Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Primož Hadalin

IZDELAVA SPLETNEGA PORTALA POSLOVNE APLIKACIJE Z UPOŠTEVANJEM RAZLIK MED SPLETNIMI BRSKALNIKI

diplomska naloga
na visokošolskem strokovnem študiju

mentor: doc. dr. Boštjan Slivnik

Ljubljana, 2007

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete

Zahvala

Doc. dr. Boštjanu Slivniku za mentorstvo.

Mag. Gregorju Pipanu za podporo.

Jaki Močniku za pripombe glede vsebine in oblike.

Borisu Turku in Marku Novaku za vsestransko pomoč.

Janji Koren za inspiracijo.

Amaliji Leban

Kazalo

Povzetek

Spletne aplikacije postajajo zelo razširjene in vse bolj konkurenčne namiznim aplikacijam. V relativno kratkem času se je razvilo veliko tehnik in tehnologij, ki so prilagojenie za njihovo izdelavo. Pojavili so se tudi različni spletni standardi, ki naj bi zagotavljali enotnost izgleda in izdelave spletnih strani, vendar je bil napredek spletnih tehnologij hitrejši od napredka teh standardov. To je povzročilo, da so razvijalci vodilnih spletnih brskalnikov lahko uveljavljali svoje standarde, kar je povzročilo nekonsistenten videz spletnih strani v različnih brskalnikih in tako oteževalo delo programerjev. Danes brskalniki mnogo bolj upoštevajo spletne standarde, še vedno pa mora imeti programer določeno znanje, da zagotovi enoten izgled spletnih strani v različnih spletnih brskalnikih. Namen te diplomske naloge je predstavitev poslovnih aplikacij s poudarkom na spletni realizaciji le-teh, predstavitev tehnologij, tehnik in postopkov za izdelavo spletne aplikacije ter seznanitev s potrebnim znanjem za zagotavljanje enotnega videza uporabniškega vmesnika v različnih brskalnikih. Področje, na katero se diplomsko delo nanaša, so predvsem spletni standardi in spletne tehnologije. V prvem delu so predstavljene poslovne aplikacije in značilnosti njihovih namiznih in spletnih realizacij. V drugem delu najdemo predstavitev spletnih standardov in tehnologij ter značilnost izdelave uporabniškega vmesnika spletne aplikacije, v zadnjem pa postrežemo s primeri prikazane razlike med različnimi spletnimi brskalniki pri prikazovanju spletnih strani. Temelji pa na spletnem portalu, pri izdelavi katerega je avtor diplomske naloge sodeloval.

Ključne besede:

spletne aplikacije, spletni standardi, HTML, CSS, spletni brskalniki

Poglavje 1

Poslovne aplikacije

V tem poglavju bomo spoznali osnovne značilnosti poslovnih aplikacij. Ogledali si bomo njihovo arhitekturo, ki je v zadnjem času največkrat uporabljena ter razliko med namizno in spletno realizacijo. V zadnjem delu pa si bomo ogledali primer spletne poslovne aplikacije, pri izdelavi katere sem v okviru diplomske naloge sodeloval.

1.1 Trinivojska arhitektura

1.1.1 Namen in izvor

Trinivojska arhitektura aplikacije (ang. three-tier architecture) [1] se je pojavila v devetdesetih letih prejšnjega stoletja, da bi odpravila slabosti dvonovojske arhitekture [2]. Slabost dvonivojske arhitekture je v tem, da sta nivoja uporabniškega vmesnika in poslovne logike združenega v enega, ločitev teh dveh nivojev pa zagotavlja večjo zmogljivost, fleksibilnost, zmožnost vzdrževanja (ang. maintainability), ponovno uporabnost (ang. reusability) in razširljivost (ang. scalability), medtem ko uporabniku skriva kompleksnost poslovne logike. Zaradi teh karakteristik je trinivojska arhitektura primerna izbira pri izdelavi tako namiznih kot tudi spletnih aplikacij.

1.1.2 Tehnične podrobnosti

Trinivojsko arhitekturo sestavljajo naslednji nivoji:

- uporabniški vmesnik,
- poslovna logika,
- podatkovna baza.

Vsak izmed teh nivojev je grajen in vzdrževan samostojno, pogosto na različnih platformah, kar pomeni, da so lahko nadgrajeni ali zamenjani neodvisno drug od drugega. Če je na primer uporabniški vmesnik narejen za operacijski sistem Microsoft Windows, bo sprememba potreba le na nivoju uporabniškega vmesnika, da bo deloval tudi na operacijskem sistemu Unix ali pa prek spletnega brskalnika (glej sliko 1.1).

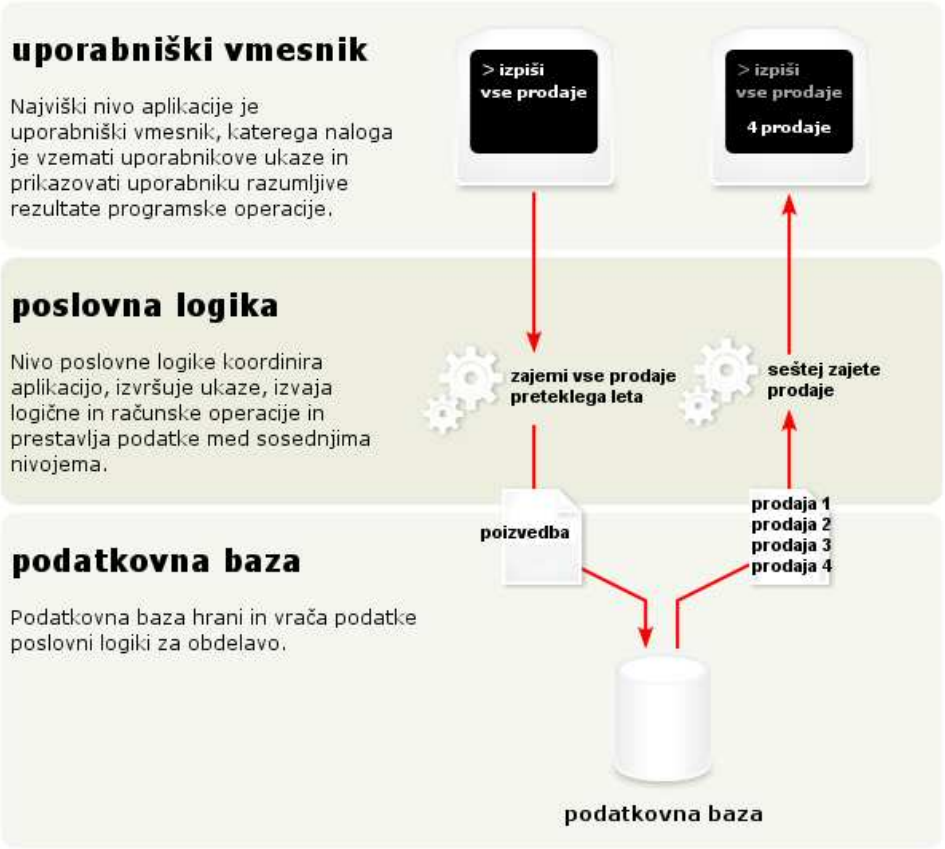
Na nivoju uporabniškega vmesnika se nahajajo uporabniške storitve, s katerimi uporabnik komunicira s sistemom. Ta nivo dostopa do nivoja poslovne logike prek vmesnika, ki zagotavlja, da je implementacija poslovne logike skrita pred uporabniškim vmesnikom. Uporabnik ne more neposredno dostopati do drugih nivojev, ampak le posredno prek poslovne logike. Uporabniški vmesnik je lahko realiziran na več načinov, vendar več o tem v poglavju 2.

Nivo poslovne logike:

- zajema vse poslovne objekte (npr. računi, transakcije, dokumenti ...),
- predpisuje, kako poslovni objekti sodelujejo med seboj,
- premešča in obdeluje informacijo med nivojema uporabniškega vmesnika in podatkovne baze.

Nivo poslovne logike je lahko tudi sam razdeljen v več nivojev. V tem primeru govorimo o večnivojski arhitekturi (ang. multi-tier architecture).

Na nivoju podatkovne baze se podatki shranjujejo in vračajo nivoju poslovne logike za obdelavo in posredovanje do nivoja uporabniškega vmesnika.



Slika 1.1: Trinivojska arhitektura

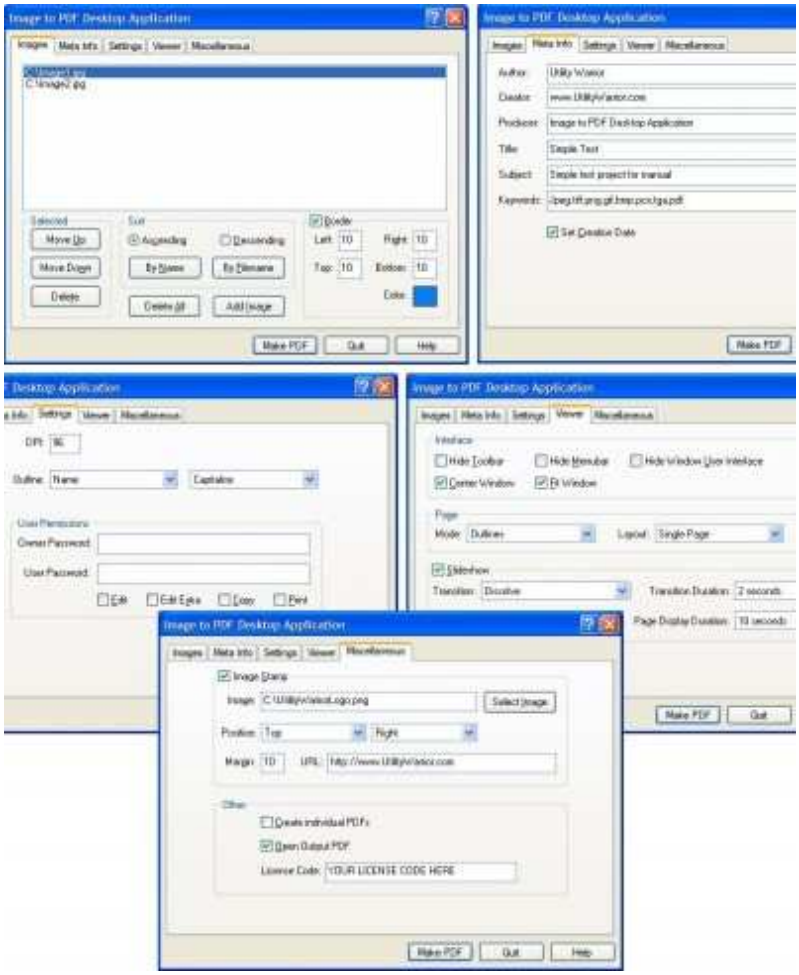
1.2 Realizacija poslovnih aplikacij

1.2.1 Namizna realizacija

Z imenom namizna aplikacija (ang. desktop application) označujemo aplikacije, ki se izvajajo na lokalnem računalniku. Značilno za tak tip aplikacije je, da je programska oprema nameščena lokalno, da se procesi izvajajo lokalno in da potrebuje dostop do omrežja le za določene storitve, kot je na primer dostop do podatkovne baze (glej sliko 1.2).

Programski jeziki, ki se danes pogosto uporabljajo pri izdelavi namiznih aplikacij, so:

- Java,



Slika 1.2: Namizna aplikacija

- C#,
- C++,
- Visual Basic.

Java

Tehnologija Java je nastala pri podjetju Sun Microsystems. Zajema štiri sklope:

- programski jezik,
- knjižnico razredov (ang. class library),
- navidezni stroj (ang. virtual machine),
- varnostni model (ang. security model).

Programski jezik Java je bil prvotno namenjen programiranju manjših elektronskih naprav, kasneje ga je računalniška industrija sprejela kot programski jezik, ki je neodvisen od strojne platforme. Danes je tehnologija Java zelo razširjena, najdemo jo na mnogih področjih, kot so mobilni telefoni, dlančniki, poslovne aplikacije in mnogo drugih.

Programski jezik Java zaznamujejo naslednje značilnosti:

neodvisnost od strojne platforme - Izvorna koda se prevede v vmesno kodo (ang. byte code), ki se izvaja v javanskem navideznem stroju (ang. Java Virtual Machine). Na ta način dosežemo prenosljivost (ang. portability) programov med platformami, ki podpirajo JVM (glej sliko 1.3).

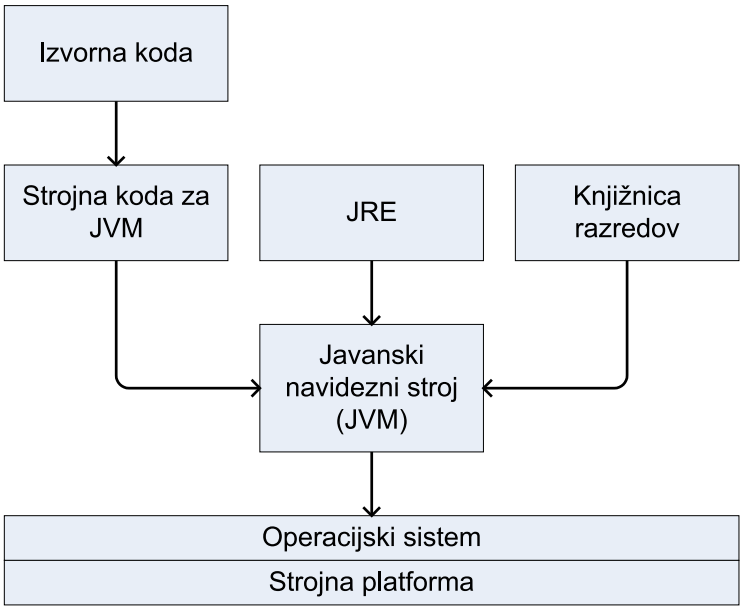
objektna usmerjenost - V Javi „je vse objekt“ in vsi objekti izhajajo iz osnovnega objekta (ang. root object).

bogata standardna knjižnica - V Javi je na voljo bogata standardna knjižnica z več sto razredi razdeljenimi v šest kategorij.

sintaksa podobna jeziku C++ - Eden izmed dejavnikov, ki omogočajo hitro prilagoditev na programski jezik Java, je podobnost sintakse s popularnim programskim jezikom C++.

Primer izvorne kode:

```
class HelloWorld
{
  public static void main(String args[])
  {
    System.out.println("Pozdravljen svet!");
  }
}
```



Slika 1.3: Delovanje javanskega navideznega stroja.

C#

Zadnji dve desetletji sta bila C in C++ najbolj uporabljana programska jezika za razvijanje poslovnih aplikacij. Oba programska jezika sta zelo fleksibilna, vendar je fleksibilnost obratno sorazmerna s produktivnostjo. V primerjavi s programskim jezikom Visual Basic traja razvoj aplikacij s programskim jezikom C ali C++ pogosto dlje.

Podjetje Microsoft poskuša s programskim jezikom C# združiti kvalitete prej omenjenih programskih jezikov. Čeprav je C# podedoval kar nekaj elementov programskega jezika Visual Basic, mu je C++ najbližji sorodnik, podobnosti pa najdemo tudi s programskim jezikom Java, kot so objektna usmerjenost, sintaksa in delovanje na principu navideznega stroja, kar omogoča neodvisnost od strojne platforme. C# namreč deluje v okolju .NET, ki je Microsoftova različica javanskega navideznega stroja.

Primer izvorne kode:

```
using System;
public class Hello
{
  public static void Main()
```

```
  {
    Console.WriteLine("Pozdravljen svet");
  }
}
```

1.2.2 Spletna realizacija

Spletna aplikacija (ang. rich internet application) [3] se od namizne razlikuje po tem, da je dostopna preko omrežja. Programska oprema je nameščena na oddaljenem strežniku in je uporabniku ni potrebno namestiti lokalno. Za dostop do spletne aplikacije običajno potrebuje le spletni brskalnik. Spletne aplikacije se postale zelo priljubljene zaradi t.i. vseprisotnosti (ang. ubiquity) odjemalca. Zmožnost vzdrževanja programske opreme, ne da bi bilo to potrebno početi na potencialno več tisoč odjemalčevih računalnikih, je ključen razlog za priljubljenost spletnih aplikacij. Tehnologije in tehnike za izdelavo so podrobneje opisane v poglavju 2 diplomske naloge.

1.2.3 Razlike med realizacijama

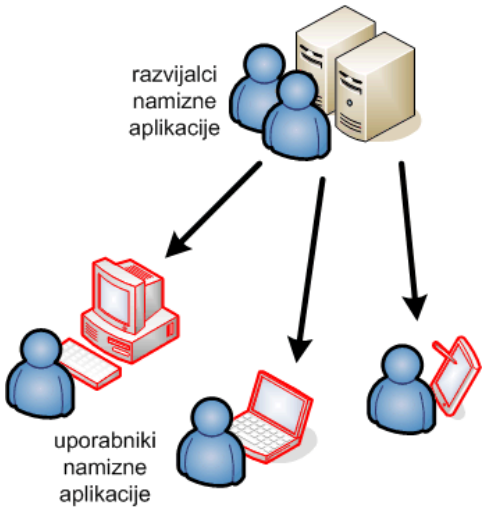
Čeprav so spletne aplikacije vedno bolj priljubljena izbira, je verjetnost, da bodo namizne aplikacije izumrle, zelo majhna. Slednje imajo namreč veliko prednosti pred spletnimi, zato si poglejmo prednosti posamezne izbire.

Razlogi za postavitev namizne aplikacije

Želimo izkoristiti sistemska sredstva. Namizne aplikacije, ki izvajajo zahtevne in časovno potratne operacije in pri katerih je hitrost izvajanja ključnega pomena, se bolje obnesejo kot spletne.

Gradimo kompleksnejši uporabniški vmesnik. Gradniki namiznega uporabniškega vmesnika postajajo boljši in zmogljivejši, medtem ko imamo pri spletnem brskalniku na voljo le osnovne gradnike, ki se razvijajo počasneje. Postavitev spletne aplikacije zahteva prilagoditev na vmesnik spletnega brskalnika. Pogosto imajo brskalniki dodatno funkcionalnost, kot so na primer različne orodne vrstice (ang. toolbars), ki spreminja podobo in način uporabnikove interakcije z aplikacijo. Pri namizni aplikaciji imamo nad vmesnikom popolno kontrolo.

Ne potrebujemo dostopa do omrežja. Prvi pogoj za delovanje spletne aplikacije je dostop do omrežja. Če mora aplikacija delovati tudi, ko dostop do omrežja ni mogoč, se odločimo za namizno aplikacijo (slika 1.4).



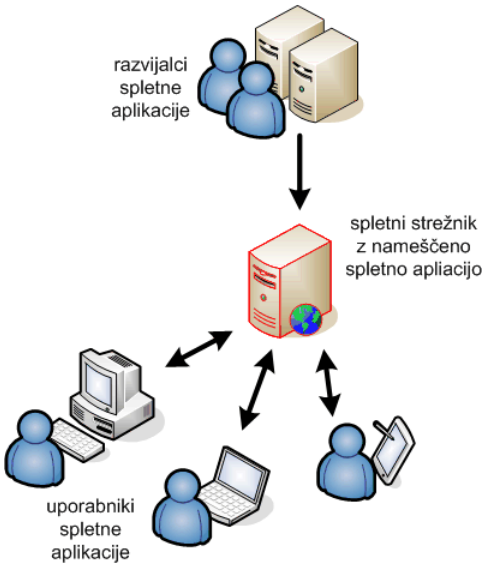
Slika 1.4: Dostop do namizne aplikacije

Razlogi za postavitev spletne aplikacije

Želimo programsko opremo na enem mestu. Ker je programska oprema spletne aplikacije nameščena na strežniku in ne na odjemalčevem računalniku, je vzdrževanje in nadgrajevanje aplikacije enostavnejše, saj to počnemo le na enem mestu. Dodatna prednost je tudi, da lahko uporabnik do aplikacije dostopa z več računalnikov (slika 1.5).

Želimo strojno neodvisno aplikacijo. Spletni brskalniki obstajajo za vse najbolj razširjene operacijske sisteme.

Želimo uporabnika hitro seznaniti z aplikacijo. Spletni brskalniki so priročni, za dostop do spletne aplikacije potrebujejo le spletni naslov, aplikacija je hitro dosegljiva. Ko želimo, da uporabniki nekaj preizkusijo ali začnejo hitro uporabljati aplikacijo, je spletna aplikacija boljša izbira.



Slika 1.5: Dostop do spletne aplikacije

1.3 Primer spletne poslovne aplikacije

V okviru diplomske naloge sem sodeloval pri izdelavi obsežnejše spletne poslovne aplikacije.

Zaradi uvedbe direktive Evropskega parlamenta o spodbujanju proizvodnje električne energije iz obnovljivih virov, je Agencija RS za energijo izrazila potrebo po spletnem portalu, ki omogoča izdajanje potrdil o izvoru (v nadaljevanju: PoI) električne energije, proizvedene iz obnovljenih virov. Ta olajša trgovanje in izboljša preglednost izdanih potrdil.

Prednosti enega nacionalnega informatiziranega registra PoI so:

- standardizacija - en sam register PoI v RS omogoča lažjo izvedbo standardizacije vseh potrebnih dokumentov in postopkov;
- ekonomičnost - manjši stroški vzpostavitve in delovanja samo enega registra PoI;
- večja transparentnost - transparentnost in preglednost nad enim sistemom PoI je večja kot v primeru delovanja več različnih sistemov;
- enostavnejše povezovanje s tujimi sistemi potrdil o izvoru.

Naročnik je podal specifikacije registra PoI, v katerem je bilo natančno opisano želeno delovanje registra PoI, zahteval pa je tudi večji poudarek na:

varnosti - uporaba zakrite povezave (s pomočjo protokola HTTPS [4]), digitalnih certifikatov, gesel;

sledljivosti - vse spremembe v registru se beležijo;

dostopnosti - dostop do sistema je z uporabo digitalnega potrdila in gesel naj bo mogoč s kateregakoli računalnika;

večjezičnosti - uporabniški vmesnik naj bo narejen tako, da dopušča enostavno nadgradnjo v tuje jezike;

oblikovanju - uporabniški vmesnik naj bo oblikovan uporabniku prijazno, uporabljeni so oblikovalski pristopi, ki zagotavljajo preglednost vsebine in enostavno uporabo sistema.

Poleg naročnika, ki je podal zahteve glede delovanja, je zunanji oblikovalec podal izgled uporabniškega vmesnika (slika 1.6). Poskusili smo ga realizirati, da bi bil konsistenten v različnih brskalnikih, vendar smo ga zaradi razlik pri prikazovanju v različnih brskalnikih prilagodili brskalniku Internet Explorer različice 6 ali novejše.

Slika 1.6: Spletna poslovna aplikacija Register PoI

Poglavje 2

Uporabniški vmesnik spletne aplikacije

V tem poglavju si bomo ogledali standarde, tehnologije in pristope, ki jih uporabljamo pri izdelavi uporabniškega vmesnika spletne aplikacije.

2.1 Spletni standardi

Obstaja več spletnih standardov, ki predpisujejo delovanje spletnih brskalnikov. Kot navaja spletna stran [5], segajo začetki standardov v začetek devetdesetih let prejšnjega stoletja, večji premik pa je se zgodil leta 1997, ko je konzorcij W3C (World Wide Web Consortium) skupaj s podjetji IBM, Microsoft, Netscape Communications Corporation, Novell, SoftQuad, Spyglass in Sun Microsystems izdal standard HTML 3.2.

Zaradi velikih razlik pri interpretaciji programske kode spletnih strani, je nastalo nekaj združenj, ki razvijalcem zagotavljajo koristne informacije, med katerimi sta Web Standards Project [6] in Web Standards Group [7].

2.1.1 Standardi W3C

Konzorcij W3C je mednarodna organizacija, ustanovljena leta 1994 z namenom, da razvije spletne standarde in smernice, ki naj bi omogočile dolgoročno

uporabnost svetovnega spleta. W3C ima več kot 400 članov iz celega sveta in je svetovno priznana organizacija. Osnovni cilji W3C konzorcija so:

- dostopnost do svetovnega spleta po celem svetu,
- uporabnost svetovnega spleta,
- zaupanje v svetovni splet.

Jezik HTML

Jezik HTML (HyperText Markup Language) [8] je na spletu zelo razširjen za opis spletnih strani. Spletni brskalniki interpretirajo te dokumente in predstavljajo njihovo strukturo uporabniku v različnih oblikah, odvisno komu je spletni brskalnik namenjen. Poudarjeno besedilo bodo vizualni spletni brskalniki na primer odebelili, medtem ko bodo brskalniki namenjeni slabovidnim to besedilo med izgovorjavo izrekli glasneje.

Jezik XML

Jezik XML (eXtensible Markup Language) [9] je podoben jeziku HTML v smislu, da je strukturiran, vendar namesto da bi uporabljali že vnaprej določene elemente kot pri jeziku HTML, lahko določimo svojo zbirko elementov. Dopušča tudi več različnih zbirk elementov v istem dokumentu z uporabo različnih imenskih prostorov (ang. namespace). Jezik XML je bolj fleksibilen kot jezik HTML zaradi možnosti dodajanja lastnih elementov.

Namenjen je opisu strukturiranih podatkov v obliki besedila, zaradi katerega je primeren za deljenje podatkov med sistemi povezanimi preko Interneta ter organizacijo velikih in kompleksnih zbirk podatkov. Najdemo ga v mnogih podatkovnih bazah in iskalnikih. Njegove prednosti so preprostost, razumljivost človeškemu bralcu, neodvisnost od platforme in predvsem razširljivost.

Jezik XHTML

Jezik XHTML (eXtensible HyperText Markup Language) [11] je nadgradnja jezika HTML, vendar temelji na jeziku XML, kar pomeni, da je strožji glede

sintakse in tako spodbuja programerje, da se držijo določenih pravil pri pisanju spletnih dokumentov.

V jezik XHTML je vpeljana tudi modularizacija [17], razdeljen je na skupine sorodnih elementov (modulov), saj so se na trgu pojavile naprave (mobilni telefoni, dlančniki, televizije), ki ne uporabljajo vseh elementov, ampak samo tiste, ki so za njih primerne. Najpomembnejši moduli so:

- osnovni moduli (zgradba dokumenta, modul besedila, modul seznamov),
- moduli za oblikovanje besedila,
- moduli za tabele,
- moduli za slike,
- moduli za obrazce.

CSS

CSS (Cascading Style Sheets) [10] je jezik, ki nam omogoča, da oblikovne lastnosti spletne strani (primer oblikovnih lastnosti so: barva, pisava, postavitev) ločimo od vsebine, kar poveča preglednost napisane kode. Najpogosteje ga uporabljamo za določanje oblikovnih lastnosti spletnih strani napisanih v jeziku HTML, lahko pa ga uporabimo tudi z dokumenti XML.

Podobno kot drugi spletni standardi se tudi CSS še vedno razvija. Začelo se je leta 1996 s CSS stopnje 1. Dve leti kasneje so objavili priporočila za CSS stopnje 2, izkazalo pa se je, da potrebuje še precej popravkov, zato leta 2002 začnejo sestavljati prvo dopolnitev. Žal si avtorji brskalnikov priporočila razlagajo vsak po svoje in občasno se zgodi, da je neka spletna stran v enem brskalniku videti precej drugače kot v drugem.

DOM

DOM (Document Object Model) je vmesnik, ki dopušča dostop in spremembo vsebine, strukture in oblike strukturiranih dokumentov na objektno usmerjen način. DOM je razdeljen v stopnje, ki vsebujejo potrebne in opcijske module. Za stopnjo 1, stopnja 2 in nekateri moduli stopnje 3 obstajajo priporočila organizacije W3C.

2.1.2 Standardi ECMA

Organizacija ECMA International s sedežem v Ženevi je mednarodna organizacija za standardizacijo informacijskih in komunikacijskih sistemov. Prvotno ime te organizacije je bilo European Computer Manufacturers Association, a se je leta 1994 preimenovala, da bi odražala svojo mednarodno aktivnost. ECMA je odgovorna za številne standarde, med katerimi so ISO 9660 (datotečni sistem za pogone CD-ROM), specifikacije jezika C# in ECMAScript.

ECMAScript

ECMAScript [12] je standardiziran skriptni jezik, ki temelji na skriptnih jezikih JavaScript in JScript.

Glavni namen tega objektno usmerjenega jezika je manipulacija objektov spletne strani, ki so dosegljivi preko vmesnika DOM. Te objekte (elemente, ki sestavljajo spletno stran) lahko dodajamo, odvzemamo, premikamo ali spreminjamo njihove lastnosti.

2.2 Spletne tehnike in tehnologije

2.2.1 CGI in Perl

CGI (Common Gateway Interface) je ena prvih tehnik, ki se uporabljajo za izdelovanje dinamičnih spletnih strani. Vse od otroških časov svetovnega spleta so strežniki potrebovali vmesnik, prek katerega so lahko komunicirali z drugimi programi. Programerji spletnih strežnikov so se tedaj premišljeno odločili za razvoj standardnega postopka, ki deluje neodvisno od strežnikove programske opreme. Med tovrstne standardne vmesnike sodi tudi vmesnik CGI. Teoretično je moč za CGI programe uporabiti poljuben programski jezik, kljub temu pa se izkaže, da so nekateri jeziki primernejši od drugih predvsem ko gre za kompleksne programe. Programski jezik Perl se je uveljavil kot nekakšen standard za CGI programe in ga podpirajo praktično vsi spletni strežniki.

2.2.2 PHP

PHP (Hypertext Preprocessor) je strežniški skriptni jezik, razvit iz programskega jezika Perl leta 1995. Glavni cilj piscev PHP-ja je bil, da omogočijo izdelovalcem spletnih strani hitro gradnjo dinamičnih spletnih strani. PHP je zelo prilagodljiv in ga je preprosto kombinirati s številnimi drugimi jeziki in aplikacijami na različnih operacijskih sistemih. Posebno močan je v sodelovanju z raznimi podatkovnimi bazami. Podprte so vse najbolj razširjene podatkovne baze kot so MySQL, PostgreSQL, Oracle in druge.

Programski jezik PHP lahko pišemo v poljubne urejevalniku besedil, samostojno ali v kombinaciji z drugimi jeziki (na primer HTML), a s samo kodo ne moremo nič. Potrebujemo še strežniški program, strežniški modul za podporo PHP-ja in brskalnik. Za strežnik je največkrat uporabljen zelo razširjen odprtokodni strežnik Apache. Brskalnik je po izbiri.

Kodo lahko vstavljamo v katerokoli vrstico HTML ali pa jo pišemo samostojno. Na primer:

```
<html>
  <head>
    <title>Pozdravljen svet</title>
  </head>
  <body>
    <?php echo "Pozdravljen svet!"; ?>
  </body>
</html>
```

2.2.3 JSP

Kot že kratica JSP (Java Server Pages) pove, bomo imeli opravka z programskim jezikom Java. Da bi razumeli delovanje strani JSP, si najprej oglejmo, kako se je JSP razvil, pri tem pa bomo spoznali tudi druge tehnologije, ki so pripomogle k njegovemu razvoju.

HTML

Strani, v katere vključujemo JSP, so strani HTML. Na začetku so bile te strani statične, razvijalci spletnih strani pa so hoteli več, zato se je razširilo pro-

gramiranje na uporabnikovi strani (ang. client-side programming) z uporabo skriptnih jezikov kot je na primer JavaScript.

JavaScript

Razen z imenom JavaScript in Java nista povezana. Za razliko od JSP, pri katerem gre za izvajanje programov na strežniku, gre pri JavaScript za izvajanje programov na strani odjemalca, torej v brskalniku. Naslednji primer kaže, kako se JavaScript dodaja v dokument HTML:

```
<html>
  <head>
    <title>Pozdravljen svet</title>
  </head>
  <body>
    <script type="text/JavaScript">
      document.write("Pozdravljen svet!");
    </script>
  </body>
</html>
```

Izvajanje programov na uporabnikovi strani je lahko zelo uporabno, saj ima kar nekaj prednosti, kot je na primer hitrost izvajana, po drugi strani pa je tudi zelo omejujoče. Zaradi varnostnih razlogov tako nimamo dostopa do datotek in ker je JavaScript vgrajen v brskalnik, mora biti relativno majhen in se po zmogljivosti ne more primerjati s programskih jezikom kot je na primer Java. Ravno v času, ko so se razvijali skriptni jeziki, se je vzporedno razvijal tudi programski jezik Java.

Java

Java originalno ni bila razvita za programiranje spletnih strani, vendar podjetje Sun, ki jo razvija, ni moglo ignorirati hitrega razvoja interneta in prvi poskus na tem področju so bili javanski programčki (ang. applets).

Brskalnikovi programčki

Brskalnikov programček je že preveden Java program, ki ga dodamo v HTML dokument. Ko odpremo spletno stran, se brskalnikov programček samodejno

prenese na lokalni računalnik in požene v brskalniku, prikaže pa se v prostoru na strani, katerega smo mu namenili. Zmožni so dela z grafiko, prikazovanja animacij ali upravljanja z raznimi elementi strani. Sledi primer, ki prikazuje izdelavo brskalnikovega programčka in njegovo dodajanje v HTML dokument. Najprej spišemo programsko kodo brskalnikovega programčka:

```
import java.applet.Applet;
import java.awt.*;

public class HelloWorld extends Applet
{
  public void paint(Graphics g)
  {
    g.drawString("Pozdravljen svet!", 60, 100);
  }
}
```

Nato kodo prevedemo, in jo vključimo v HTML dokument:

```
<html>
  <head>
    <title>Pozdravljen svet</title>
  </head>
  <body>
    <applet code="helloWorld.class"
    width="200" height="200"></applet>
  </body>
</html>
```

Brskalnikovi programčki lahko prikazujejo tekst, gumbe in druge elemente, vendar so, kot pri jeziku JavaScript, zelo omejeni. Njihov čas je že skoraj minil, saj so na trg prišle nove tehnologije, kot je na primer Flash [13]. Naslednji korak je bil premakniti izvajanje programa na strežnik, kar omogočajo strežniški programčki (ang. servlets).

Strežniški programčki

Strežniški programčki so bili prvi poskus izkoriščanja polne moči programskega jezika Java v spletnih aplikacijah. Tako kot brskalnikovi programčki so tudi strežniški programčki napisani v programskem jeziku Java. Z uporabo strežnika s podporo strežniškim programčkom lahko dostopamo do prevedene

izvorne kode in jo poganjamo, rezultat pa je poslan brskalniku. V tem smislu lahko strežniške programčke obravnavamo kot brskalnikove programčke, ki se izvajajo na strežniku. Čeprav so zelo zmogljivi, je njihovo programiranje precej zapleteno, kar je tudi razlog, da niso postali priljubljeni, zato je bil naslednji korak JSP. Sledi primer izvorne kode strežniškega programčka:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet
{
  public void doGet(HttpServletRequest request,
    HttpServletResponse response)
    throws IOException, ServletException
  {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    out.println("<html>");
    out.println("<head>");
    out.println("<title>");
    out.println("Pozdravljen svet");
    out.println("</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("Pozdravljen svet!");
    out.println("</body>");
    out.println("</html>");
  }
}
```

JSP

JSP je zgrajen vrh spletnih programčkov, njegovo programiranje pa je mnogo enostavnejše, kar je razlog, da se je obdržal.

Vsaka JSP stran se prevede v spletni programček, vendar je to očem uporabnika skrito. Za programerja so JSP strani HTML dokumenti, ki vsebujejo javanske ukaze. Na podoben princip delujejo tudi drugi jeziki, kot na primer

PHP, razlika je v sintaksi in v strežniku na katerem se izvaja programska koda. Sledi primer izvorne kode JSP strani:

```
<html>
  <head>
    <title>Pozdravljen svet</title>
  </head>
  <body>
    <% out.println("Pozdravljen svet!"); %>
  </body>
</html>
```

2.2.4 ASP.NET

ASP.NET je Microsoftova različica okolja za izdelavo dinamičnih spletnih strani. Delovanje temelji na .NET ogrodju, ki je lahko dodano ali že vključeno v operacijski sistem Windows. ASP.NET je naslednik Microsoftove tehnologije ASP (Active Server Pages).

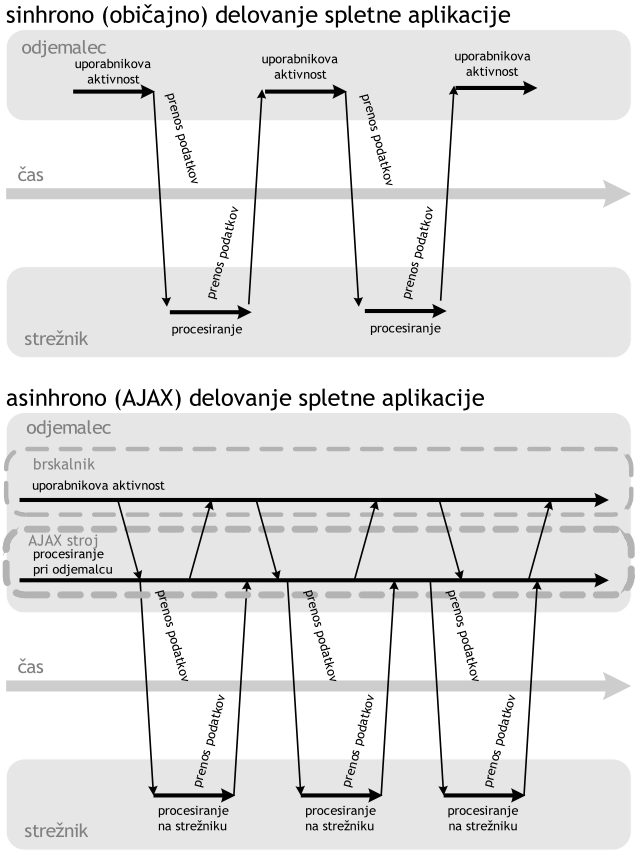
Programerji lahko pišejo dinamične spletne strani v poljubnem jeziku, ki ga podpira ogrodje .NET, med katere sodi tudi priljubljeni C#. Programsko kodo lahko vključujemo v HTML dokument kot pri drugih tehnologijah (PHP, JSP...) in s tem poenostavimo dodajanje programske funkcionalnosti spletnim stranem. Vendar pri takem početju otežujemo razumljivost kode, zato je pri večjih spletnih aplikacijah priporočljivo pisati celotno programsko kodo v enem jeziku, na primer C#.

Za razliko od tehnologije ASP, kjer se spletne aplikacije interpretirajo, se pri ASP.NET poganjajo prevedene, kar poveča hitrost izvajanja.

2.2.5 AJAX

AJAX (Asynchronous JavaScript And XML) je tehnika za izdelavo interaktivnih spletnih aplikacij. Ni tehnologija, ampak le pristop k uporabi številnih obstoječih spletnih tehnologij. Namen je izdelati bolj odzivne spletne strani z izmenjavo manjših količin podatkov med odjemalcem in strežnikom, tako da ni potrebno osveževati celotne spletne strani vsakič, ko uporabnik sproži zahtevo, temveč samo določen del. AJAX povečuje interaktivnost, hitrost in uporabnost spletnih strani.

AJAX je asinhron, saj se ne vmešava v običajen proces nalaganja spletne strani. Funkcijski klici so izvedeni s programskim jezikom JavaScript, za prenos podatkov pa je uporabljen XML. Delovanje je vidno na sliki 2.1.



Slika 2.1: Sinhrono delovanje spletne strani v primerjavi z asinhronim

2.3 Posebnost izdelave vmesnika v brskalniku

Za razliko od grafičnega vmesnika namizne aplikacije, kjer je točno določeno obnašanje gradnikov grafičnega vmesnika posamezne platforme, moramo pri spletni aplikaciji upoštevati, da različni brskalniki različno upoštevajo predstavitvena pravila.

Brskalnik za prikazovanje spletnih strani uporablja postavitveni pogon (ang. layout engine) [14]. Njegova naloga je, da za vhodna podatka vzame vsebino

spletne strani (na primer HTML ali XML dokument) in predstavitvene podatke (na primer CSS) in oblikovano vsebino prikaže v oknu brskalnika.

Obstaja več postavitvenih pogonov, najbolj razširjeni so:

Gecko - Firefox, Camino, Netscape

Trident - Microsoft Internet Explorer

WebCore - Safari, OmniWeb, Shiira

Presto - Opera 7 in novejše različice

Poglejmo si, kaj ima programer spletnih aplikacij na razpolago in katere koncepte je dobro poznati pred izdelavo spletne aplikacije.

2.3.1 Ločevanje vsebine in predstavitve dokumenta

Ena od primarnih priporočil standarda HTML 4 je ločevanje vsebine in predstavitve dokumenta. To je kritičnega pomena, saj veliko spletnih dokumentov tega ne upošteva, kar povzroča težave kot so počasnejše branje dokumenta in nekonsistentnost med različnimi brskalniki. To pa nista edina razloga, zakaj je to priporočljivo početi. Med drugim olajšuje vzdrževanje dokumenta, saj imamo predstavitveno informacijo na enem mestu in ne razdrobljene po potencialno več sto dokumentih. Prednost je tudi v možnosti zahtevnejšega oblikovanja in prenosljivosti.

Praktičen primer ločevanja strukture in oblike

Sledi primer HTML dokumenta, ki ne vsebuje predstavitvenih elementov, ampak samo povezavo na datoteko s predstavitvenimi pravili. Razvidno je tudi, da vsebuje element *div* prilastek *box* s katerim mu določimo predstavitveno pravilo.

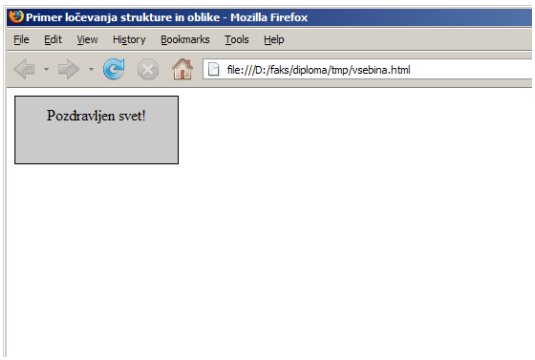
```
<html>
  <head>
    <title>Primer ločevanja strukture in oblike</title>
    <link rel="stylesheet" href="oblika.css"
    type="text/css" media="screen" />
```

```
  </head>
  <body>
    <div class="box">Pozdravljen svet!</div>
  </body>
</html>
```

Tako pa je videti vsebina datoteke *oblika.css*, ki vsebuje v našem primeru samo eno predstavitveno pravilo:

```
.box {
  background-color: #cacaca;
  border: solid 1px #000000;
  padding: 10px;
  width: 150px;
  height: 50px;
  text-align: center;
}
```

Rezultat zgornjega primera je viden na sliki 2.2.



Slika 2.2: Primer izgleda HTML dokumenta z uporabo CSS

2.3.2 Uporaba deklaracije Document Type Declaration

Pravila različic jezika HTML in XHTML so opisana v definicijah DTD (Document Type Definitions), ki tekstovne datoteke z opisom, ki določa, kako naj bo posamezen spletni dokument sestavljen, v spletni dokument pa jih vključimo z deklaracijo DOCTYPE (Document Type Declaration):

```
<!DOCTYPE html PUBLIC
```

```
"-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Definicije DTD so uporabne, ker lahko z njihovo pomočjo preverimo pravilnost napisanega spletnega dokumenta in se tako držimo standarda. Brskalnik sam po sebi ne preverja pravilnosti napisane kode spletnega dokumenta, so pa zato na voljo posebne aplikacije, imenovane validatorji.

Poglejmo tipe direktiv DOCTYPE pri HTML 4.01:

Strict Tip direktive Strict poudarja vsebino dokumenta pred predstavitvijo. To pomeni, da večina predstavitvenih elementov in atributov ni dovoljenih, je pa zato dovoljena uporaba CSS. Prav tako ni dovoljena uporaba okvirjev (ang. frames) in pri elementu povezava (ang. link) prilastek *target*.

Transitional Tip direktive Transitional, imenovan tudi ohlapen tip je bil narejen, ker CSS ni na razpolago v vseh primerih, zato dovoljuje preproste predstavitvene elemente in atribute. Čeprav so predstavitveni elementi dovoljeni, jih je še vedno priporočljivo ločiti od vsebine.

Frameset Tip direktive Frameset vsebuje informacijo specifično za spletne dokumente, ki uporabljajo okvirje (ang. frames), s katerimi lahko razdelimo brskalnikov vmesnik na več dokumentov. Spletni dokumenti znotraj posameznega okvira lahko vsebujejo poljubno različico direktive DTD.

Načini prikazovanja vsebine spletne strani

Nekateri novejši brskalniki uporabljajo tri načine za prikazovanje vsebine spletne strani. Pri prvem, t.i. standard načinu se brskalnik poskuša držati pravil W3C, pri drugem, t.i. *quirks* načinu pa posnema delovanje starejših različic brskalnika, da ohrani videz ne glede na različico. Brskalniki, ki temeljijo na Gecko postavitvenem pogonu, novejši od različice 1.0.1, (Firefox, Safari, Opera 7.5 ali novejši), uporabljajo še tretji, t.i. *almost standards* način, ki je enak *standards* načinu, razlikuje se le v postavitvi slik znotraj celic tabele. Postavlja jih po *quirks* načinu, kateri je bolj podoben brskalniku Internet Explorer.

Sprožitev različnih prikazovalnih načinov

Brskalniki določijo način prikazovanja glede na obliko direktive Document Type Declaration [20]. Če je podana polna direktiva DOCTYPE, potem bo običajno brskalnik uporabil *standards* način:

```
<!DOCTYPE html PUBLIC
  "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/DTD/strict.dtd">
```

Če direktiva DOCTYPE ni podana ali če v direktivi ni podane različice HTML dokumenta, potem bo brskalnik običajno uporabil *quirks* način:

```
<!DOCTYPE html PUBLIC>
```

2.3.3 W3C validacija

Z validacijo spletnega dokumenta, mislimo na postopek, ki preverja skladnost spletnega dokumenta z definicijo DTD, ki je podana v dokumentu. Na spletu je na voljo več spletnih aplikacij za validacijo, največkrat omenjen pa je aplikacija W3C [18].

Poglavje 3

Razlike med brskalniki

V tem poglavju si bomo ogledali kako različni brskalniki obravnavajo enake HTML in XHTML dokumente. Opisali bomo različne napake, ki se pojavljajo, in na kakšen način jih rešujemo.

Vsak programer, ki ima zadano nalogo narediti zahtevnejši uporabniški vmesnik spletne aplikacije, ki naj bi deloval v različnih brskalnikih, naleti na težave. Ne samo da se brskalniki razlikujejo po tem, kako prikazujejo vsebino, poznamo namreč brskalnike, ki prikazujejo samo besedilo brez slik, ampak tudi v tem, kako obravnavajo HTML elemente. Da je zmeda še večja, brskalniki zaradi večje učinkovitosti vpeljujejo svoje elemente, v novejših različicah pa spreminjajo interpretacijo elementov. Nekateri brskalniki, ki jih najdemo na prenosnih telefonih ali dlančnikih sploh ne uporabljajo jezika HTML ali XHTML, temveč WML ali HDML.

Čeprav je Microsoftov Internet Explorer še vedno najbolj razširjen brskalnik, tega po nekaterih internetnih virih uporablja slabih 80 odstotkov uporabnikov, se odstotek drugih brskalnikov kot sta Firefox in Safari počasi veča.

Razlog za tako velik delež uporabnikov Microsoftovega Internet Explorerja je predvsem v integraciji tega brskalnika v najbolj razširjenem operacijskem sistemu Windows. Po navedbah nekaterih internetnih virov ima družina operacijskih sistemov Windows približno 90-odstoten tržni delež. To podjetju Microsoft omogoča, da kljub vzpodbujanju različnih organizacij k enotni obravnavi spletnih standardov samo določa obnašanje brskalnika. To posledično pripelje do konfliktov med različnimi ponudniki spletnih brskalnikov.

Da bi dosegli konsistentnost prikaza spletne strani v različnih brskalnikih, se

moramo pogosto zateči k popravkom (ang. hacks).

Kaj pa popravki pravzaprav so? Običajno gre manjše spremembe CSS ali HTML kode, da bi odpravili določen oblikovni problem, ki se pojavi med različnimi brskalniki.

Pojavi se vprašanje ali uporabiti popravke ali ne. Njihova uporaba je v končni fazi v nasprotju z naravo spletnih standardov, saj se zatekamo k nestandardnemu pristopu reševanja problema. Drug razlog, da se popravkom ognemo, pa je možnost njihove nezdružljivosti s prihodnjimi različicami brskalnikov. Zagovorniki popravkov pa svoje strinjanje z njihovo uporabo utemeljujejo s tem, da so spletni standardi bolj predlogi kot pravilo in da ob morebitnih spremembah brskalnikov ne popravljamo potencialno več sto spletnih dokumentov, ampak samo dokument s CSS pravili. Avtor tega diplomskega dela je mnenja, da bi bila v popolnem svetu njihova uporaba nesmiselna, v resničnem svetu pa je vsekakor priporočljiva.

Po temeljitem pregledu odstopanj, ki sem jih našel na spletu, sem ugotovil, da je nesmiselno opisati vsa odstopanja, saj sem jih našel več kot štiristo, zato bom v naslednjem poglavju opisal tista, s katerimi sem imel opravka med razvojem spletne aplikacije Register PoI in ki se mi zdijo vredni omembe.

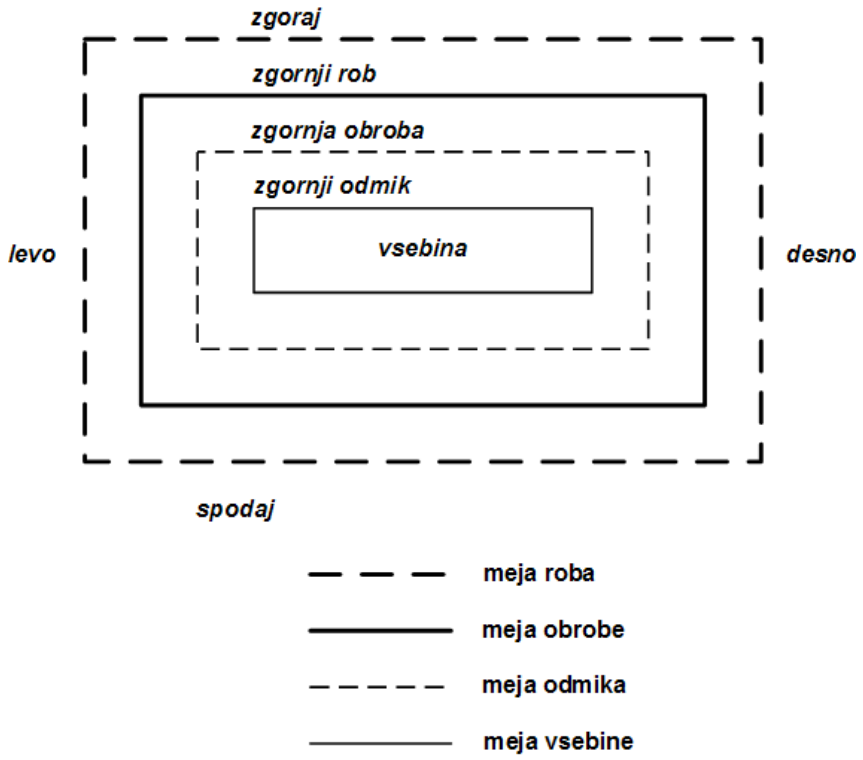
3.1 Odstopanja med različnimi brskalniki

3.1.1 Bločni model

Verjetno najbolj znana programska napaka brskalnika Internet Explorer je, da se v nekaterih primerih ne drži standarda bločnega modela. Pojavi se pri brskalnikih Internet Explorer starejših od različice 6 in pri različici 6 in 7 v quirks načinu prikazovanja elementov.

Bločni model (ang. box model) je sestavljen iz vsebine in izbirnih lastnosti: rob (ang. margin), obroba (ang. border) in odmik (ang. padding). Lastnost rob določa oddaljenost bloka od drugih blokov, lastnost obroba določa obliko roba, lastnost odmik pa določa oddaljenost vsebine bloka od roba bloka. Bločni model se uporablja, ko postavljamo elemente na spletno stran z uporabo CSS (slika 3.1).

Po standardu W3C je višina oziroma dolžina določena z dolžino vsebine bloka, brez upoštevanja dolžine lastnosti odmik, obroba in rob, brskalnik Internet



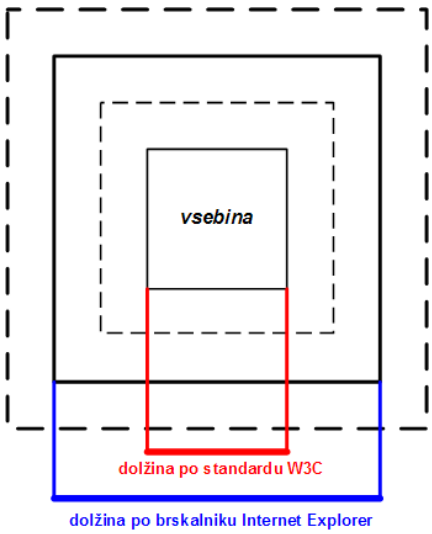
Slika 3.1: Blokovni model

Explorer pa upošteva tudi slednje (slika 3.2).

Pri brskalniku Internet Explorer različice 6 ali 7 se lahko problemu ognemo, če z ustrezno direktivo DOCTYPE (glej poglavje 2.3.2) vključimo standards način prikazovanja. Pri starejših različicah pa se moramo poslužiti popravkov. Predstavljenih je bilo že kar nekaj popravkov, vendar niso bili primerni, ker so se zaradi njih kazale napake v drugih brskalnikih. Nato je bil predstavljen popravek imenovan Tan hack [16], ki se obnese dobro, vendar ne deluje več v različici 7 brskalnika Internet Explorer, kar nas pripelje do mrtve točke glede združljivosti z različico 5 brskalnika Internet Explorer. V tem primeru se je torej priporočljiveje ogniti popravku in uporabiti *standards* način prikazovanja.

3.1.2 Uhajanje besedila iz bloka

Naslednji primer se nanaša na vse različice brskalnika Internet Explorer, razen na različico 7 v standards načinu. Gre za primer kjer brskalnik samodejno



Slika 3.2: Razlika pri obravnavanju dolžine oziroma višine blokovnega elementa med standardom W3C in brskalnikom Internet Explorer

poveča širino bloka, da bi preprečil uhajanje besedila. Spodnja slika prikazuje primer takega uhajanja.

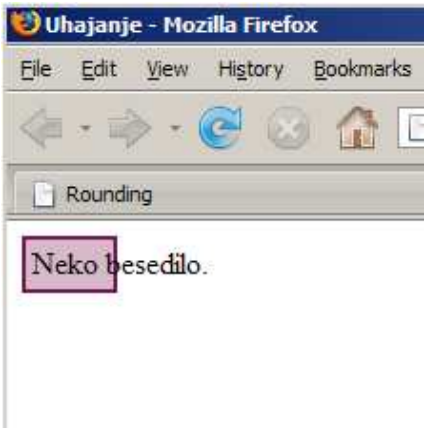
Blok na sliki 3.3 ima dodeljena naslednja CSS pravila:

```
.box {
  width: 40px;
  border: 2px solid #781351;
  padding: 3px;
  background: #d7b9c9;
  white-space: nowrap
}
```

Razlog za tako obnašanje brskalnika Internet Explorer je v tem, da obravnava deklaracijo width kot min-width, se pravi da širina ni fiksna ampak najmanjša dovoljena. Ravno tako je pri višini bloka.

Da bi zagotovili, da besedilo iz bloka ne bi uhajalo tudi pri drugih brskalnikih, moramo dodati naslednje CSS pravilo:

```
.box {
  width: 40px;
  position: relative;
  float: left;
```



Slika 3.3: Primer uhajanja besedila

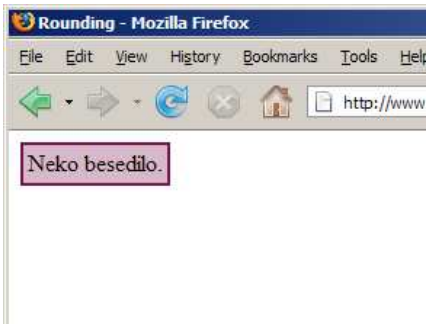
```
  border: 2px solid #781351;
  padding: 3px;
  background: #d7b9c9;
  white-space: nowrap
}

html>body .box
{
  width: auto;
  min-width: 40px;
}
```

Brskalnik Internet Explorer bo zaradi direktive *html>body* slednje pravilo ignoriral, ostali brskalniki pa ga bodo upoštevali. Na sliki 3.4 vidimo prikaz spletne strani z uporabo omenjenega popravka.

3.1.3 Širina inline elementov

Vsak HTML element je lahko določen kot *block* ali *inline*. *Block* element zasede celotno razpoložljivo širino in vrine novo vrstico pred in po elementu. Primeri *block* elementov so *div*, *p*, *h1*, *form* in *li*. *Inline* elementi zasedejo le toliko dolžine kot jo potrebujejo in ne vrinejo novih vrstic. Primeri inline elementov so *span*, *a*, *label*, *strong* in *em*.

Ena izmed karakteristik *inline* elementov je tudi, da jim ne moremo določiti



Slika 3.4: Uporaba popravka v primeru uhajanja besedila

širine. Naslednje CSS pravilo ne bi smelo delovati:

```
span {
  width: 100px;
}
```

To pravilo bo upošteval brskalnik Internet Explorer v quirks načinu prikazovanja, kjer bo imel vsak span element širino 100 slikovnih pik. V drugih brskalnikih bo dolžina span elementa dolžina števila znakov znotraj tega elementa. Rešitev je, da span element določimo kot block element.

```
span {
  width: 100px;
  display: block;
}
```

Določanje span elementa kot block element pomeni tudi, da se bo samodejno začel v novi vrstici. To lahko popravimo z dodajanjem CSS pravila float: left.

3.1.4 Napaka FOUC

Napaka FOUC[1] vseh različic brskalnika Internet Explorer povzroči, da se spletna stran za sekundo ali dve prikaže neoblikovana.

Pojavi se, če uporabimo direktivo `@import` za dodajanje CSS pravil oblikovanja spletni strani, na primer:

```
<style type="text/css">@import "styles.css";</style>
```

[1]Flash Of Unstyled Content

Napake se znebimo, če dodamo link ali script element v head element strani:

```
<script type="text/javascript" src="scripts.js"></script>
<link rel="stylesheet" href="styles.css"
type="text/css" media="print" />
```

3.1.5 (Ne)podpora oblike zapisa slik PNG

Nepodpora alfa kanala PNG oblike zapisa slik brskalnika Internet Explorer pred različico 7 je povzročila izogibanje uporabi tega slikovnega formata. Prozorni del slike je v starejših različicah tega brskalnika zamenjan s sivo, belo ali kakšno drugo barvo, odvisno od tega, v katerem slikovnem urejevalniku (ang. image editor) je bila slika narejena. Obstaja več rešitev, vsaka s svojimi prednostmi in slabostmi.

3.1.6 Content-type: text/plain

Napaka, ki je prisotna pri vseh različicah brskalnika Internet Explorer je nepravilno obravnavanje golih tekstovnih (ang. plain-text) datotek. Če imamo primer, pri katerem v element *iframe* vstavimo povezavo do gole tekstovne datoteke, bo ta s strani brskalnika Internet Explorer obravnava kot HTML datoteka. Vzemimo za primer naslednjo tekstovno datoteko (golitekst.txt):

```
Neko besedilo.

<Neko besedilo znotraj oklepajev.>

IE bo prikazal samo dva stavka.
```

ki jo vstavimo v element *iframe* spletnega dokumenta:

```
<iframe src="golitekst.txt" height="160" width="400">
</iframe>
```

Ker bo brskalnik Internet Explorer obravnaval golo tekstovno datoteko kot HTML datoteko, bo tudi stavek znotraj oklepajev obravnaval kot HTML oznako, zato ga ne bo prikazal.

Tej napaki se lahko izognemo le tako, da poskrbimo, da se med besedilom ne pojavijo HTML oznake.

3.1.7 (Ne)podpora zbirke dodatkov navigator.plugins

Z uporabo zbirke dodatkov *navigator.plugins* pri skriptnem jeziku JavaScript lahko preprosto ugotovimo, kateri dodatki (ang. plugins) so v brskalniku nameščeni, vendar ta zbirka v brskalniku Internet Explorer ni podprta, oziroma je vedno prazna, čeprav je bila dodana v JavaScript ob izidu različice 3 brskalnika Internet Explorer.

Zbirka je zelo uporabna, saj lahko na preprost način ugotovimo, če ima na primer uporabnik nameščen dodatek Flash. Na podlagi te ugotovitve lahko prikažemo Flash animacijo ali pa uporabnika preusmerimo na stran za namestitev Flash dodatka. To preprosto storimo na naslednji način:

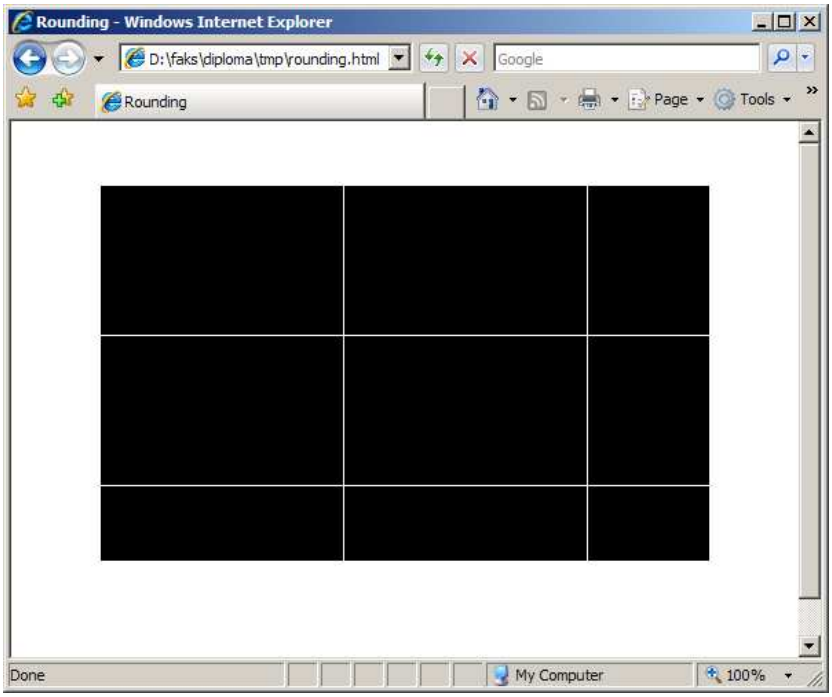
```
<script type="text/javascript">
if(navigator.plugins['Shockwave Flash']) {
  // Prikaži Flash animacijo.
}
else {
  // Preusmeri na stran za nalaganje Flash dodatka.
}
</script>
```

A kot že omenjeno, Internet Explorer tega ne podpira, zato moramo uporabiti druge, precej manj elegantne rešitve, kot je na primer *Flash Detect* [19].

3.1.8 Zaokroževanje pikslov

Ko brskalnik prikazuje elemente, pri katerih so dimenzije podane v relativnih enotah, kot na primer odstotki, mora te enote pretvoriti v dejansko število pikslov. Problem nastane, ko mora brskalnik uporabiti zaokroževanje, da izračuna potrebno število pikslov, kar pa ni nujno, da privede do pravilnega števila.

Na sliki 3.5 imamo primer matrike 5 x 5 enakomerno razporejenih *div* elementov, kar bi moralo tvoriti neprekinjen pravokotnik, vendar se zaradi zaokroževanja pojavijo vrzeli.



Slika 3.5: Pojavljanje vrteli zaradi zaokroževanja

3.1.9 Standard Data: URI

Standard Data: URI je bil uveljavljen leta 1998 in ga podpira večina brskalnikov, med brskalniki, ki tega standarda ne podpirajo, sodijo med drugim tudi vse različice brskalnika Internet Explorer.

Pri standardu Data: URI gre za način dodajana objektov v spletni dokument, ne da bi se sklicevali na nek zunanji vir. Tako lahko na primer dodamo v dokument sliko, ne da bi pri tem uporabili povezavo na datoteko. Tak način dodajana objektov v dokument je lahko videti neuporaben, vendar ga lahko izkoristimo, kadar želimo imeti v enotni datoteki celotno spletno stran. Obnese se ko na primer želimo poslati spletno stran kot eno datoteko, ne pa kot arhiv spletnega dokumenta in vseh pripadajočih slik.

Sledi primer dodajanja slike v spletni dokument:

```
<img src="... alt="Slika">
```

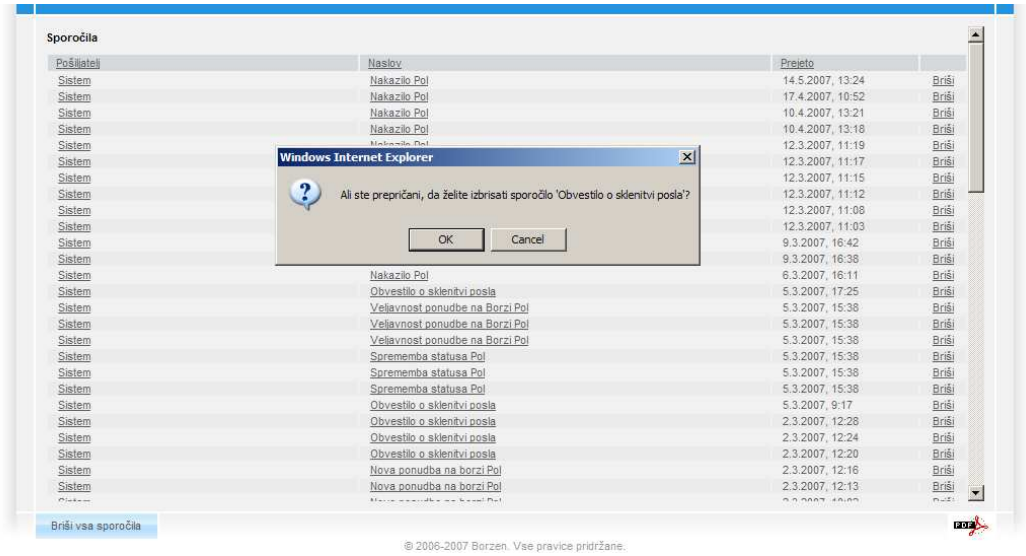
Rešitev je v tem primeru običajna povezava na datoteko izven spletnega dokumenta.

3.2 Primeri iz spletne aplikacije Register PoI

Kot že omenjeno v enem izmed prejšnjih poglavij, sem bil v okviru diplomske naloge udeležen pri izdelavi večjega informacijskega sistema Register PoI, kjer je bila moja naloga izdelati uporabniški vmesnik. V tem podpoglavju si bomo ogledali primere, kjer je bila implementacija določenega segmenta uporabniškega vmesnika težje izvedljiva ali pa celo nemogoča in je bilo zato potrebno najti alternativno rešitev. Na srečo je bil naročnik informacijskega sistema zelo prilagodljiv glede videza, tako da večjih težav nismo imeli.

3.2.1 Opozorilna okna

Ena izmed težav, pri kateri smo se morali zateči k alternativni rešitvi, so bila opozorilna okna. To so okna, ki se prikažejo, ko mora uporabnik določeno akcijo potrditi preden se ta izvede. Primer takšnega okna si lahko ogledamo na sliki 3.6.



Slika 3.6: Opozorilno okno

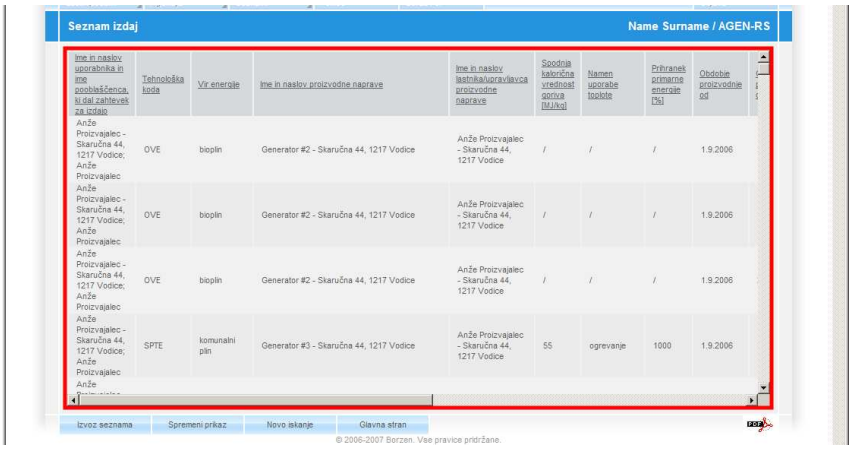
Težava je bila v tem, ker je naročnik želel imeti tudi opozorilna okna oblikovana, vendar tistih, ki so vgrajena v brskalnik ni mogoče kakorkoli spreminjati, zato smo najprej poskusili narediti nova opozorilna okna z uporabo *div* elementov in absolutnim pozicioniranjem. Izkazalo se je, da lahko uporabnik, z

nekaterimi kombinacijami tipk spreminja vsebino vnosnih polj, medtem ko je prikazano opozorilno okno, kar seveda ni prav. To hudo pomanjkljivost odpravljajo opozorilna okna, ki so vgrajena v brskalnik, zato smo morali uporabiti slednja. Primer programske kode, napisane v programskem jeziku JavaScript, ki sproži tako okno je sledeče:

```
<button onclick="return confirm('Ali ste prepičani?')">Briši
</button>
```

3.2.2 Vsebina v svojem oknu

Naročnik je želel imeti glavno vsebino, se pravi vnosna polja in izhodne podatke, prikazana znotraj posebnega okna, kot prikazuje slika 3.7.



Slika 3.7: Glavna vsebina v posebnem oknu

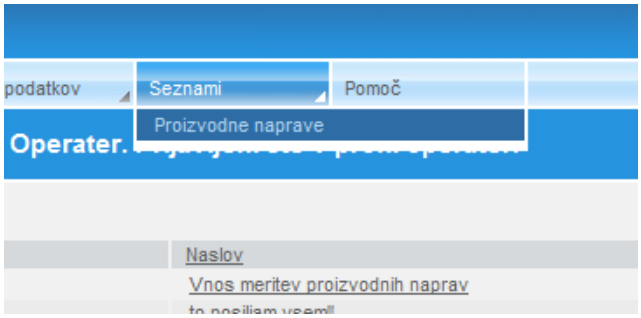
V primeru da je podatkov veliko, se morata pojaviti vertikalni in horizontalni drsnik znotraj tega okna, drsnik brskalnika pa naj bi se ne uporabljal. Želenega izgleda ni bilo enostavno doseči, saj zaradi razlik med brskalniki ni bila primerna vsaka rešitev. Rešitev je bila postavitev relativno pozicioniranega elementa *div* znotraj celice v tabeli z fiksno dolžino in širino:

```
.Content {
  position: relative;
  left: 10px;
  top: 10px;
  height: 468px;
```

```
  width: 926px;
  padding: 0px;
  overflow: auto;
}
```

3.2.3 Prevelika obroba menijev

Ena izmed težav, pri katerih nismo uspeli ugotoviti vzroka, je bila širša spodnja obroba padajočih (ang. drop-down) menijev v brskalniku Firefox, kot pa je bila določena v CSS pravilih (glej sliko 3.8). Meniji so bili izdelani z ASP.NET komponento *Menu* in bili pravilno prikazani v brskalniku Internet Explorer. Razlog za napačen prikaz padajočih menijev je po vsej verjetnosti ta, da avtorji omenjene komponente niso testirali v drugih brskalnikih kot Internet Explorer.



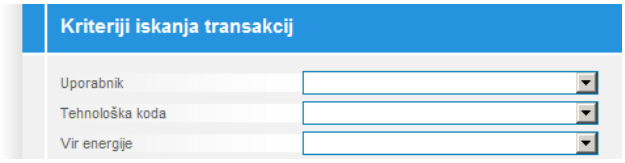
Slika 3.8: Prevelika spodnja obroba menijev v brskalniku Firefox

3.2.4 Razlike pri oblikovanju vnosnih polj

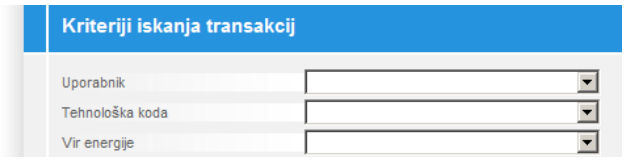
Očitna razlika med brskalnikoma Firefox in Internet Explorer je tudi prikazovanje oblikovanih padajočih vnosnih polj. Iz slik 3.9 in 3.10 je razvidno, da so polja drugače obarvana in imajo drugačno obrobo, kljub enakim CSS pravilom.

3.2.5 Dolžina fiksnih tabel

Težava, ki je povzročila mnogo preglavic je bila širina fiksnih tabel znotraj elementa *div*. Ker ustrezne rešitve, ki bi zagotovila konsistentnost izgleda



Slika 3.9: Padajoča vnosna polja v brskalniku Firefox



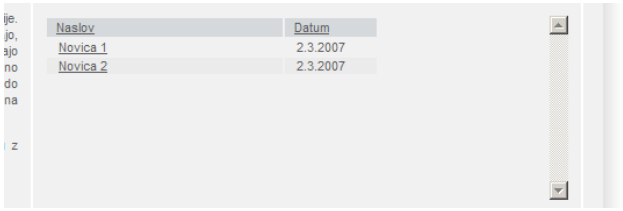
Slika 3.10: Padajoča vnosna polja v brskalniku Internet Explorer

tabel v brskalniku Firefox in Internet Explorer različice 6 in 7 ni bilo mogoče najti, smo morali dati prednost slednjemu brskalniku. Na sliki 3.11 lahko vidimo končni rezultat naslednjih pravil CSS, ki so omogočile pravilen izgled v brskalniku Internet Explorer:

```
.TableFixed {
  table-layout: fixed;
  max-width: 100%;
  text-align: left;
  border-collapse: separate;
  border-left: 0px;
}
```

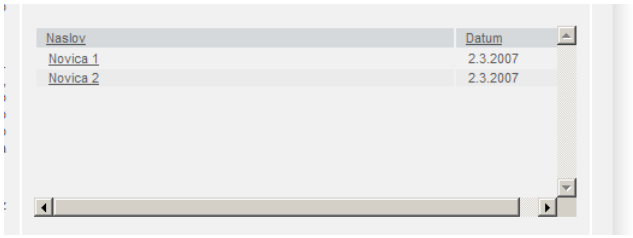
Slika 3.11: Pravilen izgled tabel

Zgornja pravila CSS omogočajo želen prikaz tabel v brskalniku Internet Explorer, vendar kot že rečeno za ceno napačnega izgleda v brskalniku Firefox, kjer tabela ni raztegnjena čez celotno razpoložljivo širino, kar lahko vidimo na sliki 3.12.



Slika 3.12: Napačna širina tabel v brskalniku Firefox

Če pravilo CSS *max-width* spremenimo v *width*, bo prikaz v brskalniku FireFox pravilen, kot ga prikazuje slika 3.11, v brskalniku Internet Explorer pa se pojavi neželeni horizontalni drsnik, ker slednji brskalnik ne upošteva vertikalnega drsnika kot dol dolžine tabele (slika 3.13).



Slika 3.13: Neželen horizontalni drsnik v brskalniku Internet Explorer

Poglavje 4

Zaključek

Lahko rečemo, da spletne aplikacije ne bodo zamenjale namiznih aplikacij, vsekakor pa postajajo vse bolj priljubljena izbira. Razlogi za njihovo so predvsem neodvisnost od strojne platforme in vseprisotnost aplikacije, kar pomeni, da lahko uporabnik do spletne aplikacije dostopa od kjerkoli v omrežju.

Najbolj pogosto uporabljena arhitektura spletne aplikacije je tronivojska arhitektura, ki jo sestavljajo nivoji uporabniškega vmesnika, poslovne logike in podatkovne baze, kar omogoča neodvisnost uporabniškega vmesnika od podatkovne baze. To pomeni, da v primeru sprememb v poslovni logiki ali podatkovni bazi ni potrebno spreminjati uporabniškega vmesnika, kar pride še posebno prav, ko se zahteve s strani naročnika aplikacije hitro dodajajo ali spreminjajo.

Razvitih je veliko spletnih tehnologij in tehnik za izdelavo spletnih aplikacij, med katerimi so najbolj razširjene ASP.NET, PHP, JSP, JavaScript in AJAX. Uveljavljenih je tudi veliko spletnih standardov, ki naj bi omogočali dolgoročno uporabnost svetovnega spleta, med katere sodijo HTML, XHTML, XML in CSS.

Izdelava uporabniškega vmesnika spletne in namizne aplikacije se v nekaterih pogledih precej razlikuje, saj imamo pri spletni aplikaciji opravka z uporabniškim vmesnikom znotraj spletnega brskalnika. Da zagotovimo delujoč uporabniški vmesnik spletne aplikacije, potrebujemo kar nekaj znanja s tega področja. Najpomembnejše načelo, ki se ga moramo držati, je strogo ločevanje vsebine in oblike spletne strani, kar lahko dosežemo z uporabo standarda CSS. Poleg tega moramo poznati še nekatere stvari, kot je na primer deklaracija

DOCTYPE, zavedati pa se moramo tudi, da se spletni brskalniki med drugim med seboj razlikujejo tudi pri obravnavanju določenih HTML in CSS pravil, čeprav obstajajo temu namenjeni spletni standardi. Razlike med spletnimi brskalniki se največkrat odražajo v drugačnem prikazu spletne strani pri enakih HTML in CSS pravilih. Da se temu ognemo, se moramo zateči k uporabi popravkov, ki so nestandarden način spreminjanja izvorne kode. Teh popravkov je več kot štiristo in se z novimi brskalniki in novimi različicami brskalnikov neprestano pojavljajo in spreminjajo.

Slike

Literatura

[1] (2007) Trinivojska arhitektura. Dostopno na:
http://www.sei.cmu.edu/str/descriptions/threetier.html

[2] (2007) Dvonivojska arhitektura. Dostopno na:
http://www.sei.cmu.edu/str/descriptions/twotier.html

[3] (2007) Spletna aplikacija. Dostopno na:
http://en.wikipedia.org/wiki/Rich_internet_application

[4] (2007) Protokol HTTPS. Dostopno na:
http://www.ietf.org/rfc/rfc2818.txt

[5] (2007) Začetki spletnih standardov. Dostopno na:
http://juicystudio.com/article/journey-through-accessibility.php

[6] (2007) Web Standards Project. Dostopno na:
http://www.webstandards.org

[7] (2007) Web Standards Group. Dostopno na:
http://webstandardsgroup.org

[8] (2007) Specifikacije jezika HTML 4.01. Dostopno na:
http://www.w3.org/TR/html4

[9] (2007) Specifikacije jezika XML. Dostopno na:
http://www.w3.org/XML

[10] (2007) Specifikacije jezika CSS. Dostopno na:
http://www.w3.org/TR/CSS2/

[11] (2007) Specifikacije jezika XHTML. Dostopno na:
http://www.w3.org/TR/xhtml1

[12] (2007) Specifikacije jezika ECMAScript. Dostopno na:
http://www.ecma-international.org/publications/standards/Ecma-262.htm

[13] (2007) Flash. Dostopno na:
http://en.wikipedia.org/wiki/Adobe_Flash

[14] (2007) Postavitveni pogon. Dostopno na:
http://en.wikipedia.org/wiki/Layout_engine

[15] (2007) Direktiva Document Type Definition. Dostopno na:
http://en.wikipedia.org/wiki/Document_Type_Definition

[16] (2007) Popravek Tan hack. Dostopno na:
http://www.communitymx.com/content/article.cfm?cid=E0989953B6F20B41

[17] (2007) Modularizacija jezika XHTML. Dostopno na:
http://www.w3.org/MarkUp/modularization

[18] (2007) W3C validator. Dostopno na:
http://validator.w3.org

[19] (2007) Rešitev za detekcijo Flash dodatka v brskalniku Internet Explorer. Dostopno na:
http://www.quirksmode.org/js/flash.html

[20] (2007) Tabela direktiv DOCTYPE. Dostopno na:
http://en.wikipedia.org/wiki/Quirks_mode#Comparison_of_document_types

Stvarno kazalo

Izjava

Izjavljam, da sem diplomsko nalogo izdelal samostojno pod vodstvom mentorja doc. dr. Boštjana Slivnika. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Ljubljana, 22. 10. 2007 Primož Hadalin