Šolski center Celje

Gimnazija Lava

IZDELAVA SPLETNEGA BRSKALNIKA

Raziskovalna naloga

Avtor

David Simunič, 2. e

Mentor

Tomislav Viher, univ. dipl. org.

Mestna občina Celje, Mladi za Celje

Celje, marec 2014

POVZETEK

V raziskovalni nalogi sem opisal postopek izdelave spletnega brskalnika. Ključno vprašanje te naloge je ali je to sploh mogoče. Najprej sem s pomočjo literature na kratko predstavil zgodovino svetovnega spleta in spletnega brskalnika, nato pa sistematično opisal postopek izdelave brskalnika~~,~~ od načrtovanja in priprav~~,~~ do izdelave vmesnika in pisanja kode ter nato še testiranja. Na koncu sem kot primer doma narejenega brskalnika na kratko predstavil še lasten izdelek, brskalnik Poseidon Browser, in navedel svoje ugotovitve.

VSEBINA

1 UVOD 1

2 KRATKA ZGODOVINA SPLETNEGA BRSKALNIKA 1

3 IZDELAVA BRSKALNIKA 2

3.1 NAČRTOVANJE 2

3.2 PRIPRAVE 2

3.3 UPORABNIŠKI VMESNIK 4

3.4 KODA 6

3.5 TESTIRANJE 7

4 PRIMER BRSKALNIKA – POSEIDON BROWSER 8

5 ZAKLJUČEK 9

6 VIRI 9

KAZALO SLIK IN TABEL

Slika 1: Začetna stran programa Visual Studio 2010 4

Slika 2: Pogled vmesnika v Visual Studiu 2010 5

Slika 3: Pogled kode v Visual Studiu 2010 6

Slika 4: Glavno okno brskalnika Poseidon Browser 8

Tabela 1: Tržni delež brskalnikov 1

1 UVOD

Spletni brskalnik je del osnovne programske opreme vsakega računalnika. Je okno v svetovni splet. Morda je nekoliko ironično, da si ga moramo pred namestitvijo prenesti iz spleta. A brez njega bi si težko predstavljali vsakdanje delo z računalnikom. Pa ni bilo vedno tako. V času ko je bil internetni dostop še prestiž, je tudi spletni brskalnik spadal med »naprednejše« programe. K sreči se je vse to spremenilo in danes je na voljo ogromno različnih brskalnikov, nekateri so bolj poznani, nekateri manj. In ker smo ljudje različni, je težko narediti nekaj, kar bi ustrezalo vsem. A kaj če bi lahko naredili svoj spletni brskalnik, ki bi povsem ustrezal našim potrebam? Je to sploh mogoče? In če je, se dejansko splača? Cilj te naloge je odgovoriti na ta vprašanja ter predstaviti postopek izdelave spletnega brskalnika, vse skupaj pa ponazoriti s primerom doma narejenega brskalnika.

2 KRATKA ZGODOVINA SPLETNEGA BRSKALNIKA

Preden se posvetim glavnemu delu naloge, naj na kratko predstavim še zgodovino spletnega brskalnika. Ker je njegov obstoj neposredno povezan s pojavom svetovnega spleta, moramo najprej vedeti kdaj je le-ta začel delovati. To je bila leta 1990. In ker nam svetovni splet nič ne koristi, če ne moremo dostopati do njega, je »oče« svetovnega spleta Tim Berners-Lee hkrati razvil tudi prvi spletni brskalnik, imenovan kar WorldWideWeb. Obstajal je samo za operacijski sistem NeXTSTEP, zadnja različica pa je bila izdana leta 1994. Leta 1993 se je pojavil prvi res »popularen« brskalnik, Mosaic. Zakaj je tako pomemben? Ker je omogočil dostop do spleta uporabnikom različnih operacijskih sistemov (Windows, Unix, Amiga OS in Mac OS), iz njega pa so se kasneje razvili brskalniki, ki se uporabljajo še danes (Microsoft Internet Explorer in Netscape Navigator, predhodnik današnjega Firefoxa). Skupaj s številom uporabnikov svetovnega spleta je raslo tudi število brskalnikov.

Danes so na voljo v vseh mogočih oblikah in za celo paleto operacijskih sistemov in naprav, med »velikih pet« pa štejemo Internet Explorer, Mozilla Firefox, Google Chrome, Safari in Opera. V tabeli spodaj lahko vidimo približen tržni delež petih najbolj priljubljenih spletnih brskalnikov na namiznih in prenosnih računalnikih februarja letos.

Tabela 1: Tržni delež brskalnikov

Brskalnik	Internet Explorer	Mozilla Firefox	Google Chrome	Safari	Opera
Tržni delež	24%	21%	47%	5%	1%

3 IZDELAVA BRSKALNIKA

3.1 NAČRTOVANJE

Kot pred vsakim projektom, je tudi tu dobro da pred začetkom izdelave malo razmislimo kako bomo delali in s čim. Najprej bi bilo pametno da se odločimo, kaj bi z našim brskalnikom sploh radi dosegli, katere bodo njegove ključne lastnosti, ali ima specifičen namen, itn. Morda je dobro, da si skiciramo približen izgled uporabniškega vmesnika, ki ga bomo razvili kasneje. Še ena ključna odločitev je ali bo projekt odprtokoden ali ne. V prvem primeru je izvorna koda dostopna vsem, kar lahko omogoči lažji in hitrejši razvoj, a če niste navdušeni nad idejo, da vsi brskajo po vašem izdelku je morda bolje, da je program zaprtokoden. Prav tako moramo vedeti, če bomo celoten projekt vodili in izdelali sami, ali nam bo pri tem kdo pomagal - porazdelitev vlog. Določimo še razvojne stopnje programa (npr. alpha, beta, ipd.) in si zamislimo smiselno shemo označevanja različic.

3.2 PRIPRAVE

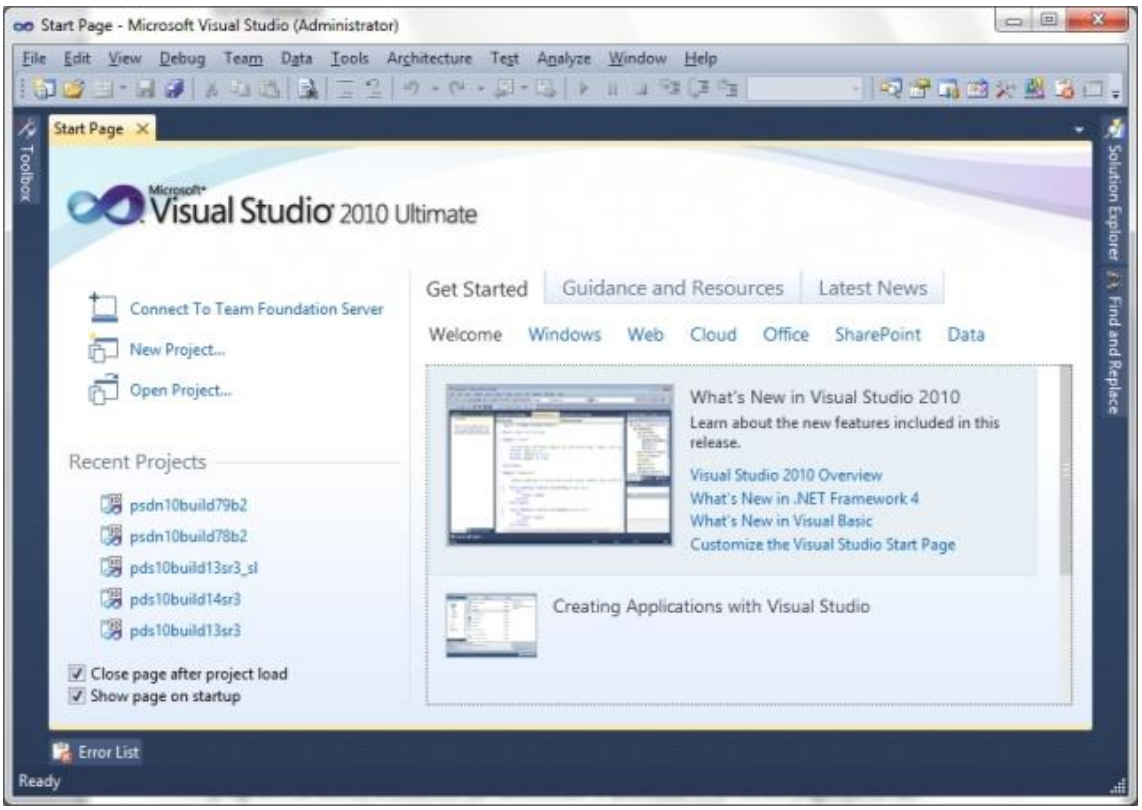
Zdaj, ko smo določili okvirno ideologijo projekta, je potrebno določiti še tehnične podrobnosti. Prva odločitev, ki jo moramo sprejeti je za kateri OS bomo razvili naš spletni brskalnik. Razvoj je tako ali tako na večini sistemov enako zahteven, torej je vse odvisno od tega na koliko uporabnikov ciljamo z našim izdelkom. V tej nalogi se bom osredotočil na Microsoftov operacijski sistem Windows. Naslednja stvar je izbira programskega jezika. Za Windows lahko razvijamo v številnih programskih jezikih, zato je tu izbira pestra. Vendar pa niso vsi programski jeziki enako zahtevni, zato je pametno, da se odločimo našem znanju in sposobnostim primerno. Za začetnike je dokaj primeren VB.NET (Microsoftov dialekt jezika Visual Basic, ki temelji na njihovem ogrodju .NET Framework), zato ga bom uporabil za potrebe te naloge. Programski jezik na nek način pogojuje razvojno okolje (angl. IDE), in za VB.NET nimamo prav velike izbire, razen Microsoftovega okolja Visual Studio. Lahko prenesemo nekoliko okrnjeno, a brezplačno različico Visual Studio Express, ali pa nekoliko sežemo v žep in si kupimo profesionalno različico Visual Studio, ki omogoča številne napredne možnosti. V mojem primeru sem se poslužil polne različice, do katere sem dostopil preko DreamSparka. Ta dostop mi omogoča dejstvo, da je Šolski center Celje član Microsoftovega akademskega omrežja MSDNAA in imamo dijaki možnost na svoj računalnik legalno pretočiti razvojna orodja iz njihovega nabora. V vsakem primeru to ne vpliva na sam razvoj, zato je tu izbira povsem trivialna. Tudi v okviru ogrodja .NET Framework lahko za izdelavo uporabniškega vmesnika izberemo dva pristopa: Windows Forms in WPF. Windows Forms je starejša tehnologija, ki omogoča enostavno izdelavo vmesnika, vendar se stvari dokaj zapletejo, če želimo uporabiti nekoliko naprednejše elemente in efekte. Rešitev za ta problem je WPF, ki je na voljo od izida Windows Viste leta 2006. Poleg kopice novih možnosti za razvoj vmesnika, lahko tega poleg grafičnega načina

urejamo tudi v pogledu kode, in sicer v jeziku XAML (prilagojen XML). Če se z izgledom programa ne obremenjujemo preveč in želimo nekaj enostavnega in hitrega, potem Windows Forms povsem zadošča, sicer pa uporabimo WPF.

Vsak grafični (GUI) program je sestavljen iz dveh delov - iz uporabniškega vmesnika (to, kar vidimo) in kode (to, kar program počne v ozadju). V primeru spletnega brskalnika se del kode, ki skrbi za prikazovanje spletne strani skupno imenuje "pogon" brskalnika (angl. layout/rendering engine). Pogoni so se skozi čas spreminjali skupaj s svetovnim spletom in se mu prilagajali (spletni standardi). Nekateri se obnašajo podobno kot drugi, nekateri nekoliko drugače. Danes na trgu spletnih brskalnikov prevladujejo trije pogoni: Microsoftov Trident (poganja Internet Explorer in cel kup manj znanih brskalnikov), Mozillin Gecko (poganja Firefox in spet veliko ostalih brskalnikov) ter Applov in Googlov WebKit (poganja brskalnike Safari, Google Chrome in še nekatere druge). Drugi in tretji sta odprtokodna, torej je koda dostopna vsem in jo lahko spreminja vsak, zato obstaja več različnih implementacij teh dveh pogonov, namenjenih različnim brskalnikom. Prav tako sta na voljo za različne platforme, medtem ko je Trident na voljo samo za okolje Windows. Vendar pa vključitev Gecko ali WebKit v program, ki ga izdelujemo v razvojnem okolju Visual Studio, ni tako »čudovit« postopek, kot tudi ne programiranje z njima, saj zahteva nekoliko več "prakse". V tem primeru je Trident začetnikom najbolj prijazna izbira, ker je dobro integriran v Visual Studio, pa še na vsakem računalniku z nameščenimi Okni se nahaja, torej ga ni potrebno prenesti skupaj z brskalnikom in je velikost slednjega zato manjša. Izbira je prosta, a v tej nalogi bom uporabil Trident zaradi zgoraj navedenih razlogov. Seveda je možno napisati lasten pogon, ampak je to daleč prezahtevno za to nalogo in novince v programiranju nasploh, zato je preprosto lažje, da uporabimo delo nekoga drugega in mu v programu napišemo zahvalo.

Tako, dogovorili smo se kako in kje bomo delali. Ciljamo na operacijski sistem Microsoft Windows, razvijali bomo v okolju Visual Studio in jeziku Visual Basic.NET, za pogon brskalnika pa bomo uporabili Trident. Zdaj pa je čas, da pljunemo v roke in začnemo z delom.

Ko odpremo Visual Studio, nas najprej pozdravi začetna stran. Tam najdemo povezave do številnih vodičev, zbirk znanja in dokumentacije, ki nam olajša razvoj programske opreme. Najprej moramo ustvariti nov projekt. V pogovornem oknu, ki se nam prikaže, izberemo ciljni jezik in vrsto programa (med drugim lahko razvijamo tudi programe brez grafičnega vmesnika in spletne aplikacije) ter naš projekt poimenujemo. Sedaj lahko pred izdelavo vmesnika spremenimo nekaj nastavitev projekta, če je to potrebno. Spremenimo lahko osnovne podatke o programu (ime izvršne datoteke, različica, ime razvijalca, itn.) in še nekatere druge možnosti razvoja.



Slika 1: Začetna stran programa Visual Studio 2010

Spletni brskalnik je dokaj obsežen in zapleten program, ki ga ne moremo stlačiti v eno samo okno, zato je včasih potrebno projektu dodati nove komponente, npr. okna, module in še kaj. Kliknemo na meni Project in na seznamu izberemo vrsto komponente, ki jo želimo dodati. Najbolje je, da najprej dodamo vse komponente in jih nato samo urejamo po vrsti.

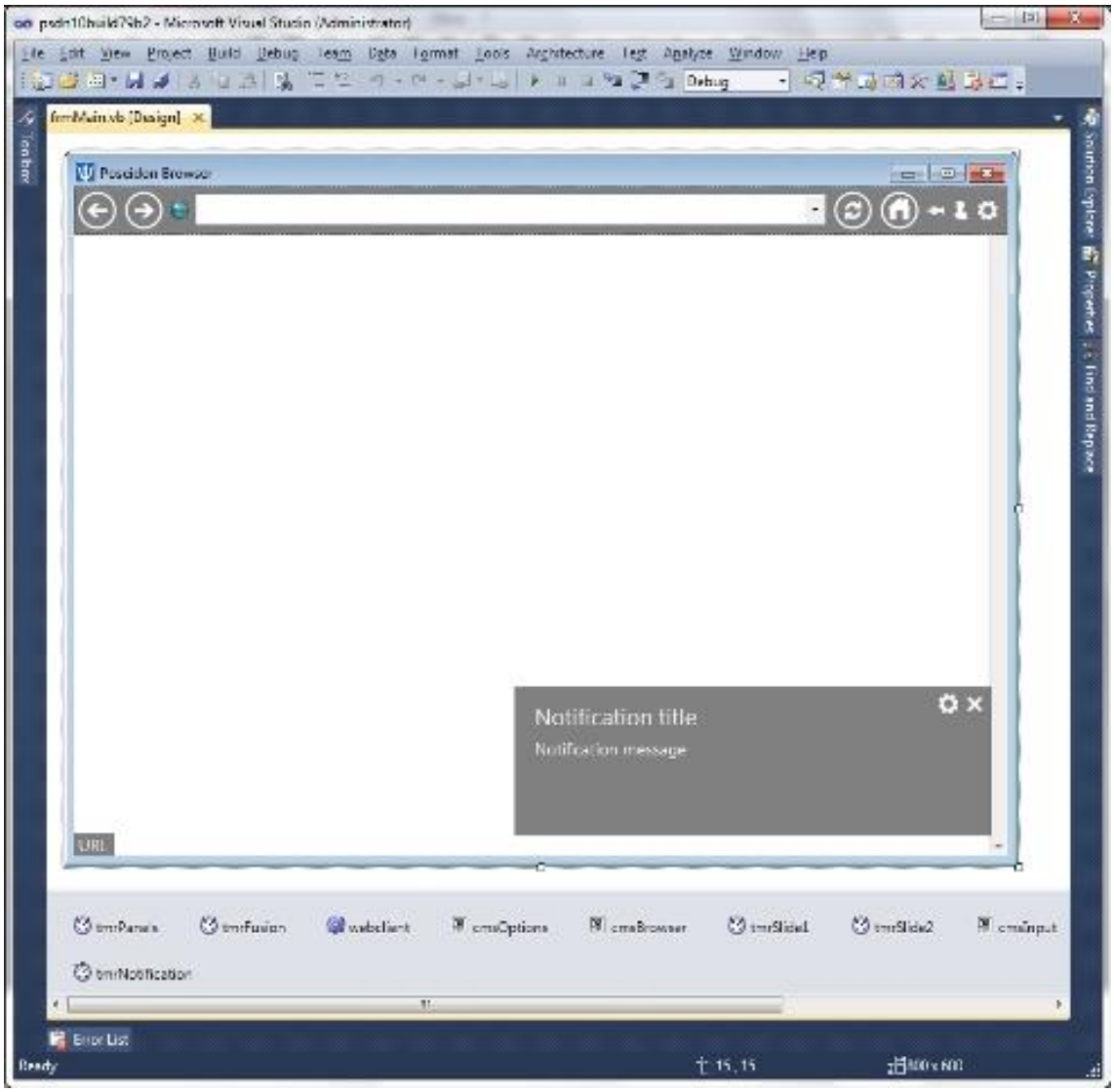
3.3 UPORABNIŠKI VMESNIK

Kot pri razvoju večine programov, je tudi tu pametno, da najprej naredimo grobo obliko uporabniškega vmesnika (GUI), preden se lotimo programiranja. Osnovni princip grafičnega uporabniškega vmesnika so razni grafični elementi z različnimi vlogami, v Visual Studiu so poimenovani "kontrole" (angl. controls). Mednje spadajo recimo gumbi, potrditvena polja, besedilna polja in še ogromno drugih. Visual Studio nam ob ustvarjanju projekta že sam pripravi glavno okno programa, seveda je zaenkrat še prazno. Nanj bomo namestili kontrole in jih malce oblikovali, nato pa se bomo spustili v kodo.

Večina brskalnikov ima nekaj gumbov (recimo za nazaj, naprej, domov, osvežitev strani, itn.), besedilno polje kamor lahko uporabnik vnese naslov spletne strani (tja tudi brskalnik izpiše naslov strani po koncu nalaganja) ter neko polje kjer se prikaže spletna stran.

Na levi strani okna Visual Studio najdemo orodno vrstico Toolbox. V njej je seznam vseh kontrol, ki jih lahko dodamo na trenutno okno. Seveda v pogledu kode

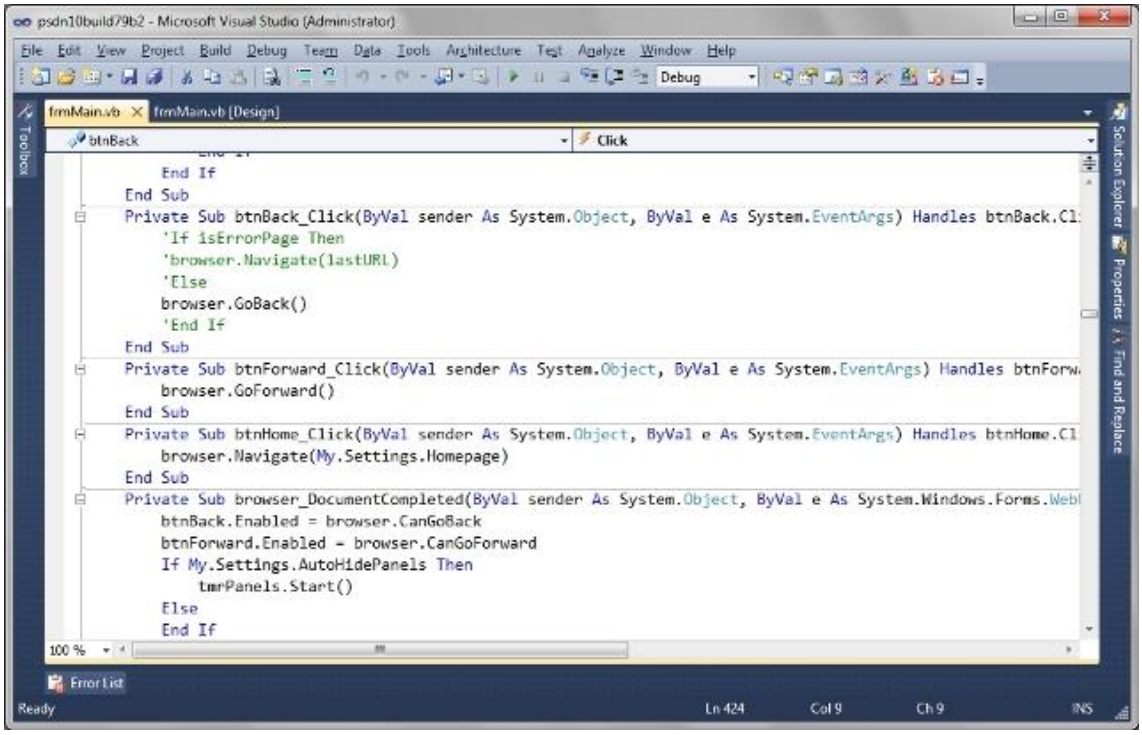
ne moremo dodajati grafičnih elementov, zato tam ta orodna vrstica izgine. Kontrole so razvrščene glede na namen, preprosto poiščemo tisto, ki jo želimo dodati in jo povlečemo na predogled okna. Če recimo želimo dodati nov gumb, na seznamu poiščemo kontrolo Button in jo povlečemo v okno. Za osnoven brskalnik dodamo še naslovno vrstico (lahko je tipa »TextBox« ali pa »ComboBox«) in kontrolo WebBrowser, kjer se bodo prikazovale spletne strani. Ta kontrola je dejansko neke vrste lupina, ki se ovije okoli pogona Trident in nam s tem olajša delo z njim. Ko je kontrola v oknu, ji lahko spremeni nekatere lastnosti. Temu je namenjena orodna vrstica Properties na desni. V njej so razvrščene lastnosti raznih vrst in njihov vrednosti. Ko izgled okna ustreza našim željam se odpravimo naprej in uredimo naslednje okno, itn. dokler nismo z vmesnikom v grobem končali. Na srečo bo Visual Studio sam poskrbel za kodo, ki skrbi za lastnosti kontrol in okna, tako da nam je kar nekaj dela prihranjenega. Zdaj je čas da začnemo s tistim »pravim« programiranjem – pisanjem kode.



Slika 2: Pogled vmesnika v Visual Studiu 2010

3.4 KODA

Ko je vmesnik približno končan (podrobnostim se lahko posvetimo kasneje), je čas da napišemo nekaj vrstic kode, da bo naš program sploh deloval. Tako kot pogled vmesnika, ima Visual Studio tudi pogled kode. Če ga želimo priklicati za določeno kontrolo, jo samo dvokliknemo in Visual Studio nas bo samodejno postavil na dogodek, ki se za to kontrolo najpogosteje uporablja. Seveda lahko izberemo tudi druge dogodke (ali pa jih ustvarimo sami), kar efektivno ustvari nov podprogram (sub). Napišemo, kaj želimo da se ob dogodku zgodi in nadaljujemo. Pri pisanju kode je daleč preveč možnosti, da bi zajel vse in jih opisal v tej nalogi, zato se pričakuje da je uporabnik delno že seznanjen z načinom dela v okolju Visual Studio. Če recimo želimo, da se ob kliku na gumb »nazaj« brskalnik vrne na prejšnjo stran, bomo v podprogram »btnBack.Click« napisali »webbrowser.goback()«.



Slika 3: Pogled kode v Visual Studiu 2010

Tudi tu nam je delo močno olajšano, saj je zahvaljujoč tehnologiji IntelliSense naša koda sproti preverjena. Če pride do težav, so nam predlagane rešitve in alternativni načini izpeljave. Kljub temu se mimogrede zmotimo in napišemo »slabo kodo«, zato je dobro da smo z jezikom in programiranjem na sploh že prej dobro seznanjeni. Ko se nam zdi, da smo programu določili vse kar naj stori ob izvajanju, je čas, da našo kodo preizkusimo.

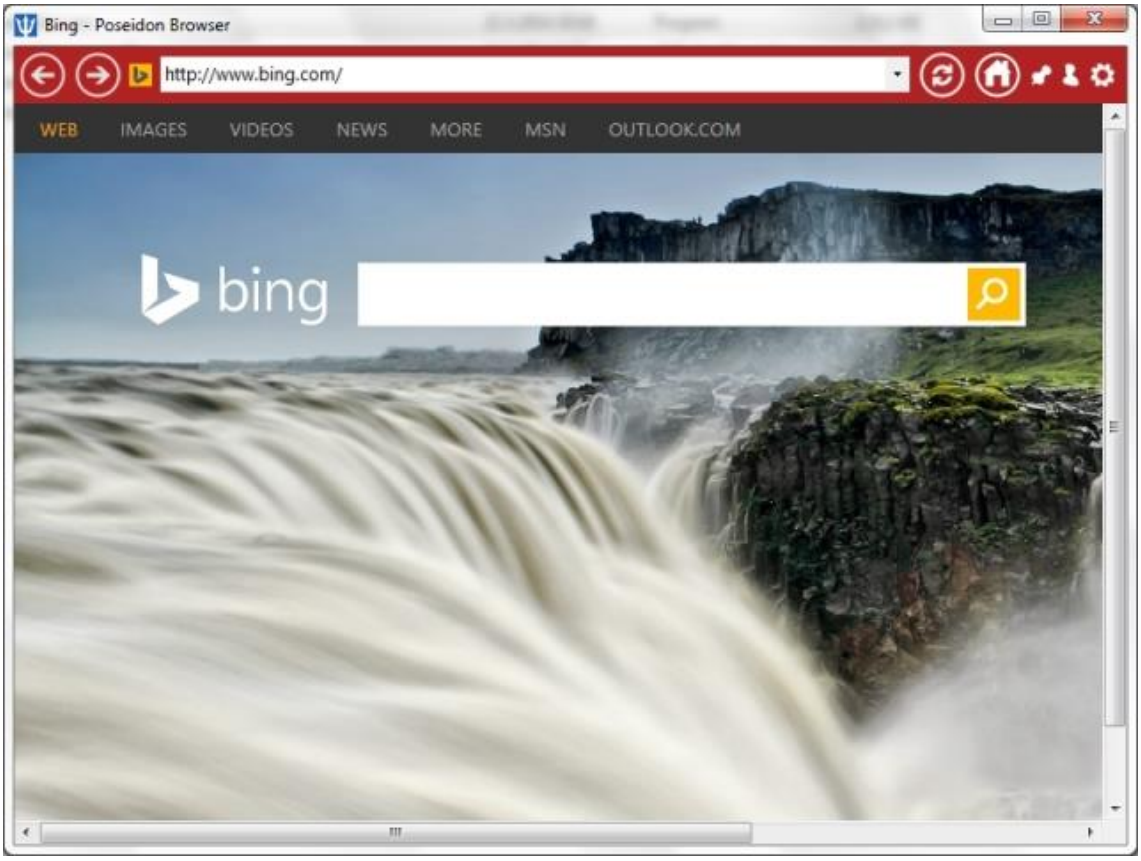
3.5 TESTIRANJE

Visual Studio ima vgrajen razhroščevalnik (angl. debugger), ki med izvajanjem programa išče napake in nas nanje opozori ter nam ponudi še nekaj podrobnosti za lažje odpravljanje napak. Če smo recimo spremenljivki pozabili določiti vrednost, ki jo kasneje prikličemo, nas bo Visual Studio na to opozoril in nam izpisal točno vrstico v kodi, kjer se to zgodi, ponudi pa še nekaj povezav do rešitev na spletu. Vsekakor je potrebno morebitne napake odpraviti in nato program še enkrat pognati ter preizkusiti. Vendar pa je to le prvi del preizkušanja. Čeprav se nam zdi, da program deluje brezhibno na razvojnem računalniku, pa ni nujno, da to drži tudi za ostale. Program moramo preizkusiti tudi na drugih računalnikih, po možnosti čim bolj raznolikih, saj se tako čim bolj približamo tržnemu okolju, kjer različni uporabniki programe poganjajo na različnih napravah.

Če preizkusi pokažejo, da je program dovolj stabilen in zanesljiv za izid, ga lahko ponudimo uporabnikom. V Visual Studiu izberemo ukaz »Build«, ki bo ustvaril končno izvršno datoteko našega programa. Ta in druge potrebne datoteke (npr. datoteke DLL, katere kliče naš program) se nahajajo v mapi programa Visual Studio v imeniku Dokumenti za vsakega uporabnika posebej. To še ni namestitvena datoteka. Čeprav Visual Studio omogoča izdelavo preprostih namestitvenih programov, se pričakuje, da zanje poskrbi razvijalec sam. Nekateri programi so dovolj enostavni, da namestitev ni potrebna in jih lahko poganjamo kar neposredno.

4 PRIMER BRSKALNIKA – POSEIDON BROWSER

Poseidon Browser je moj lasten spletni brskalnik, napisan v okolju Visual Studio in jeziku VB.NET za okolje Windows in ga poganja Trident. Projekt sem začel konec novembra 2011, trenutno pa je tik pred prehodom v stopnjo t.i. "kandidata za končno različico" (angl. release candidate). V bistvu je brskalnik, ki jemlje navdih za uporabniški vmesnik iz Microsoftovih vmesnikov Aero in Metro ter ju poskusi čim bolje povezati. Projekt ni odprtokoden, brezplačen prenos preizkusne različice pa je na voljo na uradni spletni strani.



Slika 4: Glavno okno brskalnika Poseidon Browser

Nekatere izmed ključnih lastnosti brskalnika Poseidon Browser:

- preprost in pregleden vmesnik, ki združuje vmesnika Aero in Metro
- Poseidon Deployment System – lasten namestitveni program za brskalnik, prav tako napisan v jeziku VB.NET
- »My Manager« – vsi naši prenosi, priljubljene strani, zgodovina in piškotki na enem mestu
- »Fusion button« – en sam gumb za osveži, prekliči, išči in pojdi, odvisno od konteksta.

- Update Agent – skrbi, da imate vedno nameščeno najnovejšo različico brskalnika
- Tour – kratek vodič, ki vas seznani s programom in vam pomaga ob prvem zagonu

Namen brskalnika Poseidon Browser je iztisniti čim več iz pogona Trident in okolja Visual Studio ter se čim bolj približati funkcionalnosti večjih brskalnikov. Program je za enkrat namenjen zgolj preizkušanju in še ni primeren za vsakodnevno uporabo, pričakujem da se bo to spremenilo nekje do poletja tega leta.

5 ZAKLJUČEK

Če torej povzamemo bistvo te naloge, ugotovimo, da je res mogoče doma izdelati spletni brskalnik. To je konec koncev eden izmed mnogih programov, izdelava le-teh pa že dolgo ni več omejena zgolj na poklicne programerje. Ugotovil sem, da je za amaterskega programerja to morda rahlo prevelik zalogaj, vsaj kar se funkcionalnosti tiče. En sam človek je namreč omejen tako v času kot znanju, ki sta mu na voljo, zato večjih uspehov na tem področju ne gre pričakovati. Če bi teoretično želeli konkurirati velikim brskalnikom, bi potrebovali celo ekipo ljudi in kar nekaj finančnih sredstev, kar pa, kot že rečeno, povprečnemu ljubiteljskemu programerju ni na voljo. Je pa izdelava brskalnika vsekakor dober uvod v izdelavo nekoliko naprednejših programov in ponuja dobre možnosti za učenje načrtovanja in programiranja aplikacij.

6 VIRI

- POSEIDON'S features. [online]. *Poseidon – A metro browser*. [Zadnja sprememba 24. 1. 2014]. [Citirano 11. 3. 2014]. Dostopno na spletnem naslovu: http://poseidon.robjansen.info/features.htm.
- TOP 5 Desktop Browser on Feb 2014. [online]. *StatCounter Global Stats*. [Zadnja sprememba 28. 2. 2014]. [Citirano 13. 3. 2014]. Dostopno na spletnem naslovu: http://gs.statcounter.com/#desktop-browser-ww-monthly-201402-201402-bar.
- WEB browser. [online]. *Wikipedia: The free encyclopedia*. [Zadnja sprememba 5. 3. 2014]. [Citirano 10. 3. 2014]. Dostopno na spletnem naslovu: http://en.wikipedia.org/wiki/Web_browser.
- WORLD wide web. [online]. *Wikipedia: The free encyclopedia*. [Zadnja sprememba 6. 3. 2014]. [Citirano 9. 3. 2014]. Dostopno na spletnem naslovu: http://en.wikipedia.org/wiki/World_wide_web.