UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Boris Špoljar

Analiza spletnih tehnologij

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: izr. prof. dr. Marko Bajec

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01704/2010
Datum: 01.09.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **BORIS ŠPOLJAR**

Naslov: **ANALIZA SPLETNIH TEHNOLOGIJ**
SURVEY OF WEB TECHNOLOGIES

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Danes so na voljo številne tehnologije, platforme in ogrodja, ki jih lahko uporabimo za razvoj spletnih aplikacij. Izbira tehnologije, ki je v danem primeru najbolj primerna, je kompleksna, saj ni definiranih kriterijev, na osnovi katerih bi izbiro lahko izpeljali. V diplomskem delu analizirajte področje spletnih tehnologij. Izdelajte klasifikacijo spletnih tehnologij ter jih med seboj primerjajte. Rezultat naloge naj bo model za odločanje o najprimernejši tehnologiji, upoštevajoč izbrane aplikacijske zahteve.

Mentor:

prof. dr. Marko Bajec

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01704/2010
Datum: 01.09.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **BORIS ŠPOLJAR**

Naslov: **ANALIZA SPLETNIH TEHNOLOGIJ**
SURVEY OF WEB TECHNOLOGIES

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Danes so na voljo številne tehnologije, platforme in ogrodja, ki jih lahko uporabimo za razvoj spletnih aplikacij. Izbira tehnologije, ki je v danem primeru najbolj primerna, je kompleksna, saj ni definiranih kriterijev, na osnovi katerih bi izbiro lahko izpeljali. V diplomskem delu analizirajte področje spletnih tehnologij. Izdelajte klasifikacijo spletnih tehnologij ter jih med seboj primerjajte. Rezultat naloge naj bo model za odločanje o najprimernejši tehnologiji, upoštevajoč izbrane aplikacijske zahteve.

Mentor:

prof. dr. Marko Bajec

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Boris Špoljar,

z vpisno številko 63050113,

sem avtor diplomskega dela z naslovom:

Analiza spletnih tehnologij

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Marko Bajec;
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela;
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, 28. 02. 2011 Podpis avtorja:

Zahvala

Zahvaljujem se mentorju izr. prof. dr. Marku Bajcu za vodenje in strokovno pomoč pri izdelavi diplomske naloge. Prav tako se zahvaljujem Edvinu Sovincu za pomoč in nasvete pri pisanju diplomske naloge.

Posebna zahvala gre staršem, ki so me med študijem brezpogojno podpirali.

Kazalo

Seznam uporabljenih kratic in simbolov

AJAX asinhroni JavaScript in XML (ang. Asynchronous JavaScript And XML)
API aplikacijski programski vmesnik (ang. Application Programming Interface)
ASP aktivne strežniške strani (ang. Active Server Pages)
CGI skupni prehodni vmesni (ang. Common Gateway Interface)
CSS predloge za oblikovanje spletnih strani (ang. Cascading Style Sheets)
DOM dokumentni objektni model (ang. Document Object Model)
EJB javanska strežniška zrna (ang. Enterprise JavaBeans)
GWT Googlov nabor orodij za razvoj obogatenih spletnih strani (ang. Google Web Toolkit)
HTML jezik za označevanje hiperteksta (ang. Hypertext Markup Language)
HTTP protokol za prenos hiperteksta (ang. Hypertekst Transfer Protokol)
IDE razvojno okolje (ang. Integradet Development Enviroment)
Java EE poslovna različica javanske podlage (ang. Java Platform, Enterprise Edition)
JDBC javanska povezljivost s podatkovnimi bazami (ang. Java Database Connectivity)
JSP javanske strežniške strani (ang. JavaServer Pages)
JVM javanski navidezni stroj (ang. Java virtual machine)
MVC model-pregled-krmilnik (ang. Model-view-controller)
MVP model-pregled-predstavitelj (ang. Model-view-presenter)
ORM objektno-relacijsko preslikovanje (ang. Object-relational mapping)
REST prenos reprezentativnega stanja (ang. Representational State Transfer)
RIA obogatena spletna aplikacija (ang. Rich Internet Appliaction)

SOA storitveno usmerjena arhitektura (ang. Service Oriented Architecture)

SOAP stadnard za spletne storitve, ki temelji na XML (ang. Simple Object Access Protocol)

SSI „vsebovana“ funkcija strežnika (ang. Server Side Includes)

TCP protokol za nadzor transporta (ang. Transmission Control Protocol)

URL naslov vira v enotni obliki (ang. Uniform Resourve Locator)

XHTML razširljiv jezik za označevanje hiperteksta (ang. Extensible Hypertext Markup Language)

XML razširljivi označevalni jezik (ang. Extended Markup Language)

Povzetek

Svetovni splet je postal zelo pomembna podlaga za razvoj in izvajanje aplikacij. Pomemben postopek pri razvoju spletnih aplikacij je izbira tehnologij, na katerih bo aplikacija zgrajena. Razvijalci se soočajo z množico platform, jezikov, ogrodij in tehničnih artefaktov, med katerimi lahko izbirajo. Izbira pa nato nosi posledice na večino ostalih odločitev v procesu razvoja.

Diplomska naloga vsebuje analizo, klasifikacijo in primerjavo spletnih tehnologij, ki podpirajo razvoj spletnih aplikacij. Na začetku je predstavljena kratka zgodovina razvoja svetovnega spleta in vseh pripadajočih tehnologij. Sledi analiza razvojnih platform .NET, Java in LAMP. Nato pa so predstavljena najpopularnejša ogrodja za izvajanje na strežniku in odjemalcu. Vsebujoče primerjave lahko služijo kot kažipot pri izbiri ustrezne tehnologije specifičnim aplikacijskim zahtevam.

Ključne besede:

spletne tehnologije, spletne aplikacije, spletno programiranje, programska ogrodja, ponovna uporaba

Abstract

The World Wide Web bas become an important platform for developing and running applications. A vital process while developing web applications is the choice of web technologies, on which the application will be build. The developers face a dizzying array of platforms, languages, frameworks and technical artifacts to choose from. The decison carries consequences on most other decisions in the development process.

Thesis contains analisis, classifications and comparison of web technologies supporting Web application development. The begining contains the World Wide Web history and affiliated technologies. Followed by analisis of web development platforms .NET, Java and LAMP. Then we present the most popular server-side and client-side frameworks. The included comparisons can serve as a roadmap for choosing technical artefacts according to specific application needs.

Key words:

web technologies, web applications, web programming, software frameworks, software reuse

Poglavje 1

Uvod

Industrija spletnega razvoja je od sredine 90. let prejšnjega stoletja ena izmed najhitreje rastočih industrij na svetu [2]. V tem času so se stroški razvoja ter gostovanja drastično zmanjšali. Razvijalci lahko izbirajo med množico platform, programerskih jezikov, pristopov, ogrodij in ostalih tehnoloških rešitev. S selitvijo aplikacijske logike na odjemalce in računalništvom v oblaku pa industrija dobiva nov zalet.

1.1 Področje raziskave

Zaradi velike množice razvijalskih platform in ogrodij, ki so na voljo razvijalcem spletnih aplikacij, je izbira zelo težka. Različna ogrodja rešujejo razne razvojne probleme z uporabo različnih programerskih metod. Izbira tehnologije zahteva veliko časa ter predstavlja velik strošek. Ti stroški skokovito narastejo, če mora razvojna skupina zamenjati tehnologijo zaradi neustrezne izbire. Podjetje, ki učinkovito uporablja tehnologijo, ki ustreza zahtevam podjetja in trga, ima veliko strateško prednost pred konkurenco.

1.2 Namen

Namen raziskave je analizirati tehnologije, ki so na voljo za razvoj spletnih aplikacij, ter opredeliti njihove prednosti in slabosti. Da bi lahko odgovorili na vprašanje „Katero izmed njih je najboljše?“, jih bomo opredelili in navedli, katere tehnologije bolje ustrezajo določenim kriterijem zahtev.

1.3 Struktura dela

V drugem poglavju je navedena kratka zgodovina razvoja svetovnega spleta in tehnologij, ki so to omogočale.

V tretjem poglavju so analizirane platforme, ki jih razvijalci uporabljajo za razvoj na spletu.

V četrtem poglavju so predstavljena najbolj razširjena ogrodja za izvajanje na strežniku in odjemalcu.

V petem poglavju je narejen kratek povzetek in opredeljene so tehnologije, ki ustrezajo specifičnim zahtevam.

Poglavje 2

Razvoj spletnih tehnologij

Desetletje od začetka delovanja svetovnega spleta se je omenjeni medij razvil iz medija za predstavitev statičnih spletnih strani v standardno tehnologijo za naraščujoče število interaktivnih informacijskih sistemov. Prvotna implementacija s svojo statično naravo je bila dovolj za izmenjavo dokumentov, a neustrezna za napredne aplikacije. Kljub temu se je dovolj jasno prikazal izjemno velik potencial in globalen doseg novega medija ter motiviral nadaljne raziskave, kako na svetovni splet prenesti družabne, izobraževalne in komercialne aplikacije. Osnovne zahteve za spletne aplikacije in storitve, kot so ažurne informacije ter učinkovit uporabniški vmesnik, niso bile možne le s statično vsebino. Za zadovoljitev teh zahtev so bile razvite metode za zagotavljanje dinamične vsebine na zahtevo. Implementacija teh metod je uporabnikom omogočila posredno zahtevanje strežniških virov, programov, ki so ustvarjali dokumente. Kmalu so se zelo razširile spletne komercialne, izobraževalne in komunikacijske storitve, ki so izkoriščale možnost dinamične vsebine. Značaj spleta ni bil več dokumentno usmerjen temveč aplikacijsko. Sočasno se je interaktivnost na strani odjemalca močno razširila preko prvotnih ovir, ki so bile omejene na le enostavne navigacije preko hipertekstovnih povezav.

Tehnologije je najlažje razumeti, če jih obravnavamo v kontekstu, ki je motiviral njihov začetek. Evolucija svetovnega spleta je bila primarno motivirana z iskanjem abstrakcij, ki so izboljšale njegovo uporabnost za končne uporabnike z bogatejšimi uporabniškimi vmesniki ter za razvijalce z učinkovitimi programerskimi metodami. To poglavje opisuje razvoj tehnologij svetovnega spleta.

2.1 Statične strani

Svetovni splet (ang. World Wide Web – www) je bil razvit leta 1990 na evropski organizaciji za nuklearne raziskave (CERN) kot hipertekstovni sistem za izmenjavo dokumentov med različnimi računalniki in aplikacijami preko interneta. Enostavna in odprta oblika je pritegnila pozornost tudi zunaj CERN-a. Leta 1993 je CERN izdal svojo implementacijo v javnost za brezplačno uporabo, kar je izdatno povečalo zanimanje.

Leta 1993 je državni center za superračunalniške aplikacije (NCSA) na univerzi v Illinois izdal prvi spletni brskalnik z grafičnim uporabniškim vmesnikom, imenovan NCSA Mosaic 1.0. Brskalnik je bilo moč uporabljati na X Windows, Microsoft Windows in Macintosh sistemih. Takojšen uspeh brskalnika je še dodatno povečal zanimanje poslovnih in javnih interesov za splet. CERN in NCSA sta postavila temelje, na katerih je zgodovina spletnega razvoja gradila v nenehnem iskanju izboljšav v učinkovitosti, enostavnosti uporabe ter interaktivnosti.

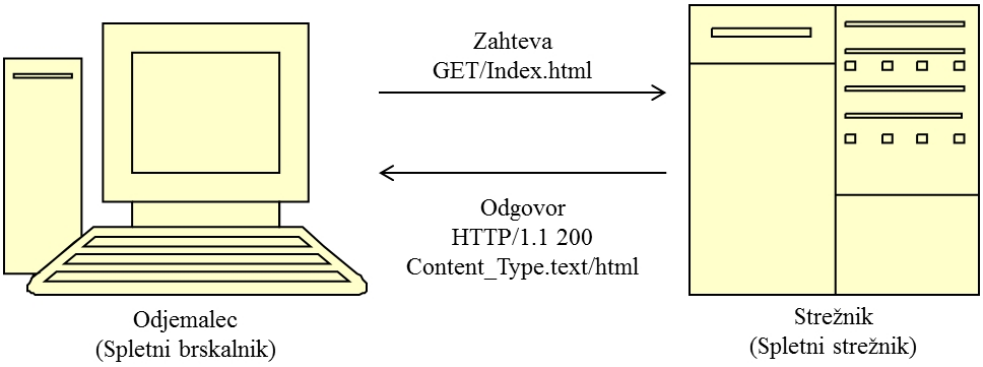
Svetovni splet temelji na HTTP protokolu ter HTML jeziku. Prvi omogoča enostavne implementacije komunikacijskih sistemov, HTML pa enostaven in enoličen mehanizem za sestavo in oblikovanje spletnih strani.

2.1.1 HTTP

Protokol za prenos hiperteksta (ang. Hypertext Transfer Protocol – HTTP) je osnovni protokov svetovnega spleta. Je enoličen, povezovalno usmerjen protokov brez stanja. Usmerjen je povezovalno, ker potrebuje protokol za nadzor transporta (ang. Transmission Control Protocol – TCP) v povezanem stanju. HTTP vzpostavi komunikacijski kanal od začetka do konca (med strežnikom in odjemalcem), po katerem tečejo podatki o spletnih straneh, zahtevkih ipd. Protokol ne vzdržuje stanja. Vsak prenos podatkov je neodvisna povezava glede na prejšnje prenose, prav tako ne obstaja nobena zveza med njimi.

Na sliki 2.1 je prikazan primer komunikacije. Najprej odjemalec vzpostavi TCP povezavo s strežnikom, preko ustreznih HTTP vrat. Nato odjemalec pošlje HTTP zahtevo po določenih virih na strežnik. Strežnik odgovori skozi isto povezavo z zahtevanimi podatki.

HTTP določa tudi šifriranje parametrov med stranmi, tunelske povezave (za sisteme s požarnim zidom), obstoj vmesnih predpomnilnih strežnikov itd.



Slika 2.1: Primer HTTP komunikacije [1].

2.1.2 HTML

Jezik za označevanje hiperteksta (ang. Hypertext Markup Language – HTML) omogoča predstavitev bogate vsebine, reference na ostale vire (slike, video, idt.), povezave na ostale dekumente, prikaz zaslonskih mask itd. Predstavlja osnovo spletnega dokumenta. Z njim lahko ustvarimo strukturo in semantično ureditev dokumenta. Trenutna verzija HTML je 4.01. Obstaja tudi razširljiv jezik za označevanje hiperteksta (ang. Extensible Hypertext Markup Language – XHTML), ki je običajno opredeljen kot veljavna XML verzija HTML-ja. Priskrbi nas z XML shemo, ki jo lahko uporabimo za preverjanje pravilne zasnove dokumenta.

CSS

Z razvojem HTML-ja se je povečala želja razvijalcev po večjih možnostih oblikovanja. W3C[1] je predstavil več različnih stilskih predlog [5], z namenom izboljšave predstavitve spletnih strani. Iz vseh predlogov so izbrali dva, ki sta postavila temelje tehnologiji predlog za oblikovanje spletnih strani (ang. Cascading Style Sheets – CSS), s čimer lahko določamo vizualno podobo spletnih strani ter ločimo predstavitev od vsebine. Prednost tega pristopa je, da lahko predstavitev spremenimo le na enem mestu, spremembe pa se takoj odražajo na vseh straneh. S tem omogoča tudi bolj obvljadljiv razvoj spletnih strani.

[1]World Wide Web Consortium (krajše W3C) je mednarodni konzorcij, ki določa standarde več internetno povezanih jezikov in tehnologij.

2.2 Dinamične vsebine

Uspeh NCSA Mosaic brskalnika in svetovnega spleta je vodil do zahtev po dinamičnih vsebinah, izboljšani interaktivnosti in povezanosti na zunanje sisteme. Doba izključno statičnih strani se je končala, ko je NCSA uvedla skupni prehodni vmesnik (ang. Common Gateway Interface – CGI) leta 1993 kot razširitev NCSA spletnega strežnika. CGI omogoča uporabniku skozi brskalnik poslati zahtevo programu, ki teče na strežniku, v nasprotju z zahtevo po statičnemu dokumentu. Ostale inovativnosti, kot so „vsebovane“ funkcije na strežniku (ang. Server Side Includes – SSI) in izpolnitev zaslonskih mask brskalnika, so bile metode, ki so skušale narediti splet interaktivnejši in bolj dinamičen.

2.2.1 CGI

CGI metoda opredeljuje mehanizem, po katerem je lahko informacija posredovana med HTTP strežnikom in zunanjim programom. Ta mehanizem se še vedno široko uporablja, ker je enostaven in ga podpira večina spletnih strežnikov; poleg tega nam daje popolno svobodo pri izbiri programskega jezika za njegov razvoj.

Operacijska shema CGI-ja je imela šibko točko; vsakič ko smo prejeli zahtevo, je spletni strežnik sprožil proces za izvajanje CGI programa. Ker je večina CGI implementacij napisanih v skriptnih programskih jezikih (Perl, PHP, Python itd.) ali v jezikih, ki potrebujejo izvajalno okolje (VisualBasic, Java itd.), je to predstavljalo težko breme za strežniške računalnike. Če je imel strežnik več CGI dostopov, je to vodilo k resnim performančnim težavam. Za rešitev teh težav sta se uporabljaja predvsem dva pristopa. Prva rešitev je bila razvoj modulov, ki so bili bolje povezani s spletnimi strežniki, druga pa razširitev spletnega strežnika z jezikovnim tolmačem. Kmalu se je število arhitektur in programerskih jezikov, s katerimi je bilo moč programirati spletne rešitve, močno povečalo. Skriptni jeziki kot Perl, PHP, Python, Ruby itd. imajo vsak svojo implementacijo CGI vmesnika. Platformi .NET in Java EE pa sta vmesnik še razširili z Java Servlet in s Handler implementacijami.

2.2.2 SSI

„Vsebovane“ funkcije na strežniku so bile predstavljene leta 1993 pri NCSA kot metoda za izvajanje trivialnih izračunov znotraj HTML strani, ki jih je izvedel strežnik direktno. Pri CGI mehanizmu jih izvaja ločeni proces. SSI predloga

vsebuje HTML besedilo in področje z ukazi, ki se izvedejo preden, se pošlje odgovor uporabniku. Prvotna implementacija je vsebovala include ukaz za vključitev tekstovnih datotek, echo ukaz za izpis spremenljivk in exec ukaz za izvajanje zunanjih programov. Kasneje je bila dodana še podpora za pogojno izvajanje. SSI se še vedno razvija kljub redki uporabi, osnovni nabor ukazov pa podpira večina spletnih strežnikov. Slabosti SSI metode so v skalabilnosti in slabi možnosti ponovne uporabe kode. Slabša je obvladljivost kode, vsak exec ukaz tvori nov proces, kar obremenjuje strežnik, poleg tega je podprto le osnovno procesiranje. Nudi enostavno, poceni alternativo za nekritične spletne aplikacije z nizkim prometom.

2.3 Izvajanje na odjemalcu

Pri statičnih in dinamičnih straneh se vse izvaja na strežnikih. Brskalnik na odjemalcu le prikaže spletno stran ter na akcijo uporabnika pošlje zahtevo na strežnik po določenem viru. Za razbremenitev strežnikov ter povečanje odzivnosti in interaktivnosti v brskalniku so se razvile tehnologije, ki omogočajo izvajanje na odjemalcu.

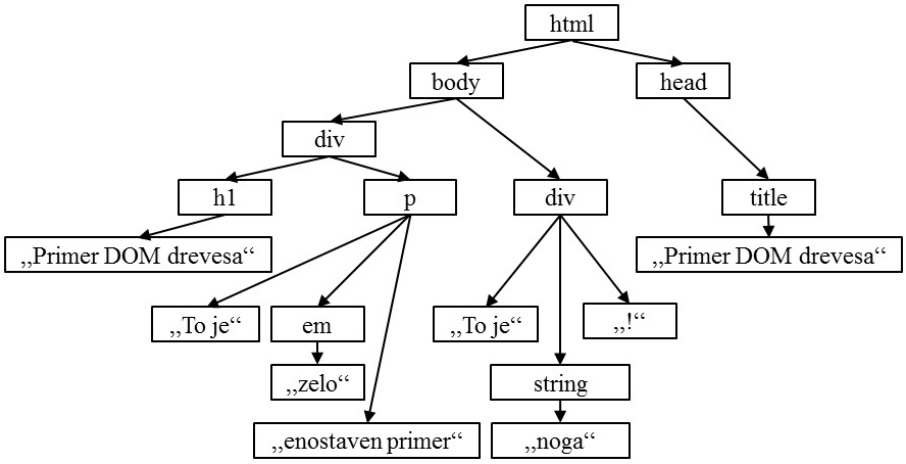
JavaScript

Leta 1995 je Netscape razvil programski jezik JavaScript, ki se je lahko vključil v spletne strani ter procesiral številke in spreminjal vsebino form. Med razvojem so ga najprej poimenovali Mocha, nato pa še LimeWire in LiveScript. Zaradi podobnosti s programerskim jezikom so ga pred izdajo poimenovali JavaScript. Način, kako se je skliceval na HTML elemente spletne strani in vhode kot otroke svojih starševskih zaslonskih mask, je bil kasneje poznan kot dokumentni objektni model (ang. Document Object Model – DOM).

Leta 1997 ga je standardizirala mednarodna neprofitna organizacija ECMA pod imenom ECMAScript. Jezik se široko uporablja na spletu, pogosto pa ga zamenjujemo z JavaScript in JScript. JScript je implementacija ECMAScript standarda, kot ga uporablja Microsoft Internet Explorer. Mozilla Firefox in Google Chrome uporabljata JavaScript implementacijo, brskalnika Opera in Safari pa ECMAScript implementacijo. Zaradi različnih implementacij prihaja do razhajanj in to je tudi največja težava JavaScript jezika. Razvijalci morajo svojo JavaScript kodo prilagajati različnim brskalnikom. Dodatno neusklajenost predstavlja DOM specifikacija, različno obravnavanje primitivnih vrednosti (primer "undefined") ter dostopnost funkcij, dodanih v kasnejši specifikaciji ECMAScript standarda.

DOM

Dokumentni objektni model je standardiziral W3C konzorcij. Predstavlja vmesnik, ki je neodvisen od platforme in jezika ter omogoča programom in skriptam dinamičen dostop in posodabljanje vsebine, strukture ter sloga dokumenta. Vsi gradniki vsebine spletne strani so v modelu DOM predstavljeni v hierarhiji objektov, kot je razvidno na sliki 2.2.



Slika 2.2: Drevesna struktura DOM dokumenta HTML [7].

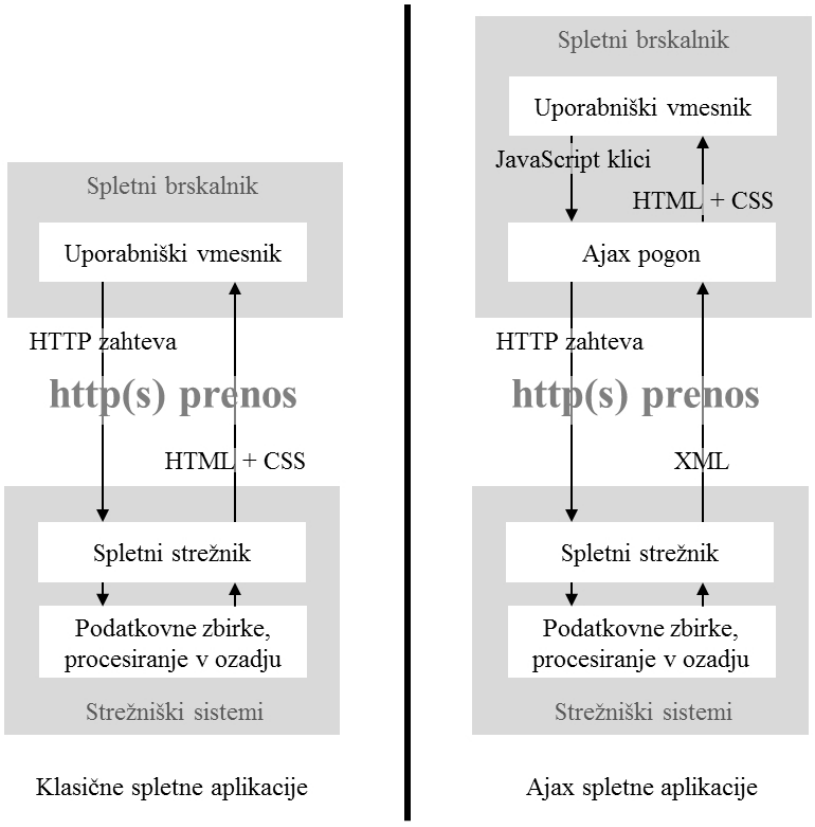
DHTML

Novejši JavaScript prevajalniki so precej zmogljivejši od svojih predhodnikov, prav tako se je razširil tudi sam jezik in DOM model. Vse to je pripomoglo k nastanku novega pojma dinamični HTML (ang. Dynamic HTML – DHTML), ki obsega sodelovanje tehnologij, ki se uporabljajo pri razvoju interaktivnih in animiranih spletnih strani. Vključuje označevalni jezik HTML, skriptni jezik na strani odjemalca (JavaScript), jezik za predstavitev predlog (CSS) ter vmesnik za dostop do HTML elementov (DOM). DHTML omogoča skriptnim jezikom enostavno spreminjanje izgleda spletne strani.

AJAX

Stopnja dinamičnosti spletnih strani se je izdatno povečala z uporabo asinhronega JavaScript-a in XML-a (ang. Asynchronous JavaScript And XML – AJAX). AJAX predstavlja nov pristop razvoja, ki uporablja kombinacijo dobro

sprejetih spletnih tehnologij: standardizirana predstavitev s XHTML in CSS, dinamični prikaz in interakcija z DOM, upravljanje podatkov in izmenjava z XMLHttpRequest ter JavaScript, ki vse skupaj povezuje. Z Ajax-om si lahko spletne aplikacije izmenjujejo podatke s strežnikom asinhrono. Kot prikazuje slika 2.3, je pred tem bilo potrebno za vsako komunikacijo s strežnikom poslati zahtevek, kot odgovor pa smo prejeli novo stran, ki se je prikazala v brskalniku.



Slika 2.3: Primerjava klasičnega modela za spletne aplikacije (levo) z Ajax modelom (desno) [38].

Temelji Ajax tehnologije so sicer nastajali že leta 1995, ko je asinhrono prenašanje podatkov postalo uporabno ob nastanku javanskih apletov. Brskalnik Internet Explorer je leta 1996 vključil IFrame HTML element, ki je omogočal asinhrono nalaganje. Leta 1999 pa je Microsoft vgradil XMLHTTP ActiveX kontrolo v Internet Explorer 5, katerega so kasneje vključili tudi ostali brskalniki kot XMLHttpRequest JavaScript objekt. Kontrola omogoča asinhrono komunikacijo odjemalca s strežnikov z uporabo JavaScript kode. Teh-

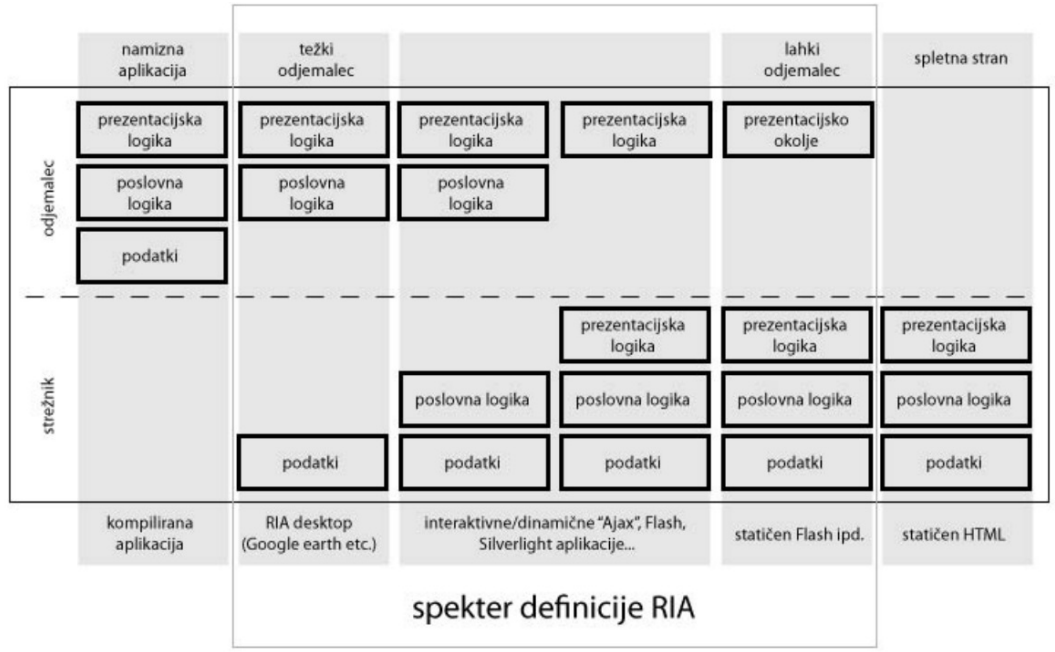
nologija je postala zelo popularna, ko se je pojavila v priljubljenih spletnih aplikacijah, kot so Outlook Web Access (2000), Oddpost (2002), še posebej pa, ko jo je začel uporabljati tudi Google v Gmail (2004) in Google Maps (2005) storitvah. Leta 2006 je W3C izdal prvo specifikacijo XMLHttpRequest objekta kot uraden spletni standard.

2.4 Obogatene spletne aplikacije

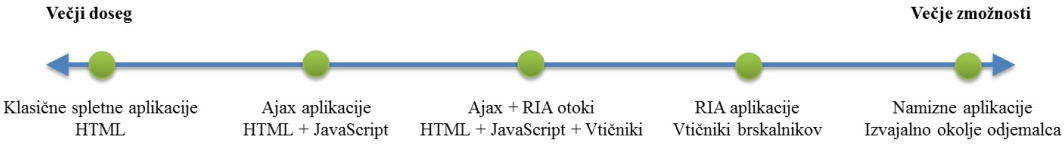
Obogatene spletne aplikacije (ang. Rich Internet Applications – RIA) so vmesni člen med namiznimi in spletnimi aplikacijami [7]. Prva omemba izraza RIA se je pojavila leta 2002 v članku podjetja Macromedia z naslovom „A next-generation rich client“ [8]. Aplikacije RIA skušajo združiti prednosti namiznih in spletnih aplikacij. Imajo večji doseg in manjše stroške vzdrževanja kot namizne aplikacije. Napredne rešitve izdelave uporabniškega vmesnika in interaktivnosti pa uporabniško izkušnjo že precej približajo namiznim aplikacijam. RIA aplikacije postajajo vedno bolj priljubljene in ta trend se bo verjetno nadaljeval tudi v prihodnosti. Razkošnost, odzivnost uporabniškega vmesnika in dostopnost navdušuje tako uporabnike kot razvijalce. Na sliki 2.4 je prikazana razplastitev RIA aplikacij v primerjavi z namiznimi in klasičnimi spletnimi stranmi glede na predstavitveno logiko, poslovno logiko ter podatke.

Tudi slika 2.5 prikazuje razlike med RIA ter namiznimi in spletnimi stranmi z večjim dosegom na eni in večjo zmožnostjo na drugi strani [15]. Aplikacije sledijo različnim scenarijem in se ujemajo z določenimi točkami na diagramu. Na skrajni levi so klasične spletne strani, katerih prioriteta je čim večji doseg. Ne dolgo nazaj so bile vse spletne strani takšne. Določene spletne strani dajejo večji poudarek uporabnosti in interaktivnosti, čeprav lahko to pomeni zmanjšanje dosega. Ob istem času veliko število spletnih strani ne uporablja le JavaScript za izvajanje na odjemalcu. Te se zanašajo na vtičnike, kot so Silverlight, Flash in Google Gears za obogatitev uporabniške izkušnje s funkcionalnostmi, kot so lokalna shramba (ang. local storage), dostop do uporabniških datotek, bogata vektorska in medijska grafika, procesiranje v ozadju in podobno.

Tabela 2.1 povzema primerjavo med namiznimi in običajnimi aplikacijami. Vidijo se slabosti in prednosti obeh ter kako obogatene spletne aplikacije združujejo prednosti obeh. Potrebno je poudariti, da prednosti spletnih aplikacij v primerjavi z namiznimi pridejo do izraza le, če je število uporabnikov dovolj veliko. Če je na primer uporabnikov le pet, je namizna aplikacija verjetno boljša izbira.



Slika 2.4: Razslojenost RIA aplikacij v primerjavi z namiznimi in klasičnimi spletnimi stranmi.



Slika 2.5: Različni pristopi spletnega razvoja [15].

Obogatene spletne aplikacije lahko glede na način razvoja in namestitve razdelimo na:

- **Uporaba vtičnikov brskalnika.** Aplikacije razvijamo na posebni platformi ali v posebnem okolju. Namestimo jih kot vstavljene rešitve ali samostojne aplikacije, ki se zaženejo iz brskalnika. Primeri takšnih tehnologij so Flash, Silverlight, Swing, JavaFX.
- **Izvajanje JavaScript kode.** Aplikacije uporabljajo kombinacijo tehnologije za doseganje svojih rezultatov, tipično vključujejo XHTML/HTML, CSS, DOM in JavaScript. Osnovna ideja je uporaba CSS in HTML za predstavitev uporabniškega vmesnika, JavaScript-a za asinhrono povezavo s strežnikom in DOM skriptiranja za spreminjanje uporabniškega

Lastnost	Namizne aplikacije	RIA aplikacije	Spletne aplikacije
Bogatejša uporabniška izkušnja	+	+	-
Interaktivnost, odzivnost	+	+	-
Manjši obseg vzdrževanaj	-	+	+
Visok doseg	-	+	+

Tabela 2.1: Lastnosti namiznih in spletnih aplikacij.

vmesnika. Ta strategija daje aplikacijam minimalne zahteve.

Trendi napovedujejo, da bo spletna platforma postala še bogatejša v odnosih na izvajalni pogon, možnostih predstavitev in kapacitet okolja. HTML5 prinaša precej izboljšav samega jedra okolja v brskalniku, čemur hitro sledijo tudi vsi brskalniki. Naslednja večja izdaja HTML standarda bo prinesla veliko funkcionalnosti, kot so potegni in spusti, predvajanje video posnetkov, lokalno shranjevanje, risanje po zaslonu itd., kar je bilo do sedaj mogoče le z uporabo vtičnikov. RIA tehnologija je vedno bolj razširjena in je začela predstavljati pomembno ter potrebno izbiro za razvoj naprednih spletnih aplikacije. Prednosti RIA aplikacije so naslednje [7]:

- **naprednejše uporabniške izkušnje;**
  Obogaten uporabniški vmesnik ki vsebuje interaktivne elemente, kot so: kontrola povleci in spusti, drevesni pregled, razvrščanje seznama idt.
- **izboljšana interaktivnost sistema;**
  Zahtevki uporabnika se lahko v celoti obravnavajo na odjemalčevi strani brez strežniške komunikacije. Ostali zahtevki, ki potrebujejo stežniško komunikacijo, se izvršijo asinhrono, tako da se izvajanje aplikacije ne ustavi.
- **zmanjšanje omrežnega prometa.**
  Večji del aplikacije se prenese le ob njeni prvi uporabi. Ustrezno delno ali celotno posodabljanje aplikacije se vrši glede na uporabnikove zahtevke. V primerjavi s klasičnimi spletnimi aplikacijami je zato lahko pri prvotnem nalaganju količina prometa večja, ker se na odjemalčevo stran prenaša tudi aplikacijska logika in kompleksnejši elementi uporabniškega vmesnika.

Imajo pa RIA aplikacije tudi določene pomankljivosti, na katere moramo biti zelo pozorni pri razvoju. Pomankljivosti so [7]:

- **daljši čas nalaganja;**
  V nasprotju z običajnimi spletnimi aplikacijami RIA aplikacije na odjemalčevo stran prenesejo tudi dodatno aplikacijsko logiko. Posledica tega je daljše prvotno nalaganje.
- **posebno okolje za izvajanje;**
  Aplikacija lahko na odjemalcu zahteva dodatno programsko opremo (na primer vtičnik za Flash ali Silverlight, omogočen JavaScript).
- **zahteva več sistemskih virov;**
  Ker se izvajanje izvršuje na odjemalcu, se razbremeni strežnik, a obremeni odjemalec, kar lahko povzroča težave pri napravah z manj procesorske moči.
- **zmanjšana dostopnost;**
  Zaradi varnosti RIA aplikacije navadno nimajo dostopa do sistemskih virov, v kolikor ji uporabnik ne zagotovi posebnih privilegijev.
- **vsebina ni vidna iskalnikom;**
  Običajne strani iskalniki lažje najdejo, saj indeksirajo glavno stran in vse podstrani. Ker so RIA aplikacije predstavljene kot ena spletna stran, se iskalniki pri njih slabše odrežejo.
- **problem z uporabo zgodovine.**
  Uporabniki so pri klasičnih straneh navajeni navigacije z gumbi naprej in nazaj v brskalniku. Takšna uporaba RIA aplikacijam ne ustreza najbolje.

2.5 Računalništvo v oblaku

Koncept oblaka uvaja korenite spremembe v načinu shranjevanja in upravljanja s podatki in aplikacijami, ki niso več nameščeni na uporabnikovi strojni opremi, ampak so na voljo v oblaku, do katerega uporabnik dostopa preko medmrežja. Računalništvo v oblaku lahko pomeni programsko opremo ali strojno opremo. Torej lahko gre za aplikacijo, do katere dostopamo preko medmrežja, ali pa predstavlja strežnike, ki jih uporabljamo, kadar jih potrebujemo. V obeh primerih gre za oblačno storitev, do katere lahko dostopamo iz vsakega računalnika, ki ima dostop do medmrežja, neodvisno od nameščenega operacijskega sistema in brskalnika. Storitve oblačnega računalništva razdelimo na tri segmente: programsko opremo, programsko platformo in računalniško infrastrukturo.

2.5.1 Programska oprema kot storitev

Programska oprema kot storitev (ang. Software as a Service – Saas) je storitev, zasnovana na konceptu posojanja programske opreme od ponudnika namesto odkupa. Aplikacije so nameščene v oblaku, do katerega odjemalci dostopajo preko medmrežja. Za vzdrževanja programske in strojne opreme skrbi ponudnik SaaS. Odjemalec plačuje uporabo aplikacije, zato mu pravimo tudi najemnik (ang. tenant). Aplikacije SaaS so običajno razvite za specifično spletno uporabo (na primer: poštni odjemalec, pisarniška aplikacija) in podpirajo sočasno večkratno uporabo. Glavne lastnosti SaaS so:

- mrežni način upravljanja in dostopanja do programske opreme,
- dostava aplikacij po modelu ena-proti-mnogo (ena instanca, večkratno-najemna arhitektura) in
- centralizirano posodabljanje nadgrajevanja brez posega odjemalca.

2.5.2 Platforma kot storitev

Platforma kot storitev (ang. Platform as a Service – PaaS) nudi razvijalno okolje za razvijalce. Končni uporabnik napiše lastno programsko kodo, ki jo PaaS ponudnik naloži in predstavi na svetovnem spletu. PaaS nudi vse, kar potrebujemo za razvoj aplikacij ali storitev preko medmrežja. Takšen model omogoča hitrejši in cenejši razvoj aplikacij. PaaS ponudniki imajo različne modele razvijalnih okolij, ki lahko omejuje razvijalca glede na uporabo tehnologij ali programskih jezikov. Primerjava PaaS ponudnikov je opravljena v naslednjem poglavju.

2.5.3 Infrastruktura kot storitev

Infrastruktura kot storitev (ang. Infrastructure as a Service – IaaS) je najnižji nivo storitev, ki ga obravnava računalništvo v oblaku. Gre za dostavo računalniške infrastrukture, pri kateri gre običajno za najem virtualnega okolja. IaaS ne ponuja aplikacij ali oblačne platforme, ampak strojno opremo, ki podpira vse koncepte računalništva v oblaku. Uporabnik lahko kupi infrastrukturo glede na zahteve v določenem trenutku, tako uporabnik plačuje le toliko, kolikor potrebuje. Pri klasičnem nakupu računalniške infrastrukture je potrebno veliko premisleka, poleg tega so začetni stroški večji. Pogosto se dogaja, da je strežniška strojna oprema neenakomerno uporabljena, zato izkoristek ni optimalen. Glavne prednosti IaaS so:

- celovita infrastruktura za računalništvo v oblaku,
- dinamično povečanje oz. pomanjšanje infrastrukture v odvisnosti od potrebe po računalniških virih,
- spremenljivi stroški, ki so odvisni od časa uporabe virov in količine uporabljenih virov,
- sočasno večkratno najemanje na isti infrastrukturi.

2.5.4 Ponudniki oblačnih storitev

Trenutno so glavni ponudniki storitev v oblaku Google, EMC, Microsoft, Amazon, Salesforce.com in IBM, ki pa se med seboj razlikujejo po tipih storitev, ki jih ponujajo, in po uporabljenih tehnologijah. V tabeli 2.2 so prikazane njihove osnovne značilnosti.

Organizacija	*Glavni produkti*	*Storitve*
Google	Google App Engine, Gmail, GoogleDocs	PaaS - App Engine za lastne aplikacije, SaaS - GMail, GoogleDocs
EMC	Podpora za DB2, Oracle, MS-SQL, SAP, Sybase ...	IaaS - podpora za VMware virtualizacijo, SaaS za shranjevanje podatkov, arhiviranje, poslovno inteligenco ...
Microsoft	Windows Azure, OfficeLive	PaaS - Windows Azure, SQL Azure, AppFabric, SaaS - Microsoft Live, OfficeLive
Amazon	Amazon Web Services	IaaS - EC2, S3, CloidFront
Salesforce.com	Salesforce CRM, Force.com Platform, Chatter	IaaS - CRM, javna uprava, finance ..., PaaS - The Service Cloud, Apex razvojno okolje
IBM	Smart Business Development/Test Cloud, Smarti Analytics Cloud, Smartin Business Storage Cloud	IaaS - CloudBurst Appliance, PaaS - IBM Websphere Platform

Tabela 2.2: Lastnosti ponudnikov računalništva v oblaku.

2.6 Poslovne in potrošniške spletne aplikacije

Čeprav so namigovanja, da se bodo poslovne in potrošniške spletne strani zbližale, so med njimi še vedno precejšnje razlike. Do razmika prihaja zaradi

različnih zahtev, ki vodijo do uporabe različnih tehnologij [27]. Glavne razlike lahko opredelimo kot:

- **Skalabilnost;** ustvarja najpomembnejšo razliko med obema tipoma spletnih aplikacij. Pri potrošniških aplikacijah se vse vrti okoli skalabilnosti glede števila uporabnikov za eno aplikacijo. Uspešne potrošniške aplikacije, kot so YouTube in Facebook, so uspešno prestale eksponentno rast uporabnikov. Na drugi strani imajo podjetja običajno omejeno število uporabnikov in naraščujoče poslovno-aplikacijske zahteve. Največji izziv za razvijalce poslovnih aplikacij je standardizacija tehnoloških arhitektur in najboljših prijemov na množici aplikacij za minimizacijo razvojnih in vzdrževalnih stroškov. Zato so poslovni razvijalci naklonjeni standardnim in dobro podprtim tehnologijam, čeprav lahko te prinesejo višje stroške licenciranja. Vzdrževanje ima običajno prednost pred učinkovitostjo in enostavnostjo arhitekture.
- **Uporabniška izkušnja** predstavlja naslednjo veliko razliko. V svetu potrošnikov morajo biti aplikacije enostavne in privlačne, da pritegnejo uporabnike. Enostavnost je pogosto pomembnejša kot zaključenost. Pri poslovnih aplikacijah so uporabniki zaposlenci, ki se zanašajo na funkcionalnosti aplikacije, ki so prioriteta. Omejevanje funkcionalnosti na račun estetike je slaba strategija za poslovne aplikacije.
- **Varnost** je pomembna za oba tipa aplikacij, kljub temu so varnostne zahteve za poslovne aplikacije višje. Pri poslovnih aplikacijah so uporabniške vloge (ang. access rule) praviloma številčnejše in bolj zapletene, zato se uporabljajo kompleksnejša varnostna ogrodja, kot je na primer Spring Security.
- **Transakcije** se lahko izvajajo različno glede na tip aplikacije. Zaradi porazdeljenosti podatkovnih strežnikov je lahko izvajanje transakcij drago ali praktično nemogoče. Potrošniške spletne aplikacije si lahko dovolijo in optimizirajo transakcije z delitvijo na manjše neodvisne transakcije, kar lahko v ekstremnih primerih privede do nepravilnih podatkov. Poslovne aplikacije tega ne smejo dovoliti. Sistemi morajo zagotavljati pravilnost podatkov v vsakem trenutku.
- **Povezovanje v celoto** potrošniških aplikacij poteka preko javnih programskih vmesnikov, preko katerih lahko komunicirajo z mnogo različnimi aplikacijami. Vmesniki so običajno enostavni (primer REST). V poslovnem svetu pa mora biti povezovanje aplikacij dobro oblikovano in nadzirano ter ima pogosto vpliv na vse plasti aplikacije.

- **Iskanje** potrošniških aplikacij se navezuje na spletne strani, ki predstavljajo enoličen način predstavitve vsebine in povezave z določeno entiteto. Optimizacija iskalnega pogona (ang. Search Engine Optimization – SEO) je dobro definiran koncept, ki omogoča strukturiranje spletne vsebina tako, da se jo najlažje odkrije. Takšen pristop je neodvisen od aplikacije. V poslovnem svetu pa je iskanje osnovano na zapisih s pripadajočimi lastnostmi. Takšen pristop prinaša boljšo usmerjenost in večjo varnost, vendar je iskanje odvisno od aplikacije. Realizacija medaplikacijskega iskanja je razmeroma dražja.

Poglavje 3

Platforme za razvoj spletnih strani

V teoriji skriptno izvajanje na strani strežnika z uporabo CGI metode, kot je na primer ASP, PHP, Python, zagotavlja elemente, ki zadoščajo gradnji velikih informacijskih sistemov, vendar ima razvijalec preveč infrastukturne odgovornosti. Namesto da bi se posvečal poslovni logiki, mora razvijalec sam skrbeti za upravljanje z viri, transakcijami in ostalimi kompleksnimi nalogami. Če aplikacije razdelimo na različne plasti, da ločimo infrastrukturo od poslovne logike in predstavitve, ogromno pridobimo na obvladljivosti razvoja in vzdrževanja. Takšna razplastitev prinaša še več nalog za razvijalce.

Za poenostavljene in bolj obvladljive razvojne procese so nastale platforme, ki vsebujejo aplikacijske strežnike in vmesne plasti; le-te ločujejo spletni strežnik od predstavitvene, poslovne in podatkovne logike. To poglavje vsebuje analizo obstoječih platform za razvoj spletnih aplikacij. Večina razvijalcev postane veščih ene platforme, le redki imajo dovolj časa, da jih osvojijo več.

3.1 Značilnosti platform

Platformo lahko razumemo kot množico programskih knjižnic napisanih v programskem jeziku, v katerem bo razvita aplikacija. Dandanes je pojem platforme precej širši, saj tehnologije znotraj platforme skrbijo za različne faze - od analize, oblikovanja do procesa razvoja. V industriji sta zelo razširjeni .NET in Java EE platformi. Kot platformo za spletni razvoj obravnavamo tudi LAMP platformo, ki sicer ni tako široko zastavljena kot prej omenjeni, a je vseeno zelo popularna in močna. Platforme nudijo večplastno arhitekturo, na kateri lahko gradimo hitre in skalabilne aplikacije. Modularne komponente rešujejo različne

kompleksne naloge, kot so upravljanje transakcij, virov, možnosti različnih komunikacij, povezave s podatkovnimi bazami ipd. Poenostavljajo razplastitev aplikacij in vsebujejo možnosti uporabe različnih standardiziranih, široko uporabljenih in učinkovitih storitev vmesne plasti s podporo principov hitrega razvoja. Tabeli 3.1 in 3.2 povzemata zahtevane vmesne storitve za poslovne sisteme [4], razdeljene na zanesljivost, proizvodnjo, povezovanje, varnost in razvoj.

Zahteva	*Rešitev*	*Opis*
Zanesljivost	Transparentne prekinitve	Preusmerjanje zahtevkov za neuspele storitve na druge strežnike
	Podpora transakcij	Potrjevanje enot dela ali povrnitev v začetno stanje
	Upravljanje sistema	Sistem za analizo in upravljanje z zmožnostmi
	Visoka dosegljivost	Minimizacija časa nedosegljivosti zaradi napak
	Reprodukcija (ang. replication)	Podvajanje virov za večjo odzivnost in obnovljivost
	Reševanje napak	Detekcija napak storitev, preusmerjanje na druge instance
Proizvodnja	Porazdelitev zahtevkov (Load balancing)	Izmenjavanje zahtevkov med strežniki za optimalno porazdelitev
	Združevanje (ang. clustering)	Sodelovanje večih strežnikov za razdelitev izvajanja zahtevkov
	Niti	Izvajanje zahtevkov skladno znotraj izvajajočega se procesa
	Učinkovitost	Izvajanje zahtevkov z minimalno uporabo virov in latenco
	Skalabilnost	Obdržitev stabilne učinkovitosti pri povečanju zahtevkov
	Deljenje virov	Optimalno deljenje virov med več uporabniki
	Predpomnjenje (ang. caching)	Shranjevanje rezultatov izvajanja za naslednje združljive zahtevke

Tabela 3.1: Prvi del specifikacij vmesnih plasti.

Zahteva	*Rešitev*	*Opis*
Povezovanje	Izvajanje oddaljenih metod	Vmesniki za sinhrono izvajanje metod
	Povezava s sistemom (ang. Back-end integration)	Vmesniki do zunanjih informacijskih sistemov
	Dostop do podatkovnih baz	Shranjevanje in branje podatkov iz podatkovnih baz
	Lokacijska transparentnost (ang. location transparency)	Dovoljevanje zahtevkov po storitvah različnih uporabnikov
	Podpora večih protokolov	Sestavljanje več protokolov v enoten vmesnik
	Pošiljanje sporočil	Asinhrona komunikacija preko pošiljanja sporočil
	Zapuščinska (ang. legacy) povezljivost	Vmesniki do zastarelih tehnologij
Varnost	Beleženje in pregledovanje	Beleženje pomembnih aktivnosti, ki se dogajajo znotraj sistema
	Preverjanje dovoljenj	Preverjanje identitete in ščitenje virov
Razvoj	Hitro razvijanje	Zanesljive aplikacije je mogoče razviti hitro
	Dinamično nameščanje	Izvajajoče aplikacije se lahko posodobijo brez prekinitev
	Razplastitev skrbi	Funkcijski prispevki so ločeni po modulih
	Modularnost	Izolirane spremembe minimalno vplivajo na ostale dele sistema
	Ponovna uporaba	Isti moduli so lahko uporabljeni v več aplikacijah
	Programska skalabilnost	Programska opreme ostane obvladljiva z naraščanjem velikosti
	Prenosljivost	Moduli se lahko izvajajo na različnih platformah brez sprememb
	Standardizacija	Tehnologija ima več ustrežljivih ponudnikov

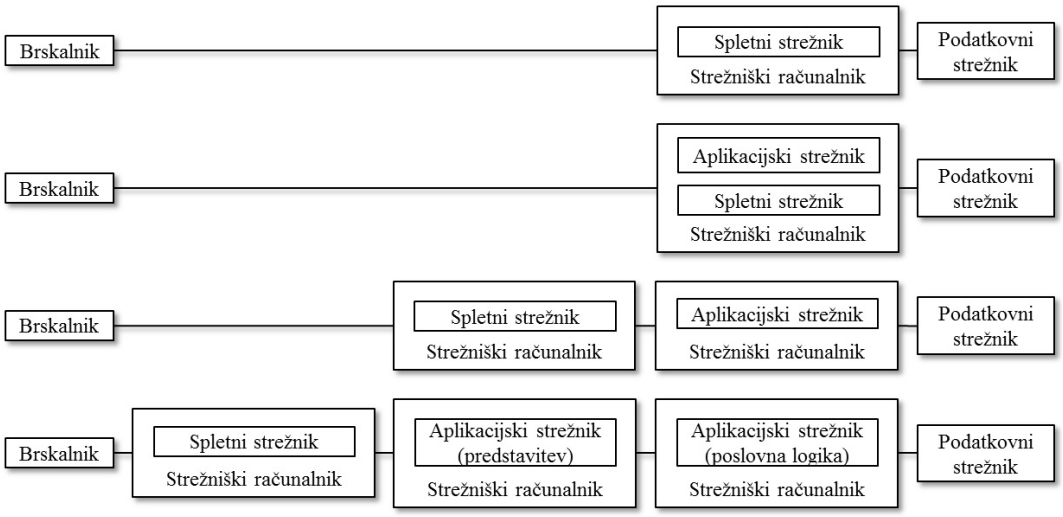
Tabela 3.2: Drugi del specifikacij vmesnih plasti.

3.1.1 Večplastna arhitektura

Večje kompleksnejše aplikacije so tipično zgrajene na arhitekturah, ki uporabljajo aplikacijske strežnike in vmesne plasti. Slika 3.1 prikazuje nekaj razplastitev, ki se pogosto uporabljajo za spletne aplikacije. Razslojena aplikacija ima običajno predstavitveno, poslovno in podatkovno plast [11].

Predstavitvena plast

Predstavitvena plast je odgovorna za prikazovanje uporabniškega vmesnika z uporabo razredov in objektov poslovne plasti.



Slika 3.1: Primeri večplastnih arhitektur za spletne aplikacije. V vsaki mora biti prisoten podatkovni in spletni strežnik [4].

Poslovna plast

Poslovna plast je odgovorna za dostopanje, spreminjanje in branje podatkov ter pošiljanje rezultatov predstavitveni plasti. Poleg tega mora skrbeti za procesiranje prejetih podatkov.

Poslovno plast lahko delimo na dve podplasti: plast poslovne logike (ang. Business Layer Layer – BLL) in plast za dostop do podatkov (ang. Data Access Layer – DAL). Plast s poslovno logiko je nad slednjo, kar pomeni, da BLL uporablja razrede in objekte DAL. DAL skrbi za dostopanje do podatkov in njihovo odpošiljanje BLL.

Podatkovna plast

Podatkovna plast predstavlja podatkovno bazo ali podatke same.

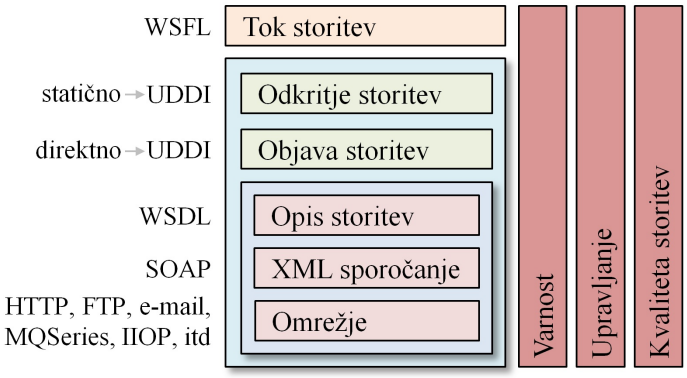
Logične in fizične plasti

Plasti lahko delimo tudi na logične in fizične. Logične plasti so ločene v različnih knjižnicah ter nameščene na istem strežniku. Fizične plasti pa predstavljajo knjižnice, ki so nameščene na različnih strežnikih z uporabo dodatnih modulov, ki skrbijo za komunikacijo med plastmi (npr. oddaljen dostop ali spletne storitve).

3.1.2 Spletne storitve

Tehnologija spletnih storitev (ang. Web Services – WS) nudi jezikovno in sistemsko neodvisen programerski model, ki pospešuje povezovanje različnih aplikacij. Spletne storitve lahko enostavno uporabimo kot ovojno tehnologijo okrog obstoječih aplikacij in informacijskih tehnologij, zato je nameščanje in sestavljanje novih aplikacij hitrejše in enostavnejše.

Klasične spletne storitve so zgrajene na obstoječih in novih tehnologijah, kot so: HTTP, XML, SOAP, WSDL in UDDI. SOAP predstavlja protokol za izmenjavo informacij med komponentami, zapisanimi v XML obliki. WSDL predstavlja opisni jezik spletnih storitev, s katerim odjemalcu poda njihove funkcionalnosti. UDDI pa je imenik spletnih storitev. Slika 3.2 prikazuje strukturo običajnih spletnih storitev.



Slika 3.2: Struktura spletnih storitev.

Pristop spletnih storitev je bil prvotno ustvarjen za porazdeljene komponentne arhitekture, ki bi lahko povezovale različne računalniške sisteme s poudarkom na poslovnih arhitekturah in za digitalne storitve. Zaradi tega so standardi spletnih storitev relativno težki, tako sta definicija vmesnika (WSDL) in definicija sporočil (SOAP) precej kompleksni instanci XML dokumentov. Zaradi tega spletne storitve zahtevajo določeno računsko moč, pasovno širino in količino pomnilnika.

REST

Leta 2001 je avtor HTTP protokola Roy Fielding predstavil arhitekturni pristop prenosa reprezentativnega stanja (ang. Representational State Transfer – REST), ki si s spletnimi storitvami deli prizadevanja po povezovanju porazdeljenih komponent, vendar je primarni cilj REST doseči to na učinkovitejši

in enostavnejši način. REST uporablja svetovni splet kot aplikacijsko platformo in popolnoma izkorišča lastnosti HTTP, kot so avtentikacija, avtorizacija, kriptiranje, kompresija in predpomnjenje. Poleg tega prinaša storitve v brskalnik brez potrebe po generiranju WSDL datotek. Te prednosti utemeljujejo naraščujočo uporabo REST storitev za Web 2.0 storitve, kot so Flickr, Facebook, Del.icio.us, Doodle, Amazon itd. Tehnologije, ki upoštevajo REST arhitekturne pristope, imenujemo RESTful tehnologije. Specifikacija REST vsebuje dve osnovni lastnosti:

- aplikacijski model je preslikan iz operacijsko-usmerjenega (kot pri klasičnih spletnih storitvah) v podatkovno-usmerjenega; vse, kar storitve nudijo, postanejo viri, ki so nedvoumno identificirani z URI[1];
- štiri osnovne operacije, ki jih nudi HTTP (GET, POST, PUT, DELETE), so edine veljavne operacije nad viri; predstavljajo enoten vmesnik do virov storitev.

Rest tehnologija ima tudi slabosti. Enostavnejša zasnova presenetljivo oteži kreiranje kompleksnejših storitev. Klasične spletne storitve se bolje odrežejo pri obsežnih poslovnih aplikacijah z daljšo življensko dobo in kompleksnejšimi zahtevami. Zaradi enostavnosti, lažje zasnove in enotnega vmesnika se RESTful storitvam daje prednost pri taktičnem, namenskem povezovanju preko svetovnega spleta[35].

Storitveno usmerjevna arhitektura – SOA

Storitveno usmerjena arhitektura (ang. Service Oriented Architecture – SOA) je fleksibilna množica razvojnih vodil skozi faze sistemskega razvoja, povezovanja in izvajanja. Cilj je razplastitev funkcionalnosti kot niz neodvisnih storitev, ki se lahko uporabljajo na različnih sistemih in poslovnih domenah. Agilnost, ki jo prinaša SOA, temelji na uporabi različnih transportnih mehanizmov, modelu komunikacije, zagotavljanju varnosti in zanesljivosti.

Storitvena arhitektura (kot vrsta porazdeljenega sistema) je sestavljena iz spletnih storitev, ki predstavljajo komponente sistema. Glavna prednost SOA vodenja je ponovna uporaba storitev, kar privede do večje fleksibilnosti in nizkih stroškov vzdrževanja.

Razvijalci za SOA sisteme običajno uporabljajo klasične spletne storitve, ki jih je industrija dobro sprejela. Lahko pa uporabimo tudi tehnologije, kot

[1]Enotni označevalnik vira (ang. Uniform Resource Identifier)

so REST, CORBA[2], Jini[3] ipd.

3.2 Java EE

Java je popularen objektno usmerjen jezik razvit in izdan leta 1995 pri nekdanjem Sun Microsystems (sedaj v lasti Oracle Corporation). Prvotno je bila Java prisotna na spletu kot tehnologija na strani odjemalca za poganjanje varnih appletov znotraj brskalnika, vendar je veliko večji pečat pustila na strežniški strani. Prenosljivost med platformami, ki podpirajo javanski navidezni stroj (ang. Java Virtual Machine – JVM), ima še posebej večjo težo na spletu kot najbolj raznolikem računalniškem sistemu. V primerjavi s skriptnimi jeziki je povečana zanesljivost, ker ima statično varne tipe. Okolje podpira spreminjanje med izvajanjem (ang. runtime reflections), ločitev implementacije vmesnikov in nalaganje dinamičnih razredov, ki predstavljajo gradbene elemente za komponente in aplikacijske strežnike. Nobena od teh značilnosti ni novost, njihova vključitev v eno samo okolje se je zgodila ob pravem času, zato je Java hitro postala popularna izbira.

Java Platform, Enterprise Edition (krajše Java EE) vsebuje poleg Java Platform, Standard Edition (krajše Java SE), ki predstavlja osnovno platformo za razvijanje v jeziku Java, tudi knjižnice za razvoj porazdeljenih, večplastnih spletnih aplikacij, osnovanih na modularnih komponentah, ki se izvajajo na aplikacijskem strežniku. Na sliki 3.3 je predstavljena arhitektura Java EE platforme.

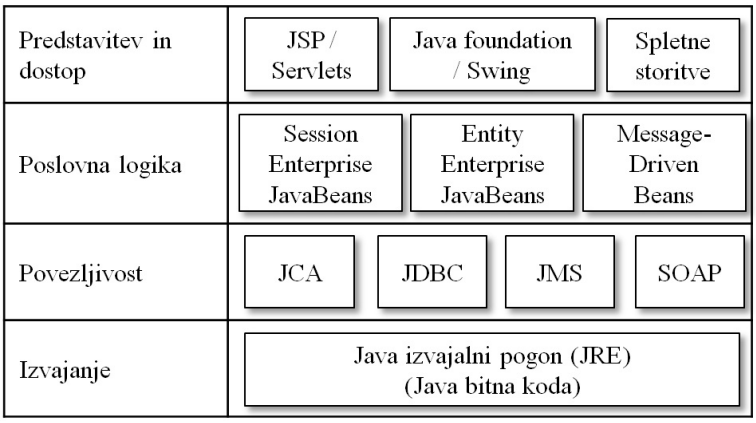
Ena najpomembnejših lastnosti Java EE platforme je število storitev na vmesni plasti, ki jih nudi razvijalcem. Množica storitev je povzeta v tabeli 3.3.

Veliko javanskih spletnih aplikacij uporablja poleg lastnih tudi tuje in odprtokodne komponente. To je zelo privlačna možnost, saj zmanjša napor razvijalcev. Poleg tega modularen pristop ustreza javanskemu okolju. Primeri komponent, ki se uporabljajo za ključne aplikacijske funkcionalnosti, so:

- Avtentikacija – JAAS, Spring Security;
- Prezentacijska plast – SiteMesh, Tapestry;
- Objektno-relacijsko preslikovanje (ang. Object-relational mapping – ORM) – Hibernate in

[2]Standard, ki omogoča povezovanje programskih komponent, napisanih v različnih jezikih in ki se izvajajo na različnih sistemih

[3]Omrežna arhitektura za kreiranje porazdeljenih sistemov v obliki modularnih, sodelujočih storitev.



Slika 3.3: Java EE arhitektura [34].

- Beleženje dogodkov – Log4J.

Java EE se je precej razširil, obstaja tudi veliko ponudnikov aplikacijskih strežnikov. Platforma je zelo skalabilna, če je pravilno uporabljena, vendar je zaradi velikosti in kompleksnosti arhitekture lahko težavna za začetnike. Krivulja učenja je za Java EE razvijalce lahko precej dolga, še posebej, če uporabljamo vse Java EE specifikacije.

3.3 .NET

.NET je razvojno ogrodje za Windows operacijske sisteme. Značilnosti .NET so primerljive z značilnostmi javanske platforme. Čeprav je bil zasnovan kot prenosljivo okolje se uporablja predvsem na Windows operacijskih sistemih. Zaradi ogromnega števila nameščenih Windows operacijskih sistemov je .NET, od svoje prvotne izdaje leta 2001 naprej, postopoma privabljal razvijalce namiznih aplikacij.

Osnovni komponenti .NET platforme sta Common Language Runtime (CLR) in .NET množica knjižnic. CLR je virtualni stroj, ki dinamično prevaja Microsoftov vmesni jezik (ang. Microsoft Intermediate Language – MSLI) bitno kodo v kodo, ki se bo izvajala ob zagonu. .NET je večjezično okolje, saj se lahko več jezikov prevede v MSLI kodo in izvaja v CLR. Sistem običajnih tipov (ang. Common Type System – CTS) opredeljuje, kako so tipi definirani ter kako se uporabljajo v CLR, kar predstavlja osnovo za mešanje tipov med moduli, implementiranimi v različnih jezikih. Samoopisljive komponente, imenovane assemblies znotraj .NET, so upravljane in izvajane v CLR. Platforma

Specifikacija	*Opis*
Enterprise JavaBeans (EJB)	Upravljanje Java komponent na strežniku.
Java Naming and Directory Services (JNDI)	Vmesnik za sledenje virom, Java komponentam.
Java Database Conectivity (JDBC)	Vmesnik za povezavo s relacijskimi podatkovimi bazami.
Java Message Service (JMS)	Omogoča komunikacijo med Java komponentami s sporočili.
Remote Method Invocation (RMI)	Vmesnik za komunikacijo med porazdeljenimi objekti.
Java Transaction API (JTA)	Mehanizem za transakcijsko procesiranje.
Java Servlets and JavaServer Pages (JSP)	Komponenta za razširitev spletnega strežnika.
JavaMail	Mehanizem za pošiljanje spletne pošte.
Java API for XML Parsing (JAXP)	Razčlenjevanje XML dokumentov.
Java EE Connector Architecture	Arhitekturno ogrodje za vključevanje virov v Java EE okolje.
Java Authentication and Authorization Setvice (JAAS)	Osnovni mehanizem za avtentikacijo in avtorizacijo uporabnikov pri dostopanju do virov.

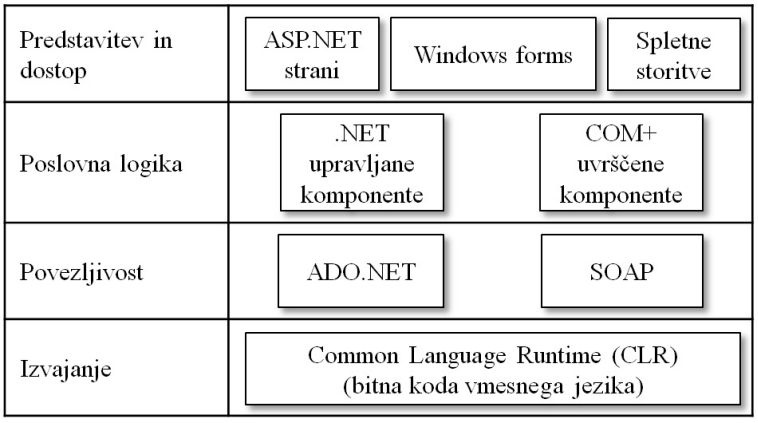
Tabela 3.3: Java EE specifikacije.

vsebuje knjižnice razredov, ki nudijo podobne funkcionalnosti kot programski vmesniki v java platformi. Primarni jezik .NET platforme je C#, ki je zelo podoben Javi, lahko pa programiramo tudi v jezikih: VisualBasic, C++, Python, itd. Tabela 3.4 primerja značilnosti Java in .NET platform.

3.4 LAMP

LAMP predstavlja skupek brezplačnih, odprtokodnih rešitev, prvotno sestavljen iz Linux operacijskega sistema, Apache HTTP strežnika, MySQL podatkovne baze in PHP programskega jezika, kot osnovnih komponent pri gradnji spletnih strani. Kombinacija rešitev v LAMP paketu se lahko razlikuje. Namesto PHP lahko uporabimo Perl, Python ali Ruby jezik. Prav tako ni nujno, da uporabljamo Linux operacijski sistem in MySQL podatkovno bazo. Nekateri ločujejo LAMP arhitekturo od Ruby v paketu s Ruby on Rails ogrodjem (krajše Ruby/RoR), v tej nalogi pa bomo obravnavali Ruby/RoR kot segment LAMP arhitekture, podobno kot Python v navezi z Django ogrodjem.

Medtem ko Java in .NET razširjata klasični vmesnik komunikacije strežnika z odjemalcem, se LAMP platforma pojmuje kot bolj direktno implementacijo



Slika 3.4: Arhitektura .NET platforme [34].

CGI modela.

Tabela 3.5 prikazuje, kateri jeziki se na spletnem strežniku prevajajo v izvorno ali bitno kodo ter kateri se tolmačijo.

3.4.1 PHP

PHP (Hypertext Preprocessor) je široko uporabljen, splošno namenski skriptni jezik, ki je bil prvotno razvit za spletno razvijanje dinamičnih strani. PHP koda je podobno kot pri ASP in JSP vključena v HTML kodo strani, ki jo spletni strežnik z ustreznim modulom interpretira v spletni dokument. Prvotno je PHP kratica pomenila Personal Home Page. Razvoj se je začel leta 1994 na osnovi Perl skript. Od tedaj se je jezik velikokrat spremenil. Razvoj PHP-ja podpira Zend Technologies, ki so naredili tudi zelo popularen Zend Frameworko za PHP. PHP5 ima izboljšano podporo za objektno usmerjeno programiranje, a še vedno nima polne podpore za Unicode[4]. PHP lahko gostujemo na večini spletnih strežnikov in številnih operacijskih sistemih. V osnovi deluje kot filter, prebere vhod iz datoteke ali vhodnega toka, ki vsebuje besedilo ter/ali PHP ukaze ter na izhod pošlje drugi tok podatkov, največkrat HTML dokument. Od verzije PHP4 se vhodni tok prevede v bytecode za izvajanje v Zend Engine, kar je povečalo hitrost delovanja. PHP se izvaja na Apache strežniku preko mod_php modula, ki je najbolj popularen modul za ta strežnik.

V PHP-ju so razvite številne zelo priljubljene spletne strani, kot so Facebook, Wikipedia, Digg, Joomla, WordPress, Drupal in Moodle ter različne

[4]Standard za kodiranje znakov v računalništvu.

Storitev, značilnost	*Java EE*	*.NET*
Jezik	Java	Primarno C#, podprti so tudi drugi jeziki
Izvajanje	Java Virtual Machine (JVM)	Common Language Runtime (CLR)
Strežniške komponente	Enterprise JavaBeans (EJB)	Assemblies, komponente s katerimi upravlja .NET CLR
Komponente na odjemalcu	JavaBeans	.NET razredi
Imenovanje	Java Naming and Directory Interface (JNDI)	Active Directory (AD)
Podatkovno povezovanje	JDBC	ADO.NET
Upravljanje transakcij	Java Transaction Server (JTS)	COM+, EnterpriseServices
Sporočanje	Java Messaging Services (JMS)	Microsoft Message Queue (MSMQ)
Oddaljeno pozivanje (ang. invocation)	Remote Method Invocation (RMI)	Remote Method Invocation (RMI)
RIA	Swing API, JavaFX	Silverlight
Strežniške skripte	Java Servlets and JavaServer Pages (JSP)	ASP.NET, Web Forms, ASP.NET MVC
Elektronska sporočila	JavaMail	Messaging API (MAPI)
XML	Java API for XML Parsing (JAXP)	.NET Framework System.XML
HTTP pogon	Več aplikacijskih strežnikov	IIS

Tabela 3.4: Primerjava Java EE in .NET specifikacij.

komponente. Mnoge od teh nudijo rešitve za običajne aplikacijske zahteve, kot so na primer:

- Spletni forumi – PHPBB, PHP-Nuke,
- Administrativne vhodne točke – PHPMyAdmin,
- Spletna pošta – SquirrelMail, IlohaMail,
- Galerije slik – Gallery,
- Nakupovalni vozički – osCommerce, ECW-Shop,
- Wiki strani – MediaWiki, WakkaWikki

Tehnologija	*Jeziki*	*Lastnosti, uporabe*
Prevedeni v izvorno kodo	C, C++	Najboljša učinkovitost, nizka uporabnost, CGI skripte in razširitve spletnega strežnika
Tolmačeni jeziki	PHP, Perl, Python, Ruby	Slabša učinkovitost, CGI skripte
Prevedeni v bitno kodo	Java, PHP4+, .NET jeziki	Dobra učinkovitost, razširljive knjižnice jezika

Tabela 3.5: Programski jeziki na strežniku.

3.4.2 Ruby

Ruby jezik je dinamičen, splošno namenski, objektno usmerjen programski jezik s sintakso, navdahnjeno po Perl in Smalltalk jezikih. Zaradi tega je podoben jezikom Python, Lisp. Aplikacije lahko gostimo na Apache spletnem strežniku z modulom mod_ruby (implementacija CGI protokola), ki je tolmač ruby jezika za Apache strežnik. Velike zasluge za popularnost jezika ima ogrodje Ruby on Rails, ki je namenjeno uporabi v povezavi z metodami agilnega programiranja za hitro razvijanje spletnih strani.

3.4.3 Python

Python je splošno namenski visoko nivojski programski jezik. Njegov poudarek je na lažjem branju kode ter na kombinaciji "velike moči z zelo jasno sintakso". Python podpira številne programerske paradigme, primarno objektno programiranje, imperativno ter do neke mere tudi funkcijsko programiranje. Vsebuje sistem dinamičnih tipov ter samodejno upravljanje s pomnilnikom, podobno kot pri Perl, Ruby, Scheme in Tcl. Python se podobno kot drugi dinamični jeziki uporablja kot skriptni jezik, ampak je široko uporaben tudi v druge namene. Leta 2008 je izšla verzija 3.0, ki kot prva ni kompatibilna s prejšnjimi verzijami. Python je pogosto uporabljen kot skriptni jezik za spletne aplikacije, to je možno preko mod_wsgi za Apache spletni strežnik, ki implementira CGI vmesnik. Modul mod_wsgi lahko uporabimo za gostovanje vseh Python aplikacij, ki podpirajo Python WSGI (Web Server Gateway Interface) vmesnik, kar vključuje tudi Django framework. Rešitve, napisane v Pythonu, lahko najdemo v podjetjih, kot so Google, CERN, NASA in spletnih straneh

YouTube, Yahoo.

3.5 Primerjava platform

Zelo težko je primerjati platforme in nemogoče določiti, katera je najboljša. Vsaka ima svoje dobre in slabe strani, ki jih bom skušal prikazati skozi kriterije, ki sledijo.

3.5.1 Stroški

Ko se podjetja odločajo, katero platformo izbrati, je vselej prisoten kriterij cene, ki jo sestavljajo stroški lastninjenja, začetne pridobitve, delovanja in vzdrževanja.

LAMP platforma temelji na odprtih in brezplačnih rešitvah. PHP gostovanje je najcenejše, poleg tega so jeziki PHP, Python in Ruby enostavni, kar prinaša lažje vzdrževanje in krajšo krivuljo učenja. Programiranje v Python-u zahteva od 2 do 3-krat manj vrstic kode kot v Javi ali .NET-u. Ruby/ROR je zaradi enostavnega vzdrževanja in najkrajše krivulje učenja verjetno najcenejša izbira.

Java je tako kot LAMP platforma osnovana na odprtih in brezplačnih rešitvah. Kljub temu da obstajajo komercialne aplikacije kot WebSphere in WebLogic, imajo le-te močne odprtokodne brezplačne alternative, osnovane na Tomcat in JBoss strežnikih. Obsežna baza dokumentacij in odprta razvojna skupnost predstavlja Javo kot poceni rešitev, primerno za začetne projekte.

Stroški lastninjenja so najšibkejša točka .NET platforme. Potrebno je kupiti licence za Visual Studio, MSDN naročnino, MS SQL strežnik, IIS strežnik in Windows operacijski sistem. Za začetne projekte pa se lahko ti stroški močno zmanjšajo [28].

3.5.2 Hitrost razvijanja

S tem kriterijem skušamo odgovoriti, katera arhitektura nudi najhitrejši, najenostavnejši in razmeroma kvaliteten razvoj tipičnih funkcionalnosti spletnih aplikacij. Velik pomena predstavljajo dokumentacija in razni spletni viri. Pomembno je tudi, kako široko zastavljena je platforma. Namenske tehnologije so lažje razumljive in s tem enostavnejše za uporabo. Vendar manj, kot je

tehnologija razširjena, večje je tveganje uporabe in učenja, saj lahko ob kompleksnih nalogah pride do večjih zapletov.

LAMP platforma omogoča najhitrejši razvoj spletnih aplikacij. PHP se je z verzijo 5.x veliko izpopolnil kot objektno usmerjen jezik. Podporo mu dajejo zelo razširjena ogrodja, kot so Symphony, Zend Framework in mnogi drugi. Ena močnejših točk je tudi PEAR/PECL mehanizem, ki omogoča lažje in učinkovitejše razvijanje ter širjenje PHP razširitev. Jezik Ruby je z ogrodjem Ruby on Rails (krajše RoR ali Rails) predstavil nove standarde za ogrodja spletnega razvijanja. RoR rešuje tipične naloge spletnega razvijanja zelo hitro in enostavno. Vsebuje množico receptov, modelov in programskih komponent, ki rešujejo običajne probleme spletnega razvoja. Po drugi strani pa lahko postane reševanja kompleksnejših težav v RoR velika težava. Tudi Python z ogrodji, kot so Pylons, Django in Turbogears, zagotavlja dobro osnovo za hitro spletno razvijanje. Python ima eno največjih izbir knjižnic za različne uporabe, tako je mogoče lažje in hitreje reševati kompleksne in nestandardne probleme.

Platforma .NET ima močno dokumentacijo in ostale izobraževalne vire. Platforma je zelo prijazna do razvijalca od samega začetka. Vsebuje obširno množico razvijalskih orodij, knjižnic in vzorce ter tako nudi impresivno hiter način za gradnjo spletnih aplikacij. Slabost je ta, da je prilagojen na ozko skupino tehnologij in programske opreme. Poleg tega je ta platforma kompleksnejša kot LAMP, zato je krivulja učenja daljša.

Čeprav je Java EE bolj zrela kot ostale platforme in je na voljo veliko dokumentacije, je njena krivulja učenja najdaljša. Zaradi splošnosti je kompleksna in močno odvisna od pravilnih nastavitev. Tako kot LAMP platforma ima veliko število ogrodij, ki so osnovana na dobro definirani in široki množici razvojnih vzorcev. Vsebujejo podobne rešitve za specifične probleme, kot so optimizacija uporabe pomnilnika, optimizacija hitrosti procesiranja, upravljanje spletnega delovnega toka, podatkovno predpomnjenje (ang. data caching), varnost ipd. Ogrodja so splošno naravnana za širok nabor možnih aplikacij. To je tako prednost kot slabost, saj so zaradi splošnosti počasnejša v hitrosti razvoja.

3.5.3 Hitrost izvajanja in skalabilnost

Zelo pomembeni kriteriji so hitrost izvajanja, odzivnost aplikacije ter kako se ta vede ob povečanem številu zahtevkov ali ob razširjanju funkcionalnosti.

Zelo poučna je situacija, v kateri se je znašla razvojna skupina, ki je napisala Twitter [37]. Ob velikem povečanju številu uporabnikov so razmišljali celo o spremembi platforme.

Skriptni jeziki so praviloma počasnejši od jezikov, ki se prevajajo v bitno kodo. Kljub temu so spletne aplikacije, napisane v jeziku Python, hitre zaradi uporabe C/C++ jezika pri implementaciji knjižnic na nižjem nivoju in lažji specifikaciji WSGI modula v primerjavi z Java EE Servlet tehnologijo. Python lahko najdemo pri časovno kritični spletni aplikaciji, kot je YouTube, medtem ko sta PHP in Ruby jezika počasnejša [16].

Platforma Java EE je v svoji dolgoletni zgodovini razvoja prestala šest različic. Rezultat tega je močno optimizirana učinkovitost, ki je v večini primerov zelo blizu hitrosti namiznih aplikacij. Obstaja tudi precej orodij, ki pomagajo z upravljanjem skalabilnosti in večjim številom zahtev. Za razvoj realno-časovnih aplikacij je Java EE najboljša izbira.

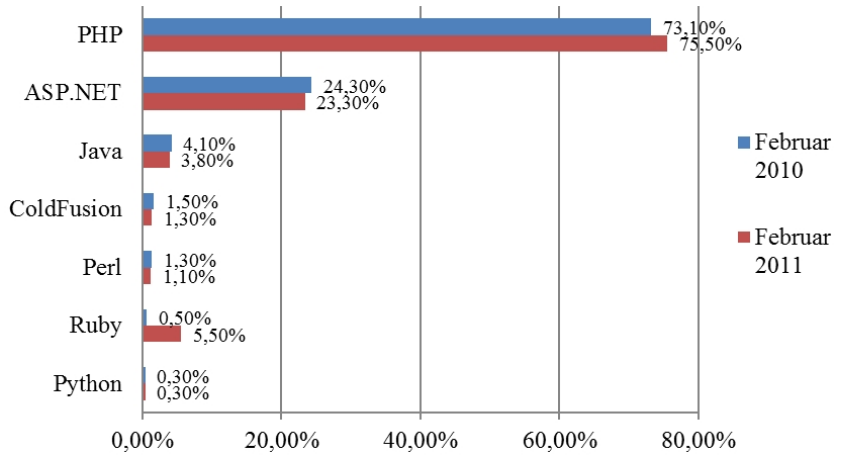
.NET platforma je precej težka, ker jo sestavlja veliko različnih plasti. V primerjavi z Java EE je manj zrela in s tem tudi počasnejša ob večjem številu zahtevkov.

3.5.4 Sprejetje v industriji

Sprejetje v industriji pove, koliko je tehnologija standardizirana v industriji. Pomembni so kriteriji kot velikost podporne skupnosti, dosegljivost dokumentacije, učnih materialov, koliko svetovno znanih spletnih strani uporablja tehnologijo. Statistiko uporabe tehnologije merijo različne spletne strani. Delujejo podobno kot spletni iskalniki, vendar se namesto vsebine osredotočajo na tehnologije, ki jih spletna stran uporablja.

Graf na sliki 3.5 prikazuje statistiko uporabe strežniških ogrodij v zadnjem letu, kot jo beleži W3Tech. PHP ima izrazito prednost pred ASP.NET in Java tehnologijama. Poleg tega se je PHP delež v zadnjem letu povečal, pri ostalih pa je večinoma padel.

Zaradi manjših zahtev gostovanja je LAMP platforma kmalu postala zelo priljubljena. Najbolj izrazite PHP aplikacije so Facebook, Wikipedia, Yahoo, Digg in Wordpress. Večina najbolj razširjenih sistemov za upravljanje z vsebino (ang. Content Management Systems – CMS) je napisanih v PHP jeziku, kar verjetno močno vpliva na statistiko uporabe PHP kot strežniškega jezika. Python je prisoten v YouTube, reddit, Yandex ter različnih Google produktih.



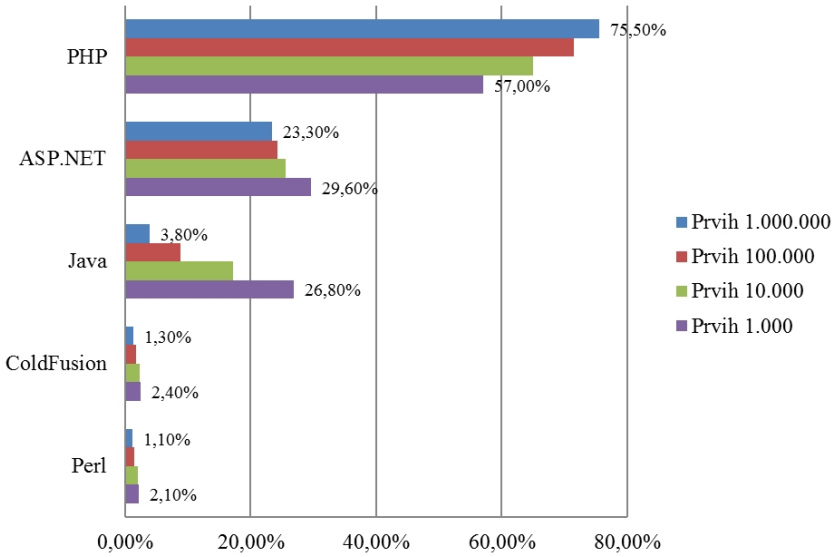
Slika 3.5: Delež uporabe strežniških tehnologij v zadnjem letu [29].

Največji predstavnik Ruby/RoR pa je spletna aplikacija Twitter.

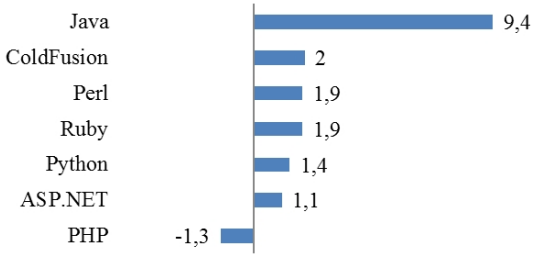
Java EE platforma se je izkazala kot pravilna izbira za spletno trgovanje, bančništvo, ERP in CRM sisteme. Predstavlja jedro spletnih aplikacij, kot so EBay, AdWords, Gmail, Amazon, Flickr in mnogi drugi.

Čeprav je na tisoče nakupovalnih in intranetnih aplikacij napisanih v .NET, je manj začetnih Web 2.0 projektov v primerjavi z Java in LAMP tehnologijama. Najbolj znani izmed teh so myspace.com, plentyoffish.com, msn, microsoft, nytimes.

Statistični podatki so precej bolj zanimivi, ko vpeljemo podatke o oceni spletnih strani. Oceno spletne strani računa spletna storitev Alexa [30] na podlagi števila obiskov dnevno in mesečno. Če analiziramo pridobljene podatke na sliki 3.6, opazimo, da se delež PHP jezika manjša, delež ASP.NET in še izraziteje Java, pa poveča. Slika 3.7 prikazuje faktor spremembe deleža uporabe tehnologije pri straneh z boljšo in povprečno oceno. Delež Java tehnologije pri najbolje ocenjenih tisoč straneh je 9.4-krat večji kot pri najbolje ocenjenenih milijon straneh. Delež PHP tehnologije pa je 1.3-krat manjši.



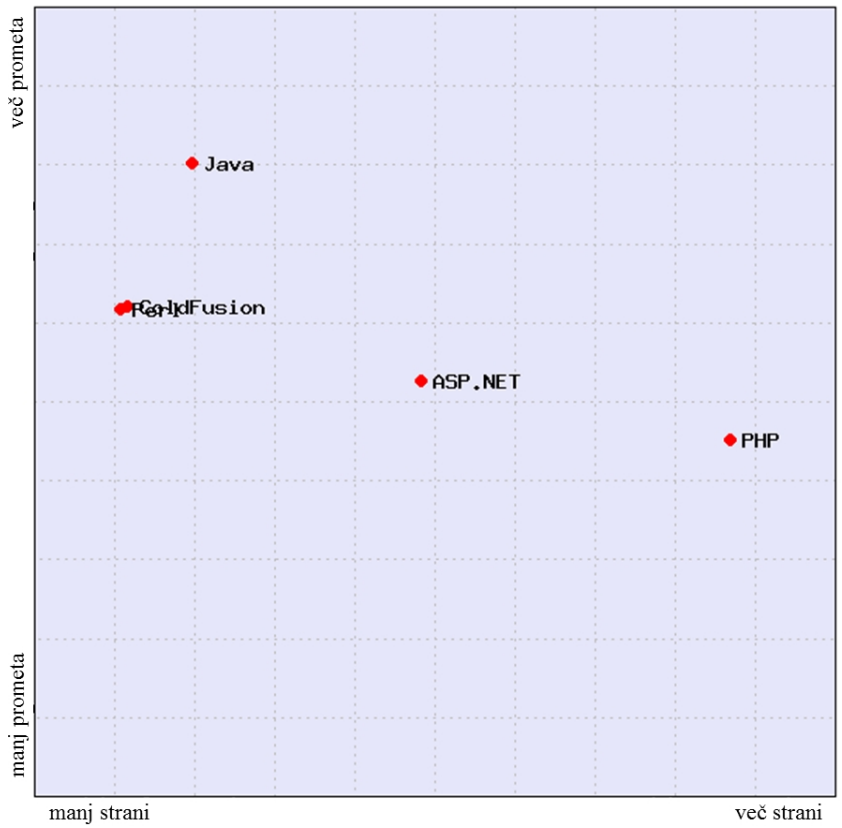
Slika 3.6: Delež strežniških tehnologij glede na ocene spletnih strani [29].



Slika 3.7: Faktor spremembe deleža strežniških tehnologij glede na ocene spletnih strani [29].

W2Tech nudi tudi zanimiv prikaz položaja tehnologije na trgu glede na število spletnih strani in količino prometa najbolj priljubljenih strežniških tehnologij. Iz grafa na sliki 3.8 je razvidno, da se PHP uporablja pri večji množici spletnih strani, ki imajo manj spletnega prometa, Java pa se uporablja pri manjši množici spletnih strani, a imajo le-te več prometa. ASP.NET tehnologija pa je nekje vmes.

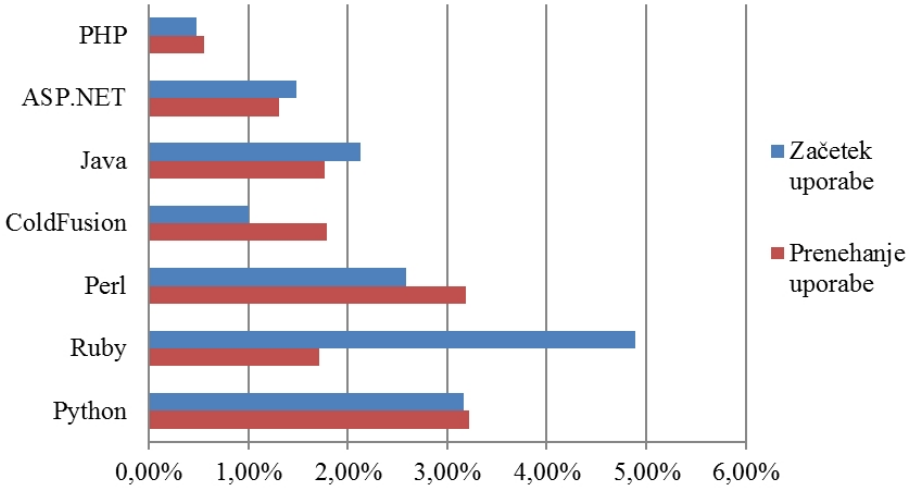
Še ena zanimiva statistika je statistika sprememb tehnologije. Graf na sliki 3.9 za vsako strežniško tehnologijo prikaže kakšen, je delež vseh sprememb, ki so dotično tehnologijo zamenjali z drugo, ter kakšen je delež vseh sprememb, kjer so prejšnjo tehnologijo zamenjali z dotično.



Slika 3.8: Položaj strežniških tehnologij na trgu glede na količino prometa (navpična os) in količino spletnih strani (vodoravna os) [29].

3.5.5 Razvojna okolja

Platformi Java EE in LAMP lahko razvijamo na praktično vseh, .NET pa le na Windows operacijskem sistemu. Zelo pomembno orodje pri razvoju predstavljajo razvojna okolja (ang. Integrated Development Enviroment – IDE). Vsa sodobna razvojna okolja temeljijo na uporabi dodatkov (ang. plug-ins), s katerimi lahko na relativno lahek način razširimo razvojno okolje. Poleg tega večina razvojnih okolij defnira komponente, kot so urejevalniki (ang. editors), pregledovalniki (ang. views), čarovniki (wizards), predloge (ang. templates), akcije (ang. actions) itd. Lahko rečemo, da na novodobna razvojna okolja ne moremo gledati več kot zgolj na zaključene produkte, temveč so predvsem razširljiva ogrodja, ki jih lahko razvijalci prilagajajo svojim potrebam v veliko večji meri, kot je bilo to mogoče do sedaj [31].



Slika 3.9: Delež sprememb uporabe strežniških tehnologij [29].

Privzeto razvojno orodje za .NET platformo je Visual Studio, ki je na voljo v več različicah, ki so zelo dobro povezane z .NET storitvami in funkcionalnostmi. IDE že v osnovni različici vsebuje veliko funkcionalnosti (na primer WYSIWYG[5] urejevalnik spletnih strani), ki jih ostala orodja nimajo ali pa moramo namestiti vtičnik. Različica Express je na voljo brezplačno.

Za razvijanje na LAMP in Java platformi imamo na voljo precej več možnosti. Najpogosteje uporabljeni Java IDE-ji so brezplačni Eclipse, Netbeans ter licenčni JBuilder, IntelliJ, JDeveloper. Največ ponovno uporabnih komponent ponujata odprtokodni Eclipse in Netbeans, kar je verjetno tudi razlog za njuno veliko popularnost. Obe orodji lahko uporabljamo z namestitvijo ustreznih vtičnikov, za razvijanje spletnih aplikacij v PHP, Python in Ruby jezikih.

Na večjih projektih ali v večjih skupinah običajno aplikacijsko logiko razvija razvijalec, grafično podobo pa oblikovalec. V takih primerih oblikovalci običajno uporabljajo specializirano programsko opremo, namenjeno izključno oblikovanju. Microsoft je predstavil Expression Studio paket, ki vsebuje program Expression Web za oblikovanje običajnih spletnih strani ter Expression Blend, s katerim lahko oblikujemo Silverlight aplikacije. Podobno je Adobe predstavil Flex Studio za gradnjo Flex aplikacij in Dreamweaver za gradnjo HTML strani. Poleg teh obstajajo tudi priljubljeni CoffeeCup, Amaya, SeaMonkey, Kompozer, Flux in mnogi drugi HTML urejevalniki.

[5]Predogled in urejanje dokumenta, kot bo prikazan v brskalniku.

3.5.6 Kompleksnost zahtev

Kriterij predstavlja kompleksnost problema, ki ga razvijalci skušajo rešiti. Platformi .NET in Java sta bolj široko zastavljeni kot LAMP. Vsebujeta več rešitev vmesnih storitev za poslovne sisteme, zato se običajno uprabljata pri razvoju kompleksnejših sistemov. Enostavnejše aplikacije z manj dinamične vsebine lahko razvijamo v enem izmed skriptnih jezikih z uporabo ustreznega ogrodja. Vendar pa se z večanjem interaktivnosti in podatkovnih zahtev skalabilnost namenskih programerskih medot slabša. Industrijsko močni aplikacijski razvijalski platformi Java in .NET, zgrajeni na več let namenskih izkušnjah, skušata olajšati aplikacijski razvoj. Na žalost je krivulja učenja za učinkovito uporabo ene izmed teh platform dolga in strma. Kompleksnost lahko opredelimo na tri razrede aplikacij:

- **Novodobne spletne aplikacije** se pogosto povezujejo z izrazom Web 2.0, ki predstavlja drugo verzijo svetovnega spleta [36] in vključuje bloge, socialne zaznamke, dnevnike, RSS, AJAX, mobilnost, CSS, standardizacijo, dostopnost, modularnost. Spletne aplikacije ponujajo interaktivno skupno rabo informacij, medopravilnost, prilagajanje vmesnika s strani uporabnika, sodelovanje na svetovnem spletu.
- **Poslovne aplikacije** so v primerjavi s spletnimi aplikacijami namenjene ožji množici uporabnikov z večjim poudarkom na funkcionalnosti, varnosti in z večjo naklonjenostjo standardiziranim in dobro podprtim tehnologijam.
- **Kompleksne poslovne aplikacije** predstavljajo informacijske sisteme raznih poslovnih sistemov. Največ poudarka se daje na arhitekturo, zaključenost in standardizacijo, včasih tudi za ceno enostavnosti in dostopnosti. Uporabljenje tehnologije morajo biti dobro standardizirane, široko zastavljene z dobro razvijalsko skupnostjo.

Java EE platforma pride do izraza pri kompleksnejših zahtevah s svojo splošno zasnovo, ki omogoča gradnjo zelo različnih rešitev. Dodatna argumenta sta tudi velika množica funkcionalnosti, ki jih platforma že vsebuje, ter odprtost arhitekture.

.NET platforma je podobno kot Java EE boljše okolje za kompleksne aplikacije kot LAMP platforma. Prav tako vsebuje ogromno funkcionalnosti, ki so primerljive z Java EE funkcionalnostmi. Slabost okolja je predvsem zaprtost in prilagojenost na ozek nabor tehnologij.

LAMP platforma lahko zaradi svoje ožje usmerjene zasnove privede do težav pri kompleksnejših problemih. Je pa zato primernejša izbira za običajne spletne aplikacije.

3.5.7 Aplikacijski strežniki

Privzeti aplikacijski strežnik za .NET platformo je Internet Information Services (krajše IIS), ki ga moramo namestiti na Windows operacijski sistem.

LAMP platforma v veliki večini uporablja Apache aplikacijski strežnik, ki je tudi daleč najbolj razširjen v svetu. PHP skripta se lahko izvaja tudi na Zend Server strežniku, ki ga razvija Zend podjetje.

Java platforma ima na voljo veliko aplikacijskih strežnikov, tako plačljivih kot brezplačnih. Najbolj razširjeni so: Apache Tomcat, JBoss, WebSphere, GlassFish in še mnogi drugi.

3.5.8 Računalništvo v oblaku

Ponudniki računalništva v oblaku nudijo različne platforme kot storitev. Uporabniki lahko nameščajo aplikacije v oblačno infrastrukturo brez upravljanja omrežja, strežnika, operacijskega sistema. Razlika med ponudniki je v omejitvi programskih jezikov in orodij. Spodaj so povzete storitve najpopularnejših PaaS ponudnikov.

Windows Azure

Microsoft Azure je najnovejši produkt v kategoriji storitev računalništva v oblaku s strani Microsofta. Omogoča uporabo podatkovne zbirke (SQL Azure) in razvojne platforme kot storitve. Podprti so vsi razvojni jeziki, ki jih omogoča .NET platforma ter PHP, Ruby, Python in Java jeziki. Platforma skrbi za avtomatizirano vzdrževanje infrastrukture, ki skrbi za visoko dosegljivost posamezne storitve. Microsoft že ponuja kar nekaj aplikacij, ki tečejo v oblaku in so na razpolago uporabnikom za povezovanje v njihove sisteme (na primer Windows Live, Microsoft Dynamics, Exchange Online, SharePoint Online). Poleg teh gradnikov platforma podpira številne standardizirane protokole, kot so: HTTP, REST, SOAP in XML. Oblačne aplikacije lahko ustvarjamo v Visual Studio, v prihodnosti pa namerava Microsoft podpreti razvoj v več razvijalskih okoljih.

Google App Engine

Google je leta 2009 za razvoj lastnih rešitev na Google infrastrukturi predstavil Google App Engine (krajše GAE) z razvojnim orodjem (ang. Software Development Kit – SDK). GAE je na razpolago na infrastrukturi, ki jo uporablja Google za lastne spletne aplilkacije (gmail, GoogleDocs ...). GAE zagotavlja celovit razvoj cikla programske opreme, še posebej učinkovito je gostovanje, vzdrževanje in nadgrajevanje, prav tako je omogočeno enostavno povezovanje z obstoječimi Google Apps. Zaenkrat so omejitve GAE:

- kot programski jeziki so podprti Java in jeziki, ki jih lahko prevedemo na JVM (JavaScript, Ruby) ter prilagojeni Python z Django ogrodjem,
- dostop do datotečnega sistema GAE je samo za branje,
- omejena izbira javanskih razredov,
- javanskim aplikacijam ni omogočeno proženje novih niti in
- aplikacije niso prenosljive na druge izvajalne platforme (ob uporabi BigTable podatkovne zbirke).

Salesforce.com

Salesforce.com je podjetje, ki ima visoke dosežke na področju avtomatizacije poslovnih procesov. Za razvijalce nudijo Force.com platformo, ki omogoča razvijalcem kreiranje aplikacij, ki se povezujejo z osnovno salesforce.com aplikacijo in gostujejo na salesforce.com infrastrukturi. V primerjavi z ostalimi ponudniki je platforma bolj zaprta. Aplikacije so zgrajene s posebnim, java podobnemu jeziku, imenovanemu Apex. Za uporabniški vmesnik v HTML, AJAX ali Flex-u pa se uporablja Visualforce jezik, ki je podoben xml jeziku.

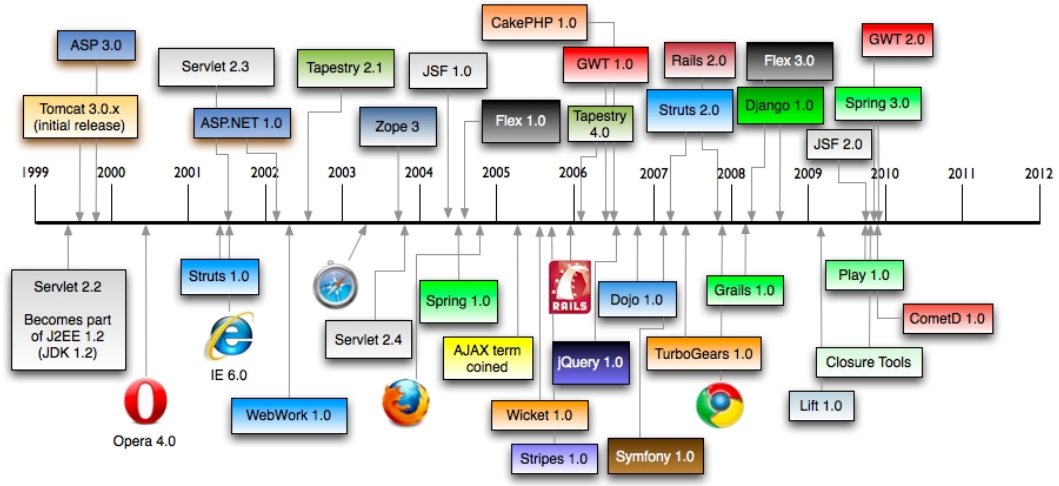
Poglavje 4

Ogrodja in modeli programiranja

Čeprav Java EE in .NET vsebujeta infrastrukturo, ki je potrebna za gradnjo zanesljivih, obsežnejših spletnih aplikacij, pade odgovornost za učinkovito uporabo tehnologije na posameznega razvijalca. Veliko spletnih aplikacij je zgrajenih namensko, brez upoštevanja inženirskih principov in brez globoke zveze med vsebino, predstavitvijo in vedenjem z nejasno strukturo. Da bi se to spremenilo, se je povečal poudarek na oblikovanju metod, ki izkoriščajo tehnologije na discipliniran način in poenostavijo proces razvoja spletnih strani. Rezultat je naraščujoče število razvijalnih ogrodij za jezike Java, Perl, PHP, Python in Ruby, medtem ko za .NET arhitekturo obstajata ASP.NET in ASP.NET MVC ogrodji. Razvijalne skupine lahko razvijejo svoje ogrodje, kar je časovno potratno in drago, ali pa uporabijo eno izmed mnogih, ki jim najbolj ustreza. Na sliki 4.1 je prikazana časovnica zgodovine priljubljenih spletnih ogrodij. Zaradi velike izbire so se izoblikovale različne metode primerjalnih analiz [21]. V tem delu bomo skušali navesti najbolj razširjena ogrodja in prikazati njihove prednosti in slabosti.

4.1 Značilnosti ogrodij

Ogrodja so prožne množice modulov, ki podpirajo hitro razvijanje ponavljajočih se zahtev na zanesljiv način z upoštevanjem uveljavljenih programerskih vzorcev. Cilj je, da se razvijalci osredotočijo le na domenske probleme, brez nizkonivojskih implementacijskih podrobnosti. Ogrodja lahko primerjamo glede na kvaliteto dokumentacije, razširljivost, modularnost, zrelost, aktivnost ra-

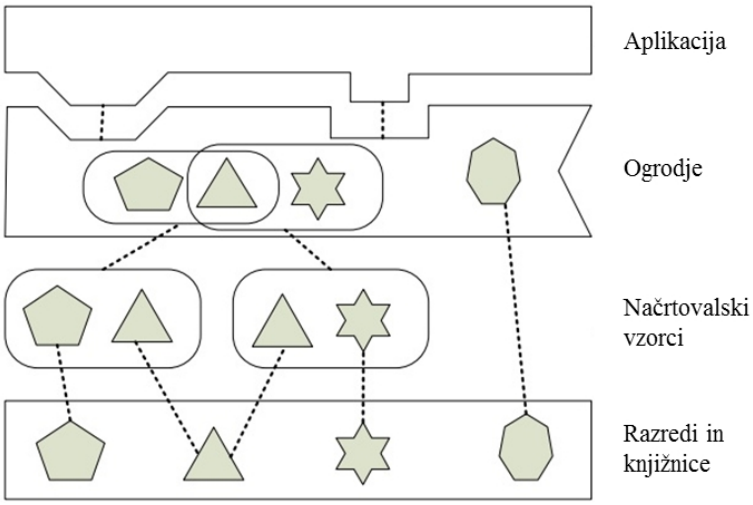


Slika 4.1: Časovnica spletnih ogrodij [39].

zvoja. Ogrodja se razlikujejo od preostalih vrst ponovne uporabe v programskem inženirstvu (programske komponente, knjižnice, načrtovalski vzorci), saj težijo k ponovni uporabi večjih delov programske kode in zasnove na višjem nivoju (na sliki 4.2) [26]. V nasprotju z drugimi tehnikami ponovne uporabe programske logike definirajo ogrodja tok izvajanja in zato delujejo kot osnova za na njih temelječih aplikacij. Ogrodja spadajo med učinkovitejše tehnike ponovne uporabe, saj enkapsulirajo še znanje načrtovanja.

Razvijalci in vodje projektov se za razvoj na osnovi ogrodij odločajo predvsem zaradi: (1) minimiziranja obsega implementacije, ki je potrebna za razvoj aplikacij, in (2) lažjega obvladovanja znanja domene, v kateri organizacija razvija aplikacije. Prednosti ogrodij so naslednje [13]:

- hitrejši in učinkovitejši razvoj;
  Aplikacije ne gradimo od začetka, ampak programsko kodo oziroma storitve, ki jih zagotavlja ogrodje, ponovno uporabimo. Ker aplikacije, ki temeljijo na enakem ogrodju, gradimo po ustaljenem vzorcu, se učinkovitost razvoja še dodatno poveča.
- boljša kakovost programske opreme;
  Izvorna koda ogrodij običajno temelji na preizkušenih programskih vzorcih in je zaradi večkratne uporabe izpostavljena obsežnim testom.
- Ogrodja omogočajo ponovno uporabo izvorne kode in načrtovanja.
- Omogočajo preusmerjanje fokusa s področja sistemskih problemov na



Slika 4.2: Ogrodja in druge tehnike ponovne uporabe.

področje domene. Sistemski problemi so rešeni v ogrodjih, zato se razvijalcem aplikacij z njimi ni potrebno ukvarjati.

- Zagotavljajo visoko stopno medizvedljivosti (ang. interoperability). Aplikacije, ki temeljijo na enakem ogrodju, so arhitekturno sorodne. Skupaj s podporo uveljavljenih standardov imajo na ogrodju temelječe aplikacije zagotovljeno visoko stopnjo medsebojne izvedljivosti.

Poznamo več vrst ogrodij, ki se razlikujejo po svoji zasnovi, obsežnosti in namenu. Najpogosteje se delijo na:

- domenska ogrodja (ang. domain framework) - naslavljajo določene problemske domene (na primer: računovodstvo, upravljanje virov),
- podporna ogrodja (ang. utility framework) - naslavljajo določene programske domene (na primer: uporabniški vmesnik, testiranje kode),
- aplikacijska ogrodja (ang. application framework) - obsežna ogrodja, ki so uporabna za različne problemske domene in naslavljajo številne programske domene.

Izbira ogrodja je ena večjih odločitev, ki jih naredijo razvijalci, ter vpliva na veliko ostalih odločitev. Pri izbiri je potrebno biti pozoren na:

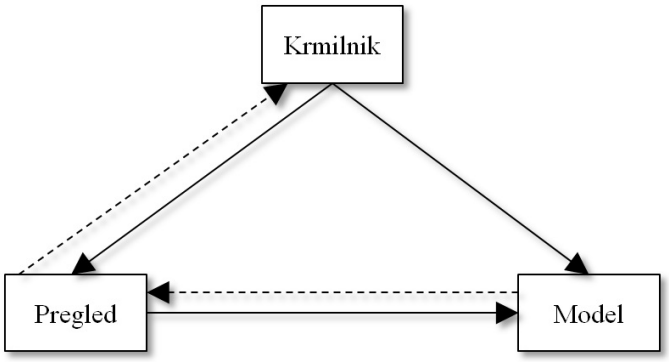
- dokumentacija mora biti obsežna in točna; ta je neprecenljivega pomena ob poskusih implementacije izbrane tehnologije;

- ogrodje mora ustrezati razvojnim metodam razvijalcev;
- kakšni so odzivi in izkušnje uporabnikov ogrodij in programskih jezikov;
- večja, kot je skupnost, večja je možnost nadaljnega razvoja in izboljšav; prav tako je lažje poiskati odgovore na morebitna vprašanja.

4.1.1 Model-View-Controller

Model-pregled-krmilnik (ang. Model-view-controller – MVC) razvijalski vzorec se običajno uporablja za programiranje uporabniških vmesnikov z namenom ločitve predstavitve od poslovne logike in kontrole. Kot takšen je logična arhitekturna izbira za spletni razvoj, saj ustreza dogodkovno vodeni naravi spletnih aplikacij. MVC arhitektura deli aplikacije na tri sloje: model, pregled in krmilnik (kot prikazuje slika 4.3). Vsaka plast skrbi za določene naloge in ima določene odgovornosti do drugih plasti.

- Model predstavlja poslovne podatke in poslovno logiko oziroma operacije za dostop in spreminjanje teh podatkov.
- Pregled prikaže vsebino modela.
- Krmilnik definira, kako se aplikacija vede.



Slika 4.3: Model-pregled-krmilnik koncept. Polna črta predstavlja direktno, prekinjena pa posredno povezavo.

Začetki spletnega razvoja niso dovoljevali razplastitve, zadnje različice tehnologij pa podpirajo modularnost, ki se poravnava z MVC zasnovo. Večjo vlogo podpori MVC modela v ogrodjih se je začelo poudarjati po letu 2000, ko se je videla prednost v ponovni uporabi ter boljšem razvojnem procesu.

Prvotno so bila ogrodja na voljo za Java EE, nato so nastajala tudi za .NET platformo ter še za skriptne jezike. Danes obstaja veliko konkurenčnih odprtokodnih ogrodij.

Obstaja tudi Model-Pogled-Prezentator (ang. Model-View-Presenter – MVP) model, ki se razlikuje od MVC modela v načinu, kako se podatki prenašajo iz modela nazaj v pogled. V MVC-ju gredo iz modela v pregled, medtem ko v MVP gredo nazaj preko prezentatorja.

4.2 Strežniška spletna ogrodja

Množica strežniških spletnih ogrodij je zelo velika. Z nekaj truda lahko najdemo veliko primerjav, ki skušajo ovrednotiti lastnosti in prikazati, katera ogrodja so v določenih situacijah primernejša.

Kai Waehner [22] je v svojem spletnem dnevniku opredelil spletne apolikacije ter java ogrodja glede na kompleksnost in interaktivnost uporabniškega vmesnika. Na wikipediji [3] je splošna primerjava velike množice strežniških ogrodij večih jezikov. Primerjava vsebuje kriterije, kot so: AJAX podpora, MVC, ORM, varnost, predpomnjenje (ang. caching), testiranje itd. Riyad Kalla [23] je skušal z uporabo Google Trends napovedati, katera ogrodja so najpopularnejša. Jaspal [24] je sestavil pregled najpopularnejših ogrodij za spletni razvoj s splošnimi opisi. Na spletni strani bestwebframeworks.com so predstavljena vsa popularna ogrodja z osnovno primerjavo in opisi. Spletna stran hotframeworks.com pa skušajo iz statistik spletnih brskalnikov izmeriti popularnost najpriljubljenejših spletnih ogrodij. Matt Raible [25] je na podlagi svojih izbušenj izdelal primerjavo java ogrodij v razpredelnici. Primerjava vsebuje 13 ogrodij, katere ocenjuje glede na 20 kriterijev. Izbiramo med ocenami 0, 0.5 in 1. Kriterije lahko poljubno utežimo in ugotovimo katera ogrodja bi naj bolje ustrezala. Java skupnost je naredila anketo glede uporabe Java EE tehnologij [14]. Vprašanja so se nanašala na Java EE standarde, Java ogrodja, IDE ipd.

V tem poglavju bomo navedli najpopularnejša ogrodja, ki so bila poudarjena v raznih primerjavah, in skušali navesti njihove prednosti in slabosti.

4.2.1 Seam

JBoos Seam (krajše Seam) je aplikacijsko ogrodje, ki ga je razvila JBoss divizija Red Hat. Avtor zamisli je Gavin King, ki je bil prav tako ključni pri razvoju ORM ogrodja Hibernate. Seam je ogrodje, ki sloni na Java EE platformi.

Kombinira dve ključni tehnologiji Enterprise JavaBean 3.0 (krajše EJB3.0) in Java ServerFages (krajše JSF). Združuje tudi druge komponente modela in ogrodja, kot so AJAX, jBPM, Drools, Hibernate. Ogrodje torej povezuje množico programskih vmesnikov in servisov o okolju, ki olajša razvoj Java EE spletnih aplikacij. Osnovni načeli sta:

- poenostavljanje Java EE: ponuja množico bližnjic in poenostavitev standardnega Java EE okolja; olajša učinkovito uporabo Java EE komponent;
- razširja Java EE: povezuje množico novih konceptov in orodij v Java EE, ki prinašajo nove funkcionalnosti.

4.2.2 Grails

Grails je odprtokodno spletno ogrodje, ki uporablja Groovy programski jezik. Groovy je objektno-usmerjen programski jezik za java platformo. Je dinamičen in ima podobne značilnosti kot Python, Ruby, Perl in Smalltalk jeziki. Dinamično se prevaja v bitno kodo za javanski navidezni stroj. Večina Java kode je sintaktično veljavna tudi za Groovy jezik. Ogrodje je osnovano kot visoko produktivno ogrodje s sledenjem paradigme kodiranja po konvencijah (ang. coding by convention), ki je samozadostno razvijalno okolje s skritimi nastavitvenimi podrobnostmi pred razvijalcem. Grails skuša v javanski svet pripeljati prednosti Ruby on Rails ogrodja, od tod tudi ime. Od leta 2008 za razvoj skrbi SpringSource. Ogrodje je bilo razvito za:

- zagotavljanje visoke produktivnosti za Java platformo;
- ponovno uporabo dokazanih Java tehnologij, kot so Hibernate in Spring na enostaven, dosleden način;
- zmanjšanje zmede in lažje učenje ogrodja;
- dobro dokumentacijo;
- nudenje funkcionalnosti, ki jih uporabniki potrebujejo na področjih, ki so pogosto kompleksna in nedosledna:
  - močna ORM podpora,
  - močne in enostavne predloge pogledov s Groovx Server Pages (krajše GSP),
  - dinamični priveski (ang. tag) knjižnic za lažjo gradnjo konpoment strani,
  - dobra enostavna AJAX podpora z možnostjo razširitve.
- primere aplikacij, ki kažejo moč ogrodja.

4.2.3 Struts 2

Ogrodje predstavlja zelo prožen in eleganten način za razvoj javanskih spletnih aplikacij za podjetja vseh velikosti in ni del platforme Java EE. Izdelano je bilo, da bi razvijalcem olajšalo razvoj kvalitetnih zahtevnih spletnih aplikacij, ki temeljijo na servletih, straneh JSP in lastnih oznakah. Gre za odprtokodno ogrodje, ki temelji na vzorcu model-pogled-krmilnik. Glavne pridobitve Struts 2 so:

- zmanjšana kompleksnost – ni potrebno graditi lastnega MVC ogrodja,
- spodbuja dobre prakse (vzorce),
- enostavna uporaba,
- ponuja veliko funkcij,
- vključuje celo paleto tretjih aplikacij,
- razširljivost,
- odprtokodnost,
- dobra povezljivost z Java EE.

4.2.4 Spring

Spring Framework (krajše Spring) je odprtokodno ogrodje z večplastno arhitekturo za Java platformo. Nastalo je leta 2002. Prvo verzijo je napisal Rod Johnson skupaj s knjigo „Expert Ono-on-One J2EE Design and Development“. Prvo izdajo je ogrodje doživelo 2003. Osnovne funkcionalnosti Spring ogrodja se lahko uporabijo v katerikoli Java aplikaciji, določeni dodatki pa so namenjeni gradnji spletnih aplikacij na Java EE platformi. Trenutno je eno najpogosteje uporabljenih javanskih ogrodij [14]. Čeprav ogrodje ne vsiljuje specifičnega programskega modela, je kmalu postalo zelo priljubljeno znotraj java skupnosti kot alternativa EJB modelu. Vsebuje ogromno modelov, ki jih lahko uporabljamo po želji in sprotno vključujemo v projekt, kot so:

- Inversion of Control container – upravljanje z Java objekti preko povratnih klicev. Skrbi za kreiranje objektov, klicanje inicializacijskih metod ter nastavljanje objektov z medsebojnim povezovanjem (ang. Injecting Dependencies – ID). Modul nastavljamo preko xml datotek, ki vsebujejo Bean definicije, ki so potrebne za kreiranjeJava objektov(Beanov);

- Aspect-oriented programming – lastna implementacija aspektno usmerjenega programiranja, ki poveča modularnost z razdelitvijo "navkrižnih skrbi";
- Model View Controler – implementacija MVC modela;
- Remote access framework – abstrakcija dela z RPC (Remote Procedure Call) tehnologijami na Java platformi za povezavo s odjemalcem in izvozom objektov na strežnik;
- Convention over configuration – Spring Roo je rešitev za hitro razvijanje aplikacij v Javi preko podajanja ukazov v ukazni lupini;
- Batch processing - funkcionalnosti, ki omogočajo logiranje/sledenje, upravljanje s transakcijami, statistika izvajanja zaposlitev (ang. job), ponovni zagon zaposlitev, upravljanje z viri itd. pri procesiranju večje količine podatkov;
- Authentication and authorization – nastavljanje varnostnih procesov, ki podpirajo razne standarde, protokole, orodja in prakse preko Spring Security projekta (prej znano kot Acegi Security System);
- Remote Management – lokalno ali oddaljeno upravljanje z objekti;
- Messaging – podpora sporočanju;
- Testing – močna podpora pisanja enotskih in integracijskih testov.

Spring ogrodje je pod okriljem SpringSource skupnosti, ki je del VMware podjetja. SpringSource se poleg ogrodja ukvarja še z več projekti, ki nudijo različne rešitve spletnim razvijalcen. Npr.:

- Spring IDE je ena izmed najbolj priljubljenih dodatkov za razvijalsko okolje Eclipse [12]. Dodatek je prilagojen za razvijanje aplikacij v Spring ogrodju in lajša nastavljanje nastavitvenih datotek itd.
- Spring Security je zmogljivo in prilagodljivo ogrodje za avtentikacijo in avtorizacijo.
- Sprint Web Flow je razširitev Spring frameworka, ki je namenjeno upravljanju dialogov med strežnikom in odjemalcem.
- Spring Web Services je namenjen gradnji prilagodljivih spletnih storitev.
- Spring BeanDoc – generiranje dokumentacije, podobno Java doc.
- Spring Data nudi podporo nerelacijskim bazam in oblačnim storitvam.

- Spring BlazeDZ Integration podpira Flex tehnologijo prikaz za Spring aplikacije.
- Spring .NET nudi Spring podporo za .NET aplikacije.

4.2.5 Apache Wicket

Apache Wicket (krajše Wicket) je komponentno osnovano spletno ogrodje za programski jezik Java, konceptualno podobno JavaServer Faces in Tapestry ogrodjem. Prva verzija je bila izdana leta 2005. Za razliko od mnogih MVC ogrodjih je programiranje grafičnega uporabniškega vmesnika bolj podobno programiranju namiznih aplikacij. Wicket aplikacijo lahko obravnavamo kot drevo komponent, ki imajo implementirane poslušalce, preko katerih lahko odreagiramo na HTTP zahteve. Oblikovanje poteka v XHTML jeziku, kar nudi ločitev predstavitvene in poslovne logike ter možnost oblikovanja z naprednimi urejevalniki. Vsaka komponenta je navezana na poimenovan element v XHTML ter odgovorna za prikaz tega elementa. Spletna stran je enostavno najvišja komponenta, ki vsebuje vse ostale. Takšen način omogoča lažjo ponovno uporabo kontrol ter dovoljuje razvijalcem več svobode.

4.2.6 CakePHP

Ogrodje CakePHP omogoča razvoj, vzdrževanje in uvajanje razširljivih spletnih aplikacij. Uporablja priznane vzorce kot sta MVC in ORM. Razvijalcem spletnih aplikacij omogoča zmanjšanje stroškov razvoja, skrajša čas razvoja in zmanjša potrebno število vrstic kode. Glavne prednosti ogrodja CakePHP so:

- ne potrebuje konfiguracij,
- preprosta uporaba,
- aktivna skupnost,
- uporaba dobrih praks programiranja,
- objektna orientiranost in
- dobra dokumentacija.

4.2.7 Zend

Zend Framework (krajše Zend) je odprtokodno, objektno orientirano spletno ogrodje, implementirano v PHP jeziku, verzije 5. Ogrodje je bilo javno

predstavljeno leta 2005 ter je hitro postalo popularno v razvijalski skupnosti. Značilnosti ogrodja so:

- vse komponente so napisane v PHP 5;
- funkcionalnosti, ki so na voljo, lahko uporabljamo po želji z minimalno medsebojno odvisnostjo;
- prožna MVC implementacija;
- podpora za številne podatkovne baze, kot so: MySQL, Oracle, Microsoft SQL Server, PostgreeSQL, SQLite, IBM DB2 in Informix Dynamic Server;
- sestavljanje in pošiljanje elektronskih sporočil;
- prilagodljiv sistem predpomnjenja (ang. caching).

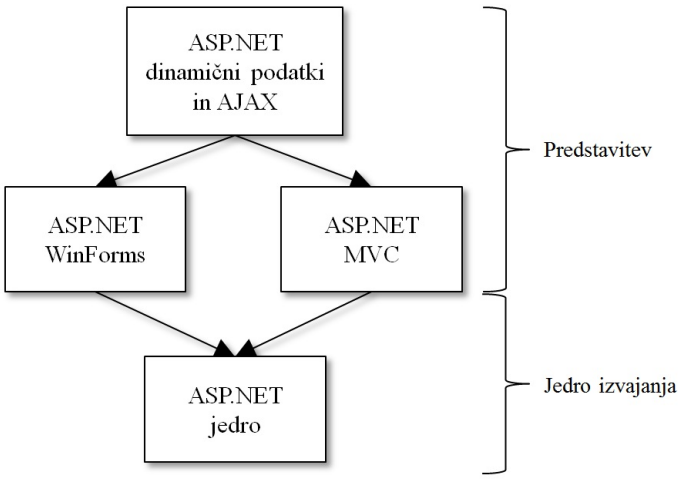
4.2.8 ASP.NET, ASP.NET MVC

ASP.NET WinForms (krajše ASP.NET) je naslednik ASP tehnologije, zgrajen na Common Language Runtime (CLR), kar omogoča razvijalcem pisanje ASP.NET kode v enem izmed .NET jezikov. Tehnologija ni nadgradnja starejše tehnologije, ampak zajema nove, inovativne gradnike za spletno programiranje. ASP.NET tako omogoča razvoj dinamičnih, interaktivnih spletnih aplikacij in spletnih storitev. Ogrodje vsebuje vrsto pripravljenih storitev in kontrol, ki pospešijo in olajšajo razvoj projekta. ASP.NET je nekaj let mlajši kot veliki tekmec Java EE. Za razliko od večine spletnih okolij je dogodkovno voden, kar je sicer značilno za namizne aplikacije. To v navezavi z močnim razvijalskim orodjem Visual Studio omogoča razvijanje spletnih strani enostavno za razvijalce z minimalnim znanjem. Vendar ta abstrakcija osnovni HTTP protokol, ki ne ohranja stanja, predstavlja kot protokol, ki beleži stanje. Posledica so kompleksni cikli zahtevkov, večje količine prenesenih podatkov in težje razumevanje tehnologije.

Velika prednost ASP.NET okolja so zelo zmogljive strežniške kontrole za prikaz podatkov, validacijo, navigacijo, AJAX podporo itd. Kontrole vsebujejo poslušalce, ki se ob akciji sprožijo in izvedejo na strežniku. Vsaka stran vsebuje skrito polje Viewstate, s katerim se sinhronizira stanje aplikacije na strežnika in odjemalcu. Izbiramo lahko tudi med veliko tako brezplačnih kot plačljivih kontrol, razvitih v .NET skupnosti.

ASP.NET MVC je spletno ogrodje, ki implemetira MVC model. Osnovano je na ASP.NET, vendar nima njegovih slabih točk. Na sliki 4.4 je prikazana

arhitekturna delitev ASP.NET in ASP.NET MVC. V primerjavi z ASP.NET se bolj zgleduje po ostalih odprtokodnih ogrodjih.



Slika 4.4: Skupne lastnosti ASP.NET in ASP.NET MVC ogrodij.

Razlike v primerjavi z ASP.NET so:

- boljša ločitev predstavitvene in poslovne logike;
- večja kontrola nad generirano HTML kodo, ASP.NET tehnologija skriva večji del generiranja spletnih strani pred razvijalcem strani ne vsebujejo Viewstate polj;
- lažje programiranje na odjemalčevi strani z JavaScript, pri ASP.NET imajo kontrole na strani kompleksnejša imena;
- lažje testiranje, ker je upravljanje naloga kontrolerja in ne vsake posamezne kontrole, je testiranje bolj obvladljivo;
- lažja optimizacija za spletne iskalnike;
- večja kontrola interakcije med strežnikom in odjemalcem; zaradi sinhronizacije stanja med strežnikom in odjemalcem ima pri ASP.NET razvijalec manj vpogleda v interakcijo;
- spletne kontrole niso tako zmogljive kot pri ASP.NET;
- lahko uporabljamo več prikazovalnih pogonov (ang. view engine).

Prva verzija je bila izdana leta 2009, tako da je še razmeroma nova tehnologije.

4.2.9 Ruby on Rails

Ruby on Rails (krajše RoR ali Rails) je odprtokodno ogrodje za spletne razvijalce v programskem jeziku Ruby. Namenjeno je za uporabo Agilnih metod razvijanja, ki jih uporabljajo spletni razvijalci za hitro razvijanje (ang. rapid development). Ustvaril ga je David Heinemeier Hansson med delom na Basecamp aplikaciji za podjetje 37signals. Javnosti je bilo ogrodje prvič dostopno leta 2004. Podobno kot ostala ogrodja tudi RoR uporablja MVC arhitekturo.

4.2.10 Django

Django je nastal kot zaprto ogrodje znotraj časopisne hiše. V nekaj letih so razvili množico knjižnic, ki se je odlično izkazala. S poenostavljanjem in avtomatizacijo splošnih nalog spletnega razvoja so lahko svoje delo opravili hitreje in učinkoviteje. Leta 2005 so ogrodje dali na voljo vsem. Django se zaradi svoje časopisne dediščine predstavlja kot „Spletno ogrodje za perfekcioniste s časovnimi roki“. Jedro ogrodja je sestavljeno iz kompaktnih, dobro testiranih knjižnic, ki pokrivajo ponavljajoče se procese spletnega programiranja:

- objektno-relacijsko povezovanje (ang. object-relational mapping – ORM);
- HTTP knjižnice za razčlenjevanje prihajajočih zahtevkov, njihovo upravljanje in generiranje odgovorov;
- usmerjanje URL-jev na določeno python kodo;
- validacijska knjižnica za prikazovanje form in procesiranje poslanih podatkov;
- sistem za generiranje HTML kode z generiranimi podatki;
- administrativni vmesniki;
- avtentikacija uporabnikov.

4.3 Ogrodja za obogatene spletne aplikacije

Ogrodja za obogatene spletne aplikacije so usmerjena predvsem v uporabniški vmesnik in izvajanje na odjemalcu. Čeprav so obravnavana ogrodja zelo različna, si delijo skupne arhitekturne lastnosti. Posplošeno so njihovi cilji [6]:

- zmanjšanje kompleksnosti razvoja AJAX aplikacij, ki je utrudljivo, težavno in z veliko možnostjo napak;

- skriti nekompatibilnost različnih spletnih brskalnikov in platform;
- poenostaviti komunikacijo med strežnikom in odjemalcem;
- doseči bogato interaktivnost in prenosljivost za končne uporabnike in lažji proces razvijanja za razvijalce.

4.3.1 GWT

Google Web Toolkit ni definiran kot ogrodje, temveč množica orodij, ki omogočajo spletnemu razvijalcu kreiranju kompleksne JavaScript aplikacije v Javi. Razvilo ga je podjetje Google za gradnjo performančno optimalnih RIA aplikacij. Uporabniški vmesnik lahko sestavimo s pomočjo knjižnice kontrol, katere GWT prevede v JavaScript kodo, ki se izvaja v brskalniku. Komunikacija s strežnikom je potrebna le, če potrebujemo podatke za kontrole na odjemalcu. Za to se izvedejo klici na definirane storitve na strežniku, ki so implementirane v Javi. Podatki se prenašajo v obeh smereh z uporabo serializacije. Zadnja različica je prinesla MVP model programiranja, kontrole za predstavitev podatkov, združitev s SpringSource orodji itd. Uporabimo lahko tudi rešitve različnih projektov obsežne skupnosti, ki razširjajo GWT funkcionalnosti. Ker je tehnologija razmeroma nova, nimamo na voljo toliko oziroma tako močne kontrole kot v nekaterih drugih ogrodjih, vendar GWT omogoča enostavno vključitev DHTML kontrol v GWT aplikacijo.

Kot že omenjeno, GWT ni ogrodje, saj je zelo ozko usmerjena rešitev. Napisan je nizkonivojsko in je zelo optimiziran. Za razvoj kompleksnih spletnih aplikacij ga lahko uporabljamo v kombinaciji s strežniškim ogrodjem. To mora podpirati RPC komunikacijo s odjemalcem, da se lahko vzpostavi komunikacija strežnika z GWT aplikacijo na odjemalcu. Najlažje je GWT uporabljati s katerim izmed Java ogrodji, saj v tem primeru celoten razvoj poteka v enakem jeziku. Google in SpringSource sta na primer dosegla dogovor o povezovanju tehnologij [17]. Obstaja tudi možnost povezave s PHP ali katerim drugim jezikom, a se v praksi to verjetno ne bo izkazalo za učinkovito, saj moramo nato aplikacijo razvijati v dveh programskih jezikih.

4.3.2 FLEX

Flash je bil predstavljen leta 1996 v nekdanjem podjetju Macromedia (sedaj Adobe Systems). Gre za multimedijsko platformo, ki spletnim stranem dodaja animacije, video vsebino in večjo interaktivnost. Flash vsebina je lahko prikazana na različnih računalniških sistemih in napravah s pomočjo Adobe Flash

predvajalnika[1], ki je na voljo brezplačno za vse razširjene brskalnike ter nekaj mobilnih naprav. Najbolj se je uveljavil na področju predstavljanja video vsebin in animacij.

Adobe FLEX (krajše FLEX) je odprtokodno ogrodje, namenjeno poenostavitvi razvoja bogatih spletnih aplikacij v flash okolju. Zagotavlja konstantno hitrost delovanja in zunanje grafične podobe na večini danes najbolj razširjenih spletnih brskalnikih in operacijskih sistemih. Zagotavlja tudi moderen in standarden programski model, ki podpira najpogostejše načrtovalske vzorce ter vsebuje tudi razvojno okolje Flex Builder. Slednji temelji na dobro poznanem Eclipse IDE orodju. Flex aplikacije se lahko namesto flash predvajalnika izvajajo tudi v posebnem izvajalnem okolju Apollo Integrated Runtime (krajše AIR), ki omogoča izvajanje Flex aplikacij enako namiznim aplikacijam. Potrebujemo le nameščeno AIR izvajalno okolje na osebnem računalniku. Ogrodje ne omogoča direktne povezave do podatkovne baze, ampak uporablja spletni strežnik za pridobivanje ustreznih podatkov. Lahko komunicira s strežniki, na katerih delujejo aplikacije JSP, ASP, ASP.NET, PHP, ColdFusion itd. Za postavitev uporabniškega vmesnika uporablja Flex označevalni jezik MXML, ki temelji na XML jeziku ter ActionScript kot skriptnem jeziku. Tehnologija je bila primarno zamišljena za animacije in video vsebine [19]. Kot posledica jo večinoma uporabljajo oblikovalci, redko pa se uporabi pri razvoju poslovnih aplikacij.

4.3.3 Silverlight

Prva verzija Silverlight je izšla leta 2007 ter nemudoma dvignila veliko prahu. Silverlight aplikacija se izvaja v brskalniku na odjemalcu v specifičnem CLR okolju.

Silverlight uporablja .NET od verzije 2 naprej. Vsebuje okleščeno verzijo osnovnih knjižnic, ki je zelo impresivno razširjena glede na velikost vtičnika (manj kot 5 MB). Zaradi podobnosti lahko veliko znanja iz programiranja aplikacij v polnem .NET ogrodju uporabimo tudi pri razvijanju Silverlight aplikacij.

Silverlight je omejena verzija namizne tehnologije Windows Presentation Foundation (krajše WPF). Določene funkcionalnosti so vključene v WPF in jih ni v Silverlightu ter obratno. Kodo, ki je bila napisana v eni tehnologiji, lahko ponovno uporabimo tudi v drugi, vendar je potrebno nekaj predhodnega načrtovanja, da je prehod bolj tekoč. Microsoft je izdal dokumentacijo, v kateri

[1] vtičnik za brskalnik.

je pojasnjeno, kakšne so razlike med obema tehnologijama [9], da ju lahko lažje povezujemo.

Z verzijo 2 je Microsoft jasno napovedal da je Silverlight namenjen tako za bogate in interaktivne aplikacije, kot tudi za napredne poslovne aplikacije v brskalniku. Slednje se lahko opazi glede na dodajanje novih kontrol, podpori za več tipov storitev in platformnih funkcionalnosti, kot je podatkovno navezovanje (ang. data binding).

Zaradi delovanja v brskalniku Silverlight aplikacije, podobno kot pri flashu, za pridobivanje podatkov potrebujejo spletne storitev. Silverlight 3 je prinesel nekaj zanimivih rešitev na tem področju s podporo binarnega XML jezika, WCF RIA storitev in poenostavljeno dvojno (ang. duplex) komunikacijo s storitvami. Silverlight 4 je nadaljevanje v tej smeri [20] z izboljšavami v podatkovnem povezovanju (ang. data binding), podpori za net.tcp komunikacijo, navkrižnemu (ang. cross-domain) dostopu do storitev preko zaupanja vrednih aplikacij itd. Vse te dodane funkcionalnosti so dokaz, da ima Microsoft obvezo predstaviti Silverlight kot platformo za gradnjo spletnih poslovnih aplikacij. Glede na odzive uporabnikov jim to tudi uspeva [18].

Da daje Microsoft večji poudarek Silvelightu kot ASP.NET in ASP.NET MVC tehnologiji, lahko sklepamo tudi po MIX konferenci [10]. Na znanem dokodku, kjer se srečajo inovativni spletni razvijalci, je bilo 43 sej na temo Silverlight tehnologije ter le 9 na temo ASP.NET in ASP.NET MVC.

4.3.4 Echo

Echo je spletno ogrodje, ki ga je razvilo podjetje NextApp. Razvijalcem omogoča gradnjo spletnih aplikacij z uporabo objektno orientiranega, komponentno usmerjenaga in dogodkovno vodenega programiranja. Vsebuje Java programski vmesnik, ki nudi kontrole za uporabniški vmesnik, objekte z lastnostmi in dogodke oziroma poslušalce za predstavitev in upravljanje stanja aplikacije in njenega uporabniškega vmesnika. Vsa funkcionalnost za prikaz komponent je zapisana v posebnem modulu Web Rendering Engine (krajše WRE). Pogon sestavlja strežniški del (napisan v Java/Java EE) in odjemalčev del (JavaScript). Interakcija med strežnikom in odjemalcem je skrita znotraj tega WRE modula ter povsem ločena od ostalih modulov. Echo2 vsebuje Update Manager, ki je odgovoren za sledenje spremembam na uporabniškem vmesniku, procesiranju vnešenih podatkov in komuniciranju s komponentami. Echo2 Client Engine teče na odjemalcu in omogoča oddaljeni uporabniški vmesnik za strežniško aplikacijo. Njegova glavna aktivnost je sinhronizacija stanja aplikacije na strežniku in odjemalcu ob interakciji uporabnika z uporabniškim

vmesnikom. ClientMessage je sporočilo v XML formatu, s katerim se spremembe stanja uporabniškega vmesnika prenesejo na srežnik. Strežnik sporočilo razčleni in ustrezno odreagira s ServerMessage sporočilom v XML formatu, ki vsebuje ukaze za morebitne spremembe uporabniškega vmesnika. Zadnja verzija ogrodja je Echo3, kjer je za razliko od Echo2 podprto tudi programiranje v JavaScript jeziku na strani odjemalca.

4.3.5 Vaadin

Vaadin je odprtokodno ogrodje za gradnjo RIA aplikacij. Za razliko od JavaScript knjižnic in uporabe brskalnikovih vtičnikov je arhitektura zasnovana na strežniški strani, torej se večina programske logike izvaja na strežniku. Na odjemalcu je AJAX tehnologija, ki zagotavlja bogato in interaktivno uporabniško izkušnjo. Odjemalčeva stran je zgrajena na GWT tehnologiji. Najlepša značilnost Vaadina je, da kot edini programski jezik uporabljamo Java. Ogrodje uporablja dogodkovno vodeno programiranje in kontrole, ki omogočajo enostavnejšo gradnjo uporabniškega vmesnika. Medtem ko se pri GWTju vsa logika izvaja na odjemalcu, kar lahko privede do varnostnih težav, Vaadin vsebuje validacijo podatkov na strežniku. To zagotavlja, da so podatki pravilni in zaščiteni, vendar pa ta način močno poveča količino prometa med strežnikom in odjemalcem ter dodatno obremenjuje strežnik.

4.3.6 SproutCore

SproutCore je odprtokodno JavaScript ogrodje. Pojavil se je leta 2007 ter kmalu doživel večjo razpoznavo, ko ga je podjetje Apple uporabilo za svojo spletno aplikacijo Mobile.me namesto flash tehnologije, vendar je kljub temu trend SproutCore kmalu precej pade.

4.3.7 Cappuccino

Cappuccino je odprto-kodno razvojno ogrodje za razvijanje spletnih aplikacij, ki dajejo vtis in izgled namiznih aplikacij. Razvili so ga v podjetju 280 North. Sestavljen je iz programerskega jezika Objective-J in objektno usmerjene knjižnice, ki je pretvorba Cocoa ogrodja (API za Mac OSx) za Objective-J (na Mac OSx se uporablja Objective-C) ter še nekaj dodatnih knjižnic.

Prevajalnik za Objective-J je v celoti napisan v JavaScriptu, zato programi, napisani v Objective-J, ne potrebujejo nobenega prevajanja na strani strežnika, saj so direktno prevedeni na odjemalcu.

Cappuccino ni klasična JavaScript knjižnica (jQuery, ExtJs ...), saj se razvijalcu nikoli ni potrebno ubadati z DOMom ali oblikovati CSS zapise. Za te zadolžitve poskrbi ogrodje samo.

4.3.8 jQuery

jQuery je najbolj popularno JavaScript ogrodje za spletne razvijalce. Iz njega izhaja tudi jQuery UI, s katerim zgradimo uporabniški vmesnik in ga upravljamo.

4.3.9 Primerjava

V tabelah 4.1 in 4.2 je prikazana splošna primerjava omenjenih ogrodij za obogatene spletne aplikacije. Sledijo opredelitve kriterijev.

Kriterij	*GWT*	*Flash*	*Silverlight*	*Echo*
Jezik	Java	ActionScript	.NET jeziki	JavaScript
Podprtost brskalnikov	10	10	7	10
Format izvajanja	JavaScript	Adobe AIR, SQF	Silverlight vtičnik	JavaScript
Platforme	vse	vse	vse*	vse
Mobilne naprave	vse	Android, Symbian, WebOS	Windows Phone 7, Symbian	vse
Izvajanje	odjemalec	odjemalec	odjemalec	strežnik

Tabela 4.1: Prvi del primerjave ogrodij.

Kriterij	*Vaadin*	*SproutCore*	*Cappuccino*	*jQuery*
Jezik	Java	JavaScript	Objective-J	JavaScript
Podprtost brskalnikov	10	9	9	10
Format izvajanja	JavaScript	JavaScript	JavaScript	JavaScript
Platforme	vse	vse	vse	vse
Mobilne naprave	vse	vse	vse	vse
Izvajanje	strežnik	odjemalec	odjemalec	odjemalec

Tabela 4.2: Drugi del primerjave ogrodij.

Jezik

Kriterij določa, v katerem jeziku programiramo aplikacije v določenem ogrodju. Definiran je le primarni jezik. Ogrodja lahko zahtevajo poznavanje tudi drugih jezikov, na primer: HTML, CSS.

Podprtost brskalnikov

Primerjana ogrodja skušajo skriti nekompatibilnost različnih brskalnikov, zato je zelo pomembno, v koliko brskalnikov je določeno ogrodje podprto. Ocena 10 pomeni, da je rešitev podprta v praktično vseh brskalnikih, ocena 9, da ni podprt en popularen brskalnik ter 7, da ni podprtih več popularnih brskalnikov.

Format izvajanja

Kriterij določa, v kakšnem formatu se izvaja rešitev posameznega ogrodja na brskalniku. Če se ne izvaja v JavaScriptu, moramo imeti nameščen vtičnik, ki ustreza formatu.

Platforme

Vse rešitve so platformno neodvisne, razen Silverlight, ki je podprt le za Windows in Mac okolja. Za Linux okolja pa obstaja brezplačna implementacija Moonlight, ki jo je razvilo podjetje Novell v povezavi z Microsoft.

Mobilne naprave

Ta kriterij določa, na katerih mobilnih napravah je mogoče poganjati rešitve, razvite v posameznih ogrodjih. Predpostavljamo, da ima vsaka mobilna naprava brskalnik, ki je zmožen poganjanja JavaScript kode, zato so rešitve, katerih format izvajanja je JavaScript, podprte na vseh novejših mobilnih napravah.

Poglavje 5

Zaključek

Predstavili in kategorizirali smo temeljne tehnologije za razvoj spletnih aplikacij. Rezultati primerjav lahko služijo kot kažipot pri izbiri ustrezne tehnologije specifičnim aplikacijskim zahtevam. Pomembno je poudariti, da lahko na vsaki platformi in s praktično vsakim ogrodjem realiziramo večino razvojnih zahtev. Z ustrezno izbrano tehnologijo pa lahko minimiziramo stroške in čas razvoja ter maksimiziramo obvladljivost razvoja. Tehnični problemi v navezi spletne infrastrukture so v večini že rešeni. Visoko skalabilnost lahko dosežemo z delitvijo zahtev na več sistemov. Dober primer je Google, ki lahko streže približno 1000 zahtevkov na sekundo. Platformi Java in .NET platformi pa omogočata razvoj zanesljivih, razširljivih, prenosljivih in varnih poslovnih spletnih aplikacij. Če povzamemo, so osnovni kriteriji pri izbiri ustrezne tehnologije izhodišče, kompleksnost zahtev, časovna kritičnost, stopnja izvajanja na odjemalcu in dostopnost rešitve.

5.1 Izhodišče

Kriterij se nanaša na morebitne zahteve stranke glede uporabe tehnologij in izkušnje razvijalcev. Stranka že lahko ima postavljeno določeno infrastrukturo in programsko opremo, na kateri bi se naj izvajala zahtevana rešitev. To pogosto omejuje možnosti izbire tehnologij, zato se morajo razvijalci prilagoditi stranki. Do omejitev lahko pride tudi, če stranka ne želi uporabljati licenčne programske opreme (podatkovne baze, aplikacijski strežniki), ki bi predstavljale dodaten strošek.

Pomembno je tudi predznanje razvijalske skupine. Ta veliko lažje osvoji znanje na primer novega ogrodja na enaki platformi, kot če ob spoznavanju novega ogrodja spoznava tudi novo platformo. Učenje nove tehnologije in njena

učinkovita uporaba praviloma zahteva zelo veliko časa.

5.2 Kompleksnost zahtev

Kriterij predstavlja kompleksnost zahtev, ki jih razvijalci skušajo rešiti. Kot je razvidno iz primerjave platform v 2. poglavju, sta platformi .NET in Java bolj široko zastavljeni kot LAMP. Slednja pa je boljša izbira pri gradnji enostavnejših spletnih aplikacij. Tabela 5.1 vsebuje tehnologije, ki ustrezajo razredom kompleksnosti spletnih aplikacij, kot smo jih definirali v 2. poglavju. Predvsem tehnologije, ki so navedene za poslovne aplikacije, lahko uporabljamo tudi za gradnjo spletnih in kompleksnih poslovnih aplikacij.

Spletne aplikacije	*Poslovne aplikacije*	*Kompleksne poslovne aplikacije*
LAMP, Python, Ruby/Rail, PHP, AJAX, CMS, Zend, CakePHP, Flex, Django	ASP.NET, Silverlight, Hibernate, Grails, Struts Spring MVC, Wicket, Vaadin, REST	.NET, Java EE, SOAP, Spring, Seam, SOA

Tabela 5.1: Ustreznost tehnologij za spletne aplikacije glede na razred kompleksnosti.

5.3 Časovna kritičnost

Ta kriterij povzema časovne zahteve izdelave rešitve. Za enostavnejše in splošnejše poslovne aplikacije je .NET platforma najhitrejša, ker že vsebuje veliko kontrol, komponent in programskih vmesnikov v osnovni različici. Poleg tega je Visual Studio zmogljivo razvojno orodje, ki je dobro povezano s specifikacijami .NET platforme. Glede lahkotnosti uporabe je pogosto prva izbira razvijalcev [33]. Tudi LAMP platforma v kombinaciji z ustreznim ogrodjem omogoča hiter razvoj. Java platforma je zaradi svoje široke zasnove in odprtosti kompleksnejša in potrebuje več nastavitev.

5.4 Stopnja izvajanja na odjemalcu

Pri izbiri tehnologije je pomemben kriterij stopnja izvajanja aplikacijske logike na odjemalcu. S tem je mišljena zahteva po interaktivnosti uporabniškega

vmesnika, vektorske in medijske grafike, procesiranju na odjemalcu ipd. Spletne aplikacije lahko razdelimo v dve kategoriji [32]:

- **spletne strani** s statično vsebino in nekaj dinamične;
- **obogatene spletne aplikacije** s skoraj nič statične vsebine.

Za razvoj spletnih strani lahko uporabimo strežniška ogrodja, kot so Spring MVC, CakePHP, Django itd. Vsa ogrodja lahko uporabljajo AJAX tehnologijo z uporabo JavaScript knjižnic ali ogrodij, kot so JQuery, Dojo, Microsoft AJAX Control Toolkit itd.

V prejšnjem poglavju smo navedli nekaj najpopularnejših ogrodij za razvoj RIA aplikacij. Večina izmed njih potrebuje za razvoj kompleksnejših aplikacij tudi strežniške tehnologije za implementacijo dostopa do podatkov, varnost itd.

5.5 Dostopnost

Odjemalci do spletnih aplikacij dostopajo skozi spletne brskalnike, ki se nahajajo na osebnih računalnikih in vedno bolj razširjenih mobilnih napravah. Praktično vsi spletni brskalniki podpirajo osnovne spletne standarde, kot so HTTP, HTML, CSS in JavaScript. Omejitve se pojavijo pri uporabi tehnologij, ki zahtevajo namestitev vtičnikov. V drugem poglavju je navedeno, katere mobilne naprave imajo možnost namestitve vtičnikov za izvajanje Flex in Silverlight aplikacij.

Slike

Tabele

Literatura

[1] C. Mateu. (2010). Introduction to web applications development. Barcelona: Eureca Media.

[2] Web development. Dostopno na: http://en.wikipedia.org/wiki/Web_development

[3] Comparison of web application frameworks. Dostopno na: http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks

[4] B. Doyle, C. V. Lopes. (2008, Januar 17). Survey of Technologies for Web Application Development. Dostopno na: http://arxiv.org/abs/0801.2618

[5] Zgodovina CSS. Dostopno na: http://virtuelvis.com/archives/2005/01/css-history

[6] A. Meshab, A. van Deursen. (2006). An Architectural Style for Ajax. Delft University of Technology. Dostopno na: http://arxiv.org/ftp/cs/papers/0608/0608111.pdf

[7] B. Ličen. (2009, November). Obogatene spletne aplikacije, AJAX in orodjarne JavaScript. Univerza v Ljubljani. Dostopno na: http://eprints.fri.uni-lj.si/946/

[8] J. Allaire. (2002, Marec). A next-generation rich client. Macromedia Inc. Dostopno na: http://www.c2isoft.in/white-papers/richclient.pdf

[9] Guidance on Differences Between WPF and Silverlight. Dostopno na: http://wpfslguidance.codeplex.com/

[10] MIX Conference. Dostopno na: http://live.visitmix.com

[11] J. Yang. (2004, Avgust), Boosting Your .NET Application Performance. Dostopno na: http://www.developerfusion.com/article/3058/boosting-your-net-application-performance/2/

[12] I. Skerrett. (2010, November). Top 10 Most Popular Eclipse Plugins. Dostopno na: http://eclipse.dzone.com/articles/top-10-most-popular-eclipse

[13] M. Krebelj. (2010, December). Zasnova in izvedba ogrodja za razvoj spletnih aplikacij. Univerza v Ljubljani. Dostopno na: http://eprints.fri.uni-lj.si/1230/

[14] Java EE Productivity Report 2011. Dostopno na: http://www.zeroturnaround.com/java-ee-productivity-report-2011/

[15] N. Kothari. (2008, April). Ajax vs. Silverlight and .NET. Dostopno na: http://www.nikhilk.net/Ajax-vs-Silverlight-and-NET.aspx

[16] T. Hentenaar. (2008, Marec). Benchmark: PHP vs. Python vs. Perl vs. Ruby. Dostopno na: http://hentenaar.com/serendipity/index.php?/archives/27-Benchmark-PHP-vs.-Python-vs.-Perl-vs.-Ruby.html

[17] A. Fitzgerald. (2010, Maj). Spring adds GWT and Google App Engine Integration. Dostopno na: http://www.springsource.org/node/2595

[18] Is there any benefit of Silverlight for business applications? Dostopno na: http://betaforums.silverlight.net/forums/p/191860/444126.aspx

[19] J. Rose. (2007, December). Top 10 Adobe Flex Misconceptions. Dostopno na: http://www.infoq.com/news/2007/12/top-10-flex-misconceptions

[20] C. Eberhardt. (2009, November). Silverlight 4 beta released leaving Flex behind. Dostopno na: http://www.scottlogic.co.uk/blog/colin/2009/11/silverlight-4-beta-released-leaving-flex-behind/

[21] B. Kukovec. (2008). Primerjalna analiza spletnih programskih ogrodij. Univerza v Mariboru. Dostopno na: http://dkum.uni-mb.si/IzpisGradiva.php?id=9362

[22] K. Wahner. (2010, December). Categorization of Web-Frameworks in the Java Environment. Dostopno na: http://www.kai-waehner.de/blog/2010/12/30/categorization-of-web-frameworks-in-the-java-environment/

[23] R. Kalla. (2008, Junij). Which is the Hottest Java Web Framework? Or Maybe Not Java? Dostopno na: http://www.thebuzzmedia.com/which-is-the-hottest-java-web-framework-or-maybe-not-java/

[24] Jaspal. (2010, Junij). The Best Web Development Frameworks. Dostopno na: http://www.webdesignish.com/the-best-web-development-frameworks.html

[25] M. Raible. (2010, November). My Comparing JVM Web Frameworks Presentation from Devoxx 2010. Dostopno na: http://raibledesigns.com/rd/entry/my_comparing_jvm_web_frameworks

[26] G. Polančič. (2008). Integriran model ocenjevanja uspešnosti programskih ogrodij. Univerza v Mariboru. Dostopno na: http://dkum.uni-mb.si/IzpisGradiva.php?id=9554

[27] J. Chone. (2009, Marec). Enterprise Web vs Consumer Web [2.0]: Top Six Differences. Dostopno na: http://www.bitsandbuzz.com/article/enterprise-web-vs-consumer-web-20-top-six-differences/

[28] Microsoft BizSpark. Dostopno na: http://www.bizspark.com/Pages/home.aspx

[29] W3Techs - World Wide Web Technology Surveys. Dostopno na: http://w3techs.com/

[30] Alexa top Sites. Dostopno na: http://www.alexa.com/topsites

[31] A. Kranjc, C. Petr, G. Gos, B. Štok. (2005). Razvojna okolja kot ogrodja. Dostopno na: http://cot.uni-mb.si/cotl/jesen2005/krajnc.html

[32] S. Jivan. (2008, April). Thoughts on Java web frameworks and RIA. Dostopno na: http://www.jroller.com/sjivan/entry/thoughts_on_java_webframeworks_and

[33] K. Richards. (2010, April). Developers Rank Microsoft .NET Ahead of Google and Other Frameworks. Dostopno na: http://adtmag.com/articles/2010/04/27/developers-rank-ms-net-ahead-of-google.aspx

[34] S. Kachru, E. F. Gehringer. (2003, September). On The Relative Advantages of Teaching Web Services in J2EE vs. .NET. Dostopno na: http://research.csc.ncsu.edu/efg/oo/papers/J2EE_.NET.pdf

[35] D. Guinard, V. Trifia, O. Liechti. (2009, Junij). Toward Physical Maships in the Web of Things. Dostopno na: http://www.vs.inf.ethz.ch/publ/papers/guinardSensorMashups09.pdf

[36] Web 2.0. Dostopno na: http://en.wikipedia.org/wiki/Web_2.0

[37] J. O'Dell. (2010, Junij). After More Downtime, We Ask: Can Twitter Truly Scale? Dostopno na: http://mashable.com/2010/07/21/twitter-scalability/

[38] J. J. Garrett. (2005, Februar). Ajax: A New Approach to Web Applications. Dostopno na: http://experiencezen.com/wp-content/uploads/2007/04/adaptive-path-ajax-a-new-approach-to-web-applications1.pdf

[39] M Raible. (2010, April). History of Web Frameworks. Dostopno na: http://www.flickr.com/photos/mraible/4378559350/