UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

DIPLOMSKO DELO

PREDNOSTI IN SLABOSTI RAZVOJNE TEHNOLOGIJE AJAX V ASP.NET

Ljubljana, november 2009 ALEŠ PAŠKULIN

IZJAVA

Študent Aleš Paškulin izjavljam, da sem avtor diplomskega dela, ki sem ga napisal pod mentorstvom dr. Tomaža Turka, in dovolim njegovo objavo na spletnih straneh fakultete.

V Ljubljani, 10. 11. 2009 Podpis: _____

KAZALO

UVOD **1**

1 AJAX **2**

1.1 TEHNOLOGIJA AJAX 3
1.1.1 XML *3*
1.1.2 HTML *3*
1.1.3 XHTML *3*
1.1.4 CSS *4*
1.1.5 DOM *4*
1.1.6 Objekt XMLHttpRequest *5*
1.1.7 XSLT *5*
1.1.8 JavaScript *6*
1.2 AJAX IN KLASIČNI MODEL SPLETNIH APLIKACIJ 6
1.2.1 Klasični model *6*
1.2.2 AJAX model *7*
1.3 ARHITEKTURA ASP.NET AJAX 8
1.3.1 ASP.NET *8*
1.3.2 ASP.NET AJAX *8*
1.3.3 Osnovni gradniki ASP.NET AJAX *8*
1.3.4 Dodatni gradniki ASP.NET AJAX *9*
1.3.5 Fleksibilnost gradnikov ASP.NET AJAX *13*

2 ASP.NET AJAX Z VIDIKA RAZVIJALCA **18**

2.1 FAZA NAČRTOVANJA 18
2.1.1 Standardni način razvijanja *18*
2.1.2 Razvijanje s tehnologijo AJAX *19*
2.2 FAZA IZVEDBE 20
2.2.1 Standardni način razvijanja *20*
2.2.2 Razvijanje s tehnologijo AJAX *20*
2.2.3 Primerjava obeh tehnik ob praktičnem primeru *20*

3 ASP.NET AJAX Z VIDIKA DRUGIH SUBJEKTOV **29**

3.1 ASP.NET AJAX Z VIDIKA UPORABNIKA 29
3.1.1 Spletne aplikacije v okolju ASP.NET *30*
3.1.2 Spletne aplikacije v okolju ASP.NET z uporabo tehnologije AJAX *30*
3.2 ASP.NET AJAX Z VIDIKA NAROČNIKA 30
3.3 ASP.NET AJAX Z VIDIKA MENEDŽMENTA RAZVOJNIH PROJEKTOV PONUDNIKA 32
3.3.1 Pozitivni vplivi na zaposlene (razvijalce) *32*
3.3.2 Pozitivni vplivi na stroške *33*

4 SLABOSTI TEHNOLOGIJE AJAX IN ASP.NET AJAX **34**

4.1 SLABOSTI, KI JIH PRINAŠA AJAX NAČINA DELOVANJA 34
4.1.1 Primer 1 ..... *36*
4.1.2 Primer 2 ..... *37*
4.2 SLABOSTI PAKETOV ASP.NET AJAX IN AJAX CONTROL TOOLKIT 37
4.2.1 Primer 1 ..... *38*
4.2.2 Primer 2 ..... *38*

5 PRIMERI UPORABE TEHNOLOGIJE AJAX V SPLETU **39**

5.1 GOOGLE MAPS 39
5.2 GMAIL 40
5.3 NETFLIX 40
5.4 A9 41
5.5 PANIK GOODS 41
5.6 AJAXTRANS 42
5.7 PROTOPAGE 42
5.8 INSTANT DOMAIN SEARCH 43
5.9 FLICKR 44
5.10 DRUGI PRIMERI 45

6 ALTERNATIVE PAKETU ASP.NET AJAX 45

6.1 PLATFORME, TEMELJEČE NA STREŽNIŠKI KODI 45
6.2 PLATFORME, TEMELJEČE NA ODJEMALČEVI KODI 47

SKLEP 49

LITERATURA IN VIRI 51

KAZALO SLIK

SLIKA 1: PRIMER OBLIKE DOKUMENTA XHTML 4
SLIKA 2: PRIMER OBLIKE DOKUMENTA CSS 4
SLIKA 3: PRIKAZ STRUKTURE DOM TABELE V DOKUMENTU HTML 5
SLIKA 4: PRIMER JAVASCRIPT FUNKCIJE V OBLIKI ODJEMALČEVE SKRIPTE 6
SLIKA 5: PRIMERJAVA KLASIČNEGA DELOVANJA SPLETNIH APLIKACIJ IN NAČINA AJAX 7
SLIKA 6: PRIKAZ DELOVANJA GRADNIKA CALENDAR 10
SLIKA 7: PRIKAZ DELOVANJA GRADNIKA COLLAPSIBLEPANEL 10
SLIKA 8: PRIKAZ DELOVANJA GRADNIKA Z UPORABO ČISTEGA OPOZORILA JAVASCRIPT 11
SLIKA 9: PRIKAZ DELOVANJA GRADNIKA Z UPORABO GRADNIKA POPUPCONTROL 11
SLIKA 10: PRIKAZ DELOVANJA GRADNIKA LISTSEARCHEXTENDER 12
SLIKA 11: PRIKAZ DELOVANJA GRADNIKA POPUPCONTROL 13
SLIKA 12: PRIKAZ NOVIH LASTNOSTI, UPORABLJENIH NA STREŽNIŠKI STRANI 14
SLIKA 13: PRIKAZ NOVIH LASTNOSTI, UPORABLJENIH NA ODJEMALČEVI STRANI 15
SLIKA 14: DEFINIRANJE NOVIH SPREMENLJIVK ZA DOSTOP DO NOVIH LASTNOSTI 15
SLIKA 15: PREVERJANJE TRENUTNEGA STANJA GRADNIKA 15
SLIKA 16: SPREMEMBA ATRIBUTA CLASS GRADNIKU PANEL, KO JE VIDNA SAMO GLAVA 15
SLIKA 17: SPREMEMBA ATRIBUTA CLASS GRADNIKU PANEL, KO JE VIDNA VSEBINA 16
SLIKA 18: UPORABA DOGRAJENEGA GRADNIKA COLLAPSIBLEPANEL 16
SLIKA 19: RAZRED CSS DOGRAJENEGA GRADNIKA 17
SLIKA 20: PRIMER UPORABE SPREMENJENEGA GRADNIKA AJAX COLLAPSIBLEPANEL 17
SLIKA 21: PRIKAZ OPOZORILA Z DVEMA MOŽNIMA AKCIJAMA IN S PREKLICEM 21

SLIKA 22: PRIKAZ IZBRANEGA GUMBA TEST Z UPORABO TIPKE TAB NA TIPKOVNICI 24
SLIKA 23: PRIKAZ SPROŽITVE ONEMOGOČENEGA GUMBA 24
SLIKA 24: PRIKAZ PRIPENJANJA FUNKCIJE JAVASCRIPT POSAMEZNEMU GRADNIKU 26
SLIKA 25: PRIKAZ SPREMEMBE ATRIBUTA CSS S STREŽNIŠKO KODO 26
SLIKA 26: PRIMER VKLJUČITVE GRADNIKA SCRIPTMANAGER V KODO HTML 27
SLIKA 27: PRIKAZ SKRIVANJA OPOZORILA, PRIKAZANEGA V NAČINU AJAX 28
SLIKA 28: PRIKAZ SPREMEMBE, IZVEDENE S TEHNOLOGIJO AJAX 36
SLIKA 29: PRIKAZ OBVEŠČANJA UPORABNIKA OB KOMUNIKACIJI S STREŽNIKOM 37
SLIKA 30: PRIMER GRADNIKA FILEUPLOAD 38
SLIKA 31: GOOGLOV PRIKAZOVALNIK ZEMLJEVIDOV 40
SLIKA 32: NETFLIXOVO PONUJANJE DODATNIH INFORMACIJ UPORABNIKU 41
SLIKA 33: PRIKAZ MOŽNOSTI POVLECI IN SPUSTI V SPLETNI TRGOVINI 42
SLIKA 34: PROTOPAGEEV OSEBNI SPLETNI PORTAL 43
SLIKA 35: DELOVANJE SPLETNE APLIKACIJE INSTANT DOMAIN SEARCH 44
SLIKA 36: PRIKAZ OSVEŽEVANJA VSEBINE NA FLICKRJEVI STRANI 44
SLIKA 37: PRIMER DELOVANJA POVLECI IN SPUSTI Z GRADNIKI GAIA AJAX WIDGETS 46
SLIKA 38: PRIMER PREPROSTEGA PROGRAMIRANJA V PLATFORMI PROTOTYPE 48

KAZALO TABEL

TABELA 1: OCENA NALOG, IZVEDENIH S STANDARDNIMI METODAMI 24
TABELA 2: OCENA NALOG, IZVEDENIH S STANDARDNIMI METODAMI IN JAVASCRIPT SKRIPTAMI 27
TABELA 3: OCENA NALOG, IZVEDENIH Z METODO AJAX 29

Uvod

V svetovnem spletu je vedno več aplikacij, ki uporabniku ponujajo določene izdelke oziroma storitve. Z večanjem števila takih aplikacij se povečuje tudi konkurenca med ponudniki teh izdelkov oziroma storitev. Za dosego dolgoročnega uspeha mora podjetje poskrbeti za uspešno razlikovanje svoje ponudbe. To vključuje tako razlikovanje ponujenih izdelkov oziroma storitev kot tudi razlikovanje ponujene spletne aplikacije. Uporabnik namreč veliko raje uporablja spletno aplikacijo, ki deluje na preprost in njemu znan način ter hkrati zagotavlja veliko možnosti, tako da ga pri delu ne omejuje.

Cilj podjetja je tako s spletno aplikacijo uporabniku ponuditi najboljšo možno uporabnikovo izkušnjo. Za dosego takega cilja pa je treba v spletno aplikacijo vložiti veliko sredstev, saj njeno programiranje zahteva veliko delovnih ur. Za marsikatero podjetje pa bi bili stroški in termini izdelave take spletne aplikacije nesprejemljivi.

V diplomskem delu je razložen pomen tehnologije ASP.NET AJAX pri izdelavi sodobnih spletnih aplikacij. Prikazana je raven učinkovitosti, ki jo njegova uporaba omogoča, analizirani pa so tudi njegovi negativni vplivi na razvoj spletnih aplikacij.

Da bi razvojno tehnologijo AJAX, specifično za okolje ASP.NET, bolje razumeli, je treba najprej opredeliti pojem AJAX. V prvem poglavju je tako navedena splošna definicija, predstavljene pa so tudi struktura in najpomembnejše razlike glede na klasični model. Sledi podrobnejša analiza tehnologije AJAX, specifične za okolje ASP.NET (v nadaljevanju ASP.NET AJAX).

V naslednjih poglavjih so predstavljene prednosti uporabe tehnologije ASP.NET AJAX za posamezne subjekte, ki sodelujejo v procesu nastanka nove spletne aplikacije. Natančneje so analizirane prednosti uporabe tehnologije ASP.NET AJAX za razvijalce, uporabnike, naročnike in menedžment razvojnih projektov ponudnika spletne aplikacije. V teh poglavjih so določene prednosti predstavljene tudi ob konkretnih primerih.

V četrtem poglavju so predstavljene slabosti, s katerimi se srečamo pri uporabi tehnologije ASP.NET AJAX. Najprej so predstavljene splošne slabosti, ki so značilne za spletne aplikacije AJAX, neodvisno od okolja, v katerem se izvajajo. Sledi predstavitev povsem specifičnih slabosti, ki so značilne zgolj za tehnologijo ASP.NET AJAX. Za lažje razumevanje je predstavitev slabosti podkrepljena s konkretnimi primeri, in sicer kar z javno dostopnimi spletnimi aplikacijami.

Pred sklepnim delom je predstavljena še uporabnost tehnologije AJAX v primeru spletnih aplikacij, dostopnih v svetovnem spletu. Te dokazujejo, da je uporaba tehnologije AJAX že zdaj pogostejša, kot bi pričakovali.

1 AJAX

AJAX je kratica, ki opisuje delovanje spletnih aplikacij, oziroma natančneje, način njihove komunikacije s strežnikom. Krajše lahko AJAX opredelimo kot asinhrono komunikacijo s strežnikom na podlagi JavaScript skripte in XML (angl. **A**synchronous **J**avaScript **A**nd **X**ML). Za boljše razumevanje tehnologije AJAX je treba preučiti njegovo temeljno sestavo.

Da aplikacija lahko deluje na način AJAX, mora biti na voljo več različnih tehnologij, ki se med seboj prepletajo in dopolnjujejo. AJAX tako sestavljajo naslednje tehnologije (Garrett, 2006):

- HTML oz. XHTML in CSS za predstavitev oblike strani;
- DOM-standard za dostop do elementov na strani;
- objekt XMLHttpRequest za asinhrono komunikacijo s strežnikom;
- XML in XSLT za izmenjavo in manipulacijo s podatki;
- JavaScript skripte, ki omogočajo povezavo vseh komponent v celoto.

Vse navedene tehnologije so poznavalcem znane že kar nekaj časa, tako da sestava AJAX-a ne daje prave predstave o njegovi dejanski vrednosti. Bistvo AJAX-a je v tem, da so se te tehnologije povezale v nekakšno logično celoto in razvijalcem delo precej poenostavile. Skoraj vse, kar bi na primer lahko naredili z novimi gradniki tehnologije AJAX, lahko namreč naredimo tudi s standardnimi pristopi, to je z uporabo enega izmed standardnih razvijalskih ogrodij za razvoj spletnih aplikacij (ASP.NET, PHP …), v katero bi vključili kodo na odjemalčevi strani. Pri takem pristopu pa bi bili stroški dela, za isti rezultat, precej višji.

Prednosti, ki jih ponuja AJAX s svojimi gradniki, so povzeti v naslednjih točkah:

- objekti, ki jih lahko večkrat uporabimo;
- konsistentne rešitve;
- testirane rešitve;
- združljive rešitve;
- hitre rešitve.

Poudarek je predvsem na razmerju vloženega časa in drugih sredstev ter dobljenega rezultata, ki je pri uporabi tehnologije AJAX neprimerljivo boljši.

Razlog, zakaj se AJAX ni pojavil že prej, pa je objekt XMLHttpRequest (glejte razdelek 1.1.6), natančneje njegova standardizacija. Za delovanje AJAX-a je objekt XMLHttpRequest ključnega pomena. Objekt na odjemalčevi strani omogoča komunikacijo s strežnikom in posledično osveževanje strani brez ponovnega nalaganja. Šele ko je objekt

začelo podpirati vedno več spletnih brskalnikov, je njegova uporaba postala mogoča tudi v resnejših spletnih aplikacijah. Dandanes je objekt XMLHttpRequest že de facto standard, saj ga podpira kar 90 odstotkov vseh spletnih brskalnikov. Razcvet tehnologije AJAX je bil tako pogojen ravno s procesom standardizacije objekta XMLHttpRequest (Garrett, 2006).

1.1 Tehnologija AJAX

1.1.1 XML

XML je kratica za Extensible Markup Language, tj. razširljiv označevalni jezik. Gre za univerzalno obliko zapisa za strukturirane dokumente in podatke v spletu ter enega izmed najpogostejših standardov za zapis podatkov, ki jih želimo prenašati po internetu brez podatkov o spletnem oblikovanju (Pahor, 2002, str. 664).

XML spremlja mnogo vidikov računalništva, še posebej na področju medsebojne komunikacije aplikacij in strežnikov. Jezik je razširljiv, saj si lahko sami izmislimo imena značk (angl. *tag*). Zelo uporaben je za komunikacije, saj je zgradba zelo preprosta in pregledna. Primer dokumenta, ki temelji na XML, je XHTML.

1.1.2 HTML

HTML je kratica za Hypertext Markup Language, ki je standardni jezik za oblikovanje spletnih sestavkov na način vpisovanja posebnih značk. Tolmač za HTML je vgrajen v spletne brskalnike. Ti pri nalaganju dokumenta značke HTML obdelajo in vsebino dokumenta prikažejo kot spletno stran. Tolmač omogoča opis besedila hkrati z vključenimi slikami, obrazci za vnos podatkov in s tabelami. V besedilo lahko vrinemo povezave, znotraj besedila pa tudi povezave z drugimi dokumenti v svetovnem spletu. HTML dolguje priljubljenost prav razvoju slikovnega spleta (Pahor, 2002, str. 157).

Zaradi njegove preprostosti pa lahko HTML pišemo v vsakem urejevalniku besedil in ga lahko kombiniramo tudi z drugimi programskimi jeziki.

1.1.3 XHTML

XHTML je kratica za Extensible HyperText Markup Language. Je označevalni jezik, ki je novejša oziroma posodobljena različica jezika HTML. Najpomembnejša novost jezika je usklajenost s sintakso XML (glejte razdelek 1.1.1). Jezik HTML med nastajanjem ni bil zasnovan za današnji sodobni internet, saj je bil namenjen predvsem izmenjavi podatkov in dokumentov med znanstveniki. Zato so pravila za pisanje v jeziku HTML precej ohlapna in dopuščajo veliko nepravilnosti, zaradi česar se v današnjih brskalnikih pojavlja veliko težav pri interpretiranju takih dokumentov HTML. Nasprotno pa so za jezik XHTML, prav zaradi usklajenosti s sintakso XML, značilna veliko strožja pravila pisanja dokumentov. Najpomembnejše značilnosti dokumenta XHTML so naslednje:

- element XHTML mora biti pravilno hierarhično razvrščen;

- element XHTML mora biti vedno zaključen (angl. *closed*);
- elementi XHTML morajo biti pisani z malimi tiskanimi črkami;
- dokument XHTML mora vsebovati en glavni element (angl. *root element*).

Slika 1: Primer oblike dokumenta XHTML

```
<div>
    <asp:LinkButton ID="lnkbtnPopup" runat="server" Text="Prikaži opozorilo"
        onclick="lnkbtnPopup_Click">
    </asp:LinkButton>
    <asp:Button ID="btnTest" runat="server" Text="Test" OnClientClick="alert('Test')" />
</div>

<div id="divOpozorilo" runat="server" Visible="false">
    <asp:Panel ID="pnlSivina" runat="server" CssClass="sivina"></asp:Panel>
    <asp:Panel ID="pnlOpozorilo" runat="server" CssClass="opozorilo" >
        opozorilo...
        <br /><br />
        <asp:Button ID="btnAkcija1" runat="server" Text="Akcija 1"
            onclick="btnAkcija1_Click" />
        <asp:Button ID="btnAkcija2" runat="server" Text="Akcija 2"
            onclick="btnAkcija2_Click" />
        <asp:Button ID="btnPreklici" runat="server" Text="Prekliči"
            onclick="btnPreklici_Click" />
    </asp:Panel>
</div>
```

1.1.4 CSS

CSS je kratica za Cascading Style Sheets in predstavlja predloge, ki določajo videz spletnih strani. Z njimi določamo pisavo, velikosti črk in preostalo vizualno predstavitev spletne strani. HTML naj bi tako predstavljal semantično strukturo in smiselno hierarhijo dokumenta, CSS pa predstavitveno vlogo. CSS podpirajo vsi novejši spletni brskalniki (Internet Explorer, FireFox, Mozilla, Opera …). Spletni brskalniki, ki standarda CSS ne podpirajo, dokument HTML prikažejo brez oblikovanja (Bos, 2009).

Slika 2: Primer oblike dokumenta CSS

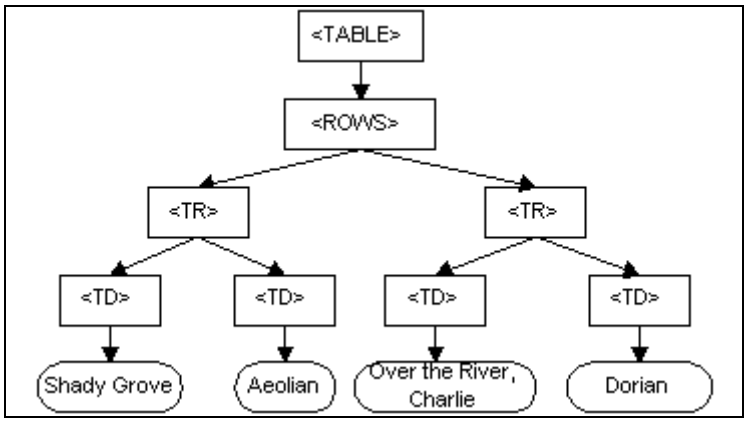
```
h1 {
    font-size:16px;
    font-weight: bold;
    padding-top: 26px;
    padding-bottom:5px;
    margin: 0px;
    color:#AB1B1B;
    text-align: center;
}

h2 {
    font-size:14px;
    font-weight: bold;
    color: #3A688C;
    font-style: normal;
    margin: 0px;
    padding: 10px;
}
```

1.1.5 DOM

DOM je kratica za Document Object Model in pomeni jezikovno nevtralen vmesnik, ki omogoča dinamičen dostop do strukturiranih objektov XML. Tako dobljenim objektom lahko spreminjamo vsebino, strukturo in videz. Najpomembnejša lastnost vmesnika DOM

je, da lahko vse spremembe znova vgradimo v prvoten dokument in jih ponovno prikažemo. Tako lahko na primer z JavaScript funkcijo tabeli spremenimo ozadje, ne da bi pri tem morali osvežiti celotno stran (Robie, 2004).

Slika 3: Prikaz strukture DOM tabele v dokumentu HTML



Vir: What is the Document Object Model?, 2004.

1.1.6 Objekt XMLHttpRequest

Gre za objekt v programski kodi, ki na odjemalčevi strani omogoča prenos podatkov med odjemalcem in strežnikom. Prvič se je pojavil že leta 1999 z izidom Internet Explorerja 5. V tistem obdobju je bila podpora zanj izjemno majhna, tako da ga je podpirala le peščica brskalnikov. Dokler določena tehnologija oziroma metoda ne postane nekakšen neuraden standard, je ne moremo uporabiti za razvoj resnejše aplikacijske rešitve. Ravno to je bila težava pri uvedbi tehnologije AJAX. Šele pred približno dvema letoma je objekt XMLHttpRequest začenjal pridobivati na popularnosti, zato je bil vključen v delovanje vedno več brskalnikov. To je privedlo do današnjega stanja, ko ga podpira kar 90 odstotkov vseh brskalnikov. Če upoštevamo samo glavne spletne brskalnike (Internet Explorer, FireFox), pa je podpora kar stoodstotna. Taka podpora pa že pomeni, da lahko določeno tehnologijo uporabimo pri razvoju resnih aplikacijskih rešitev (McClure, Scott, Glavich & Shoemaker, 2006).

Vse večja podpora objekta XMLHttpRequest v brskalnikih je razlog nenadnega pojava tehnologije AJAX. Vsa druga tehnologija, ki jo uporablja, je namreč že obstajala in tudi imela zadostno podporo. Težave je povzročala slaba podpora objekta XMLHttpRequest, ki pa je za delovanje na način AJAX ključnega pomena. Slaba podpora objekta XMLHttpRequest je tudi razlog za to, da razvijalci pri Microsoftu delovanja tehnologije AJAX niso mogli vključiti že v samo okolje ASP.NET 2.0.

1.1.7 XSLT

XSLT (Extensible Stylesheet Language Transformation) je jezik za pretvarjanje dokumentov XML v druge dokumente XML in je eden izmed sestavnih gradnikov za XSL. Ta predstavlja skupek priporočil za prikaz dokumentov XML (glejte razdelek 1.1.1). V

osnovi je zelo podoben CSS, le da ima nekoliko drugačno sintakso. XSL tako sestavljajo (Quin, 2009):

- **XSLT**, jezik za transformacijo dokumentov XML;
- xPath, jezik za dostop do elementov dokumenta XML;
- XSL-FO, slovar XML za specifikacijo semantike dokumenta.

XSLT je tako del XSL in določa način pretvarjanja med različnimi dokumenti XML, ki uporabljajo različne slovarje XML (HTML, XSL-FO …).

1.1.8 JavaScript

Gre za skriptni programski jezik, sintaktično podoben Javi. Uporaben je na različnih področjih, od nastavitev administrativnih opravil v okolju Windows do manipulacij datotek PDF. Prvotni in glavni namen jezika pa je dinamično spreminjanje vsebine HTML. Spletni brskalniki imajo v njihovem delovanju vgrajen modul za izvajanje skript, napisanih v jeziku JavaScript (Slika 4). S takimi skriptami lahko na podlagi vmesnika DOM (glejte razdelek 1.1.5) manipuliramo prikazane objekte na internetni strani, kar omogoča razvoj naprednih uporabniških vmesnikov in dinamičnih internetnih strani (Heilmann, 2006).

Slika 4: Primer JavaScript funkcije v obliki odjemalčeve skripte

```
function MaxLength_KeyUp(source,maxlength,e)
{
    if (source.value.length > maxlength)
    {
        source.value = source.value.substr(0,maxlength);
    }
}
```

V diplomskem delu sta pogosto uporabljena izraza **JavaScript skripta** in **JavaScript funkcija**, s katerima je vedno označena skripta ali pa funkcija znotraj nje, ki:

- je napisana v jeziku JavaScript;
- se izvaja na odjemalčevi strani;
- omogoča manipulacijo vsebine HTML.

1.2 AJAX in klasični model spletnih aplikacij

1.2.1 Klasični model

Klasični model predstavlja delovanje standardnih aplikacij oziroma aplikacij, ki ne temeljijo na tehnologiji AJAX. Zanje je značilno delovanje po načelu pošiljanja ukazov na strežnik v obliki zahtevkov HTML. To se zgodi po vsakem kliku določenega gumba oziroma drugega elementa, za katerim je določena strežniška koda. Ko zahtevek HTML prispe do strežnika, se tam izvede pripadajoča strežniška koda in celotna stran se v obliki HTML in dokumenta CSS pošlje nazaj k odjemalcu. Na odjemalčevi strani se tako celotna

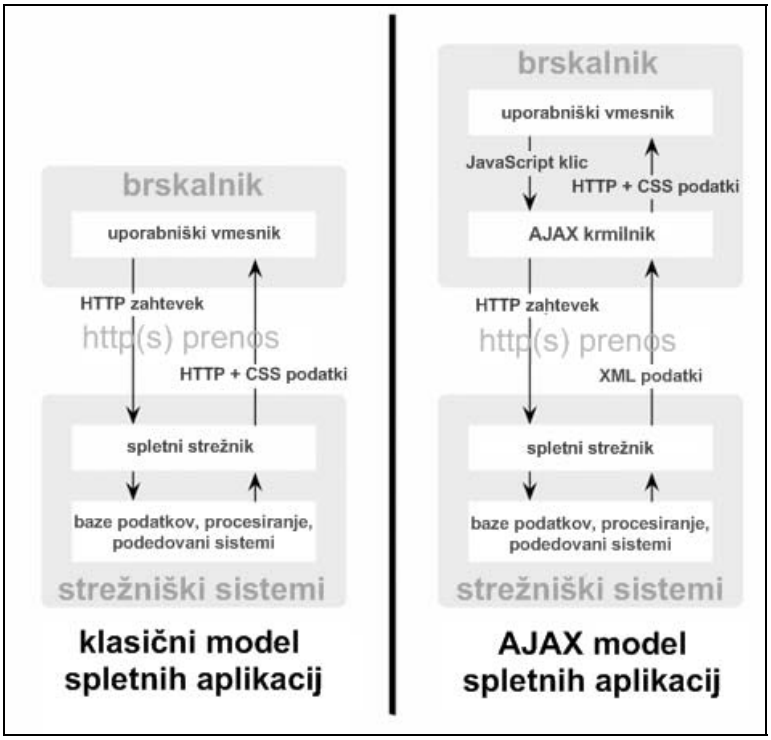
stran, glede na novi dokument HTML in CSS, ponovno naloži. Tako delovanje je za uporabnika neprivlačno in moteče, saj se ob vsaki njegovi akciji celotna stran ponovno naloži. Vsako ponovno nalaganje pa se prikaže kot utripanje zaslonske slike (Garrett, 2006).

1.2.2 AJAX model

AJAX model se od klasičnega razlikuje po načinu komuniciranja s strežnikom. Iz spodnje slike (Slika 5) je razvidno, da se pri modelu AJAX med strežnikom in odjemalcem izmenjujejo zgolj podatki v obliki datotek XML. Za pretvorbo prejetih podatkov XML v obliko za prikazovanje v spletnih brskalnikih, torej HTML in CSS, pa se poskrbi na odjemalčevi strani. Ključnega pomena je dejstvo, da se prikažejo le nastale spremembe v dokumentih HTML in CSS, s čimer se izognemo ponovnemu nalaganju celotne strani (Garrett, 2006).

Tudi komunikacija v nasprotni smeri, torej pretok podatkov od odjemalca na strežnik, se razlikuje od klasičnega modela. Pri klasičnem modelu se vedno že na začetku sproži zahtevek HTTP, medtem ko je pri modelu AJAX to nekoliko drugače. Ob akciji uporabnika se najprej izvedejo JavaScript skripte, ki šele pozneje, na podlagi ogrodja AJAX, sprožijo zahtevek HTTP (Garrett, 2006).

Slika 5: Primerjava klasičnega delovanja spletnih aplikacij in načina AJAX



Vir: A new approach to web applications, 2006.

1.3 Arhitektura ASP.NET AJAX

AJAX model in njegov način delovanja lahko vgradimo v različna razvijalska okolja. Najbolj znana sta Microsoftov ASP.NET in njegova najpopularnejša alternativa PHP. V diplomskem delu je pozornost namenjena zgolj Microsoftovemu okolju ASP.NET, kot najbolj razširjenemu in uveljavljenemu med spletnimi aplikacijami v svetovnem spletu.

1.3.1 ASP.NET

ASP.NET je Microsoftovo okolje za razvoj interaktivnih spletnih aplikacij, ki temeljijo na strežniških kodah. Gre za enega izmed standardnih načinov razvijanja sodobnih spletnih aplikacij. V diplomskem delu je predstavljeno okolje ASP.NET 2.0, ki je prvo omogočalo vključitev delovanja načina AJAX na podlagi dodatnega paketa, imenovanega ASP.NET AJAX. Danes je na voljo že novejša različica ogrodja ASP.NET, in sicer ASP.NET 3.5. V najnovejši različici je funkcionalnost paketa ASP.NET AJAX že vključena v ogrodje in tako ni potrebe po posebni namestitvi. Princip delovanja načina AJAX v okolju ASP.NET pa ostaja nespremenjen.

1.3.2 ASP.NET AJAX

Paket ASP.NET AJAX omogoča vgradnjo tehnologije AJAX v okolje ASP.NET 2.0 (v nadaljevanju ASP.NET). ASP.NET AJAX je sestavljen iz knjižnic z odjemalčevimi skriptami (angl. *client-script libraries*) in strežniških komponent, ki se integrirajo v okolje ASP.NET. Po namestitvi paketa ASP.NET AJAX so na voljo osnovni gradniki, s katerimi lahko gradnike ASP.NET spremenimo v gradnike AJAX. Osnovnemu paketu pa lahko dodamo nabor dodatnih gradnikov AJAX (angl. *AJAX Control Toolkit*), ki temeljijo na tehnologiji AJAX in omogočajo še boljši izkoristek nove tehnologije.

1.3.3 Osnovni gradniki ASP.NET AJAX

Gradniki ASP.NET AJAX so sestavljeni iz strežniške in odjemalčeve kode. Obe kodi ob vsaki interakciji uporabnika med seboj komunicirata, kar privede do delovanja v tipičnem AJAX načinu. Po namestitvi paketa ASP.NET AJAX so na voljo štirje osnovni strežniški gradniki:

- ScriptManager,
- UpdatePanel,
- UpdateProgressBar,
- Timer.

1.3.3.1 ScriptManager

ScriptManager zagotavlja delovanje JavaScript skript na odjemalčevi strani, torej delno spreminjanje prikazane vsebine, lokalizacijo uporabnika in izvajanje JavaScript skript po meri.

Gradnik nima svojega prikaza, tako da na spletni strani ni viden. Uporaba gradnika pa je obvezna na vsaki strani, ki jo želimo nadgraditi s katerim koli gradnikom AJAX.

1.3.3.2 UpdatePanel

Zagotovo najpogosteje uporabljen gradnik AJAX je UpdatePanel. Ta namesto osveževanja celotne strani omogoča osveževanje le želenega dela. V gradnik lahko namestimo poljubno število gradnikov ASP.NET in njihovo delovanje ne bo potrebovalo vedno vnovičnega osveževanja celotne strani, temveč samo sprememb znotraj gradnika UpdatePanel.

UpdatePanel je eden najuporabnejših gradnikov paketa ASP.NET AJAX. Z njim lahko delovanje v AJAX načinu vpeljemo v že izdelane aplikacije, in to brez spreminjanja kode.

1.3.3.3 UpdateProgressBar

Ta gradnik je razširitev za gradnik UpdatePanel in omogoča prikazovanje statusa med delnim osveževanjem strani.

Uporaba gradnika je pomembna med daljšim osveževanjem, ko moramo uporabnika opozoriti na dogajanje v teku. V nasprotnem primeru bi lahko uporabnik sklepal, da je s spletno aplikacijo nekaj narobe.

1.3.3.4 Timer

Gradnik omogoča intervalno pošiljanje strani v strežnik, kar pomeni, da lahko stran samodejno osvežimo v določenih intervalih brez kakršne koli interakcije uporabnika. Gradnik lahko uporabimo za osveževanje celotne strani v strežniku ali pa v kombinaciji z gradnikom UpdatePanel samo za del strani.

1.3.4 Dodatni gradniki ASP.NET AJAX

Poleg osnovnih gradnikov, ki jih dobimo že ob namestitvi paketa ASP.NET AJAX, so nam voljo še dodatni, v poseben paket združeni gradniki (AJAX Control Toolkit).

V tem poglavju so predstavljeni le najpogosteje uporabljeni in po mojem mnenju najuporabnejši gradniki, ki so v paketu dodatnih gradnikov (za opis vseh gradnikov iz paketa glejte prilogo 4).

1.3.4.1 Calendar

Calendar je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik TextBox. Po kliku takega gradnika TextBox se prikaže koledar, ki ob izboru datuma le-tega prepiše v TextBox.

Gradnik zelo olajša delo z datumi oziroma vpisovanje le-teh v za to določena polja. Takšno na prvi pogled zelo preprosto opravilo pa tako razvijalcem kot tudi uporabnikom pogosto povzroča veliko težav. Največkrat težave povzroča format zapisa datuma. Odločimo se

namreč lahko za uporabo ameriškega (mesec/dan/leto), evropskega (dan/mesec/leto) ali pa določimo kar svoj format zapisa. Pri tem je pomembno, da sta tako razvijalec kot tudi uporabnik usklajena pri interpretaciji datumov v polja. Primer datuma, kjer ima format zapisa odločilno vlogo, je na primer 7/6/2009. Vpisani datum je po vseh pravilih veljaven, od formata zapisa pa je odvisno, ali ga bo aplikacija obravnavala kot 7. junij 2009 ali kot 6. julij 2009.

Ravno takim težavam se lahko preprosto izognemo z uporabo gradnika Calendar. Po kliku določenega datuma se ta samodejno prepiše v polje za vpis datuma, upoštevajoč pred tem določen format zapisa.

Slika 6: Prikaz delovanja gradnika Calendar

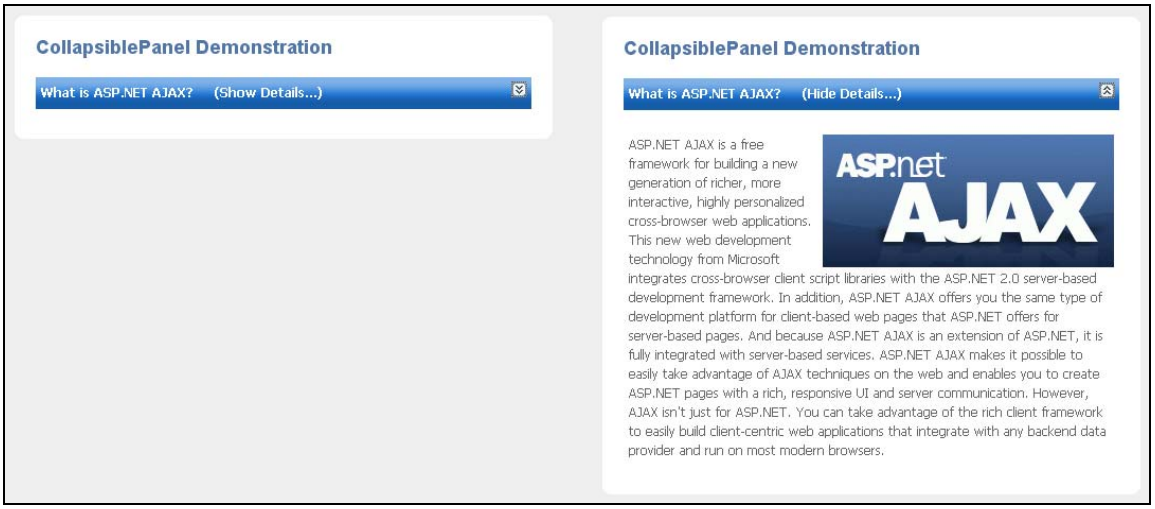
Vir: Calendar Demonstration, b.l.

1.3.4.2 CollapsiblePanel

CollapsiblePanel je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik Panel. Pri takem gradniku Panel je vidna samo glava, vsebina pa se prikaže, ko ga kliknemo .

Gradnik omogoča skrivanje za uporabnika nepomembnih podatkov in hkrati organizacijo ter strukturiranje prikaza internetne strani. Na stran lahko tako postavimo več vsebine, ne da bi s tem zmanjšali preglednost ali motili delo uporabnika.

Slika 7: Prikaz delovanja gradnika CollapsiblePanel



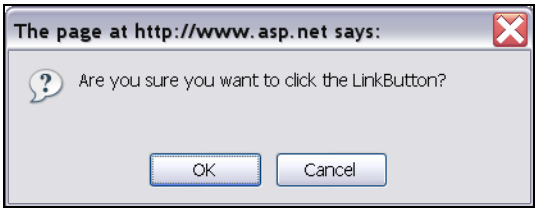
Vir: CollapsiblePanel Demonstration, b.l.

1.3.4.3 ConfirmButton

ConfirmButton je AJAX razširitev, ki jo lahko povežemo na kateri koli gumb. Po kliku takega gumba se pojavi sporočilo, ki od uporabnika zahteva potrditev izvedene akcije. Če uporabnik akcije ne potrdi, se sporočilo skrije, klik gumba pa se prezre.

Gradnik je zelo uporaben, ko od uporabnika zahtevamo potrditev pomembne akcije, na primer brisanja vrednosti iz zbirke podatkov. Na voljo sta dve različici delovanja gradnika, in sicer na principu čistega JavaScript opozorila ali s prikazom AJAX gradnika PopupControl. Uporaba prve različice je preprosta in zahteva zgolj vpis besedila sporočila.

Slika 8: Prikaz delovanja gradnika z uporabo čistega opozorila JavaScript



Vir: ConfirmButton Demonstration, b.l.

Če želimo uporabiti drugo različico, moramo najprej z dodatnim gradnikom PopupControl (glejte razdelek 1.3.4.7) izdelati okence, ki naj se prikaže kot opozorilo. Ta različica zahteva nekoliko več časa in znanja, a je rezultat precej boljši.

Slika 9: Prikaz delovanja gradnika z uporabo gradnika PopupControl

Vir: ConfirmButton Demonstration, b.l.

V obeh primerih pa je ključnega pomena, kako se ti dve sporočili prikažeta, in to je v modalnem načinu. To pomeni, da dokler so sporočila vidna, uporabnik ne more uporabljati nobene funkcionalnosti spletne aplikacije oziroma strani. Modalni obrazci so namenjeni vzdrževanju konsistentnosti programske rešitve, saj onemogočajo spreminjanje podatkov v neaktivnih formah.

1.3.4.4 FilteredTextbox

FilteredTextBox je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik TextBox in s tem omejimo nabor znakov, ki jih lahko vanj vpišemo.

Gradnik lahko na primer uporabimo, ko želimo od uporabnika pridobiti podatek o količini naročenega blaga. V takem polju ne sme biti nobenega drugega znaka kot število. Z

gradnikom FilteredTextBox to naredimo preprosto z nastavitvijo lastnosti na vrednost `'Numbers'`, gradnik pa bo v vnosno polje dovolil samo vnos števil.

1.3.4.5 ListSearchExtender

ListSearchExtender je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik ListBox ali DropDownList in omogoča iskanje po seznamu z vpisovanjem niza črk.

Gradnik je zelo uporaben, kadar gre za obsežen seznam, po katerem moramo iskati vrednost. Po kliku gradnika DropDownList se nad njim pojavi prostor, kjer se prikazujejo na tipkovnici vpisani znaki. Med vpisovanjem znakov se samodejno premikamo po seznamu in označi se vrednost, ki je najbližji zadetek vpisanim znakom.

Slika 10: Prikaz delovanja gradnika ListSearchExtender

Vir: ListSearchExtender Demonstration, b.l.

1.3.4.6 ModalPopup

ModalPopup je AJAX razširitev, ki jo lahko povežemo na kateri koli gumb in omogoča prikazovanje vsebin na modalni način.

Gradnik ModalPopup uporabimo, ko želimo od uporabnika pred nadaljevanjem dela pridobiti nujno informacijo, na primer potrditev ali zavrnitev akcije, vpis parametra in podobno.

1.3.4.7 PopupControl

PopupControl je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik ASP.NET in omogoča prikaz dodatnih gradnikov, ko ga kliknemo.

Primer je lahko prikazovanje koledarja po kliku vnosnega polja. V primerjavi z gradnikom Calendar pa lahko tukaj poleg koledarja hkrati prikažemo še kateri drug gradnik, na primer sliko z logotipom podjetja ali personalizirani koledar.

Slika 11: Prikaz delovanja gradnika PopupControl

Vir: PopupControl Demonstration, b.l.

1.3.5 Fleksibilnost gradnikov ASP.NET AJAX

Ena izmed pomembnejših lastnosti gradnikov ASP.NET AJAX je možnost njihovega prilagajanja in dograjevanja našim potrebam. Microsoft je namreč izvorno kodo vseh gradnikov objavil na internetu in tako razvijalcem omogočil poseg v njihovo delovanje. Prav vse gradnike lahko razvijalec po želji spremeni in tako še dodatno poveča njihovo koristnost. Seveda je za tak poseg potrebnega precej več znanja, saj se moramo zaradi specifik delovanja gradnikov AJAX spopasti s precej zapletenimi JavaScript skriptami. Ne glede na to pa je možnost prilagajanja in spreminjanja gradnikov za razvijalce velika prednost. To namreč pomeni večjo fleksibilnost pri izvedbi določene rešitve. Lahko se zgodi, da določen gradnik AJAX ustreza vsem merilom, a ga ne moremo uporabiti zaradi določene specifike, ki je ta ne podpira. V takih primerih bi bili prisiljeni uporabo takega gradnika AJAX opustiti, kar pa bi pomenilo korak nazaj. Prav zaradi fleksibilnosti gradnikov AJAX pa se temu skoraj vedno lahko izognemo.

1.3.5.1 Primer dograditve gradnika ASP.NET AJAX

Za ponazoritev fleksibilnosti gradnikov AJAX je v nadaljevanju predstavljena dograditev gradnika CollapsiblePanel, ki je v paketu dodatnih gradnikov (AJAX Control Toolkit).

Gradnik CollapsiblePanel povežemo na določen gradnik Panel (angl. *Panel Control*). Pri takem gradniku Panel je vidna samo njegova glava, vsebina pa se prikaže, ko ga kliknemo (glejte Slika 7 v razdelku 1.3.4). Kot je razvidno iz slike, ostanejo oblikovne lastnosti glave gradnika Panel vedno enake. Zamislimo pa si lahko situacijo, ko bi želeli glavo prikazati z različnimi oblikovnimi lastnostmi glede na to, ali je vsebina vidna ali ne. Tako delovanje bi lahko na primer potrebovali, ko je na internetni strani več gradnikov Panel in želimo nazorneje prikazati, kateri elementi so »odprti« (vidna je samo njihova glava) in kateri »zaprti« (vidna je celotna vsebina).

Postopek dograditve gradnika AJAX izvedemo na podlagi napotkov, ki jih je Microsoft objavil na strani http://msdn.microsoft.com. Tam so opisani postopki izdelave novega gradnika AJAX in nove AJAX razširitve, a iz njih lahko vseeno razberemo kaj je treba narediti, če želimo izvesti le spremembo gradnika AJAX. Čeprav gre torej v tem primeru le

za spremembo gradnika, pa je postopek vseeno precej zapleten. Zaradi tega so v diplomskem delu predstavljeni zgolj glavni koraki za izvedbo opravila.

Za izvedbo opisanega primera potrebujemo:

- Visual Studio 2008 (program za razvoj spletnih aplikacij v ASP.NET okolju),
- ASP.NET 2.0,
- ASP.NET AJAX 1.0,
- ASP.NET AJAX Control Toolkit,
- rešitev AjaxControlToolkit.

AjaxControlToolkit rešitev (angl. *solution*) lahko prenesemo z Microsoftove strani http://ajaxcontroltoolkit.codeplex.com. Gre za Visual Studio 2008 rešitev z več projekti, ki jih lahko zaženemo posamično. Glavna projekta sta projekt z izvorno kodo vseh gradnikov AJAX in projekt s primeri uporabe gradnikov AJAX, s katerim preverimo pravilno delovanje sprememb, ki jih izvedemo v izvorni kodi.

Cilj zgoraj opisanega primera je torej gradniku Panel določiti ene oblikovne lastnosti v primeru, ko je vidna samo njegova glava, in druge oblikovne lastnosti v primeru, ko je vidna njegova celotna vsebina. Oblikovne lastnosti gradniku Panel določimo na podlagi razreda CSS, opredeljenega v dokumentu CSS. To pomeni, da moramo gradniku CollapsiblePanel dodati dve novi lastnosti. Prva bo določala razred CSS gradniku Panel, ko bo »zaprt«, druga pa mu bo razred CSS določala, ko bo »odprt«.

Najprej moramo te lastnosti vgraditi v dokument **CollapsiblePanelExtender.cs**, ki hrani lastnosti, do katerih dostopamo na strežniški strani.

Slika 12: Prikaz novih lastnosti, uporabljenih na strežniški strani

```
[DefaultValue("")]
[ExtenderControlProperty]
public string ImageControlCssClassClosed
{
    get { return GetPropertyValue("ImageControlCssClassClosed", ""); }
    set { SetPropertyValue("ImageControlCssClassClosed", value); }
}

[DefaultValue("")]
[ExtenderControlProperty]
public string ImageControlCssClassOpened
{
    get { return GetPropertyValue("ImageControlCssClassOpened", ""); }
    set { SetPropertyValue("ImageControlCssClassOpened", value); }
}
```

Ker pa se delovanje gradnika AJAX v precejšnji meri odvija na odjemalčevi strani, mora biti do teh lastnosti omogočen dostop tudi v dokumentu **CollapsiblePanelBehavior.js**. Ta namreč vsebuje vse potrebne JavaScript skripte gradnika, ki se izvajajo na odjemalčevi strani. Pri tem koraku je bistvenega pomena, da so imena lastnosti, definiranih na strežniški strani, enaka tistim na odjemalčevi. Za branje in pisanje vrednosti v lastnostih moramo definirati tako funkcije `set` kot tudi `get`.

Slika 13: Prikaz novih lastnosti, uporabljenih na odjemalčevi strani

```
get_ImageControlCssClassClosed : function() {
        return this._imageControlCssClassClosed;
},

set_ImageControlCssClassClosed : function(value) {
    if (this._imageControlCssClassClosed != value) {
        this._imageControlCssClassClosed = value;
        this.raisePropertyChanged('ImageControlCssClassClosed');
    }
},

get_ImageControlCssClassOpened : function() {
    return this._imageControlCssClassOpened;
},

set_ImageControlCssClassOpened : function(value) {
    if (this._imageControlCssClassOpened != value) {
        this._imageControlCssClassOpened = value;
        this.raisePropertyChanged('ImageControlCssClassOpened');
    }
},
```

S tem omogočimo dostop do obeh novih lastnosti in jih že skoraj lahko uporabimo za spreminjanje razreda CSS gradniku Panel. Še zadnji korak pred tem je definiranje spremenljivk, ki bodo hranile vrednosti obeh lastnosti na odjemalčevi strani in bodo omogočale tako njihovo branje kot tudi pisanje.

Slika 14: Definiranje novih spremenljivk za dostop do novih lastnosti

```
this._imageControlCssClassClosed = null;
this._imageControlCssClassOpened = null;
```

Po storjenem lahko do novih lastnosti v celotni datoteki dostopamo preko ukazov:

- `this._imageControlCssClassClosed` in
- `this._imageControlCssClassOpened.`

Sledi še najtežji del, uporaba novih lastnosti za določitev atributa `class` gradniku Panel, s katerim določimo njegove oblikovne lastnosti. To najprej naredimo v funkciji `initialize` (še vedno v dokumentu CollapsiblePanelBehavior.js), ki se sproži ob inicializaciji AJAX gradnika. Najprej moramo preveriti njegovo trenutno stanje, torej, ali je vidna samo glava gradnika Panel ali pa celotna vsebina (Slika 15).

Slika 15: Preverjanje trenutnega stanja gradnika

```
if (this._collapsed)
```

Glede na ugotovljeno trenutno stanje gradnika Panel je treba nastaviti primerno vrednost za njegov atribut `class` (Slika 16 in Slika 17).

Slika 16: Sprememba atributa class gradniku Panel, ko je vidna samo glava

```
//če je CSS za "zaprt" gradnik določen -> nastavimo CSS
if (this._imageControlCssClassClosed != "")
{
    var triggerControl = $get(this._imageControlID);
    triggerControl.setAttribute("class", this._imageControlCssClassClosed);
    triggerControl.setAttribute("className", this._imageControlCssClassClosed);
}
```

Slika 17: Sprememba atributa class gradniku Panel, ko je vidna vsebina

```
//če je CSS za "odprt" gradnik določen -> nastavimo CSS
if (this. imageControlCssClassOpened != "")
{
    var triggerControl = $get(this._imageControlID);
    triggerControl.setAttribute("class", this._imageControlCssClassOpened);
    triggerControl.setAttribute("className", this._imageControlCssClassOpened);
}
```

Tako poskrbimo za prikaz pravilnega oblikovanja ob inicializaciji gradnika, torej ob naložitvi internetne strani. Naslednji korak je vpeljava spreminjanja oblikovanja ob kliku na gradnik Panel, ki sproži prikaz oziroma skrivanje njegove vsebine.

Če je vidna samo glava gradnika Panel, je treba, ko jo kliknemo, oblikovanje spremeniti v tisto, ki je določeno v spremenljivki `this._imageControlCssClassOpened`. To naredimo znotraj funkcije `_doOpen`, ki se izvede po kliku gradnika Panel, ko je ta zaprt. Za ta namen uporabimo kodo, predstavljeno že na zgornji sliki (Slika 16). Poskrbeti moramo tudi za delovanje v nasprotni smeri. Če je torej vidna celotna vsebina gradnika Panel, moramo po tem, ko ga kliknemo, oblikovanje spremeniti v tisto, ki je določeno v spremenljivki `this._imageControlCssClassClosed`. To naredimo znotraj funkcije `_doClose`, ki se izvede po kliku gradnika Panel, ko je ta odprt. V ta namen prav tako uporabimo že predstavljeno kodo na zgornji sliki (Slika 17).

Zatem je spremenjeni gradnik AJAX končan in ga lahko uporabimo v spletni aplikaciji (seveda moramo najprej posodobiti knjižico gradnikov AJAX v projektu). Na spodnji sliki (Slika 18) je prikazana uporaba dograjenega gradnika CollapsiblePanel v kodi HTML. Opazni sta tudi dve novi lastnosti, s katerima določimo en razred CSS za odprt gradnik (`ImageControlCssClassOpened`), in drugega za zaprtega (`ImageControlCssClassClosed`).

Slika 18: Uporaba dograjenega gradnika CollapsiblePanel

```
<ajaxToolkit:CollapsiblePanelExtender ID="cpeDemo" runat="Server"
    TargetControlID="Panel1"
    ExpandControlID="Panel2"
    CollapseControlID="Panel2"
    Collapsed="True"
    TextLabelID="Label1"
    ImageControlID="Panel2"
    ExpandedText="(Hide Details...)"
    CollapsedText="(Show Details...)"
    ExpandedImage="~/images/collapse_blue.jpg"
    CollapsedImage="~/images/expand_blue.jpg"
    SuppressPostBack="true"
    SkinID="CollapsiblePanelDemo"
    ImageControlCssClassClosed="testClosed"
    ImageControlCssClassOpened="testOpened" />
```

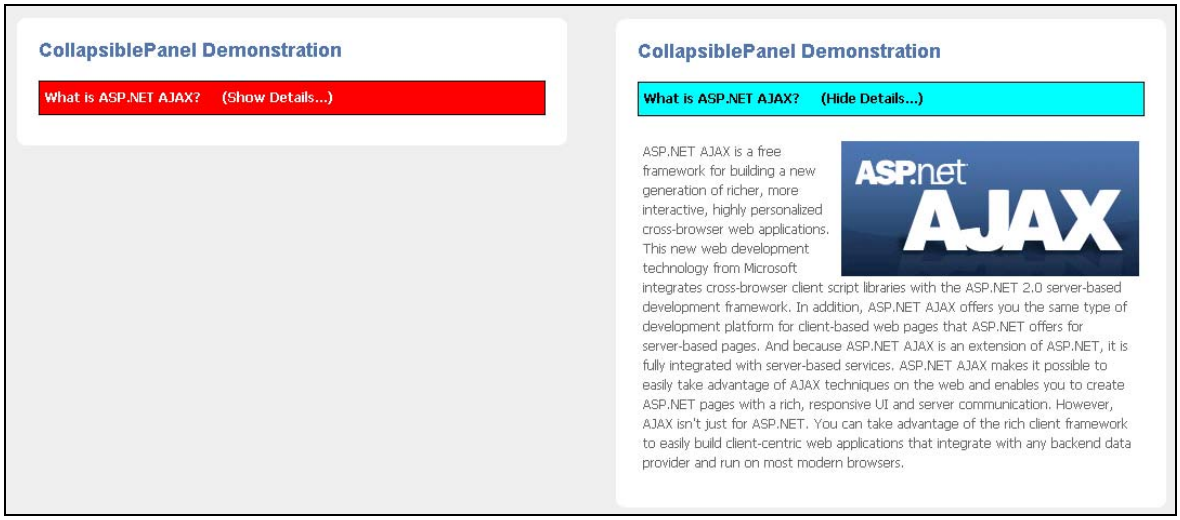
Za oblikovanje gradnika poskrbimo z razredoma CSS, prikazanima na spodnji sliki (Slika 19).

Slika 19: Razred CSS dograjenega gradnika

```
.testClosed
{
    background-color:Red;
    font-weight:bold;
    color:White;
    border:1px solid black;
}

.testOpened
{
    background-color:Aqua;
    font-weight:bold;
    color:Black;
    border:1px solid black;
}
```

Končna oblika gradnika je prikazana na spodnji sliki (Slika 20), kjer je razvidna razlika v oblikovanju glave odprtega in zaprtega gradnika Panel.

Slika 20: Primer uporabe spremenjenega gradnika AJAX CollapsiblePanel



Za v diplomskem delu predstavljen primer so bile gradniku Panel določene zgolj preproste oblikovne spremembe, to je oblikovanje barve ozadja, pisave in robov. Lahko pa bi razreda CSS razširili in gradniku Panel dodali slike, mu spremenili velikost ali lego. Dodali pa bi mu lahko tudi povsem druge oblikovne lastnosti, ko bi se nad njega pomaknili z miško.

Spreminjanje oziroma dograjevanje gradnikov AJAX vsekakor ni trivialno opravilo, a je v primeru manjših sprememb povsem izvedljivo. Zagotovo lahko tako rešimo morda za projekt ključen problem.

Omeniti velja tudi, da je razvijalcem omogočeno tudi izdelovanje povsem novih gradnikov AJAX, ki jih lahko, prav tako kot vse druge, nato uporabimo v vsakem projektu. Postopek razvoja povsem novega gradnika pa je še kompleksnejši, tako da se ga bodo najverjetneje razvijalci lotili le v primeru zares specifične zahteve naročnika. Upoštevati namreč moramo, da razvoj povsem nove kontrole vedno pomeni določeno tveganje. Po eni strani nam tako opravilo vzame kar precej časa, po drugi pa testiranje zagotovo ni nikoli tako izčrpno, kot to velja za že obstoječe gradnike AJAX.

2 ASP.NET AJAX z vidika razvijalca

Razvijalec je oseba, zadolžena za analizo, načrtovanje in izvedbo rešitve oziroma spletne aplikacije, ki je odgovor na določeno naročnikovo potrebo. Razvoj se vedno začne z analizo, kjer naj se razvijalec ne bi preveč obremenjeval z orodji in s tehnikami, ki jih bo uporabil pri rešitvi. V praksi pa se pogosto izkaže, da se analiza in načrt velikokrat prepletata. Razvijalec namreč že pri analiziranju problema razmišlja o načinu rešitve določene zadeve. Tako razmišljanje razvijalcu omogoča poglobitev analize tam, kjer to zahteva določena ideja o načrtu rešitve. Ali je zamišljena rešitev izvedljiva, je namreč odvisno tudi od predvidenih tehnik razvoja.

V fazi analize je tako razmišljanje o tehnologiji, ki jo bomo uporabili pri rešitvi, že prisotno, a v minimalni obliki. Ideje, ki se porajajo pri analizi, so namreč zelo grobe in ne ciljajo specifične tehnologije.

V naslednjih fazah, torej v fazi načrtovanja in izvedbe, pa je razmišljanje o uporabljenih orodjih in tehnikah precej bolj prisotno. Naslednji dve fazi sta zato tudi podrobneje analizirani, pri čemer je analiza izvedena na podlagi primerjave standardnega načina razvijanja in razvijanja s tehnologijo AJAX. Z izrazom **standardni način razvijanja** med analiziranjem označujemo razvijanje v razvojnem okolju ASP.NET 2.0. Z izrazom **razvijanje s tehnologijo AJAX** pa označujemo razvijanje v istem okolju, le da je vanj vključen AJAX, torej paket ASP.NET AJAX (glejte razdelek 1.3.2) in paket dodatnih gradnikov AJAX Control Toolkit (glejte razdelek 1.3.4).

2.1 Faza načrtovanja

Glede na analizo se izoblikuje načrt, ki naj bi (v analizi) zaznan problem poizkušal rešiti čim preprosteje. To najpogosteje pomeni, da mora biti rešitev natančna, časovno izvedljiva in do uporabnika prijazna. Natančnost pomeni, da rešitev deluje točno tako, kakor bi morala, časovna izvedljivost pomeni, da jo je mogoče realizirati v določenem časovnem obdobju, prijaznost do uporabnika pa, da je delovanje za uporabnika intuitivno in tekoče.

Razvoj aplikacijske rešitve je načeloma vedno časovno in stroškovno omejeno opravilo. Povsod, kjer so omejitve, pa je treba določiti tudi prioritetne in neprioritetne naloge. Natančno delovanje aplikacije v vsakem podjetju zagotovo spada med prioritetne naloge. Ta je namreč tako pomembna, da bi bila aplikacija brez nje povsem neuporabna. Med neprioritetnimi nalogami pa je največkrat do uporabnika prijazno delovanje, saj je v teoriji tudi brez nje aplikacija še vedno uporabna. Povsem drugo vprašanje pa je, ali se bo zaradi okornega delovanja taka aplikacije tudi v resnici uporabljala.

2.1.1 Standardni način razvijanja

Načrtovanje rešitve z uporabo standardnih tehnologij je največkrat razdeljeno na dve nasprotujoči si komponenti. Po eni strani se razvijalec obremenjuje s časovnimi

omejitvami razvoja, da bi aplikacijo zaključil v čim krajšem času, po drugi strani pa mora aplikacija delovati natančno, kar narekuje previdnost in premišljenost. Razvijalec lahko v takih situacijah namerno ali nenamerno zanemari tretjo komponento, to je zahtevo po za uporabnika prijaznem delovanju. Razvijalec je namreč nenehno deležen pritiskov po hitrem dokončanju določenega projekta, zato se osredini na izvedbo najpomembnejših sestavin za delovanje aplikacije, to je na natančno delovanje.
V praksi se tako lahko glede na časovne omejitve razvijalca pojavljajo naslednje skrajnosti:

- aplikacija deluje natančno, a za uporabnika neprijazno;
- aplikacija deluje natančno, za uporabnika prijazno, a vsebuje manj funkcionalnosti;
- aplikacija deluje natančno in za uporabnika prijazno (razvijalec je za aplikacijo porabil več kot dogovorjeno število ur).

Naročniki pogosto za uporabnika prijaznega delovanja eksplicitno niti ne zahtevajo, a ga v resnici vseeno pričakujejo. To je vsekakor odvisno od naročnika in sektorja, v katerem deluje, vendar pa to v praksi velikokrat velja.

Za uporabnika prijazno delovanje pa je navsezadnje tudi cilj podjetja, ki rešitev razvija. Taka aplikacija je namreč uporabnikom zagotovo precej bolj všeč, kar z drugimi besedami pomeni zadovoljne uporabnike. Zadovoljni uporabniki pa pomenijo zadovoljne naročnike, kar je bistveno za dolgoročni obstoj razvijalskega podjetja.

Že v fazi načrtovanja se torej zaradi časovne stiske moramo odrekati nekaterim funkcionalnostim aplikacije in (ali) za uporabnika prijaznemu delovanju. Še najpogosteje trpi za uporabnika prijazno delovanje, ki je v mnogih podjetjih opravilo z dna seznama prioritet. Ravno v tem je slabost razvoja s standardno tehnologijo. Določene rešitve bi za pravilno izvedbo zahtevale veliko več časa, kot ga je v resnici na voljo. To še najbolj velja, kadar gre za vgraditev za uporabnika prijaznega delovanja. Ker se razvijalec tega zaveda, mora skladno s tem ukrepati in rešitev nekoliko prilagoditi zahtevanemu času izvedbe.

2.1.2 Razvijanje s tehnologijo AJAX

Ko začne razvijalec načrtovati aplikativno rešitev z uporabo tehnologije AJAX, je zadeva nekoliko drugačna. Že odločitev o tem, da bo aplikacija delovala na podlagi tehnologije AJAX, pomeni osredinjenost na za uporabnika prijazno delovanje. Z gradniki AJAX namreč to dosežemo hitro in preprosto. Ker se razvijalec tega zaveda, lahko že med načrtovanjem zastavi veliko boljšo aplikativno rešitev, kot bi jo v nasprotnem primeru. Za določene naloge bi z uporabo tehnologije AJAX potrebovali okoli deset do dvajset minut, medtem ko bi za enak učinek z uporabo standardnih metod porabili uro ali več. Razlika je očitna in prednosti, ki jih prinaša tak prihranek časa, tudi. Razvijalec lahko načrtuje rešitev, ki je boljša, torej vsebuje več funkcionalnosti, je učinkovitejša in predvsem za uporabnika

mnogo prijaznejša. Hkrati lahko rešitev razvije v povsem enakem časovnem okviru kot do zdaj.

2.2 Faza izvedbe

V tej fazi se rešitev ideje prenese v dejansko »fizično« rešitev. Največ prekoračitev načrtovanih delovnih ur je prav v tej fazi. Določene težave se namreč pojavijo šele med izvedbo, kar pa ne pomeni vedno, da je bil načrt slabo zastavljen. Lahko se namreč pojavijo okoliščine, ki jih med načrtovanjem nismo mogli predvideti. Morda šele pri izvedbi ugotovimo, da je del rešitve v praksi neizvedljiv, bodisi zaradi omejitev tehnologije bodisi zaradi nepraktičnosti. Razvijalec se mora v takih primerih vrniti k načrtovanju in izvesti primerne prilagoditve oziroma spremembe, ki pa zahtevajo veliko časa. Tako je lahko ogrožena pravočasna izvedba rešitve. V takih situacijah postane najpogostejša rešitev zanemarjenje za uporabnika prijaznega delovanja.

2.2.1 Standardni način razvijanja

Kadar je časovna stiska velika, se razvijalec ne more posvetiti razvijanju za uporabnika prijaznega delovanja. Najprej se mora posvetiti natančnemu delovanju in šele ko je ta pogoj izpolnjen, se lahko osredini na druge zahteve. V standardnih oblikah programiranja sta namreč natančno delovanje in za uporabnika prijazno delovanje povsem ločena drugo od drugega. Razvijalec najprej poskrbi, da določen postopek deluje natančno, šele nato se lahko posveti uporabniku in zanj prijaznemu delovanju.

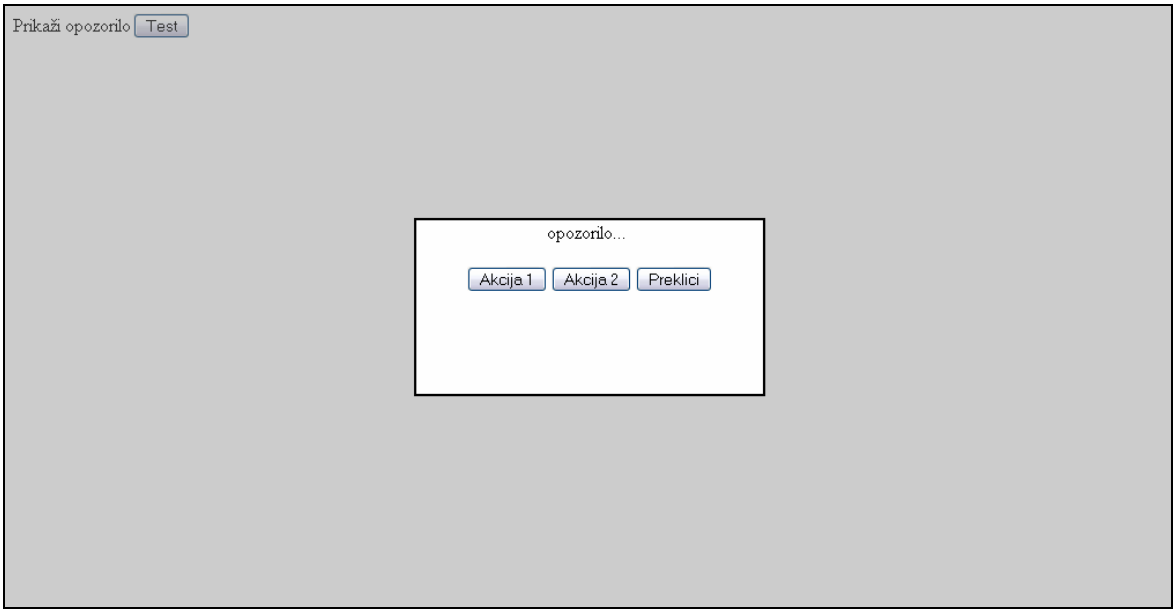
2.2.2 Razvijanje s tehnologijo AJAX

V nasprotju s standardnim načinom programiranja pomeni programiranje na podlagi tehnologije AJAX združevanje natančnega in za uporabnika prijaznega delovanja. Razvijalec se z uporabo gradnikov AJAX že posveča uporabniku. Aplikacijo z uporabo tehnologije AJAX zaključimo v skorajda enakem časovnem razponu, le da je za končnega uporabnika z vidika prijaznosti vmesnika precej boljša. Seveda pa se moramo zavedati, da je časovni razpon v obdobju uvajanja razvijanja tehnologije AJAX v podjetju nekoliko daljši, kar je tudi razumljivo. Ko pa se razvijalci seznanijo z novimi tehnikami dela in jih osvojijo, je časovni razpon primerljiv s standardnim razvojem.

2.2.3 Primerjava obeh tehnik ob praktičnem primeru

Razlike med obema načinoma programiranja je najlažje ponazoriti s praktičnim primerom. Zamislimo si situacijo, ko uporabniku po kliku določenega besedila v spletni aplikaciji želimo prikazati opozorilo. Opozorilo naj bi uporabniku omogočalo izbiro izvedbe ene izmed dveh akcij ali preklic (Slika 21).

Slika 21: Prikaz opozorila z dvema možnima akcijama in s preklicem



Osnovno stran sestavljata dva gradnika ASP.NET – labela (angl. *Label*) »Prikaži opozorilo« in gumb (angl. *Button*) »Test«. Po kliku labele se prikaže opozorilo, vidno na zgornji sliki (Slika 21). Med prikazom opozorila se mora ozadje onemogočiti in posiviti, tako da lahko uporabnik upravlja samo s tremi gumbi na opozorilu. Gumb »Test« na osnovni strani mora biti za uporabnika onemogočen. Opozorilo se mora torej prikazati na modalni način.

V nadaljevanju je predstavljen primer, kako bi se morali zadeve lotiti glede na uporabljeno tehniko razvijanja. Postopek je za lažjo primerjavo razdeljen v tri sklope (posamezni dokumenti so vidni v prilogi 3):

- sklop oziroma dokument **HTML**, viden v uporabniškem brskalniku;
- sklop oziroma strežniško kodo **C#**, ki se izvaja na strežniku;
- sklop **CSS** oziroma oblikovne lastnosti elementov, ki se odražajo v uporabniškem brskalniku.

Po analizi vseh sklopov za posamezno tehniko je podan kratek sklep, pri čemer je uporabljena tehnika ocenjena glede na zahtevnost uporabljenih JavaScript skript, zahtevnost same izvedbe in časovne potratnosti. Sledi še ocena končnega rezultata, pri čemer je delovanje izdelane rešitve ocenjeno glede na uporabniško prijaznost, odzivnost in tekoče delovanje. Ocena je ovrednotena z eno do s petimi zvezdicami, pri čemer pet zvezdic pomeni najboljšo oceno.

2.2.3.1 Standardni način (samo z uporabo strežniške kode)

Problem lahko rešimo na klasičen način in pri tem uporabljamo izključno strežniško kodo. V praksi to pomeni, da se za vsako uporabnikovo akcijo stran pošlje nazaj na strežnik. Tam se izvede pripadajoča strežniška koda in celotna stran se spet pošlje nazaj k uporabniku,

kjer se v brskalniku ponovno naloži. Klasični način razvijanja je za razvijalce v mnogih primerih časovno najmanj potraten, a z vidika za uporabnika prijaznega končnega proizvoda daje tudi najslabše rezultate. Stran za vsako uporabnikovo akcijo, zaradi prenosov strani na strežnik in nazaj, neprestano utripa pred očmi uporabnika.

Ko pa želimo v klasičnem načinu izdelati nekakšno nestandardno rešitev, kot je to predstavljen primer, pa naletimo na težave, saj izvedba zahteva kar precej več časa, kot je običajno. Največji problem pri izdelavi opisane rešitve predstavlja prikaz opozorila na modalni način, ki za izvedbo zahteva uporabo določenih trikov in specifičnih rešitev.

Za izdelavo spletne aplikacije s to metodo je potrebno izvesti naslednje korake:

HTML (priloga 3, Slika 1):

- Prva naloga je postavitev gumba in labele, ki bo sprožila prikaz opozorila. Ker pa je labela namenjena zgolj prikazovanju besedila, s klikom labele ne moremo sprožiti izvedbe strežniške kode. Rešitev je uporaba gradnika LinkButton, ki predstavlja gumb v obliki besedila. Gradniku določimo strežniško kodo, ki naj se, ko ga kliknemo, izvede (`lnkbtnPopup_Click`).
- Naslednji korak je izdelava opozorila, ki naj se prikaže. V tem koraku je potrebno o nadaljnjih potezah nekoliko premisliti. Za izvedbo delovanja na modalni način je treba na strani celotno ozadje posiviti in v ospredju prikazati opozorilo. To lahko dosežemo z uporabo dveh panelov, enega za prikaz sivega ozadja in drugega za prikaz opozorila. V kodo HTML tako postavimo dva nova panela, `pnlSivina` in `pnlOpozorilo`. Za lažje upravljanje z njima pa ju postavimo v skupen »div« z imenom `divOpozorilo`.
- V naslednji fazi moramo zgraditi vsebino prikazanega opozorila. Na panel `pnlOpozorilo` tako vstavimo tri nove gumbe in vsakemu določimo strežniško kodo.

C# (priloga 3, Slika 2):

- Na strežniški strani poskrbimo za pravilno prikazovanje in skrivanje opozorila. Najprej v kodi, ki se izvede po kliku gumba LinkButton (`lnkbtnPopup`), poskrbimo, da uporabniku opozorilo prikažemo.
- V naslednji fazi poskrbimo za izvajanje ustrezne kode in skrivanje opozorila po kliku gumbov v opozorilu. Po kliku gumba »Prekliči« poskrbimo, da se opozorilo skrije. Po kliku enega izmed drugih dveh gumbov pa najprej izvedemo poljubno kodo in šele nato opozorilo skrijemo.

CSS (priloga 3, Slika 3):

- Do tega trenutka aplikacija še vedno ne deluje, kakor bi morala. Da dosežemo delovanje na modalni način, moramo poskrbeti za pravilno oblikovanje vseh

dodanih gradnikov. Najprej oblikujemo sivo ozadje. Določimo barvo ozadja (`background-color`), širino in višino (`width, height`), raven prozornosti (`filter, -moz-opacity`) in oddaljenost od zgornjega oziroma levega roba (`top, left`). Najpomembnejši atribut je prozornost, ki omogoča, da skozi sivino lahko še vedno vidimo preostale gradnike. Za pravilno delovanje pa so ključnega pomena pravzaprav vsi atributi. Če je kateri izmed njih nepravilno določen, delovanje ne bo tako, kakor smo si ga zamislili.

- V drugi fazi se posvetimo določitvi atributov sporočila, ki se bo prikazalo. V prvem sklopu so atributi, ki določajo njegov videz in za delovanje niso pomembni, so pa zato toliko pomembnejši atributi, ki so v drugem sklopu, to so lega (`position`), oddaljenost od zgornjega oziroma levega roba (`top, left`) in z-indeks (`z-index`). Z atributi `position, top` in `left` določimo, da bo naše opozorilo na strani prikazano v središču spletne aplikacije. Vrednosti `top` in `left` moramo določiti glede na velikost sporočila in moramo zato pravilno vrednost sami določiti z nekaj poizkusi. Najpomembnejši atribut pa je `x-index`. Atribut predpostavlja, da je stran razdeljena na več slojev. Prvi sloj ima vrednost nič in je pod slojem z vrednostjo 1. Sloji si tako sledijo drug za drugim in vrednost atributa `z-index`, ki jo ima posamezni sloj, določa, ali ta drugega prekrije ali ne. Gradniki na sloju z atributom `z-index` nič ne bodo dosegljivi (čeprav bodo morda vidni), če njihov sloj prekrijemo s slojem, ki mu določimo `z-index` ena. Po takem principu deluje tudi opozorilo. S tem, ko mu določimo `z-index` tisoč, prekrijemo vse morebitne podsloje, tako da bodo njegovi gradniki nedosegljivi, čeprav bodo še vedno vidni.

Ko izvedemo vse zgoraj opisane postopke, lahko spletno aplikacijo testiramo. Kliknemo gumb »Prikaži opozorilo« in opozorilo se tudi prikaže. Uporabnik lahko v tem trenutku z miško klikne samo tri gumbe, ki so na voljo v opozorilu. Gumba »Test« v ozadju opozorila na primer ne more klikniti.

Izdelava zahtevanih funkcionalnosti z uporabo standardnih metod ne zahteva praktično nobenega znanja jezika JavaScript, kar je za mnoge razvijalce pomemben dejavnik. Poleg tega, da jezika ne obvladajo ravno vsi, pa je njegov največji problem odpravljanje napak. Ravno zaradi tega je izdelava spletne aplikacije brez uporabe JavaScript skript največkrat hitrejša in preprostejša. V konkretnem primeru je še največ težav povzročala zasnova modalnega sporočila. Čeprav se morda zdi rešitev precej preprosta, pa ni ravno samoumevna. Preden pridemo do take rešitve, potrebujemo kar nekaj testiranja in spreminjanja kode HTML in CSS. Ravno zaradi tega je prikazana rešitev časovno precej neugodna. Tudi končni rezultat ni ravno blesteč. Rešitev resda deluje pravilno, a ne tudi optimalno in za uporabnika prijazno. Po vsakem kliku, ki ga uporabnik izvede, se namreč celotna stran, po izvedbi strežniške kode, ponovno naloži v brskalnik. To se odraža v utripanju zaslonske slike, kar je z uporabniškega vidika precej moteče.

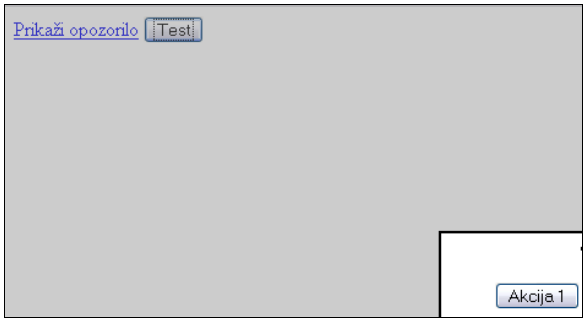
Tabela 1: Ocena nalog, izvedenih s standardnimi metodami

Naloga\lestvica	***1***	***2***	***3***	***4***	***5***
Zahtevnost skript JavaScript	x				
Zahtevnost izvedbe			x		
Časovna potratnost				x	
Končni rezultat	★★☆☆☆				

Legenda: 1 pomeni najmanj, 5 največ

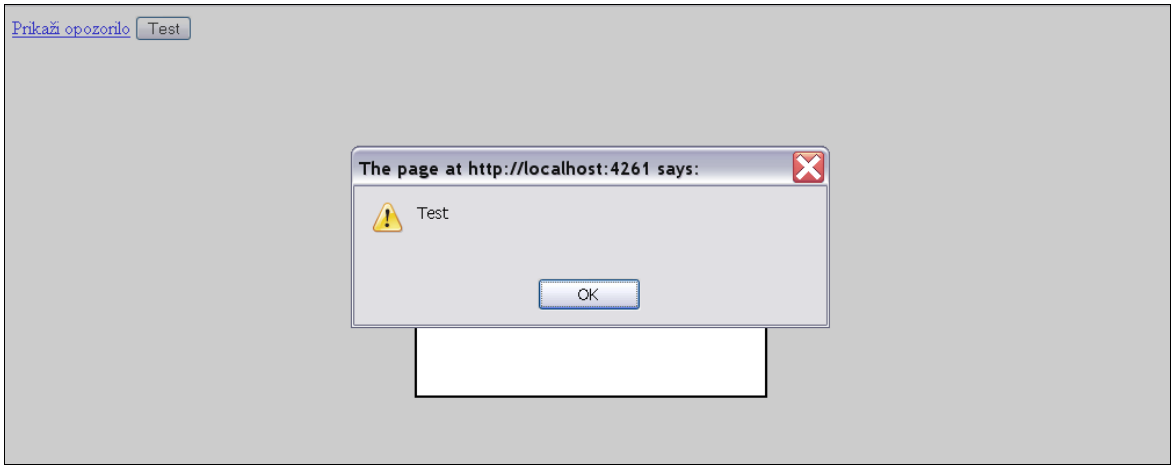
Za rešitev pa je značilna še ena večja pomanjkljivost, ki bi jo lahko preprosto spregledali, pa čeprav je za pravilno delovanje zelo pomembna. Ob prikazu opozorila je gumb »Test« na osnovni strani z miško res nedosegljiv, a do njega lahko pridemo po drugi poti, z uporabo tipke »Tab« na tipkovnici. Ta namreč omogoča zaporedno izbiro gradnikov v spletnih aplikacijah in je namenjena hitrejšemu delu oziroma delu brez miške. V predstavljenem primeru tipko »Tab« pritiskamo toliko časa, dokler se na gumbu »Test« ne pojavi črtasti kvadratek, kar pomeni, da je gumb izbran (Slika 22).

Slika 22: Prikaz izbranega gumba Test z uporabo tipke Tab na tipkovnici



Za potrebe prikaza sprožitve gumba »Test« smo mu v kodi HTML določili preprosto JavaScript skripto, ki po kliku nanj izpiše opozorilo z vsebino »Test«. Na takšen način lahko testiramo, ali akcijo gumba z izbiro in s pritiskom tipke »Enter« dejansko sprožimo.

Slika 23: Prikaz sprožitve onemogočenega gumba



Kot je razvidno iz zgornje slike (Slika 23) lahko, miški onemogočen gumb, sprožimo s tipkovnico. Tako delovanje je vsekakor nepravilno in ga ne moremo uporabiti v kaki pomembnejši spletni aplikaciji. Rešitev samo na prvi pogled ponuja pravi modalni način delovanja, pravzaprav pa gre za njegovo simulacijo, ki pa bi vseeno preslepila kar lepo število uporabnikov. Tako rešitev bi lahko uporabili le v najpreprostejših spletnih aplikacijah, saj je nevarnost, ki jo predstavlja možnost dostopa do vseh gumbov, v drugih primerih izjemno velika.

2.2.3.2 Standardni način (z uporabo strežniške kode in JavaScript skript)

Če želimo, da bo stran nekoliko učinkovitejša in za uporabnika prijaznejša, se moramo zateči k JavaScript skriptam. Te namreč omogočajo osveževanje vsebine in videza strani, ne da bi za to morali klicati strežnik in ponovno nalagali celotno stran. Akcije, ki v strežniški kodi ne naredijo ničesar drugega kot to, da prikažejo oziroma skrijejo opozorilo, lahko tako izvedemo z JavaScript skriptami. V predstavljenem primeru velja to pri kliku besedila »Prikaži opozorilo« in po kliku gumba »Prekliči« v opozorilu.

Postopek za izdelavo strani po tej metodi je v mnogih delih enak prejšnjemu načinu, zato bomo tu poudarili samo razlike:

HTML (priloga 3, Slika 4):

- Prva razlika je že dejstvo, da lahko JavaScript akcijo, na primer `onclick`, določimo skoraj vsem gradnikom. V tem primeru ni več potrebno uporabiti gradnika LinkButton, zato ga lahko zamenjamo s preprosto labelo.
- Druga razlika je posledica uporabe JavaScript skripte za skrivanje in prikazovanje celotnega opozorila (divOpozorilo). JavaScript skripta namreč ne more operirati z lastnostjo (`Visible`), ki je tipična strežniška lastnost. Za prikazovanje in skrivanje opozorila moramo poskrbeti z atributi CSS, natančneje z atributom `display`. Da se ob prvem zagonu strani opozorilo ne bo prikazalo, prenastavimo privzeto vrednost atributa opozorila na `display:none`.
- Največja razlika pa je v zgornjem delu dokumenta HTML, natančneje v elementu `<script> </script>`. Tukaj so JavaScript skripte, ki omogočajo prikazovanje oziroma skrivanje opozorila. Obe funkciji kot vhodni parameter sprejmeta ime gradnika, ki ga želimo uporabiti. Glede na ime v kodi kreiramo referenco na objekt, v tem primeru na gradnik `divOpozorilo`, in mu spremenimo atribut `display`, ki določa, ali naj se gradnik prikaže ali ne.

C# (priloga 3, Slika 5):

- Prva razlika je v tem, da v strežniški kodi ni več funkcij, ki bi se izvedle po kliku besedila »Prikaži opozorilo« in gumba »Prekliči« v opozorilu. Te naloge namreč izvedemo z JavaScript skriptami.

- Da se po kliku labele »Prikaži opozorilo« in gumba »Prekliči« izvedejo v kodi HTML napisane JavaScript funkcije, jih moramo nekako pripeti na omenjena gradnika. To naredimo v metodi `Page_Load` z uporabo metode `Attributes.Add` posameznega gradnika. Metodi moramo podati ime akcije, ki naj sproži določeno JavaScript funkcijo, v tem primeru želimo, da se sproži po kliku gradnika, torej `onclick`. Kot drugi parameter pa določimo ime funkcije in njene vhodne parametre.

Slika 24: Prikaz pripenjanja funkcije JavaScript posameznemu gradniku

```
protected void Page_Load(object sender, EventArgs e)
{
    //labeli dodamo JavaScript funkcijo za prikaz opozorila
    lblPopup.Attributes.Add("onclick", "prikaziOpozorilo('" + divOpozorilo.ClientID + "')");
    btnPreklici.Attributes.Add("onclick", "return skrijOpozorilo('" + divOpozorilo.ClientID + "')");
}
```

- Poskrbeti pa je treba še za drugačno prikazovanje oziroma skrivanje opozorila. Ker smo v aplikacijo za ta namen vključili JavaScript skripto, moramo, kot že rečeno, to izvesti nad atributom CSS gradnika `divOpozorilo`, in ne več nad njegovo lastnostjo `Visible`.

Slika 25: Prikaz spremembe atributa CSS s strežniško kodo

```
protected void btnAkcija1_Click(object sender, EventArgs e)
{
    //izvedemo željeno akcijo
    //...

    //skrijemo opozorilo
    divOpozorilo.Style.Add("display", "none");
}
```

CSS (priloga 3, Slika 6):

- V postopku ni razlik.

Če po storjenem preverimo delovanje spletne aplikacije, opazimo, da je povsem enako tistemu, ki je prikazano pri prejšnji metodi, le da zdaj za prikazovanje in skrivanje opozorila (po kliku gumba »Prekliči«) strani ni treba ponovno naložiti. Obe spremembi se namreč izvedeta v JavaScript skripti in sta vidni takoj, brez nepotrebne komunikacije s strežnikom in motečega utripanja zaslonske slike.

Z vidika učinkovitosti in uporabniške prijaznosti je ta metoda zagotovo boljša od prejšnje. Slabost pa je večja zahtevnost glede znanja jezika JavaScript, saj ne glede na relativno preprostost JavaScript funkcij zahteva kar nekaj izkušenj, da lahko celotno zadevo uspešno povežemo v končno obliko. Zahtevnost izvedbe je zaradi tega tu nekoliko višja. Razvijalec mora biti veliko pozornejši na morebitne napake, saj je podpora za razvijanje na podlagi skript JavaScript z vidika razhroščevanja zelo slaba. Vse to povzroči tudi daljši čas razvoja, saj je treba posameznim nepravilnostim nameniti kar precej časa. Tudi za testiranje je potrebnega nekoliko več časa, saj je med drugim treba preveriti pravilnost delovanja JavaScript skripte v vseh najpogosteje uporabljenih brskalnikih. Pri uporabi

JavaScript skript je namreč veliko krat delovanje v enem brskalniku brezhibno, v drugem pa naletimo na napako. Največje razlike so med brskalnikoma Internet Explorer in FireFox. Ko težave odpravimo, pa je končni rezultat že veliko boljši kot v prejšnjem primeru.

Tabela 2: Ocena nalog, izvedenih s standardnimi metodami in JavaScript skriptami

Naloga\lestvica	***1***	***2***	***3***	***4***	***5***
Zahtevnost JavaScript skript			x		
Zahtevnost izvedbe				x	
Časovna potratnost					x
Končni rezultat	★★★☆☆				

Legenda: 1 pomeni najmanj, 5 največ

Na žalost pa se tudi v tem primeru nismo izognili napaki, ki se je pojavila že v prejšnjem primeru. S tipkovnico lahko tako še vedno zaženemo akcijo gumba »Test«, tudi takrat, ko naj bi bilo opozorilo prikazano v modalnem načinu. Napaki bi se lahko izognili samo s posebno JavaScript funkcijo, ki bi omogočila onesposobitev takega delovanja. Pisanje take funkcije pa bi od nas zahtevalo že kar precejšnje znanje jezika JavaScript in mnogo več časa, tako da je z vidika časovne omejenosti razvoja to mnogokrat neizvedljivo.

2.2.3.3 Način AJAX

Za dosego modalnega načina prikazovanja določenega opozorila je v okolju ASP.NET z vključenim načinom AJAX poseben gradnik, ki se imenuje ModalPopupExtender. Pri uporabi tega gradnika se ni treba ukvarjati z onemogočanjem gradnikov v ozadju in s prikazovanjem sivine, saj se to izvede samodejno. V nadaljevanju je naveden primer rešitve naloge na podlagi tehnologije AJAX.

HTML (priloga 3, Slika 7):

- Predpogoj za uporabo katerih koli gradnikov AJAX je postavitev gradnika `ScriptManager`, ki omogoča izvedbo JavaScript funkcij, uporabljenih v gradnikih AJAX.

Slika 26: Primer vključitve gradnika ScriptManager v kodo HTML

```
<form id="form1" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server" />
```

- V naslednjih korakih lahko začnemo dodajati želene gradnike. Najprej dodamo labelo »Prikaži opozorilo« in gumb »Test«, ki bosta vidna na osnovni strani.
- Nato v gradniku Panel (`pnlOpozorilo`) oblikujemo sporočilo, ki ga želimo prikazati. Sporočilu dodamo tudi tri gumbe, in sicer »Akcija 1«, »Akcija 2« in »Prekliči«.

- Za prikazovanje pravkar izdelanega opozorila na modalni način uporabimo AJAX gradnik, imenovan `ModalPopupExtender`. Temu moramo najprej določiti gradnik, ki naj ga prikaže na modalni način, v predstavljenem primeru je to ravnokar izdelan Panel `pnlOpozorilo`. Nato določimo še gradnik, ki sproži prikazovanje opozorila, (`lblPopup`), in tistega, ki sproži njegovo skrivanje (`btnPreklici`). Tako določena gradnika za prikazovanje in skrivanje opozorila bosta to izvajala s prednastavljeno JavaScript skripto v gradniku AJAX, zato komunikacija s strežnikom ne bo potrebna. Zadnji atribut, ki ga moramo še določiti gradniku `ModalPopupExtender`, je ozadje, ki naj se uporabi ob prikazovanju sporočila na modalni način. Z njim določimo zgolj obliko ozadja, funkcionalnost, torej onemogočenje gradnikov v ozadju, se izvede povsem samodejno. Za lepši videz tako za ozadje določimo razred CSS, ki je podrobneje predstavljen v nadaljevanju. V zadnjem koraku moramo poskrbeti za delovanje drugih dveh gumbov na prikazanem opozorilu, torej »Akcija 1« in »Akcija 2«. Da dosežemo tipično delovanje v načinu AJAX, ki ne zahteva ponovnega nalaganja celotne strani, moramo vpisano kodo HTML obdati z gradnikom AJAX, imenovanim `UpdatePanel`. Na takšen način omogočimo spreminjanje vsebine znotraj tega gradnika, in to brez ponovnega nalaganja strani.

C# (priloga 3, Slika 8):

- Obseg strežniške kode, ki jo moramo napisati za pravilno delovanje rešitve, je v primeru načina AJAX najmanjši. Prikazovanje in skrivanje opozorila namreč omogoči prednastavljena JavaScript skripta. Po eni strani tako v strežniški kodi tega ni potrebno posebej programirati, po drugi pa nobenim gradnikom ni treba pripenjati JavaScript funkcij. V strežniški kodi moramo tako določiti samo želeno kodo, ki naj se izvede po kliku ene izmed akcij v opozorilu. Ob koncu vsake akcije poskrbimo še za skrivanje opozorila (Slika 27).

Slika 27: Prikaz skrivanja opozorila, prikazanega v načinu AJAX

```
//skrijemo opozorilo
ModalPopupExtender.Hide();
```

CSS (priloga 3, Slika 9):

- Tudi CSS je v tem primeru mnogo preprostejši. Vse, kar moramo narediti, je, da poskrbimo za povsem oblikovne lastnosti opozorila in ozadja, ki se prikaže skupaj z njim. Opozorilu tako določimo obrobo (`border`), barvo (`background-color`), velikost (`width, height`) in poravnavo besedila (`text-align`). Za ozadje, prikazano ob sporočilu, pa določimo samo barvo (`background-color`) in raven prozornosti (`filter, -moz-opacity`).

Pravkar izvedena rešitev je v mnogih pogledih boljša od prejšnjih dveh. Prikazovanje in skrivanje opozorila se izvede brez komunikacije s strežnikom s prednastavljeno JavaScript skripto. Ta funkcija je že vgrajena v gradnik `ModalPopupExtender`, zaradi česar je naloga

skoraj trivialna. Po kliku gumba »Akcija 1« ali »Akcija 2« pa se strežniška koda izvede brez ponovnega nalaganja strani. Uporabnik komunikacije s strežnikom sploh ne opazi, tako da je delovanje v mnogih pogledih podobno delovanju namiznih aplikacij.

Še najpomembnejše pa je dejstvo, da se opozorilo dejansko prikaže v modalnem načinu. Ko je opozorilo vidno, uporabnik nikakor ne more sprožiti gumba »Test«, tudi s tipko »Tab« ne. V gradnik je namreč vgrajena tudi JavaScript funkcija, ki, dokler opozorilo ostane vidno, poskrbi, da se navigacija po gradnikih s tipko »Tab« onemogoči.

Če analiziramo zahtevnost izvedbe, ugotovimo, da je ta dokaj preprosta. Še največ težav povzroča morebitno nepoznavanje tehnologije in njenih značilnosti. Vsekakor pa je ta podatek zanemarljiv, saj je krivulja učenja v tem primeru zelo strma in osvojitev zahtevanega znanja poteka izjemno hitro. Prav zaradi teh lastnosti je tudi časovna potratnost mnogo manjša. To predvsem velja, če jo primerjamo z drugo metodo, pri kateri je treba pisati lastne JavaScript funkcije in poskrbeti za njihovo pravilno delovanje.

Končni rezultat je precej boljši kot v prejšnjih primerih. Delovanje je tekoče, brez motečega utripanja zaslonske slike. Uporabniku tako omogočimo podoživljanje delovanja namiznih aplikacij, kar je zagotovo eden glavnih ciljev današnjih spletnih aplikacij.

Tabela 3: Ocena nalog, izvedenih z metodo AJAX

Naloga\lestvica	***1***	***2***	***3***	***4***	***5***
Zahtevnost JavaScript skript	x				
Zahtevnost izvedbe		x			
Časovna potratnost		x			
Končni rezultat	★★★★★				

Legenda: 1 pomeni najmanj, 5 največ

3 ASP.NET AJAX z vidika drugih subjektov

3.1 ASP.NET AJAX z vidika uporabnika

Z vidika uporabnika ja najpomembnejše, da spletna aplikacija deluje hitro in odzivno, torej brez večjih prekinitev oziroma čakanja na nalaganje strani, in da je njena uporaba intuitivna. To je posledica pričakovanj, ki jih imajo na podlagi delovanja nespletnih oziroma namiznih aplikacij. Te aplikacije pa delujejo na povsem drugačnem principu, predvsem pa najpogosteje delujejo lokalno, torej na računalniku posameznega uporabnika. Posledično je njihovo delovanje, razen v izjemnih primerih, hitro in brez prekinitev. Ker pa so se uporabniki najprej srečali s takimi aplikacijami, je zanje njihov način delovanja postal samoumeven in pričakovan. Ob pojavu spletnih aplikacij tako večina uporabnikov pričakuje enako hitrost in odzivnost ter intuitivno uporabo. Zaradi popolnoma drugačnega

principa delovanja spletnih aplikacij pa je doseganje njihovega pričakovanja izjemno težavno (Housley, 2008).

K odzivnosti in hitrejšemu delovanju je veliko prispevala razširitev uporabe širokopasovnega interneta. Ta omogoča veliko hitrejši prenos podatkov, kar pomeni, da so po eni strani spletne aplikacije v resnici postale aplikacije, po drugi pa se je povečala hitrost njihovega delovanja.

3.1.1 Spletne aplikacije v okolju ASP.NET

Za spletne aplikacije, razvite v standardnem okolju ASP.NET, je značilno delovanje po načelu vnovičnega nalaganja strani. Tako delovanje je za uporabnika moteče, saj pomeni, da moramo po vsakem kliku določenega gumba počakati, da se stran pošlje na strežnik in od tam nazaj v njegov brskalnik, kjer se v celoti ponovno naloži. To se v brskalniku opazi kot nekakšno utripanje zaslonske slike.

Takemu delovanju se lahko izognemo le z uporabo JavaScript skripte, ki omogoča spreminjanje prikazane vsebine brez komunikacije s strežnikom. Za njeno uporabo pa je potrebnega veliko znanja, ki ga marsikateri razvijalec nima, predvsem za izvedbo zahtevnejših opravil. Delo še dodatno otežuje slaba podpora IDE-vmesnikov za programski jezik JavaScript. Zaradi tega je pisanje JavaScript skript omejeno na raven preprostih opravil, za vse druge operacije pa aplikacije še vedno uporabljajo strežniške kode. To privede do utripajoče spletne aplikacije, ki z vidika uporabniške prijaznosti ne zagotavlja primerljivih rezultatov z namiznimi aplikacijami (McClure et al., 2006).

3.1.2 Spletne aplikacije v okolju ASP.NET z uporabo tehnologije AJAX

Z uporabo tehnologije AJAX se utripanju strani izognemo preprosto in hitro. Tehnologija AJAX omogoča osveževanje samo določenih delov spletne aplikacije, in to brez kakršnega koli motečega utripanja ali ponovnega nalaganja strani. Delovanje spletnih aplikacij približamo delovanju navadnih namiznih aplikacij, seveda samo navidezno. V ozadju še vedno ostajajo glavne razlike med delovanjem obeh rešitev, a je vsaj z vidika uporabnika delovanje obeh precej podobno. Prav to je cilj današnjih rešitev – uporabniku ponuditi podoživljanje uporabe spletnih aplikacij na način, kot to že vrsto let omogočajo namizne aplikacije. Ob uporabi tehnologije AJAX je spletna aplikacija za uporabnika precej prijaznejša in domača (Crane, Pascarello & Darren, 2006).

3.2 ASP.NET AJAX z vidika naročnika

Z vidika naročnika je najpomembnejše razmerje med naložbo in funkcionalnostjo ter kakovostjo rešitve. Naročniki poleg natančnega delovanja kot najpomembnejšega dejavnika od razvijalcev rešitve vselej pričakujejo, da bo aplikacija učinkovita in za uporabnika prijazna. Velikokrat veljajo stroga pravila o tem, kaj oziroma kako mora spletna aplikacija delati, in je zaradi tega manevrski prostor na tem področju za razvijalce

precej majhen. Naročnika je v takih primerih mogoče presenetiti le na področju učinkovitosti delovanja in prijaznosti uporabniškega vmesnika.

Z uporabo tehnologije AJAX lahko v razmeroma enakem razvijalnem času naročniku ponudimo precej boljši uporabniški vmesnik. Zadovoljstvo naročnika bo tako večje, saj bo tak uporabniški vmesnik deležen precej bolj naklonjene kritike končnih uporabnikov. Pri uvedbi spletne aplikacije podjetja pa so zadovoljni končni uporabniki bistvenega pomena. Samo v takih primerih se bodo uporabniki za uporabo spletne aplikacije tudi odločali. Z vidika naročnika je namreč najslabše investirati v spletno aplikacijo, ki pa je zaradi slabega uporabniškega vmesnika uporabniki preprosto ne uporabljajo (Arlekar, 2006).

V primeru, da naročnik želi vzpostaviti spletno aplikacijo za interno uporabo (informatizacijo določenega procesa), ima s tehnologijo AJAX še dodatne prednosti, ki jih lahko razdelimo na naslednja področja (White, 2008):

- čas, porabljen za čakanje na prenos podatkov;
- čas, porabljen za izvedbo opravila;
- obremenitev omrežja pri izvedbi celotnega opravila.

Čas, porabljen za čakanje na prenos podatkov, je z uporabo tehnologije AJAX precej krajši. Vsaka akcija uporabnika namreč ne zahteva prenosa celotne vsebine strani, ampak samo njenega dela. Z vsako akcijo uporabnika tako privarčujemo dragoceni čas, ki se (po velikem številu ponovitev) pretvori v denarni prihranek. Zadeva postane še toliko očitnejša, če predpostavimo, da je na primer uporabnikov več tisoč. Hkrati s časovnim prihrankom smo deležni tudi prihranka z vidika obremenjenosti omrežja. Kumulativni prenos podatkov je za izvedbo določenega opravila zaradi istih prednosti manjši in je z večanjem števila uporabnikov še toliko očitnejši. Spet zaradi krajšega časa prenosa podatkov je tudi čas, potreben za izvedbo celotnega opravila, krajši, saj so nam zadeve hitreje na voljo za delo (White, 2008).

Poleg navedenih prednosti lahko z uspešno uporabo tehnologije AJAX na področju uporabniškega vmesnika pridobimo še pri (White, 2008):

- številu korakov za izvedbo opravila;
- znanem in uveljavljenem uporabniškem vmesniku;
- odzivnosti spletne aplikacije.

S pravilno zasnovanim vmesnikom AJAX lahko precej zmanjšamo število korakov, potrebnih za izvedbo določenega opravila. Z manjšanjem števila korakov se precej poveča produktivnost, saj je opravilo izvedeno v krajšem času. Poleg tega manj korakov pomeni tudi manjšo verjetnost za napake, kar se odraža v krajšem času, namenjenem odpravljanju

napak (nepravilnih vnosov). Produktivnost poveča tudi znan uporabniški vmesnik, ki ga lahko uporabniku ponudimo s tehnologijo AJAX. Tako lahko delovanje spletne aplikacije približamo delovanju namiznih aplikacij, s katerimi uporabniki delajo pogosteje. Poleg večje produktivnosti pa je prednost znanega uporabniškega vmesnika tudi ta, da so stroški uvajanja ljudi za delo z aplikacijo nižji, manj je tudi napak, začetna produktivnost uporabnikov pa je večja. AJAX omogoča tudi večjo odzivnost aplikacij, kar spet povečuje produktivnost, saj poleg odstranjevanja potrebe po »čakanju« na aplikacijo omogoča tekoče in neprekinjeno delo (White, 2008).

Kot vidimo, je prednosti z vidika naročnika mnogo, še največ v primeru, ko je spletna aplikacija namenjena za interno uporabo. AJAX namreč na mnogih področjih omogoča znižanje stroškov in povečanje učinkovitosti.

3.3 ASP.NET AJAX z vidika menedžmenta razvojnih projektov ponudnika

Za menedžment razvojnih projektov razvijalskega podjetja je najpomembnejše uspešno poslovanje podjetja, ki pa je odvisno od zadovoljstva zaposlenih (razvijalcev), zadovoljstva naročnika in končnega uporabnika ter seveda od kakovosti izdelanega proizvoda ali storitve. AJAX omogoča povečanje zadovoljstva vseh omenjenih subjektov, torej zaposlenih, naročnika in končnega uporabnika. Primeri, ko se zaradi vpeljave neke novosti v podjetje poveča zadovoljstvo pri vseh akterjih, so izjemno redki. Največkrat se namreč srečujemo z nasprotujočimi si interesi, predvsem med razvijalci in naročniki. Klasičen primer je lahko dogovor o številu ur, ki je razvijalcu na voljo za razvoj rešitve. Velikokrat se na primer naročniki ne zavedajo pomembnosti faze analize pri razvoju, zato želijo zanjo nameniti minimalno število delovnih ur. Takih, nasprotujočih si, interesov bi lahko našli še kar nekaj. Ravno zaradi tega je AJAX nekaj posebnega. Koristi imajo tako razvijalec kot naročnik in tudi končni uporabnik.

3.3.1 Pozitivni vplivi na zaposlene (razvijalce)

Prednosti, ki jih prinaša uporaba tehnologije AJAX, so najočitnejše na področju zadovoljstva končnega uporabnika. Vsekakor pa to ni edina prednost, ki jo prinaša AJAX. Za razvijalce je AJAX val novosti v podjetju. Zanje je precejšnjega pomena, da so naloge, ki jih opravljajo, zanimive in spodbujajoče. Pred prihodom tehnologije AJAX je bil razvoj spletnih aplikacijskih rešitev nekoliko omejen s tehnologijo, ki je bila na voljo. Čeprav je tehnologija za izvedbo nalog na način AJAX obstajala, pa še ni bila v dovolj zreli fazi, da bi jo lahko podjetja množično uporabljala v svojih rešitvah.

Tudi takrat, ko so se razvijalci odločili za uporabo takih metod, so velikokrat naleteli na neodobravanje, predvsem pri naročnikih, ki nikakor niso želeli povečati obsega delovnih ur za razvoj rešitve. Take primitivne rešitve v načinu AJAX pa so zaradi zahtevnosti njihove vpeljave zahtevale kar precej časa. Nasprotujoči si interesi so največkrat povzročili

prilagajanje naročniku in izdelavo spletnih rešitev z do tedaj standardnimi metodami. Po eni strani je to pomenilo zaviranje napredka pri razvoju rešitev, po drugi pa demotiviranje razvijalcev, ki jim je bilo onemogočeno raziskovanje alternativnih metod ter posvečanje učinkovitemu in za uporabnika prijaznemu delovanju.

AJAX razvijalcem povrne veliko svobode, saj v enakem času omogoča izdelavo precej boljše aplikacije z vidika tekočega in za uporabnika prijaznega delovanja. Uporaba nove tehnologije pomeni razbijanje monotonosti delovnih nalog. Poleg tega lahko AJAX uporabimo na več načinov in za različne naloge. Vse te novosti prispevajo k izboljšanju motivacije zaposlenih, kar pripelje do boljših in hitrejših rezultatov. Povečata se tako produktivnost kot tudi dobri odnosi med zaposlenimi. Nova tehnologija vedno znova sproži izmenjavo idej med zaposlenimi, kar poveča komunikacijo med njimi in izboljša timsko delo. Hkrati pa se poveča tekmovalnost med zaposlenimi, ki se pomerijo v tem, kdo bo hitreje in bolje uporabil novo tehnologijo, ki mu je na voljo.

Vse to so izjemno pozitivne spremembe, ki jih je menedžment razvojnih projektov še kako vesel. Pozitivni vplivi vpeljave tehnologije AJAX na zaposlene so torej naslednji:

- povečanje zadovoljstva naročnikov in končnih uporabnikov;
- povečanje motivacije zaposlenih;
- povečanje produktivnosti;
- povečanje komunikacije med zaposlenimi;
- povečanje tekmovalnosti med zaposlenimi.

3.3.2 Pozitivni vplivi na stroške

Podjetje, ki razvije določeno spletno aplikacijo, je lahko zadolženo tudi za njeno gostovanje na svojih strežnikih. V takem primeru ima z uporabo tehnologij AJAX določene prednosti. Že v tretjem poglavju so omenjena področja, na katerih AJAX prinaša prednosti, ki jih opisuje White (2008) v svojem članku. Nekatere izmed njih lahko neposredno pretvorimo tudi v pozitivne vplive na zniževanje stroškov, ki jih ima razvijalsko podjetje z uporabo tehnologije AJAX:

- obremenitev omrežja pri izvedbi celotnega opravila;
- znan in uveljavljen uporabniški vmesnik.

Če se za izvedbo določenega opravila po omrežju prenese manj podatkov, to pomeni manjšo obremenjenost strežnikov in omrežja oziroma manjše zahteve glede zmogljivosti strežnikov in omrežja. To pa za podjetje pomeni nižje stroške.

Z znanim uporabniškim vmesnikom se zmanjšajo nejasnosti uporabnikov glede načina uporabe aplikacije tako s tehničnega kot tudi z vsebinskega vidika. To pomeni, da je

potreba po uporabniški podpori precej manjša, kar pomeni manj časa, namenjenega vodenju uporabnikov po telefonu, odgovarjanje na vprašanja po elektronski pošti in podobno. Prihranki so v večjih podjetjih lahko zelo veliki. Zaradi uporabniške izkušnje, ki jo AJAX ponuja, pa lahko podjetje pričakuje prihranke tudi na povsem drugih področjih. Po navedbah podjetja Tometa Software (2009) se lahko AJAX v določenih primerih popolnoma enači z uporabniškimi vmesniki Flash. Ti omogočajo visoko raven kompleksnosti in interaktivnosti uporabniškega vmesnika. Njihovo oblikovanje pa zahteva strokovnjaka na tem področju, ki je na trgu veliko dražji kot razvijalec ASP.NET AJAX. S tega vidika lahko podjetje z uporabo tehnologije AJAX privarčuje pri stroških dela, saj so ti za enak rezultat precej nižji.

4 Slabosti tehnologije AJAX in ASP.NET AJAX

V prejšnjem delu diplomskega dela so bili predstavljeni primeri, ki dokazujejo, kako lahko AJAX precej prispeva k izboljšanju videza in delovanja spletnih aplikacij. Vendar pa se tudi pri uporabi tehnologije AJAX pojavljajo določene slabosti in pasti, ki so predstavljene v analizi v nadaljevanju diplomskega dela. Analiza je potekala v dveh delih. V prvem je pozornost namenjena slabostim in pastem tehnologije AJAX z vidika univerzalne tehnologije oziroma metod za izdelavo sodobnih spletnih aplikacij. Predstavljene pa so tudi slabosti in pasti, ki se pojavljajo pri uporabi paketov ASP.NET AJAX in AJAX Control Toolkit in so specifične za tehnologijo AJAX, vgrajeno v Microsoftovo okolje ASP.NET.

4.1 Slabosti, ki jih prinaša AJAX načina delovanja

Za AJAX način delovanja so značilne nekatere slabosti in pasti, ki lahko povzročijo, da spletna stran postane za uporabnika manj prijazna. Sprva se lahko to zdi protislovje, saj smo vse do tega trenutka poizkušali prikazati, kako z AJAX načinom dosežemo za uporabnika prijaznejše delovanje, vendar pa to lahko ponazorimo s prikazom konkretnih primerov.

Uporaba tehnologije AJAX je za spletno aplikacijo nevarna, če se zadeve ne lotimo dovolj premišljeno in previdno. Vsekakor pa AJAX načina ne moremo in ga tudi ni dobro uporabljati v vsakršni situaciji, vsaj ne brez pravilnega načrta njegove vpeljave.

Že sama uporaba tehnologije AJAX je povezana z določenimi slabostmi, na katere moramo biti pozorni. Najpogostejše med njimi so naslednje (Bosworth, 2008):

- onemogočena osnovna funkcionalnost gumba Nazaj (angl. *Back*) pri brskalnikih;
- počasno delovanje pri slabših internetnih povezavah;
- deset odstotkov brskalnikov delovanja tehnologije AJAX ne podpira;
- čezmerna uporaba povzroči veliko komunikacije s strežnikom, kar lahko upočasni odzivnost;

- zaradi dinamičnega spreminjanja strani uporabniki trenutnega pogleda ne morejo shraniti med zaznamke;
- če ima uporabnik izključeno izvajanje JavaScript skript, je delovanje tehnologije AJAX onemogočeno.

Če navedene slabosti analiziramo podrobneje, ugotovimo, da so zares relevantne le nekatere. Glede na zdaj že množično uporabo širokopasovnega interneta bi lahko dejali, da je počasno delovanje pri slabših internetnih povezavah precej irelevantno. Podjetja namreč že dalj časa v svojih spletnih aplikacijah ponujajo vsebine, prilagojene širokopasovnemu internetu. Tudi podatek, da deset odstotkov brskalnikov tehnologije AJAX ne podpira, ni tako zaskrbljujoč, glede na to, da ta deluje v vseh najpogosteje uporabljenih brskalnikih, kot so Internet Explorer, FireFox in Opera. Podobno velja tudi za uporabnike, ki v brskalnikih izključijo delovanje JavaScript skript. Po objavljenih raziskavah se je delež uporabnikov, ki izključijo delovanje JavaScript skript, od leta 2000 do 2008 z dvajsetih odstotkov znižal na pet (w3scools.com, 2009). Glede na padajoči trend je tako mogoče pričakovati, da bo ta delež še naprej padal. Glede počasne odzivnosti pri čezmerni uporabi je zagotovo res, da je teoretično to mogoče, a je največ odvisno od razvijalca in njegovega načina programiranja. Dober razvijalec namreč zna oceniti količino gradnikov AJAX, ki jih je še smiselno zbrati na eni strani.

Bolj problematične so druge slabosti, še najbolj je za uporabnika moteče nepravilno delovanje gumba »Nazaj«, ki naj bi ga popeljal v stanje pred izvedbo zadnje akcije. Pri uporabi načina AJAX omenjeni gumb ne deluje pravilno, saj akcije oziroma spremembe, izvedene v tem načinu, niso obravnavane kot dejanska akcija uporabnika. Brskalniki kot akcijo prepoznajo samo celotno pošiljanje strani na strežnik, kar pa se pri načinu AJAX ne zgodi.

Tudi nezmožnost shranjevanja trenutnega pogleda strani pod zaznamke je za uporabnika moteča. V načinu AJAX se izvedene spremembe zgodijo po tem, ko se stran že naloži. To pomeni, da je novi pogled drugačen od osnovne strani, saj se je z uporabniškimi dejanji spreminjal. Ob shranjevanju med zaznamke pa se v brskalniku shrani le naslov URL, ki vodi do določene spletne strani. V praksi bi to pomenilo, da bi po kliku zaznamka pravzaprav spet prišli na osnovno stran, torej tako, na kateri še ni bilo sprememb.

Danes so že mogoči določeni triki in metode, kako lahko uporabniku povrnemo osnovne funkcionalnosti gumba »Nazaj« in možnost pravilnega zaznamka strani. Rešitve pa so vse prej kot preproste, tako da se razvijalci le pri redkih primerih temu v resnici posvetijo. Najpogosteje se morajo tako uporabniki kar sprijazniti z nevšečnostmi, ki jih prinaša AJAX, v korist preprostejšemu in bolj tekočemu delovanju.

Poleg splošnih slabosti in pasti, ki jih moramo pred uporabo tehnologije AJAX dobro poznati, pa dodatne težave lahko povzroči tudi njegova nepravilna oziroma nepremišljena

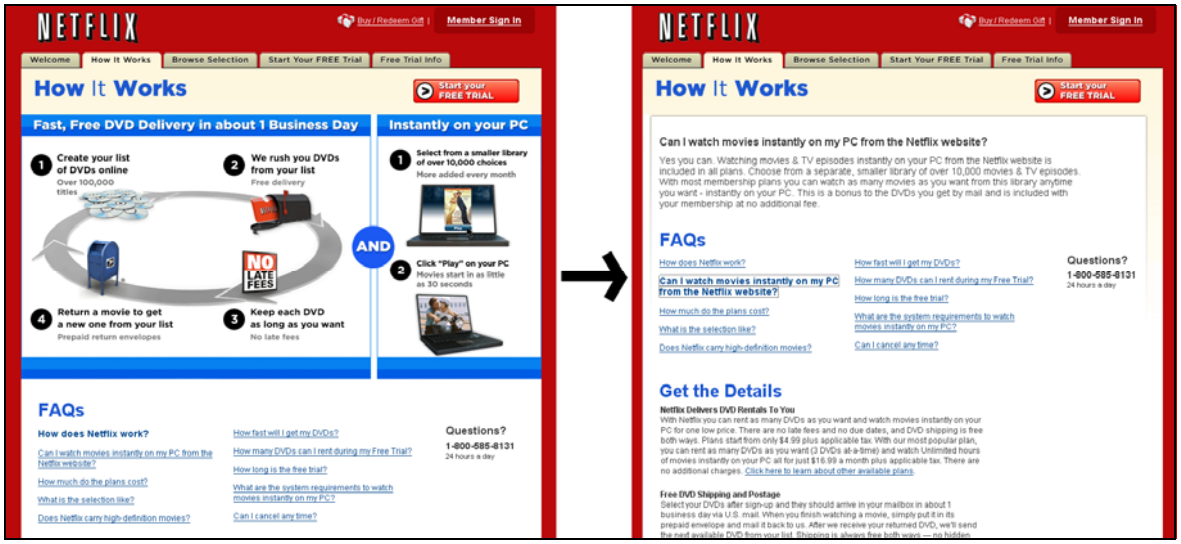
vpeljava v spletno aplikacijo. Za lažjo predstavo je v nadaljevanju predstavljenih nekaj konkretnih primerov takih vpeljav.

4.1.1 Primer 1

Kot je bilo omenjeno že v prejšnjih poglavjih, AJAX omogoča spreminjanje vsebin v spletni aplikaciji, ne da bi za to potrebovali ponovno osveževanje celotne kode HTML. Spremembe zaslonske slike tako nastanejo hitro in brez opozorila. Nepravilna oziroma nepopolna uporaba tehnologije AJAX lahko zato uporabnika zmede, saj se ne zaveda povsem, kaj se je pravzaprav na strani zgodilo.

Primer takega delovanja je na spletni strani www.netflix.com (Slika 28), na kateri so odgovori na pogosto zastavljena vprašanja. Ob prvem nalaganju strani se prikaže začetna stran, ki v zgornjem delu vsebuje sliko predstavitve delovanja strani. Pod predstavitveno sliko je seznam najpogostejših vprašanj, ki jih lahko, če želimo dobiti odgovor, kliknemo. S klikom določenega vprašanja se prej omenjena slika takoj skrije, namesto nje pa se izpiše odgovor na pravkar izbrano vprašanje. Ker pa ponujeni odgovor na strani zaseda manj vertikalnega prostora kakor pred njim slika, se celotna stran tudi premakne, in sicer navzgor. Tako delovanje uporabnika zmede, saj zaradi hitre izvedbe sploh ne ve, kaj se je pravzaprav zgodilo. Podobno se zgodi, če z miško kliknemo kako drugo vprašanje, ki je na voljo, le da se v tem primeru besedilo v ponujenem odgovoru le zamenja. Rezultat ostane isti, saj se dolžina odgovorov razlikuje in vedno pride do podobnega »premikanja« strani.

Slika 28: Prikaz spremembe, izvedene s tehnologijo AJAX



Vir: How It Works, b.l.

Taka uporaba tehnologije AJAX je za mnoge poznavalce nepravilna. Vsebina, ki jo trenutno uporabljamo, v predstavljenem primeru je to seznam najpogostejših vprašanj, se med posameznimi spremembami v načinu AJAX ne sme premakniti iz izvirne lege. Le tako bo uporabniku jasno, da sprememba pomeni samo prikaz pravkar izbranega vprašanja. Alternativa bi lahko bila animirana sprememba. Prvotna slika bi morala počasi izginiti in

odgovor na izbrano vprašanje bi se moral na zaslonu pojavljati postopoma. Premik strani bi bil tako viden za uporabnika, ki bi mu bilo tudi razumljivo, kaj se je zgodilo.

4.1.2 Primer 2

Drugi primer nepravilne uporabe tehnologije AJAX je zanemarjanje uporabnika ob daljšem nalaganju vsebine. Ko uporabnik s klikom sproži določeno spremembo spletne aplikacije, mora biti o tem obveščen. Še posebej to velja, če posodabljanje strani traja dalj časa. V nasprotnem primeru bo stran za uporabnika delovala, kakor da se ni zgodilo nič. Pri takih daljših opravilih v načinu AJAX moramo uporabnika vedno opozoriti na delovanje, ki se odvija v ozadju. To najelegantneje naredimo s prikazom napisa »Nalagam …« in (ali) s prikazom preproste animacije, ki ponazarja nalaganje sprememb. Tak način uporablja tudi Gmail ob sprožitvi iskanja elektronske pošte po vpisanem nizu. Ko se iskanje sproži, se v zgornjem delu strani prikaže polje z napisom »Loading …«, ki uporabnika opozori na iskanje v teku (Slika 29). Če opozorila ne bi prikazali, uporabnik ne bi vedel, ali se je iskanje sprožilo ali ne. Posledica bi lahko bile večkratne ponovne sprožitve iskanja, dokler uporabnik sam ne bi ugotovil, da se je iskanje pravzaprav že sprožilo in da mora za rezultat iskanja le počakati.

Slika 29: Prikaz obveščanja uporabnika ob komunikaciji s strežnikom

Vir: Gmail, b.l.

4.2 Slabosti paketov ASP.NET AJAX in AJAX Control Toolkit

Poleg splošnih slabosti, ki jih prinaša delovanje spletnih aplikacij v načinu AJAX, pa so značilne tudi določene slabosti pri paketih, ki ponujajo gradnike AJAX. V diplomskem delu je pozornost namenjena paketoma ASP.NET AJAX in AJAX Control Toolkit kot najbolj uporabljenima med razvijalci, zato je v tem razdelku pozornost prav tako namenjena zgolj njima.

V prejšnjih poglavjih so bili predstavljeni gradniki iz paketa ASP.NET AJAX. Za enega najuporabnejših smo označili gradnik UpdatePanel. Vsi gradniki, postavljeni znotraj njega, se spremenijo v gradnike AJAX, torej njihovo spreminjanje ne zahteva ponovnega nalaganja celotne strani. Spremembe se namreč izvedejo z uporabo objekta XMLHttpRequest na podlagi JavaScript skripte. Prednosti takega delovanja je več, značilne pa so tudi določene slabosti.

4.2.1 Primer 1

Pravilno delovanje gradnikov znotraj gradnika UpdatePanel omogočajo različne JavaScript funkcije na odjemalčevi strani. Vsaka ima določeno ime in vhodne parametre, uporablja pa tudi določene globalne spremenljivke. Do težav lahko pride, ko želimo v UpdatePanel postaviti gradnik, ki ni Microsoftov in deluje na principu lastnih JavaScript funkcij. Te se lahko imensko ujemajo z že obstoječimi JavaScript funkcijami, namenjenimi delovanju v načinu AJAX. V takem primeru gradnik ne bo deloval, kot bi moral. Do težav pride tudi, kadar nove JavaScript funkcije uporabljajo enake globalne spremenljivke kot gradniki AJAX, zato so vrednosti v njih napačne. V takih primerih je rezultat najpogosteje nepravilno delovanje novega gradnika.

Odličen primer za prikaz takega nepravilnega delovanja je gradnik, namenjen urejanju datotek HTML, in sicer FreeTextBox. Oblikovno je gradnik zelo podoben najpopularnejšemu namiznemu urejevalniku besedil Microsoft Word (glejte prilogo 4, Slika 10). Omogoča preprosto pisanje in oblikovanje besedila tako v načinu Microsoft Word kot tudi v načinu HTML. Njegovo delovanje v celoti temelji na JavaScript funkcijah zato sta delovanje in odzivnost podobna kot v urejevalnikih za domačo rabo.

Ko pa poizkušamo gradnik FreeTextBox vstaviti v UpdatePanel, naletimo na nepričakovane težave. Najprej opazimo deformacijo gumbov za urejanje vpisanega besedila, ki postanejo mnogo večji, in tudi popolno nedelovanje funkcionalnosti gradnika FreeTextBox. Po kliku posameznega gumba za urejanje se ne zgodi nič. Po podrobnejši analizi ugotovimo, da ob posameznih klikih gumbov naletimo na napako v izvedbi JavaScript funkcije, ki naj bi zagotavljala pravilno delovanje gradnika FreeTextBox. Pride do konflikta med delovanjem JavaScript funkcije, ki omogoča pravilno delovanje gradnika UpdatePanel in JavaScript funkcijo, ki zagotavlja pravilno delovanje gradnika FreeTextBox. V takih primerih je edina možnost, da gradnik iz gradnika UpdatePanel umaknemo.

4.2.2 Primer 2

V drugem primeru pravzaprav ne gre za slabost, temveč bolj za omejitev uporabe gradnikov AJAX v ASP.NET. Pod to kategorijo spada na primer gradnik FileUpload, s katerim lahko uporabniku omogočimo izbiro datoteke, ki jo želi prenesti na strežnik.

Slika 30: Primer gradnika FileUpload

Ko poizkušamo omenjeni gradnik uporabiti v gradniku UpdatePanel, spet naletimo na nepričakovane težave. S takim gradnikom namreč datoteke nikakor ne moremo naložiti na strežnik. Tako delovanje pa je pri Microsoftu načrtno in temelji na varnosti. Vsi gradniki znotraj gradnika UpdatePanel s strežnikom komunicirajo na podlagi JavaScript funkcij, kar pa bi v tem primeru pomenilo veliko varnostno luknjo. Če bi Microsoft v gradniku UpdatePanel omogočil tudi uporabo gradnika FileUpload, bi to pomenilo, da bi se datoteke od odjemalca do strežnika prenašale na podlagi JavaScript funkcij. Tako delovanje pa bi bilo zelo neverno, saj bi lahko izkušeni hekerji (angl. *hackers*) to izkoristili za napade na strežnik. Prav zaradi tega je uporaba gradnika FileUpload znotraj gradnika UpdatePanel onemogočena.

5 Primeri uporabe tehnologije AJAX v spletu

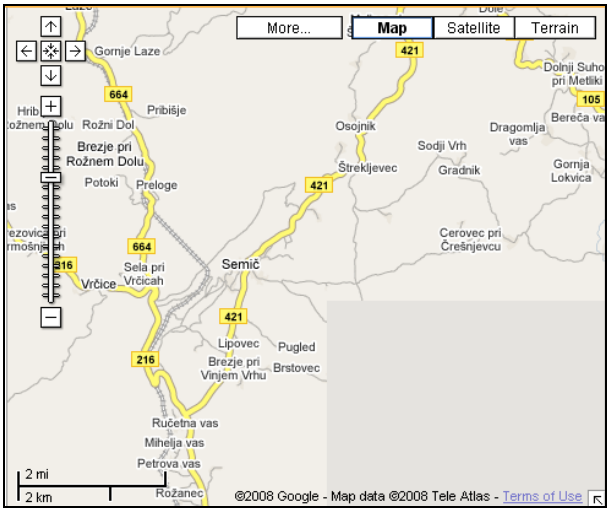
V spletu je že mnogo spletnih rešitev, ki za komunikacijo s strežnikom izkoriščajo način AJAX. Nekatere od teh strani so bile že velikokrat uporabljene, ne da bi vedeli, da je v ozadju AJAX. Določene strani so način delovanja AJAX uporabljale celo pred pojavom samega pojma AJAX, ki se zdaj uporablja za označevanje takega delovanja. V primerih v nadaljevanju je predstavljena tipična uporaba načina AJAX v sodobnem spletu, na kratko pa so analizirane tudi prednosti posamezne uporabe.

5.1 Google maps

Najznačilnejši primer uporabe tehnologije AJAX je Google maps. Čeprav je XMLHttpRequest izumil Microsoft, pa ima največje zasluge za hitro širitev zanimanja za AJAX prav Google. Lahko bi trdili, da je Google na tem področju pionir, saj je metode komunikacije s strežnikom v načinu AJAX uporabljal že pred pojavom termina AJAX.

Spletna aplikacija Google maps v načinu AJAX omogoča nemoteno sprehajanje z miško po zemljevidu. Poleg tega omogoča prikazovanje poti do določenega cilja, upoštevajoč različne omejitve, kot so samo avtoceste, samo lokalne ceste in podobno. Celotno delovanje je, zahvaljujoč komunikaciji s strežniki na podlagi tehnologije AJAX, tekoče in deluje brez ponovnega nalaganja strani.

Slika 31: Googlov prikazovalnik zemljevidov



Vir: Google maps, b.l.

5.2 Gmail

Tudi Gmail, ki je prav tako izdelek Googla, za izboljšanje delovanja uporablja AJAX. Zaradi popularnosti smo se najverjetneje že vsi srečali z vmesnikom Gmail, ki omogoča izvedbo skoraj vseh opravil brez ponovnega nalaganja strani. Brskanje po elektronskih sporočilih je tako hitrejše in veliko bolj tekoče.

5.3 NetFlix

NetFlix je spletna aplikacija, ki uporabnikom omogoča izposojo filmov na DVD-jih. Na strani lahko brskamo po seznamu vseh filmov, pregledujemo lestvice najbolj gledanih filmov in jih tudi ocenjujemo.

Na strani je AJAX vključen tako, da se ob pomiku s kazalcem miške na določen naslov filma prikaže oblaček s kratkim opisom in z njegovo oceno. Tako brskanje po filmih je veliko hitrejše, saj za opis filma ni treba zapustiti trenutne strani, temveč lahko to storimo kar na isti strani. To je zelo dober primer, kako lahko s tehnologijo AJAX zelo preprosto precej izboljšamo uporabnikovo izkušnjo.

Slika 32: Netflixovo ponujanje dodatnih informacij uporabniku

Vir: Netflix Top 100, b.l.

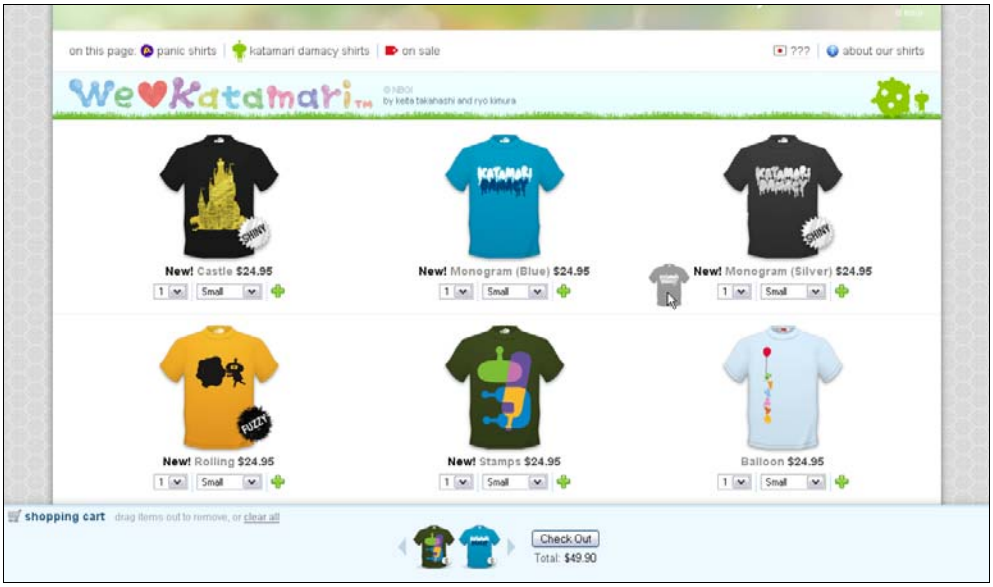
5.4 A9

Spletna aplikacija A9 je spletni brskalnik s podobno uporabnostjo kot mnogi drugi spletni brskalniki, na primer Google.com, njegova posebnost pa je prikazovanje zadetkov ob spremembi iskalnih pogojev. Če želimo na primer v zadetke vključiti tudi iskanje po knjigah, to naredimo s klikom ustreznega polja. V trenutku, ko to izvedemo, se stran posodobi in prikažejo se dodatni zadetki, seveda vse brez ponovnega nalaganja. Enako velja, če želimo določene zadetke s seznama odstraniti. Z miško odznačimo želeno polje in zadetki se odstranijo s seznama.

5.5 Panik Goods

Stran predstavlja spletno trgovino, ki ponuja različne majice s kratkimi rokavi. Specifična lastnost strani je možnost povleci in spusti (angl. *drag-drop*), s katero želeno majico v nakupovalni voziček na dnu strani preprosto prenesemo z miško. Delovanje strani je tekoče in zabavno, kar pritegne marsikaterega uporabnika.

Slika 33: Prikaz možnosti povleci in spusti v spletni trgovini



Vir: Nice T-Shirts For You, b.l.

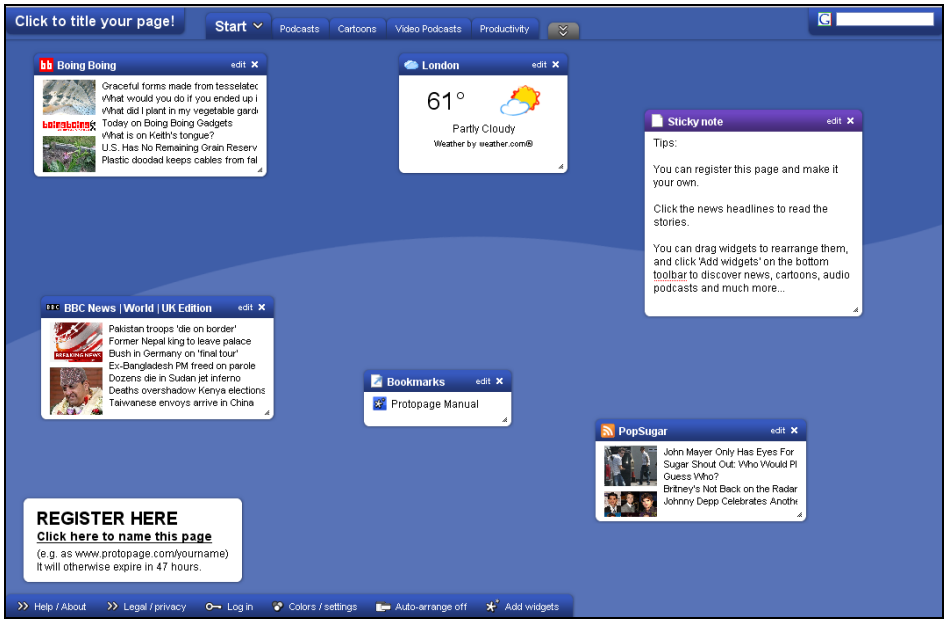
5.6 AJAXtrans

Naslednja spletna aplikacija je spletni slovar, ki omogoča prevod vpisanih besed in tudi celotnega stavka v realnem času (angl. *real-time*). Delovanje je preprosto, izberemo vhodni in izhodni jezik ter začnemo pisati. Že med pisanjem se v oknu za izhodni jezik prikazuje prevod glede na trenutno vpisane besede v vhodnem jeziku (glejte prilogo 4, Slika 11).

5.7 Protopage

Naslednji primer je povezan s trendom, ki se vse bolj širi po spletu, to je osebni spletni portal. Vedno več je ponudnikov brezplačnih osebnih spletnih portalov, ki omogočajo kreacijo in personalizacijo osebnega spletnega portala. Osebni portal je vstopna točka v svet spleta, na katero lahko naložimo vrsto za nas zanimivih povezav in informacij. Večina omogoča tudi poljubno razporejanje na zaslonu, tako da lahko izdelamo povsem personalizirano vstopno točko v svetovni splet.

Slika 34: Protopageev osebni spletni portal



Vir: Protopage, b.l.

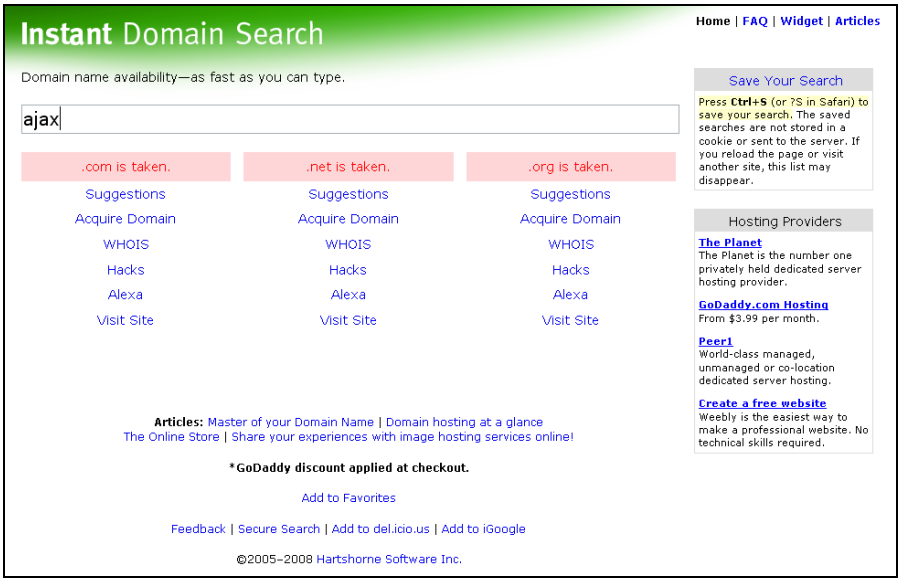
5.8 Instant domain search

Spletna aplikacija s tehnologijo AJAX na zelo preprost in učinkovit način reši eno večjih težav, ki nas doleti ob registraciji nove domene. Dandanes je v spletu objavljenih že več kot 21 milijard strani (WebHosting.Info, 2009), prijavljenih domen, na primer www.google.com, pa je več kot sto milijonov (Kunder, 2009). Kdor želi izdelati spletno stran, mora najprej izbrati domeno, ki še ni v uporabi.

Brez uporabe tehnologije AJAX je iskanje prostega imena za domeno zelo zamudno opravilo. Za vsako domeno, ki jo želimo preveriti, moramo najprej v polje vpisati domeno in nato s klikom gumba sprožiti ponovno nalaganje strani, kjer je mogoče preveriti, ali je vpisana domena prosta ali ne. Če domena še ni prosta, moramo celoten postopek ponoviti od začetka.

Spletna aplikacija Instant Domain Search postopek precej olajša. Želeno domeno vpišemo v polje in že med pisanjem nas spletna aplikacija obvešča, ali je vpisana domena prosta ali ne. S tako rešitvijo, pa čeprav je videti sila preprosta, lahko privarčujemo ure in ure iskanja.

Slika 35: Delovanje spletne aplikacije Instant Domain Search



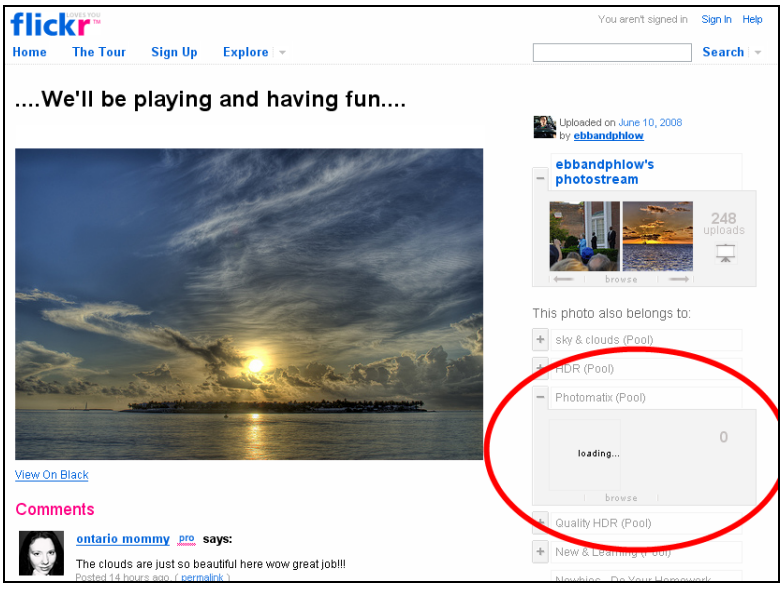
Vir: Domain Name Search, b.l.

5.9 Flickr

Flickr je znana spletna aplikacija, ki uporabnikom nudi možnost gostovanja slik in video posnetkov. Uporabnik izdela spletno stran, kjer lahko objavlja slike in video posnetke. Spletna aplikacija je zelo priljubljena med fotografi, saj omogoča izmenjavo pohval in pripomb ter vsesplošno medsebojno komunikacijo.

Stran na prvi pogled ne deluje ravno kot tipična stran, oprta na tehnologijo AJAX. Funkcionalnosti v načinu AJAX so namreč združene v celoto in tako ne preveč izstopajoče. Tak primer je na primer pregled podrobnosti o sliki. Za pregled podrobnost kliknemo in že se opisno okno razširi in iz strežnika prebere ustrezno besedilo.

Slika 36: Prikaz osveževanja vsebine na Flickrjevi strani



Vir: We'll be playing and having fun, b.l.

5.10 Drugi primeri

Takih in podobnih primerov spletnih aplikacij, ki učinkovito uporabljajo tehnologijo AJAX, je v spletu vedno več. Zanimiv primer je spletna aplikacija, ki omogoča grafično podprto izdelavo načrta zbirke podatkov, ali pa spletna aplikacija, ki omogoča risanje preprostejših slik v obliki slikovnega urejevalnika. Kot zanimivejše primere si je tako vredno ogledati še naslednje spletne aplikacije:

- http://code.communityserver.org/;
- http://www.componentart.com/webui/demos/demos_ajax-techniques/technique1/default.aspx;
- http://www.mdeeng.com/netview/.

Nekatere izmed naštetih spletnih aplikacij so pravzaprav samo nekakšni osnutki ali prototipi, so pa zaradi tega toliko zanimivejše, in to prav zaradi inovativne uporabe tehnologije AJAX.

6 Alternative paketu ASP.NET AJAX

Glede na izjemno popularnost delovanja spletnih aplikacij v načinu AJAX so se na trgu pojavili številni alternativni načini, kako v okolje ASP.NET tako delovanje tudi vpeljati. Za Microsoftov paket ASP.NET AJAX je značilno delovanje, ki temelji na strežniški kodi. To pomeni, da se v najpogostejših primerih razvijalcu za vpeljavo tehnologije AJAX v spletne aplikacije ASP.NET ni treba ukvarjati s pisanjem JavaScript skript. Gradniki paketa namreč že vsebujejo kodo, ki omogoča pravilno izvajanje tudi na odjemalčevi strani. Za razvijalca je dovolj, da pozna osnove jezika JavaScript.

Poleg paketa ASP.NET AJAX pa lahko delovanje v načinu AJAX v spletno aplikacijo vpeljemo tudi z drugimi paketi. Ti se razlikujejo po tem, ali temeljijo na strežniški kodi ali kodi na odjemalčevi strani, torej na JavaScript skriptah.

6.1 Platforme, temelječe na strežniški kodi

Platforme, katerih delovanje temelji na strežniški kodi, so za razvijalca ASP.NET bolj intuitivne. Tako razvijanje aplikacij je namreč značilno že pri standardnih spletnih aplikacijah ASP.NET. Pri uporabi tehnologije AJAX je tako za razvijalca manj težav pri prilagajanju načina programiranja, predvsem pa se izogne pisanju kode v jeziku JavaScript. To je vsekakor velika prednost, saj je razvijanje kompleksnejših rešitev v jeziku JavaScript izjemno težavno. K temu še najbolj prispeva slaba podpora uporabniških vmesnikov, ki bi učinkovito podpirali razvoj in razhroščevanje kode v jeziku JavaScript. Razvijalci si morajo zato pri programiranju v jeziku JavaScript pomagati s številnimi različnimi orodji, kar pa je zelo zamudno. Razvoj aplikacij AJAX, temelječih na strežniški kodi, je tako precej lažji, predvsem pa hitrejši. Razvijalcu ni treba spremeniti načina programiranja, saj ostaja princip razvoja enak.

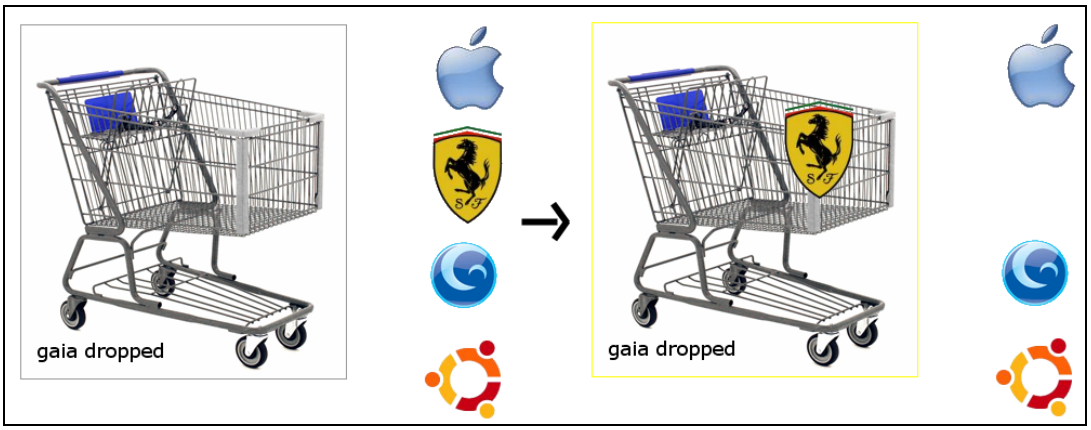
Na trgu se je, kot alternativa paketu ASP.NET AJAX, pojavilo več različnih rešitev (Zeiss, 2006), kot so:

- ComfortASP.NET,
- MagicAjax,
- Anthem.NET,
- Ajax.NET Professional,
- Gaia AJAX Widgets (plačljivi paket).

Za vse navedene rešitve velja, da je vpeljava v okolje ASP.NET hitra in preprosta. Vsaka se vpeljave delovanja AJAX v spletno aplikacijo loteva drugače. Nekatere rešitve, podobno kot paket ASP.NET AJAX, vsebujejo svoj nabor gradnikov AJAX gradnikov, ki jih razvijalec lahko uporabi.

V paketu Gaia AJAX Widgets sta tako na primer gradnika AspectDraggable in AspectDroppable, ki omogočata preprosto vpeljavo povleci in spusti (angl. *drag-drop*) delovanja v spletno aplikacijo.

Slika 37: Primer delovanja povleci in spusti z gradniki Gaia AJAX Widgets



Vir: Ajax Shopping Cart with Gaia, b.l.

Druge rešitve, na primer ComofortASP.NET, ne ponujajo nabora gradnikov, ampak celotno delovanje spletne strani ASP.NET spremenijo v delovanje v načinu AJAX. Vsaka komunikacija s strežnikom se samodejno izvede v načinu AJAX, pri čemer se izognemo ponovnemu nalaganju strani.

Vsekakor je kakovostnih alternativ paketu ASP.NET AJAX veliko. Težave pa lahko povzroča prav dejstvo, da gre za alternative. Rešitve namreč ne omogočajo nikakršne podpore v primeru težav in nezdružljivosti. Tudi njihov nadaljnji razvoj je vprašljiv. Prav zaradi teh značilnosti rešitev ne moremo uporabiti v kakih pomembnejših aplikacijah. V

primeru težav se lahko zgodi tudi to, da je treba določene gradnike zamenjati ali odstraniti, kar pa bi lahko povzročilo veliko ur dodatnega dela.

6.2 Platforme, temelječe na odjemalčevi kodi

Platforme AJAX lahko poleg na strežniških kodah temeljijo tudi na kodi na odjemalčevi strani, na primer v jeziku JavaScript. To pomeni, da moramo delovanje spletne aplikacije na način AJAX implementirati v JavaScript skriptah. Taki paketi zahtevajo od razvijalca veliko več dela in znanja. Zelo dobro se mora namreč spoznati na programski jezik JavaScript. Pri takih rešitvah dosežemo delovanje AJAX z uporabo prednastavljenih funkcij v jeziku JavaScript, ki jih platforme ponujajo. Najznačilnejši predstavniki takega delovanja so:

- Prototype,
- Script.aculo.us,
- Dojo,
- DWR.

Od slednjih sta se še najbolj uveljavila Prototype in Dojo, zato je eden izmed njiju v nadaljevanju predstavljen podrobneje.

Prototype

Prototype deluje na principu pošiljanja parametrov na določeno internetno stran. Na strežniku se nato parametri preberejo in na podlagi njih se vrnejo vrednosti, na osnovi katerih se spremeni trenutni pogled strani. Primer delovanja je izpis občine na podlagi vpisane poštne številke, ki je predstavljen v članku »AJAX Tutorial with Prototype« Peta Freitaga (Freitag, 2007).

Na strani sta gradnik TextBox za vpis poštne številke in prostor, kamor bomo izpisali ime občine. Za TextBox določimo JavaScript funkcijo, ki naj se izvede ob vpisani številki, natančneje ob dogodku `onkeyup` (vrstica 25). V tej funkciji najprej določimo naslov strani, kamor naj se izvedejo poizvedbe (vrstica 9) in nato preberemo vpisano številko (vrstica 10). Naslednji korak je pošiljanje prebranih parametrov na strežnik preko metode Ajax.Updater, ki ji še prej določimo, kam naj se vrnjena vrednost zapiše (vrstica 12).

Slika 38: Primer preprostega programiranja v platformi Prototype

```
<html>
<head>
<title>AJAX Zip Checker </title>
<link rel="stylesheet" href="style.css" type="text/css" />
<script src="prototype.js" language="JavaScript" type="text/javascript"></script>
<script type="text/javascript" language="JavaScript">
function checkZip() {
if($F('zip').length == 5) {
var url = 'checkZip.cfm';
var params = 'zip=' + $F('zip');
var ajax = new Ajax.Updater(
{success: 'zipResult'},
url,
{method: 'get', parameters: params, onFailure: reportError});
}
}
function reportError(request) {
$F('zipResult') = "Error";
}
</script>
</head>
<body>
<form>
<label for="zip">zip:</label>
<input type="text" name="zip" id="zip" onkeyup="checkZip();" />
<div id="zipResult"></div>
</form>
</body>
</html>
```

Vir: AJAX Tutorial with Prototype, 2007.

Ko osvojimo novi način dela, je izvedba preproste poizvedbe na strežniku precej preprosta. Izvedba kompleksnejše rešitve pa bi zahtevala bistveno zahtevnejši postopek.

Čeprav platforme AJAX, ki temeljijo na odjemalčevi kodi, zahtevajo kar nekaj znanja jezika JavaScript, pa so zanje značilne tudi prednosti. Zagotovo imamo pri takem načinu programiranja veliko večji pregled nad dogajanjem v odjemalčevi kodi. To omogoča optimizacijo določenih opravil, saj lahko posamezno JavaScript funkcijo popolnoma prilagodimo specifični nalogi. Platforme, ki temeljijo na strežniških kodah, imajo namreč take JavaScript funkcije že vnaprej napisane in se razvijalec z njimi le redko sreča. Prav zaradi tega so velikokrat prezapletene glede na preprostost naloge, ki jo želimo izvesti.

Razmerje med vloženim časom in rezultatom je zagotovo na strani strežniških platform AJAX, saj so, ne glede na njihovo, v določenih primerih prezapleteno delovanje, na odjemalčevi strani veliko hitrejši pri vpeljavi načina AJAX v aplikacijo. Po drugi strani pa velja poudariti stopnjo učinkovitosti, ki jo lahko dosežemo z odjemalčevimi platformami.

SKLEP

Svetovni splet že leta navdušuje podjetja zaradi preprostega načina promoviranja poslovanja. Vedno več podjetij namizne aplikacije prenaša v spletno okolje, predvsem zaradi za uporabnike lažjega dostopa, ki ne zahteva nikakršne namestitve. Ena večjih pomanjkljivosti spletnih aplikacij pa je bilo vedno njihovo drugačno delovanje v primerjavi z namiznimi aplikacijami. To še zdaleč ni bilo tako prijazno in intuitivno.

S pojavom paketov ASP.NET AJAX in AJAX Control Toolkit (v nadaljevanju AJAX) pa je na voljo nabor orodij za izdelavo spletnih aplikacij, ki se po uporabniški prijaznosti že kosajo z namiznimi. Tudi njihovo delovanje je odzivno in hitro. Uporabniku se tako ni več treba prilagajati na drugačnost spletnih aplikacij, saj je uporabniški vmesnik, ki jim je na voljo, skoraj v celoti primerljiv s tistimi, ki so jih vajeni iz namiznih aplikacij.

AJAX pa pomeni tudi nove možnosti, saj so določena opravila in naloge spletnih aplikacij šele zdaj postali smiselni, predvsem z vidika omejenega časa, ki je na voljo za razvoj. Uporabniku lahko tako zdaj že med izpolnjevanjem vnosnih polj ponudimo interaktivno pomoč v obliki različnih oblačkov, ki se pojavijo na njegovo zahtevo ali glede na vpisane vrednosti. Uporabnik lahko tako do novih informacij pride brez motečega ponovnega nalaganja strani.

Če AJAX analiziramo z vidika razvijalca, ugotovimo, da je odprl možnosti za ustvarjalno reševanje problemov, in sicer z gradniki AJAX. Vsi gradniki lahko namreč s strežnikom komunicirajo, ne da bi pri tem zmotili uporabnika med delom. To je vsekakor velika prednost, saj lahko razvijalec uporabnikove akcije sproti obravnava in se na podlagi njih odloča o nadaljnjih akcijah spletne aplikacije. Ena glavnih prednosti pa je precej hitrejša izvedba za uporabnika prijazno delovanje. Prikaz oblačka brez ponovnega nalaganja strani je pred pojavom tehnologije AJAX od razvijalca zahteval pisanje zapletenih JavaScript funkcij, kar je v marsikaterem primeru časovno zelo potratno. Enako funkcionalnost lahko zdaj razvijalec, z uporabo tehnologije AJAX, izvede v manj kot petih minutah.

V diplomskem delu so predstavljene izjemne prednosti uporabe tehnologije AJAX, in to tako z vidika končnega rezultata kot tudi z vidika produktivnosti. V prihodnosti je tako mogoče pričakovati razširitev uporabe tehnologije AJAX na vedno več spletnih aplikacij. Razvoj v klasičnem načinu ASP.NET bo postopoma v celoti nadomeščen z načinom AJAX, saj bo postal neučinkovit.

Že čez nekaj let tako lahko pričakujemo, do bo AJAX že postal standard v razvoju spletnih aplikacij in pojavila se bo potreba po novih tehnologijah. Fuzija namiznih in spletnih aplikacij se bo še stopnjevala in kmalu lahko pričakujemo, da uporabnik obeh aplikacij sploh ne bo znal več razlikovati. Potreba po namiznih aplikacijah bo pojenjala, saj jih bodo

postopoma nadomeščale spletne aplikacije. Kmalu se lahko zgodi, da bo dostop do okolja Windows mogoč kar preko spleta.

LITERATURA IN VIRI

[1.] *Ajax Shopping Cart with Gaia [Gaiaware]*. Najdeno 21. maja 2009 na spletnem naslovu http://samples.gaiaware.net/Aspects/AspectDraggable/ShoppingCart/

[2.] Arlekar, S. (2006, junij). *Ajax User experience, Benefits of Ajax, Ajax Experience.* Najdeno 14. aprila 2009 na spletnem naslovu http://www.roseindia.net/ajax/ajax-user-interface.shtml

[3.] Bos, B. (2009, 02. april). *CSS Template Layout Module.* Najdeno 4. julija 2009 na spletnem naslovu http://www.w3.org/TR/2009/WD-css3-layout-20090402/

[4.] Bosworth, A. (2008, 24. julij). *Ajax Mistakes*. Najdeno 10. septembra 2008 na spletnem naslovu http://alexbosworth.backpackit.com/pub/67688

[5.] *Calendar Demonstration [Microsoft Corporation]*. Najdeno 18. aprila 2009 na spletnem naslovu http://www.asp.net/AJAX/AjaxControlToolkit/Samples/Calendar/Calendar.aspx

[6.] *CollapsiblePanel Demonstration [Microsoft Corporation]*. Najdeno 18. aprila 2009 na spletnem naslovu http://www.asp.net/AJAX/AjaxControlToolkit/Samples/CollapsiblePanel/CollapsiblePanel.aspx

[7.] *ConfirmButton Demonstration [Microsoft Corporation]*. Najdeno 18. aprila 2009 na spletnem naslovu http://www.asp.net/AJAX/AjaxControlToolkit/Samples/ConfirmButton/ConfirmButton.aspx

[8.] Crane, D., Pascarello E., & Darren J. (2006). *AJAX In Action*. Greenwich: Manning Publications Co.

[9.] *Default FreeTextBox Configuration [FreeTextBox]*. Najdeno 21. maja 2009 na spletnem naslovu http://freetextbox.com/demos/

[10.] *Domain Name Search [Hartshorne Software Inc.]*. Najdeno 20. maja 2009 na spletnem naslovu http://instantdomainsearch.com/

[11.] Freitag,P. (2007, 13. december). *AJAX Tutorial with Prototype*. Najdeno 21. maja 2009 na spletnem naslovu http://www.petefreitag.com/item/515.cfm

[12.] Garrett, J. J. (2006, 18. februar). *A new approach to web applications*. Najdeno 9. junija 2008 na spletnem naslovu http://www.adaptivepath.com/ideas/essays/archives/000385.php

[13.] *Gmail [Google]*. Najdeno 6. junija 2009 na spletnem naslovu https://mail.google.com

[14.] *Google maps [Google]*. Najdeno 6. junija 2009 na spletnem naslovu http://maps.google.com/?ie=UTF8&ll=45.658688,15.205078&spn=0.155493,0.407524&z=12

[15.] Heilmann, C. (2006). *Beginning JavaScript with DOM Scripting and Ajax*. New York: Apress.

[16.] Housley, S. (2007). *Comparing Desktop Applications With Web Applications*. Najdeno 22. maja 2009 na spletnem naslovu http://www.rlrouse.com/desktop-web-applications.html

[17.] *How It Works [Netflix]*. Najdeno 6. junija 2009 na spletnem naslovu http://www.netflix.com/HowItWorks

[18.] Kunder,M. (2009). *The size of the World Wide Web*. Najdeno 2. junija 2009 na spletnem naslovu http://www.worldwidewebsize.com/

[19.] *ListSearchExtender Demonstration [Microsoft Corporation]*. Najdeno 18. aprila 2009 na spletnem naslovu http://www.asp.net/AJAX/AjaxControlToolkit/Samples/ListSearch/ListSearch.aspx

[20.] McClure, W., Cate, S., Glavich P., & Shoemaker C. (2006). *Beginning Ajax with ASP.NET*. Indianapolis: Wiley Publishing, Inc.

[21.] Microsoft Corporation (2007, 03. maj). *AJAX Control Toolkit.* Najdeno 15. junija 2008 na spletnem naslovu http://ajaxcontroltoolkit.codeplex.com/

[22.] *Netflix Top 100 [Netflix]*. Najdeno 6. junija 2009 na spletnem naslovu http://www.netflix.com/Top

[23.] *Nice T-Shirts For You [Panic Goods]*. Najdeno 20. maja 2009 na spletnem naslovu http://www.panic.com/goods/

[24.] Pahor, D. (2002). *Leksikon racunalništva in informatike*. Ljubljana: Založba Pasadena, d. o. o.

[25.] Parish, J. *AjaxTrans.* Najdeno 21. maja 2009 na spletnem naslovu http://www.ajaxtrans.com/

[26.] *PopupControl Demonstration [Microsoft Corporation]*. Najdeno 18. aprila 2009 na spletnem naslovu http://www.asp.net/AJAX/AjaxControlToolkit/Samples/PopupControl/PopupControl.aspx

[27.] *Protopage.* Najdeno 20. maja 2009 na spletnem naslovu http://www.protopage.com/

[28.] Quin, L. (2009, 24. februar). *The Extensible Stylesheet Language Family (XSL).* Najdeno 12. aprila 2009 na spletnem naslovu http://www.w3.org/Style/XSL/

[29.] Robie, J. (2004, 07. april). *What is the Document Object Model?*. Najdeno 19. junija 2008 na spletnem naslovu http://www.w3.org/TR/DOM-Level-3-Core/introduction.html

[30.] Tometa Software, (2006, 13. junij). *Ajax - What, When, and Why of Ajax*. Najdeno 18. februarja 2009 na spletnem naslovu http://www.tometasoftware.com/ajax_whitepaper.asp

[31.] w3schools, (2009, junij). *Browser Statistics*. Najdeno 11. julija 2009 na spletnem naslovu http://www.w3schools.com/browsers/browsers_stats.asp

[32.] WebHosting.Info (2009). *Total Domains Trends Worldwide*. Najdeno 2. junija 2009 na spletnem naslovu http://www.webhosting.info/domains/global_stats/total_domains/

[33.] *We'll be playing and having fun [Flickr]*. Najdeno 20. maja 2009 na spletnem naslovu http://www.flickr.com/photos/ebbnflow/2568448256/

[34.] White, A. (2008, 06. oktober). *Measuring the Benefits of Ajax*. Najdeno 18. februarja 2009 na spletnem naslovu http://www.developer.com/xml/article.php/3554271

[35.] Zeiss, D. (2006, 09. september). *ASP.NET AJAX framework comparison*. Najdeno 15. junija 2008 na spletnem naslovu http://www.daniel-zeiss.de/AJAXComparison/Results.htm

PRILOGE

PRILOGA 1: SLOVAR KRATIC 1
PRILOGA 2: GRADNIKI PAKETA AJAX CONTROL TOOLKIT 2
PRILOGA 3: PRIMERJAVA RAZVIJANJA S STANDARDNIMI METODAMI IN Z METODAMI AJAX 6
PRILOGA 4: DRUGE SLIKE 12

PRILOGA 1: Slovar kratic

Kratica	*Pomen kratice*	*Razlaga*
AJAX	Asynchronous JavaScript and XML	skupek medsebojno povezanih tehnologij
ASP.NET	Active Server Pages for .NET	ogrodje za razvijanje dinamičnih spletnih strani, ki ga podpira Microsoft
PHP	Personal Home Page	splošno uporaben skriptni programski jezik, namenjen izdelavi dinamičnih spletnih strani
CSS	Cascading Style Sheets	predloge, ki določajo videz spletnih strani
DOM	Document Object Model	jezikovno nevtralen vmesnik, ki aplikacijam in skriptom omogoča dinamičen dostop do strukturiranih objektov XML
HTML	HyperText Markup Language	označevalni jezik za izdelavo spletnih strani
XHTML	HyperText Markup Language	označevalni jezik, podoben HTML-ju, vendar usklajen s sintakso XML
XML	Extensible Markup Language	preprost računalniški jezik, ki omogoča format za opisovanje strukturiranih podatkov
xPath	XML Path Language	jezik za dostop do elementov dokumenta XML
XSL	Extensible Stylesheet Language Family	skupek priporočil za transformacijo in prikaz dokumentov XML
XSL-FO	XSL Formatting Objects	slovar XML za specifikacijo semantike dokumenta
XSLT	XSL Transformations	jezik za pretvorbo in urejanje dokumentov XML

PRILOGA 2: Gradniki paketa AJAX Control Toolkit

Accordion
Gradnik omogoča prikazovanje gradnikov Panel v slogu harmonike. V vsakem trenutku je tako vidna vsebina samo enega gradnika, pri preostalih pa je vidna samo glava.

AlwaysVisibleControl
Gradnik omogoča lepljenje poljubnega gradnika na določeno mesto na strani, tako da je videti, kot bi lebdela. Njen položaj se ne spreminja niti ob uporabi drsnika.

Animation
Omogoča uporabo različnih prednastavljenih animacij pri določenem gradniku.

AutoComplete
AutoComplete je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik TextBox. Ob pisanju v tak TextBox se prikaže nabor besed, ki se začnejo na vpisane znake.

Calendar
Calendar je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik TextBox. Po kliku takega gradnika TextBox se prikaže koledar, ki po kliku datuma TextBox zapolni z izbranim datumom.

CascadingDropDown
CascadinDropDown je AJAX razširitev, ki jo lahko povežemo na kateri koli padajoči meni. Ob izbiri elementa iz takega padajočega menija se glede na izbiro zapolnijo preostali padajoči meniji.

CollapsiblePanel
CollapsiblePanel je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik Panel. Pri takem gradniku Panel je vidna samo glava, vsebina pa se prikaže, ko ga kliknemo.

ConfirmButton
ConfirmButton je AJAX razširitev, ki jo lahko povežemo na kateri koli gumb. Po kliku gumba se pojavi sporočilo, ki od uporabnika zahteva potrdilo izvedene akcije. Če uporabnik akcije ne potrdi, se sporočilo skrije, klik gumba pa se prezre.

DragPanel
DragPanel je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik Panel. Tak gradnik lahko nato na strani poljubno premikamo.

DropDown

DropDown je AJAX razširitev, ki jo lahko povežemo s skoraj vsakim gradnikom ASP.NET. Po kliku takega gradnika se prikaže padajoči meni, ki je lahko Panel ali kateri drugi gradnik.

DropShadow

DropShadow je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik Panel. Takemu gradniku Panel se prikažejo sence in se po želji zaoblijo robovi.

DynamicPopulate

DynamicPopulate je AJAX razširitev, ki nadomesti vsebino določenega gradnika z rezultatom iz spletne storitve (angl. *web service*) oz. metode strani (angl. *page method*). Rezultat je vedno v obliki besedila in se vstavi v želeni gradnik.

FilteredTextBox

FilteredTextBox je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik TextBox. V tak gradnik TextBox ne bomo mogli vpisovati znakov, ki niso dovoljeni, na primer znaka '<'.

HoverMenu

HoverMenu je AJAX razširitev, ki jo lahko povežemo na vsak gradnik ASP.NET. Ob lebdenju z miško nad takim gradnikom se prikaže poljuben meni.

ListSearch

ListSearch je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik ListBox ali DropDownList in omogoča iskanje po seznamu z vpisovanjem niza črk.

MaskedEdit

MaskEdit je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik TextBox in omejuje vrsto besedila, ki ga lahko vpišemo.

ModalPopup

ModalPopup je AJAX razširitev, ki jo lahko povežemo na kateri koli gumb in omogoča prikazovanje vsebin na modalni način (dokler se ta vsebina ne skrije, uporabnik ne more uporabljati drugih funkcionalnosti na strani).

MutuallyExclusiveCheckBox

ModalPopup je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik CheckBox in omogoča izbiro samo ene vrednosti v posamezni skupini.

NoBot

NoBot je AJAX razširitev, ki na podlagi različnih kontrolnih funkcij preprečuje vnašanje neželene vsebine (angl. *spam*).

NumericUpDown
NumericUpDown je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik TextBox in omogoča večanje ter manjšanje vpisanega števila s klikanjem gumbov.

PagingBulletedList
PagingBulletedList je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik BulletedList in omogoča sortiranje vsebine po straneh.

PasswordStrength
PasswordStrength je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik TextBox in omogoča preverjanje kompleksnosti vnesenega gesla.

PopupControl
PopupControl je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik ASP.NET in omogoča prikaz dodatnih gradnikov, ko ga kliknemo.

Rating
Rating je gradnik v načinu AJAX, ki omogoča izbiranje števila zvezdic, ki predstavljajo oceno določene zadeve.

ReorderList
ReorderList je gradnik v načinu AJAX, ki omogoča spreminjanje vrstnega reda prikazanega seznama. Vrstni red seznama uporabnik spreminja s preprostim načinom povleci in spusti (angl. *drag–drop*).

ResizableControl
ResizableControl je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik in omogoča spreminjanje njegovih dimenzij.

RoundedCorners
RoundedCorners je AJAX razširitev, ki jo lahko povežemo na kateri koli element in omogoča prikazovanje zaobljenih robov izbranega elementa.

Slider
Slider je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik TextBox in ga tako spremenimo v drsnik. Drsnik lahko uporabimo na primer za nadzor glasnosti.

SlideShow
SlideShow je AJAX razširitev, ki jo lahko povežemo na gradnik Image. Omogoča samodejno ali ročno prikazovanje slik v obliki diapozitivov, drugega za drugim.

Tabs
Tabs je kontrola v načinu AJAX, ki omogoča uporabo jezičkov tudi v spletnih aplikacijah.

TextBoxWatermark

TextBoxWatermark je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik TextBox. Ko je tak gradnik TextBox prazen, se v njem prikaže sporočilo z napotki za uporabnika. Ko ga kliknemo, sporočilo izgine in uporabnik lahko vpisuje želeno besedilo.

ToggleButton

ToggleButton je AJAX razširitev, ki jo lahko povežemo na kateri koli gradnik CheckBox in omogoča prikazovanje sličic glede na stanje gradnika.

UpdatePanelAnimation

UpdatePanelAnimation je AJAX razširitev, ki omogoča prikazovanje različnih animacij, medtem ko se UpdatePanel osvežuje oziroma ko se je že osvežil.

ValidatorCallout

ValidatorCallout je AJAX razširitev, ki jo lahko povežemo na katero koli validacijo ASP.NET. Ob nepravilnem vnosu omogoča prikazovanje oblačka s poljubno vsebino.

PRILOGA 3: Primerjava razvijanja s standardnimi metodami in z metodami AJAX

Standardni način (samo z uporabo strežniške kode)

Slika 1: Izsek iz dokumenta HTML

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Standardni način</title>
    <link href="1_normal.css" rel="Stylesheet" media="screen" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:LinkButton ID="lnkbtnPopup" runat="server" Text="Prikaži opozorilo"
                onclick="lnkbtnPopup_Click">
            </asp:LinkButton>
            <asp:Button ID="btnTest" runat="server" Text="Test" OnClientClick="alert('Test')" />
        </div>

        <div id="divOpozorilo" runat="server" Visible="false">
            <asp:Panel ID="pnlSivina" runat="server" CssClass="sivina"></asp:Panel>
            <asp:Panel ID="pnlOpozorilo" runat="server" CssClass="opozorilo" >
                opozorilo...
                <br /><br />
                <asp:Button ID="btnAkcija1" runat="server" Text="Akcija 1"
                    onclick="btnAkcija1_Click" />
                <asp:Button ID="btnAkcija2" runat="server" Text="Akcija 2"
                    onclick="btnAkcija2_Click" />
                <asp:Button ID="btnPreklici" runat="server" Text="Prekliči"
                    onclick="btnPreklici_Click" />
            </asp:Panel>
        </div>
    </form>
</body>
</html>
```

Slika 2: Izsek iz strežniške kode

```
protected void Page_Load(object sender, EventArgs e)
{

}

protected void lnkbtnPopup_Click(object sender, EventArgs e)
{
    //prikažemo opozorilo
    divOpozorilo.Visible = true;
}

protected void btnAkcija1_Click(object sender, EventArgs e)
{
    //izvedemo željeno akcijo
    //...

    //skrijemo opozorilo
    divOpozorilo.Visible = false;
}

protected void btnAkcija2_Click(object sender, EventArgs e)
{
    //izvedemo željeno akcijo
    //...

    //skrijemo opozorilo
    divOpozorilo.Visible = false;
}

protected void btnPreklici_Click(object sender, EventArgs e)
{
    //skrijemo opozorilo
    divOpozorilo.Visible = false;
}
```

Slika 3: Dokument CSS

```
1
2   .sivina
3   {
4       background-color: Gray;
5       position: fixed;
6       width: 100%;
7       height: 100%;
8       filter: alpha(opacity=40);
9       -moz-opacity: .40;
10      top: 0px;
11      left: 0px;
12  }
13
14  .opozorilo
15  {
16      border:solid 2px black;
17      background-color:White;
18      width: 300px;
19      height:150px;
20      text-align:center;
21
22      position: fixed;
23      top: 35%;
24      left:35%;
25      z-index: 1000;
26  }
```

Standardni način (z uporabo strežniške kode in JavaScript skripte)

Slika 4: Izsek iz dokumenta HTML

```
<head runat="server">
    <title>Untitled Page</title>
    <link href="1_javascript.css" rel="Stylesheet" type="text/css" />
</head>
<body>
<script type="text/javascript">

    function prikaziOpozorilo(targetID)
    {
        opozorilo = document.getElementById(targetID);
        opozorilo.style.display = 'inline';
    }

    function skrijOpozorilo(targetID)
    {
        opozorilo = document.getElementById(targetID);
        opozorilo.style.display = 'none';
        return false;
    }

</script>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="lblPopup" runat="server" Text="Prikaži oblaček" ></asp:Label>
            <asp:Button ID="btnTest" runat="server" Text="Test" OnClientClick="alert('Test')" />
        </div>

        <div id="divOpozorilo" runat="server" style="display:none">
            <asp:Panel ID="pnlSivina" runat="server" CssClass="sivina"></asp:Panel>
            <asp:Panel ID="pnlOpozorilo" runat="server" CssClass="opozorilo" >
                opozorilo...
                <br /><br />
                <asp:Button ID="btnAkcija1" runat="server" Text="Akcija 1"
                    onclick="btnAkcija1_Click" />
                <asp:Button ID="btnAkcija2" runat="server" Text="Akcija 2"
                    onclick="btnAkcija2_Click" />
                <asp:Button ID="btnPreklici" runat="server" Text="Prekliči" />
            </asp:Panel>
        </div>
    </form>
</body>
</html>
```

Slika 5: Izsek iz strežniške kode

```
protected void Page_Load(object sender, EventArgs e)
{
    //labeli dodamo JavaScript funkcijo za prikaz opozorila
    lblPopup.Attributes.Add("onclick", "prikaziOpozorilo('" + divOpozorilo.ClientID + "')");
    btnPreklici.Attributes.Add("onclick", "return skrijOpozorilo('" + divOpozorilo.ClientID + "')");
}

protected void btnAkcija1_Click(object sender, EventArgs e)
{
    //izvedemo željeno akcijo
    //...

    //skrijemo opozorilo
    divOpozorilo.Style.Add("display", "none");
}

protected void btnAkcija2_Click(object sender, EventArgs e)
{
    //izvedemo željeno akcijo
    //...

    //skrijemo opozorilo
    divOpozorilo.Style.Add("display", "none");
}
```

Slika 6: Dokument CSS

```
1
2   .sivina
3   {
4       background-color: Gray;
5       position: fixed;
6       width: 100%;
7       height: 100%;
8       filter: alpha(opacity=40);
9       -moz-opacity: .40;
10      top: 0px;
11      left: 0px;
12  }
13
14  .opozorilo
15  {
16      border:solid 2px black;
17      background-color:White;
18      width: 300px;
19      height:150px;
20      text-align:center;
21
22      position: fixed;
23      top: 35%;
24      left:35%;
25      z-index: 1000;
26  }
```

Način AJAX

Slika 7: Izsek iz dokumenta HTML

```
<head runat="server">
    <title>AJAX način</title>
    <link href="1_ajax.css" rel="Stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <asp:ScriptManager ID="ScriptManager1" runat="server" />

        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <div>
                    <asp:Label ID="lblPopup" runat="server" Text="Prikaži oblaček"></asp:Label>
                    <asp:Button ID="btnTest" runat="server" Text="Test" OnClientClick="alert('Test')" />
                </div>

                <asp:Panel ID="pnlOpozorilo" runat="server" CssClass="opozorilo" style="display:none;">
                    opozorilo...
                    <br /><br />
                    <asp:Button ID="btnAkcija1" runat="server" Text="Akcija 1"
                        onclick="btnAkcija1_Click" />
                    <asp:Button ID="btnAkcija2" runat="server" Text="Akcija 2"
                        onclick="btnAkcija2_Click" />
                    <asp:Button ID="btnPrekliciAjax" runat="server" Text="Prekliči" />
                </asp:Panel>

                <cc1:ModalPopupExtender ID="ModalPopupExtender" runat="server"
                    TargetControlID="lblPopup" PopupControlID="pnlOpozorilo"
                    CancelControlID="btnPrekliciAjax"
                    BackgroundCssClass="sivina" >
                </cc1:ModalPopupExtender>
            </ContentTemplate>
        </asp:UpdatePanel>
    </form>
</body>
</html>

```

Slika 8: Izsek iz strežniške kode

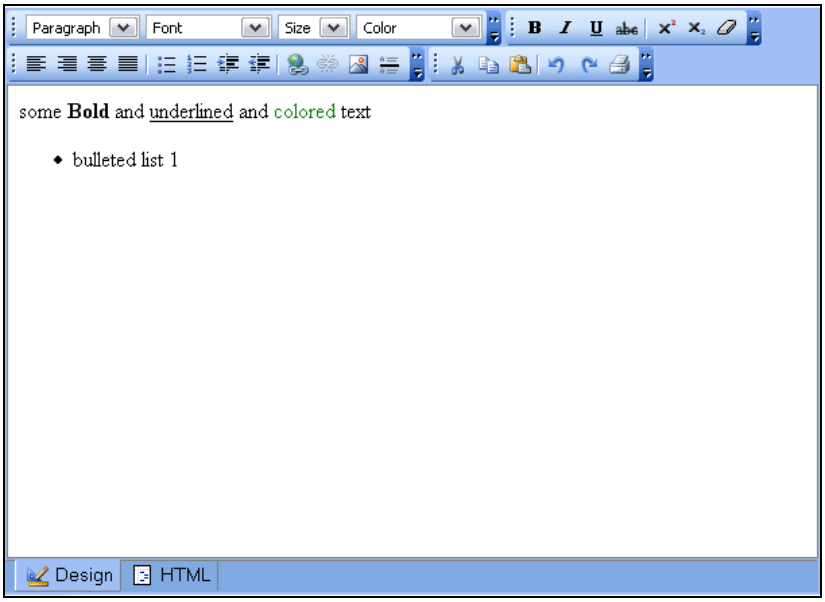
```
protected void Page_Load(object sender, EventArgs e)
{

}
protected void btnAkcija1_Click(object sender, EventArgs e)
{
    //izvedemo željeno akcijo
    //...

    //skrijemo opozorilo
    ModalPopupExtender.Hide();
}

protected void btnAkcija2_Click(object sender, EventArgs e)
{
    //izvedemo željeno akcijo
    //...

    //skrijemo opozorilo
    ModalPopupExtender.Hide();
}
```

Slika 9: Dokument CSS

```
1
2   .sivina
3   {
4       background-color:Gray;
5       filter: alpha(opacity=40);
6       -moz-opacity: .40;
7   }
8
9   .opozorilo
10  {
11      border:solid 2px black;
12      background-color:White;
13      width: 300px;
14      height:150px;
15      text-align:center;
16  }
17
18
19
```

PRILOGA 4: Druge slike

Slika 10: Primer gradnika FreeTextBox



Vir: Default FreeTextBox Configuration, b.l.

Slika 11: Vmesnik AJAXTrans

Vir: AjaxTrans, b.l.