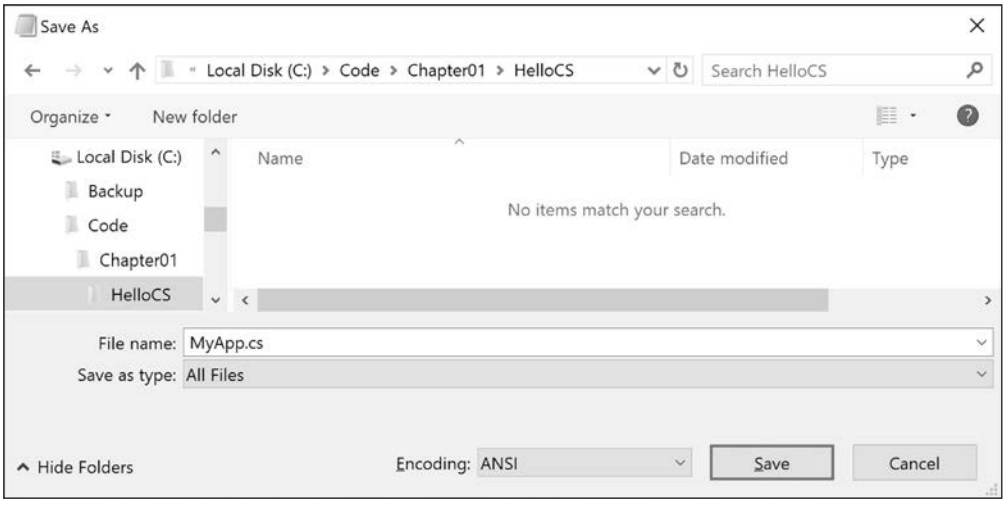
Naravno, najbolje je da napišete kod na isti način kao i drugi programeri da biste mogli mesecima ili godinama kasnije jasno da ga čitate.

Ako koristite Windows Notepad

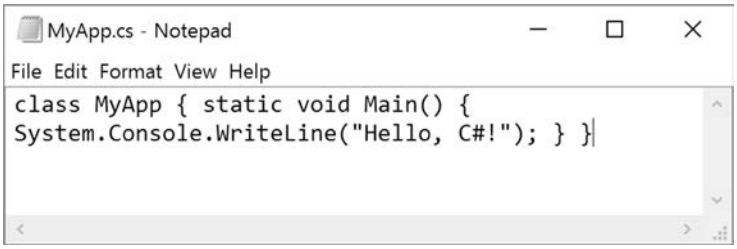
U Notepadu kliknite na **File | Save As**....

U okviru za dijalog **Save As** promenite lokaciju na drajv C: (ili bilo koji drajv koji želite da koristite za snimanje projekata), kliknite na dugme **New folder** i dodelite direktorijumu naziv `Code`. Otvorite direktorijum `Code`, kliknite na dugme **New folder** i direktorijumu dodelite naziv Chapter1. Otvorite direktorijum Chapter01, kliknite na dugme **New folder** i dodelite direktorijumu naziv HelloCS. Otvorite direktorijum HelloCS.

U polju **Save as type** izaberite **All Files** iz padajuće liste da biste izbegli dodavanje ekstenzije `.txt` i unesite naziv fajla `MyApp.cs`, kao što je prikazano na sledećoj slici:



Kod u Notepadu treba da izgleda kao na sledećoj slici:



Ako koristite macOS TextEdit

U prozoru **TextEdita** kliknite na **File | Save**... ili pritisnite *Cmd + S*.

U okviru za dijalog Save promenite lokaciju na direktorijum user (moj se zove `markj-price`) ili bilo koji direktorijum koji želite da upotrebite za snimanje projekata, kliknite na dugme **New Folder** i direktorijumu dodelite naziv `Code`. Otvorite direktorijum `Code`, kliknite na dugme **New Folder** i direktorijumu dodelite naziv Chapter01. Otvorite direktorijum Chapter01, kliknite na dugme **New Folder** i dodelite direktorijumu naziv HelloCS. Otvorite direktorijum HelloCS.

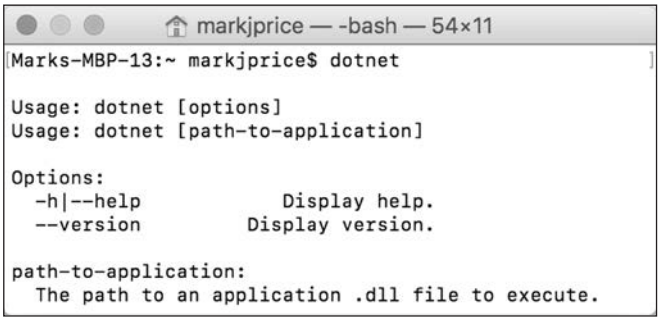
U polju **Plain Text Encoding** selektujte **Unicode (UTF-8)** iz padajuće liste, isključite polje za **If no extension is provided, use „.txt“** da biste izbegli dodavanje ekstenzije fajla `.txt`, unesite naziv fajla `MyApp.cs` i kliknite na **Save**.

Kreiranje i kompajliranje aplikacija pomoću alatke .NET Core CLI

Ako koristite Windows, pokrenite **Command Prompt**.

Ako koristite macOS, pokrenite **Terminal**.

U odzivnik unesite komandu `dotnet` i vidite rezultat, kao što je prikazano na slici za macOS:



Rezultat iz alatke komandne linije dotnet će biti identičan na Windows, macOS i Linux sistemu.

Kreiranje konzolske aplikacija u Command Promptu

Unesite sledeće komande u odzivnik da biste:

- promenili direktorijum za projekat
- kreirali novu konsolsku aplikaciju u direktorijumu
- izlistali fajlove koje je kreirala alatka komandne linije `dotnet`

Ako koristite Windows, u **Command Prompt** unesite sledeći kod:

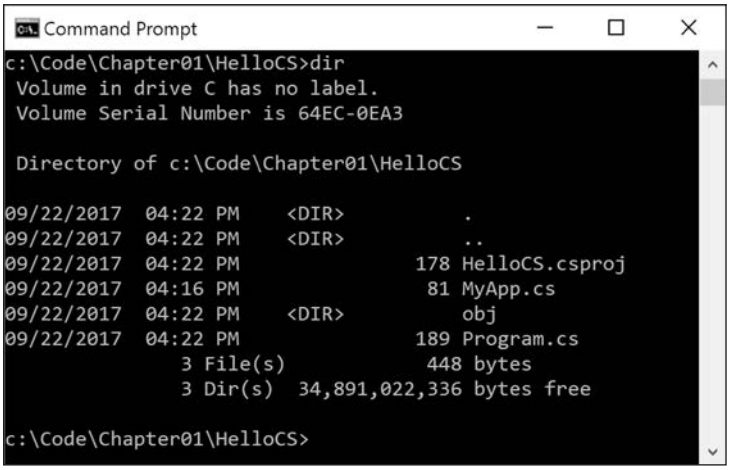
```
cd C:\Code\Chapter01\HelloCS
dotnet new console
dir
```

Ako koristite macOS, u **Terminal** unesite sledeći kod:

```
cd Code/Chapter01/HelloCS
dotnet new console
ls
```

Trebalo bi da vidite da je alatka `dotnet` kreirala dva nova fajla, kao što je prikazano na sledećoj slici na Windowsu:

- `Program.cs` - izvorni kod za jednostavnu konzolsku aplikaciju
- `HelloCS.csproj` - fajl projekta koji lista zavisnosti i konfiguraciju koja se odnosi na projekat



Za ovaj primer treba da izbrišemo fajl pod nazivom `Program.cs`, jer smo već kreirali sopstvenu klasu u fajlu pod nazivom `MyApp.cs`.

Ako koristite Windows, u **Command Prompt** unesite sledeću komandu:

```
del Program.cs
```

Ako koristite macOS, u Terminal unesite sledeću komandu:

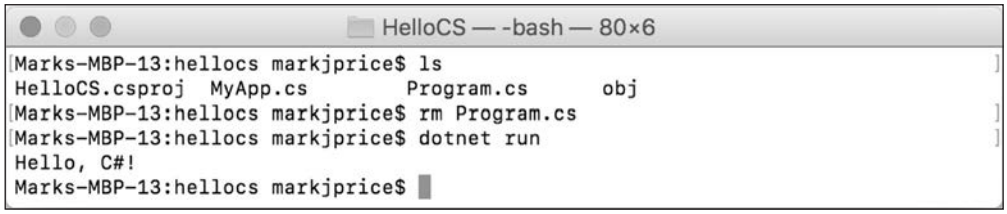
```
rm Program.cs
```

U svim narednim primerima upotrebićemo fajl Program.cs koji je generisala alatka, umesto da ručno kreiramo sopstveni fajl.

Obnavljanje paketa, kompajliranje koda i pokretanje aplikacije

U odzivnik unesite komandu `dotnet run`.

Nakon nekoliko sekundi, svi paketi koji su potrebni za kod će biti preuzeti, izvorni kod će biti kompajliran, a aplikacija pokrenuta, kao što je prikazano na sledećoj slici na macOS sistemu.



Izvorni kod fajl `MyApp.cs` je kompajliran u programski sklop pod nazivom `HelloCS.dll` u poddirektorijumu `bin/Debug/netcoreapp2.0` (potražite u fajl sistemu ovaj poddirektorijum, nakon čega ćemo da nastavimo rad).

Za sada, ovaj programski sklop može da izvrši samo komanda `dotnet run`. U Poglavlju 7, „Razumevanje i pakovanje .NET Standard tipova“, naučićete kako da pakujete kompajlirane programske sklopove za upotrebu na bilo kom operativnom sistemu koji podržava .NET Core.

Ispravljanje grešaka kompajlera

Ako kompajler prikaže greške, pročitajte ih pažljivo i ispravite ih u editoru za tekst. Snimite promene i pokušajte ponovo da kompajlirate kod.

U odzivniku možete da na tastaturi pritisnete taster strelice nagore ili nadole da biste kružili kroz komande koje ste prethodno uneli.

Uobičajena greška može da bude upotreba pogrešne komande, nepostojanje znaka tačka-zarez na kraju linije ili nepodudarnost para velikih zagrada. Na primer, ako ste pogrešno otkucali malo slovo m za metod `Main`, videćete sledeću poruku o grešci:

```
error CS5001: Program does not contain a static 'Main' method
suitable for an entry point
```

Razumevanje posredničkog jezika

C# kompajler (pod nazivom **Roslyn**), koji koristi CLI alatka `dotnet`, konvertuje C# izvorni kod u IL kod i skladišti IL kod u programskom sklopu (DLL ili EXE fajlu).

Iskazi IL koda su kao instrukcije jezika programskog sklopa, ali ih izvršava virtuelna mašina .NET Corea, poznata kao **CoreCLR**.

U vreme izvršenja CoreCLR učitava IL kod iz programskog sklopa, JIT ga kompajlira u izvorne CPU instrukcije, a CPU ga izvršava na mašini.

Prednost ovog procesa kompajliranja, koji se sastoji iz dva koraka, je što „Microsoft“ može da kreira CLR-ove za Linux i macOS kao i za Windows. Isti IL kod se pokreće bilo gde, zbog drugog procesa kompajliranja koji generiše kod za izvorni operativni sistem i skup instrukcija za CPU.

Na primer, bez obzira u kojem jeziku je pisan izvorni kod, u C# ili F# jeziku, sve .NET aplikacije koriste IL kod za instrukcije koje su sačuvane u programskom sklopu. „Microsoft" i drugi proizvođači obezbeđuju disasembler alatke koje mogu da otvore programski sklop i otkriju njegov IL kod.

U stvari, ne koriste sve .NET aplikacije IL kod. Neke koriste .NET Nativeov kompajler za generisanje izvornog koda, umesto IL koda, poboljšavajući, na taj način, performansu i smanjujući upotrebu memorije, ali na štetu prenosivosti.

PISANJE I KOMPAJLIRANJE KODA POMOĆU MICROSOFT VISUAL STUDIOA 2017

Sada ćemo kreirati sličnu aplikaciju pomoću Microsoft Visual Studioa 2017. Ako ste izabrali da koristite Visual Studio for Mac ili Visual Studio `Code`, preporučujem da pregledate ove instrukcije i snimke ekrana, jer Visual Studio for Mac i Visual Studio `Code` imaju slične, ali ne suviše opširne funkcije.

Ja sam više od jedne decenije podučavao studente da koriste Visual Studio i uvek se iznenadim koliko programera ne uspe da iskoristi ovu alatku.

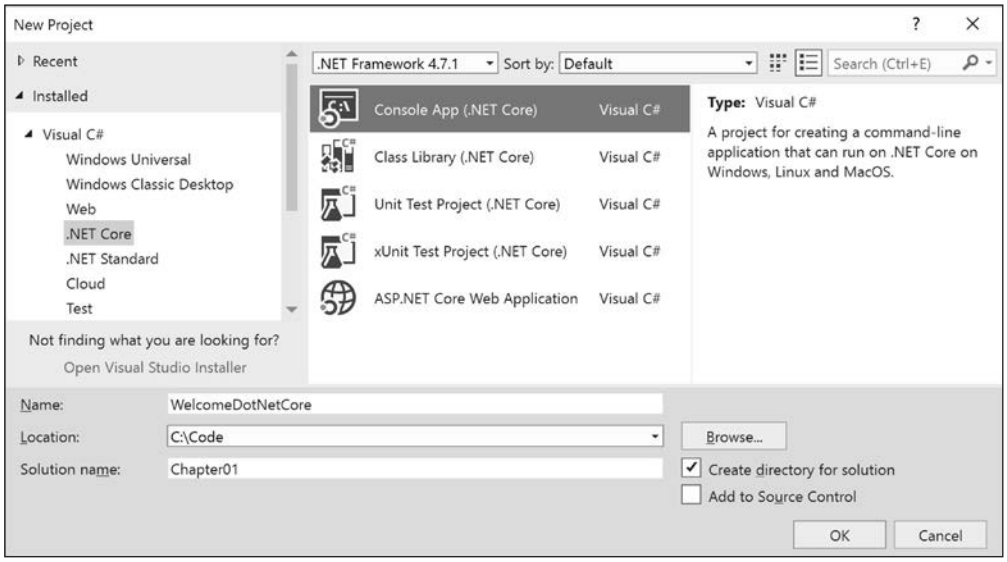
Na sledećih nekoliko stranica ja ću vam objasniti pisanje linije koda. Možda će vam to izgledati suvišno, ali ćete sigurno imati koristi ako vidite kakvu pomoć i informacije Visual Studio pruža dok unosite kod. Ako želite da postanete brz, precizan programer, velika je prednost kada omogućite da Visual Studio ispiše vaš kod umesto vas.

Pisanje koda pomoću Microsoft Visual Studioa 2017

Pokrenite Visual Studio 2017.

Kliknite na **File | New | Project** ili pritisnite prečicu *Ctrl* + *Shift* + *N*.

Iz liste Installed, koja se nalazi na levoj strani prozora, proširite stavku **Visual C#** i izaberite **.NET Core**. Iz liste u centru prozora izaberite opciju `Console` **App (.NET Core)**. Unesite naziv WelcomeDotNetCore, podesite lokaciju na C:\`Code`, unesite Chapter01 kao naziv i kliknite na **OK** ili pritisnite *Enter*, kao što je prikazano na sledećoj slici.

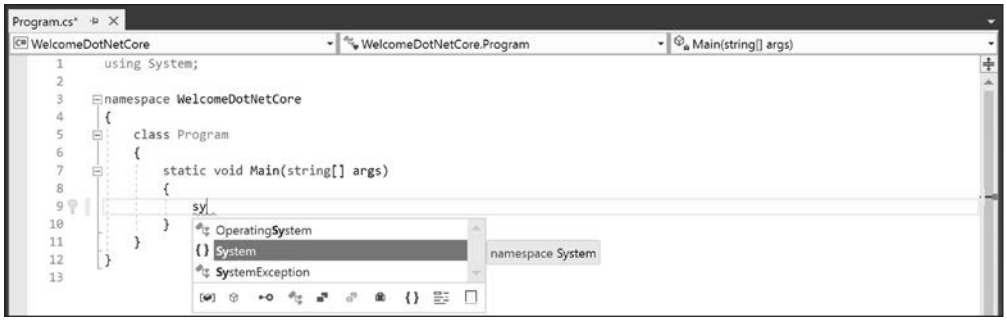


Ignorišite cilj podešen na .NET Framework 4.7.1. Ova padajuća lista ne utiče na .NET Core projekte.

U editoru koda izbrišite iskaz u liniji 9 koja glasi ovako:

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

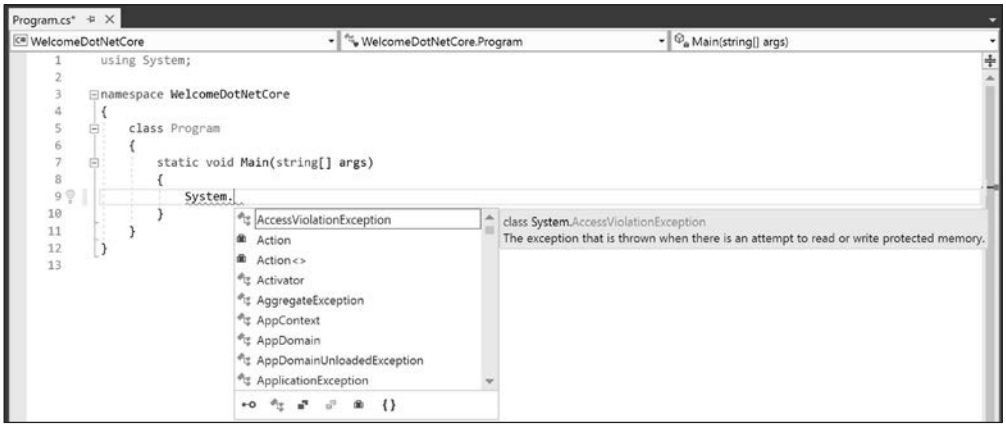
Unutar metoda `Main` ukucajte slova `sy`, kao što je prikazano na sledećoj slici, i videćete da će biti prikazan IntelliSencse meni:



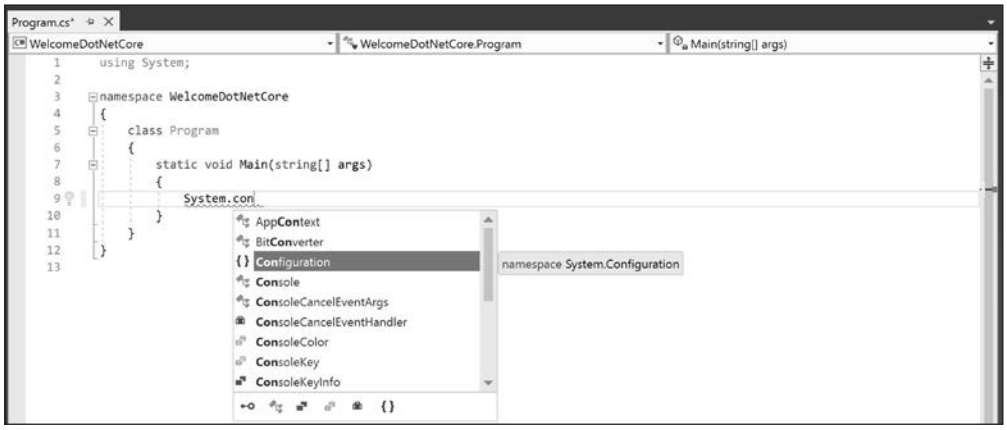
IntelliSense prikazuje filtriranu listu ključnih reči, imenskih prostora i tipova, koja sadrži slova `sy` i ističe stavke koje započinje slovima `sy`, što je, ujedno, i imenski prostor koji mi tražimo - `System`.

Ukucajte tačku (poznata je i kao decimalna tačka).

IntelliSense automatski završava reč `System` umesto vas, unosi tačku i prikazuje listu tipova (na primer, `AggregateException` i `Action`) u imenskom prostoru `System`, kao što je prikazano na sledećoj slici.

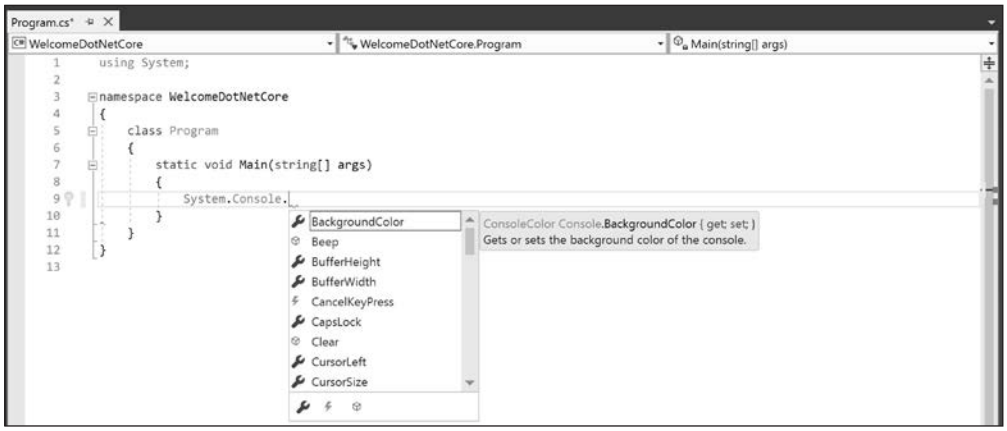


Ukucajte slova `con` i IntelliSense će prikazati listu podudarnih tipova i imenskih prostora, kao što je prikazano na sledećoj slici.



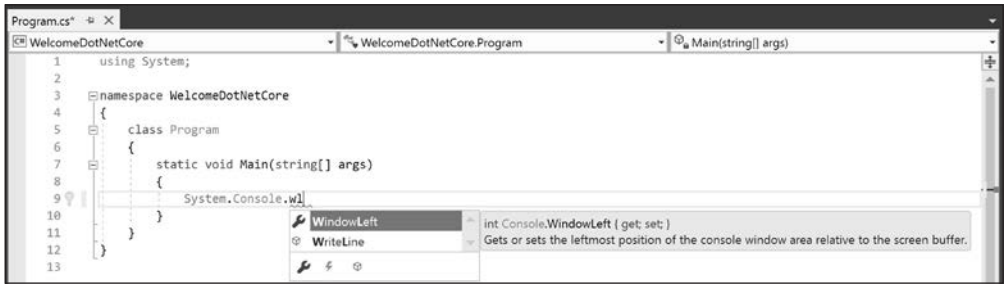
Mi želimo `Console`. Pritisnite taster strelice nadole na tastaturi da biste istakli stavku `Console`. Kada je ta stavka selektovana, ukucajte tačku.

IntelliSense prikazuje listu **članova** klase `Console`, kao što je prikazano na sledećoj slici.



Članovi uključuju svojstva (atribute objekta, kao što je BackgroundColor), metode (akcije koje objekat može da izvrši, kao što je Beep), događaje i drugo.

Ukucajte slova `wl`. IntelliSense prikazuje dva člana koja se podudaraju i sadrže ova slova u naslovu - `WindowLeft` i `WriteLine`, kao što je prikazano na sledećoj slici.



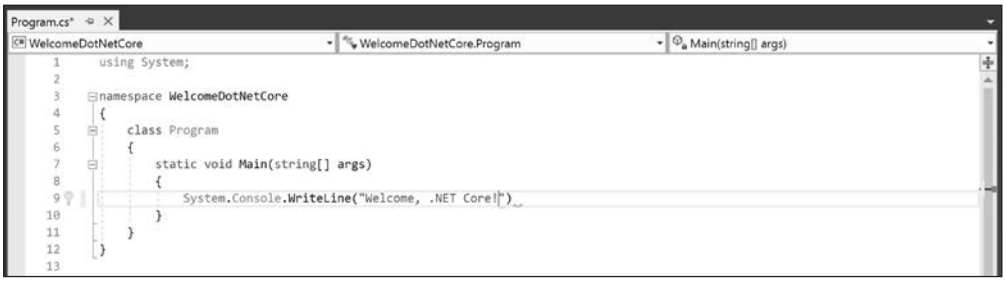
Upotrebite taster strelice nadole da biste selektovali `WriteLine`, a zatim ukucajte otvorenu zagradu (.

IntelliSense automatski popunjava `WriteLine` i unosi par zagrada.

Takođe ćete videti opis alatke koji ukazuje da metod `WriteLine` ima 18 varijacija, kao što je prikazano na sledećoj slici.

Ukucajte dvostruki navodnik (`"`). IntelliSense unosi par dvostrukih navodnika umesto vas i ostavlja kursor između njih.

Ukucajte tekst `Welcome, .NET Core!`, kao što je prikazano na sledećoj slici.



Crvena škrabotina na kraju linije ukazuje na grešku, zato što svaki C# iskaz treba da se završi znakom tačka-zarez. Pomerite kursor na kraj linije i ukucajte taj znak da biste ispravili grešku.

Kompajliranje koda pomoću alatke Visual Studio 2017

Kliknite na **Debug | Start Without Debugging** ili pritisnite tastere *Ctrl + F5*.

Statusna linija Visual Studioa ispisuje **Build started**..., pa **Build succeeded**, a zatim će konzolska aplikacija biti pokrenuta u prozoru Command Prompta, kao što je prikazano na sledećoj slici.

Da bih uštedeo prostor u ovoj knjizi i rezultat učinio jasnijim, neću uključiti snimke ekrana rezultata iz konzolske aplikacije kao što sam uradio na prethodnoj slici. Umesto toga, prikazaću rezultat na sledeći način:

```
Welcome, .NET Core!
```

Ispravljanje grešaka pomoću liste grešaka

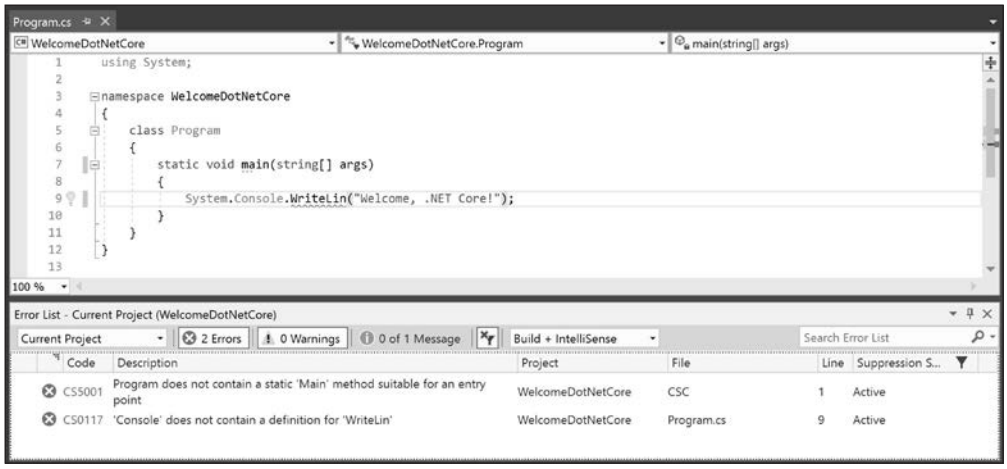
Hajde da namerno napravimo dve greške.

- Promenite slovo `M` u metodu `Main` na malo slovo `m`.
- Izbrišite slovo `e` na kraju naziva metoda `WriteLine`.

Kliknite na **Debug | Start Without Debugging** ili pritisnite tastere *Ctrl + F5*.

Nakon nekoliko sekundi, statusna linija će ispisati **Build failed** i biće prikazana poruka o grešci. Kliknite na **No**.

Error List postaje aktivan, kao što je prikazano na sledećoj slici.

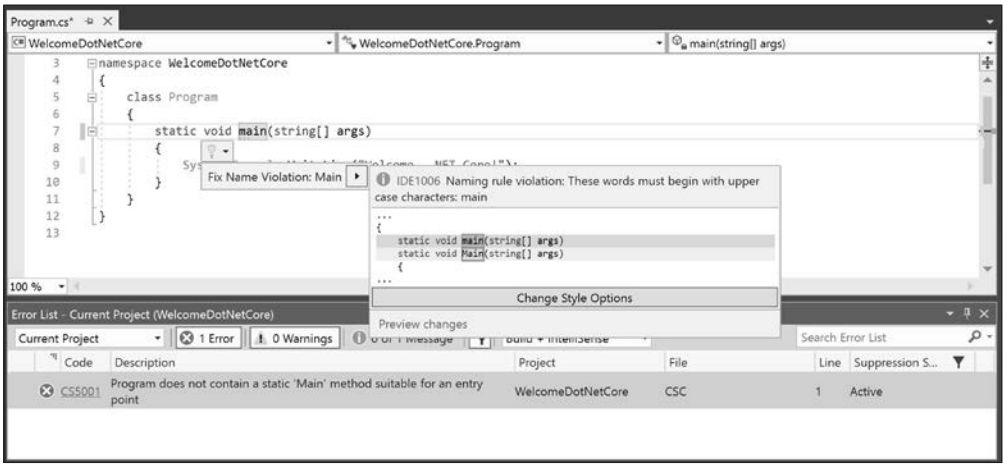


Lista grešaka može da se filtrira da prikazuje greške (**Errors**), upozorenja (**Warnings**) i poruke (**Messages**), tako što ćete kliknuti na dugmad u liniji sa alatima na vrhu prozora **Error List**.

Ako greška prikaže fajl i broj linije - na primer, **File: Program.cs** i **Line: 9**, možete dvostruko da kliknete na grešku da bi kursor preskočio do linije koja izaziva problem.

Ako je greška uobičajena, kao što je nedostatak metoda `Main`, kompajler neće moći da prikaže broj linije. Možda ćete želeti metod pod nazivom main i metod pod nazivom `Main` (ne zaboravite da jezik C# razlikuje velika i mala slova).

Međutim, Visual Studio takođe može da analizira kod i istakne predloge pomoću tri male sive tačke ispod potencijalnog problema. Kada kliknete na iskaz označen tačkama, biće prikazana „sijalica"; ako kliknete na nju, biće prikazani predlozi za poboljšanja - na primer, da nazivi metoda započinju karakterom velikog slova, kao što je prikazano na sledećoj slici.



Ispravite ove dve greške i ponovo pokrenite aplikaciju da biste, pre nego što nastavite rad, bili sigurni da ona funkcioniše. Vidite da će prozor Error List biti ažuriran i da ne prikazuje greške.

Dodavanje postojećih projekata u Visual Studio 2017

Ranije ste kreirali projekat pomoću `dotnet` CLI alatke. Sada, kada imate rešenje u Visual Studiou 2017, možda ćete želeti da dodate prethodni projekat.

Kliknite na **File | Add | Existing Project**..., pronađite direktorijum C:\`Code`\Chapter01\HelloCS i selektujte fajl `HelloCS.csproj`.

Da biste mogli da pokrenete ovaj projekat, u **Solution Exploreru** kliknite desnim tasterom miša na **Solution ‚Chapter01' (2 projects)** i iz kontekstnog menija izaberite **Properties** ili pritisnite *Alt* + *Enter*.

Za opciju **Startup Project** kliknite na **Current** selection, a zatim na **OK**.

U prozoru **Solution Explorer** kliknite na fajl unutar projekta **HelloCS**, pa pritisnite *Ctrl + F5*, ili kliknite na **Debug | Start Without Debugging**.

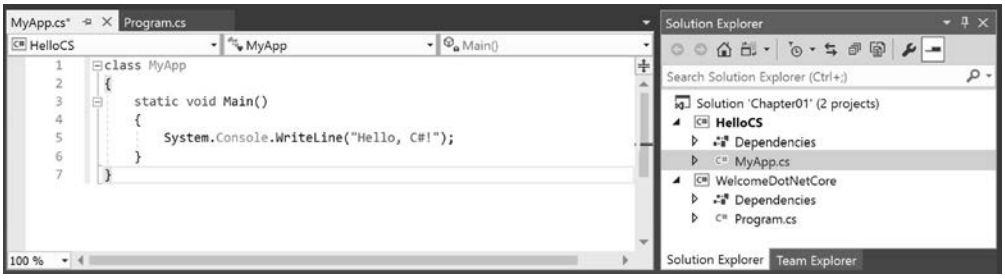
Automatsko formatiranje koda

Kod je mnogo jednostavniji za čitanje i razumevanje ako je dosledno uvučen i razmaknut.

Ako kod može da se kompajlira, Visual Studio 2017 može automatski da ga formatira da bude lepo razmaknut i uvučen.

U prozoru **Solution Explorer** dvostruko kliknite na fajl pod nazivom `MyApp.cs`, kao što je prikazano na sledećoj slici.

Kliknite na **Build | Build HelloCS** ili pritisnite *Shift + F6*, sačekajte da se kod izgradi, a zatim kliknite na **Edit | Advanced | Format Document**, ili pritisnite *Ctrl + E, D*. Kod će biti automatski formatiran, kao što je prikazano na sledećoj slici.



Eksperimentisanje sa C# Interactiveom

Iako je Visual Studio uvek imao Immediate prozor sa ograničenom podrškom **Read-eval-print loop (REPL)**, Visual Studio 2017 uključuje poboljšan prozor koji sadrži ceo IntelliSense i kolorni kod sintakse **C# Interactive**.

U prozoru Visual Studio 2017 kliknite na **View | Other Windows | C# Interactive**.

Napisaćemo interaktivni kod za preuzimanje stranice About sa javnog web sajta „Microsofta“.

Ovo je samo primer - ne treba još da razumete kod.

U komandnu liniju **C# Interactive** unećemo komande za izvršavanje:

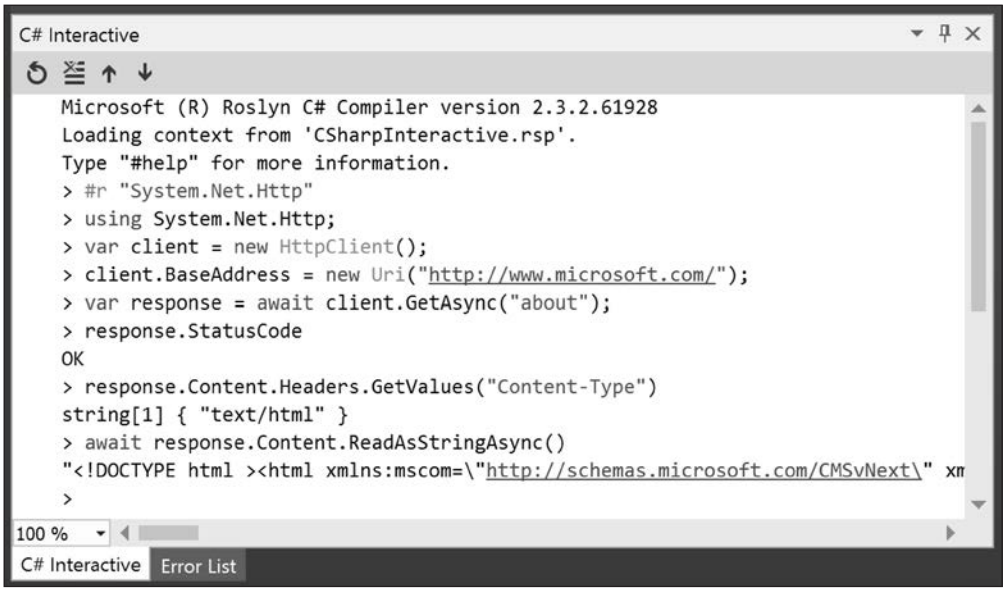
- referenciranja `System.Net.Http` programskog sklopa
- importovanja `System.Net.Http` imenskog prostora
- deklarisanja i instanciranja promenljive HTML klijenta
- podešavanja glavne adrese klijenta na Microsoft web sajtu
- asinhronog čekanja odgovora za GET zahtev za stranicu About
- čitanja koda statusa vraćenog sa veb servera
- čitanja tipa sadržaja zaglavlja
- čitanja sadržaja HTML stranice kao znakovnog niza

Ukucajte svaku od sledećih komandi iza > odzivnika, a zatim pritisnite Enter:

```
    > #r "System.Net.Http"
> using System.Net.Http;
> var client = new HttpClient();
> client.BaseAddress = new Uri("http://www.microsoft.com/");
> var response = await client.GetAsync("about");
> response.StatusCode
OK
> response.Content.Headers.GetValues("Content-Type")
string[1] { "text/html" }
> await response.Content.ReadAsStringAsync()
"<!DOCTYPE html ><html
xmlns:mscom="http://schemas.microsoft.com/CMSvNext"
xmlns:md="http://schemas.microsoft.com/mscom-data" lang="en"
xmlns="http://www.w3.org/1999/xhtml"><head><meta http-equiv="X-UA-
Compatible" content="IE=edge" /><meta charset="utf-8" /><meta
    name="viewport" content="width=device-width, initial-scale=1.0"
```

```
/><link rel="shortcut icon"
href="//www.microsoft.com/favicon.ico?v2" /><script
type="text/javascript"
src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-
1.7.2.min.js">rn // Third party scripts and code linked to
or referenced from this website are licensed to you by the parties
that own such code, not by Microsoft. See ASP.NET Ajax CDN Terms of
Use - http://www.asp.net/ajaxlibrary/CDN.ashx.rn
</script><script type="text/javascript"
language="javascript">/*<![CDATA[*/if($(document).bind("mobileinit
",function(){$.mobile.autoInitializePage=!1}),navigator.userAgent.ma
tch(/IEMobile\/10\.0/)){var
msViewportStyle=document.createElement("style ...
```

Na sledećoj slici prikazano je kako Visual Studio 2017 treba da izgleda nakon što unesete prethodne komande u prozor **C#Interactive**.



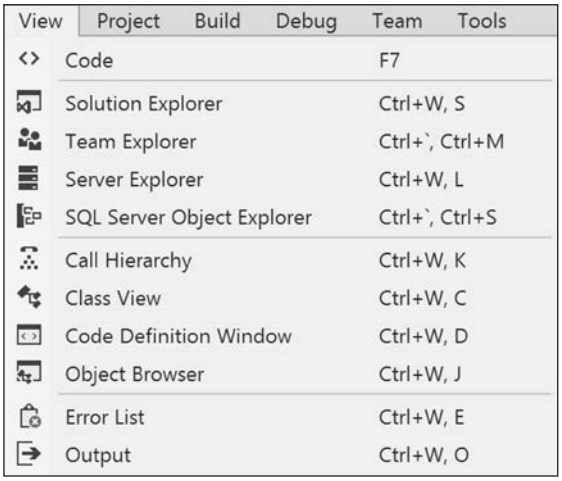
Roslyn je naziv C# kompajlera. Roslyn verzija 1.0 je bila za C# 6, Roslyn verzija 2.0 za C# 7, a Roslyn 2.3 i novije verzije su za C# 7.1.

Ostali korisni prozori

Visual Studio 2017 ima mnoštvo drugih korisnih prozora, uključujući sledeće:

- prozor **Solution Explorer** za upravljanje projektima i fajlovima
- prozor **Team Explorer** za alatke za upravljanje izvornim kodom
- prozor **Server Explorer** za upravljanje konekcijama sa bazom podataka i resursima za upravljanje u Microsoft Azureu.

Ako ne možete da vidite prozor koji vam je potreban, kliknite na meni **View** da bi prozor bio ponovo prikazan ili pritisnite prečicu na tastaturi za prozor koji želite. Neke od ovih prečica su prikazane na sledećoj slici.



Ako se vaše prečice na tastaruri razlikuju od onih koje su prikazane na prethodnoj slici, razlog je činjenica da ste izabrali drugi skup kada ste instalirali Visual Studio. Možete da resetujete prečice na tastaturi da se podudaraju sa onima koje ćemo upotrebiti u ovoj knjizi - kliknite na meni Tools, pa na Import and Export Settings…, izaberite Reset all settings, a zatim izaberite resetovanje Visual C# kolekciju podešavanja.

PISANJE I KOMPAJLIRANJE KODA POMOĆU VISUAL STUDIO CODEA

Instrukcije i snimci ekrana u ovom odeljku su za macOS, ali će iste akcije funkcionisati i za Visual Studio `Code` na Windows ili Linux sistemu. Osnovne razlike će biti izvorne akcije komandne linije, kao što je brisanje fajla - i komanda i putanja će, verovatno, biti drugačije. CLI alatka `dotnet` će biti identična na svim platformama.

Pisanje koda pomoću Visual Studio Codea

Pokrenite Visual Studio `Code`.

Kliknite na **File | Open**... ili pritisnite *Cmd + O*.

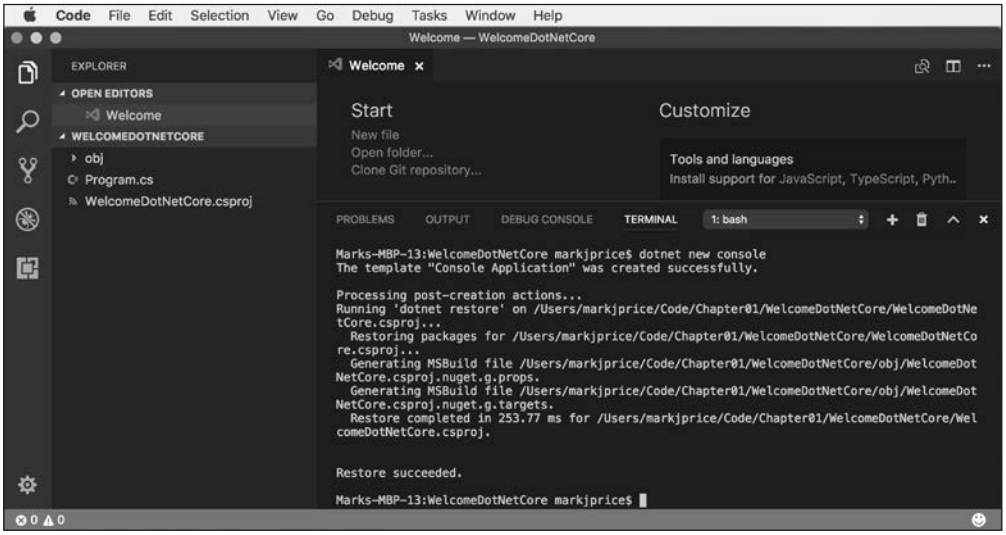
U okviru za dijalog otvorite direktorijum `Code`, selektujte direktorijum Chapter01, kliknite na dugme New Folder, unesite naziv `WelcomeDotNetCore` i kliknite na **Create**. Selektujte direktorijum **WelcomeDotNetCore** i kliknite na **Open** ili pritisnite *Enter*.

U Visual Studio Codeu kliknite na **View | Integrated Termina**l ili pritisnite *Ctrl + `*.

U Terminal unesite sledeću komandu:

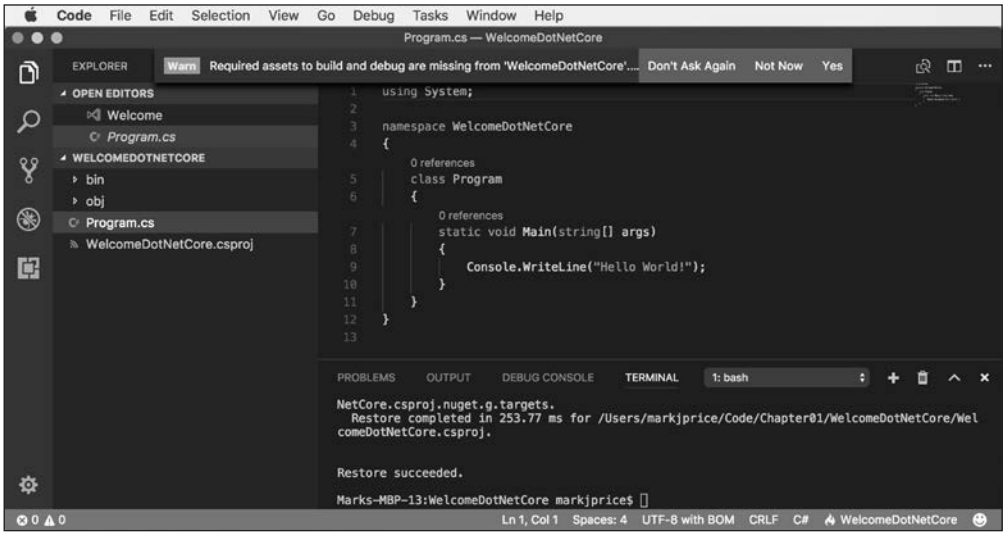
```
dotnet new console
```

Videćete da alatka komandne linije `dotnet` kreira novi projekat konzolske aplikacije u aktuelnom direktorijumu, a prozor **Explorer** prikazuje dva fajla koja su kreirana, kao što je prikazano na sledećoj slici.



U prozoru **EXPLORER** kliknite na fajl pod nazivom `Program.cs` da biste ga otvorili u prozoru editora.

Ako vidite upozorenje da nedostaju potrebni elementi, kliknite na **Yes**, kao što je prikazano na sledećoj slici.



Modifikujte tekst koji je napisan u konzoli, tako da glasi: `Welcome, .NET Core!`

Kliknite na **File | Auto Save**. Ovo podešavanje će vam pokazati da ne treba svaki put da snimite fajl pre nego što ponovo izgradite aplikaciju.

Kompajliranje koda pomoću Visual Studio Codea

Kliknite na **View | Integrated Terminal** ili pritisnite *Ctrl* + `` ` `` i unesite sledeću komandu:

```
dotnet run
```

Rezultat u prozoru **Terminal** će prikazati rezultat pokrenute aplikacije.

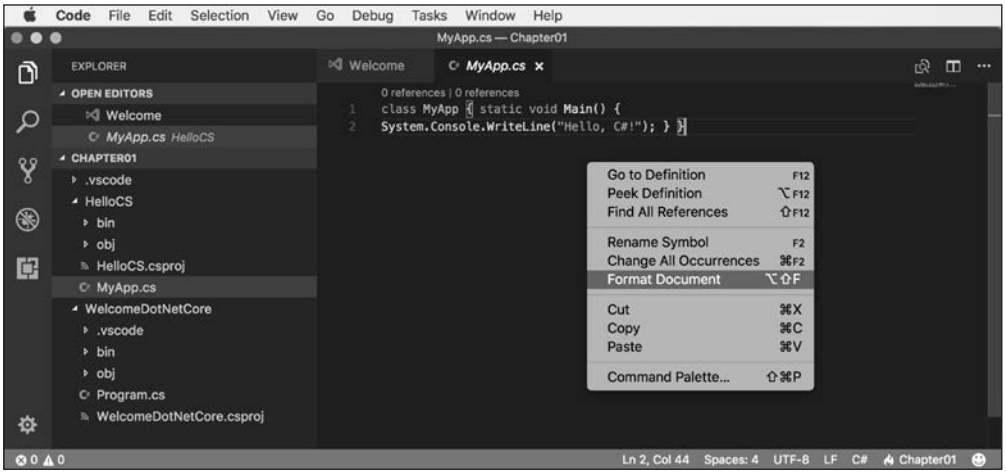
Automatsko formatiranje koda

U Visual Studio Codeu kliknite na **File | Open** i otvorite direktorijum Chapter01.

U Exploreru proširite **HelloCS** i selektujte **MyApp.cs.**

Kliknite na **Yes** kada se zatraži da dodate potrebne elemente.

U Visual Studio Codeu kliknite desnim tasterom miša i izaberite opciju **Format Document** ili pritisnite *Alt* + *Shift* + *F*, kao što je prikazano na sledećoj slici.



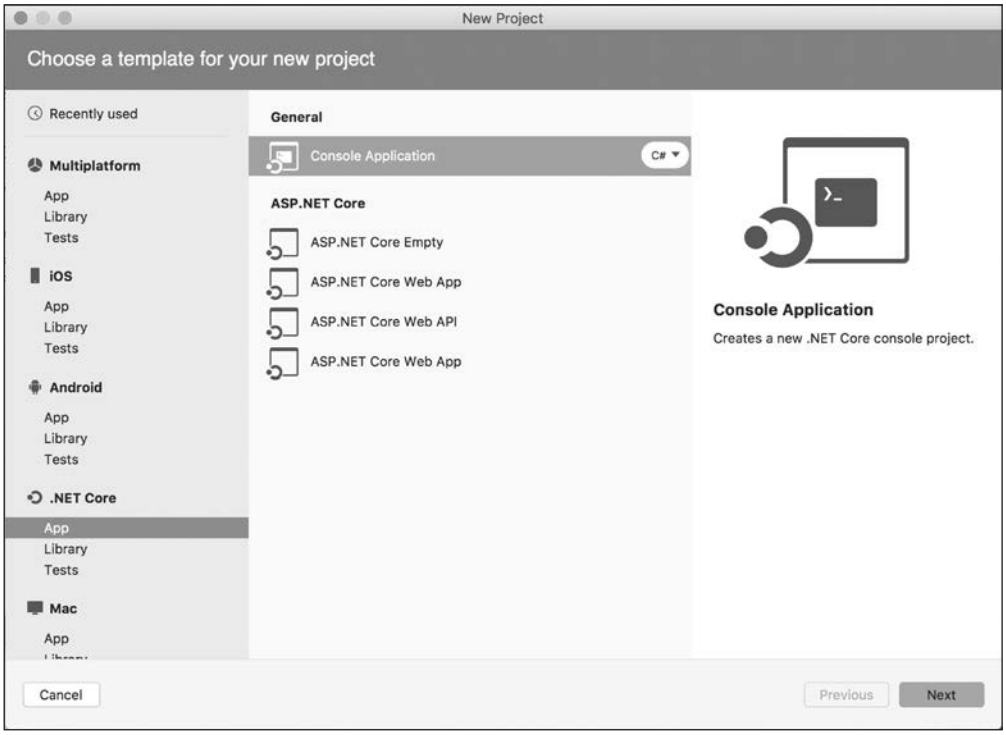
Visual Studio Code brzo se približava funkcijama Visual Studio 2017 na Windows sistemu.

PISANJE I KOMPAJLIRANJE KODA POMOĆU IDE-A VISUAL STUDIO FOR MAC

Pokrenite Visual Studio for Mac i kliknite na **File | New Solution**.

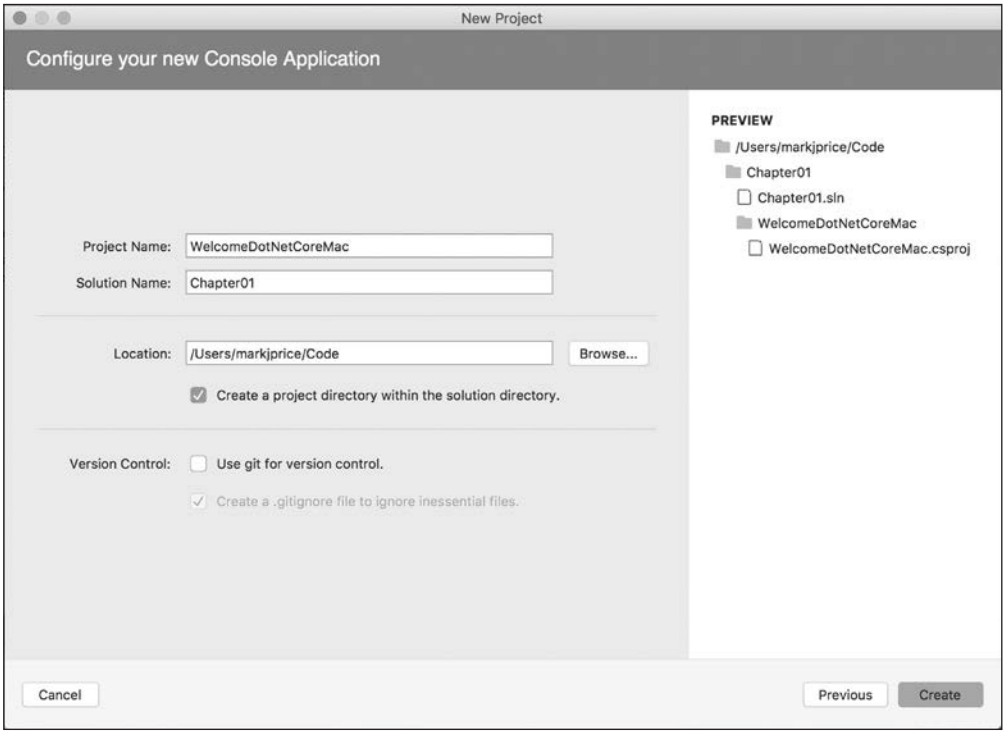
U listi sa leve strane prozora u odeljku .**NET Core** selektujte **App**.

Iz liste šablona projekta u sredini prozora selektujte `Console Application`, pa kliknite na **Next**, kao što je prikazano na sledećoj slici.



U koraku Configure your new **Console Application** selektujte **Target Framework** za **.NET Core 2.0** i kliknite na **Next**.

U koraku **Configure your new Console Application** za **Project Name** unesite `Welcome-DotNetCoreMac`, za **Solution Name** unesite Chapter01, podesite **Location** na direktorijum `Code` i kliknite na **Create**, kao što je prikazano na sledećoj slici.

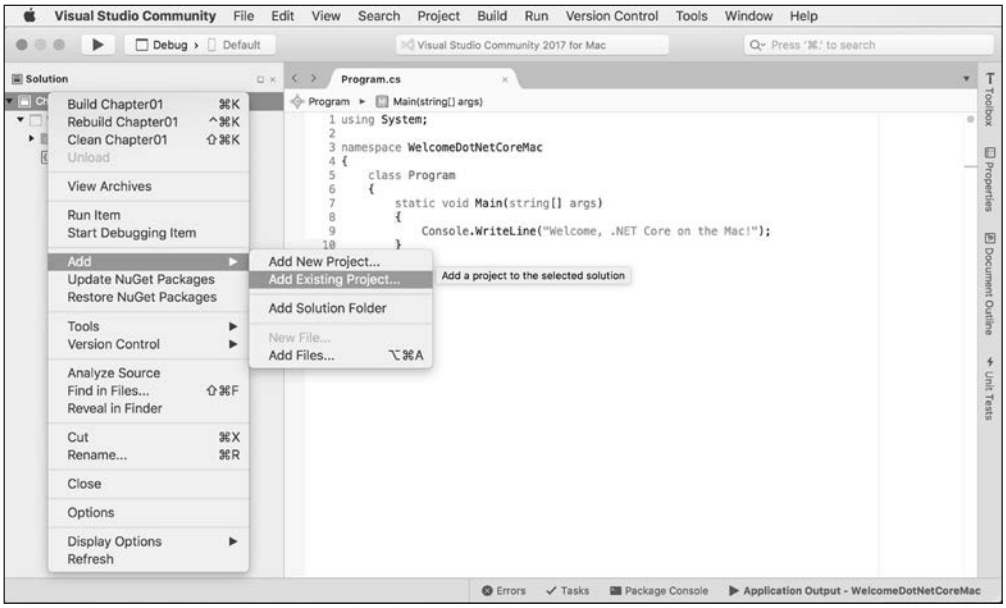


Modifikujte tekst koji je napisan u konzoli da glasi: `Welcome, .NET Core on the Mac!`

U prozoru Visual Studio for Mac kliknite na `Run | Start Without Debugging` ili pritisnite *Cmd* + *Option* + *Enter*.

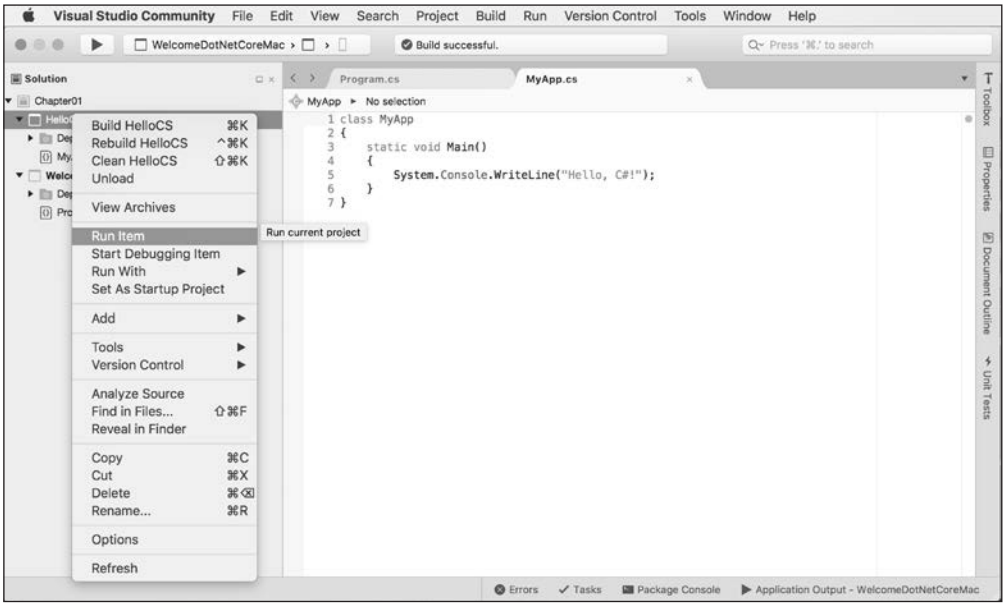
Rezultat u **Terminalu** će prikazati rezultat pokenute aplikacije.

U odeljku Solution kliknite desnim tasterom miša na Chapter01 i kliknite na **Add | Add Existing Project...**.



U direktorijumu **HelloCS** selektujte fajl `HelloCS.cs.proj`.

U odeljku **Solution** kliknite desnim tasterom miša na **HelloCS** i selektujte **Run Item**, kao što je prikazano na sledećoj slici.



Sledeći koraci

Znate kako da kreirate i izgradite jednostavne .NET Core aplikacije za Windows i macOS (isto tako je jednostavno i za Linux).

Moći ćete da izradite skoro sve vežbe u svim poglavljima u ovoj knjizi, koristeći Visual Studio 2017 na Windowsu, Visual Studio for Mac ili Visual Studio `Code` na Windows, macOS ili Linux sistemu.

UPRAVLJANJE IZVORNIM KODOM POMOĆU GITHUBA

Git je sistem za upravljanje izvornim kodom koji se najčešće koristi. GitHub održava veb sajt i desktop aplikaciju i olakšava upravljanje Gitom.

Ja sam upotrebio GitHub za čuvanje rešenja za sve praktične vežbe koje se nalaze na kraju svakog poglavlja na sledećem URL-u:

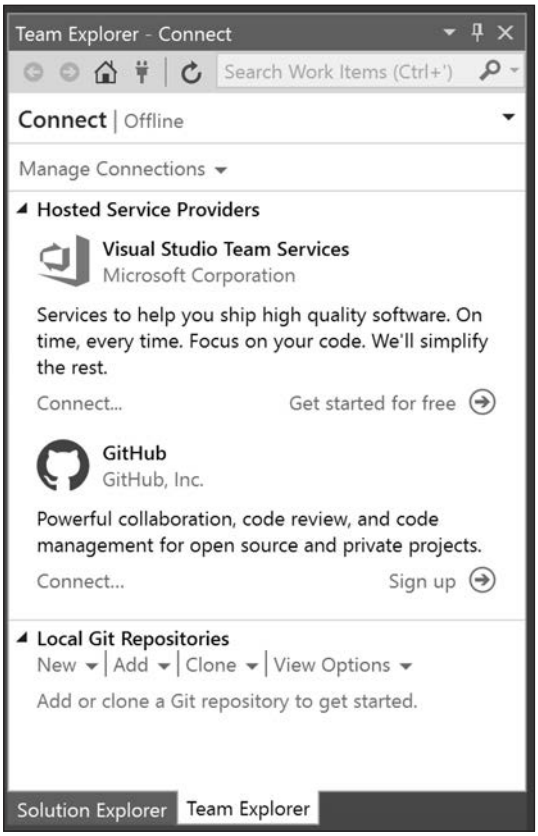
https://github.com/markjprice/cs7dotnetcore2

Upotreba Gita sa IDE-om Visual Studio 2017

Visual Studio 2017 ima ugrađenu podršku za upotrebu Gita sa GitHubom kao i „Microsoftov“ sistem za upravljanje izvornim kodom, pod nazivom **Visual Studio Team Services (VSTS)**.

Upotreba prozora Team Explorer

U prozoru Visual Studio 2017 kliknite na **View | Team Explorer**, kao što je prikazano na sledećoj slici.



Iako bi bilo dobro da se prijavite provajderu za online sistem za upravljanje izvornim kodom, možete da klonirate GitHub skladište bez prijavljivanja i otvaranja naloga.

Kloniranje GitHub skladišta

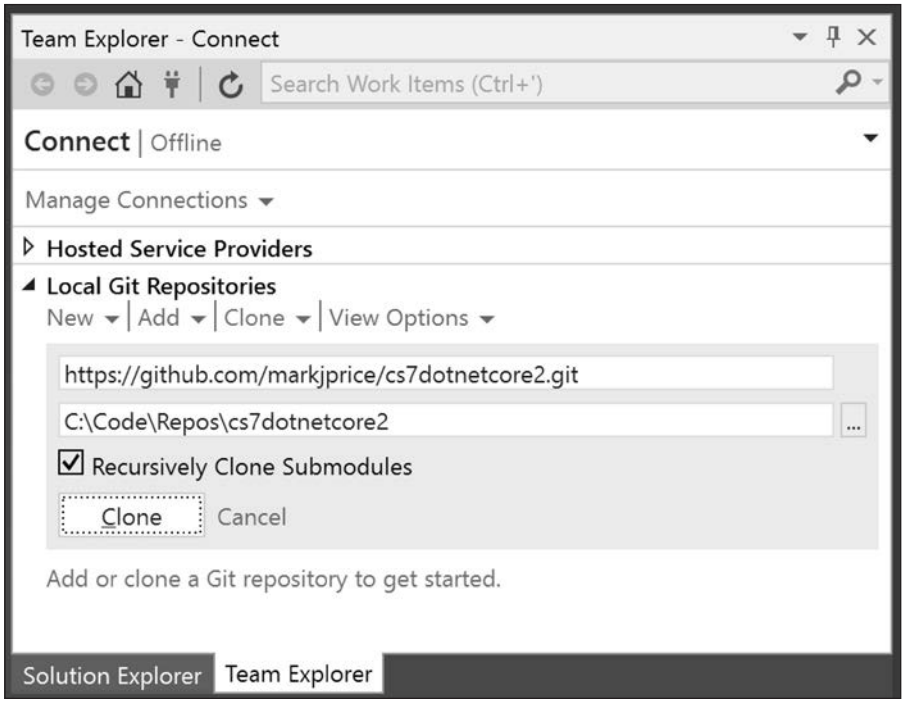
U prozoru Team Explorer proširite stavku **Local Git Repositories**, kliknite na meni **Clone**, a zatim za **Git** skladište koje želite da klonirate unesite sledeći URL:

https://github.com/markjprice/cs7dotnetcore2.git

Unesite putanju za klonirano Git skladište:

C:\Code\Repos\cs7dotnetcore2

Kliknite na dugme **Clone**, kao što je prikazano na sledećoj slici.



Sačekajte da se Git skladište klonira lokalno.

Sada ćete imati lokalnu kopiju kompletnih rešenja za sve praktične vežbe iz ove knjige.

Upravljanje GitHub skladištem

Dvostruko kliknite na skladište cs7dotnetcore2 da biste otvorili detaljan prikaz.

Možete da kliknete na opcije u odeljku **Project** da biste prikazali **Pull Requests** i **Issues** i druge aspekte skladišta.

Možete dvostruko da kliknete na unos u odeljku **Solutions** da biste ga otvorili u prozoru **Solution Explorer**.

Uporeba Gita u IDE-u Visual Studio Code

Visual Studio `Code` ima podršku za Git, ali će koristiti OS-ovu Git instalaciju, pa treba 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 radije koristite GUI, preuzmite GitHub Desktop sa sledećeg linka: https://desktop.github.com

Konfigurisanje Gita u komandnoj liniji

Pokrenite Terminal i unesite sledeću komandu da biste proverili konfiguraciju:

```
git config --list
```

Rezultat treba da uključuje korisničko ime i e-mail adresu, jer će ove informacije biti upotrebljene za svako potvrđivanje koje izvršite:

```
...other congfiguration...
user.name=Mark J. Price
user.email=markjprice@gmail.com
```

Ako korisničko ime i e-mail adresa nisu podešeni, da biste ih podesili, unesite sledeće komande, koristeći svoje ime i svoj e-mail:

```
git config --global user.name "Mark J. Price"
git config --global user.email markjprice@gmail.com
```

Možete da proverite pojedinačna podešavanja konfiguracije na sledeći način:

```
git config user.name
```

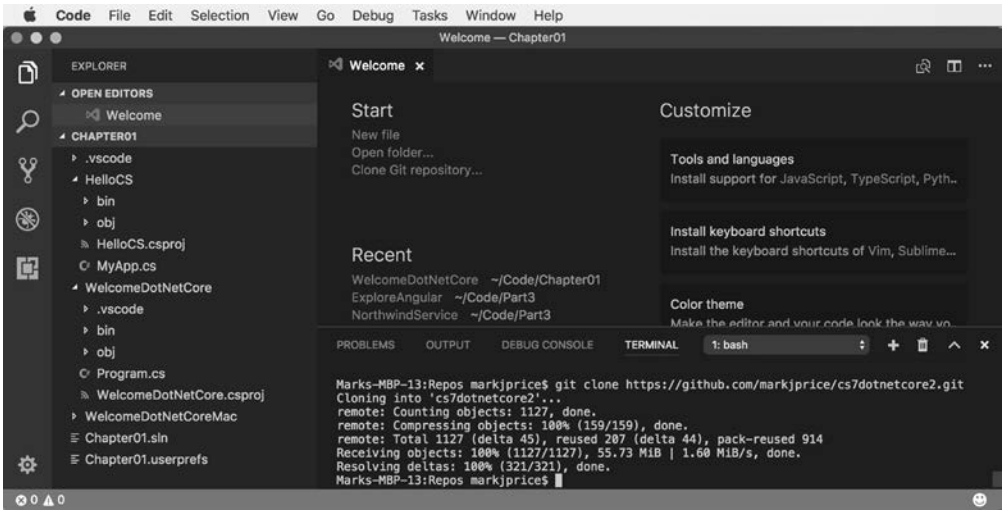
Upravljanje Gitom pomoću Visual Studio Codea

Pokrenite Visual Studio `Code` i otvorite direktorijum `Code`.

Kliknite na **View | Integrated Terminal** ili pritisnite *Ctrl* + `` ` `` i unesite sledeće komande:

```
mkdir Repos
cd Repos
git clone https://github.com/markjprice/cs7dotnetcore2.git
```

Kloniranje svih rešenja za sva poglavlja na vaš lokalni drajv će potrajati nekoliko minuta, kao što je prikazano na sledećoj slici.



Za više informacija o verzijama izvornog koda u Visual Studio Codeu pogledajte sledeći link:
https://code.visualstudio.com/Docs/editor/versioncontrol

VEŽBANJE I ISTRAŽIVANJE

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

Vežba 1.1 Testirajte svoje znanje

Odgovorite na sledeća pitanja:

1. Zašto programer može da upotrebi različite jezike - na primer, C# i F#, za pisanje aplikacija koje se pokreću na .NET Coreu?
2. Šta treba da ukucate u komandnu liniju da biste izgradili i izvršili C# izvorni kod?

3. Koje su prečice na tastaturi programerskih podešavanja Visual C#-a za snimanje, kompajliranje i pokretanje aplikacija bez priključivanja funkcije za uklanjanje grešaka?

4. Koja je prečica na tastaturi za Visual Studio Code za prikaz Integrated Terminala?
5. Da li je Visual Studio 2017 bolji od Visual Studio Codea?
6. Da li je .NET Core bolji od .NET Frameworka?
7. Po čemu se .NET Native razlikuje od .NET Corea?
8. Šta je .NET Standard i zašto je važan?
9. Koja je razlika između Gita i GitHuba?
10. Koji je naziv metoda za unos .NET konzolske aplikacije i kako treba da bude deklarisan?

Vežba 1.2 Vežbajte C# svuda

Nisu vam potrebni Visual Studio 2017, Visual Studio for Mac ili Visual Studio Code da biste vežbali jezik C#.

Pogledajte sledeće veb sajtove i započnite kodiranje:

- .NET Fiddle: https://dotnetfiddle.net/
- Cloud9: https://c9.io/web/sign-up/free

.

Vežba 1.3 Istražite teme

Upotrebite sledeće linkove da biste pročitali više detalja o temama koje su opisane u ovom poglavlju:

- **Welcome to .NET Core**: http://dotnet.github.io
- **.NET Core Command Line Interface (CLI) tool:** https://github.com/dotnet/cli
- **.NET Core runtime, CoreCLR**: https://github.com/dotnet/coreclr/
- **.NET Core Roadmap**: https://github.com/dotnet/core/blob/master/roadmap.md
- **.NET Standard FAQ**: https://github.com/dotnet/standard/blob/master/docs/faq.md
- **Visual Studio Documentation**: https://docs.microsoft.com/en-us/visualstudio/
- **Visual Studio Blog: https**://blogs.msdn.microsoft.com/visualstudio/
- **Git and Team Services**: https://www.visualstudio.com/en-us/docs/git/overview
- **The easiest way to connect to your GitHub repositories in Visual Studio**: https://visualstudio.github.com/

REZIME

U ovom poglavlju podesili smo radno okruženje, upotrebili smo Windowsov Command Prompt i macOS-ov terminal za kompajliranje i pokretanje konzolske aplikacije, upotrebili smo Visual Studio 2017, Visual Studio for Mac i Visual Studio Code za kreiranje slične aplikacije i opisali smo razlike između .NET Frameworka, .NET Corea, .NET Standarda i .NET Nativea.

U sledećem poglavlju ćete naučiti kako da govorite C# jezikom.

Deo 1

C# 7.1

Ovaj deo knjige posvećen je C# jeziku – gramatici i rečniku koje ćete koristiti svakog dana za pisanje izvornog koda za aplikacije.

Programski jezici imaju mnogo sličnosti sa ljudskim jezicima, osim što u programskim jezicima možete da smišljate sopstvene reči, kao što je učinio Teodor Seuss, koji je pod pseudonimom Dr.Seuss, pisao dečije knjige za decu.

U knjizi „If I Ran the Zoo“, koja je objavljena 1950. godine, on kaže sledeće:

> *„And then, just to show them, I'll sail to Ka-Troo And Bring Back an It-Kutch a Preep and a Proo A Nerkle, a Nerd and a Seersucker, too!“*

U ovoj knjizi opisan je C# 7.1. „Microsoft“ ima planove za buduće verzije C# jezika, kao što je prikazano u sledećoj tabeli:

VERZIJA	FUNKCIJE
C# 7.2	ref readonly, blittable, strongname, interior pointer, nontrailing imenovani argumenti, zaštita, separator Digit iza osnovnog specifikatora
C# 7.3	rasponi upotrebom operatora double-dot - na primer, 1..10
C# 8.0	metodi standardnog interfejsa, referentni tipovi koji prihvataju null

Možete da naučite više na linku

https://github.com/dotnet/roslyn/blob/master/docs/Language%20Feature%20Status.md

Upravo kada smo izdali treće izdanje, „Microsoft“ je izdao C# 7.2. Pogledajte članak na blogu o poboljšanjima u verziji C# 7.2 na adresi https:// https://goo.gl/4b76zu ili na „Packtovom“ veb sajtu, na linku https://www.packtpub.com/books/content/exploring-languageimprovements -c-72-and-73
Ja ću ažurirati ovaj blog i obavestiću vas o detaljima poboljšanja verzije C# 7.3 kada ova verzija bude izdata.

Da biste naučili jezik C#, potrebno je da kreirate neke jednostavne aplikacije. Da biste izbegli da se previše rano opteretite sa previše informacija, u poglavljima u prvom delu ove knjige će biti upotrebljen najjednostavniji tip aplikacije - konzolska aplikacija.

U sledećim poglavljima obradićemo sledeće teme:

- **u Poglavlju 2** - gramatika i rečnik jezika C#
- **u Poglavlju 3** - grananje i ponavljanje iskaza jezika C# i konvertovanje između C# tipova
- **u Poglavlju 4** - kako da ponovo upotrebite kod pomoću C# funkcija i kako da ispravite greške i testirate C# kod
- **u Poglavlju 5** - kako da upotrebite objektno-orijentisane funkcije jezika C#
- **u Poglavlju 6** - kako da upotrebite napredne objektno-orijentisane funkcije jezika C#