UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matej Poklukar

Analiza značilnosti uporabe ogrodja GoogleWebToolkit za izdelavo uporabniških vmesnikov

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Mentorica: doc. dr. Mojca Ciglarič
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: 01786/2011
Datum: 02.11.2011

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

Kandidat: **MATEJ POKLUKAR**

Naslov: **ANALIZA ZNAČILNOSTI UPORABE OGRODJA GOOGLE WEB TOOLKIT ZA IZDELAVO UPORABNIŠKIH VMESNIKOV**

BUILDING USER INTERFACES WITH GOOGLE WEB TOOLKIT: USAGE PROPERTIES ANALYSIS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Opišite tipične težave v podjetju pri izdelavi uporabniških vmesnikov. Navedite, na kakšne različne načine lahko za izdelavo uporabniških vmesnikov uporabimo Google Web Toolkit in jih primerjajte z uporabo tehnologije Javascript. Za vsakega od načinov opišite teoretični koncept, izdelajte pilotno implementacijo in izpostavite prednosti in slabosti. Nazadnje primerno utemeljite izbiro najprimernejše možnosti.

Mentor:

M. Ciglarič

doc. dr. Mojca Ciglarič

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani **Matej Poklukar**,

z vpisno številko **63020124,**

sem avtor diplomskega dela z naslovom:

Analiza značilnosti uporabe ogrodja GoogleWebToolkit za izdelavo uporabniških vmesnikov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mojce Ciglarič
- 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, 1.12.2011 Podpis avtorja:

Zahvala

Ob zaključku študija, ki je trajal malo dlje kot je bilo pričakovano, se iz srca zahvaljujem svoji mentorici, doc. dr. Mojci Ciglarič za pomoč in svetovanje pri izdelavi diplomskega dela. Zahvaljujem se tudi dr. Matjažu Pančurju za usmerjanje pri nastajanju diplomskega dela.

Zahvaljujem se sodelavcem podjetja ResEvo d.o.o. za vso strokovno pomoč in znanje, ki so mi ga posredovali.

Posebej pa se zahvaljujem svoji družini, ki je me podpirala in mi stala ob strani vsa leta študija.

KAZALO VSEBINE

SEZNAM SLIK

SEZNAM UPORABLJENIH KRATIC IN SIMBOLOV

AJAX	Asynchronous JavaScript and XML (asinhroni JavaScript in XML)
ANT	Another Neat Tool (orodje, uporabljeno za avtomatsko prevajanje predvsem javanskih programov)
C, C++	splošnonamenski računalniški programski jezik (razvit med leti 1969 in 1973 v podjetju Bell Telephone Laboratories, njegova nadgradnja je C++, ki je bil razvit 1983)
CSS	Cascading Style Sheets (predloge, ki določajo izgled spletnih strani; z njimi določamo pisavo, velikosti črk ter vizualno predstavitev spletne strani)
DOM	Document Object Model (konvencija za predstavitev HTML, XHTML in XML dokumentov)
GWT	Google Web Toolkit
HTML	Hyper Text Markup Language (označevalni jezik za izdelavo spletnih strani)
JSON	JavaScript Object Notation (odprt, tekstovni in človeku lahko berljiv standard za izmenjavo podatkov)
JVM	Java Virtual Machine (navidezni stroj, ki je zmožen izvajati prevedeno javansko kodo)
UI	User Interface (opisuje procedure in metode, s katerimi uporabnik upravlja računalniški program)
WEB SERVICE	Web service oz. spletna storitev (sistem za interakcijo dveh računalniških sistemov preko omrežja)
XML	Extensible Markup Language (enostaven, fleksibilen tekstovni format)

POVZETEK

Namen diplomskega dela je pregled in analiza različnih tehnik razvoja uporabniškega vmesnika spletnih strani oz. aplikacij z uporabo programskega orodja Google Web Toolkit.

Pri razvoju spletnih strani oz. aplikacij se programer in oblikovalec srečata s problemom, na kakšen način naj oblikovalec poda obliko spletne strani programerju. Pri tem pa ne gre samo za način prenosa ideje oblike internetne strani oz. aplikacije, ampak tudi za razmejitev odgovornosti pri vzdrževanju in morebitnem odpravljanju napak.

Oblikovalec lahko obliko internetne strani oz. aplikacije nariše enostavno na papir ali v ustreznem programu in jo pošlje programerju, lahko pa se posluži naprednejših načinov prenosa oblike – obliko spletne strani poda kot množico različnih datotek, ki jih programer opremi z ustrezno funkcionalnostjo, oz. jih uporabi pri svojem delu.

Cilj diplomske naloge je analizirati različne načine prenosa oblike internetne strani od oblikovalca do programerja in ugotoviti, kateri od teh načinov je za programerja najprimernejši. Pri analizi upoštevamo tendenco, da naj se programer čim bolj ukvarja s funkcionalnostjo spletne strani in čim manj s spreminjanjem oblike, saj je to delo oblikovalca.

Ključne besede: spletna stran, razvoj, oblikovalec, programer, uporabniški vmesnik, tehnika razvoja, Google Web Toolkit

ABSTRACT

This Bachelor thesis provides an overview of different techniques used for web pages or web applications user interface development using Google Web Toolkit.

During development process programmer and designer encounter the problem in which way should design be passed from designer to programmer. Designer can plot web page design easily on paper or in an appropriate program and sends it to the programmer. Or he can create design in more advanced form, as a multitude of different files. Programmer will use those files and add them appropriate functionality and content.

The goal of this thesis is to analyze different ways of transferring design from designer to programmer, to figure out a way, which would be most convenient for the programmer. In analysis we are considering the trend, that programmer should, as much as he can, deal with webpage functionality and as little as possible with design changing, which is the work of the designer.

Key words: web page, development, designer, programmer, user interface, development tehniques, Google Web Toolkit

1 UVOD

V preteklosti so bili računalniki veliki in dragi, programi, ki so se izvajali na njih, pa so bili bolj ali manj namenjeni izključno delu. Računalniški programi so bili večinoma tekstovni, interakcija z njimi pa je potekala preko tipkovnice. Z razmahom osebnih računalnikov so se spremenile vsebine, s katerimi uporabnik upravlja na računalniku. Tekstovne programe so zamenjali programi z grafičnim prikazom podatkov, interakciji s tipkovnico pa se je pridružila tudi miška. Z razmahom interneta v začetku 90 let prejšnjega stoletja pa je bilo praktično vsakomur omogočeno izdelati internetno stran in nanjo postaviti različne vsebine.

Od leta 1990, ko so se pojavile prve internetne strani, pa do današnjih dni so se le-te nepredstavljivo spremenile. Če so bili uporabniki še pred 20 leti zadovoljni z internetnimi stranmi, ki so vsebovale samo besedilo, lahko danes za take strani napovemo, da ne bodo imele veliko obiska. Obiskovalci internetnih strani oz. portalov namreč poleg besedila in slik zahtevajo tudi avdio in video vsebine.

Poleg tega uporabniki niso več samo pregledovalci vsebin, ampak jih želijo tudi soustvarjati. Z WEB 2.0 revolucijo, ko so internetne strani postale dinamične in interaktivne, je uporaba AJAX tehnologije postala *de facto* standard pri razvoju internetnih strani.

Da bo internetna stran oblikovalsko brezhibna, morajo razvijalci nameniti veliko pozornosti njeni obliki. Stran je lahko tehnološko še tako dovršena, hitra in polna informacij, vendar, če ni tudi oblikovno prijetna očem in če uporabniška izkušnja ni pozitivna, bo ostala neopažena. Zato se v razvoj internetnih strani poleg programerjev vključuje tudi oblikovalce, ki definirajo njeno obliko, katero programerji nato opremijo s funkcionalnostjo oz. jo napolnijo z dinamično vsebino.

Razvoj internetnih strani tako ni več samo v domeni razvijalca, temveč je potrebno njegovo sodelovanje tudi z oblikovalcem. Pri tem sodelovanju pa pogosto prihaja do vprašanja, kje je meja med delom razvijalca in oblikovalca, oz. kako uskladiti njuno delo tako, da bo prenos oblike med njima čim lažji in čim bolj nedvoumen. Z jasno mejo med obliko in vsebino strani ter z enostavnim prenosom njene oblike bo oblikovalec odgovoren samo za obliko, razvijalec pa samo za funkcionalnost internetne strani.

V diplomskem delu bom poskušal poiskati optimalen način sodelovanja med razvijalcem in oblikovalcem, pri tem pa se bom srečal z vprašanjem, na kakšen način naj oblikovalec poda obliko internetne strani razvijalcu. Analiziral bom nekaj različnih tehnik grajenja uporabniškega vmesnika na tehnologiji Google Web Toolkit in na programskem jeziku JavaScript.

Diplomsko delo je razčlenjeno po smiselnih enotah oz. poglavjih. To poglavje je kratek uvod v delo. V drugem poglavju je opisan problem in predstavljen načrt njegove rešitve. Tretje poglavje vsebuje teoretično razlago pojmov in tehnologij, ki jih mora bralec poznati za razumevanje analize različnih tehnologij, ki so predstavljene v četrtem poglavju. V petem poglavju sledi analiza tehnologij, v šestem pa so podane sklepne ugotovitve.

2 PROBLEM IN CILJI

2.1 Problem grajenja uporabniškega vmesnika

Pri razvoju internetne strani naletimo na dva problema, in sicer na kakšen način naj bo zgrajen uporabniški vmesnik spletne aplikacije (tehnologija) ter kako naj bo dostavljen izgled aplikacije (design) od oblikovalca do programerja.

Oblikovalec lahko obliko strani nariše enostavno na papir ali v ustreznem programu in jo pošlje programerju. Ta nato internetno stran sprogramira v skladu z obliko, ki jo je prejel. Pri takem načinu razvoja internetne strani lahko prihaja do napačnega programerjevega razumevanja oblike, kar povzroči dodatno delo in razlago oblikovalcu. Prihaja lahko tudi do popravljanja osnovne oblike strani, kar povzroči veliko dodatnega dela programerju. Tak način podajanja oblike internetne strani je zato nezaželen, saj z njim nastane veliko dela pri spremembah in terja ogromno usklajevanja.

Zaradi opisanih težav je tendenca razvijalcev internetnih strani, da dobijo obliko internetne strani v takem formatu, ki ga lahko dopolnijo s funkcionalnostjo. Oblikovalci internetnih strani se morajo zaradi teh tendenc naučiti tehnologij, za katere bi sami smatrali, da jim jih ni potrebno poznati. Seveda se od njih ne zahteva znanja programiranja, poznati pa morajo osnove internetnih strani oz. HTML-ja [1] [2] in CSS-a [3].

V primerih, kjer oblikovalec pozna in uporablja osnove internetnih strani, se razvijalec ne ukvarja s postavitvijo objektov na strani (npr. na kateri poziciji je kateri objekt in koliko je velik), ampak samo z njegovo funkcionalnostjo. Tak način razvoja internetnih strani je hitrejši in z manj konflikti med oblikovalci in programerji, saj so njihove pristojnosti in odgovornosti točno določene in znane.

Pri takem razvoju oblikovalec kreira datoteke z zapisom oblike strani in jih dostavi razvijalcu. Vendar pa je tudi pri tem načinu sodelovanja več možnosti zapisa oblike. Razvijalec lahko prejme datoteke z neko dogovorjeno obliko zapisa, lahko pa prejme končne HTML in CSS datoteke.

S podobnim problemom sem se soočil v službi, kjer je bila moja naloga razviti uporabniški vmesnik multimedijskega portala za projekt DomProd.

2.2 Kratka predstavitev projekta DomProd

Projekt DomProd je izdelek, ki vsebuje vsa orodja, ki jih potrebujemo za postavitev svoje internetne televizije oz. portala z multimedijskimi vsebinami. To nakazuje že samo ime aplikacije – DomProd – domača produkcija.

Celoten projekt DomProd je sestavljen iz dveh modulov, in sicer iz administratorskega in uporabniškega. Funkcija administratorskega modula je sestavljanje programa internetne televizije iz posnetkov, ki so predhodno obdelani v programu Final Cut Server. Po obdelavi posnetka ima uporabnik administratorskega modula možnost posnetek uvrstiti v spored internetne televizije.

Uporabniški modul je sestavljen iz Wowza streamerja in internetnega portala, ki skupaj posredujeta vsebine uporabniku.

Ker nisem želel razvijati uporabniškega vmesnika v tehnologiji, ki sem jo že poznal, sem namenil nekaj časa analizi meni manj znanih tehnologij.

Slika 2.1: Stran portala za predvajanje multimedijskih vsebin

Slika 2.1: Stran portala za predvajanje multimedijskih vsebin

2.3 Cilji

Cilji tega diplomskega dela so predstaviti razvoj internetnih strani oz. aplikacij z orodjem Google Web Toolkit, oblikovati načrt rešitve, s katero bodo predstavljene različne možnosti izdelave uporabniškega vmesnika, in oblikovani načrt izvesti. Po izvedbi praktičnega dela načrta sem naredil analizo izvedbe ter izpeljal sklep, kateri način prenosa oblike internetne

strani oz. aplikacije med oblikovalcem in programerjem je najboljši (gledano s stališča učinkovitega razvoja, sprotnega testiranja ter hitrega izvajanja aplikacije).

3 NAČRT REŠITVE

Za preučitev različnih možnosti grajenja uporabniškega vmesnika bom izdelal spletno stran oz. aplikacijo z izbirnikom multimedijskih vsebin. Spletno aplikacijo bom izdelal z uporabo orodja Google Web Toolkit (GWT) [4].

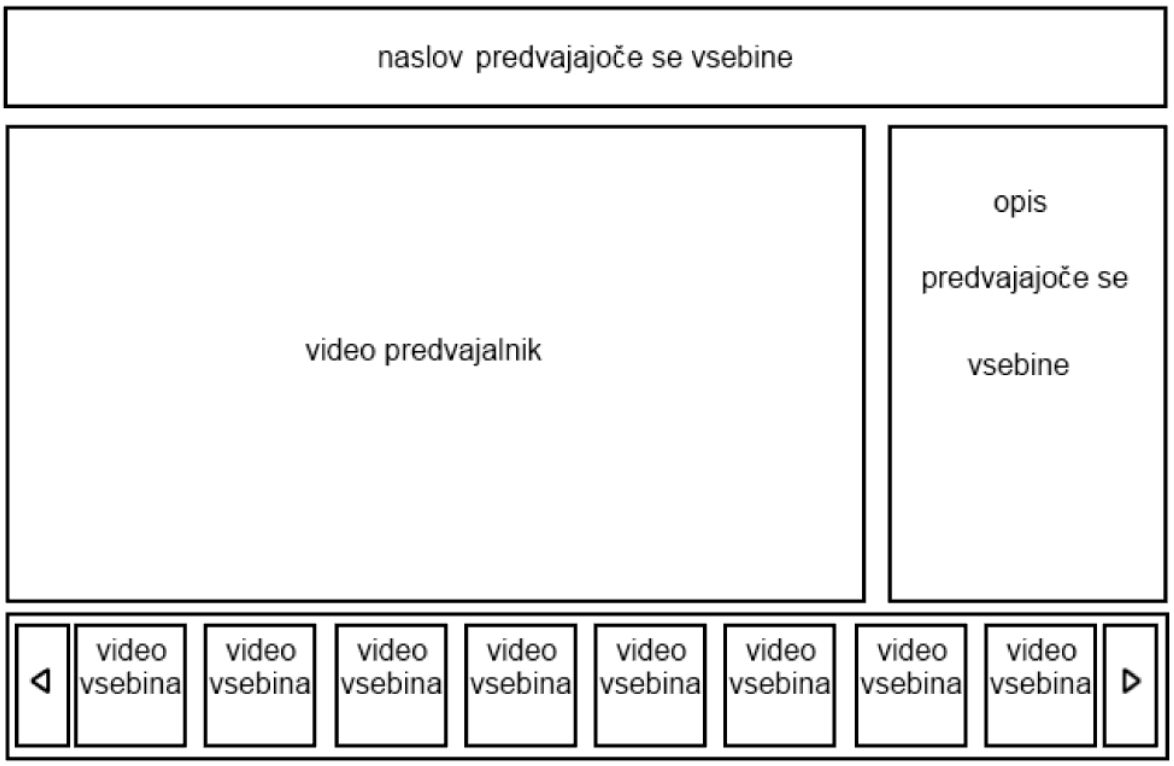
Uporabniški vmesnik spletne aplikacije bo zgrajen za 4 različne načine, ki se med seboj razlikujejo v načinu podajanja oblike med oblikovalcem in razvijalcem aplikacije ter tudi po načinu opremljanja aplikacije s funkcionalnostjo.

Ti načini so:

- način grajenja uporabniškega vmesnika na podlagi slikovne oblike z uporabo GWT objektov,
- način grajenja uporabniškega vmesnika z uporabo tehnike GWT UiBinder,
- način grajenja uporabniškega vmesnika iz HTML predloge,
- način grajenja uporabniškega vmesnika z uporabo knjižnice jQuery.

Za analizo zgoraj naštetih načinov grajenja uporabniškega vmesnika bom z vsako izmed tehnologij izdelal del aplikacije DomProd.

Aplikacija bo sestavljena iz video predvajalnika in treh panelov, ki bodo postavljeni nad, levo in pod video predvajalnik.



Slika 3.1: Osnovna postavitev elementov multimedijskega portala

Panel nad video predvajalnikom bo prikazoval naslov trenutno predvajajoče se video vsebine, panel desno od predvajalnika pa bo prikazoval opis predvajajoče se vsebine.

Panel pod video predvajalnikom bo kompleksnejši, saj bo prikazoval vsebine, med katerimi lahko uporabnik izbira. Ta panel vsebuje puščici, med katerima so elementi s slikami video vsebin. S pritiskom na puščice se ti elementi pomikajo levo in desno. S klikom na element je video vsebina izbrana. Njen naslov se prikaže nad video predvajalnikom, opis se prikaže desno od video predvajalnika, vsebina pa se prične predvajati.

Poudarek praktičnega dela diplomske naloge ni na kompleksnosti in funkcionalnosti aplikacije. Le-ta je zelo okrnjena. Poudarek je na različnosti metod za grajenje uporabniškega vmesnika.

4 TEHNOLOGIJE IN IZVEDBA

V tem poglavju so na kratko razloženi pojmi, ki jih mora bralec poznati, da razume način delovanja internetnih aplikacij, narejenih z GWT. Najprej so predstavljeni osnovni pojmi internetnih tehnologij, nato orodje GWT, knjižnica jQuery in predvajalnik FlowPlayer.

4.1 Osnovni pojmi internetnih tehnologij

Pojem internetne tehnologije pomeni različne tehnologije, s katerimi so zgrajene internetne strani, portali oz. aplikacije.

V tem poglavju si bomo podrobneje ogledali:

- programski jezik Java,
- Java Virtual Machine,
- Hyper Text Markup Language (HMTL),
- JavaScript,
- Asynchronous JavaScript and XML (Ajax),
- JavaScript Object Notation (JSON),
- Servlet.

Programski jezik Java

Java je programski jezik, ki ga je razvil James Gosling pri Sun Microsystemsu (danes Oracle Corporation), in je bil predstavljen leta 1995 kot jedro platforme Java podjetja Sun Microsistems [5]. Sintaksa jezika izhaja iz sintakse programskih jezikov C in C ++, vendar ima preprostejši objektni model in malo objektov z majhno stopnjo abstrakcije.

Java aplikacije so običajno prevedene v datoteko z bitno kodo (*bytecode*), katero lahko izvaja Java Virtual Machine ne glede na arhitekturo računalnika. Java je splošno namenski objektno orientiran programski jezik, ki je zasnovan tako, da je njegovo izvajanje čim bolj neodvisno od zunanjih dejavnikov.

Namen programskega jezika Java je omogočiti razvijalcem, da aplikacijo razvijejo enkrat, izvajajo pa kjer koli (ang. *write once, run anywhere*). Java je trenutno eden izmed najbolj priljubljenih programskih jezikov in se uporablja za razvoj namiznih, mobilnih in spletnih aplikacij.

Java Virtual Machine

Java Virtual Machine (JVM) je platformno neodvisno izvajalno okolje, ki pretvori bitno kodo Java aplikacij v ustrezne ukaze strojne opreme in jih izvede [6]. Java je bila zasnovana tako, da je program napisan in preveden enkrat, izvaja pa se enako na različnih računalniških arhitekturah. To omogoča JVM, saj upošteva specifiko različnih dolžin ukazov in ostalih posebnosti platform, kjer se izvaja.

Hyper Text Markup Language

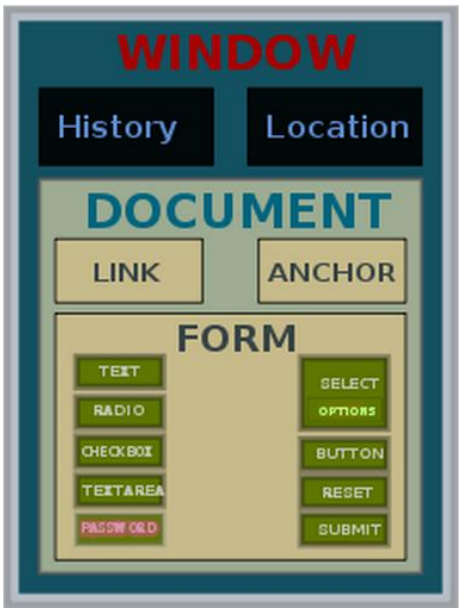
Hyper Text Markup Language (HTML) je osnovni gradnik internetnih strani. V HTML-ju je definiran izgled internetne strani, ki jo definira serija oznak oz. tagov.

JavaScript

JavaScript je programski jezik, ki se izvaja znotraj brskalnika in omogoča kreiranje interaktivnih, dinamičnih internetnih strani oz. aplikacij [7], [8]. Z uporabo JavaScripta je omogočena zakulisna komunikacija med odjemalcem in strežnikom, osveževanje vsebine in prilagajanje izgleda internetne strani različnim odjemalcem.

Document Object Model

Document Object Model (DOM) [9] je platformno in jezikovno neodvisen način predstavitve in upravljanja z objekti v HTML, XHTML in XML dokumentih. Objekti so dosegljivi in manipulirani s sintaksami jezikov preko aplikacijskega programskega vmesnika (API).



Slika 4.1: Enostaven prikaz arhitekture dokumenta v brskalniku

Asynchronous JavaScript and XML

Asynchronous JavaScript and XML (AJAX) [10] je skupina odjemalskih metod za kreiranje interakcije med odjemalcem in strežnikom. Z uporabo AJAX metod lahko internetne aplikacije na odjemalski strani pridobijo podatke s strežnika asinhrono, brez spreminjanja trenutno prikazujoče se strani. Programski jezik JavaScript in AJAX sta glavna razloga za razvoj dinamičnih internetnih strani (WEB 2.0).

JavaScript Object Notation

JavaScript Object Notation (JSON) [11] je enostaven tekstovni standard za prenos in izmenjavo berljivih podatkov. Izhaja iz programskega jezika JavaScript. Z uporabo JSON-a je poenostavljena predstavitev podatkovnih struktur in asociativnih tabel. Kljub dejstvu, da JSON izhaja iz jezika JavaScript, je jezikovno neodvisen in je uporabljen v mnogih programskih jezikih. Zaradi teh lastnosti je JSON idealen standard za prenos podatkov.

Servlet

Servlet [12] je razred programskega jezika Java, uporaben za razširitev zmogljivosti strežnikov, ki gostijo internetne aplikacije, dostopne preko programskega modela zahteva – odziv. Za take strežniške aplikacije tehnologija Java Servlet definira poseben HTTP razred. Z uporabo servletov lahko internetni razvijalci kreirajo hitre in učinkovite strežniške dele aplikacij, ki strežejo zahtevam odjemalcev.

4.2 Google Web Toolkit

Google Web Toolkit (GWT) je odprtokodno okolje za razvoj in optimiziranje kompleksnih internetnih aplikacij, ki temeljijo na tehnologiji AJAX. Orodje GWT je izdano pod licenco Apache 2.0 [13].

Cilj orodja GWT je omogočiti programerjem razvoj visokoperformančnih internetnih aplikacij brez poznavanja podrobnosti brskalnika, vmesnika XMLHttpRequest in skriptnega jezika JavaScript. To je doseženo z možnostjo programiranja odjemalskega in strežniškega dela aplikacije v programskem jeziku Java.

GWT je uporabljen v veliko produktih podjetja Google, vključno z Google Wave in novo verzijo AdWords. Okolje GWT je odprtokodno in tako v celoti brezplačno, uporabljajo pa ga razvijalci po celem svetu.

GWT ponuja učinkovite rešitve za ponavljajoče AJAX klice, imenovane klici oddaljenih procedur, za upravljanje z zgodovino, zaznamki, večjezičnostjo in prenosljivostjo med različnimi brskalniki.

Orodje Google Web Toolkit je bilo predstavljeno 16. maja 2006 z verzijo 1.0. RC1. Od takrat je bilo predstavljenih še 15 posodobitev GWT. Trenutno je v uporabi verzija 2.4.

Delovanje orodja GWT

Delovanje GWT bom razložil v več podpoglavjih.

Programiranje odjemalskega dela aplikacije

Ena izmed največjih prednosti orodja GWT je možnost programiranja odjemalskega dela v programskem jeziku Java. Pri razvoju uporabljamo sintakso in objekte programskega jezika Java. Pri programiranju odjemalskega dela se seveda moramo zavedati, da smo omejeni z

internetnim brskalnikom, oz. da lahko izvajamo operacije, ki jih brskalniki omogočajo (npr. ne moremo odpreti in brati datoteke iz diska).

Pri programiranju odjemalskega dela aplikacije v programskem jeziku Java lahko vključimo tudi jezik JavaScript. Z vključitvijo JavaScripta se nam razširi nabor funkcij na odjemalcu, oz. z uporabo JavaScripta nam je omogočen dostop do virov, do katerih ne moremo dostopati z Javo. S tem mislim na dostop do lastnosti brskalnika, krmiljenje raznih vtičnikov ter raznih objektov na spletni strani – npr. predvajalniki glasbe in videa ponujajo dostop do svojih notranjih funkcij preko JavaScript vmesnika.

Komunikacija med Javo in JavaScriptom poteka preko vmesnika JavaScript Native Interface (JSNI) [13]. Preko JSNI vmesnika lahko funkcijam JavaScripta posredujemo podatke podatkovnih tipov String, int, boolean in podatkovnega tipa JavaScriptObject, ki je implementiran v GWT. Telo funkcije JSNI je obdano s posebnimi znaki /*-{ in }-*/;, v deklaraciji funkcije pa je identifikator *native*.

```
40
41  private void testJsCall()
42  {
43    jsAlert("Klic iz Jave v JavaScript");
44  }
45
46  public native void jsAlert(String text) /*-{
47    alert(text);
48  }-*/;
49
```

Slika 4.2: Primer klica JavaScript funkcije iz Jave

Komunikacija med programskima jezikoma je obojesmerna – iz JavaScripta lahko dostopamo tudi do Java funkcij na odjemalskem delu aplikacije. Klic funkcije je v tem primeru malo drugačen. Pri klicu javanske funkcije iz JavaScripa moramo najprej navesti ime paketa (v primeru na sliki 4.3 je to diploma.client), ime razreda (Diploma) in nato ime javanske funkcije. Sledi navedba tipov argumentov in na koncu še argumenti, ki jih podamo ob klicu.

```
51
52  public native void testJavaCall(String name) /*-{
53
54    var s = 'Klic iz JavaScripa v Javo';
55    this.@diploma.client.Diploma::javaAlert(Ljava/lang/String;)(s);
56  }-*/;
57
58  private void javaAlert(String text)
59  {
60    Window.alert(text);
61  }
62
```

Slika 4.3: Primer klica Java funkcije iz JavaScripta

Zaradi pisanja v programskem jeziku Java so omogočene številne funkcije v razvojnem okolju, ki nam olajšajo programiranje. S tem mislim predvsem na avtomatično preverjanje sintakse, pretvorbe med podatkovnimi tipi, iskanje hierarhije klicev ipd. Te operacije omogočajo različna razvojna okolja, med katerimi so najbolj znana Eclipse, NetBeans in IntelliJ IDEA.

Zaradi zgoraj naštetih funkcij razvojnih okolij je programiranje v Javi veliko lažje kot v JavaScriptu. Posebno poglavje pa je razhroščevanje odjemalskega dela, kar bom predstavil malo kasneje.

Komunikacija med odjemalcem in strežnikom

Bistvena razlika med AJAX in HTML internetnimi aplikacijami je v tem, da AJAX aplikacijam ni potrebno nalagati celotne strani s strežnika ob interakciji z uporabnikom. AJAX aplikacije živijo v brskalniku, zato ne potrebujejo novega HTML-ja s strežnika za spremembo uporabniškega vmesnika, ampak le posodobijo že obstoječega. Pri spremembi uporabniškega vmesnika pa potrebujejo podatke s strežnika.

V GWT je implementiran poseben mehanizem za komunikacijo brskalnika s strežnikom. Ta mehanizem se imenuje klic oddaljene procedure (*remote procedure call* – RPC) [15]. GWT RPC mehanizem bazira na Java servletih in omogoča dostop do strežniških virov.

Prednost GWT RPC mehanizma je, da omogoča serializacijo in deserializacijo javanskih objektov in njihov prenos med brskalnikom in strežnikom preko http protokola. Zaradi možnosti prenosa javanskih objektov, ki jih na odjemalskem delu uporabimo v Javi, je komunikacija med odjemalcem in strežnikom zelo poenostavljena.

Pravilna uporaba GWT RPC omogoča prenos dela logike uporabniškega vmesnika na brskalnik, kar se rezultira v velikem performančnem izboljšanju sistema, zmanjšanem prometu na omrežju, zmanjšanju obremenitve strežnika in bolj tekoči uporabniški izkušnji.

Komunikacija med brskalnikom in strežnikom je asinhrona. Koda se ne izvaja nujno zaporedno, kar sili programerje v pokrivanje situacij, ko se klic na strežnik izvaja, vendar še ni končan. Čeprav asinhronost na prvi pogled predstavlja slabost, nam omogoča paralelno izvajanje kode v brskalniku, kljub temu da nimamo večnitnosti.

Pri asinhronem delovanju mora programer ves čas paziti, da se bodo dogodki, kljub asinhronosti klicev, izvajali zaporedno. V zgornjem primeru mora sprožiti polnjenje tabele s podatki ob uspešnem dokončanju klica na strežnik. To programer zagotovi z začetkom polnjenja pridobljenih podatkov v tabelo v metodi onSuccess, ki je definirana v asinhronem vmesnika RPC.

Strežnik

Strežniški del GWT aplikacij oz. del strežniške aplikacije, ki komunicira z brskalnikom, je implementiran v programskem jeziku Java. Ta omogoča pridobivanje podatkov iz podatkovnih baz preko spletnih storitev, oz. na strežniškem delu aplikacije lahko uporabimo polno funkcionalnost, ki nam jo ponuja programski jezik Java.

Prevajanje odjemalske kode

Za prevajanje odjemalske kode iz Jave v JavaScript se uporablja GWT prevajalnik. Če programiramo v okolju Eclipse, potem zaženemo predavanje s klikom na Compile project gumb. Če pa prevajamo brez programskega okolja, ali če je prevajanje samodejno (na primer na repozitoriju), potem GWT projekte prevajamo z uporabo orodja Another Neat Tool (ANT).

Pri prevajanju prevajalnik kreira več verzij aplikacije, in sicer za vsak brskalnik in vsako lokalizacijo svojo verzijo. Če imamo aplikacijo v 3 jezikih in jo podpremo v 4 brskalnikih, potem GWT pri prevajanju skreira 12 verzij aplikacije [16]. Med izvajanjem GWT nato uporabi ustrezno verzijo aplikacije, ki jo pokaže v brskalniku.

Pri programiranju odjemalskega dela programerju tako ni potrebno pretirano skrbeti za izvajanja kode na različnih brskalnikih. Če pri programiranju uporabljamo objekte, ki so implementirani v GWT, bodo aplikacije delovale podobno na vseh novejših verzijah brskalnikov Internet Explorer, Firefox, Chrome, Safar in Opera. Uporabniški vmesnik je zelo prilagodljiv posameznemu brskalniku, vsekakor pa je potrebno aplikacijo stestirati na vsakem brskalniku.

Kadar koli je možno, GWT prevede svoje elemente v elemente brskalnikov. Tako na primer objekt GWT Button postane HTML objekt <button>, namesto da bi postal gumbu podoben element . To pomeni, da se GWT gumb ustrezno pretvori in prikaže v različnih brskalnikih in operacijskih sistemih. GWT prepusti prikazovanje in kontrolo objektov brskalniku, saj je le-ta hitra, dostopna in uporabnikom znana.

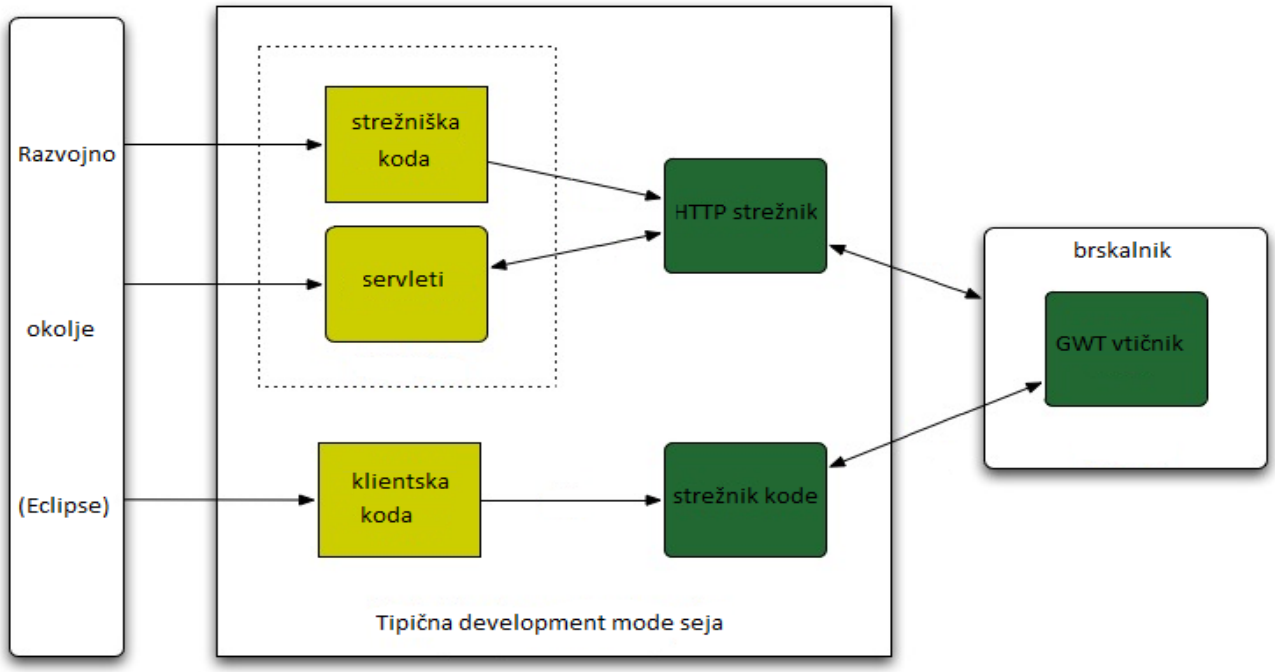
Prevajanju aplikacije za posamezni brskalnik pravimo prevajanje permutacije. Ker je prevajanje več permutacij zamudno (še posebno med razvojem, ko je prevajanje dokaj pogosto), ga lahko omejimo na eno permutacijo – tako bo aplikacija prevedena samo za en brskalnik.

Razhroščevanje odjemalskega dela

Razhroščevanje odjemalskega dela je omogočeno tako v razvojem okolju kot v brskalniku. Pri razhroščevanju aplikacije le-to zaženemo v t. i. *development modu* [17]. Pri tem načinu izvajanja se zažene vgrajen strežnik Jetty, kar nam omogoča zagon aplikacije brez namestitve na testni strežnik.

Development mode način izvajanja aplikacije je način izvajanja aplikacije v brskalniku, brez da je le-ta prevedena v JavaScript. Java Virtual Machine (JVM) izvaja kodo aplikacije, z uporabo GWT vtičnika pa se odjemalski del nato izvaja v brskalniku.

GWT vtičnik zapolni vrzel med javansko kodo v razvojem okolju in kodo, ki se izvaja v brskalniku. Tako lahko v razvojem okolju spremljamo koračno izvajanje javanske kode (*step mode execution*), postavljamo prekinitvene točke (*breakpoint*) in pregledujemo vrednosti spremenljivk, skratka uporabimo vse načine razhroščevanja, kot da bi razhroščevali namizno aplikacijo.



Slika 4.4: Arhitektura izvajanja GWT projekta v »*development modu*«

Pri razhroščevajnu kode lahko v brskalniku uporabimo tudi orodje Firebug, s katerim pregledujemo grafično postavitev objektov, njihove lastnosti in uporabo mrežnih virov.

Pripomočki za delo z GWT-jem, dodatne knjižnice, JavaDoc

Za delo z GWT-jem obstaja GWT vtičnik za orodje Eclipse. Z uporabo tega vtičnika lahko z nekaj kliki zgeneriramo osnovni GWT projekt, ki ga potem uporabimo za nadaljnje delo.

Za GWT je razvitih nekaj knjižnic, ki nam ponujajo lepe grafične objekte in katere uporabimo za lep prikaz podatkov. Dve taki knjižnici sta na primer SmartGWT in ExtGWT.

GWT ima tudi obširno dokumentacijo, ustvarjeno z generatorjem JavaDoc. Spletna stran code.google.com/webtoolkit je zelo bogata z raznimi vodiči za začetek dela z GWT, s primeri uporabe GWT-ja in primeri dobrih praks programiranja.

4.3 Knjižnica jQuery

jQuery [18] je JavaScript knjižnica, s katero je poenostavljeno programiranje odjemalskega dela spletnih aplikacij. Izdana je bila januarja 2006, danes pa je sestavni del preko 41 % od 10000 najbolj obiskanih internetnih strani. Zaradi tega je jQuery najbolj uporabljena JavaScript knjižnica na svetu. jQuery nam omogoča enostavno navigacijo dokumentov, izbiranje DOM objektov, kreiranje animacij, obvladovanje dogodkov in izvajanje AJAX klicev.

Knjižnica jQuery je uporabljena v več naprednejših JavaScript knjižnicah, med njimi tudi v knjižnici jQTouch, ki se uporablja za razvijanje odjemalskega dela spletnih aplikacij mobilnih naprav.

4.4 Predvajalnik FlowPlayer

Flowplayer [19] je flash video predvajalnik, s pomočjo katerega vgradimo video posnetke v internetne aplikacije. Predvajalnik je brezplačen za nekomercialno uporabo. Omogoča visoko stopnjo prilagoditve in krmiljenje preko JavaScript vmesnika.

4.5 Pregled tehnologij

Pri analizi izdelave uporabniškega vmesnika z orodjem GWT bomo uporabili 4 različne tehnologije oz. tehnike. Te so:

- način grajenja uporabniškega vmesnika z uporabo GWT objektov. Pri tem načinu grajenja celotni uporabniški vmesnik zgradi programer z uporabo GWT objektov, ki jih ustrezno opremi s stili ter jih sestavlja do končnega izgleda aplikacije.
- Način grajenja uporabniškega vmesnika z uporabo GWT UiBinderja. Pri uporabi UiBinderja oblikovalec dostavi programerju predloge uporabniškega vmesnika, na katerih bazira izgled. Programer predlogam doda funkcionalnost, ne ukvarja pa se s postavitvijo in stili gradnikov.
- Način grajenja uporabniškega vmesnika z uporabo GWT in zavijanja objektov. Z uporabo metode zavijanja programer zavije gradnike iz HTML datoteke, ki mu jo dostavi oblikovalec, v GWT objekte, in jih opremi z funkcionalnostjo in vsebino.
- Način grajenja uporabniškega vmesnika z uporabo JavaScript knjižnice jQuery. Podobno kot pri prejšnji metodi tudi tukaj oblikovalec dostavi programerju HTML datoteko z vsebino. Programer pri tem načinu ne uporabi orodja GWT, ampak se spusti na nivo čistega JavaScripta, s katerim HTML datoteko opremi s funkcionalnostjo ter napolni z vsebino.

5 ANALIZA IZDELAV UPORABNIŠEGA VMESNIKA Z GWT

V tem poglavju so predstavljene različne tehnologije izdelave uporabniškega vmesnika ter njihova analiza.

V tradicionalnem načinu programiranja z uporabo JavaScripta je grajenje dinamičnega uporabniškega vmesnika narejeno z manipulacijo brskalnikovega DOM-a. Ker GWT omogoča dostop do brskalnikovega DOM-a neposredno z uporabo DOM razredov, je uporaba gradnikov iz hierarhije Widget za grajenje uporabniškega vmesnika enostavna. Widget gradniki tako omogočajo hitro gradnjo uporabniškega vmesnika, ki bo deloval pravilno na vseh brskalnikih.

Razredi uporabniškega vmesnika GWT so podobni tistim v obstoječih tehnologijah grajenja uporabniških vmesnikov, kot na primer Java Swing. Razlikujejo se po načinu renderiranja, saj uporabljajo dinamično zgrajen HTML, namesto grafik, temelječih na pikslih.

5.1 Način grajenja uporabniškega vmesnika z uporabo GWT objektov

Oblikovalec poda obliko internetne strani kot sliko. Programer nato z uporabo panelov in gradnikov (*widget*) sprogramira uporabniški vmesnik.

5.1.1 Teorija

Programer ima pri načinu grajenja UI z uporabo GWT objektov (zaradi podobnosti z Java Swing tehnologijo jo bom v nadaljevanju imenoval swing metoda) na voljo več različnih objektov, ki mu jih ponudi GWT. Objekti se v grobem delijo na 2 vrsti, in sicer na:

- panele [20],
- gradnike [21].

Paneli so v GWT svetu zelo podobni kot v drugih knjižnicah za uporabniške vmesnike. Glavna razlika je v tem, da GWT uporabi HTML elemente za postavitev svojih otrok. GWT ponuja 12 različnih panelov, od najvišjega statičnega panela do panelov, ki omogočajo drsnike, se razlikujejo po različnih načinih dodajanja gradnikov (horizontalno, vertikalno) in imajo funkcionalnost prikazovanja posebej določenega HTML-ja.

Paneli lahko vsebujejo gradnike in druge panele. Pogosto se uporabljajo za osnovno definicijo postavitve sklopov v brskalniku.

Gradniki so elementi, preko katerih GWT aplikacija izvaja interakcijo z uporabnikom. Preko gradnikov lahko uporabnik sporoča aplikaciji svoje ukaze in podatke (klikne na gumb, vnese tekst, izbere eno izmed možnosti ipd.), lahko pa jih uporabi za različno prikazovanje teksta ali kompleksnejših struktur (dreves).

Paneli in gradniki skupaj tvorijo uporabniški vmesnik. Paneli kontrolirajo postavitev elementov uporabniškega vmesnika na strani, gradniki pa komunikacijo z uporabnikom. Pri GWT paneli in gradniki delujejo na enak način v vseh brskalnikih, kar poenostavi programiranje.

Vsi gradniki uporabniškega vmesnika so po hierarhiji pritrjeni na RootPanel. Ta je najvišji panel na hierarhični lestvici in glavni nosilec vmesnika.
Vsem gradnikom na uporabniškem vmesniku lahko določimo stil. To lahko naredimo na dva načina: direktno z določitvijo stila posameznemu elementu ali z uporabo stilov, ki so definirani v CSS datoteki, pri čemer se poslužimo uporabe identifikatorja gradnika ali razreda, kateremu gradnik pripada. Ker je prvi način določanja stila precej zamuden in nepriročen v primeru popravljanja, saj zahteva popravek kode in ponovno prevajanje, se raje poslužujemo drugega. V tem primeru ima vsak element točno določen stil, katerega definicijo pa lahko spreminjamo v CSS datoteki. Take spremembe ne potrebujejo ponovnega prevajanja kode, ampak samo ponovno naložitev internetne strani v brskalnik.

5.1.2 Praktičen primer uporabe

V našem primeru je programer aplikacije dobil skico uporabniškega vmesnika z določenimi velikostmi in postavitvijo panelov, tipom in barvami črk ter z definicijo osnovnega obnašanja programa.

Ker je uporabniški vmesnik razdeljen na 4 osnovne dele, bo uporabil več panelov, s katerimi bo zagotovil želeno postavitev. Pri tem bo uporabil panel VerticalPanel, ki ima to lastnost, da so njegovi podrejeni elementi naloženi vertikalno eden pod drugega (prvi dodani element je najvišji).

Ker sta panela z video predvajalnikom in opisom (slednji vsebuje labelo, kateri se bo tekst z opisom spreminjal) postavljena eden zraven drugega, bo programer za postavitev teh dveh panelov uporabil HorizontalPanel, katerega lastnost je, da so njegovi podrejeni elementi postavljeni eden zraven drugega (prvi dodani element je na levi strani).

```
HTMLPanel playerPanel = new HTMLPanel("<a style=\"display:block;width:634px;height:372px;\" id=\"player\"> </a>");

HorizontalPanel descPanel = new HorizontalPanel();
descPanel.setStyleName("descPanel");
vodDesc.setStyleName("descTextStyle");
descPanel.add(vodDesc); //dodajanje labele s opisom

HorizontalPanel playerDescPanel = new HorizontalPanel();
playerDescPanel.add(playerPanel); //dodajanje panela s predvajalnikom
playerDescPanel.add(descPanel); //dodajanje panela s opisom
```

Slika 5.1: Kreiranje in postavite panela za predvajalnik ter panela z opisom vsebine

Najnižji panel pa je bolj kompleksen, saj zahteva postavitev puščic in elementov z video vsebinami. Vsem pa mora biti dodana še enostavna logika premikanja in izbiranja elementov. Za ta primer bo uporabil AbsolutePanel, katerega lastnost je, da so njegovi otroci postavljeni absolutno in se lahko prekrivajo. S tem bo dosegel, da se bo srednji del, ki vsebuje video elemente tega panela, premikal levo in desno, puščici pa bosta ostali na svojem mestu.

Video element, ki vsebuje informacijo o posamezni video vsebini, predstavlja osnovni AbsolutePanel, ki združuje gradnik za sliko in labelo za naslov videa. Sliki je dodan lovilec klik dogodkov (*click listener*), ki ob proženju poskrbi, da se naslov in opis video elementa izpišeta na za to pripravljenih labelah na panelih ter da se prične predvajati video vsebina.

Video elemente bo programer zgradil iz podatkov, ki jih je pridobil s strežnika. Ko bo zgradil posamezni video element, mu bo priredil stil, ga dodal na panel, slednjega pa povečal za širino video elementa. Zapomnil pa si bo tudi število video elementov, kar bo uporabil pri premikanju elementov levo in desno.

```
for(final Video show  : scrollItems)
 {
  scrItem = new ScrollItem(this, show.getImageUrl(), show.getName(), show.getDesc(),
     show.getStreamUrl(), loadedIndexRight, show.getId());  //kreiranje video elementa
  scrItem.addStyleName("lowerTabItem");
  scrollPanel.add(scrItem, ClientData.scrollPanelShowW, 0); //dodajanje video elementa

  ClientData.scrollPanelShowW += ClientData.SCROLL_ITEM_WIDTH_ALL;
  getScrollPanel().setWidth(String.valueOf(ClientData.scrollPanelShowW));

  loadedIndexRight++;
 }
```

Slika 5.2: Dodajanje video elementov na panel

Gumba za premikanje panela z video elementi bo zgradil z widgetom Buttonom, kateremu bo priredil ustrezni stil in lovilec klik dogodkov, ki ob proženju pomakne panel z video elementom v ustrezno smer.

```
buttonRight = new Button();
buttonRight.setStyleName("lowerSlideButtonRightStyle");
buttonRight.addClickHandler(new ClickHandler()
  {
    public void onClick(final ClickEvent event)
    {
      scrollSlide(ScrollMoveDirection.LEFT); //akcija ob kliku na gumb
    }
  });
```

Slika 5.3: Dodajanje lovilca klik dogodka

Panel z video elementi bo dodal na nov AbsolutePanel, kateremu bo dodal še gumba, ki predstavljata puščici levo in desno, in tako bo zgradil še najkompleksnejši panel.

```
getBackgroundPanel().add(gumbLevoSh, 0, 0);
getBackgroundPanel().add(scrollPanel, getVideoElementsPanel(), 0);
getBackgroundPanel().add(buttonRight, 928, 0);
```

Slika 5.4: Združitev gumbov s puščicami za levo in desno ter panela z video elementi

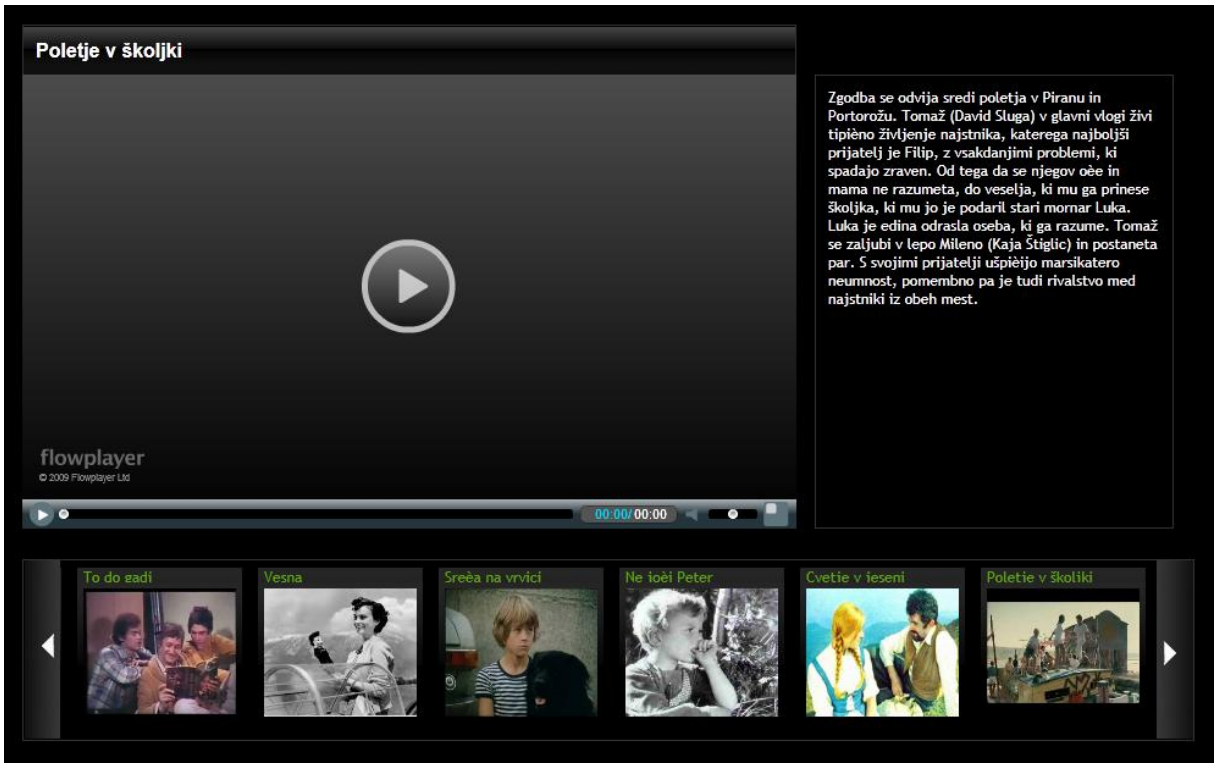
Nato bo vse tri panele (panel z naslovom video vsebine, panel s predvajalnikom in opisom video vsebine ter panel z video elementi) dodal na VerticalPanel enega za drugim, tako da bodo prikazani eden pod drugim. Za panelom z video predvajalnikom bo vrinil še panel za razmejitev. Panel, ki vsebuje vso vsebino bo nato dodal *RootPanelu*, poklical pa bo še JavaScript funkcijo, ki inicializira FlowPlayer.

```
VerticalPanel placeholder = new VerticalPanel();
placeholder.setStyleName("placeholderPanel");

placeholder.add(titlePanel);
placeholder.add(playerDescPanel);
HorizontalPanel delimiter = new HorizontalPanel();
delimiter.setStyleName("delimiterStyle");
placeholder.add(delimiter);
placeholder.add(slider.getBackgroundPanel());
RootPanel.get().add(placeholder, 490, 60);
FlowPlayerControl.init();
```

Slika 5.5: Združitev vseh panelov na skupnem panelu

Ko bo aplikacijo zagnal in se bodo paneli napolnili s tekstom, gradniki za slike pa z dejanskimi slikami, bo izgled aplikacije sledeč:



Slika 5.6: Končni izgled aplikacije zgrajene na SWING način

5.1.3 Prednosti

Največja prednost (mogoče tudi slabost) swing načina grajenja uporabniškega vmesnika je, da ima programer popoln nadzor nad vsemi elementi v uporabniškem vmesniku. Programer določa velikosti objektov, njihovo postavitev ter funkcionalnost. Ker ima programer popoln nadzor, je za kakršno koli spremembo potreben le njegov popravek, ne pa tudi oblikovalčev.

Programer ima pri tem načinu grajenja povsem proste roke pri izbiri posameznih gradnikov, saj mora upoštevati samo končni izgled aplikacije. Tako lahko pri izbiri gradnikov in njihovi postavitvi že upošteva funkcionalnosti, ki jih morajo gradniki zagotavljati. S tem lahko ohrani celotno zasnovo aplikacije zelo odprto za popravljanje oz. razširitve.

5.1.4 Slabosti

Največja slabost swing način grajenja uporabniškega vmesnika je, da je za vsak večji popravek na uporabniškem vmesniku, predvsem na postavitvi objektov, potreben poseg v kodo aplikacije. Po posegu v kodo je potrebno ponovno prevajanje aplikacije in namestitev na strežnik.

To, da ima programer popoln nadzor nad aplikacijo, je lahko prednost ali pa tudi slabost opisanega načina grajenja uporabniškega vmesnika. Predstavljena lastnost se kot slabost pokaže v primeru odsotnosti programerja, saj le-ta praviloma pomeni daljši odzivni čas pri popravkih oz. spremembah aplikacije. Izostanek programerja, ki je postavljal aplikacijo, pomeni vključitev novega programerja v projekt in njegovo študiranje že napisane kode, kar zahteva določen čas, ki pa je lahko veliko daljši od časa, ki ga popravek aplikacije dejansko terja (na primer zamenjava vrstnega reda dveh objektov).

5.2 Način grajenja uporabniškega vmesnika z uporabo GWT UiBinderja

Oblikovalec poda obliko internetne strani kot množico XML datotek, ki določajo postavitev objektov v aplikaciji. Tem XML datotekam pravimo UiBinder predloge, ki jih programer uporabi pri dodajanju funkcionalnosti internetne strani.

5.2.1 Teorija

GWT je v svoji osnovi internetna stran in pri postavljanju internetnih strani se običajno poslužujemo pisanja HTML-ja in CSS-a. Ogrodje UiBinder omogoča prav to – grajenje aplikacij kot HTML strani, ki vključujejo GWT gradnike [22].

V primerjavi z običajnejšim grajenjem uporabniškega vmesnika s pisanjem programske kode je grajenje z UiBinderjem učinkovitejše. Brskalniki so namreč boljši pri grajenju DOM struktur s trpanjem velikih delov HTML v strukturo DOM kot pri grajenju preko programskega vmesnika.

Pri grajenju uporabniškega vmesnika oblikovalec oblikuje uporabniški vmesnik z razvrščanjem gradnikov v predloge. Predloge so XML datoteke, ki jih GWT pri grajenju DOM strukture vzame za osnovo razvrstitve gradnikov po aplikaciji. Nato s programsko kodo napolni gradnike z ustrezno vsebino. Pri ustvarjanju UiBinder predloge se lahko poslužujemo GWT gradnikov (paneli, gumbi, HTML gradnik, labele), lahko pa vključimo tudi svoje gradnike.

UiBinder predloge ponujajo tudi možnost vključitve CSS stilov za gradnike.

```
<!-- PozdravljenSvet.ui.xml -->

<ui:UiBinder xmlns:ui='urn:ui:com.google.gwt.uibinder'>
  <ui:style>
    .pretty { background-color: Skyblue; }
  </ui:style>
  <div class='{style.pretty}'>
    Pozdravljen  <span ui:field='nameSpan'/>.
  </div>
</ui:UiBinder>
```

Slika 5.7: Preprost primer UiBinder predloge

Vsaka UiBinder predloga ima svoj lastniški razred (ang. *owner class*), ki jo poveže s programsko kodo, ima pa tudi funkcije, s katerimi dostopamo do objektov v predlogi, oz. preko katerih napolnimo objekte z vsebino.

```
public class PozdravljenSvet extends UIObject {
  interface MyUiBinder extends UiBinder<DivElement, PozdravljenSvet > {}
  private static MyUiBinder uiBinder = GWT.create(MyUiBinder.class);

  @UiField SpanElement nameSpan;

  public PozdravljenSvet () {
    setElement(uiBinder.createAndBindUi(this));
  }

  public void setName(String name) { nameSpan.setInnerText(name); }
}
```

Slika 5.8: Lastniški razred UiBinder predloge

Po kreiranju instance lastniškega razreda predloge se tega uporablja kot vsak objekt.

```
PozdravljenSvet pozdravljenSvet = new PozdravljenSvet ();
pozdravljenSvet.setName ("svet");
```

Slika 5.9: Uporaba UiBinder predloge in njenega lastniškega razreda

5.2.2 Praktičen primer uporabe

V primeru naše aplikacije je programer dobil UiBinder predloge. Njegova naloga je te predloge ustrezno napolniti z vsebino.

Najprej mora izdelati lastniške razrede za UiBinder predloge. Pri vsaki datoteki s predlogo je potrebno pregledati, katere gradnike vsebuje, in ugotoviti, katera vsebina jim je namenjena. Oblikovalec je programerju dostavil 6 datotek s predlogami, ki se delijo na 3 skupine, in sicer:

- na predlogo za mejnik med elementi v spodnjem gradniku, ki je zelo enostavna. Lastniški razred te predloge vsebuje samo konstruktor.
- Na predlogi za levo in desno puščico, ki sta prav tako enostavni. Poleg konstruktorja vsebujeta še lovilca klik dogodkov ter nekaj logike za menjavanje slike.
- Na tri predloge, ki pa so bolj kompleksne in zahtevajo podrobnejšo razlago.

Predloga gradnika VideoItem, ki bo predstavljal posamezno video vsebino, je sestavljena iz VerticalPanela, ki jo vsebuje VerticalPanel z dvema gradnikoma.

```
<g:VerticalPanel styleName="{style.backpanel}" ui:field="verticalpanel1" >
        <g:VerticalPanel styleName="{style.frontpanel}" ui:field="verticalpanel" >
                <g:Label ui:field="title" styleName="{style.title}"/>
                <g:Image ui:field="image" styleName="{style.imagesmall}" />
        </g:VerticalPanel>
</g:VerticalPanel>
```

Slika 5.10: UiBinder predloga gradnika VideoItem

Razlika med paneloma je v velikosti, kar je razvidno iz različnih CSS stilov, ki so prirejeni gradnikom v predlogi.

```
<ui:style>
        .frontpanel {
                background: url("images/video-bcg.jpg");
                padding-left: 5px;
                padding-right: 5px;
                padding-top: 5px;
                width: 135px;
                height: 120px;
                vertical-align: middle;
                border-spacing: 0;
        }
        .backpanel {
                background-color: black;
                width: 135px;
                height: 159px;
                vertical-align: middle;
                padding-top: 15px;
                border-spacing: 0;
        }
        .title {
                color: #66A612;
                font-weight: bold;
                font-family: "Trebuchet MS",Trebuchet,Verdana,Helvetica,Arial,sans-serif;
                font-size: 12px;
                height: 20px;
                margin-bottom: -4px;
                padding: 0 0 5px 5px;
                width: 113px;
                border-spacing: 0;
        }
        .imagesmall {
                width: 135px;
                height: 105px;
                border-spacing: 0;
        }
</ui:style>
```

Slika 5.11: CSS stil gradnika VideoItem

Lastniški razred te predloge je dokaj enostaven. Ker objekt, skreiran iz te predloge, nima nobene logike, lastniški razred vsebuje le konstruktor, funkcije za branje podatkov razreda ter poslušalca za klik dogodek, ki ob proženju priredi naslov in opis objekta elementom nad in ob predvajalniku.

Predloga gradnika LowerSlide, ki bo vseboval objekte VideoItem, je zelo enostavna, saj vsebuje le 2 gradnika, in sicer HTMLPanel, ta pa vsebuje gradnik FlexTable. Ta gradnik bo

vseboval objekte VideoItem, ki so definirani v prejšnji predlogi. Ker oblikovalcu ni znano število video elementov, le-teh ne more dodati v predlogo. Zato je definiral gradnik, kateremu bodo objekti VideoItem dodani programsko. Funkcija, ki bo izvedla dodajanje, je definirana v lastniškem razredu.

```
<g:HTMLPanel styleName="{style.visibleItems}" >
        <g:FlexTable styleName="{style.allItems}" ui:field='itemsTable' />
</g:HTMLPanel>
```

Slika 5.12: UiBinder predloga gradnika LowerSlide

Lastniški razred pa je bolj kompleksen. Vsebuje namreč logiko prirejanja vsebine gradnikom. Konstruktor ne samo da skreira novo instanco objekta, ampak mu tudi doda obe puščici ter 5 video elementov. Ti so med seboj ločeni z mejnikom. Lastniški razred ima tudi funkcijo, ki ob izvajanju priredi vsebino kreiranim VideoItemom. Funckija se izvede ob kliku na levo ali desno puščico. Takrat se vsebina video elementov pomakne v ustrezno smer, kar je realizirano tako, da se obstoječim elementom priredijo novi podatki (ime, slika in opis).

```
<g:VerticalPanel styleName="{style.placeholder}"  ui:field="placeholder" >
        <g:HorizontalPanel styleName="{style.titlepanel}"  ui:field="titlepanel"  >
                <g:Label styleName="{style.titletext}" ui:field="titletext"/>
        </g:HorizontalPanel>
        <g:HorizontalPanel ui:field="playerDescPanel">
                <g:HTMLPanel styleName="{style.playerpanel}" ui:field="playerpanel">
                        <a style="display:block;width:634px;height:372px;" id="player"> </a>
                </g:HTMLPanel>
                <g:HorizontalPanel styleName="{style.descpanel}" ui:field="descpanel" >
                        <g:Label styleName="{style.desctext}" ui:field="desctext"/>
                </g:HorizontalPanel>
         </g:HorizontalPanel>
         <g:HorizontalPanel styleName="{style.delpanel}" ui:field="delimiter" ></g:HorizontalPanel>
        <f:LowerSlide styleName="{style.sliderpanel}" ui:field="lowerslide"></f:LowerSlide>
 </g:VerticalPanel>
```

Slika 5.13: UiBinder predloga glavnega gradnika aplikacije BackgroundPanel

Predloga gradnika BackgroundPanel je najbolj kompleksna. Vsebuje namreč postavitev panelov, ki vsebujejo labele za prikaz naslova in opisa, predvajalnik ter panel z video elementi.

Lastniški razred tega gradnika je tudi zelo enostaven, saj vsebuje le konstruktor ter metodi za nastavljanje naslova in opisa predvajajoče se vsebine.

Pri tem gradniku lahko opazimo podobnost z zadnjim delom kode v prejšnjem poglavju. V obeh primerih so uporabljeni gradniki HorizontalPanel, ki se dodajajo na VerticalPanel. Tega programer nato pritrdi na RootPanel.

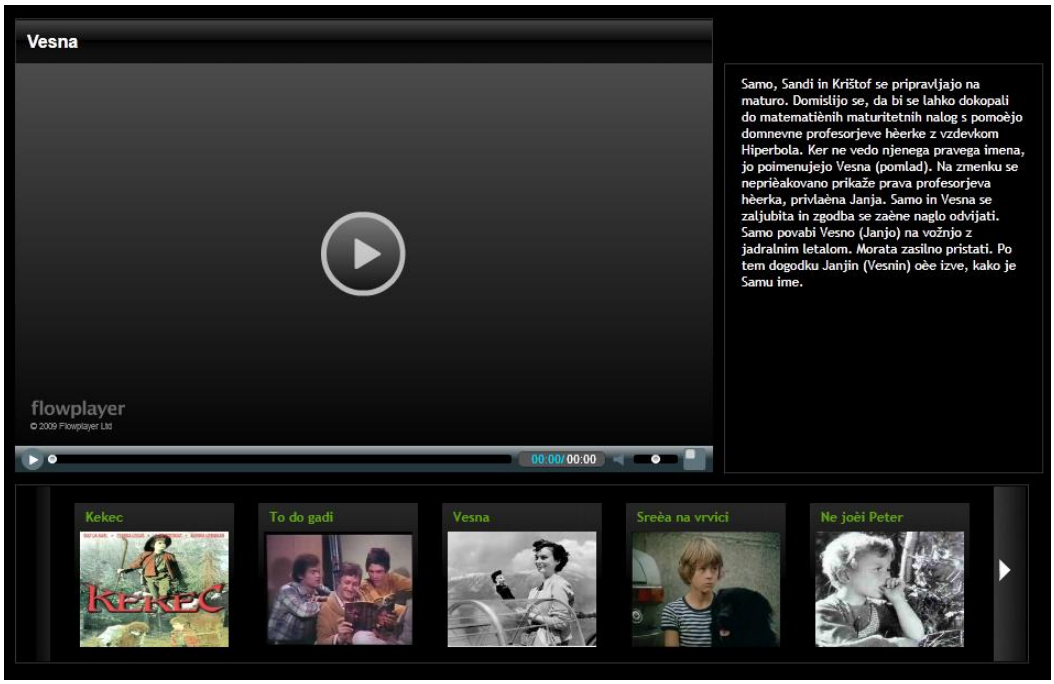
Za zgraditev uporabniškega vmesnika z ogrodjem UiBinder v programu programer skreira instanco razreda BackgroundPanel, kateremu priredi še podatke, ki jih pridobi s strežnika. Zatem pravkar zgrajeni objekt pritrdi na RootPanel in pokliče funkcijo za inicializacijo predvajalnika FlowPlayer.

```
serverService.getVideoContent(new AsyncCallback<List<Video>>()
 {
  @Override
  public void onSuccess(List<Video> result)
  {
   BackgroundPanel panel = new BackgroundPanel();
   panel.getLowerSlide().fillObjectWithData(result);
   RootPanel.get().clear();
   RootPanel.get().add(panel, 490, 60);
   FlowPlayerControl.init();
  }

  @Override
  public void onFailure(Throwable caught)
  {
  }
 });
```

Slika 5.14: Kreiranje in postavitev BackgroundPanela

Aplikacija zgrajena s pomočjo ogrodja UiBinder ima sledeč izgled:



Slika 5.15: Končni izgled aplikacije zgrajene z orodjem UiBinder

5.2.3 Prednosti

Grajenje in vzdrževanje uporabniškega vmesnika z ogrodjem UiBinder je hitro in enostavno. Oblikovalec lahko oblikuje razporeditev in obliko gradnikov brez predhodnih priprav elementov, saj so osnovni gradniki, katere podpira GWT, že na voljo. Pri tem se lahko poslužuje kopiranja iz drugih projektov, saj je vsaka predloga lahko tudi zaključena celota in kot taka je neodvisna od drugih predlog. UiBinder omogoča postopen prehod v razvoj

končnega HTML, hkrati pa omogoča izdelavo realnega in interaktivnega uporabniškega vmesnika.

Med razvojem programsko okolje Eclipse programerju sproti izvaja preverjanje sklicevanja na gradnike iz predloge v lastniški razred in obratno. Ponuja tudi podporo za internacionalizacijo aplikacije in omogoča učinkovito rabo virov brskalnika, zato spodbuja uporabo lahkih HTML elementov pred uporabo težkih gradnikov in panelov.

Uporaba ogrodja UiBinder omogoča lažje sodelovanje z oblikovalci UI, ki uporabljajo tehnologije XML, HTML in CSS, ne uporabljajo pa Jave. Ta lastnost je zelo pomembna, saj se programer lahko osredotoči izključno na pridobivanje in posredovanje vsebine uporabniškemu vmesniku. Oblikovalec pa lahko oblikuje dokončni izgled aplikacije, saj programer ne spreminja postavitve in oblike gradnikov.

Uporaba UiBinderja predstavlja razvoj po načinu Model-View-Presenter (MVP), kjer je osnova delovanja ločitev prikazanega dela aplikacije (XML predloge) od njenega programskega dela (pridobivanje podatkov iz različnih virov, posredovanje podatkov na odjemalca, logika na odjemalcu - Java razredi).

5.2.4 Slabosti

Ogrodje UiBinder ima veliko dobrih lastnosti, ima pa tudi nekaj pomanjkljivosti.

UiBinder sam po sebi ni prikazovalnik. Ne podpira zank, vejitev in pogojnih stavkov. Omogoča postavitev gradnikov, njihova funkcionalnost pa je še vedno odvisna od logike v njihovih lastniških razredih.

Ena izmed pomanjkljivosti UiBinder predlog je, da mora programer sprogramirati lastniške razrede, kar v primeru velikega števila predlog ne smemo zanemariti.

Pri programiranju funkcionalnosti oz. pri polnjenju podatkov se mora programer posvetovati z oblikovalcem.

5.3 Način grajenja uporabniškega vmesnika z uporabo GWT in zavijanja objektov

Oblikovalec poda obliko internetne aplikacije v HTML datoteki, ki je opremljena s CSS stili. Programer opremi HTML elemente z identifikatorji, preko katerih dostopa do objektov, ko jim prireja vsebino.

5.3.1 Teorija

Kot smo spoznali, GWT omogoča grajenje uporabniškega vmesnika ali s postavljanjem objektov ali preko predloge, ki se jo opremi z vsebino. Pri obeh načinih GWT gradi nove objekte – kličejo se konstruktorji objektov.

Pri metodi zavijanja objektov (zaradi povezave z imenom funkcij, ki izvajajo zavijanje objektov, se ta metoda imenuje tudi wrap metoda) temu ni tako. Oblikovalec v tem primeru dostavi programerju HTML datoteke, v katerih je definirana oblika aplikacije. Tak HTML je dokončen, vsebuje namreč tudi CSS stile, tako da aplikacijo lahko pogledamo v brskalniku, vendar je brez vsebine. Vidijo se obrisi tabel, gumbi in vnosna polja.

Programer elemente v HTML datotekah opremi z unikatnimi identifikatorji, preko katerih v GWT kodi dostopa do njih. Pri tem mora paziti, da so identifikatorji res unikatni, saj v nasprotnem primeru pride do napake pri izvajanju.

Ko programer elemente iz HTML-ja zavije v GWT elemente, lahko z njimi operira, kot bi bili samo GWT elementi. Lahko jim prireja vsebine, jim spreminja stile (čeprav to ni smiselno) in doda funkcionalnosti [23].

GWT omogoča zavijanje label, seznamov, slik, panelov, navadnih gumbov, gumbov za označevanje, gumbov za izbiranje ter tekstovnih polj. V HTML se ti elementi med seboj ločijo po oznakah, mnogi pa imajo isto oznako in se ločijo po tipu elementa (npr. vnosno polje za geslo in okenček za označevanje imata oba HTML oznako input, ločita pa se po tipu vnosnega polja, ki je pri prvem password, pri drugem pa checkbox).

Primer zavijanja labele:

```
<div id="testLabel" class="right-placeholder-text"></div>
```

Slika 5.16: Primer HTML kode labele

```
Label label = Label.wrap(Document.get().getElementById("testLabel"));
label.setText("Testna labela");
```

Slika 5.17: Primer GWT kode, ki uporabi HTML labelo

Primer zavijanja gumba. Gumbu po zavijanju dodamo poslušalca za klikanje.

```
<button id='testButton'></button>
```

Slika 5.18: Primer HTML kode gumba

```
final Button button = Button.wrap(Document.get().getElementById("testButton"));

button.setText("Click me");
button.addClickHandler(new ClickHandler(){

 @Override
 public void onClick(ClickEvent event)
 {
  Window.alert("Kliknili smo gumb "+button.getText());
 }
});
```

Slika 5.19: Primer GWT kode, ki uporabi HTML gumb

5.3.2 Praktičen primer uporabe

V našem primeru aplikacije je programer dobil HTML datoteko, ki vsebuje tudi CSS stile elementov. Ker se pri aplikaciji prikazuje samo ena maska, je HTML preprost. Vsebuje elementa za naslov in opis predvajajočega se videa, prostor za predvajalnik ter seznam, ki vsebuje video elemente.

```
<div id="mainHtmlDiv" class="wrapper">
        <div class="upperclear"></div>
            <div style="border-left:1px solid #333333;border-bottom:1px solid #333333; margin-
            bottom:1px;" class="video-placeholder">
                <div  class="title-text"><div id="playingItemTitle" style="margin-left:10px;padding-top:10px;">test</div></div>
                        <div class="right-placeholder">
                        <div id="playingItemDesc" class="right-placeholder-text"></div>
                </div>
                <div class="delimiterDesc"></div>
                <div style="margin-left:2px;">
                <a style="display:block;width:631px;height:369px; margin-bottom:1px;" id="player"> </a>
                </div>
        </div>
        <div class="middleclear"></div>
        <div class="lowerSliderItems">
                <ul class="lista">
                        <li><img id="itemArrowLeft" class="itemArrowLeft"></img></li>
                        <li><div class="itemBackgroundDelimiter"></div></li>
                        <li>
                          <div class="itemBackground">
                            <div class="itemBackgroundInner">
                                <div id="item1Title" class="itemTitle"></div>
                                  <div class="itemBody">
                                        <img id="item1Img" src="" width="135" height="105" alt="Video" />
                                  </div>
                                </div>
                          </div>
            </li>
                        <li><div class="itemBackgroundDelimiter"></div></li>
                        <!-- se ponovi še 4x -->
                        <li><img id="itemArrowRight" class="itemArrowRight"></img></li>
                  </ul>
        </div>
```

Slika 5.20: HTML, ki definira izgled spletne strani

HTML je opremljen s CSS stili, ki so v glavi dokumenta. Kot je razvidno iz HTML predloge, so vsi elementi, ki so nosilci vsebine, opremljeni z identifikatorji. Elementi, katerim vsebina ne bo prirejena, teh identifikatorjev nimajo. Pri seznamu video elementov imajo elementi oznake tudi oštevilčene.

HTML koda se v projektu nahaja v osnovni HTML datoteki GWT projekta. HTML koda, ki je napisana v tej datoteki, se prikazuje ves čas v ozadju brskalnika. Zato mora biti HTML koda razdeljena na module, ki jih na začetku njenega izvajanja na brskalniku odstranimo in jih po potrebi dodajamo. To storimo tako, da najvišji element kode, ki vsebuje elemente, ki se bodo zavijali, zavijemo v GWT objekt *Element*, katerega nato odstranimo iz DOM strukture.

Ko potrebujemo posamezni modul HTML kode, le-tega dodamo nazaj na RootPanel v programu.

```
//odstranitev HTML kode iz DOM strukture
html_page = (Element)Document.get().getElementById("mainHtmlDiv");
DOM.removeChild(RootPanel.getBodyElement(), html_page);


//postavitev HTMl kode nazaj v DOM strukturo
RootPanel.get().clear();
DOM.appendChild(RootPanel.getBodyElement(), Diploma.html_page);
```

Slika 5.21: Odstranitev ter postavitev HTML elementa iz DOM strukture

Ko so elementi vrnjeni v DOM-u, lahko do njih programsko dostopamo. GWT koda, ki zavije HTML elemente v GWT objekte, je enostavna. Poleg zavijanja elementov koda doda lovilce klik dogodkov na slike video elementov. Koda vsebuje tudi nekaj logike, ki pri pomikanju seznama levo in desno priredi ustrezno vsebino elementom v seznamu video elementov.

```
titleLabel = Label.wrap(Document.get().getElementById("playingItemTitle"));
descLabel = Label.wrap(Document.get().getElementById("playingItemDesc"));
arrowRight = Image.wrap(Document.get().getElementById("itemArrowRight"));
arrowLeft = Image.wrap(Document.get().getElementById("itemArrowLeft"));

item1Title = Label.wrap(Document.get().getElementById("item1Title"));
item2Title = Label.wrap(Document.get().getElementById("item2Title"));
item3Title = Label.wrap(Document.get().getElementById("item3Title"));
item4Title = Label.wrap(Document.get().getElementById("item4Title"));
item5Title = Label.wrap(Document.get().getElementById("item5Title"));

item1Img = Image.wrap(Document.get().getElementById("item1Img"));
item1Img.addClickHandler(new ClickHandler()
{
  public void onClick(final ClickEvent event)
  {
    fillHeaderText(0);
  }
});
```

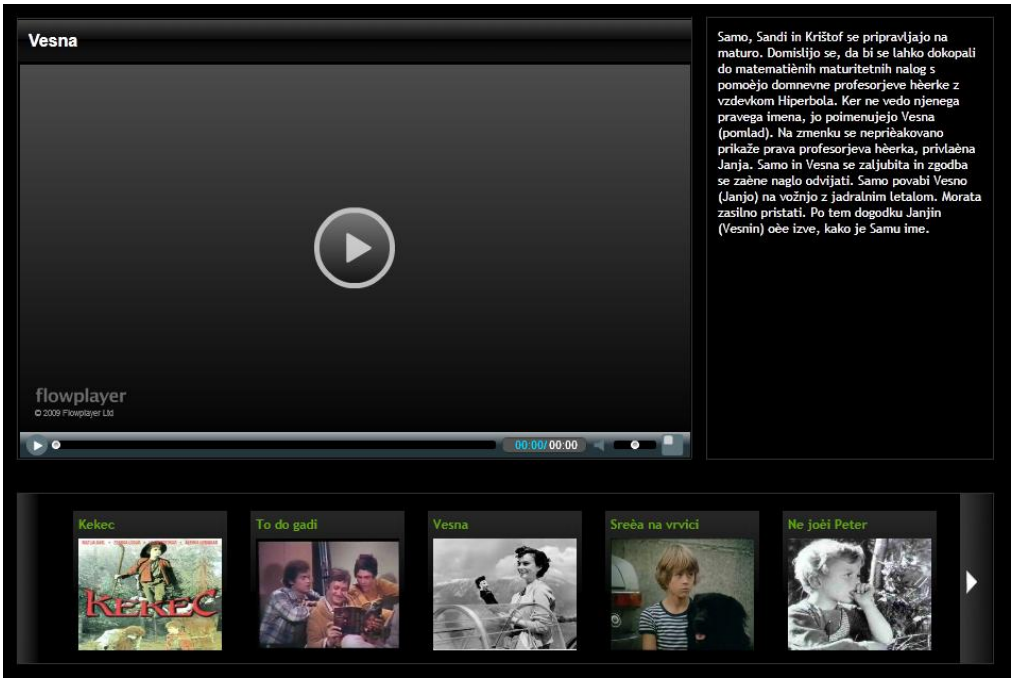
Slika 5.22: GWT koda katera zajame HTML elemente

Ko uporabnik premika seznam levo in desno, se vsebina prireja ustreznim labelam in slikam.

```
item1Title.setText(videos.get(leftIndex).getName());
item1Img.setUrl(videos.get(leftIndex).getImageUrl());
```

Slika 5.23: Prirejanje vrednosti HTML elementom

Aplikacija, zgrajena z metodo zavijanja, ima sledeč izgled:



Slika 5.24: Končni izgled aplikacije, zgrajene z metodo zavijanja

5.3.3 Prednosti

Največja prednost grajenja uporabniškega vmesnika z metodo zavijanja je v tem, da od oblikovalca zahteva izključno znanje HTML-ja in CSS-a. Oblikovalec se zaradi tega lahko osredotoči izključno na izgled aplikacije in se mu ni potrebno truditi s sintakso XML predlog kot v primeru grajenja uporabniškega vmesnika z UiBinder metodo.

Velika prednost te tehnologije je tudi v tem, da oblikovalec lahko sproti preverja izgled aplikacije. Ker za predogled ni potrebno prevajanje in izvajanje nobene kode, ki bi dokončno zgradila aplikacijo, je predogled izgleda možen v vsakem trenutku izdelave oblike strani. Celoten proces grajenja uporabniškega vmesnika je zaradi te lastnosti tudi enostaven. Pri usklajevanju funkcionalnosti aplikacije med oblikovalcem in programerjem to lahko poteka ob gledanju predogleda aplikacije. Zaradi tega prihaja do manj napačnega razumevanja, saj lahko oblikovalec točno uskladi funkcionalnost elementov aplikacije s programerjem.

Stran (še posebno enostavne strani), postavljena s tehnologijo zavijanja, je tudi zelo hitra. Ker se za postavitev strani uporabijo HTML elementi, se ti pokažejo hitro, saj jim ni potrebno izvajati konstruktorjev, ki objekte zgradijo in jih pretvorijo v HTML kodo.

Internetne strani oz. aplikacije, zgrajene z metodo zavijanja, prav tako predstavljajo razvoj po načinu Model-View-Presenter (MVP).

5.3.4 Slabosti

V našem primeru aplikacije je začetni HTML, ki vsebuje njene elemente, enostaven. V primeru, da ima aplikacija veliko različnih mask, postane HTML datoteka velika in s tem zelo nepregledna. Ker je potrebno na začetku izvajanja kode na odjemalcu odstraniti elemente iz

DOM strukture, to pomeni veliko odstranjevanja elementov, kar pa lahko povzroči malo daljši čas postavljanja začetne maske. Ko ima programer opravka z veliko maskami, ki se prikazujejo uporabniku, mora poskrbeti za dober sistem odstranjevanja in dodajanja delov HTML kode.

Ena izmed slabosti metode zavijanja je tudi ta, da mora programer HTML elemente opremiti z identifikatorji, kar pomeni, da se ne more izogniti posegu v HTML. To operacijo mora narediti programer in ne oblikovalec, saj mora vedeti, kako so označeni elementi, katerim priredi vsebino.

Metoda zavijanja ima tudi to slabost, da so vsi elementi v pomnilniku, dokler se ne zapre okno brskalnika. Ta lastnost se pojavi zaradi avtomatičnega klica funkcije RootPanel.detachOnWindowClose(), ki se izvede ob zavijanju vsakega elementa posebej. Zaradi tega ostane element v pomnilniku tudi, ko se ga že dolgo ne uporablja več, kar lahko vodi v veliko izgubo pomnilnika.

5.4 Način grajenja uporabniškega vmesnika z uporabo JavaScript knjižnice jQuery

Podobno kot pri grajenju uporabniškega vmesnika z metodo zavijanja tudi pri tem načinu gradnje uporabniškega vmesnika oblikovalec poda obliko internetne aplikacije v HTML datoteki, ki je opremljena s CSS stili. Programer opremi HTML elemente z identifikatorji, preko katerih dostopa do objektov, ko jim prireja vsebino.

5.4.1 Teorija

Grajenje uporabniškega vmesnika z JavaScriptom je podobno kot pri načinu, predstavljenim v prejšnjem poglavju. Tehnologiji grajenja se razlikujeta v tem, da pri grajenju uporabniškega vmesnika z JavaScriptom na odjemalcu ne teče GWT koda, ampak so uporabljene že napisane JavaScript knjižnice (v našem primeru knjižnica jQuery). Uporabimo jih za pridobivanje podatkov in za dodajanje efektov elementom v aplikaciji.

Programer v tem primeru HTML datoteki doda JavaScript knjižnice, preko katerih bo izvedel AJAX klice in s katerimi se bodo dodali posebni efekti elementom.

5.4.2 Praktičen primer uporabe

Programer pridobi HTML datoteko, s katero je oblikovalec aplikacije določil njen izgled. Na odjemalcu ne teče GWT koda, zaradi tega pridobivanje podatkov s strežnika ni tako enostavno. Podatke bo v takem primeru pridobival z AJAX klici na strežnik. V prejšnjih primerih je na odjemalca s strežnika vedno dobil Java razrede z vsebino. V tem primeru pa ima na odjemalcu JavaScript, ki ne izvira iz Java kode. Način prenosa podatkov s strežnika je zato drugačen. Podatki se prenašajo v JSON formatu, ki ga zgradi na strežniku.

Na strežniku AJAX klice odjemalca postreže poseben servlet, ki se razlikuje od servletov, ki jih je uporabili v prejšnjih primerih. Če so imeli prejšnji servleti vmesnik, s katerimi je lahko odjemalec dostopal do poljubnih funkcij servleta, je programer v tem primeru omejen, saj

lahko dostopa do goGet in do doPost funkcij servleta. Ker bo podatke bral s strežnika, bo v tem primeru klical goGet funkcijo. Funkcija bo pridobila podatke z istega vira kot servleti GWT RPC, na odjemalca pa jih bo vrnila v JSON formatu – programer bo vrnil seznam JSON objektov.

Koda na strežniku ustrezno nastavi tip odziva in vrne podatke o video vsebinah.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{
 JSONArray videoItems = new JSONArray();
 response.setContentType("application/json");
 getVideItems(videoItems );
 PrintWriter out = response.getWriter();
 out.print(videoItems);
 out.flush();
 out.close();
}
```

Slika 5.25: Java koda servleta za strežbo podatkov o video vsebinah

Na odjemalski strani je HTML skoraj enak kot v prejšnjem primeru, saj je izgled isti. HTML datoteka, ki je v tem primeru druga kot osnovna HTML datoteka projekta, nima definiranih video elementov. Video elemente za prikaz video vsebine, ki jo bo programer pridobili s strežnika, bi HTML datoteka lahko vsebovala, vendar jih bo skreiral z uporabo JavaScripa. Ko bodo skreirani, bo poklical JavaScript funkcijo, ki bo dodala efekt drsenja elementov. Tako se bodo elementi ob pomikanju levo/desno zapeljali, kar bo lepše kot koračno premikanje v prejšnjih primerih.

JavaScript koda za AJAX klic na strežnik in za kreiranje video elementov.

```
function createShowItem(vodItem, indeks){
    return "<div onClick=\"showTitleAndDesc("+indeks+");\" class=\"video-item\"><div class=\"video-item-
    time\">"+vodItem.name+"</div><img src=\""+vodItem.image+"\" width=\"135\" height=\"105\" alt=\"Video\" /> </div>";

}
function getLowerSlideContent(){
        $.ajax({
                url : 'horizontalSlideServlet', type : 'GET', timeout : 1000,dataType : "json",
                error : function(){
                        alert("Napaka pri pridobivanju podatkov!");
                },
                success : function(jsonresults){
                        resultServer = jsonresults;
                        var results = $('#mycarousel');
                        results.empty();
                        for (indeks = 0; indeks < jsonresults.length; indeks++){

                                $("<li jcarouselindex=\"1\" class=\"jcarousel-item\">" ).append(
                                createShowItem(jsonresults[indeks],
                                indeks)).appendTo(results);
                        }
                        initLowerSlider();
                }
        });
}
```

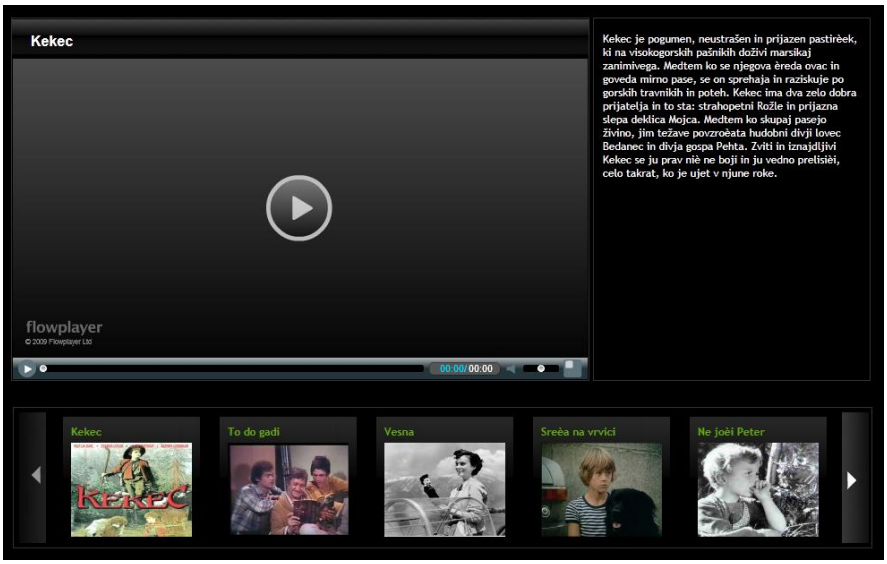
Slika 5.26: AJAX klic z uporabo jQuery knjižnice

Video elementi se dodajo HTML elementu z identifikatorjem mycarousel, katerega programer enostavno vključi v JavaScript kodo z uporabo knjižnice jQuery. jQuery knjižnica se uporabi tudi pri dodajanju video elementov v seznam.
Po končanem dodajanju pokliče še funkcijo, ki doda efekt zapeljave elementov.

```
function initLowerSlider()
{
  jQuery('#mycarousel').jcarousel();
  scroll: 1
}
```

Slika 5.27: Kreiranje efekta zapeljave video elementov

Uporabniški vmesnik, zgrajen z uporabo JavaScript jQuery knjižnice, ima naslednji izgled:



Slika 5.28: Končni izgled aplikacije, zgrajene z uporabo knjižnice jQuery

5.4.3 Prednosti

Ker sta si metoda zavijanja in tehnologija z uporabo knjižnice jQuery zelo podobni, veljajo pri uporabi tehnologije jQuery iste prednosti.

Prednost (kot bomo videli kasneje, tudi slabost) te metode je uporaba JavaScripta. Brskalniki izvajajo malo JavaScripta zelo hitro, kar v kombinaciji s prenosom podatkov v JSON obliki naredi odjemalca hitrega.

Pri razvoju v JavaScriptu se lahko poslužujemo veliko različnih knjižnic, ki nam olajšajo delo, dodajo posebne efekte na elemente in omogočijo lažje klice na strežnik. Knjižnic, ki bazirajo na knjižnici jQuery, je zelo veliko, velika večina pa jih je namenjena enostavnejšemu dostopanju do DOM elementov in dodajanju posebnih efektov, ki naredijo internetne aplikacije zelo lične. Uporaba teh knjižnic je v GWT težja oz. pri uporabi teh knjižnic moramo prehajati iz Java kode v JavaScript.

5.4.4 Slabosti

Programer se tudi v tem primeru ne izogne posegu v HTML oblikovalca. Pogled opremljanja elementov z identifikatorji mora dodati ukaze, ki vključijo JavaScript knjižnice in CSS stile, ki pripadajo tem knjižnicam.

Največja slabost grajenja uporabniškega vmesnika z uporabo te metode je programiranje v JavaScriptu in ne v orodju GWT, kjer proramer programira v Javi. Zaradi te lastnosti je tudi razhroščevanje težje, saj GWT omogoča dodajanje breakpointov in opazovanje vrednosti spremenljivk v razvojnem okolju in ne v brskalniku, kot to dopušča programiranje v JavaScriptu.

6 UGOTOVITVE

Med analiziranjem različnih metod za izdelavo uporabniškega vmesnika smo ugotovili, da ima vsaka svoje prednosti in slabosti. Te ugotovitve so predstavljene v spodnji tabeli:

	GWT - swing	GWT - UiBinder	GWT - wrap	JavaScript jQuery
Pregled oblike pred razvojem funkcionalnosti	✗	✗	✓	✓
Enostavno spreminjanje oblike med razvojem	✗	✓	✓	✓
Uporaba hitrih elementov brskalnika	✗	✓	✓	✓
Enostavna podpora različnim brskalnikom	✓	✓	✓	✗
Programiranje v Javi na strežniku in odjemalcu	✓	✓	✓	✗
Prikaz začetne maske brez zakasnitev	✓	✓	✗	✓

Slika 6.1: Tabela z lastnostmi posameznih tehnik grajenja uporabniškega vmesnika

Iz analize sledi, da je najboljši način grajenja uporabniškega vmesnika metoda zavijanja. Oblikovalec mora obvladati HTML in CSS tehnologijo, kar pa se od oblikovalcev v današnjem času sme realno pričakovati. Že pred začetkom programerskega dela je možen predogled strani, kar omogoča, da se programer in oblikovalec pogovorita o posameznih funkcionalnostih strani, med razvojem strani pa ima oblikovalec možnost njenega pregleda in enostavnega popravljanja. Ko programer dobi oblikovalčeve datoteke, jih opremi z identifikatorji, nato pa sprogramira logiko za pridobivanje podatkov ter za menjavanje delov HTML-ja v aplikaciji.

Aplikacijo, razvito z uporabo metode zavijanja, si je možno ogledovati že med razvojem. Med razvijanjem je tako možno dokaj enostavno spreminjati obliko, saj lahko oblikovalec spreminja HTML in CSS datoteke, ne da bi pri tem motil delo programerja.

Programer aplikacijo razvija z orodjem GWT oz. v Javi tako na strežniškem kot na njenem odjemalskem delu. Tako se lahko poslužuje *development mode* načina izvajanja aplikacije, kar omogoča enostavno in hitro razhroščevanje. Zaradi možnosti prehajanja iz Jave v JavaScript na odjemalcu programer ni prikrajšan za vrsto funkcij, ki jih ponujajo različne JavaScript knjižnice.

Internetna stran, zgrajena z metodo zavijanja, je tudi najhitrejša. V povprečju se stran, zgrajena z metodo zavijanja, naloži enkrat hitreje kot internetne strani, zgrajene s swing načinom gradnje in z uporabo UiBinderja (razmerje je 20 ms : 40 ms). Stran, zgrajena z uporabo knjižnice jQuery, se v povprečju nalaga okrog 70 ms, kar je najpočasneje. To pripisujemo dejstvu, da se pri metodi zavijanja HTML datoteka v brskalniku zelo hitro naloži. Da je izvajanje strani, zgrajene z orodjem GWT, najhitrejše, pa je posledica izvajanja zelo optimiziranega JavaScripta, za kar poskrbi prevajalnik iz Jave v JavaScript. Pri tem moramo

upoštevati, da smo pri prevajanju programa prevajalniku naložili nalogo, da optimizira JavaScript za vsak posamezni brskalnik. Pri celotni analizi moramo upoštevati tudi dejstvo, da je HTML datoteka pri metodi zavijanja majhna, oz. da je HTML enostaven. V primeru večje HTML datoteke, bi se le-ta naložila počasneje, saj bi na začetku izvajanja programa morali odstraniti HTML module, ki se ne prikazujejo na prvi strani.

7 SKLEP

V tej diplomski nalogi sem si zadal cilj odgovoriti na dve vprašanji, in sicer na kakšen način naj bo zgrajen uporabniški vmesnik spletne aplikacije (tehnologija) ter kako bo dostavljen izgled aplikacije (design) od oblikovalca do programerja.

Za odgovor na ti dve vprašanji sem uporabil programersko okolje GWT, ki ponuja več različnih načinov razvoja internetnih strani. Z uporabo orodja GWT sem izdelal uporabniški vmesnik internetne strani na 4 različne načine:

- način grajenja uporabniškega vmesnika na podlagi slikovne oblike z uporabo GWT objektov,
- način grajenja uporabniškega vmesnika z metodo GWT UIBinder,
- način grajenja uporabniškega vmesnika iz HTML predloge z uporabo GWT,
- način grajenja uporabniškega vmesnika iz HTML predloge z uporabo JavaScript knjižnice jQuery.

Med razvojem programa in kasnejšo analizo tehnologij sem prišel do ugotovitve, da je za razvoj programa najustreznejši način grajenja uporabniškega vmesnika iz HTML predloge z uporabo GWT oz. z metodo zavijanja.

Pri tej tehnologiji se programer lahko osredotoči samo na razvoj pridobivanja podatkov oz. dodajanja funkcionalnosti programa, saj za obliko skrbi oblikovalec, ki razvije HTML datoteke, ki jih opremi s CSS stilom. Pri samem razvoju aplikacije se programer lahko poslužuje razvijanja v Javi tako na strežniškem kot tudi na odjemalskem delu aplikacije, oblikovalec pa lahko spremlja aplikacijo že med samim razvojem in jo po potrebi tudi sproti popravlja.

8 LITERATURA

[1] Elizabeth Castro, HTML for the world wide web, Fifth Edition, Peachpit Press, Berkley CA, 94710 2003 ISBN: 0-321-13007-3

[2] HTML, vir: http://www.hypergurl.com/whatishtml.html, dostopno maja 2011

[3] David Sawyer McFarland, CSS: The Missing Manual, O'Reilly Media, Inc. ©2009 ISBN:0596802447 9780596802448

[4] San Murugesan, Understanding Web 2.0, IT Pro July, August 2007

[5] James Gosling, The Java language specification, Sun Microsystems Inc. 901 San Antonio road, Mountain View, California 94303 U.S.A, 2000

[6] JVM, vir http://searchsoa.techtarget.com/definition/Java-virtual-machine, dostopno maja 2011

[7] JavaScript, vir http://javascript.about.com/od/reference/p/javascript.htm, dostopno maja 2011

[8] JavaScript, vir http://www.boutell.com/newfaq/definitions/javascript.html, dostopno maja 2011

[9] DOM, vir http://en.wikipedia.org/wiki/Document_Object_Model,

[10] AJAX, vir http://www.robertspahr.com/teaching/nmp/ajax_web_applications.pdf, dostopno septembra 2011

[11] JSON, vir http://www.json.org/, dostopno maja 2011

[12] Servlet, vir http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets2.html, dostopno maja 2011

[13] GWT dokumentacija, http://code.google.com/intl/sl-SI/webtoolkit/doc/latest/DevGuide.html, dostopno maja 2011

[14] GWT povezovanje JavaScripta z Javo, http://googlewebtoolkit.blogspot.com/2008/07/getting-to-really-know-gwt-part-1-jsni.html, dostopno maja 2011

[15] komunikacija med strežnikom in odjemalcem pri GWT, http://code.google.com/intl/sl-SI/webtoolkit/doc/1.6/DevGuideServerCommunication.html, dostopno maja 2011

[16] GWT prevajanje permutacij, vir http://stackoverflow.com/questions/3984624/what-does-compiling-permutation-mean-in-gwt, dostopno maja 2011

[17] GWT razhroščevanje, vir http://code.google.com/intl/sl-SI/webtoolkit/doc/latest/DevGuideCompilingAndDebugging.html, dostopno maja 2011

[18] jQuery, vir http://en.wikipedia.org/wiki/JQuery, dostopno maja 2011

[19] FlowPlayer, vir http://flowplayer.org/documentation/index.html, dostopno septembra 2011

[20] GWT paneli, vir http://code.google.com/intl/sl-SI/webtoolkit/doc/latest/DevGuideUiPanels.html, dostopno maja 2011

[21] GWT gradniki, vir http://code.google.com/intl/sl-SI/webtoolkit/doc/latest/DevGuideUiWidgets.html, dostopno maja 2011

[22] GWT UI binder, vir http://code.google.com/intl/sl-SI/webtoolkit/doc/latest/DevGuideUiBinder.html, dostopno maja 2011

[23] primer uporabe wrap tehnologije, vir http://code.google.com/p/google-web-toolkit/source/browse/trunk/user/test/com/google/gwt/user/client/ui/ElementWrappingTest.java?r=3650, dostopno maja 2011

9 PRILOGE

CD s programsko kodo.