SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1811

KORISNIČKO SUČELJE ZA POTROŠAČU PRILAGOĐENO PROGRAMIRANJE NAD SKUPOM UDOMLJENIKA

Matija Horvat

Zagreb, rujan 2009.

Ovu zahvalu upućujem svima koji su mi velikodušno pomagali tijekom izrade diplomskog rada. Veliko hvala upućujem prof. dr. sc. Siniši Srbljiću na ukazanom povjerenju, pruženoj potpori i prilici za rad na vrlo zanimljivom projektu. Posebno se zahvaljujem mr. sc. Dejanu Škvorcu na izuzetnom strpljenju, neprocjenjivoj pomoći i korisnim sugestijama pri pisanju diplomskog rada.

Zahvaljujem obitelji na razumijevanju, potpori i pomoći koju su mi pružili tijekom studiranja.

1. Uvod

Korisnici web stranica sve češće se aktivno uključuju u prilagodbu, odnosno poosobljenje web stranica. Razlog tome je rastuća jednostavnost stvaranja novog sadržaja, potreba za kreativnim izražavanjem i dijeljenjem sadržaja s ostalim korisnicima. Mogućnost poosobljivanja web stranica korisnicima pruža gotovo neograničene načine utjecaja na oblik, sadržaj i primjensku funkciju web stranica. Mogućnošću utjecaja na izgled i funkcionalnost web stranice korisnik prelazi iz stanja pasivnog potrošača izloženog sadržaja u stanje aktivnog sudionika u postupku stvaranja i ponude sadržaja. Raspon korisnikovog utjecaja na izgled i funkcionalnost web stranice proteže se od jednostavnih oblika prilagodbe veličine i boje elemenata web stranice, preko udruživanja sadržaja s različitih web stranica u objedinjenu web stranicu (engl. *mashup*), do stvaranja složenog primjenskog sustava (engl. *application*) izgrađenog od jednog ili više jednostavnijih primjenskih sustava povezanih u cjelinu unutar udomiteljske web stranice.

Razred primjenskih programa za web koji omogućava visok stupanj prilagodbe zahtjevima korisnika su primjenski programi zasnovani na udomljenicima (engl. *widgets*, *gadgets*) [6]. Udomljenici su jednostavni primjenski programi koji obavljaju nedjeljive primjenske funkcionalnosti, kao što je praćenje prognoze vremena, praćenje stanja na burzi, pretvorba valuta i slično. Primjenom skupa nezavisnih udomljenika, korisnik gradi poosobljenu radnu okolinu za obavljanje vlastitih radnih zadataka. Primjenski program zasnovan na udomljenicima sastoji se od korisnički odabranog skupa udomljenika koji se izvode unutar udomiteljske stranice.

Tijekom postupka poosobljavanja udomiteljske stranice korisnik primjenjuje udomljenike dostupne na javnim poslužiteljima. U nedostatku odgovarajućih javno izloženih udomljenika, potrebno je izgraditi novi udomljenik koji je izgledom i funkcionalnošću prilagođen zahtjevima korisnika. Međutim, samostalna gradnja vlastitih primjenskih programa u obliku udomljenika od korisnika zahtijeva poznavanje programskih jezika i paradigmi koje se koriste u području programskog inženjerstva. S obzirom na to da su krajnji korisnici, odnosno potrošači primjenskih programa vrlo rijetko upoznati s programskim jezicima i pripadajućim tehnikama programiranja, razvoj primjenskih programa ograničen je na uski krug školovanih programera. Približavanje razvoja primjenskih programa korisnicima koji nisu školovani u području programskog

inženjerstva nastoji se postići oblikovanjem programskih paradigmi posebno prilagođenih za pojedine grane ljudske djelatnosti (engl. *end-user development*) [8]. S druge strane, za izgradnju primjenskih programa opće namjene, ali posebno prilagođenih vlastitim potrebama i željama pojedinaca, pokrenuto je istraživanje u području programiranja prilagođenog potrošaču (engl. *consumer programming*) [4]. Programiranje prilagođeno potrošaču namijenjeno je širokom krugu korisnika računala koji ovladavaju osnovnim znanjima o uporabi primjenskih programa putem grafičkog korisničkog sučelja (engl. *graphical user interface*) [7].

U okviru ovog diplomskog rada programski je ostvareno grafičko korisničko sučelje za potrošaču prilagođeno programiranje nad skupom udomljenika. Primjenski programi grade se povezivanjem skupa udomljenika u radni tijek (engl. *workflow*) izvođenjem radnji nad pripadajućim elementima grafičkog korisničkog sučelja. Grafičko korisničko sučelje za potrošaču prilagođeno programiranje pojavljuje se u obliku plivajućeg izbornika (engl. *pop-up menu*) za obilazak udomljenika i definiranje radnji nad elementima grafičkog korisničkog sučelja udomljenika.

U radu su obrađene teme vezane uz korisničku prilagodbu web stranica, primjenske sustave zasnovane na udomljenicima te arhitekturu i implementaciju grafičkog korisničkog sučelja za potrošaču prilagođeno programiranje nad skupom udomljenika. U drugom poglavlju opisane su značajke i način korištenja alata *GreaseMonkey*. U trećem poglavlju opisana je arhitektura primjenskih sustava zasnovanih na udomljenicima. Objašnjen je način izrade i korištenja udomljenika udomiteljske stranice *iGoogle*. U četvrtom poglavlju opisana je arhitektura korisničkog sučelja za potrošaču prilagođeno programiranje nad skupom udomljenika. U petom poglavlju opisana je i obrazložena programska izvedba korisničkog sučelja. U šestom poglavlju istaknut je početni problem, izloženo je rješenje problema i navedeni su mogući koraci poboljšanja rješenja.

2. Korisnička prilagodba web stranica

Stalnim unaprjeđivanjem mogućnosti koje pružaju suvremene web stranice raste i profinjenost korisničkih zahtjeva. Osim naprednih primjenskih funkcionalnosti, nerijetko se zahtijeva mogućnost prilagodbe sadržaja stranice pojedinačnim potrebama korisnika. Prilagodbom sadržaja stranice moguće je obuhvatiti širok opseg izmjena, kao što je promjena veličine teksta u cilju poboljšanja čitljivosti, kombiniranje sadržaja s različitih web stranica, obogaćivanje stranice kontrolama za jednostavniju navigaciju ili naprednije oblike međudjelovanja između korisnika i stranice i slično. Jedna od metoda prilagodbe sadržaja zasniva se na prepoznavanju korisnika od strane poslužitelja i dinamičko stvaranje sadržaja na osnovi prethodno prikupljenih korisničkih odrednica (engl. *user profile*).

S druge strane, razvijeni su posebni alati koji korisniku omogućavaju samostalnu prilagodbu sadržaja. Takvi alati najčešće su dostupni u obliku proširenja ili dodataka za preglednik, odnosno računalnih programa oblikovanih za ugradnju u preglednik s ciljem poboljšavanja ili obogaćivanja postojećih funkcionalnosti. Premda ne postoji preglednik koji podržava ugradnju svih raspoloživih dodataka, za gotovo svaki preglednik postoji odgovarajući alat s potporom za samostalnu prilagodbu sadržaja od strane korisnika. Pregled najkorištenijih preglednika i odgovarajućih dodataka prikazan je u tablici 2.1.

Tablica 2.1. Preglednici s pripadnim dodacima za prilagodbu sadržaja

Preglednik	Dodaci za samostalnu prilagodbu sadržaja
Internet Explorer	IE7Pro, Turnabout, Greasemonkey for IE
Mozilla Firefox	Greasemonkey
Safari	GreaseKit, PithHelmet
Konqueror	Konqueror Userscript

U ovom poglavlju opisuju se svojstva i princip rada alata *Greasemonkey* koji omogućava prilagodbu sadržaja od strane korisnika, a ugrađuje se u preglednik *Mozilla Firefox*. U sljedećim potpoglavljima opisani su mehanizmi dohvata web stranica uz objašnjenja pripadnih prednosti i nedostataka. Opisane su značajke i osnovna načela rada alata *Greasemonkey* uz izložen pregled ključnih odrednice nastanka i razvoja alata. Opisani su načini prilagodbe web stranica primjenom alata *Greasemonkey*, prikazana su pravila pisanja korisničkih skripti te je objašnjen postupak konfiguracije alata. Poglavlje završava osvrtom na najčešća odstupanja od standardnog *JavaScript* jezika, pojašnjenje uzroka odstupanja i postupke za ostvarenje odgovarajućih funkcionalnosti. Pruža se uvid u načine zaobilaska sigurnosnih značajki alata *Greasemonkey* uz opis pripadnih rizika.

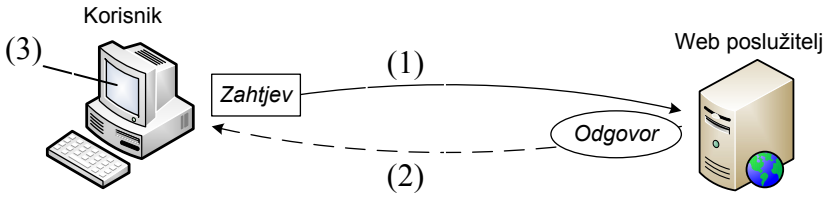
2.1. Mehanizmi dohvata web stranice

U ovom poglavlju opisana su tri mehanizma dohvata web stranica. Prvi opisani mehanizam je osnovni mehanizam dohvata web stranica. Slijedi opis mehanizma s prilagodbom sadržaja na strani web poslužitelja, odnosno mehanizma koji je nastao proširenjem funkcionalnosti web poslužitelja u osnovnom mehanizmu dohvata web stranica. Poglavlje završava opisom mehanizma s prilagodbom sadržaja na strani preglednika, nastalog obogaćivanjem funkcionalnosti preglednika u osnovnom mehanizmu dohvata web stranica. Za svaki mehanizam navedene su i opisane pripadne prednosti i nedostaci.

2.1.1. Osnovni mehanizam dohvata web stranice

Osnovni mehanizam dohvata web stranice čine dva dijela. Prvi dio predstavlja web poslužitelj, čija se funkcionalnost sastoji od primanja zahtjeva i slanja odgovora. Drugi dio predstavlja jedan korisnik ili više korisnika koji upućuju zahtjeve za web stranicom i dobivaju odgovore u obliku traženih web stranica.

Osnovni mehanizam dohvata web stranice promatran sa korisničke strane, odnosno strane preglednika, sastoji se od tri koraka. U prvom koraku upućen je zahtjev za web stranicom prema web poslužitelju (1). U drugom koraku dobiven je odgovor od web poslužitelja u obliku tražene web stranice (2). U trećem koraku prikazan je sadržaj web stranice u pregledniku (3).

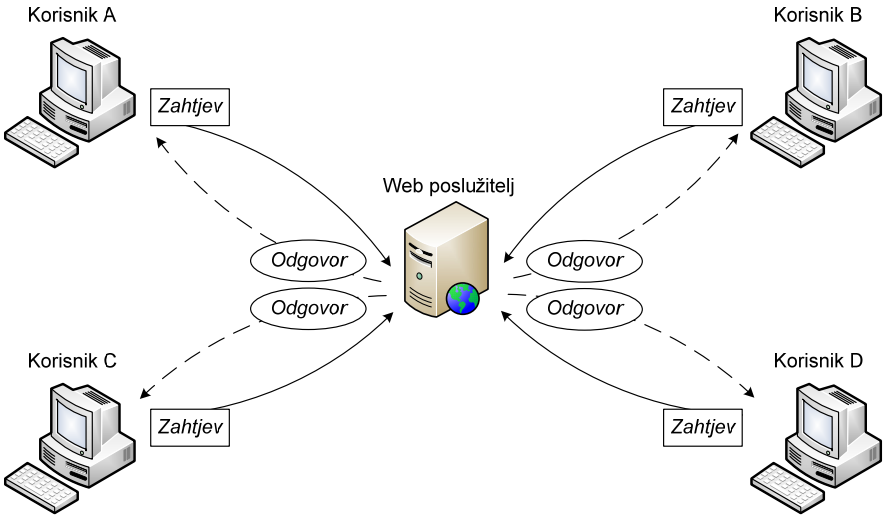


Slika 2.1. Osnovni mehanizam dohvata web stranice

U upućenom zahtjevu nalazi se adresa web poslužitelja i naziv datoteke koji odgovara traženoj web stranici. Odgovor čini web stranica sastavljena od niza *HTML* naredbi, čijim izvođenjem preglednik stvara i oblikuje elemente prikaza web stranice.

> HTML (engl. *Hyper Text Markup Language*) je programski jezik za stvaranje web stranica. Jezikom HTML opisuje se način prikaza sadržaja web stranice korisniku. Tumačenjem jezika HTML, preglednik stvara kontrole kojima se ostvaruje međudjelovanje korisnika i stranice.

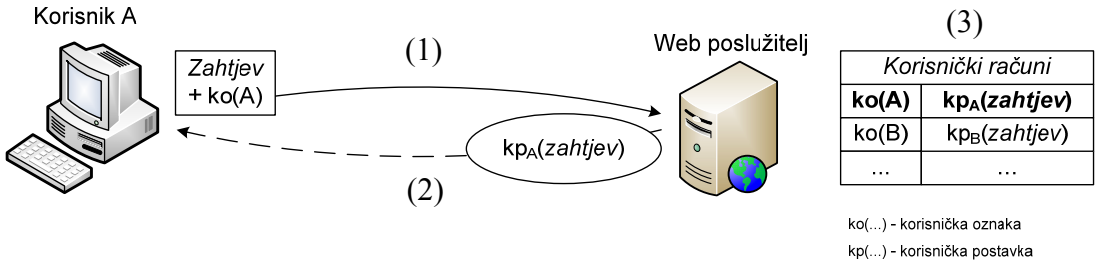
Osnovni mehanizam dohvata web stranice je jednostavan. Poslužitelj na jednake zahtjeve odgovara jednakom web stranicom, a preglednik ne utječe na sadržaj primljene web stranice. Nedostatak opisanog mehanizma je nemogućnost prilagodbe prikaza sadržaja web stranice pojedinačnim korisničkim zahtjevima. Na slici 2.2. prikazan je web poslužitelj okružen sa četiri korisnika. Svaki od korisnika upućuje zahtjev web poslužitelju za istom web stranicom. Web poslužitelj na svaki od upućenih zahtjeva odgovara jednakom, pojedinom korisniku neprilagođenom web stranicom.



Slika 2.2. Korisnički zahtjevi za istom web stranicom (1)

2.1.2. Mehanizam s prilagodbom sadržaja na strani web poslužitelja

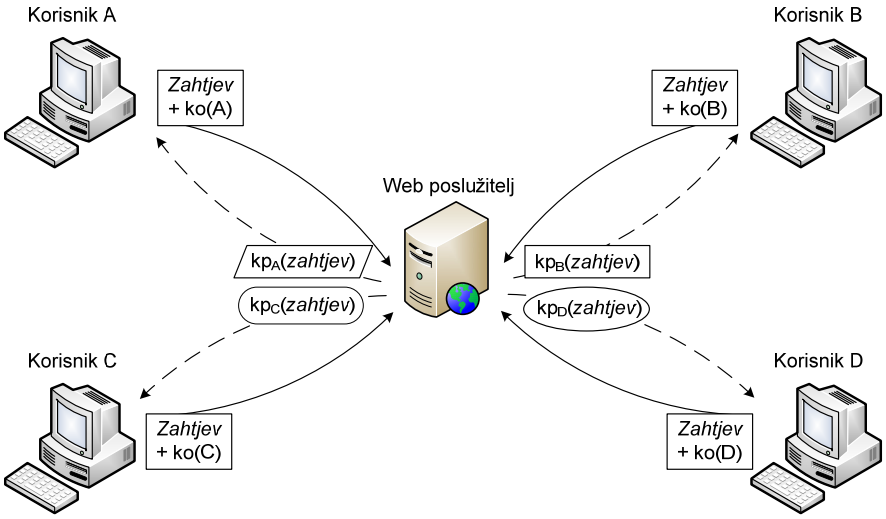
Kako bi se posluživanje web stranice na zahtjev korisnika proširilo mogućnošću prilagodbe sadržaja, osnovna funkcionalnost web poslužitelja proširuje se uvođenjem sustava za upravljanje korisničkim računima. Spomenutim proširenjem nastaje mehanizam s prilagodbom sadržaja na strani web poslužitelja prikazan slikom 2.3.



Slika 2.3. Sadržaj odgovora prilagođen je primjenom korisničkih postavki

Na strani poslužitelja održava se baza korisničkih računa (3). Korisnički račun sastoji se od korisničke oznake pridružene odgovarajućim korisničkim postavkama. Korisnik stvara korisničke postavke definiranjem oblika i vrste sadržaja web stranice te ih pohranjuje na web poslužitelju. Preglednik uvrštava korisničku oznaku u zahtjev za web stranicom, čime se na jedinstven način predstavlja web poslužitelju (1). Web poslužitelj nakon primanja zahtjeva prepoznaje korisničku oznaku i primjenjuje odgovarajuće korisničke postavke na traženu web stranicu te njome odgovara na zahtjev (2).

Slika 2.4. prikazuje web poslužitelj okružen sa četiri korisnika. Svaki od korisnika upućuje zahtjev web poslužitelju za istom web stranicom. Web poslužitelj primjenjuje odgovarajuće korisničke postavke na zahtijevanu web stranicu i svakom korisniku odgovara prilagođenom web stranicom.

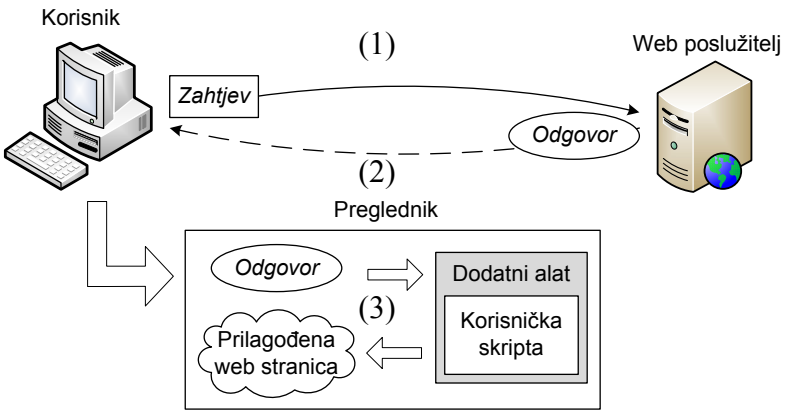


Slika 2.4. Korisnički zahtjevi za istom web stranicom (2)

Mehanizam s prilagodbom sadržaja na strani web poslužitelja složeniji je od osnovnog mehanizma dohvata web stranice. Glavni nedostaci mehanizma su dodatno opterećenje web poslužitelja zbog obrade korisničkih postavki, nepostojanje sustava za upravljanje korisničkim računima na svim web poslužiteljima i početni postupak definiranja postavki na web poslužitelju.

2.1.3. Mehanizam s prilagodbom sadržaja na strani preglednika

Drugi način prilagodbe sadržaja korisničkih stranica je mehanizam s prilagodbom sadržaja na strani preglednika. Proširenjem funkcionalnosti preglednika upotrebom dodatnih alata uz zadržanu funkcionalnost web poslužitelja iz osnovnog mehanizma dohvata web stranice, nastaje mehanizam s prilagodbom sadržaja na strani klijenta prikazan slikom 2.5.



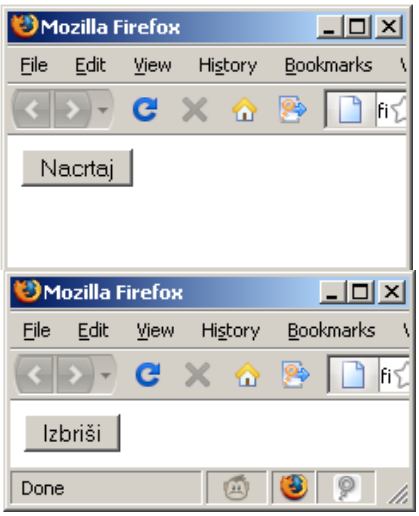
Slika 2.5. Prilagodba odgovora web poslužitelja dodatnim alatom preglednika

Upućivanje zahtjeva web poslužitelju (1) i dobivanje odgovora od web poslužitelja (2) odvija se kao u osnovnom mehanizmu dohvata web stranice. Nakon dobivanja odgovora web poslužitelja, prije prikaza sadržaja web stranice u pregledniku, u dodatnom alatu se odvija korak prilagodbe sadržaja web stranice određen programom u korisničkoj skripti (3). Nedostatak ovog mehanizma je potreba za prilagođavanjem dodatnog alata i nužnost poznavanja skriptnog jezika zbog pisanja korisničkih skripti. Prednosti mehanizma su mogućnosti vrlo široke prilagodbe sadržaja neograničenog skupa web stranica te raspodijeljenost postupka prilagodbe sadržaja web stranica na preglednike.

Primjerom 2.1. pokazana je prilagodba sadržaja web stranice u pregledniku. Preglednik tumači *HTML* kod i stvara web kontrolu, odnosno gumb. Izvođenje skriptnog programa mijenja natpis na gumbu "Nacrtaj" u "Izbriši".

```
<html>
<body>
<button id="g" name="g">
    Nacrtaj
</button>
</body>
</html>

// Korisnička skripta
var element;
element = document.getElementById("g");
element.textContent = "Izbriši";
```



Primjer 2.1. Prilagodba web stranice promjenom naziva gumba

2.2. Načelo rada alata Greasemonkey

Greasemonkey je dodatak web pregledniku *Mozilla Firefox*. Ugradnjom alata u preglednik omogućava se lokalna izmjena sadržaja učitane web stranice bez utjecaja na sadržaj web stranice spremljen na poslužitelju. Na taj način alat *Greasemonkey* omogućava posluživanje jezgrenih funkcionalnosti web stranice s poslužitelja pružatelja usluge, dok se preciznija prilagodba sadržaja poosobljenim potrebama korisnika odvija unutar korisnikovog web preglednika. Pravila za izmjenu sadržaja samostalno definira krajnji korisnik u obliku skriptnih programa (engl. *user scripts*).

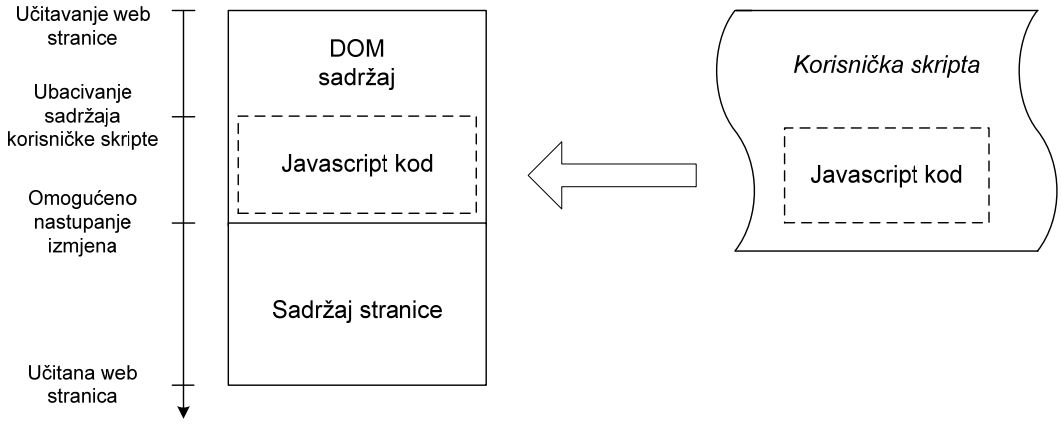
Skriptni program (korisnička skripta, skripta) sastoji se od jedne ili više *JavaScript* naredbi, a izvodi se izravno iz izvornog koda unutar preglednika. Skriptnim programom korisnik definira upravljanje sadržajem preglednika.

S obzirom na način izvođenja izvornog koda korisničke skripte te način međudjelovanja korisničke skripte i web stranice, postoje dvije inačice alata *Greasemonkey*. Alat *Greasemonkey* u starijoj inačici ugrađivao je sadržaj korisničke skripte u odabranu web stranicu, a način pristupa objektima korisničke skripte nije se razlikovao od načina pristupa objektima web stranice. Skup naredbi dostupnih za pisanje korisničkih skripti činio je skup naredbi jezika *JavaScript*. Alat *Greasemonkey* u novijoj inačici odjeljuje sadržaj korisničke skripte od sadržaja web stranice. Uklonjena je mogućnost međudjelovanja web stranice prema korisničkoj skripti i izmijenjen je način pristupa objektima web stranice. Skup naredbi dostupnih korisničkoj skripti čini podskup naredbi jezika *JavaScript*.

2.2.1. Inačica alata Greasemonkey zasnovana na izravnom ubacivanju koda

Na slici 2.6 prikazano je načelo rada alata Greasemonkey zasnovanog na izravnom ubacivanju koda (engl. *code injection*) s naznačenim tijekom vremena na lijevoj strani slike. Učitavanje web stranice započinje učitavanjem DOM sadržaja (engl. *Document Object Model content, DOM content*).

DOM je sučelje neovisno o platformi i jeziku, a programima i skriptama omogućava pristup strukturi, sadržaju i stilu HTML dokumenta.



Slika 2.6. Izvođenje korisničke skripte kod alata Greasemonkey zasnovanog na izravnom ubacivanju koda

DOM sadržaj je strukturirani popis elemenata web stranice. Strukturu *DOM* sadržaja čini stablo čvorova građeno od *HTML* oznaka (engl. *tag*) i stablo *DOM* elemenata. *HTML* oznake sastoje se od para rezerviranih riječi jezika *HTML*, npr. `<script>...</script>`. Svaki čvor stabla je spremnik s ugniježđenim čvorom niže razine, odnosno sadržajem definiranim između *HTML* oznaka. Za oznaku `script` sadržaj se definira na temelju jezika *JavaScript*. *DOM* elementi sastoje se od izvornih i korisničkih *DOM* funkcija i objekata.

Nakon učitavanja *DOM* sadržaja web stranice, alat *Greasemonkey* proširuje *DOM* sadržaj stvaranjem čvora `script`. Alat *Greasemonkey* preuzima *JavaScript* izvorni kod korisničke skripte i ubacuje ga u sadržaj stvorenog čvora `script`, čime izvorni kod korisničke skripte postaje dio web stranice. Osim čvorova `script` stvorenih alatom *Greasemonkey*, *DOM* sadržaj može sadržavati čvorove `script` koji su sastavni dijelovi web stranice nastali na temelju ugrađenih skripti. Opisani postupak ponavlja se za svaku korisničku skriptu. Nakon ugradnje posljednjeg čvora `script` omogućeno je nastupanje izmjena web stranice definirano korisničkim skriptama. Na slici 2.7. prikazan je primjer *HTML* koda i njime određeno stablo čvorova *DOM* sadržaja.

```
<html>
   <head>
   </head>
   <body>
      <button id="gNacrtaj" name="gNacrtaj">
         Nacrtaj
      </button>
      <script type="text/javascript">
      </script>
   </body>
</html>
```

Slika 2.7. Prikaz HTML izvornog koda i pripadnog DOM sadržaja

Čvorovi `script` dodani *DOM* sadržaju istovjetni su čvorovima `script` ugrađenih korisničkih skripti. Istovjetnost čvorova omogućava upotrebu jednakog skupa naredbi jezika *JavaScript* pri pisanju korisničkih skripti i ugrađenih skripti web stranice. Tumačenjem naredbi korisničkih skripti i naredbi ugrađenih skripti preglednik izravno pristupa objektima i funkcijama iz *DOM* sadržaja, omogućavajući potpuno međudjelovanje korisničkih skripti i ugrađenih skripti. Primjer 2.2. pokazuje međudjelovanje korisničke skripte i ugrađene skripte.

```
// korisnička skripta
function autoPrijava(){
      element_ime = document.getElementById('polje_korIme');
      element_zap = document.getElementById('polje_korZap');
      element_ime.value = 'matija';
      element_zap.value = 'abc123';
}

// ugrađena skripta
autoPrijava();
```

Primjer 2.2. Međudjelovanje korisničke skripte i ugrađene skripte

U primjeru 2.2. u korisničkoj skripti definirana je funkcija `autoPrijava`, koja na web stranici upisuje korisničke podatke za prijavu na sustav. Izvornom *DOM* funkcijom `document.getElementById` dohvaćaju se polja za upis korisničkog imena i korisničke zaporke. Dohvaćena polja su pridružena varijablama `element_ime` odnosno `element_zap`. Svojstva varijabli `element_ime.value` i `element_zap.value` postavljaju se na odgovarajuće vrijednosti. Iz ugrađene skripte poziva se korisnička *DOM* funkcija `autoPrijava` definirana u korisničkoj skripti.

2.2.2. Inačica alata Greasemonkey zasnovana na korištenju pješčanika

Opisani primjer 2.2. je jednostavan i u općenitom slučaju ispravan, no sadrži sigurnosne propuste zasnovane na arhitekturi starije inačice alata *Greasemonkey*. Nesigurnost starije inačice alata *Greasemonkey* pojavljuje se zbog korištenja *DOM* funkcija web stranice i ubacivanja čvorova `script` sa sadržajem korisničkih skripti u *DOM* sadržaj web stranice.

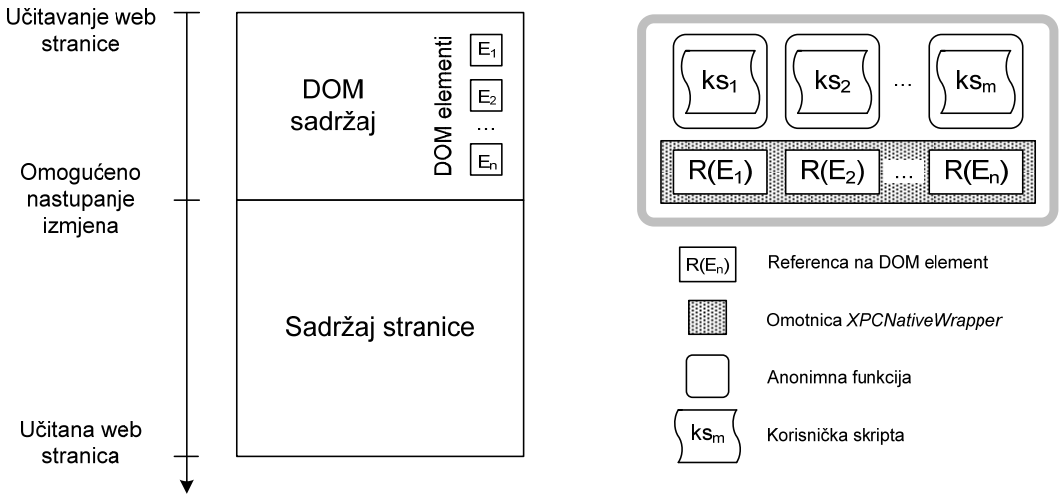
Pri pozivu *DOM* funkcije `getElementById` korisnička skripta se oslanja na izvornu funkcionalnost pozvane *DOM* funkcije, no funkcionalnost svake izvorne *DOM* funkcije moguće je neprimjetno izmijeniti ugrađenom skriptom web stranice. Nepoznata funkcionalnost pozvane *DOM* funkcije izravno utječe na ponašanje korisničke skripte, čime korisnička skripta postaje neupotrebljiva, a njena funkcionalnost neodređena. Slika 2.8. prikazuje neke od najčešće korištenih izvornih (engl. *native*) *DOM* funkcija.

```
function getElementById() {    [native code] }
function getElementsByClassName() {    [native code] }
function getElementsByName() {    [native code] }
function getElementsByTagName() {    [native code] }
function getElementsByTagNameNS() {    [native code] }
```

Slika 2.8. Dio najčešće korištenih izvornih DOM funkcija

Ubacivanjem čvora `script` korisničke skripte u *DOM* sadržaj, izvorni kod korisničke skripte postaje sastavni dio *DOM* sadržaja. Ugrađenom skriptom posebne namjene web stranici je omogućeno preuzimanje sadržaja svakog čvora iz *DOM* sadržaja. Preuzimanjem sadržaja čvora korisničke skripte iz primjera 2.2. pribavljeni su korisnički podaci koje ugrađena skripta posebne namjene može isporučiti web poslužitelju i narušiti sigurnost korisnika.

Novija inačica alata *Greasemonkey* napušta tehniku izravnog ubacivanja koda u *DOM* sadržaj i koristi pješčanik (engl. *sandbox*) kao sigurnu okolinu za izvođenje korisničkih skripti. Alat *Greasemonkey* gradi pješčanik omatanjem korisničkih skripti anonimnim funkcijama te omatanjem referenci na objekte omotnicom `XPCNativeWrapper` ugrađenom u web preglednik *Mozilla Firefox*. Napuštanjem tehnike ubacivanja izvornog koda, web stranici je onemogućen dohvat sadržaja korisničke skripte.



Slika 2.9. Izvođenje skripte kod alata Greasemonkey zasnovanog na korištenju pješčanika

Opis osnovnog načela rada alata Greasemonkey popraćen je slikom 2.9. s naznačenim tijekom vremena na lijevoj strani slike. Učitavanje web stranice započinje učitavanjem *DOM* sadržaja web stranice i postavljanjem korisničkih skripti u pješčanik. Završetak učitavanja *DOM* sadržaja obilježen je događajem (engl. *event*) `DOMContentLoaded`, što Greasemonkey koristi kao signal za omotavanje referenci na izvorne DOM elemente omotnicom `XPCNativeWrapper`.

Slijedi izvođenje programskog koda korisničkih skripti, odnosno upravljanje sadržajem web stranice. Nastupanje izmjena web stranice moguće je započeti ili ponoviti prikladnim korisničkim djelovanjem, npr. odabirom aktivnog dijela web stranice ili automatiziranom reakcijom na neki događaj.

Svaka korisnička skripta omotana je anonimnom funkcijom koja ograničava globalni doseg i životni vijek objekata skripte, odnosno ostvaruje izolaciju skripte od okoline koju čine objekti ostalih skripti i web stranice. Izolacijom skripte uklanja se mogućnost škodljivog međudjelovanja korisničkih skripti. Učinak izolacije skripte istovjetan je učinku korištenja prostora imena (engl. *namespace*), čime objekti iz okoline postaju nedostupni. Primjer 2.3. pokazuje različite ishode izvođenja istog izvornog koda web stranice i korisničke skripte.

```
var varijabla = 2;
alert(window['varijabla']);
```

Primjer 2.3. Izvođenje programskog koda ovisnog o okolini

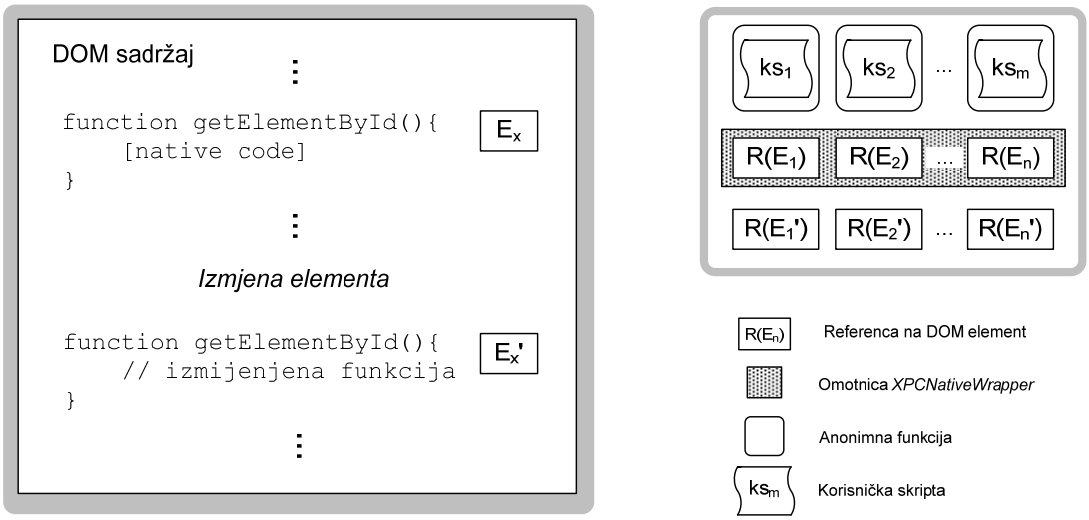
Izvođenje izvornog koda web stranice završava očekivanim prikazom dijaloškog prozora s vrijednošću `'2'`, no izvođenje izvornog koda korisničke skripte završava prikazom dijaloškog prozora s vrijednošću `'undefined'`, jer je nestandardni objekt `window['varijabla']` nedostupan izoliranoj korisničkoj skripti.

Posljedice neprovođenja izolacije skripte u starijoj inačici alata Greasemonkey zasnovanoj na izravnom ubacivanju koda pokazuju se primjerom 2.4.

```
var index = 5;
while (index > 0){
        index--;
}
```

Primjer 2.4. Izvođenje programskog koda neizoliranje skripte

Izvođenjem istog izvornog koda više korisničkih skripti, varijabla `index` bila bi višestruko deklarirana i inicijalizirana, a broj iteracija petlji bio bi manji od 5. Završetkom izvođenja korisničke skripte nestaje cijeli kontekst skripte, uključujući omotavajuću anonimnu funkciju. Nestanak konteksta skripte onemogućava kasniji pristup objektima skripte. Problemi vezani uz ovakvo ponašanje korisničkih skripti obrađuju se u poglavlju 2.3.4. Upotreba omotanih referenci *DOM* elemenata omotnicom `XPCNativeWrapper` osigurava pristup izvornim elementima *DOM* modela te onemogućava pristup dodanim ili izmijenjenim objektima. Omotnice štite korisničku skriptu od nekontrolirane promjene funkcionalnosti zasnovane na nepoznatom ponašanju korištenog objekta.



Slika 2.10. Pristup sigurnim i nesigurnim objektima

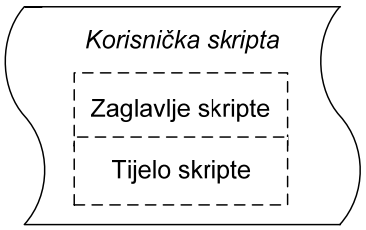
Greasemonkey podržava pristup nesigurnim *DOM* elementima i njihovim izmijenjenim ili dodanim svojstvima. Na slici 2.10 pokazano je istovremeno zadržavanje omotane reference $R(E_1)$ izvornog DOM elementa i nove reference $R(E_1')$ izmijenjenog (korisničkog) DOM elementa. Usporedba ova dva pristupa provedena je u poglavlju 2.3.4.

2.3. Postupak prilagodbe sadržaja primjenom alata Greasemonkey

Prilagodba sadržaja web stranice primjenom alata *Greasemonkey* izvodi se pisanjem korisničkih skripti. Korisnička skripta za Greasemonkey sadrži zaglavlje i skriptni program u jeziku *JavaScript* za izmjenu sadržaja web stranice učitane u web preglednik. Zaglavljem korisničke skripte moguće je ograničiti djelovanje skripte na odabrani skup web stranica, uključiti pomoćne skripte itd. Korisničke skripte su tekstualne datoteke s datotečnim nastavkom `.user.js`. Za uređivanje sadržaja korisničkih skripti dovoljan je uređivač teksta.

2.3.1. Pisanje korisničke skripte

Zaglavlje korisničke skripte sastoji se od metapodataka (podaci o podacima) kojima je posredno putem obilježja definirano ponašanje alata *Greasemonkey* i pružene su detaljnije informacije o korisničkoj skripti, primjerice porijeklo skripte, ime autora, kratak opis funkcionalnosti skripte i sl.. Shematski prikaz korisničke skripte prikazan je na slici 2.11.



Slika 2.11. Shematski prikaz korisničke skripte

Alat *Greasemonkey* pristupa metapodacima tijekom instalacije korisničke skripte i istovremeno gradi obilježja, zbog čega naknadne izmjene nad korisničkom skriptom nemaju učinak na ponašanje alata. Izmjena obilježja može se provesti uređivanjem konfiguracijske datoteke alata *Greasemonkey* ili ponovnom instalacijom izmijenjene skripte.

Iako se metapodaci mogu navesti bilo kojim redoslijedom na bilo kojem mjestu u skripti, najčešće se navode na početku skripte, nakon čega slijedi skriptni program u obliku *JavaScript* koda. Uobičajena korisnička skripta ilustrirana je primjerom 2.5.

```
// ==UserScript==
// @name               Automatizirana prijava
// @namespace          http://www.skripte.hr/primjeri/
// @description   Primjer korisnicke skripte
// @include       *
// @exclude       http://www.b.com/*
// @resource      imeResursa     slika1.jpg
// @require       pomocnaSkripta.js
// @author        Matija
// @version       1.0
// @homepage
// @copyright
// @statussize
// @defaulticon
// ==/UserScript==

function autoPrijava(){
   element_ime = document.getElementById('polje_korIme');
   element_zap = document.getElementById('polje_korZap');
   element_ime.value = 'matija';
   element_zap.value = 'abc123';
}
```

Primjer 2.5. Korisnička skripta

Omotnica metapodataka sadrži metapodatke u obliku komentara, a počinje i završava komentarima:

```
// ==UserScript==

// ==/UserScript==
```

Omotnica metapodataka je isključivo ovog oblika i nužan je dio svake korisničke skripte. Alat *Greasemonkey* koristi omotnicu za označavanje početka i kraja niza metapodataka. Bez omotnice bi prilikom prevođenja bili zanemareni metapodaci, kao i ostali *JavaScript* komentari.

U omotnici iz primjera 2.5. postoji 7 oblika metapodataka koji definiraju ponašanje alata Greasemonkey: *name*, *namespace*, *description*, *include*, *exclude, resource* i *require*.

Preostali oblici metapodataka ne definiraju ponašanje alata Greasemonkey: *author*, *version*, *homepage*, *copyright*, *statussize* i *defaulticon*.

U općenitom slučaju, metapodaci su oblika:

```
// @ključ        vrijednost
```

Name je metapodatak oblika:

```
// @name         Automatizirana prijava
```

Name označava ime kojim je korisnička skripta predstavljena korisniku u dijaloškom okviru tijekom instalacije i u popisu instaliranih skripti unutar sučelja alata Greasemonkey. *Name* nije nužan, a navodi se najviše jednom. Ako se izostavi *name*, njegova vrijednost će se izjednačiti s imenom datoteke korisničke skripte bez nastavka `.user.js`.

Namespace je metapodatak oblika:

```
// @namespace         http://www.skripte.hr/primjeri/
```

Namespace je *URI* kojim alat *Greasemonkey* razlikuje korisničke skripte s istim imenom i različitim autorima. *Namespace* nije nužan, a navodi se najviše jednom. Moguće vrijednosti metapodatka *namespace* su *FQDN* nizovi znakova ili oznaka `tag:`URI. Ako se izostavi *namespace*, njegova vrijednost će se izjednačiti s imenom domene s koje je preuzeta korisnička skripta. Za lokalno napisane korisničke skripte može se postaviti vrijednost

```
// @namespace         http://localhost
```

Description je metapodatak oblika:

```
// @description       Primjer korisnicke skripte
```

Description predstavlja opis funkcionalnosti skripte. Njegov sadržaj je prikazan u dijaloškom okviru prilikom instalacije korisničke skripte te u popisu instaliranih skripti unutar sučelja alata *Greasemonkey*. Iako *description* nije nužan, preporuča se njegovo navođenje kako bi se olakšala navigacija među većim brojem korisničkih skripti. *Description* se definira najviše jednom i to u preporučenoj duljini ne većoj od dvije

rečenice. Ako se izostavi *description*, njegova vrijednost će se izjednačiti s praznim znakovnim nizom.

Include i *exclude* su metapodaci oblika:

```
// @include      *
// @exclude      http://www.b.com/*
```

Include i *exclude* imaju značenje *URL* upute, jednake su sintakse, a zbog svoje funkcionalnosti spadaju među najvažnije metapodatke. *Include* i *exclude* definiraju skup *URL*-ova i domena nad kojima je dopušteno odnosno zabranjeno izvođenje korisničke skripte. Dozvoljene vrijednosti metapodataka *include* i *exclude* su *URL*, *URL* sa znakom * (znak * predstavlja bilo koju domenu / putanju) te samo znak *. Metapodatak *exclude* većeg je prioriteta od metapodatka *include*. Primjer 2.5. ilustrira slučaj gdje se dopušta izvođenje skripte za sve domene, osim domene `www.b.com` i pripadnih putanja.

Include i *exclude* nisu nužni. Dozvoljava se navođenje neograničenog broja *include* i *exclude* metapodataka navedenih u zasebnim linijama. Ako se izostave oba metapodatka, *include* će se interno postaviti na vrijednost *, odnosno dozvolit će se izvršavanje korisničke skripte na svim domenama.

Resource je metapodatak oblika:

```
// @resource          imeResursa slika1.jpg
```

Resource je namijenjen uključivanju lokalnih i udaljenih resursa u korisničku skriptu tijekom prve instalacije skripte. Dozvoljava se navođenje neograničenog broja *resource* s jedinstvenim imenima unutar korisničke skripte. Vrijednost *resource* može sadržavati `file`, `http`, `https` ili `ftp` sheme imenovanja. Izostavljanjem sheme imenovanja Greasemonkey predviđa da je resurs lokalno pohranjen, a vrijednost *resource* izjednačava se s imenom domene s koje je preuzeta korisnička skripta. Za uključivanje / isključivanje lokalnih i udaljenih resursa potrebno je ukloniti korisničku skriptu iz alata *Greasemonkey* i ponoviti njenu instalaciju.

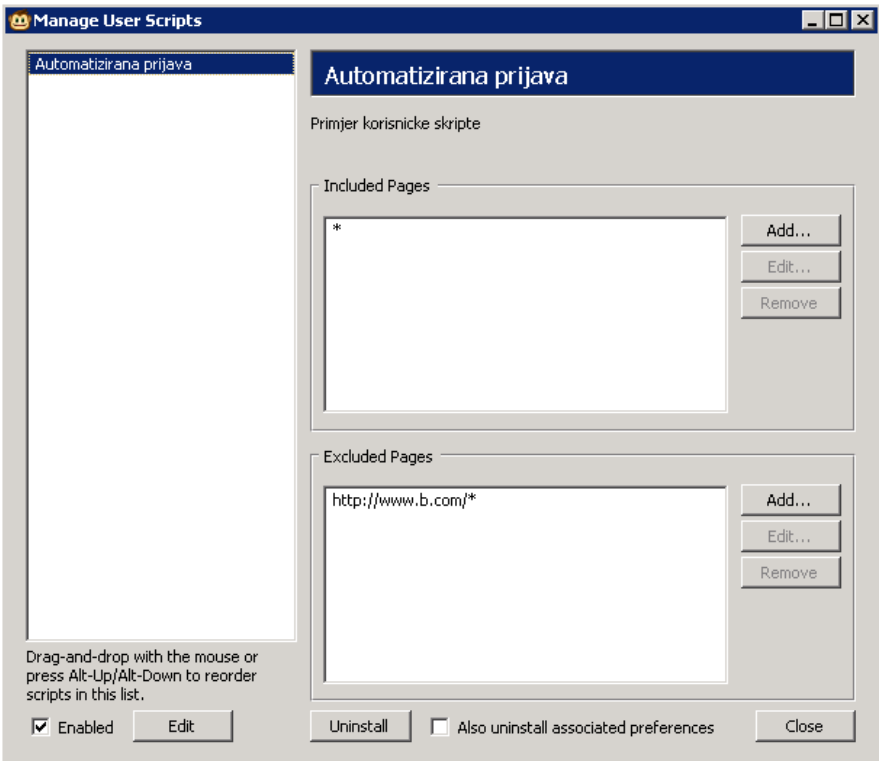
Require je metapodatak oblika:

```
// @require      pomocnaSkripta.js
```

Require služi za uključivanje lokalnih i udaljenih skripti u korisničku skriptu tijekom instalacije. Vrijednost resource može sadržavati `file`, `http`, `https` ili `ftp` sheme imenovanja. Izostavljanjem sheme imenovanja *Greasemonkey* predviđa da je korisnička skripta lokalno pohranjena, a vrijednost *require* izjednačava se s imenom domene s koje je instalirana skripta. Za naknadno uključivanje / isključivanje lokalnih i udaljenih skripti potrebno je ukloniti korisničku skriptu iz alata *Greasemonkey* i ponoviti njenu instalaciju.

2.3.2. Konfiguriranje alata

Uz uređivanje zaglavlja skripte i ručno uređivanje konfiguracijske datoteke, alat *Greasemonkey* moguće je konfigurirati putem njegovog sučelja. Sučelje alata *Greasemonkey* pojednostavljuje izradu zaglavlja skripte i omogućava upravljanje korisničkim skriptama, tj. omogućava postavljanje, uklanjanje, uključivanje i isključivanje pojedinih skripti. Postupak konfiguracije alata *Greasemonkey* prikazan je slikom 2.12.



Slika 2.12. Sučelje alata *Greasemonkey*

U lijevom polju sučelja nalazi se popis postavljenih skripti. Svaku postavljenu skriptu moguće je trajno ukloniti iz alata ili ju samo privremeno uključiti, odnosno isključiti. U polju naziva *Included pages* navodi se skup domena ili stranica za koje je potrebno pokrenuti izvođenje skripte. U polju naziva *Excluded pages* navodi se skup domena ili

stranica za koje je zabranjeno pokrenuti izvođenje skripte. Definiranjem ovih dvaju skupova web stranica, ograničava se doseg djelovanja skripte. Na taj način *Greasemonkey* omogućava definiranje različitih pravila za izmjenu sadržaja različitih web stranica.

Upravljanje domenama

Upravljanje domenama odnosi se na postupak odabira web stranica odnosno web sjedišta nad kojima je dozvoljena ili zabranjena primjena korisničke skripte. Upravljanje domenama određeno je jednostavnom sintaksom za definiranje dozvoljenih i nedozvoljenih web stranica, primjerice:

1. Obuhvaćanje domena oblika `http://b.com` i `http://www.b.com`

   ```
   @include  http://b.com/*
   ```

   ```
   @include  http://www.b.com/*
   ```

2. Obuhvaćanje poddomena oblika `http://pc.b.com` i `http://www.b.com`

   ```
   @include  http://*.b.com/
   ```

3. Obuhvaćanje vršnih domena oblika `http://www.b.com`, `http://www.b.co.uk` i `http://www.b.hr`

   ```
   @include  http://www.b.tld/
   ```

   `.tld` je posebna oznaka koja obuhvaća sve vršne domene i domene zemalja

4. Obuhvaćanje svih domena

   ```
   @include  *
   ```

5. Obuhvaćanje sigurnih domena

   ```
   @include  https://*
   ```

6. Obuhvaćanje nesigurnih domena

   ```
   @include  http://*
   ```

Navedeni primjeri jednako vrijede za `@exclude` metapodatke.

2.3.3. Odstupanje od standardnog *JavaScript* koda

Sigurnosna ograničenja alata *Greasemonkey* ne podržavaju pojedine *JavaScript* naredbe, što predstavlja odstupanje od standardnog *JavaScript* koda. Kroz nekoliko primjera pokazuju se najčešća odstupanja od standardnog *JavaScript* koda uz ispravan način ostvarenja potrebne funkcionalnosti.

Jezik *JavaScript* podržava definiranje *odzivnih funkcija* (engl. c*allback functions*) u obliku znakovnih nizova (engl. *string*). Odzivna funkcija je funkcija koja se poziva ispunjenjem nekog od uvjeta odziva, primjerice istekom zadanog vremenskog intervala. Poziv odzivne funkcije `OdzivnaFunkcija()` prikazan je primjerom 2.6.:

```
window.setTimeout("OdzivnaFunkcija()", 1000);
```

Primjer 2.6. Primjer odzivne funkcije (1)

`window` je *DOM* element, odnosno objekt s ugrađenim svojstvima i funkcijama. `window.setTimeout` je jedna od ugrađenih izvornih funkcija objekta `window`, a koristi se za odgođeno pozivanje funkcije navedene kao prvi argument nakon isteka vremenskog intervala, odnosno drugog argumenta.

Ako je odzivna funkcija definirana unutar web stranice, nakon isteka 1000 ms od trenutka poziva funkcije `window.setTimeout` *JavaScript* će analizirati znakovni niz i pozvati funkciju `OdzivnaFunkcija()`. Ako je odzivna funkcija definirana u korisničkoj skripti, navedeni primjer neće biti ispravan. Uzrok ovog problema leži u sigurnoj okolini korisničke skripte, odnosno anonimnoj funkciji kojom je omotana korisnička skripta. Nakon isteka 1000 ms funkcija `window.setTimeout()` će potaknuti analizu znakovnog niza `"OdzivnaFunkcija()"`, no dotada će nestati anonimna funkcija i njen sadržaj, uključujući odzivnu funkciju. Ispravan pristup problemu prikazan je primjerom 2.7.:

```
window.setTimeout(OdzivnaFunkcija, 1000);
```

Primjer 2.7. Primjer odzivne funkcije (2)

U primjeru 2.7. argument funkcije `window.setTimeout()` nije znakovni niz, već referenca na odzivnu funkciju `OdzivnaFunkcija()`. *JavaScript* zadržava okolinu određenog objekta, primjerice lokalno definirane varijable, objekta ili funkcije, sve do trenutka u kojem više ne postoji referenca na takav objekt (engl. *closure*). Nakon nestanka reference na objekt nestaju objekt i njegova okolina, odnosno omotavajuća anonimna funkcija. Referenca na odzivnu funkciju `OdzivnaFunkcija` uzrokuje zadržavanje odzivne funkcije u pješčaniku do završetka izvođenja ugrađene funkcije `window.setTimeout()`.

JavaScript dopušta dinamičko postavljanje *rutine za obradu događaja* (engl. *event handler*), primjerice `onclick`. Uobičajeno se rutina za obradu događaja `onclick` postavlja pridruživanjem znakovnog niza svojstvu `.onclick` referenciranog elementa `element` kao u primjeru 2.8.:

```
var element = document.getElementById('neki_link');
element.onclick = 'Funkcija(this)';
```

Primjer 2.8. Rutina za obradu događaja (1)

`document.getElementById` je ugrađena izvorna funkcija *DOM* objekta `document`, a koristi se za dohvat reference objekta određenog identifikatorom. Referenca dohvaćenog objekta pridružuje se varijabli `element`. U dohvaćen objekt ugrađeno je svojstvo `onclick`, kojem je pridruženo ime rutine za obradu događaja `onclick`, odnosno znakovni niz `'Funkcija'`. Opisan način pridruživanja rutine za obradu događaja ispravno se izvodi u ugrađenoj skripti, ali prilikom pridruživanja rutine za obradu događaja u korisničkoj skripti se javlja problem istovjetan problemu definiranja odzivnih funkcija u korisničkoj skripti. Nakon ispunjenja uvjeta poziva rutine za obradu događaja odgovarajućom korisničkom akcijom, funkcija `Funkcija` više neće postojati.

Sukladno prethodnom primjeru, rješenje problema se naizgled može ostvariti referenciranjem funkcije `Funkcija` kao u primjeru 2.9.:

```
var element = document.getElementById('neki_link');
element.onclick = Funkcija;
```

Primjer 2.9. Rutina za obradu događaja (2)

Kao u primjeru 2.9., svojstvu dohvaćenog objekta `onclick` pridružuje se referenca funkcije `Funkcija`. Ovim pristupom je osigurana dostupnost funkcije `Funkcija`, no ne i postavljanje rutine za obradu događaja. Povratna vrijednost funkcije `getElementById` je omotnica `XPCNativeWrapper` oko traženog objekta. Omotnice nemaju ugrađeno svojstvo `onclick` i ne podržavaju ovakav pristup postavljanju rutina za obradu događaja. Iako omotnica `XPCNativeWrapper` podržava postavljanje većine svojstava omotanog objekta, rutine za obradu događaja moraju se postaviti kao u primjeru 2.10.:

```
var element = document.getElementById('neki_link');
element.addEventListener("click", Funkcija, false);
```

Primjer 2.10. Rutina za obradu događaja (3)

Funkcija `addEventListener` je ugrađena funkcija svih objekata *DOM* sadržaja uključujući objekte `XPCNativeWrapper`, a koristi se za pridruživanje rutine za obradu događaja uz događaj definiran prvim argumentom "`click`". Ovo je ispravan pristup postavljanja rutine za obradu događaja za sve elemente, `window` i `document` objekte te sve *DOM* događaje.

Jezik *JavaScript* podržava proširenje skupa ugrađenih svojstava objekta *korisnički definiranim svojstvima*. Unutar web stranice korisnička svojstva se definiraju kao u primjeru 2.11.:

```
var element = document.getElementById('neki_objekt');
element.novoSvojstvo = 'plavo';
```

Primjer 2.11. Korisnički definirana svojstva (1)

Kako `element` nije referenca na objekt, već referenca na omotnicu `XPCNativeWrapper`, ovaj pristup nije primjenjiv na korisničku skriptu. Ispravan način postavljanja vrijednosti korisnički definiranih svojstava unutar korisničke skripte prikazan je primjerom 2.12.:

```
var element = document.getElementById('neki_objekt');
element.setAttribute('novoSvojstvo', 'plavo');
```

Primjer 2.12. Korisnički definirana svojstva (2)

Funkcija `setAttribute` ugrađena je u sve DOM objekte, a koristi se za postavljanje vrijednosti korisničkog svojstva. Dohvat vrijednosti korisnički definiranih svojstava unutar korisničke skripte prikazan je primjerom 2.13.:

```
var svojstvo = element.getAttribute('novoSvojstvo');
```

Primjer 2.13. Dohvat korisnički definiranih svojstava

Funkcija `getAttribute` ugrađena je u sve *DOM* objekte, a koristi se za dohvat vrijednosti korisničkog svojstva. Povratna vrijednost funkcije `getAttribute` je znakovni niz koji odgovara dohvaćenom svojstvu.

2.3.4. Zaobilaženje sigurnosnih značajki

Odstupanja od standardnog *JavaScript* koda mogu se umanjiti zaobilaženjem sigurnosnih značajki. Iako se ovim načinom pisanja korisničkih skripti može ostvariti potrebna funkcionalnost, uveden je rizik neočekivanog ponašanja skripte. Takav rizik zasniva se na izravnom pristupu nesigurnim elementima *DOM* sadržaja. Nesigurnim *DOM* elementom smatra se svaki *DOM* element za koji je nemoguće utvrditi ima li izmijenjenu funkcionalnost. Korištenje nesigurnog *DOM* elementa može uzrokovati nepredviđeno ponašanje korisničke skripte. Zbog opisanog rizika u općenitom slučaju nije preporučljivo pristupanje nesigurnim objektima.

Nesigurnom objektu `window` može se pristupiti na dva ekvivalentna načina: imenom `unsafeWindow`, ili imenom `window.wrappedJSObject`. Ovim pristupom zaobilaze se sigurnosne značajke zasnovane na omotnici `XPCNativeWrapper`. Jedna od mogućih posljedice korištenja objekta pokazana je primjerima 2.14. i 2.15.

```
<html>
  <head>
     <script>
        var nesiguranObjekt = {};
        nesiguranObjekt.title = "Nesiguran objekt";
        document.getElementById =
          function(parametar){
            return nesiguranObjekt;
          }
     </script>
  </head>
  <body>
     <div id="element" title="Siguran
objekt"></div>
  </body>
</html>
```

Primjer 2.14. Web stranica s izmijenjenom funkcijom `document.getElementById`

```
window.addEventListener('load', function(){
   var dohvacenObjekt =
     unsafeWindow.document.getElementById("element");
   alert(dohvacenObjekt.title);
}, false);
```

Primjer 2.15. Dio koda korisničke skripte s pozivom nesigurne funkcije

U primjeru 2.15. poziva se nesigurna funkcija s očekivanom povratnom vrijednosti u obliku objekta `HTMLDivElement`. Pozvana funkcija izmijenjena je u primjeru 2.14. korištenjem ugrađene skripte web stranice nad kojom se izvodi korisnička skripta. Zaobilaskom sigurnosnih značajki korisnička skripta poziva izmijenjenu funkciju te kao povratnu vrijednost dobiva korisnički definiran objekt `Object`. U slučajevima gdje nije potpuno poznata funkcionalnost web stranice, povratna vrijednost nesigurne funkcije može narušiti tijek izvođenja korisničke skripte i promijeniti njenu funkcionalnost.

Suprotno nesigurnom pozivu funkcije, poziv funkcije oblika `document.getElementById('element')` upućen je izvornoj funkciji koja vraća izvornu povratnu vrijednost odnosno objekt `HTMLDivElement` u omotnici `XPCNativeWrapper`.

3. Primjenski sustavi zasnovani na udomljenicima

Udomljenici (engl. *gadgets*) su jednostavni, samostalni primjenski sustavi koji se izvode unutar korisnikovog web preglednika i omogućavaju visok stupanj prilagodbe korisničkim zahtjevima. Namijenjeni su obavljanju manjeg skupa precizno definiranih zadataka, kao što je prikaz vremenske prognoze, pretvorba novčanih valuta ili prevođenje stranih riječi. Širokim izborom udomljenika pokrivena su različita područja primjene, kao što su znanost, tehnologija, gospodarstvo, sport i mnoga druga. Slika 3.1. pokazuje izgled udomljenika za prevođenje stranih riječi.

Slika 3.1. Udomljenik za prevođenje stranih riječi

Udomljenici su namijenjeni ugradnji u udomiteljsku, spremničku (engl. *container*) stranicu kojom je ostvareno vizualno i logičko odvajanje skupa udomljenika od ostatka web stranice. Korištenjem udomljenika moguće je izgraditi složene primjenske sustave. Složeni primjenski sustavi nalikuju poosobljenim web portalima koje korisnik organizira sukladno svojim potrebama. Za razliku od web portala čiji se sadržaj sastoji od funkcionalno pasivnih elemenata, primjerice članaka teksta, video isječaka i slika, udomljenici u složenom primjenskom sustavu su funkcionalno aktivni elementi koji podržavaju međudjelovanje s korisnikom, dohvat informacija, obradu informacija i odgovarajuću prezentaciju sadržaja. Slika 3.2. prikazuje smještaj udomljenika u udomiteljsku stranicu

Slika 3.2. Smještaj udomljenika u udomiteljskoj stranici

Udomljenike je prema potrebi moguće povezivati, odnosno okupljati njihove funkcionalnosti (engl. *aggregation*). Upotrebom udomljenika bez povezivanja njihovih funkcionalnosti korisnik stvara složeni primjenski sustav po uzoru na korisniku otprije poznatu radnu okolinu ostvarenu kroz zasebne aplikacije na osobnom računalu. S druge strane, okupljanjem funkcionalnosti različitih udomljenika moguće je ciljano ostvariti funkcionalnost složenog primjenskog sustava i na taj način izgraditi modularan složeni primjenski sustav. Za izgradnju oba oblika složenog primjenskog sustava koriste se udomljenici s potrebnom funkcionalnosti koje je stvorio korisnik, ili udomljenici iz drugih izvora koji ostvaruju željenu funkcionalnost i dostupni su za ugradnju u složeni primjenski sustav.

U sljedećim potpoglavljima opisana je struktura iGoogle udomljenika kao gradivnih elemenata složenih primjenskih sustava. Navedeni su i opisani oblici udomljenika te načini njihove izgradnje. Opisan je sustav za pohranu i objavljivanje udomljenika s pomoćnim mehanizmima. Opisana je struktura udomiteljske stranice i izvođenje udomljenika u iGoogle udomiteljskoj stranici. Opisana je komunikacijska infrastruktura za razmjenu podataka među udomljenicima uz objašnjenje pripadajućih mehanizama.

3.1. Struktura i oblik udomljenika

XML (engl. *eXtensible Markup Language*) je jezik za označavanje podataka, a sintaksom je vrlo sličan jeziku *HTML*. Nastao je iz potrebe za jezikom koji je jednako čitljiv ljudima i računalnim aplikacijama, a široku primjenu nalazi u razmjeni podataka, spremanju podataka, povećanju dostupnosti podataka i izradi novih jezika za označavanje podataka.

Udomljenici su definirani strukturiranim *XML* datotekama. *XML* datoteka sadrži oznaku `<Module>` koja ugnježđuje sljedeće tri oznake. Prva oznaka, `<ModulePrefs>`, služi za zadavanje svojstava udomljenika, kao što su veličina udomljenika, naslov udomljenika, autor udomljenika itd. Druga oznaka, `<UserPrefs>`, služi za definiranje jednog korisničke postavke udomljenika, npr. postavke za komunikaciju među udomljenicima. Za definiranje više korisničkih postavki navodi se više oznaka `<UserPrefs>`. Sadržajem treće oznake, `<Content>`, određen je oblik udomljenika. Udomljenici su dostupni za ugradnju u primjenski sustav u obliku *HTML* udomljenika te u obliku *URL* udomljenika. Oznaka `<Content>` *HTML* udomljenika sadrži programski kod kojim je definirano korisničko sučelje i funkcionalnost udomljenika. Korisničko sučelje udomljenika opisano je jezikom *HTML*, a funkcionalnost udomljenika opisana je jezikom *JavaScript*. Struktura i sadržaj *XML* datoteke udomljenika pokazani su primjerom 3.1. na udomljeniku imena `Zbrajatelj`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Module>
   <ModulePrefs title="Zbrajatelj"/>
   <UserPrefs name="korisnik"/>
   <Content type="html"><![CDATA[
      <div id="d1"></div>
      <!-- funkcionalnost udomljenika-->
      <script type="text/javascript">
         var objekt = document.getElementById("d1");
         objekt.innerHTML = 1 + 2;
      </script>
   ]]></Content>
</Module>
```

Primjer 3.1. Programski kod udomljenika `Zbrajatelj`

U primjeru 3.1. zadan je naslov `Zbrajatelj` kao svojstvo udomljenika i definirano je korisničko svojstvo u ulozi korisnikovog imena. Korisničko sučelje udomljenika sastoji se od *HTML* elementa `<div>`, a funkcionalnost udomljenika sastoji se od dohvata reference na *HTML* element `<div>` i postavljanje njegove vrijednosti na rezultat računske operacije zbrajanja. Iako je opisani primjer jednostavan, struktura pripadajuće *XML* datoteke primjenjiva je na sve *HTML* udomljenike.

Oznaka `<Content>` *URL* udomljenika (engl. *Uniform Resource Locator*) sadrži poveznicu na zadanu web stranicu. Kako web stranice imaju ugrađeno korisničko sučelje i funkcionalnost, za *URL* udomljenike suvišno je definirati korisničko sučelje, funkcionalnost ili oznake korisničkih postavki oznakom `<UserPrefs>`. Primjerom 3.2. prikazan je sadržaj *XML* datoteke *URL* udomljenika.

```
<?xml version="1.0" encoding="UTF-8"?>
<Module>
   <ModulePrefs title="Bloomberg URL udomljenik"/>
   <Content type="url"
href="http://www.bloomberg.com"/>
</Module>
```

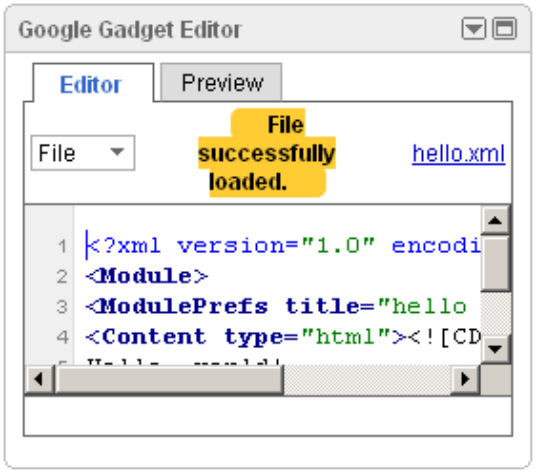
Primjer 3.2. *URL* udomljenik

3.2. Izgradnja udomljenika

Alat za uređivanje tekstualnih datoteka lokalno instaliran na korisnikovom računalu pruža svu potrebnu funkcionalnost za stvaranje ili uređivanje *XML* datoteke udomljenika. Kao zamjenu ili dopunu takvim alatima razvijen je *Google Gadget Editor*, udomljenik za programiranje, spremanje i objavljivanje korisničkih iGoogle udomljenika. *Google Gadget Editor* izvodi se u web pregledniku unutar *iGoogle* udomiteljske stranice.

Google Gadget Editor kroz međudjelovanje sa sustavom za pohranjivanje i objavljivanje udomljenika korisniku omogućava obavljanje potrebnih operacija pri radu s datotekama. U takve operacije uvršteni su dohvat *XML* datoteka udomljenika s korisničkog računala ili poslužitelja udomljenika, uređivanje i pohranjivanje *XML* datoteka udomljenika, brisanje *XML* datoteka udomljenika, ugradnja udomljenika u korisnikovu web stranicu te objavljivanje udomljenika u *iGoogle* udomiteljskoj stranici korisnika ili

imeničkom prostoru udomljenika (engl. *content directory*). Slika 3.3. prikazuje udomljenik *Google Gadget Editor* s otvorenom ispitnom datotekom `hello.xml`.

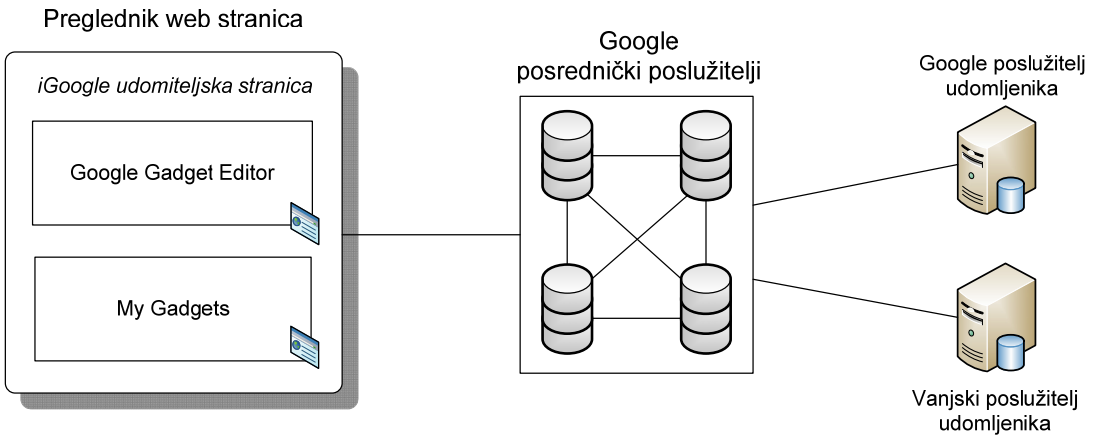


Slika 3.3. Primjer udomljenika *Google Gadget Editor*

Osim opisanog pristupa izgradnji udomljenika korištenjem udomljenika *Google Gadget Editor*, korisnik može upotrijebiti neki od alata za uređivanje tekstualnih datoteka i njime izgraditi *XML* datoteku udomljenika na lokalnom računalu. *XML* datoteku udomljenika moguće je učitati s lokalnog računala u *Google Gadget Editor* i zatim primijeniti neku od prethodno spomenutih funkcionalnosti.

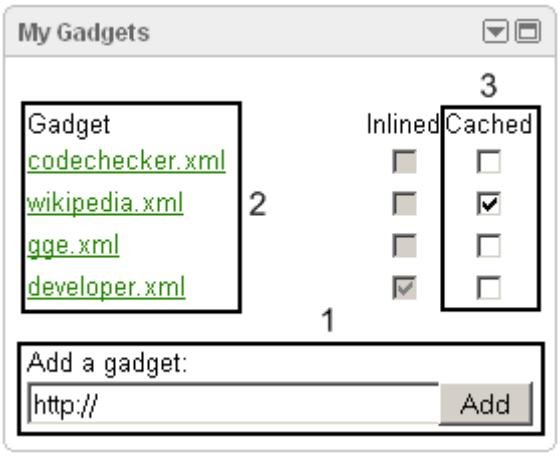
3.3. Sustav za pohranjivanje i objavljivanje udomljenika

Shema sustava za pohranjivanje i objavljivanje udomljenika prikazana je na slici 3.4. Sustav za pohranjivanje i objavljivanje udomljenika sastoji se od poslužitelja udomljenika i posredničkih poslužitelja povezanih s preglednikom web stranica na korisnikovom računalu.



Slika 3.4. Shema sustava za pohranjivanje i objavljivanje udomljenika

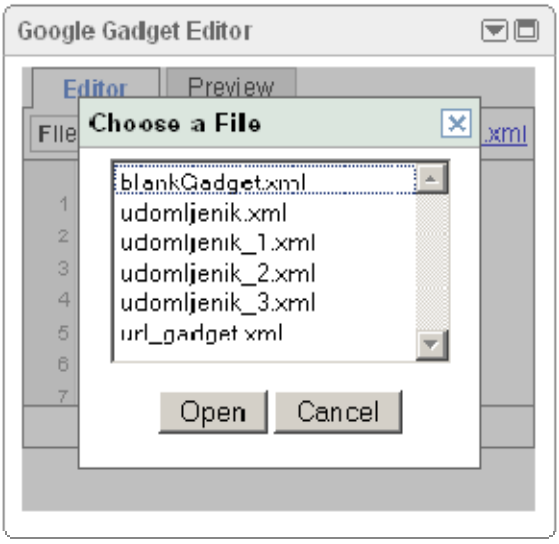
Upravljanje sustavom za pohranjivanje i objavljivanje udomljenika zasnovano je na naredbama *Open/Save/Delete* (hrv. *otvori/spremi/obriši*) te naredbi *Publish* (hrv. *objavi*) udomljenika *Google Gadgets Editor* (slika 3.5.). Kako je spomenutim udomljenikom moguće objavljivati samo *XML* datoteke koje je napisao korisnik, dodatnim udomljenikom *My Gadgets* omogućeno je dodavanje stranih, vanjskih udomljenika u korisnikov primjenski sustav. Udomljenikom *My Gadgets* podržano je dodavanje *XML* datoteke udomljenika sa zadane *URL* adrese vanjskog poslužitelja (1), dostupne su informacije o udomljenicima u primjenskom sustavu (2) i omogućeno je upravljanje načinom dohvata udomljenika (3).



Slika 3.5. Primjer udomljenika *My Gadgets*

Poslužitelji udomljenika dijele se u dvije kategorije, s razlikom u načinu pohrane i posluživanja udomljenika. U prvoj kategoriji poslužitelja nalaze se *Google* poslužitelji namijenjeni pohrani i posluživanju korisničkih udomljenika izgrađenih upotrebom

udomljenika *Google Gadget Editor*. Poslužitelji ove kategorije su poslužitelji korisničkog sadržaja. Kako bi se određenom korisniku poslužila udomiteljska stranica s odabranim udomljenicima, poslužitelji trebaju odrediti koje su udomiteljske stranice u vlasništvu pojedinog korisnika. Sustav *iGoogle* raspoznaje korisnike te organizira i bilježi korisničke postavke na osnovi korisničkih računa za prijavu (engl. *user account*). Korisnički računi primjenjuju se prilikom organiziranja i pohranjivanja, a potom i posluživanja udomljenika korisnicima. Na slici 3.6. prikazan je primjer popisa pohranjenih udomljenika raspoloživih za uređivanje udomljenikom *Google Gadget Editor*.



Slika 3.6. Popis pohranjenih udomljenika

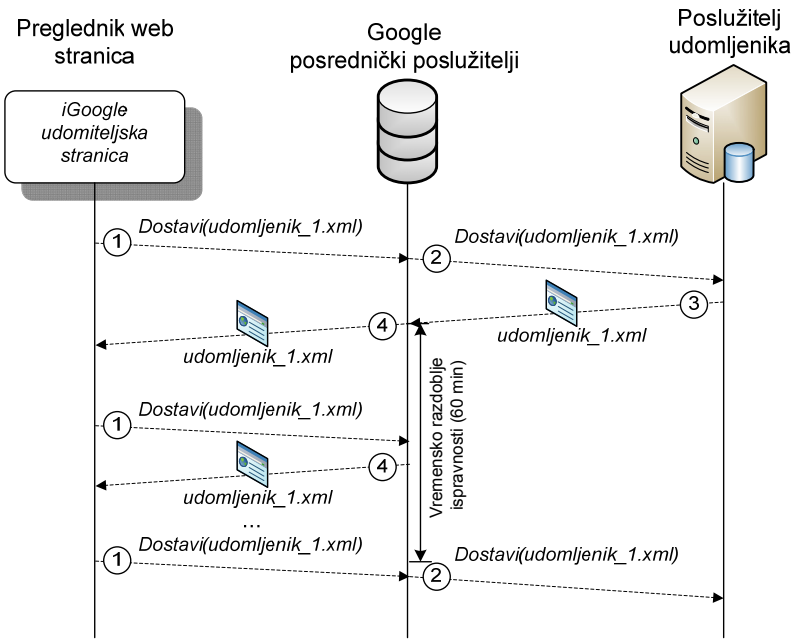
U drugoj kategoriji poslužitelja nalaze se poslužitelji općeg sadržaja, primjerice odlagališni (engl. *repository*) poslužitelji. Poslužitelji općeg sadržaja najčešće podržavaju dva načina pristupa pohranjenom sadržaju namijenjenom posluživanju: pristup s pravom čitanja i pisanja te pristup s pravom čitanja.

Pristup s pravom čitanja i pisanja korisnicima omogućava neograničeno uređivanje pohranjenog sadržaja. Kako bi poslužitelj omogućio korisniku pristup s pravom čitanja i pisanja, korisnik se mora predstaviti poslužitelju upisivanjem ispravnog korisničkog imena i zaporke. Pristup s pravom čitanja i pisanja moguće je ostvariti korištenjem web preglednika, no vrlo često se ostvaruje korištenjem primjenskih sustava posebne namjene koji podržavaju automatiziranu prijavu korisnika na poslužitelj i jednostavno upravljanje pohranjenim sadržajem na poslužitelju.

Pristup s pravom čitanja korisnicima omogućava pristupanje sadržaju poslužitelja bez prethodne korisničke prijave, no istovremeno ne dopušta izmjenu sadržaja na poslužitelju. Ovim načinom pristupa ostvarena je odgovarajuća razina dostupnosti sadržaja na poslužitelju uz zadržanu postojanost sadržaja. Pristup s pravom čitanja pokazao se pogodnim za javno objavljivanje različitog sadržaja, uključujući *XML* datoteke udomljenika kojima je moguće pristupiti na temelju poznate *URL* adrese uz korištenje web preglednika ili udomljenika *My Gadgets*.

Poslužitelj udomljenika iz perspektive korisnika primjenskih sustava ispunjava dvostruku ulogu. Poslužitelj udomljenika dohvaća i pohranjuje udomljenike tijekom njihove izgradnje, čime predstavlja potporu izgradnji udomljenika. S druge strane, poslužitelj udomljenika poslužuje udomljenike prilikom njihovog objavljivanja u udomiteljskoj stranici, čime predstavlja potporu objavljivanju udomljenika. Radne karakteristike poslužitelja udomljenika usklađene su s ispunjenjem obje uloge, no poteškoće nastaju postavljanjem udomljenika u imenik udomljenika, odnosno javnim objavljivanjem udomljenika. Javno objavljen udomljenik dostupan je velikom broju korisnika, što je uzrok mogućem dovođenju poslužitelja u stanje primitka neočekivano velikog broj zahtjeva za posluživanjem udomljenika, odnosno stanje prekomjernog opterećenja. Dovođenje poslužitelja u stanje prekomjernog opterećenja nerijetko dovodi do djelomičnog ili potpunog gubitka funkcionalnosti i ispada poslužitelja.

S ciljem smanjenja opterećenja poslužitelja udomljenika uveden je određeni broj posredničkih poslužitelja. Posrednički poslužitelji svojim radnim karakteristikama i mehanizmom međusobnog raspodjeljivanja opterećenja (engl. *load balancing*) učinkovito sudjeluju u održavanju dostupnosti objavljenih udomljenika i preuzimanju suvišnog opterećenja od poslužitelja udomljenika. Slika 3.7. prikazuje postupak dostavljanja udomljenika u četiri koraka.



Slika 3.7. Postupak dostavljanja udomljenika u četiri koraka

U prvom koraku korisnik iz primjenskog sustava upućuje zahtjev za objavljivanjem udomljenika `udomljenik_1.xml` prema posredničkom poslužitelju (1). Posrednički poslužitelj prosljeđuje primljeni zahtjev prema poslužitelju udomljenika (2). Poslužitelj udomljenika šalje traženi udomljenik prema posredničkom poslužitelju (3). Posrednički poslužitelj bilježi porijeklo, vremensko razdoblje ispravnosti (vremensko razdoblje ispravnosti *iGoogle* udomljenika iznosi 60 minuta) i naziv udomljenika u bazu udomljenika, pohranjuje primljeni udomljenik u privremenu memoriju (engl. *caching*) i prosljeđuje udomljenika prema korisniku (4). Tijekom trajanja vremenskog razdoblja ispravnosti udomljenika ne obavljaju se koraci (2) i (3), već posrednički poslužitelj preuzima funkcionalnost objavljivanja udomljenika od poslužitelja udomljenika, odnosno prima zahtjeve svih korisnika i odgovara na zahtjeve. Po isteku vremenskog razdoblja ispravnosti udomljenika odbacuje se pohranjeni udomljenik i ponavlja se cijeli postupak u 4 koraka.

Smanjenjem opterećenja ostvaren je pozitivan učinak na poslužitelj udomljenika i dostupnost udomljenika, međutim istodobno je uvedena poteškoća u postupak izgradnje udomljenika. Prije isteka vremenskog razdoblja ispravnosti udomljenici se poslužuju s posredničkog poslužitelja, a dohvaćanje i pohranjivanje udomljenika u postupku izgradnje događa se na poslužitelju udomljenika. Ispitivanje funkcionalnosti nove inačice udomljenika u postupku izgradnje moguće je provesti nakon objavljivanja udomljenika, no

postupak objavljivanja pokreće se tek nakon odbacivanja privremeno pohranjenog udomljenika na posredničkom poslužitelju. Spomenuta poteškoća uočljiva je u obliku zakašnjelog objavljivanja te znatno doprinosi usporavanju postupka izgradnje udomljenika. Kako bi utjecao na čekanje prije objavljivanja, korisnik može onemogućiti značajku vremenske ispravnosti odabranog udomljenika i time osigurati dohvat udomljenika izravno s poslužitelja udomljenika. Omogućavanje ili onemogućavanje značajke vremenske ispravnosti udomljenika primjenjivo je na primjenski sustav u vlasništvu korisnika te nema utjecaj na primjenske sustave drugih korisnika, a ostvaruje se udomljenikom *My Gadgets* prikazanim na slici 3.5., dio (3).

3.4. Struktura udomiteljske stranice i izvođenje udomljenika

Udomiteljska stranica sadrži primjenske sustave zasnovane na udomljenicima. Na sadržaj udomiteljske stranice moguće je izravno utjecati promjenom sadržaja njene *DOM* strukture, građene od hijerarhijski organiziranih čvorova *HTML* elemenata. Korisnik može utjecati na sadržaj udomiteljske stranice dodavanjem, izmjenom ili brisanjem udomljenika iz primjenskog sustava, što se preslikava na sadržaj *DOM* strukture udomiteljske stranice kao dodavanje, izmjena ili uklanjanje odgovarajućeg čvora udomljenika. Dodavanje čvora udomljenika u *DOM* strukturu udomiteljske stranice potaknuto je korisnikovim dodavanjem udomljenika u udomiteljsku stranicu i odvija se bez daljnjeg korisnikovog djelovanja.

Struktura čvora udomljenika sastoji se od spremnika udomljenika u koji su ugniježđene omotnica udomljenika i definicija udomljenika. Spremnik udomljenika izgrađen je na osnovi predloška pohranjenog u korisničkim postavkama, a sadrži općeniti dio grafičkog sučelja udomljenika i infrastrukturu udomljenika, primjerice gumbe za smanjenje i povećanje udomljenika. Definicija udomljenika izgrađena je sukladno sadržaju *XML* datoteke kojom je definiran izgled i funkcionalnost udomljenika. Oblikom omotnice određeni su način ugradnje, komunikacijske mogućnosti i vrsta udomljenika.

Omotnica starije inačice udomljenika građena je od *HTML* oznake `<div>`. *HTML* oznaka `<div>` ne odvaja sadržaj od okoline, što udomljeniku omogućava međudjelovanje s udomiteljskom stranicom i ostalim istovrsnim udomljenicima. Udomljenici starije inačice dijele zajednički programski kontekst i čine cjelinu s udomiteljskom stranicom, zbog čega

nose naziv *umetnuti* (engl. *inline*) *udomljenici*. Nepoznavanje funkcionalnosti javno dostupnih umetnutih udomljenika u sprezi s neograničenim pristupom udomiteljskoj stranici predstavlja sigurnosni problem zbog rizika od neovlaštenog upravljanja udomljenicima u korisnikovoj udomiteljskoj stranici. Kako bi se umanjio sigurnosni problem, uvedena je nova inačica omotnice udomljenika.

Omotnica nove inačice udomljenika građena je od *HTML* oznake `<iframe>`. Za razliku od oznake prethodne omotnice udomljenika, *HTML* oznaka `<iframe>` odjeljuje svoj sadržaj od okoline i sprječava međudjelovanje nove inačice udomljenika s primjenskom stranicom i drugim udomljenicima, zbog čega udomljenici novije inačice nose naziv *odijeljeni udomljenici*. Programski kontekst odijeljenih udomljenika odvojen je od programskog konteksta udomiteljske web stranice i drugih udomljenika, čime je postignuta potrebna razina sigurnosti udomljenika i izbjegnut je rizik od neovlaštenog upravljanja drugim udomljenicima.

3.5. Komunikacijska infrastruktura

Odvajanjem programskih konteksta udomljenika i onemogućavanjem neposredne razmjene podataka među udomljenicima onemogućeno je okupljanje funkcionalnosti udomljenika prilikom izgradnje složenih primjenskih sustava. Uvođenjem odgovarajuće komunikacijske infrastrukture u složeni primjenski sustav moguće je provesti okupljanje funkcionalnosti udomljenika uz precizno upravljanje komunikacijskim aktivnostima među udomljenicima.

Komunikacijska infrastruktura pogodna za povezivanje udomljenika ne smije umanjiti razinu sigurnosti udomljenika te istovremeno mora podržati odgovarajuću razmjenu podataka između udomljenika. Kako bi se ispunili prethodni zahtjevi, komunikacijska infrastruktura ostvarena je korištenjem komunikacijskih mehanizama: mehanizmom objave/pretplate (engl. *publish/subscribe*) i mehanizmom poziva udaljene procedure (engl. *remote procedure call*).

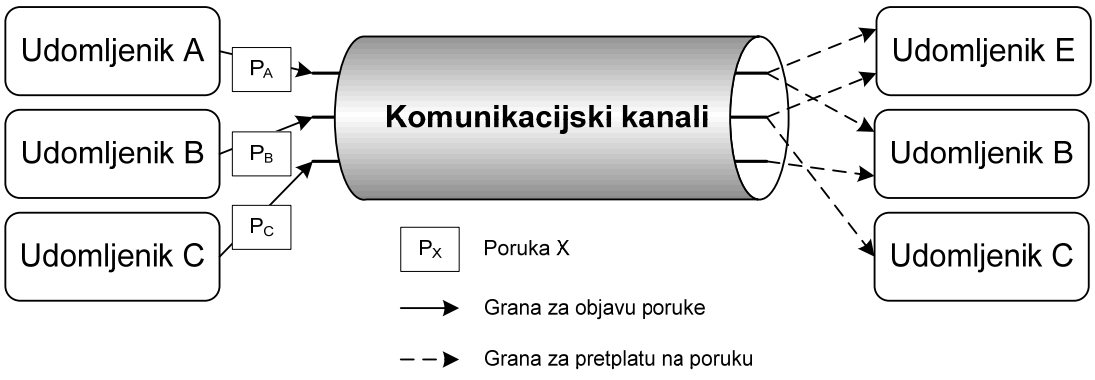
3.5.1. Mehanizam objave/pretplate

Mehanizam objave/pretplate odvojenim *HTML* udomljenicima omogućava komunikaciju zasnovanu na međusobnom obavještavanju, a sastoji se od udomljenika objavljivača, udomljenika pretplatnika, komunikacijskog kanala i poruke. Mehanizam objave/pretplate dostupan je uz odgovarajuću prilagodbu *XML* datoteke udomljenika. Prilagodba *XML* datoteke udomljenika objavljivača provodi se dodavanjem nove korisničke postavke, što je prikazano primjerom 3.3.

```
...
<UserPref name="test000"
   default_value="20"
   publish="true" />
...
```

Primjer 3.3. Dodana korisnička postavka udomljenika objavljivača

Izjednačavanjem naziva dodane korisničke postavke i naziva komunikacijskog kanala ostvareno je povezivanje udomljenika objavljivača i komunikacijskog kanala. Korisnička postavka osim naziva komunikacijskog kanala sadrži svojstvo `default_value`, kojim je određena početna vrijednost korisničke postavke, odnosno sadržaj poruke za objavu. Vrijednost svojstva `publish` utječe na mogućnost, odnosno nemogućnost objave poruke kroz komunikacijski kanal. Moguće vrijednosti svojstva `publish` su `true` (hrv. *istina*) i `false` (hrv. *laž*).



Slika 3.8. Povezivanje udomljenika mehanizmom objave/pretplate

Prilagodba *XML* datoteke udomljenika pretplatnika provodi se dodavanjem nove korisničke postavke, što je prikazano primjerom 3.4. Izjednačavanjem naziva dodane korisničke postavke i naziva komunikacijskog kanala ostvareno je povezivanje

udomljenika pretplatnika i komunikacijskog kanala. Korisnička postavka osim naziva komunikacijskog kanala sadrži svojstvo `default_value`, kojim je određena početna vrijednost korisničke postavke. Vrijednost svojstva `listen` utječe na mogućnost, odnosno nemogućnost primitka poruke kroz komunikacijski kanal. Moguće vrijednosti svojstva `listen` su `true` (hrv. *istina*) i `false` (hrv. *laž*). *XML* datoteka udomljenika pretplatnika sadrži svojstvo `on_change`, kojem je pridružena odzivna funkcija `odzivnaFunkcija` ugrađena u udomljenik. Odzivna funkcija poziva se nakon svake promjene (početne) vrijednosti korisničke postavke, a funkcionalnost odzivne funkcije može se razlikovati u svakom od udomljenika pretplatnika.

```
...
<UserPref name="test000"
   default_value="26"
   listen="true"
   on_change="odzivnaFunkcija" />
...
```

Primjer 3.4. Dodana korisnička postavka udomljenika pretplatnika

Završetkom prilagodbe *XML* datoteka svih udomljenika namijenjenih povezivanju mehanizmom objave/pretplate, moguće je ostvariti komunikaciju, odnosno razmjenu poruka između *HTML* udomljenika unutar udomiteljske stranice, kao što je prikazano slikom 3.8. Udomljenici pretplatnici su posredstvom komunikacijskog kanala povezani s udomljenikom objavljivačem. Udomljenik objavljivač objavljuje promijenjenu vrijednost svoje korisničke postavke slanjem poruke, a primitak poruke s izmijenjenom vrijednosti kod udomljenika pretplatnika uzrokuje izmjenu njihovih vrijednosti korisničke postavke i pozivanje definiranih odzivnih funkcija.

3.5.2. Mehanizam poziva udaljene procedure

Mehanizam poziva udaljene procedure, odnosno funkcije drugi je spomenuti mehanizam uspostavljanja komunikacije između odijeljenih udomljenika ili između odijeljenog udomljenika i udomiteljske stranice, a idejno je zasnovan na istoimenom mehanizmu primjenjivanom u raspodijeljenim računalnim sustavima. Pozivom udaljene procedure udomljeniku je omogućen pristup funkcionalnostima iz programskog konteksta drugog odijeljenog udomljenika, čime je omogućeno okupljanje funkcionalnosti i izgradnja složenog primjenskog sustava. Kako bi se očuvala razina sigurnosti nastala odvajanjem

udomljenika, mehanizam poziva udaljene procedure oslanja se na pojedinačno i izričito izlaganje procedura promatranog, matičnog udomljenika. Pojedinačnim izlaganjem procedura prema vanjskoj okolini, odnosno oblikovanjem komunikacijskog kanala, korisniku je omogućeno precizno oblikovanje skupa udaljenih procedura dostupnih pozivajućim udomljenicima. Ovim pristupom korisnik je u mogućnosti izričito dozvoliti ili ne dozvoliti korištenje odabranih funkcionalnosti izvan matičnog udomljenika.

Izlaganje procedure provodi se u matičnom udomljeniku, nakon čega udaljena procedura kroz komunikacijski kanal postaje dostupna drugim udomljenicima. Proceduru je moguće izložiti kao zadanu (engl. *default*) udaljenu proceduru i kao imenovanu udaljenu proceduru. Udomljenik upućuje poziv zadanoj udaljenoj proceduri ukoliko pri pozivu udaljene procedure nije naveden pseudonim udaljene procedure. Primjeri 3.5. i 3.6. prikazuju izlaganje zadane i imenovane udaljene procedure, te pozive upućene zadanoj i imenovanoj proceduri.

```
// izlaganje zadane procedure
// u matičnom udomljeniku
register(udaljenaProcedura)

// poziv udaljene procedure
// iz ostalih udomljenika
call(oznakaMaticnogUdomljenika, odzivnaFunkcija,
     argumentiUdaljeneProcedure)
```

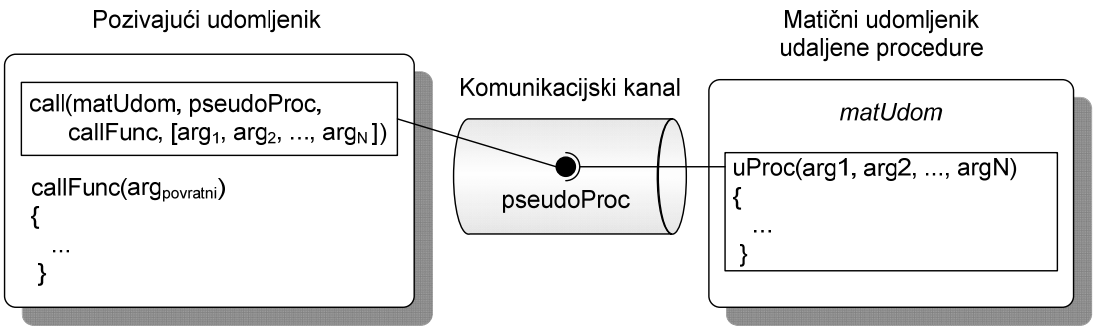
Primjer 3.5. Izlaganje i poziv zadane udaljene procedure

Udaljenu proceduru moguće je imenovati prilikom izlaganja, a pseudonim udaljene procedure može se razlikovati od imena udaljene procedure. Prilikom pozivanja udaljene procedure moguće je navesti odzivnu funkciju, kojom se obrađuje povratna vrijednost udaljene procedure.

```
// izlaganje imenovane procedure matičnog udomljenika
register(pseudonimUdaljeneProcedure,
         udaljenaProcedura)

// poziv imenovane udaljene procedure
// iz ostalih udomljenika
call(oznakaMaticnogUdomljenika,
pseudonimUdaljeneProcedure,
     odzivnaFunkcija, argumentiUdaljeneProcedure)
```

Primjer 3.6. Izlaganje i poziv imenovane udaljene procedure



Slika 3.9. Poziv udaljene funkcije uProc

Na slici 3.9. prikazan je poziv udaljene procedure `uProc` upućen s pozivajućeg udomljenika. Parametri poziva udaljene procedure su naziv matičnog udomljenika udaljene procedure `matUdom`, pseudonim udaljene procedure `pseudoProc`, odzivna funkcija `callFunc` za obradu povratnog rezultata udaljene procedure i argumenti udaljene procedure. Zbog preglednosti slike prikazan je jedan pozivajući udomljenik i jedan pseudonim udaljene procedure, no u mehanizmu poziva udaljene procedure može sudjelovati gotovo neograničen broj udomljenika i udaljenih procedura. Primjer 3.7. prikazuje postupak uklanjanja zadane i imenovane udaljene procedure iz komunikacijskog kanala.

```
// uklanjanje zadane udaljene procedure
unregisterDefault()

// uklanjanje imenovane udaljene procedure
unregister(pseudonimUdaljeneProcedure)
```

Primjer 3.7. Uklanjanje udaljenih procedura iz komunikacijskog kanala

4. Arhitektura korisničkog sučelja

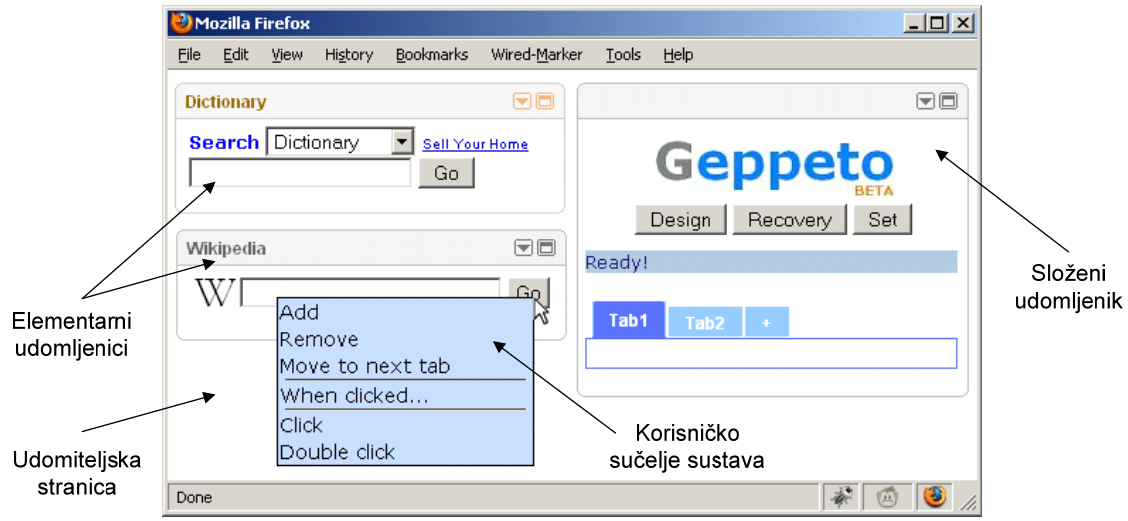
Primjenski sustav zasnovan na udomljeniku sastoji se od ugrađene funkcionalnosti i grafičkog korisničkog sučelja putem kojeg je podržano međudjelovanje korisnika s udomljenikom. Korisnik pomoću upravljačkih elemenata, primjerice tipki (engl. *button*), polja za označavanje (engl. *checkbox*) ili padajućeg izbornika (engl. *drop-down list*) upravlja ugrađenom funkcionalnosti udomljenika, dok udomljenik pomoću podatkovnih elemenata grafičkog korisničkog sučelja, primjerice tekstovnog prostora (engl. *text area*) obavještava korisnika o svom radnom stanju, rezultatu izvođenja određene operacije nad zadanim skupom podataka ili pojavi greške u postupku obrade podataka. Opisano načelo međudjelovanja korisniku omogućava brzo upoznavanje funkcionalnosti udomljenika i jednostavno korištenje udomljenikom.

Nasuprot korištenju udomljenika putem grafičkog korisničkog sučelja, izgradnja udomljenika od korisnika, potrošača zahtijeva poznavanje određenih programskih jezika kojima korisnik definira funkcionalnost i gradi grafičko korisničko sučelje udomljenika. Izgradnja udomljenika pisanjem naredbi nepogodna je za većinu korisnika, koji nisu upoznati s postupcima programiranja. Kako bi se način izgradnje udomljenika približio korisniku naviknutom na način korištenja udomljenika, ostvareno je korisničko sučelje za potrošaču prilagođeno programiranje. Korisničko sučelje je dio sustava za potrošaču prilagođeno programiranje, koji omogućava izgradnju novog udomljenika korištenjem elemenata grafičkog korisničkog sučelja postojećih udomljenika. Korisničko sučelje za potrošaču prilagođeno programiranje nad skupom udomljenika ostvareno je u obliku kontekstno ovisnog izbornika s naredbama, koji korisniku nudi izbor svih dostupnih naredbi za odabrani element grafičkog korisničkog sučelja udomljenika.

U ovom poglavlju obrađene su teme vezane uz arhitekturu korisničkog sučelja sustava za potrošaču prilagođeno programiranje nad skupom udomljenika. Kroz sljedeća potpoglavlja opisani su dijelovi arhitekture, način njihovog povezivanja te njihove uloge u postupku stvaranja i upotrebe korisničkog sučelja za programiranje nad skupom udomljenika.

4.1. Pregled arhitekture korisničkog sučelja

Arhitektura korisničkog sučelja za potrošaču prilagođeno programiranje sastoji se od skupa elementarnih udomljenika, udomiteljske stranice, složenog udomljenika i korisničkog sučelja sustava za potrošaču prilagođeno programiranje. Proširena funkcionalnost obje vrste udomljenika sastoji se u izvedbi odabira grafičkog elementa udomljenika i prikupljanju pripadajućih informacija. Proširenjem funkcionalnosti udomiteljske stranice omogućen je prikaz korisničkog sučelja i definiranje radnje nad odabranim grafičkim elementom udomljenika. Proširenje funkcionalnosti složenog udomljenika omogućava oblikovanje programske instrukcije i njeno spremanje u popis programskih instrukcija. Udomljenici komuniciraju s udomiteljskom stranicom putem komunikacijskih kanala ostvarenih proširenim funkcionalnostima. Slikom 4.1. prikazan je primjer udomiteljske stranice s udomljenicima.



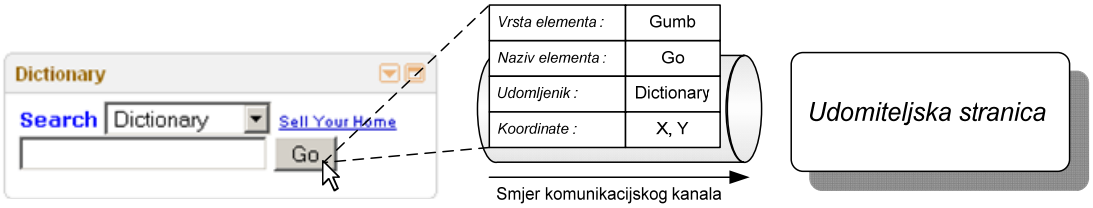
Slika 4.1. Udomiteljska stranica s udomljenicima

4.2. Elementarni udomljenici

U elementarnim udomljenicima izvodi se prvi korak postupka potrošačkog programiranja. Ugrađenom funkcionalnosti elementarnih udomljenika nije moguće podržati postupak potrošačkog programiranja, stoga je elementarne udomljenike potrebno proširiti dodatnom funkcionalnosti. Dodatna funkcionalnost obuhvaća definiranje aktivnosti kojom korisnik odabire element grafičkog korisničkog sučelja elementarnog

udomljenika i prikupljanje informacija o odabranom elementu grafičkog korisničkog sučelja.

Odabirom elementa grafičkog korisničkog sučelja elementarnog udomljenika započinje prikupljanje informacija o odabranom elementu. Prikupljene informacije obuhvaćaju naziv elementarnog udomljenika odabranog elementa, vrstu i naziv odabranog elementa, primjerice gumb ili polje za unos teksta i koordinate pokazivača miša u trenutku odabira elementa (engl. *cursor coordinates*). Prikupljene informacije dostavljaju se komunikacijskim kanalom udomiteljske stranice u njen programski kontekst, gdje započinje drugi korak postupka potrošačkog programiranja. Slika 4.2. prikazuje nastajanje i prijenos prikupljenih informacija u udomiteljsku stranicu.



Slika 4.2. Prijenos prikupljenih informacija u udomiteljsku stranicu

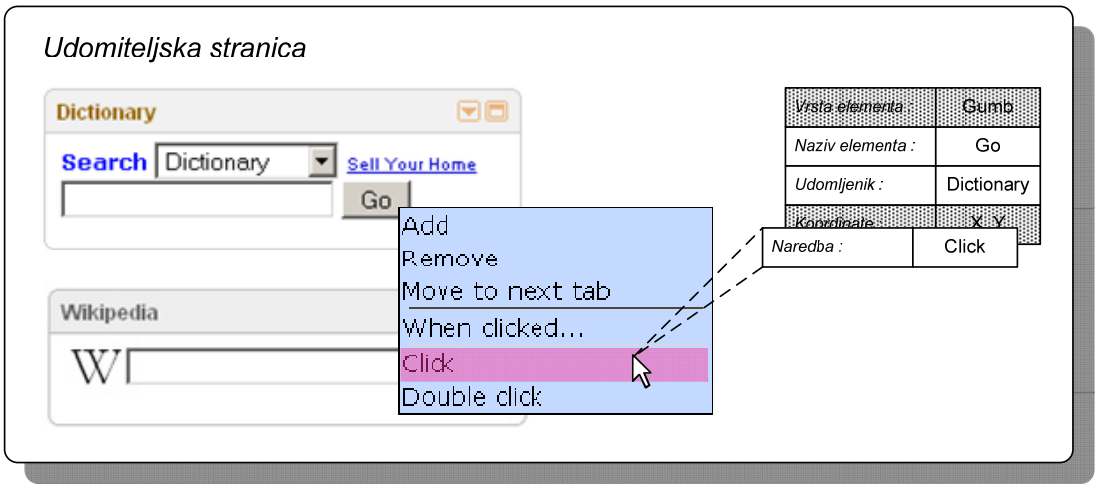
4.3. Udomiteljska stranica

U udomiteljskoj stranici izvodi se drugi korak postupka potrošačkog programiranja. Ugrađena funkcionalnost udomiteljske stranice sastoji se u obuhvaćanju udomljenika u isti logički i vizualni prostor. Ugrađenom funkcionalnosti udomiteljske stranice nije moguće podržati postupak potrošačkog programiranja, stoga je udomiteljsku stranicu potrebno proširiti dodatnim funkcionalnostima.

Udomiteljska stranica na osnovi dodatne funkcionalnosti stvara komunikacijski kanal za razmjenu podataka s ostalim udomljenicima, kojim preuzima prikupljene informacije od elementarnog udomljenika i unosi ih u svoj programski kontekst. Dodatna funkcionalnost udomiteljskoj stranici omogućava pripremu i prikaz kontekstno osjetljivog izbornika s naredbama.

Na osnovi prikupljenih informacija, odnosno sukladno informaciji o vrsti odabranog elementa grafičkog korisničkog sučelja elementarnog udomljenika oblikovan je skup dostupnih naredbi za odabrani element. Prema skupu dostupnih naredbi pripremljen je i

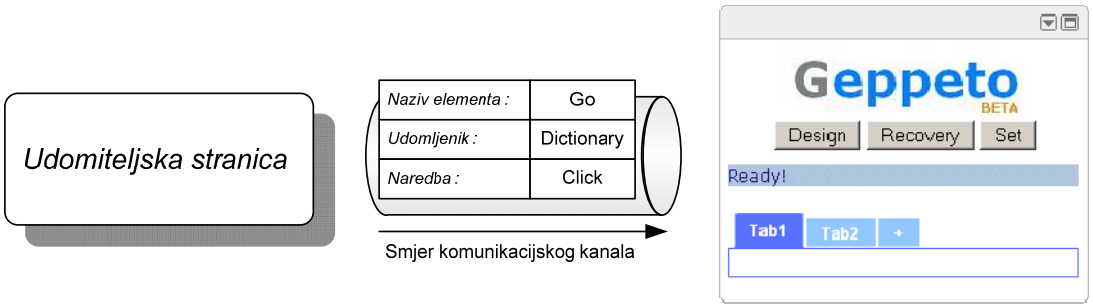
oblikovan kontekstno osjetljivi izbornik. Kontekstno osjetljivi izbornik potrebno je prikazati na mjestu odabira elementa grafičkog korisničkog sučelja elementarnog udomljenika, stoga su na osnovi informacija o koordinatama pokazivača miša određene koordinate za prikaz izbornika. Elementi kontekstno osjetljivog izbornika su grafički elementi u ulozi dostupnih naredbi. Za svaki grafički element definirana je aktivnost kojom korisnik odabire pripadajuću naredbu. Na slici 4.3. prikazano je prikupljanje informacije o odabranoj naredbi `click` nad gumbom *Go* udomljenika *Dictionary*.



Slika 4.4.3. Prikupljanje informacije o odabranoj naredbi

Odabirom naredbe iz kontekstno ovisnog izbornika nastaje informacija o odabranoj naredbi, koja se sastoji od naziva odabrane naredbe. Djelomično odstupanje javlja se kod naredbe `Type in...` dostupne pri odabiru polja za unos teksta, uz čiju se informaciju o odabiru dodatno veže zadani tekst. Informacija o odabranoj naredbi kontekstno osjetljivog izbornika pridružena je prethodno prikupljenim informacija o odabranom elementu grafičkog korisničkog sučelja elementarnog udomljenika.

Informacije o vrsti odabranog elementa i koordinatama pokazivača miša odbacuju se nakon prikazivanja kontekstno ovisnog izbornika. Prikupljene informacije na kraju drugog koraka obuhvaćaju nazive elementarnog udomljenika i u njemu odabranog elementa grafičkog korisničkog sučelja i informaciju o odabranoj naredbi kontekstno ovisnog izbornika. Prikupljene informacije dostavljaju se komunikacijskim kanalom složenog (engl. *composite*) udomljenika u njegov programski kontekst (slika 4.4.), gdje započinje treći, završni korak postupka potrošačkog programiranja.



Slika 4.4. Prijenos prikupljenih informacija u složeni udomljenik

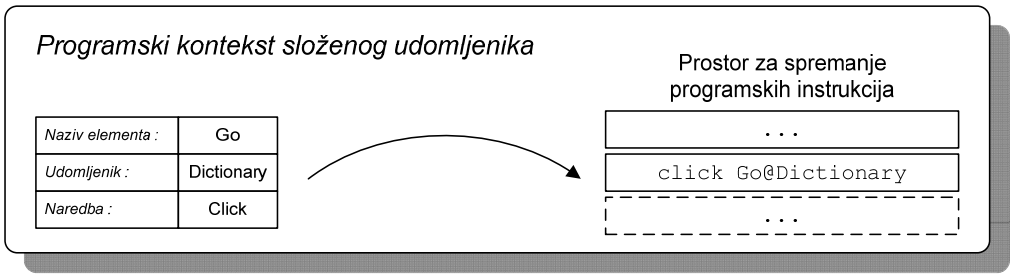
4.4. Složeni udomljenik

U složenom udomljeniku izvodi se treći korak postupka potrošačkog programiranja. Složeni udomljenik namijenjen je prikazu grafičkog korisničkog sučelja korisnikovog udomljenika i izvođenju programskih instrukcija. Ugrađena funkcionalnost složenog udomljenika proširena je dodatnom funkcionalnosti za potporu postupku potrošačkog programiranja, kojom je obuhvaćeno oblikovanje programske instrukcije.

```
oznakaNaredbe  nazivElementa@nazivUdomljenika
```

Primjer 4.8. Predložak oblikovane programska instrukcija

Složeni udomljenik stvara komunikacijski kanal za razmjenu podataka s ostalim udomljenicima. Od udomiteljske stranice preuzima prikupljene informacije o odabranom elementu grafičkog korisničkog sučelja elementarnog udomljenika i odabranoj naredbi kontekstno ovisnog izbornika te ih unosi u svoj programski kontekst. Na osnovi prikupljenih informacija, odnosno naziva elementarnog udomljenika, naziva odabranog elementa i naziva odabrane naredbe oblikovana je programska instrukcija.



Slika 4.5. Oblikovanje i spremanje programske instrukcije

Postupak oblikovanja programske instrukcije izveden je sastavljanjem znakovnog niza na sljedeći način. Naziv odabrane naredbe kontekstno ovisnog izbornika zamijenjen je odgovarajućom oznakom `oznakaNaredbe` i dodan na početak znakovnog niza. Znakovnom nizu dodan je naziv odabranog elementa grafičkog korisničkog sučelja elementarnog udomljenika `nazivElementa`. Znakovni niz završava simbolom za odjeljivanje `@` i nazivom elementarnog udomljenika `nazivUdomljenika`. Primjerom 4.1. pokazan je predložak oblikovane programske instrukcije. Slika 4.5. prikazuje oblikovanje programske instrukcije na osnovi prikupljenih informacija.

Tablica 4.1. Naredbe i primjeri oblikovanih instrukcija

Naredba	Oblikovana programska instrukcija	Opis programske instrukcije
Add	(ne oblikuje se)	Dodaje element sučelja u složeni udomljenik
Remove	(ne oblikuje se)	Uklanja element sučelja iz složenog udomljenika
Move to next tab	(ne oblikuje se)	Premješta element sučelja u susjedni prostor
When clicked...	`# nazivElementa@nazivUdomljenika`	Započinje izvođenje potrošačkog programa odabirom elementa
Type in... (tekst)	`"tekst" => nazivElementa@nazivUdomljenika`	Upisuje zadani tekst u tekstovno polje
Click	`click nazivElementa@nazivUdomljenika`	Odabire element grafičkog korisničkog sučelja
Double click	`doubleclick nazivElementa@nazivUdomljenika`	Odabire element grafičkog korisničkog sučelja dvostrukim klikom
Check	`check nazivElementa@nazivUdomljenika`	Omogućava kućicu za označavanje
Uncheck	`uncheck nazivElementa@nazivUdomljenika`	Onemogućava kućicu za označavanje
Select	`select nazivElementa@nazivUdomljenika`	Omogućava radio gumb
Copy	`nazivElementa@nazivUdomljenika => nazivElementa1@nazivUdomljenika1`	Zadaje izvorište vrijednosti za kopiranje
Paste		Zadaje odredište vrijednosti za kopiranje

U tablici 4.1. dani su opisi naredbi i primjeri programskih instrukcija. Dio naredbi izvodi se neposredno nakon odabira, stoga za takve naredbe nisu oblikovane programske instrukcije. Odstupanja od opisanog oblika programske instrukcije nastaju odabirom naredbi `Type in...` i `Copy/Paste`, čiji su oblici programskih instrukcija također navedeni u tablici 4.1.

5. Programska izvedba korisničkog sučelja

Programskom izvedbom korisničkog sučelja ostvaren je kontekstno osjetljiv plivajući izbornik za definiranje akcija nad elementima pripadajućih grafičkih korisničkih sučelja. Izvedeno korisničko sučelje dio je sustava za potrošaču prilagođeno programiranje kojim je krajnjem korisniku omogućena izgradnja i programiranje vlastitog udomljenika korištenjem grafičkih elemenata postojećih elementarnih udomljenika. Programska izvedba korisničkog sučelja sukladno radnoj okolini podijeljena je na tri dijela. Prvi dio programske izvedbe ostvaren je korištenjem ugrađene i proširene funkcionalnosti udomljenika iz skupa elementarnih udomljenika. Drugi dio programske izvedbe ostvaren je korištenjem ugrađene i proširene funkcionalnosti udomiteljske stranice. Treći dio programske izvedbe ostvaren je korištenjem ugrađene i proširene funkcionalnosti složenog udomljenika.

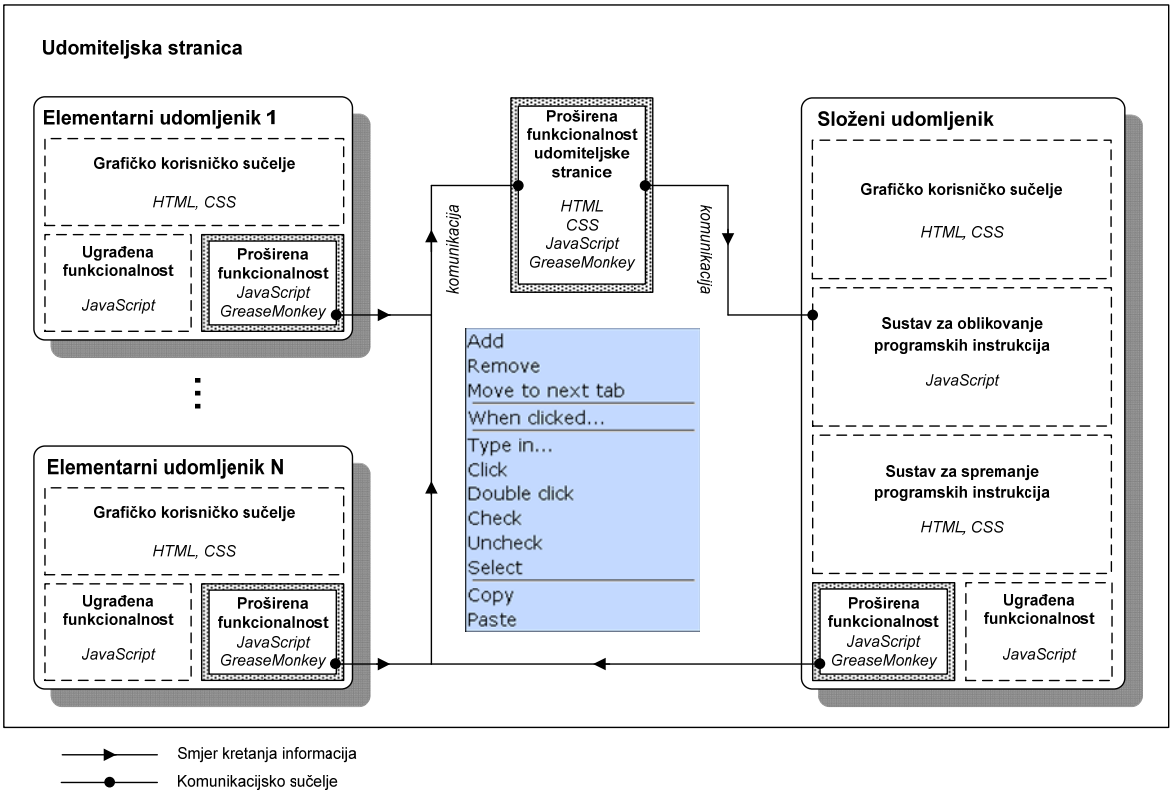
U sljedećim potpoglavljima dan je općeniti pregled programske izvedbe korisničkog sučelja i opisana su tri dijela programske izvedbe. Pregled programske izvedbe korisničkog sučelja daje sažete informacije o izvedbi korisničkog sučelja, korištenim programskim jezicima i načelu rada korisničkog izbornika. Opis prvog dijela programske izvedbe pokriva teme vezane uz ostvarenje odabira grafičkog elementa udomljenika. Opis drugog dijela programske izvedbe odnosi se na postupak odabira grafičkog elementa izbornika i prikaz kontekstno-osjetljivog izbornika. Opis trećeg dijela programske izvedbe izlaže informacije o postupku oblikovanja programske instrukcije.

5.1. Pregled programske izvedbe korisničkog sučelja

Ukupna funkcionalnost udomljenika ovisi o njegovoj ugrađenoj i proširenoj funkcionalnosti. Ugrađenom funkcionalnosti ostvarena je osnovna namjena elementarnog udomljenika, odnosno pružanje određene usluge korisniku. Kako bi elementarni udomljenici postali pogodni za postupak potrošaču prilagođenog programiranja, njihovoj ugrađenoj funkcionalnosti te ugrađenim funkcionalnostima udomiteljske stranice i složenog udomljenika dodana je proširena funkcionalnost. Proširena funkcionalnost definirana je korisničkom skriptom čiji je sadržaj posredstvom alata *GreaseMonkey* ugrađen u elementarne udomljenike, udomiteljsku stranicu i složeni udomljenik. Algoritmi i pripadajuće strukture podataka proširenih funkcionalnosti definirane su programskim

jezikom *JavaScript*. Kontekstno osjetljiv izbornik definiran je u korisničkoj skripti programskim jezicima *JavaScript*, *HTML* i *CSS*. Komunikacija među udomljenicima ostvarena je mehanizmom *PostMessage* ugrađenim u web preglednik *Mozilla Firefox*.

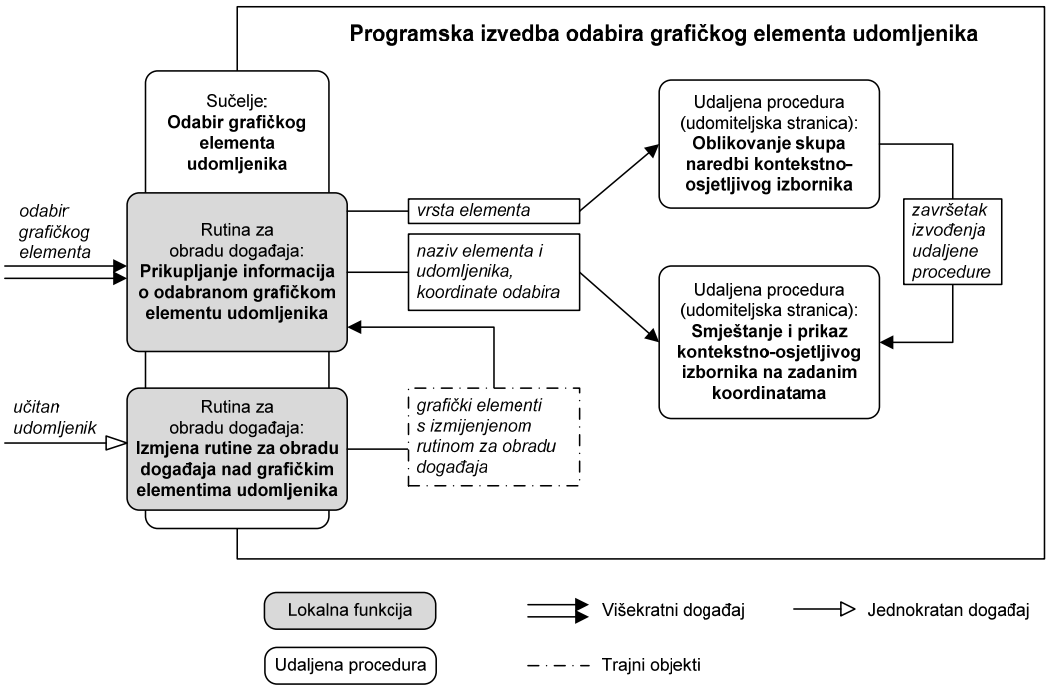
Prvim dijelom programske izvedbe ostvareno je definiranje korisnikove akcije za odabir grafičkog elementa elementarnog ili složenog udomljenika i prosljeđivanje informacija o odabranom grafičkom elementu prema udomiteljskoj stranici. Drugim dijelom programske izvedbe ostvareni su prikaz kontekstno-osjetljivog izbornika na odgovarajućim koordinatama udomiteljske stranice te prosljeđivanje informacija o odabranoj naredbi izbornika i odabranom grafičkom elementu prema složenom udomljeniku. Trećim dijelom programske izvedbe ostvareno je oblikovanje i spremanje programske instrukcije nastale na osnovi informacija o odabranoj naredbi izbornika i odabranom elementu grafičkog korisničkog sučelja elementarnog udomljenika. Slika 5.1. prikazuje povezanost dijelova programske izvedbe, pripadajuće programske jezike upotrijebljene u ostvarenju ugrađenih i proširenih funkcionalnosti te ostvareno korisničko sučelje u obliku kontekstno-osjetljivog izbornika.



Slika 5.1. Povezanost dijelova programskih izvedbi i pripadajući programski jezici

5.2. Odabir grafičkog elementa udomljenika

Prvi dio programske izvedbe korisničkog sučelja namijenjen je izvođenju na skupu elementarnih udomljenika, gdje je proširenom funkcionalnosti potrebno ostvariti prikupljanje i proslijeđivanje informacija o odabranom grafičkom elementu korisničkog sučelja udomljenika. Izvođenje započinje pojavom događaja `učitan udomljenik` kojim je označen kraj učitavanja udomljenika, odnosno dostupnost njihovih grafičkih elemenata. Nad grafičkim elementima udomljenika provedena je trajna izmjena zadane rutine za obradu događaja `odabir grafičkog elementa`. Izmjenom rutine za obradu događaja onemogućen je prikaz zadanog izbornika, a prikupljanje informacija o odabranom grafičkom elementu započinje pojavom događaja `odabir grafičkog elementa`. Postupak odabira grafičkog elementa prikazan je slikom 5.2.



Slika 5.2. Programska izvedba odabira grafičkog elementa udomljenika

Prikupljene informacije organizirane su u odgovarajućoj strukturi znakovnih nizova, a sadrže vrstu i naziv odabranog elementa, naziv roditeljskog udomljenika elementa i koordinate pokazivača miša zabilježene prilikom odabira grafičkog elementa. Završetkom prikupljanja informacija o odabranom elementu upućen je poziv udaljenoj proceduri `oblikujSkupNaredbi` za oblikovanje skupa naredbi kontekstno-osjetljivog izbornika. Udaljena procedura `oblikujSkupNaredbi` pripada proširenoj funkcionalnosti

udomiteljske stranice, a argument udaljene procedure sadrži informaciju o vrsti odabranog grafičkog elementa udomljenika.

Nakon provedenog oblikovanja skupa naredbi u udomiteljskoj stranici i završetka izvođenja pripadajuće udaljene procedure, upućen je poziv udaljenoj proceduri udomiteljske stranice `prikaziIzbornik` za smještanje i prikaz kontekstno-osjetljivog izbornika na zadanim koordinatama. Vrijednosti argumenata udaljene procedure `prikaziIzbornik` određene su informacijama o nazivu odabranog grafičkog elementa i roditeljskog udomljenika te koordinatama za prikaz kontekstno-osjetljivog izbornika. Završetkom izvođenja udaljene procedure `prikaziIzbornik` privremeno završava (engl. *suspend*) postupak odabira grafičkog elementa udomljenika. Postupak ulazi u sljedeće ponavljanje (engl. *iteration*) pojavom događaja `odabir grafičkog elementa`.

5.3. Definiranje radnje nad grafičkim elementom udomljenika

Drugi dio programske izvedbe korisničkog sučelja namijenjen je izvođenju u udomiteljskoj stranici, gdje je proširenom funkcionalnosti potrebno ostvariti definiranje radnje nad grafičkim elementom korisničkog sučelja udomljenika. Izvođenje započinje pojavom događaja `učitana udomiteljska stranica` kojim je označen kraj učitavanja udomiteljske stranice. Pripadajuća rutina za obradu događaja potaknuta pojavom događaja `učitana udomiteljska stranica` poziva funkciju `dodajElement`. Pozvana funkcija udomiteljskoj stranici dodaje kontekstno-osjetljivi izbornik i pripadajuće grafičke elemente u ulozi naredbi izbornika. Svakom grafičkom elementu kontekstno-osjetljivog izbornika pridružena je rutina za obradu događaja koja bilježi odabrani grafički element izbornika. Kontekstno-osjetljiv izbornik s grafičkim elementima i pridruženim rutinama za obradu događaja postaje trajan objekt u udomiteljskoj stranici, dostupan za korištenje u trenutnom i sljedećim ponavljanjima (engl. *iteration*). Kontekstno-osjetljiv izbornik ostvaren je u udomiteljskoj stranici kako bi se postigao plivajući obilazak svih udomljenika, a prije poziva funkcije `prikaziIzbornik` izbornik je skriven.

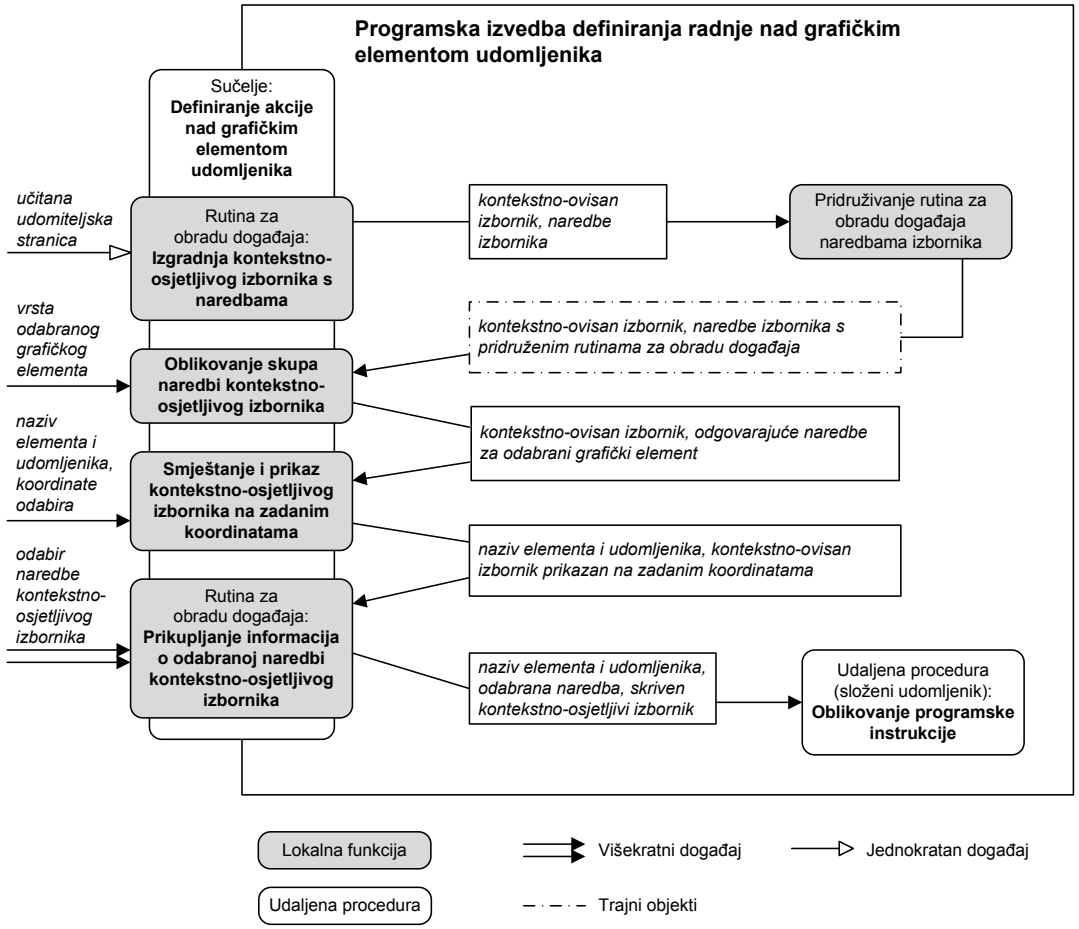
Udaljenim pozivom lokalne funkcije `oblikujSkupNaredbi` uz argument koji sadrži informaciju o vrsti odabranog grafičkog elementa, započinje postupak oblikovanja

skupa naredbi dostupnih za odabrani grafički element udomljenika. Definicije dostupnih naredbi za grafičke elemente udomljenika zapisane su jednodimenzionalnim poljima čiji su članovi znakovni nizovi. Znakovni nizovi predstavljaju oznake grafičkih elemenata izbornika koji odgovaraju skupu dostupnih naredbi za promatrani grafički element udomljenika. Vidljivost grafičkih elemenata izbornika određena je definicijom dostupnih naredbi za odabrani grafički element udomljenika. Rezultat poziva lokalne funkcije `oblikujSkupNaredbi` su uređene vidljivosti grafičkih elemenata izbornika. Primjerom 5.1. prikazana je definicija dostupnih naredbi za vrstu grafičkog elementa `button` (hrv. *tipka*).

```
elemType["button"] = ["add", "remove","send_to_back",
   "wfc", "click", "double_click"];
```

Primjer 5.1. Definicija dostupnih naredbi na vrstu grafičkog elementa button

Udaljeni poziv lokalne funkcije `prikaziIzbornik` uz argumente koji sadrže informacije o koordinatama odabira grafičkog elementa udomljenika, nazivu odabranog elementa i pripadajućeg roditeljskog udomljenika uzrokuje aktiviranje vidljivosti, odnosno prikaz kontekstno-osjetljivog izbornika na zadanim koordinatama. U kontekstno-osjetljivom izborniku vidljive su dostupne naredbe za odabran grafički element udomljenika. Odabirom naredbe kontekstno-osjetljivog izbornika, odnosno pojavom događaja `odabir naredbe kontekstno-osjetljivog izbornika` definirana je radnja nad grafičkim elementom udomljenika.



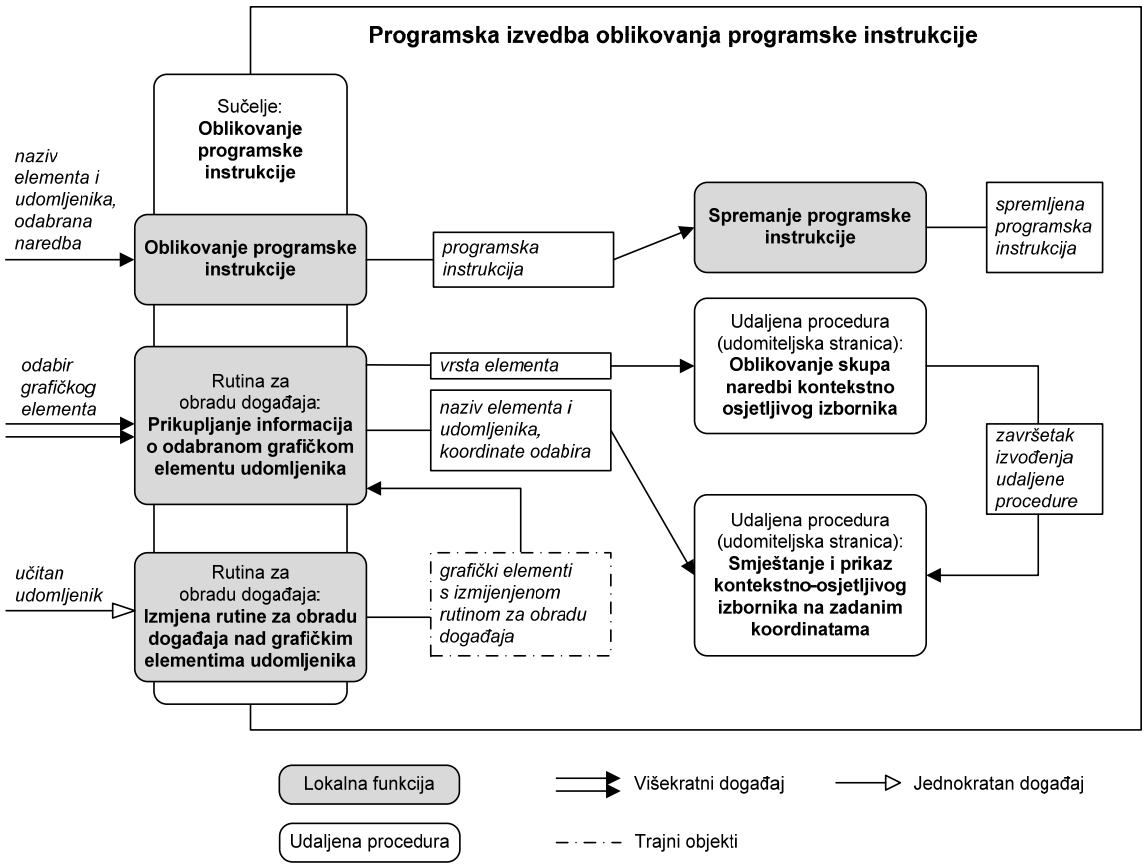
Slika 5.3. Programska izvedba definiranja radnje nad grafičkim elementom udomljenika

Informacija o definiranoj radnji nad grafičkim elementom udomljenika pridružena je informacijama o nazivu grafičkog elementa i pripadajućeg roditeljskog udomljenika. Na osnovi prikupljenih informacija upućuje se poziv udaljenoj proceduri složenog udomljenika `oblikujInstrukciju`. Postupak ulazi u sljedeće ponavljanje udaljenim pozivom funkcije `oblikujSkupNaredbi`. Postupak definiranja radnje nad grafičkim elementom udomljenika prikazan je slikom 5.3.

5.4. Oblikovanje programske instrukcije

Treći dio programske izvedbe korisničkog sučelja namijenjen je izvođenju na složenom udomljeniku, gdje je proširenom funkcionalnosti potrebno ostvariti oblikovanje programske instrukcije. S izuzetkom funkcionalnosti oblikovanja programske instrukcije, ostatak proširene funkcionalnosti složenog udomljenika istovjetan je proširenoj

funkcionalnosti elementarnih udomljenika stoga nije opisan u ovom poglavlju. Na slici 5.4. prikazana je programska izvedba oblikovanja programske instrukcije.



Slika 5.4. Programska izvedba oblikovanja programske instrukcije

Izvođenje započinje udaljenim pozivom lokalne funkcije `oblikujInstrukciju` uz argumente čije su vrijednosti određene informacijama o odabranoj naredbi, nazivu grafičkog elementa i pripadajućeg roditeljskog udomljenika. Rezultat poziva lokalne funkcije je programska instrukcija u obliku znakovnog niza nastala spajanjem spomenutih informacija. Pozivom funkcije `dodajUPopis` s programskom instrukcijom kao argumentom funkcije izvodi se spremanje programske instrukcije u popis programskih instrukcija.

6. Zaključak

Korisnici nerijetko imaju potrebu poosobljivanja svoje radne okoline dodavanjem željene funkcionalnosti ostvarene vlastitim programima. Poteškoće pri pisanju vlastitih programa, a time i poosobljivanju radne okoline korisnicima stvara nedostatak potrebnih znanja o postojećim programskim jezicima i paradigmama. S ciljem pojednostavljenja izrade poosobljenih radnih okolina napravljen je korak u smjeru definiranja nove programske paradigme zasnovane na udomljenicima. Programska paradigma zasnovana na udomljenicima korisnicima omogućava ne samo jednostavno poosobljivanje web stranice proizvoljnim skupom udomljenika, već i izgradnju novog primjenskog programa. U toj paradigmi udomljenici su osnovni elementi za izgradnju programa, a jezik za povezivanje udomljenika je opisivanje radnji na elementima grafičkog korisničkog sučelja udomljenika.

U ovom radu praktično je ostvareno grafičko korisničko sučelje u obliku plivajućeg kontekstno-osjetljivog izbornika za potrošaču prilagođenu izgradnju primjenskih programa povezivanjem skupa udomljenika. Primjenom plivajućeg izbornika korisnik obilazi skup udomljenika i definira slijed radnji nad elementima pripadajućih grafičkih korisničkih sučelja. Slijed radnji definiran primjenom plivajućeg izbornika sprema se u obliku računalnog programa. Za potrebe generiranja programa na osnovi rada korisnika putem plivajućeg izbornika, programski je ostvaren generator programa. Na osnovi informacija o odabranom elementu korisničkog sučelja udomljenika i odabranoj stavci kontekstno-osjetljivog izbornika, generator programa oblikuje slovčanu naredbu kojom se opisuje definirana radnja nad elementom grafičkog korisničkog sučelja udomljenika. Generirana naredba sprema se za kasnije izvođenje. Kontekstno-osjetljivi izbornik i generator programa ostvareni su u obliku korisničkih skripti u jeziku *JavaScript* koje se ugrađuju u alat *GreaseMonkey*. Primjenom alata *GreaseMonkey* postignuto je proširenje funkcionalnosti udomljenika i udomiteljske stranice na strani korisnika, bez potrebe za izmjenom programskog kôda na strani poslužitelja.

Primjenom kontekstno-osjetljivog izbornika, korisniku je omogućen jednostavan i razumljiv način izgradnje računalnih programa putem grafičkog korisničkog sučelja. Jednostavnosti načina izgradnje računalnih programa doprinosi precizno definiranje skupa dostupnih naredbi za svaku vrstu elemenata grafičkog korisničkog sučelja udomljenika. Primjerice, neke od definiranih naredbi tipke su `When clicked...` za pokretanje

izvođenja, `Click` za odabir elementa i `Double click` za dvostruki odabir elementa, a neke od definiranih naredbi znakovnog polja su `Type in...` za upis teksta te `Copy` i `Paste` za upotrebu znakovnog polja kao izvora ili odredišta teksta. Moguće proširenje izbornika ostvarivo je ugradnjom stavki za definiranje naprednih oblika međudjelovanja, kao što je pomak miša ili pomicanje objekata web stranice (engl. *drag and drop*).

7. Literatura

[1] D. Goodman, B. Eich: **"JavaScript Bible "**, 6th Edition, John Wiley & Sons, 2007

[2] M. Pilgrim: "**Greasemonkey Hacks**", O'Reilly, 2005

[3] M. Pilgrim: "**Avoid Common Pitfalls in Greasemonkey**", O'Reilly, 2005, http://www.oreillynet.com/pub/a/network/2005/11/01/avoid-common-greasemonkey-pitfalls.html?page=1

[4] D. Škvorc, "Programiranje prilagođeno potrošaču", doktorska disertacija u pripremi, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, Zagreb, 2009.

[5] J. Resig: "**The Browser Scripting Revolution**", 2007, http://ejohn.org/blog/the-browser-scripting-revolution/

[6] D. Nations: "**What's the Difference Between a Widget and a Gadget**?", http://webtrends.about.com/od/widgets/a/widgetgadget.htm, 1.9.2009.

[7] D. Redmond-Pyle, A. Moore: "**Graphical User Interface Design and Evaluation Guide**", 1st edition, Prentice Hall, 1995

[8] H. Lieberman, F. Paternò, V. Wulf: "**End User Development**", Springer, 2006

[9] Google, "**gadgets.* API Developer's Guide**", http://code.google.com/apis/gadgets/docs/dev_guide.html, 1.9.2009.

[10] Google, "**Frequently Asked Questions**", http://code.google.com/apis/gadgets/faq.html, 1.9.2009.

[11] Google, "**Gadgets API Reference**", http://code.google.com/apis/gadgets/docs/legacy/reference.html, 1.9.2009.

[12] Google, "**Gadget-to-gadget Communication**", http://code.google.com/apis/gadgets/docs/pubsub.html, 1.9.2009.

[13] J. Chaterjee, "**Working with IFRAME in JavaScript**", http://www.devarticles.com/c/a/JavaScript/Working-with-IFRAME-in-JavaScript/, 1.9.2009.

[14] P. Koch, "**Objects as associative arrays**", http://www.quirksmode.org/js/associative.html, 1.9.2009.

[15] P. Koch, "**Event order**", http://www.quirksmode.org/js/events_order.html, 1.9.2009.

[16] P. Koch, "**Event properties**", http://www.quirksmode.org/js/events_properties.html, 1.9.2009.

[17] R. Shannon, "**CSS Layout**", http://www.yourhtmlsource.com/stylesheets/csslayout.html, 1.9.2009.

[18] R. Shannon, "**CSS and Borders**", http://www.yourhtmlsource.com/stylesheets/cssborders.html, 1.9.2009.

[19] "**JavaScript Function Reference**", http://www.w3schools.com/jsref/jsref_obj_global.asp, 1.9.2009.