SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 241

Sustav za izravnu i posredovanu komunikaciju s procesima u Web pregledniku putem HTTP protokola

Tomislav Čoh

Zagreb, rujan 2011.

Zahvaljujem se mentoru prof.dr.sc Siniši Srbljiću i svim njegovim asistentima, a posebno dipl. ing. Ivanu Žužaku na uloženom trudu, strpljivosti i velikoj pomoći kod izrade ovog i prethodnih radova na preddiplomskom i diplomskom studiju. Zahvaljujem se obitelji i prijateljima na potpori i pomoći tijekom studija.

SADRŽAJ

1. Uvod

World Wide Web, ili skraćeno Web, trenutno je najkorištenija usluga Interneta. Web se koristi za razmjenu podataka preko globalne mreže Internet a zasniva se na asimetričnoj korisnik-poslužitelj (eng. *client-server*) arhitekturi. Poslužitelji javno nude svoja sredstva na Internetu, a korisnici mogu samo dohvaćati ta sredstva. Najpopularniji Web korisnici su Web preglednici. Krajnji korisnici Interneta mogu pomoću Web preglednika dohvaćati, pregledavati i pretraživati javno dostupna sredstva na Webu.

Web poslužitelji su računalne aplikacije koje omogućuju posluživanje sadržaja s računala na Webu. Te se aplikacije mogu izvršavati na bilo kojem računalu, ali mogućnost pristupa na Internetu bit će uvjetovana mrežnim okruženjem poslužiteljskog računala. S druge strane, Web korisnicima je dovoljna samo veza s Internetom da bi u potpunosti mogli zauzeti svoju ulogu u arhitekturi Weba.

Web se u početku svog razvoja koristio za posluživanje i dohvaćanje jednostavnih statičkih sredstava. U tom su modelu krajnji korisnici bili pasivni, zbog čega asimetrija uloga korisnika i poslužitelja nije predstavljala problem. Moderne Web aplikacije zahtijevaju sve veću interakciju s korisnicima, što je teško postići zbog podređenog položaja korisnika u arhitekturi Weba. Iz tog su se razloga razvile tehnologije kao što su COMET [25] i WebSocket [17] koje omogućuju asinkronu komunikaciju poslužitelja i korisnika, te tako ublažavaju asimetriju u arhitekturi Weba.

Spomenute tehnologije ne rješavaju u potpunosti nedostatke asimetrične arhitekture Weba, već ih samo ublažavaju. Logično rješenje problema nastalih asimetričnom arhitekturom Weba jest uklanjanje asimetrije u arhitekturi. Asimetrija u arhitekturi Weba može se ukloniti izjednačavanjem uloga sudionika, što bi značilo da bi svaki korisnik ujedno imao i ulogu poslužitelja. Svaki bi korisnik omogućavao pristup nekim svojim procesima što bi vrlo lako moglo naći primjenu u velikom broju modernih Web aplikacija.

U diplomskom radu istražene su metode pristupa procesima u Web pregledniku, posebno one koje se pritom koriste HTTP protokolom. Osmišljen je i programski ostvaren sustav za pristup procesima u Web pregledniku zasnovan na HTTP protokolu.

Taj sustav Web pregledniku daje ulogu poslužitelja i tako rješava problem asimetrije sudionika u arhitekturi Weba. Proširenje uloge preglednika omogućava razvoj Web aplikacija s većom interakcijom sudionika i razmjenu poruka među Web korisnicima bez posredovanja poslužitelja.

U 2. poglavlju detaljno je opisana arhitektura Weba, HTTP protokol i specifičnosti Web preglednika. 3. poglavlje idejno opisuje pristup procesima u Web pregledniku, osnovna načela te različite metode pristupa. U 4. poglavlju dan je detaljan opis ostvarenog sustava za pristup procesima u Web pregledniku. Opis sustava za mjerenje i rezultati mjerenja dani su u 5. poglavlju, a zatim slijedi jednostavan primjer korištenja sustava u 6. poglavlju, zaključak u 7. poglavlju te popis literature.

2. Arhitektura mreže World Wide Web

Web je sustav međusobno povezanih dokumenata u kojem su građevni elementi, odnosno resursi, jedinstveno određeni globalnim identifikatorima koji se nazivaju *Uniform Resource Identifier* [27]. Web je zasnovan na asimetričnoj korisnik-poslužitelj arhitekturi. Izmjena informacija između korisnika i poslužitelja odvija se razmjenom zahtjeva i odgovora. Najpopularniji protokol za prijenos podataka na Webu je HTTP [13].

U poglavlju 2.1 objašnjeno je adresiranje na Webu. Detaljan opis HTTP protokola nalazi se u poglavlju 2.2. Asimetrija sudionika na Webu objašnjena je u poglavlju 2.3. Smjer razvoja Weba opisan je u poglavlju 2.4, a poglavlje 2.5 objašnjava WebHook koncept i ideju programabilnog Weba. Specifičnosti Web preglednika u odnosu na ostale Web korisnike opisane su u poglavlju 2.6.

2.1. Adresiranje na Webu

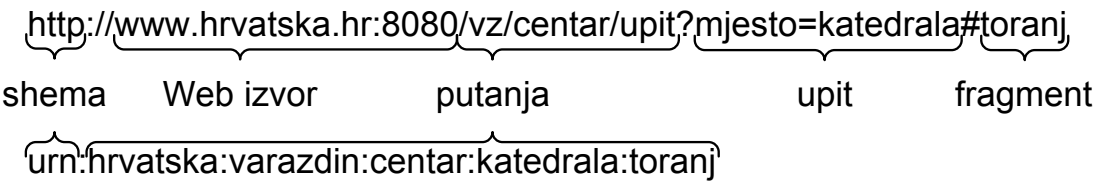
Web se sastoji od javno dostupnih resursa. Da bi resurs bio javno dostupan, mora se moći adresirati, odnosno mora imati svoju jedinstvenu lokaciju. Na Webu taj je problem rješen uvođenjem URL adresa. Pojmovi URI, URL, URN i njihov odnos objašnjeni su u poglavlju 2.1.1.

Svi resursi Weba se nalaze na računalima spojenim na Internet. Da bi se moglo adresirati te resurse, mora se prvo moći adresirati računala na kojim se oni nalaze. Sva računala spojena na Internet imaju IP adresu pomoću koje im se može pristupiti. Ipak, ljudima je nepraktično pamtiti IP adrese. Iz tog se razloga razvio sustav domena [20] koji omogućuje dodjeljivanje imena (domene) određenoj IP adresi, odnosno računalu. Detaljniji opis sustava domena nalazi se u poglavlju 2.1.2.

2.1.1. URI identifikatori

Sva sredstva na Webu imaju svoj jedinstveni identifikator. U početku razvoja Weba, vodila se diskusija o Web identifikatorima (URI). Smatralo se da će se identifikatori dijeliti u nekoliko klasa ovisno o njihovoj namjeni. Na primjer, identifikator bi mogao određivati lokaciju nekog resursa, ili njegovo ime. Tako bi se URI dijelio na klase URL (*Uniform Resource Locator* [28]) i URN (*Uniform Resource Name* [19]). Predlagao se generalizirani koncept koji bi omogućavao kasnije definiranje dodatnih klasa. Jedina takva predložena klasa bila je URC (*Uniform Resource Characteristic* [24]) koja je zamišljena da pokazuje na metapodatke umjesto na stvarni resurs.

Kasnije je ideja o dodatnoj razini u hijerarhiji počela gubiti na značaju. Moderni pogled na Web identifikatore ne razlikuje klase identifikatora, već samo URI sheme koje mogu definirati mehanizam pristupa u slučaju URL-a ili prostor imena (eng. *namespace*) u slučaju URN-a. U modernom pogledu URL i URN nisu podskupovi URI prostora, već su samo nazivi koji se koriste za precizniji opis identifikatora. Odnos URI-a, URL-a i URN-a detaljnije je opisan u izvješću tijela URI Planning Interest Group [29].



Slika 2.1: Sintaksna podjela URI identifikatora

Slika 2.1 prikazuje sintaksne dijelove jednog URL i jednog URN identifikatora. Vidljivo je da URL i URN ne sadrže jednake dijelove. URL adresa sastoji se od sheme, Web izvora, putanje, upita i fragmenta, dok URN sadrži samo shemu i putanju.

Shema

Prethodno je spomenuto da se URI u modernom pogledu identifikatora ne dijeli na klase, već se dijeli po shemama. Shema URI identifikatora određuje sintaksu i semantiku ostatka identifikatora. Registar URI shema održava IANA [6]. Trenutno je registrirano preko 60 trajnih shema. Registar za svaku registriranu shemu sadrži i RFC dokument sa specifikacijom pojedine sheme. Specifikacija sheme opisuje njezinu sintaksu.

Web izvor

Mnoge URI sheme sadržavaju hijerarhijski element koji definira Web izvor tako da se upravljanje prostorom imena, koje definira ostatak identifikatora, prosljeđuje tom Web izvoru. Sintaksa elementa Web izvora omogućuje razlikovanje Web izvora na temelju IP adrese ili registrirane domene. Dodatno je moguće navesti vrata Web izvora i podatke o korisničkom računu na Web izvoru.

Putanja

Putanja, zajedno s upitom, određuje resurs u domeni Web izvora ako je on naveden u identifikatoru. Putanja je najčešće hijerarhijski organizirana. Kod URL identifikatora putanja je podjeljena na više jedinica razdvojnim znakom '/'. Jedinice mogu označavati fizičke direktorije na poslužitelju ili simboličku lokaciju resursa.

Upit

Nehijerarhijski organizirane podatke za određivanje resursa može se upisati u upit URI identifikatora. Upit u URI identifikatoru započinje nakon znaka '?'. Često se koristi za prosljeđivanje podataka poslužiteljskoj skripti.

Fragment

Fragment URI identifikatora omogućava indirektno određivanje sekundarnog resursa referenciranjem primarnog resursa i prosljeđivanjem dodatnih podataka u fragmentu. Sekundarni resurs može biti podskup primarnog resursa ili poseban način prikaza primarnog resursa. Fragment dio URI identifikatora započinje nakon znaka '#'.

2.1.2. DNS sustav domena

Sustav domena (eng. *Domain Name System*) je raspodjeljeni hijerarhijski sustav imena računala, servisa ili bilo kakvih resursa na Internetu. Primarna mu je zadaća prevođenje imena pamtljivih ljudima u numeričke identifikatore (IP adrese) koji izravno omogućuju adresiranje na Internetu.

Sustav pruža konzistentan prostor imena koji korisnicima nudi uslugu identifikacije resursa neovisno o njegovoj stvarnoj lokaciji. Na primjer, ako želimo pristupiti resursu koji se nalazi na Web izvoru s IP adresom A, bez sustava domena bismo trebali pamtiti adresu A. U slučaju da se Web izvor premjesti na neko drugo poslužiteljsko računalo s IP adresom B, korisnika bi se trebalo obavijestiti o toj promjeni, te mu javiti novu

adresu *B*. Sustav domena omogućuje transparentan pristup resursu neovisno o njegovoj lokaciji. U sustavu se dodaje zapis o lokaciji Web izvora i daje mu se ime, odnosno domena. Kad se lokacija Web odredišta promjeni, potrebno je samo obnoviti zapis u sustavu. Korisnik će neovisno o fizičkoj lokaciji resursa uvijek moći pristupiti resursu koristeći ime Web izvora.

Zbog veličine Weba i frekvencije promjena, sustav je raspodjeljen, a koristi lokalno pretpohranjivanje (eng. *caching*) zbog veće učinkovitosti. Domene su hijerarhijski podjeljene u zone. DNS zona može se sastojati od samo jedne domene ili mnogo domena i poddomena. Upravljanje nad prostorom imena poddomene može se delegirati namjenskim DNS poslužiteljima.

Domena se sastoji od jedne ili više labela odvojenih točkom. Te labele hijerarhijski dijele prostor imena domene. Labela s krajnje desne strane određuje vršnu domenu, a svaka lijeva labela dodatno određuje podprostor vršne domene. Na primjer, domena *www.primjer.hr* sastoji se od labela *www*, *primjer* i *hr*. Labela *hr* određuje vršnu domenu, labela *primjer* određuje domenu unutar vršne domene *hr*, a *www* poddomenu domene *primjer.hr*. Domena može imati do 127 razina.

Sustav domena je raspodjeljena baza imena koja koristi korisnik-poslužitelj model a čiji su čvorovi DNS poslužitelji. Svaka domena ima barem jedan jedan DNS poslužitelj koji izdaje informacije o toj domeni.

2.2. HTTP protokol

Hypertext Transfer Protocol (HTTP) [13] je protokol aplikacijskog sloja za raspodjeljene hipermedijske sustave. HTTP protokol zasnovan je na razmjeni zahtjeva i odgovora. Protokol je tekstualan, a osim za prijenos teksta, može se koristiti i za prijenos binarnih podataka.

2.2.1. Format HTTP poruka

Paketi HTTP protokola sastoje se od zaglavlja i tijela. Zaglavlja sadrže informacije o paketu, a tijelo podatke koji se prenose. Kao što je već spomenuto, paketi mogu biti zahtjevi ili odgovori. Klijenti šalju zahtjeve, a poslužitelji vraćaju odgovore. Prvo zaglavlje HTTP zahtjeva sadrži metodu zahtjeva, URI identifikator i verziju HTTP protokola. Metoda zahtjeva opisuje operaciju nad resursom čiji je URI identifikator zadan. Prvo zaglavlje HTTP odgovora je statusna linija koja sadrži informaciju o verziji protokola i o ishodu operacije opisane HTTP zahtjevom.

Nakon obaveznog prvog zaglavlja HTTP paketa slijedi proizvoljan broj dodatnih HTTP zaglavlja koja ovisno o tome da li se radi o zahtjevu ili odgovoru mogu sadržavati informacije o klijentu ili poslužitelju, konekciji, i slično. Nakon zadnjeg zaglavlja slijedi prazan red koji odvaja zaglavlja od tijela paketa.

2.2.2. HTTP metode

Trenutnom verzijom protokola HTTP/1.1 definire su metode zahtjeva: OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE i CONNECT. Od njih su najčešće korištene metode GET i POST. Metoda GET se koristi za dohvat sadržaja nekog sredstva s poslužitelja a također se može koristiti i za slanje manje količine podataka preko URI-a. Metoda POST koristi se za slanje veće količine podataka na obradu poslužitelju. Za razliku od metode GET, POST šalje podatke u tijelu zahtjeva. Metoda POST omogućuje slanje veće količine podataka, čak i datoteka.

Od ostalih metoda, OPTIONS služi za dohvat informacija o poslužitelju. Slanjem ovog zahtjeva, poslužitelj koji implementira ovu metodu vraća odgovor bez tijela. Odgovor sadrži samo zaglavlja s informacijama o poslužitelju. Korisnik u zaglavljima zahtjeva može specificirati koje ga opcije poslužitelja zanimaju. Metoda HEAD omogućava dohvat samo zaglavlja nekog resursa, bez tijela odgovora. Za stvaranje novih resursa na poslužitelju može se koristiti metoda PUT, a za brisanje resursa metoda DELETE. Metoda TRACE se koristi za praćenje puta HTTP zahtjeva na aplikacijskom sloju. Ukoliko se za dohvat resursa koristi neki HTTP posrednik (eng. proxy), metoda TRACE omogućava dohvat informacija o tim posrednicima. Specifikacija rezervira metodu CONNECT za korištenje kod posrednika koji dinamički mogu promjeniti način rada, odnosno mogu postati tuneli [22] između korisnika i poslužitelja. Iz sigurnosnih razloga, mogućnost korištenja ove metode često je ograničena na nekolicinu ovlaštenih korisnika.

2.3. Asimetrija arhitekture Weba

Sudionici na Webu dijele se na poslužitelje i korisnike. Uloga poslužitelja je posluživanje sredstava, a uloga korisnika dohvaćanje tih sredstava. U početku razvoja Weba, resursi su najčešće biti statičke Web stranice koje su korisnici dohvaćali i prikazivali. U slučaju da Web stranica sadrži dinamički sadržaj, korisnik bi morao ponovo dohvatiti stranicu s dinamički promjenjenim sadržajem. Moderne Web aplikacije postaju sve interaktivnije. Asimetrija korisnika i poslužitelja otežava razvoj takvih aplikacija.

Skup metoda za razvoj bogatih Web aplikacija Ajax (*Asynchronous JavaScript + XML*) [15] razvio se da bi se ublažili problemi nastali zbog asimetrične arhitekture Weba. Ajax metode omogućavaju razvoj interaktivnih Web aplikacija koje asinkrono osvježavaju sadržaj s poslužitelja. Na primjer, moguće je razviti Ajax aplikaciju koja očekuje neki korisnički unos koji će promijeniti sadržaj Web stranice bez potrebe za njenim osvježavanjem.

Iz primjera Ajax aplikacije može se primjetiti da je korisnik uvijek taj koji inicira osvježavanje resursa. Pomogu Ajaxa nemoguće je ostvariti poslužiteljem iniciranu asinkronu dojavu događaja. Razlog tome jest nemogućnost pristupa korisniku na Webu, odnosno nemogućnost adresiranja korisnika. Zaobilaženje ovog problema omogućavaju tehnologije COMET [25] i WebSocket [17] koje su opisane u poglavljima 3.4.1 i 3.4.2.

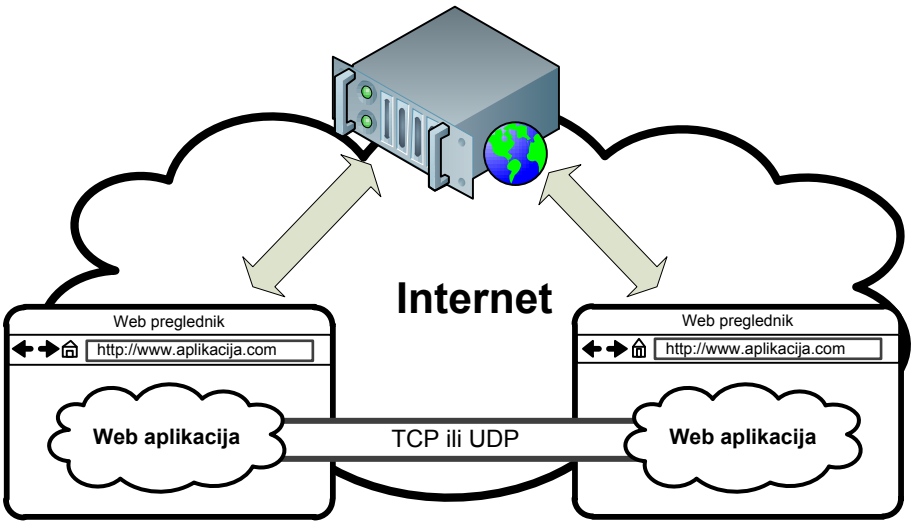
Sva predložena rješenja samo ublažavaju probleme nastale asimetričnom arhitekturom Weba. Još uvijek je nemoguće ostvariti izravan pristup korisniku na Webu što bitno ograničava mogućnosti razvoja modernih Web aplikacija.

2.4. Razvoj Weba

Arhitektura Weba jako se malo mijenjala od devedesetih godina prošlog stoljeća kad se Web počeo ubrzano razvijati. Glavni razlog tome je njegova brzina rasta, zbog kojeg je trom na promjene. Aktualna specifikacija HTTP protokola (HTTP/1.1) datira iz 1999. godine. S druge strane, velike promjene na Webu doživio je sadržaj. Taj trend razvoja neki nazivaju Web 2.0. Korisnike se potiče da sudjeluju u kreiranju sadržaja, i tako postaju aktivniji sudionici Weba. Istovremeno ubrzano se razvijaju i Web preglednici koji omogućavaju prikazivanje multimedijalnih sadržaja pomoću posebnih dodataka, ubrzava se izvođenje Web aplikacija, a aplikacije postaju računalni sve zahtjevnije. Sljedeći korak u razvoju Web preglednika donosi HTML5 standard. Osim standardizacije ponude multimedijalnih sadržaja, standard donosi i brojne promjene koje omogućuju daljnu evoluciju Web aplikacija.

U okviru HTML5 [16] standarda predlažu se programska sučelja za moderne Web preglednike kojim je cilj ublažavanje asimetrije u arhitekturi Weba. Programsko sučelje *ConnectionPeer* [4] omogućava razmjenu poruka među Web preglednicima bez posredovanja poslužitelja. Ovo sučelje bi se u bližoj budućnosti trebalo moći koristiti za razmjenu tekstualnih poruka, slika i datoteka među Web korisnicima, a u kombinaciji sa *Stream* [7] programskim sučeljem može se koristiti za uspostavljanje tokova podataka među Web korisnicima. U kombinaciji s device elementom [5], ova sučelja

se mogu iskoristiti za uspostavljanje video razgovora između Web korisnika bez posredovanja poslužitelja. Navedena sučelja već su eksperimentalno ostvarena [9].



Slika 2.2: Otvaranje veze pomoću *ConnectionPeer* programskog sučelja

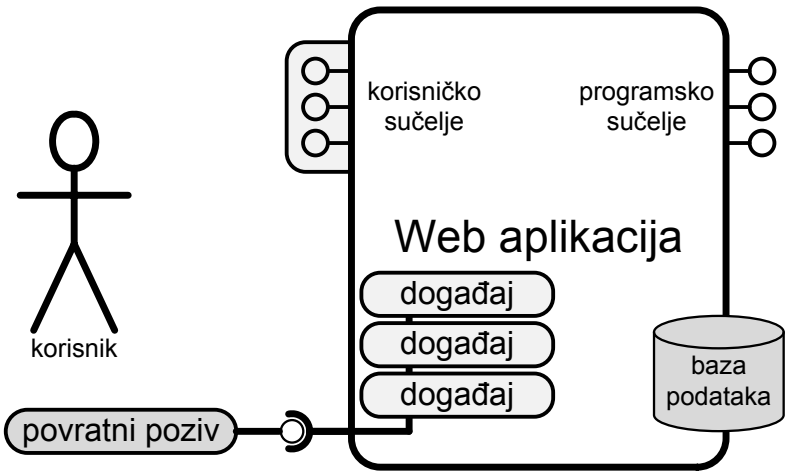
Slika 2.2 prikazuje proces uspostavljanja veze između dva Web preglednika pomoću ConnectionPeer programskog sučelja. Veza se uspostavlja pomoću poslužitelja koji služi kao posrednik za razmjenu podataka o preglednicima. Svaki korisnik otvara spojnicu (eng. *socket*), i podatke o otvorenoj spojnici šalje poslužitelju. Korisnici preko poslužitelja izmjenjuju podatke o spojnicama i otvaraju vezu preko koje razmjenjuju podatke. Za prijenos podataka mogu se koristiti TCP ili UDP protokol ovisno o tome želi li se osigurati pouzdana komunikacija.

Pristup procesima kakav omogućuje *ConnectionPeer* programsko sučelje nije karakterističan za Web. Resursi se nemogu izravno adresirati zbog čega je komunikacija ovisna o posredovanju poslužitelja.

2.5. WebHook i programabilni Web

WebHook [21] je jednostavan mehanizam objave i pretplate (eng. *publish/subscribe*) za dojavu događaja. Objavljivač (eng. *publisher*) određuje događaje na koje se pretplatnici (eng. *subscriber*) mogu prijaviti.

Slika 2.3 prikazuje strukturu Web aplikacija koje podržavaju WebHook. Kao i svaka standardna Web aplikacija, i aplikacija sa slike sadrži korisničko i programsko sučelje, te bazu podataka. Dodatno, aplikacija sa slike sadrži i sustav za dojavu događaja. Aplikacija korisnicima nudi korisničko sučelje preko kojeg, između ostalog,



Slika 2.3: Struktura WebHook aplikacije

mogu prijavljivati povratne pozive (eng. *callback*) na određene događaje. Povratni poziv može biti bilo koji globalno dostupan resurs na Webu, a korisnik ga prijavljuje koristeći njegovu URL adresu.

WebHook je zamišljen da na Webu rješava sljedeće probleme:

- obavješćivanje
- sinkronizacija
- ulančavanje
- mijenjanje
- proširivanje

Osnovna funkcija WebHooka je obavješćivanje o događajima. Problem sinkronizacije nekih dviju aplikacija, pomoću WebHooka se rješava izmjenom obavijesti o promjenama sadržaja tih aplikacija koje ažuriraju sadržaj sinkroniziranih aplikacija. Problem ulančavanja se može riješiti prijavom WebHook povratnog poziva koji prilikom neke korisničke akcije pokreće određenu akciju na udaljenoj Web aplikaciji.

Problemi mijenjanja i proširivanja Web aplikacija uvod su u programabilni Web. Evolucija Web aplikacija dovela je do razvoja Web programskih sučelja vrlo široke primjene. Ta općenita programska sučelja potakla su razvoj takozvanih *mashup* Web aplikacija. *Mashup* je Web aplikacija koja koristi kombinirani sadržaj, prezentaciju ili funkciju s dva ili više izvora za pružanje novih usluga. Ipak, zbog velikog broja nestandardiziranih programskih sučelja, još nije došlo do potpune integracije aplikacija što je preduvjet za razvoj programibilnog Weba.

Ideja programabilnog Weba je razvoj skupa Web aplikacija između kojih bi korisnik, slično kao u Unix ljusci, mogao upravljati protokom podataka. Poticao bi se razvoj manjih fokusiranih Web aplikacija koje bi u kombinaciji s većim aplikacijama s podrškom za WebHook sačinjavale nove bogate i fleksibilne Web aplikacije. Korisnicima bi se omogućilo razvijanje dodataka za postojeće Web aplikacije i jednostavno povezivanje s već postojećim dodacima. Programabilni Web omogućio bi personalizaciju korisničkog Web iskustva.

2.6. Specifičnosti Web preglednika

Web korisnik je računalna aplikacija koja može dohvaćati resurse s Web poslužitelja. Najkorišteniji Web korisnici su Web preglednici. Web preglednici po mnogočemu specifični Web korisnici. Predviđeni su za prikazivanje dokumenata i Web aplikacija koje su ostvarene standardiziranim Web tehnologijama. Glavna značajka Web aplikacija je prenosivost. Pravilno ostvarena Web aplikacija trebala bi se jednako ili vrlo slično izvršavati na svim programskim ostvarenjima Web preglednika. Sva ostvarenja Web preglednika imaju jednake probleme koje više ili manje uspješno rješavaju. Neki od problema su pružanje jednakog korisničkog iskustva svim korisnicima, sigurnost korisnika i privatnost korisničkih podataka.

Korisničko iskustvo nužno ovisi o mnogim parametrima kao što su brzina Internet veze, brzina računala, operacijski sustav, mrežno okruženje i slično. Nužno je prilikom izrade Web aplikacija te parametre imati u vidu. Specifikacija protokola HTTP/1.1 [13] pisana je u vrijeme sporih Internet veza, i u skladu s tim specifikacija ograničava broj otvorenih veza korisnika s jednim Web izvorom na najviše dvije veze. Ta je specifikacija i danas aktualna, pa brojni preglednici i danas imaju ugrađeno to ograničenje. To ograničenje onemogućava paralelizirano učitavanje Web stranica. Iz tog je razloga nedavno tvrtka Microsoft odstupila od specifikacije i ovećala ograničenje na šest istovremenih veza s istim Web izvorom u svom pregledniku Internet Explorer 8.

Sigurnost je zbog velikog broja korisnika Web preglednika vrlo važna. Nije rijetka pojava da hakeri nađu sigurnosni propust u Web pregledniku koji iskoriste za napad na korisničko računalo. Također su vrlo česte krađe korisničkih podataka i narušavanje privatnosti korisnika. Iz tog razloga postoje sigurnosni mehanizni zajednički svim Web preglednicima kojim je cilj što je moguće više onemogućiti sigurnosne propuste u dizajnu Web preglednika.

Jedan od najvažnijih sigurnosnih koncepata u Web pregledniku je politika istog izvora (eng. *Same Origin Policy*, *SOP*) [26]. Politika istog izvora ograničava kon-

tekst Web stranice Web izvorom te stranice. Web izvor određen je sredstvom pristupa (protokol), adresom (IP ili domena) i vratima poslužitelja. Skripte koje se izvršavaju unutar konteksta Web stranice učitane s nekog Web izvora ne smiju čitati ili postavljati postavke dokumenata učitanih s drugih izvora. Također, zabranjeno je unutar konteksta Web stranice asinkrono dohvaćati resurse s drugih Web izvora osim ako drugi izvor to dopušta.

Dijeljenje resursa različitog porijekla (eng. *Cross-Origin Resource Sharing*) [30] je mehanizam koji omogućava slanje zahtjeva iz konteksta Web stranice na različite Web izvore. Ovaj je mehanizam transparentan za Web skripte. Kad se iz skripte šalje zahtjev za resursom drugog porijekla, preglednik u pozadini šalje HTTP zahtjev metode OPTIONS na poslužitelj tog resursa. U zaglavlja OPTIONS paketa upisuju se parametri originalnog zahtjeva, te porijeklo Web skripte. Poslužitelj vraća HTTP odgovor u čijim zaglavljima se nalaze postavke poslužitelja. Na temelju tih postavki preglednik odlučuje hoće li dozvoliti slanje originalnog zahtjeva.

3. Udaljeni pristup korisničkim procesima u Web pregledniku

Web aplikacije sastoje se od korisničkog i poslužiteljskog dijela. Poslužiteljski dio Web aplikacije izvršava se na poslužitelju, a korisnički u Web pregledniku. Korisnički dio aplikacije sadrži dinamičke dijelove koji mogu ali nemoraju biti sinkronizirani s poslužiteljem. Svaki takav dio aplikacije može se smatrati procesom u Web pregledniku, odnosno resursom. Pristup procesima u Web pregledniku podrazumjeva posluživanje takvih resursa na Webu.

Poglavlje 3.1 sadrži motivaciju za ostvarenje pristupa procesima u Web pregledniku. U poglavlju 3.2 objašnjena su osnovna načela pristupa procesima u Web pregledniku. Pristup može biti ostvaren izravno ili posredovano. Ideja izravnog pristupa opisana je u poglavlju 3.3, a posredovanog u poglavlju 3.4.

3.1. Motivacija

Asimetrija arhitekture Weba opterećuje poslužitelje, a ograničava korisnike. Razmjerno s povećanjem broja korisnika, mora rasti i broj poslužitelja. Izjednačavanje uloge korisnika i poslužitelja omogućila bi razvoj novih metoda u izradi Web aplikacija koje bi olakšale razvoj bogatih Web aplikacija i poboljšale njihovo svojstvo razmjernog rasta.

Izravan pristup korisniku omogućio bi zaobilaženje poslužitelja i izravnu izmjenu poruka među korisnicima. Tako bi se smanjio dojam poslužitelja kao uskog grla sustava. Ovaj problem djelomično rješava specifikacija HTML5 s programskim sučeljem *ConnectionPeer* koje je opisano u poglavlju 2.4.

Izravno adresiranje procesa u Web pregledniku omogućilo bi primjenu Web koncepata iz domene poslužitelja u domeni korisnika. Jedan od tih koncepata je WebHook koji je objašnjen u poglavlju 2.5. Integracija WebHooka u preglednik omogućila bi ugradnju korisničkih procesa u Web aplikacije smanjujući tako jaz između koriničkog

i poslužiteljskog dijela Web aplikacije. Korisnički WebHook omogućio bi korisnicima da postanu sastavni dio programabilnog Weba.

3.2. Osnovna načela

Problem pristupa procesima u Web pregledniku dijeli se na problem adresiranja i problem prezentacije korisničkih procesa na Webu. Problem adresiranja opisan je u poglavlju 3.2.1. Prezentacija procesa u Web pregledniku podrazumjeva ostvarenje Web sučelja prema procesu, odnosno posluživanj tog procesa. Budući da ovaj rad opisuje sustav za pristup procesima u Web pregledniku korištenjem HTTP protokola, posluživanje procesa u Web pregledniku ostvareno je obradom HTTP zahtjeva za resursom koji prezentira proces. Problem obrade zahtjeva u Web pregledniku obrađen je u poglavlju 3.2.2.

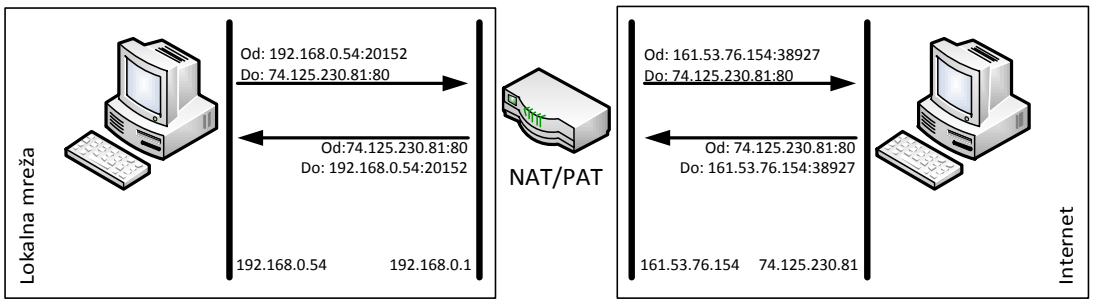
3.2.1. Adresiranje procesa u Web pregledniku

Na Webu se za adresiranje resursa koriste URL adrese. URL adresa određuje sredstvo pristupa, Web izvor resursa i njegov položaj u domeni Web izvora. Sredstvo pristupa resursu označava protokol prijenosa, a to je najčešće HTTP. Web izvor može biti određen IP adresom ili domenom. Adresiranje resursa u domeni Web izvora zadatak je samog Web izvora. Posluživanje i adresiranje u domeni Web preglednika opisano je u poglavlju 3.2.2.

Zbog velikog broja korisnika Interneta i ograničenog adresnog prostora IPv4 [1] protokola, nemoguće je pridijeliti jedinstvenu javnu IP adresu svakom korisniku. Većina korisnika na Internet se spaja preko usmjernika koji dijeli lokalnu mrežu od globalne mreže Internet. Svaki korisnik ima svoju lokalnu IP adresu, a svi korisnici u lokalnoj mreži dijele jednu ili više javnih IP adresa pridjeljenih usmjerniku. Adresiranje između lokalne i javne mreže omogućuje prevoditelj mrežnih adresa (eng. *Network Address Translator*, NAT) [18] čija je zadaća prevođenje IP adresa iz jednog adresnog prostora u drugi. Prevoditelj mrežnih adresa može biti statički ili dinamički. Statički prevoditelji pomoću statičih tablica prevode adrese iz jednog adresnog prostora u drugi u odnosu 1 na 1, a dinamički mogu prevoditi jednu adresu u jednom adresnom prostoru u više adresa u drugom koristeći preslikavanje para adresa i vrata. (eng. *Port Address Translation*, PAT). Prevoditelji mrežnih adresa kakvi se nalaze u usmjeriteljima

[1] IPv4 je verzija 4 protokola IP koja koristi 32-bitno adresiranje. U planu je uvođenje IPv6 protokola koji bi za adresiranje koristio 128 bita i tako riješio problem nedostatka IP adresa

lokalnih mreža najčešće su dinamički jer najčešće dijele jednu javnu IP adresu među svim korisnicima u lokalnoj mreži.

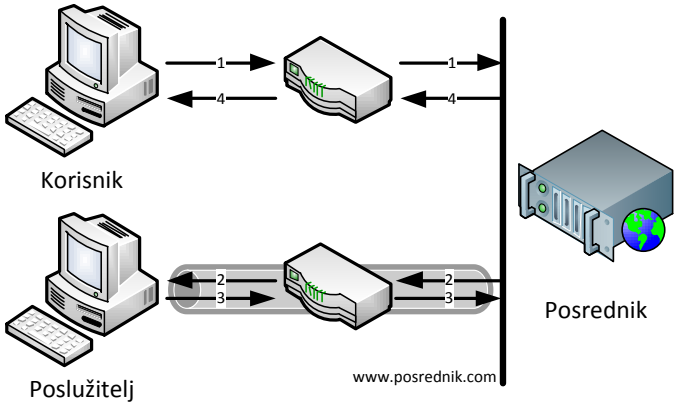


Slika 3.1: Primjer rada dinamičkog prevoditelja mrežnih adresa

Slika 3.1 prikazuje osnovni primjer rada dinamičkog prevoditelja mrežnih adresa gdje korisnik iz lokalne mreže šalje zahtjev poslužitelju na Internetu koji vraća odgovor korisniku. Korisnik prilikom slanja zahtjeva otvara vrata (20152) na svom mrežnom sučelju te u IP paket zahtjeva upisuje ta vrata i svoju lokalnu IP adresu (192.168.0.54) kao izvorište paketa. Odredište paketa određeno je javnom IP adresom poslužitelja (74.125.230.81.80) i vratima (80) koja poslužitelj osluškuje. Prevoditelj mrežnih adresa prilikom prolaska paketa kroz usmjernik mijenja izvorišnu mrežnu adresu i vrata javnom IP adresom usmjernika (161.53.76.154) i novootvorenim vratima (38927) preko kojih usmjernik prosljeđuje paket do poslužitelja. Poslužitelj odgovor vraća na javnu IP adresu usmjernika i vrata preko kojih je usmjernik proslijedio zahtjev korisnika. Prevoditelj mrežnih adresa prilikom prolaska paketa odgovora kroz usmjernik odredišnu adresu odgovora zamjenjuje IP adresom korisnika i vratima s kojih je poslan originalni zahtjev.

Dinamički prevoditelj mrežnih adresa korisnicima lokalne mreže omogućuje pristup Internetu, ali otežava adresiranje takvih korisnika na Internetu. Korisniku iza prevoditelja mrežnih adresa može se pristupiti samo dinamički. Korisnik prilikom slanja IP paketa na Internet prisiljava usmjernik na privremeno otvaranje vrata preko kojih može dočekivati IP pakete s Interneta. Moguće je i trajno otvoriti vrata na usmjerniku i ostvariti prosljeđivanje prometa s tih vrata određenom korisniku na lokalnoj mreži, ali takve postavke može mijenjati samo korisnik s administratorskim ovlastima na usmjeritelju. Postoje brojne tehnike zaobilaženja prevoditelja mrežnih adresa koje omogućavaju izravno adresiranje korisnika, ali ni jedna od njih nije primjenjiva u svakom mrežnom okruženju.

Slika 3.2 prikazuje jedno od mogućih rješenja uspostave veze između korisnika



Slika 3.2: Ostvarenje posredovane veze između korisnika koji se nalaze iza NAT-a

koji se nalaze iza prevoditelja mrežnih adresa. Problem je riješen postavljanjem javno dostupnog posredničkog poslužitelja za posredovanje komunikacije među korisnicima. Donji korisnik poslužuje svoje resurse tako da s posrednikom uspostavlja dvosmjernu vezu. Gornji korisnik u koraku 1 šalje HTTP zahtjev posredniku koji u koraku 2 preko ostvarene veze prosljeđuje zahtjev donjem korisniku. Donji korisnik obrađuje zahtjev i u koraku 3 odgovor na prosljeđeni zahtjev vraća posredniku koji u koraku 4 kao odgovor na zahtjev iz koraka 1 šalje prosljeđeni odgovor donjeg korisnika.

Za razliku od programskog sučelja *ConnectionPeer* koje je opisano u poglavlju 2.4, gdje se korisnici ne mogu izravno adresirati, ovaj pristup korisnicima daje mogućnost adresiranja svojih resursa. Specifikacija *ReverseHTTP* opisuje Web uslugu koja služi kao posrednik za pristup korisnicima na Webu. Detaljan opis specifikacije nalazi se u poglavlju 3.4.3.

Zbog lakšeg pristupa resursima, na Webu se umjesto IP adresa koriste domene. Sustav domena je objašnjen u poglavlju 2.1.2. Kod pristupa korisniku preko posrednika, posrednik može dio svojeg adresnog prostora prepustiti korisniku ili mu čak dodjeliti poddomenu. Ako se korisniku pristupa izravno, svaki korisnik mora prijaviti svoju domenu na neki DNS poslužitelj. Dodatno, budući da većina korisnika ima promjenjivu IP adresu, prilikom svake promjene korisnik je zadužen za ažuriranje DNS zapisa.

3.2.2. Posluživanje korisničkih zahtjeva u Web pregledniku

Uvjet posluživanja korisničkih zahtjeva u Web pregledniku je određivanje resursa, odnosno sučelja prema procesima u Web pregledniku. Procesi kojim želimo pristupiti

nalaze se u DOM [2] okruženju Web stranice, što znači da ćemo pristup tim procesima morati ostvariti u istom okruženju. Skriptni jezik JavaScript najrašireniji je jezik koji se može izvršavati u DOM okruženju, pa je on logičan izbor za ostvarenje sučelja prema procesima u Web pregledniku.

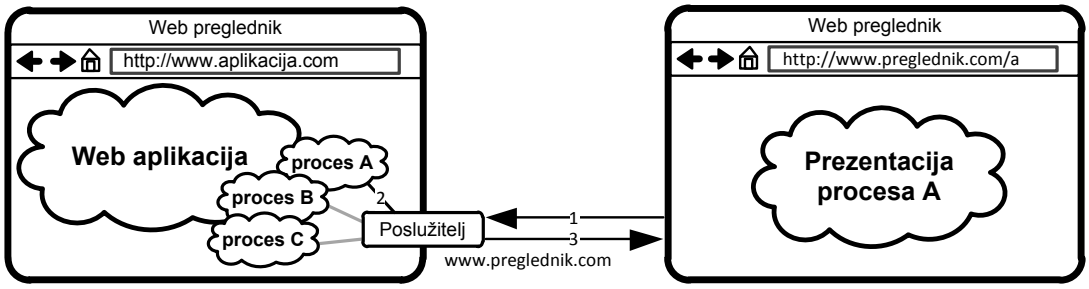
Definicija resursa ovisi o programskom sučelju poslužitelja. Standard CGI [12] određuje resurs zasebnom aplikacijom koja kao ulaz prima korisnički zahtjev, a na standardni izlaz ispisuje odgovor. Nedostatak ovog pristupa jest u tome što se prilikom svakog primljenog zahtjeva na poslužiteljskom računalu pokreće novi proces koji se gasi nakon obrade zahtjeva što uvodi kašnjenje kod obrade. Zbog ovog problema CGI se uglavnom više ne koristi kod izrade Web aplikacija, ali mnoga moderna poslužiteljska sučelja izrađena su po uzoru na CGI. Problem pokretanja novih procesa za svaki korisnički zahtjev riješen je uvođenjem veće ovisnosti o tehnologiji implementacije. CGI aplikaciju zamjenjuju moduli za obradu zahtjeva koji se mogu izvršavati u istom procesu bez potrebe za učestalim učitavanjem i brisanjem konteksta procesa. Prednosti takvog pristupa nad CGI su bolja prenosivost, lakša ugradnja u postojeće sustave i nadogradnja, ugrađena podrška za višedretvenost, ugrađena sigurnost i bolje svojstvo razmjernog rasta. Neka od takvih sučelja su WSGI [1] za programski jezik Python, Servlet tehnologija za programski jezik Java i JSGI [23] za poslužiteljski JavaScript.

Specifikacija JSGI resurse predstavlja JavaScript funkcijama koje kao parametar primaju objekt okoline, koji sadrži HTTP zahtjev i odgovor. Zadaća funkcije za obradu zahtjeva je obraditi zahtjev i upisati zaglavlja i tijelo odgovora. Zbog razlika u okruženju izvođenja poslužiteljskog JavaScripta i onog koji se izvršava u Web preglednicima, ova se specifikacije nemože izravno iskoristiti za pristup procesima u Web pregledniku. Potrebno je odrediti novo sučelje za obradu korisničkih zahtjeva koje će resurse predstavljati JavaScript funkcijama.

3.3. Izravan pristup

Izravan pristup pregledniku ostvaruje se ugrađivanjem poslužitelja u Web preglednik. Taj poslužitelj mora biti javno dostupan na Internetu, što može predstavljati problem za korisnike koji se spajaju na Internet preko usmjernika s prevoditeljem mrežnih adresa. Za takve korisnike moguće je rješenje ručno postavljanje prosljeđivanja prometa s usmjernika na korisničko računalo, ili zaobilaženje prevoditelja mrežnih adresa (eng. *NAT traversal*).

[2] DOM (*Document Object Model*) - konvencija za predstavljanje i manipulaciju objektima u HTML, XHTML i XML dokumentima

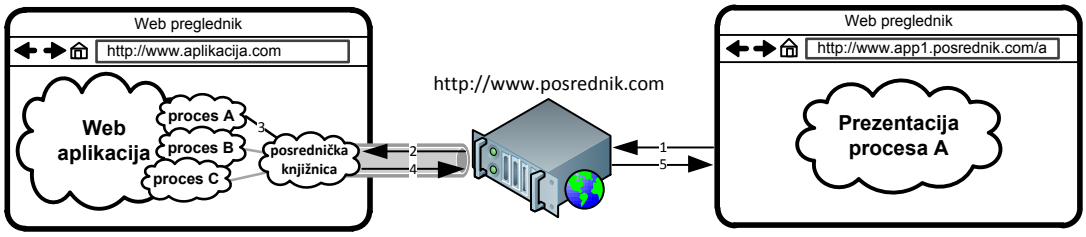


Slika 3.3: Izravan pristup procesima u Web pregledniku

Slika 3.3 prikazuje sustav za izravan pristup procesima u Web pregledniku. U lijevi Web preglednik ugrađen je poslužitelj s domenom *www.preglednik.com* koji omogućava pristup procesima u Web pregledniku. Posluživani procesi su dio Web aplikacije koju izvršava lijevi Web preglednik. Desni Web preglednik sa slike pristupa procesu A iz prvog preglednika tako da u koraku 1 šalje HTTP zahtjev ugrađenom poslužitelju u lijevom pregledniku koji na zahtjev odgovara u koraku 2. Web aplikacija koju izvršava lijevi preglednik određuje posluživane resurse tog preglednika i za svaki od njih prijavljuje funkciju za obradu zahtjeva koju prijavljuje na ugrađeni poslužitelj.

3.4. Posredovan pristup

Posredovan pristup procesima u Web pregledniku ostvaruje se uvođenjem posredničkog poslužitelja koji rješava problem adresiranja Web preglednika tako što prosljeđuje sav promet prijavljenih korisnika.



Slika 3.4: Posredovani pristup procesima u Web pregledniku

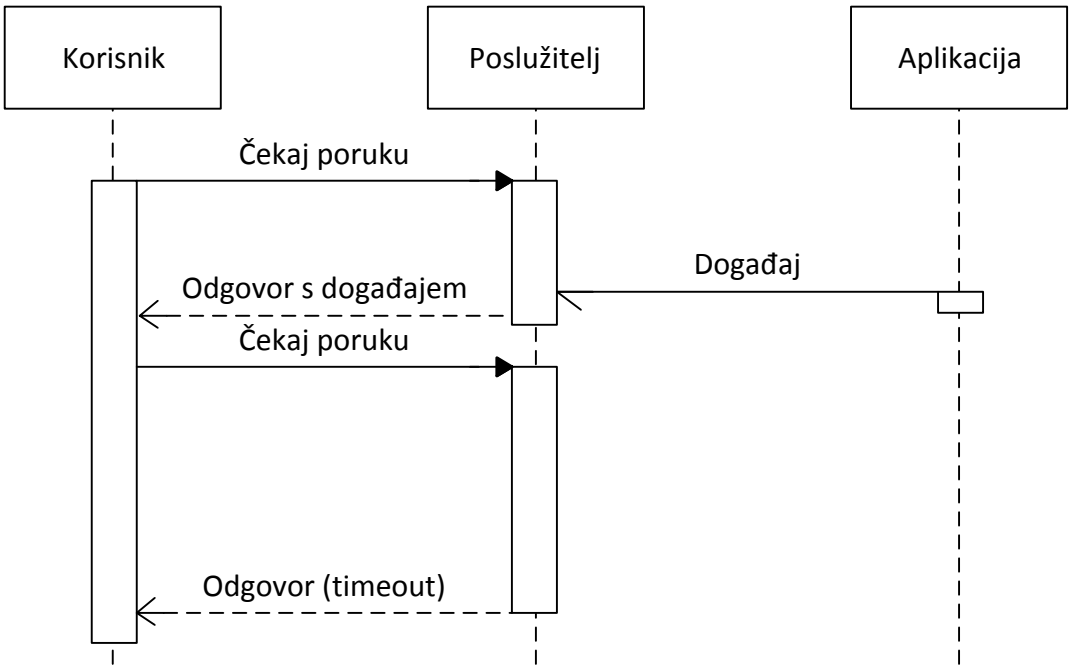
Slika 3.4 prikazuje posredovani pristup Web pregledniku. Za razliku od izravnog pristupa opisanog u prethodnom poglavlju, kod posredovanog pristupa problem adresiranja, odnosno pristupa pregledniku rješava posrednički poslužitelj. Lijevi preglednik

se prijavljuje na posrednik i dobiva na raspolaganje njegovu poddomenu *www.app1.-posrednik.com* za adresiranje svojih procesa. Web aplikacija koja se izvršava u lijevom pregledniku posluživane procese prijavljuje posredničkoj knjižnici koja je veza preglednika s posrednikom. Na slici desni preglednik u koraku 1 posredniku šalje zahtjev s adresom procesa A iz lijevog preglednika (*http://www.app1.posrednik.com/a*). Posrednik u koraku 2 prosljeđuje zahtjev do posredničke knjižnice lijevog preglednika koja u koraku 3 poziva funkciju za obradu zahtjeva koja je prijavljena za posluživanje procesa A. Rezultat obrade je prezentacija procesa A u obliku HTTP odgovora koji posrednička knjižnica u koraku 4 šalje posredniku, a posrednik u koraku 5 prosljeđuje desnom pregledniku.

Veza posredičke knjižnice s posrednikom omogućuje dvosmjernu razmjenu poruka između preglednika i posrednika. Ta se veza može ostvariti korištenjem COMET modela opisanog u poglavlju 3.4.1 ili WebSocket standarda opisanog u poglavlju 3.4.2

3.4.1. COMET model dostavljanja podataka

COMET je zajednički naziv za tehnike izrade Web aplikacija koje koriste mogućnost odgode obrade HTTP zahtjeva za ostvarenje modela dostavljanja podataka korisniku.



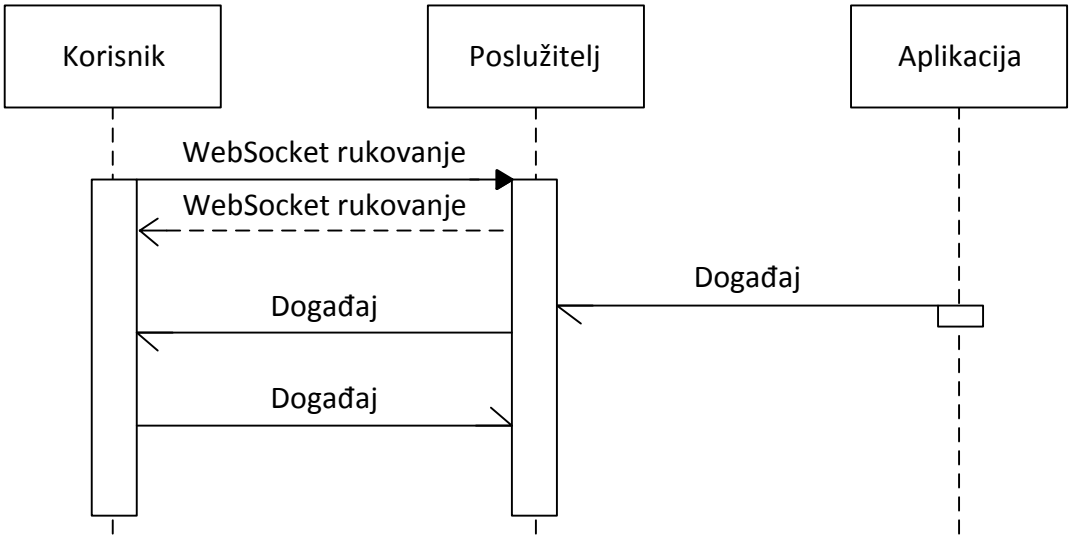
Slika 3.5: Dijagram slijeda COMET modela dostavljanja podataka

Slika 3.5 prikazuje dijagram slijeda COMET modela dostavljanja podataka. Dija-

gram prikazuje korisnika, poslužitelja i aplikaciju koju poslužitelj poslužuje. Korisnik ostvaruje vezu s poslužiteljem koji osluškuje događaje aplikacije o kojim obavještava korisnika. Problem pristupa korisniku rješava se neprekidnim slanjem zahtjeva korisnika poslužitelju koje poslužitelj obrađuje asinkrono, kad aplikacija poslužitelj obavijesti o određenom događaju. Poslužitelj tada korisniku šalje odgovor u kojem omata obavijest o događaju aplikacije. Korisnik, nakon što primi odgovor poslužitelja, šalje novi zahtjev poslužitelju koji će poslužitelj također asinkrono obraditi. U slučaju da aplikacija za vrijeme aktivnosti zahtjeva ne javi poslužitelju nikakav događaj, zahtjev se odbacuje, a korisnik zatim šalje novi zahtjev.

3.4.2. WebSocket standard

WebSocket je standard koji je razvijen u sklopu HTML5 standarda, a zamišljen je kao jednostavnija i učinkovitija alternativa COMET modelu. Standard WebSocket korisnicima omogućuje otvaranje TCP veze s poslužiteljem za dvosmjernu razmjenu poruka upravljanu događajima.



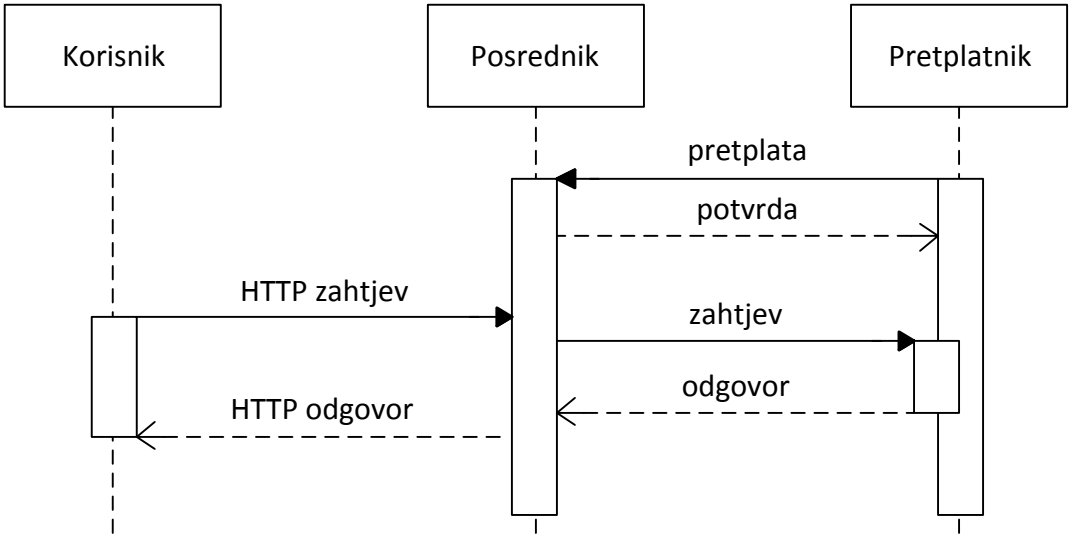
Slika 3.6: Dijagram slijeda korištenja WebSocket veze

Na slici 3.6 prikazan je dijagram slijeda uspostave i korištenja WebSocket veze. WebSocket veza uspostavlja se dvokoračnim rukovanjem. Korisnik šalje HTTP zahtjev metode GET poslužitelju s dodatnim zaglavljima za uspostavu WebSocket veze zajedno s 2 ključa. Poslužitelj iz priloženih ključeva generira kontrolnu sumu koju s ostalim zaglavljima odgovora šalje korisniku. Korisnik pomoću kontrolne sume provjerava da li je poslužitelj primio originalni zahtjev. Nakon uspješne provjere, otvorena

TCP veza se koristi za razmjenu poruka. Komunikacija između korisnika i poslužitelja je asinkrona, a primitak poruke sa strane Web preglednika, odnosno korisnika, proizvest će DOM događaj na koji se mogu prijavljivati povratne JavaScript funkcije za obradu događaja.

3.4.3. ReverseHTTP specifikacija

ReverseHTTP specifikacija [14] propisuje pravila za izradu posredničke Web usluge za omogućavanje pristupa procesima u Web preglednicima.



Slika 3.7: Dijagram toka pretplate i korištenja ReverseHTTP usluge

Slika 3.7 prikazuje dijagram toka pretplate i i korištenja ReverseHTTP Web usluge. Dijagram opisuje interakciju običnog korisnika, ReverseHTTP posredničke usluge i preglednika pretplaćenog na ReverseHTTP uslugu. Pretplatnik se prijavljuje na posredničku uslugu na koju usluga odgovara potvrdom prijave. Nakon toga korisnik upućuje HTTP zahtjev posredničkoj usluzi namjenjen pretplatniku. Posrednik primljeni zahtjev prosljeđuje pretplatniku koji ga obrađuje i posredniku vraća odgovor. Posrednik primljeni odgovor šalje korisniku kao odgovor na originalni HTTP zahtjev.

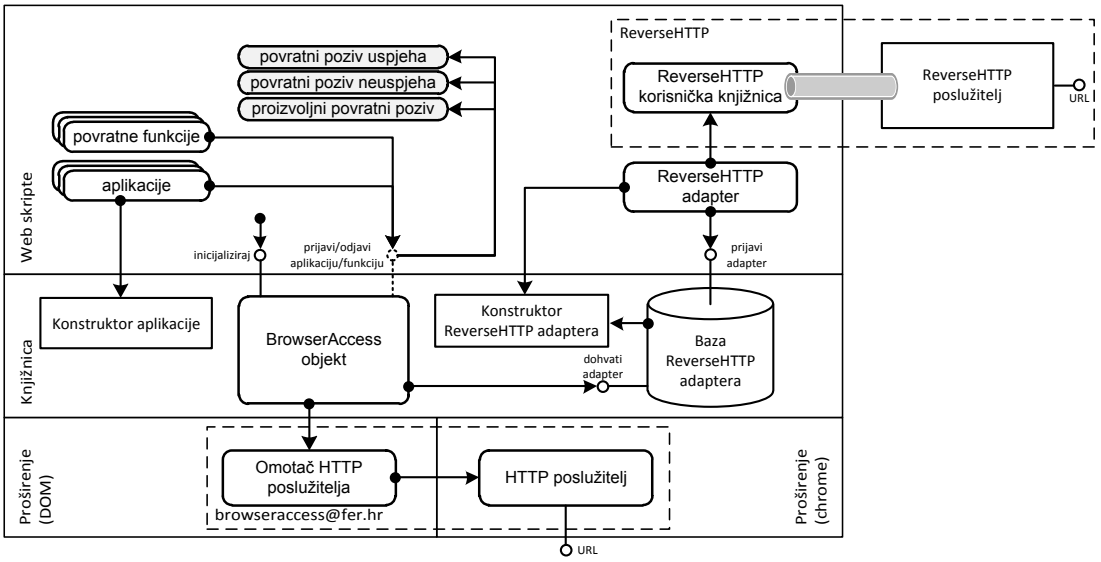
Specifikacija ReverseHTTP određuje da se komunikacija pretplatnika ostvaruje upotrebom COMET modela, odnosno dugim pozivanjem (eng. *long polling*), ali može se ostvariti i upotrebom WebSocketa. Komunikacijski protokol za razmjenu kontrolnih poruka između posrednika i pretplatnika nije određen specifikacijom, pa ovisi o implementaciji usluge.

4. Sustav za pristup procesima u Web pregledniku

U sklopu diplomskog rada programski je ostvaren sustav za pristup procesima u Web pregledniku. U poglavlju 4.1 opisana je arhitektura sustava, a poglavlje 4.2 opisuje detalje implementacije.

4.1. Arhitektura sustava

Sustav se sastoji od podsustava za izravan pristup Web pregledniku, podsustava za posredovani pristup i ujedinjenog sučelja za oba sustava.



Slika 4.1: Arhitektura sustava za pristup procesima u Web pregledniku

Slika 4.1 prikazuje arhitekturu ostvarenog sustava. Sustav je podijeljen po okruženjima u kojim se izvršavaju pojedini njegovi dijelovi. Veći dio sustava izvršava se u Web pregledniku koji je podijeljen na okruženje Web skripti, knjižnice i proširenja koje se

dodatno dijeli na proširenja u DOM i chrome [1] okruženju. Iako se Web skripte, knjižnice i DOM dio proširenja izvršavaju u DOM okruženju, mogu se međusobno logički podijeliti. DOM dio proširenja određuje dijelove sustava koje proširenje ugrađuje u okruženje Web stranice, knjižnica sadrži ujedinjeno sučelje za pristup procesima u Web pregledniku, a okvir Web skripta sadrži dijelove kojim upravlja Web aplikacija koja koristi sustav.

Okvir s oznakom *ReverseHTTP* na slici omeđuje podsustav za posredovani pristup Web pregledniku a okvir s oznakom *browseraccess@fer.hr* omeđuje proširenje za Mozilla Firefox Web preglednik koje omogućava izravan pristup pregledniku. Podsustav *ReverseHTTP* detaljno je opisan u poglavlju 4.1.2, podsustav *browseraccess@fer.hr* u poglavlju 4.1.1, a ujedinjeno sučelje u poglavlju 4.1.3.

4.1.1. Proširenje preglednika HTTP poslužiteljem

Izravan pristup Web pregledniku, u sklopu ovog rada, ostvaren je kao proširenje Web preglednika Mozilla Firefox koje u preglednik ugrađuje HTTP poslužitelj. Proširenje se sastoji od samog poslužitelja i omotača tog poslužitelja. Ugrađen je Mozillin poslužitelj za testiranje jedinica (eng. *HTTP server for unit tests*) [3] ostvaren u programskom jeziku JavaScript kao XPCOM [11] komponenta.

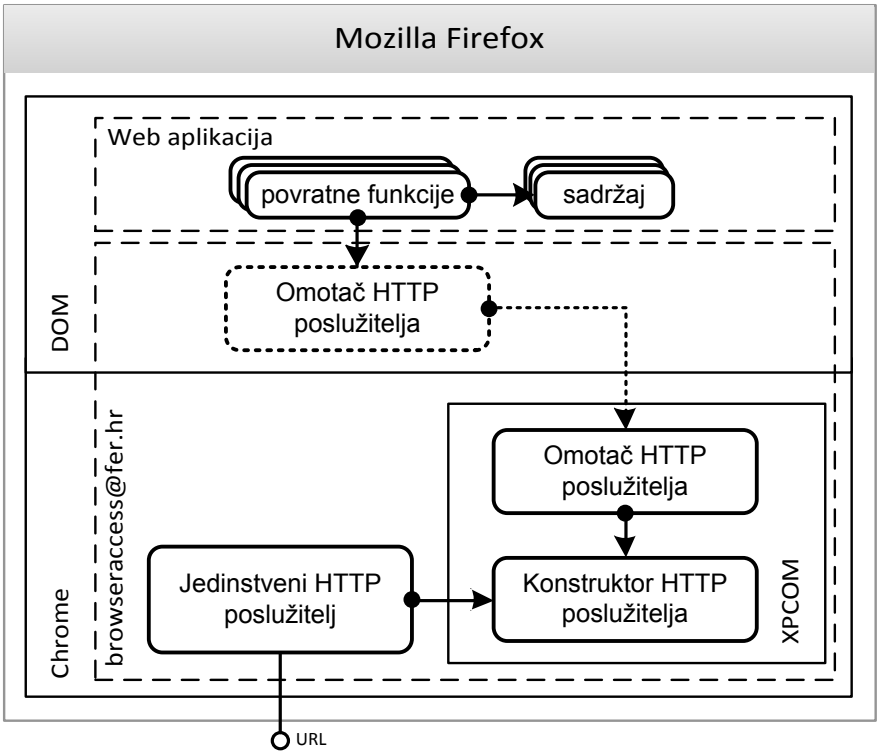
Poslužitelju ugrađenom u preglednik moguće je uz odgovarajuće dozvole pristupiti pomoću tehnologije XPConnect [2] koja služi kao most između JavaScript radnog okruženja i XPCOM komponenata. Programsko sučelje ugrađenog poslužitelja nije prilagođeno za izravno korištenje u Web stranicama. Sučelje nudi mogućnost pokretanje poslužitelja na određenim vratima, posluživanje sadržaja sa sustava datoteka (eng. *file system*) korisničkog računala, čitanje iz toka podataka zahtjeva, pisanje u tok odgovora i mnoge druge napredne mogućnosti koje zbog sigurnosti i jednostavnosti upotrebe ne bi smjele biti dostupne u kontekstu Web stranica. Iz tog je razloga potrebno u kontekstu Web stranica ostvariti sučelje prema ugrađenom poslužitelju.

Sučelje prema ugrađenom poslužitelju ostvareno je kao XPCOM komponenta koja se ugrađuje u DOM strukturu svake Web stranice. Ta je komponenta omotač jedinstvenog (eng. *singleton*) poslužitelja prema kojem nudi ograničeno sučelje i dodatnim ograničenjima donekle osigurava od zloupotrebe poslužitelja. Pristup poslužitelju se omogućava prijavom window elementa, korijenskog čvora DOM strukture Web stranice, omotaču poslužitelja. Omotač zatim pokreće poslužitelj ukoliko već nije pokrenut,

[1] U Web pregledniku Mozilla Firefox, skripte unutar proširenja izvršavaju se sa chrome dozvolama, odnosno bez ikakvih sigurnosnih ograničenja

i sprema prijavljeni window element u registar omotača. Iz konteksta Web stranice se zatim mogu prijavljivati JavaScript funkcije za obradu zahtjeva omotaču koristeći window element za autentifikaciju Web stranice.

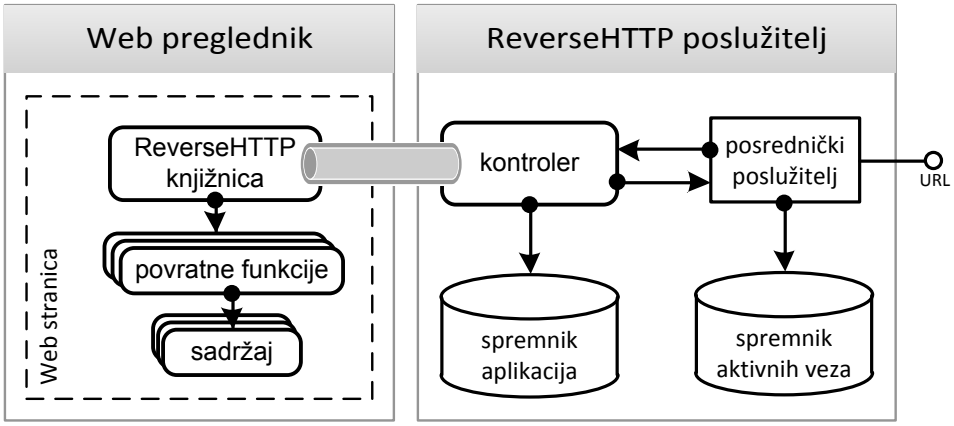
Omotač poslužitelja osim što nudi sučelje prema poslužitelju, također i prilagođava sučelje JavaScript funkcija za obradu zahtjeva. Sučelje funkcija za obradu zahtjeva ugrađenog poslužitelja pretpostavlja obradu u XPCOM okruženju zbog čega se podacima u zahtjevima i odgovorima pristupa preko tokova podataka, a u kontekstu Web stranice ne postoji programsko sučelje za omogućavanje takvog pristupa.



Slika 4.2: Arhitektura ostvarenog proširenja Web preglednika Mozilla Firefox

Arhitektura ostvarenja proširenja opisana je na slici 4.2. Konstruktor poslužitelja i omotač poslužitelja su XPCOM komponente. Omotač nema izravnog doticaja s poslužiteljem, nego samo s njegovom jedinstvenom instancom. Tako se osigurava da se iz konteksta Web stranica ne može pokrenuti velik broj poslužitelja što bi usporilo rad korisničkog računala i otvorilo velik broj vrata na mrežnom sučelju. Omotaču poslužitelja moguće je pristupiti iz DOM okruženja, odnosno konteksta Web stranice. Sredstva Web stranice prijavljuju se za posluživanje omotaču poslužitelja pomoću povratnih funkcija koje se izvršavaju u DOM okruženju.

4.1.2. ReverseHTTP usluga



Slika 4.3: Arhitektura ostvarene ReverseHTTP usluge

Arhitektura ostvarene ReverseHTTP usluge prikazana je na slici 4.3. Usluga se sastoji od korisničkog i poslužiteljskog dijela. Na korisničkoj strani, odnosno u pregledniku, nalazi se ReverseHTTP knjižnica koja skriptama Web stranice nudi programsko sučelje prema ReverseHTTP usluzi. Poslužitelj sadrži dva sučelja, jedno za prijavu i razmjenu poruka s prijavljenim preglednicima, a drugo prema korisnicima sredstava koji žele pristupiti sredstvima prijavljenih preglednika.

Sučelje prema prijavljenim preglednicima na slici je označeno kao kontroler. ReverseHTTP knjižnica prilikom prijave uspostavlja vezu s kontrolerom poslužitelja i šalje željeno ime aplikacije koju će preglednik posluživati. Kontroler gledanjem u spremnik aplikacija provjerava da li je željeno ime aplikacije već zauzeto. Ako jest, na kraj imena dodat će se redni broj. U protivnom, u spremnik aplikacija sprema se novo ime aplikacije koje se mapira na objekt koji predstavlja vezu s ReverseHTTP knjižnicom preglednika koji je prijavio aplikaciju.

Posrednički poslužitelj poslužuje sve zahtjeve koji dolaze na poddomene domene vanjskog sučelja poslužitelja. Prilikom zaprimanja zahtjeva na javno sučelje, posrednički poslužitelj zaustavlja sinkronu obradu zahtjeva, generira jedinstveni identifikacijski ključ te ga mapiranog na objekt zaprimljenog zahtjeva sprema u spremnik aktivnih veza. Nakon toga se zahtjev prosljeđuje kontroleru koji u spremniku aplikacija pokušava naći aplikaciju s imenom koje odgovara prefiksu domene na koju je upućen proslјeđeni zahtjev. Ukoliko takva aplikacija ne postoji u spremniku, kontroler javlja posredničkom poslužitelju da izbriše zahtjev iz spremnika zahtjeva i odgovori na zahtjev porukom HTTP greške 404 koja označava nepostojanje zatraženog sredstva.

Ako aplikacija postoji u spremniku aplikacija, iz spremnika se dohvaća veza prema pregledniku koji poslužuje aplikaciju, te se zahtjev prosljeđuje tom pregledniku koristeći programsko sučelje ReverseHTTP knjižnice. Knjižnica pronalazi odgovarajuću funkciju za obradu zahtjeva pa joj prosljeđuje zahtjev. Funkcija preuzima zahtjev i asinkrono ga obrađuje. Nakon obrade, odgovor se šalje natrag kontroleru koji ga prosljeđuje posredničkom poslužitelju. Posrednički poslužitelj iz spremnika aktivnih veza dohvaća odgovarajuću vezu pomoću prethodno generiranog identifikacijskog ključa, te preko nje šalje odgovor krajnjem korisniku.

4.1.3. Ujedinjeno sučelje

Ujedinjeno sučelje za pristup korisničkim procesima u Web pregledniku nudi korisnicima pristup usluzi za posluživanje sredstava u kontekstu Web stranice apstraktno o ostvarenju usluge. Sučelje provjerava da li je u pregledniku ugrađen HTTP poslužitelj. Ako jest, ujedinjeno sučelje će ga koristiti kao pretpostavljeni način pristupa korisničkim procesima u Web pregledniku. Ukoliko HTTP poslužitelj nije ugrađen u preglednik, ujedinjeno sučelje će uspostaviti vezu s jednim od ReverseHTTP poslužitelja prethodno navedenih u listi dostupnih ostvarenja ReverseHTTP usluge.

Arhitektura ujedinjenog sučelja prikazana je na slici 4.1. Okvir s nazivom *knjižnica* obuhvaća sve dijelove ujedinjenog sučelja za pristup korisničkim procesima u Web pregledniku. Središnji objekt knjižnice je objekt BrowserAccess. On služi za transparentno korištenje poslužitelja ugrađenog u preglednik ili ReverseHTTP usluge. Svako ostvarenje ReverseHTTP usluge mora se prilagoditi ujedinjenom sučelju ostvarivanjem ReverseHTTP adaptera. Ti se adapteri spremaju u bazu ReverseHTTP adaptera. U slučaju da preglednik nema ugrađen HTTP poslužitelj, ujedinjeno sučelje će redom provjeravati dostupnost ReverseHTTP usluga iz baze. Koristit će prvu uslugu s kojom uspije uspostaviti vezu.

Sredstva se na sučelje mogu prijavljivati izravno kao povratne CGI funkcije, ili koristeći sučelje za ostvarenje aplikacija. Prednost korištenja aplikacija jest mogućnost prijave jedne aplikacije za obradu više putanja. Aplikacija je objekt koji obrađuje zahtjeve upućene na podputanje korijenske putanje. Na aplikaciju se mogu prijavljivati povratne funkcije za obradu određenih podputanja. Aplikacija stvara stablastu strukturu funkcija za obradu pomoću koje usmjerava zahtjev na odgovarajuću funkciju za obradu.

4.2. Programsko ostvarenje sustava

Sustav za ostvarivanje pristupa korisničkim procesima u Web pregledniku sastoji se od dva ostvarenja pristupa. Prvi ugrađuje HTTP poslužitelj u preglednik Mozilla Firefox. Njegovo ostvarenje opisano je u poglavlju 4.2.1. Ostvarenje drugog pristupa, ReverseHTTP usluge, opisano je u poglavlju 4.2.2. Ostvarenje ujedinjenog sučelja za oba pristupa opisano je u poglavlju 4.2.3.

4.2.1. Ostvarenje proširenja preglednika HTTP poslužiteljem

Izravan pristup korisničkim procesima u Web pregledniku Mozilla Firefox ostvaren je proširenjem preglednika koje ugrađuje HTTP poslužitelj u preglednik. U sklopu ovog rada ostvareno je proširenje koje u preglednik ugrađuje Mozillin HTTP poslužitelj za testove jedinki (eng. *HTTP server for unit tests*). Osnovna primjena odabranog poslužitelja jest izvršavanje testova jedinki (eng. *unit test*), a predviđeno radno okruženje mu je JavaScript ljuska s pristupom XPCOM komponentama. U proširenju Web preglednika Mozilla Firefox ovaj se poslužitelj može koristiti kao XPCOM komponenta. XPCOM komponente moguće je razviti u jezicima C, C++, Python ili JavaScript. Programski jezik ostvarenja neovisan je o njezinom korištenju. Transparentnost tehnologije ostvarenja omogućuje definicija sučelja u jeziku IDL (eng. *Interface description language*) koja ograničava ostvarenje te uvodi tipove podataka koji ne postoje u jeziku JavaScript. Kod ugradnje XPCOM komponente u proširenje, potrebno je prevesti IDL definiciju sučelja komponente u xpt datoteku odgovarajućim xpidl prevoditeljem koji je sastavni dio xulrunner razvojnog alata (eng. *Software Development Kit, SDK*).

Ugrađeni HTTP poslužitelj sastoji se od sljedećih XPCOM komponenti: *nsHttpServer*, *nsHttpStoppedCallback*, *nsHttpServerIdentity*, *nsHttpServerRequestHandler*, *nsHttpRequest* i *nsHttpResponse*. *nsHttpServer* predstavlja ugrađeni poslužitelj i sadrži programsko sučelje za pokretanje poslužitelja, te prijavu i odjavu sredstava. Komponenta *nsHttpStoppedCallback* predstavlja povratnu funkciju koja će se pozvati prilikom zaustavljanja poslužitelja. *nsHttpServerIdentity* predtavlja identitet poslužitelja, a sadrži protokol, domenu i vrata na kojima je poslužitelj dostupan. Ako do poslužitelja dođe zahtjev koji nije upućen na jedan od prethodno određenih identiteta, poslužitelj odbija obraditi takav zahtjev. To ograničenje u poslužitelj je ugrađeno zbog toga što mu je glavna namjena izvođenje testova jedinki, pa nema smisla omogućavati pristup poslužitelju s mreže. To je ograničenje izbačeno iz ostvarenja da bi se omogućio pristup s bilo kojeg identiteta, jer je pretpostavka da u velikom broju slučajeva koris-

nik neće prethodno znati identitet preko kojeg će poslužitelj biti dostupan budući da će se izvršavati u Web pregledniku u mreži s nepoznatom konfiguracijom. Komponenta *nsHttpServerRequestHandler* ostvaruje povratnu funkciju za obradu zahtjeva, a *nsHttpRequest* i *nsHttpResponse* zahtjev i odgovor koje povratna funkcija prima kao argumente.

Omotač poslužitelja ostvaruje komponente *nsHttpServerWrapper* koja omata komponentu *nsHttpServer*, *nsHttpRequestHandlerWrapper* koja omata *nsHttpRequestHandler*, te *nsHttpRequestWrapper* i *nsHttpResponseWrapper* koji omataju komponente *nsHttpRequest* i *nsHttpResponse*. Navedeni omotači ograničavaju pristup omotanim komponentama, te tako omogućuju jednostavnije i sigurnije korištenje poslužitelja. Sučelje omotača poslužitelja definirano je ovako:

```
interface nsIHttpServerWrapper : nsISupports {
    readonly attribute boolean running;

    boolean requestUsage(in nsIDOMWindow window);
    boolean unrequestUsage(in nsIDOMWindow window);

    string registerPathHandler(in nsIDOMWindow window,
        in string path,
        in nsIHttpRequestHandlerWrapper handler);
    boolean unregisterPathHandler(in nsIDOMWindow window,
        in string path);

    string registerApplication(in nsIDOMWindow window,
        in string appName,
        in nsIHttpRequestHandlerWrapper app);
    boolean unregisterApplication(in nsIDOMWindow window,
        in string appName);
};
```

Sučelje omotača poslužitelja sadrži atribut *running* koji označava da li je poslužitelj pokrenut ili nije. Metode *requestUsage* i *unrequestUsage* prijavljuju, odnosno odjavljuju Web stranicu na omotač, odnosno od omotača poslužitelja. Kao identifikacija Web stranice, koristi se vršni element DOM strukture, window objekt. Metode *registerPathHandler* i *unregisterPathHandler* prijavljuju, odnosno odjavljuju povratne funkcije za obradu zahtjeva. Metode *registerApplication* i *unregisterApplication* pri-

javljuju, odnosno odjavljuju aplikaciju. Razlika između povratne funkcije i aplikacije je u tome što povratna funkcija obrađuje zahtjeve poslane na točno određenu putanju, dok aplikacija obrađuje zahtjeve poslane na bilo koju podputanju prijavljene putanje. Sve navedene metode koriste objekt window za identifikaciju Web stranice kao korisnika poslužitelja.

Sučelje omotača povratne funkcije definirano je ovako:

```
[scriptable, function, uuid(...)]
interface nsIHttpRequestHandlerWrapper : nsISupports {
    void handle(in nsIHttpRequestWrapper request,
        in nsIHttpResponseWrapper response);
};
```

U dodatnim podacima definicije sučelja navedena je ključna riječ *function* koja označava da komponenta nije objekt, već funkcija, tako da povratna funkcija ne mora sadržavati metodu handle, već ju izravno ostvaruje. Povratna funkcija prima dva parametra, omotani zahtjev i odgovor. Zadaća je funkcije da pročita zahtjev, upiše rezultat obrade u odgovor i asinkrono ga vrati pošiljatelju zahtjeva. Ugrađeni poslužitelj nudi mogućnost sinkrone i asinkrone obrade zahtjeva a omotač zbog jednostavnosti sučelja odbacuje mogućnost sinkrone obrade zahtjeva.

Sučelje omotača zahtjeva *nsHttpRequestWrapper* i odgovora *nsHttpResponseWrapper* definirana su ovako:

```
interface nsIHttpRequestWrapper : nsISupports {
    readonly attribute string bodyText;
    readonly attribute string url;
    readonly attribute string version;
    readonly attribute string method;
    readonly attribute string queryString;

    string getHeader(in string key);

    void initWrappedJSObject();
};

interface nsIHttpResponseWrapper : nsISupports {
    attribute string bodyText;
    attribute unsigned short status;
```

```
    void setHeader(in string key,
        in string value,
        in boolean merge);
    void send();

    void initWrappedJSObject();
};
```

Vidljivo je da omotači definiraju sučelje koje je prilagođeno korištenju u kontekstu Web stranice. Omotač zahtjeva nudi nekoliko atributa u koje ne može pisati (eng. *readonly*) te metodu za dohvaćanje zaglavlja zahtjeva *getHeader*. Omotač odgovora nudi dva atributa u koje se može pisati: *bodyText* i *status*. Također sadrži i metodu za postavljanje zaglavlja odgovora te metodu za slanje zahtjeva. Samo postojanje te metode znači da se obrada zahtjeva vrši asinkrono kao što je prethodno već napomenuto. I omotač zahtjeva i omotač odgovora sadrže metodu *initWrappedJSObject*. Ova se metoda koristi za pojednostavljenje pristupa zaglavljima zahtjeva i odgovora. Metoda stvara JavaScript objekt u koji upisuje JavaScript objekt headers u kojeg se zapisuju sva zaglavlja zahtjeva, odnosno odgovora. Tako se zaobilazi ograničenje omotača preko kojeg inače ne bi bilo moguće dohvatiti sva zaglavlja zahtjeva, odnosno odgovora, jer su oni spremljeni u XPCOM objekte koje bi u kontekstu Web stranice bilo nemoguće izravno čitati.

4.2.2. Ostvarenje ReverseHTTP Web usluge

U sklopu ovog rada ostvarena je ReverseHTTP usluga u tehnologiji Jetty u programskom jeziku Java. Za spajanje na ostvarenu uslugu iz preglednika koristi se ostvarena JavaScript knjižnica koja uspostavlja vezu koristeći HTML5 WebSocket standard. Specifikacija WebSocket standarda još se razvija, pa su bila potrebna određena zaobilazna rješenja za ostvarivanje podrške za sve moderne preglednike. Web preglednik Mozilla Firefox verzija 4 i 5 imaju podršku za WebSocket, ali zbog sigurnosnog propusta u specifikaciji postavljeno je da korisnik u postavkama mora izričito uključiti podršku za WebSocket. Trenutna stabilna verzija 6 uključuje ostvarenje nove specifikacije WebSocket standarda s popravljenim sigurnosnim propustom, ali zbog toga što se specifikacija još razvija, u DOM okruženju ne postoji objekt *WebSocket*, već objekt *mozWebSocket*. Također, različiti preglednici ostvaruju različite verzije protokola, ali u većini slučajeva najnovije verzije poslužitelja i preglednika mogu uspostaviti Web-

Socket vezu iako ne ostvaruju istu verziju protokola.

Nakon uspostave WebSocket veze s poslužiteljem, iz skripte u Web stranici moguće je prijaviti aplikacije za obradu zahtjeva. Prilikom prijave aplikacije, ReverseHTTP knjižnica sprema objekt aplikacije u lokalnu mapu, a naziv aplikacije šalje poslužitelju. Poslužitelj odgovara da li je prijava uspjela, i ako jest, na kojoj je domeni dostupna. Poslužitelj će zahtjeve na sve putanje te domene preko WebSocket veze prosljeđivati ReverseHTTP knjižnici koja će obradu zahtjeva delegirati prijavljenoj aplikaciji.

4.2.3. Ostvarenje ujedinjenog sučelja

Ujedinjeno sučelje za izravan i posredovan pristup Web pregledniku ostvaren je kao JavaScript knjižnica koja se izvršava u kontekstu Web stranice. Knjižnica u DOM okruženje ugrađuje objekte *BrowserAccess*, *PathTreeNode*, *Application*, sučelje *ReverseHttpServiceAdapter* i mapu prilagodnika *ReverseHttpServiceAdapters*.

Objekt *BrowserAccess* središnji je element knjižnice. On pruža sučelje za pristup proširenju Web preglednika, odnosno ReverseHTTP usluzi, ovisno o tome je li preglednik proširen HTTP poslužiteljem. *BrowserAccess* sadrži metodu *init* koja provjerava da li je u preglednik ugrađen poslužitelj. Ako jest, metoda u objekt ugrađuje metode *registerApplication* i *unregisterApplication* za prijavu, odnosno odjavu aplikacija. Te metode zovu istoimene metode proširenja, a podržana je i prijava povratnih funkcija za obradu događaja. Izravno podržani događaji su događaji greške *onError* i uspjeha *onSuccess*, ali podržani su i proizvoljni događaji definirani od strane korisnika.

Ako preglednik ne sadrži proširenje koje ugrađuje HTTP poslužitelj, metoda *init* pokušat će uspostaviti vezu s nekim ReverseHTTP poslužiteljem. Metoda će prolaziti kroz mapu prilagodnika *ReverseHttpServiceAdapters* sve dok ne nađe dostupnu ReverseHTTP uslugu. ReverseHTTP usluzi se pristupa preko prilagodnika da bi se postigla nezavisnost ujedinjenog sučelja i ostvarenja ReverseHTTP usluge. Tako ujedinjeno sučelje može koristiti bilo koje ostvarenje ReverseHTTP usluge, a svaka ReverseHTTP usluga može se koristiti i bez ujedinjenog sučelja. U slučaju korištenja ReverseHTTP usluge mogućnnost obrade proizvoljnih događaja dolazi do izražaja jer se mogu obraditi događaji specifičnih ostvarenja ReverseHTTP usluga.

5. Mjerenja

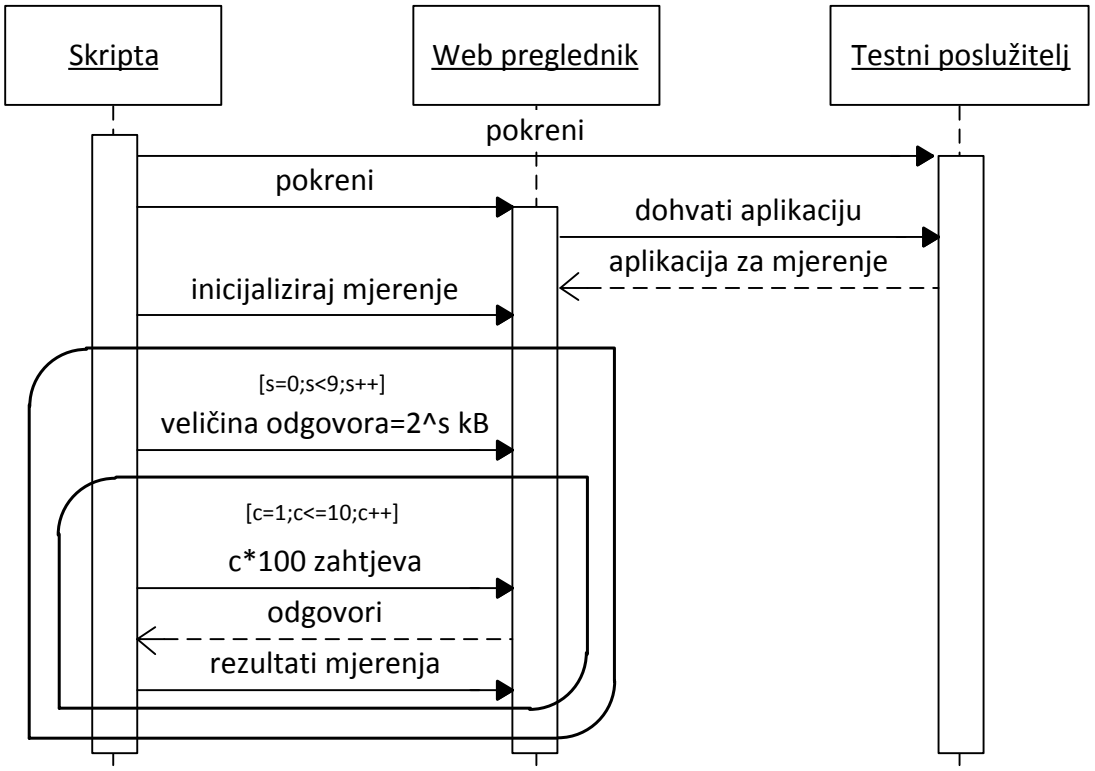
Cilj mjerenja sustava za pristup korisničkim procesima u Web pregledniku je usporedba dvaju ostvarenih načina pristupa Web pregledniku. Ostvaren sustav za mjerenje opisan je u poglavlju 5.1, a rezultati mjerenja dani su u poglavlju 5.2.

5.1. Sustav za mjerenje

Sustav za mjerenje sastoji se od BASH skripte, testnog poslužitelja koji poslužuje aplikaciju za mjerenje, te Web preglednika. Budući da je cilj mjerenja usporedba dvaju ostvarenih načina pristupa, mjerenja su provedena u pregledniku koji ima ugrađen HTTP poslužitelj, i ima pristup ReverseHTTP usluzi. Iz tog je razloga odabrana stabilna verzija preglednika Mozilla Firefox 6.0.1 s ugrađenim proširenjem Browser Access 0.1a0. Mjerenja su provedena na računalu s procesorom Intel(R) Core(TM)2 Duo T9300 2.50GHz, 4GB radne memorije koje pokreće Linux operacijski sustav.

Sustav za mjerenje u Firefox pregledniku pokreće dva sučelja za pristup korisničkim procesima. Jedan koristi proširenje za pristup, a drugi ReverseHTTP uslugu. Da bi se osiguralo što pravednije mjerenje, oba sučelja poslužuju istu aplikaciju. Aplikacija kao odgovor vraća proizvoljan niz znakova prethodno zadane veličine. Da bi mjerenje moglo biti izvršeno, potrebno je sinkronizirati rad skripte i Web preglednika. Iz tog razloga aplikacija koju poslužuje sustav za pristup korisničkim procesima u Web pregledniku poslužuje dva dodatna sredstva preko kojih se izmjenjuju kontrolne poruke mjerenja. Prvo sredstvo *measure-control* služi za slanje kontrolnih poruka za provjeru stanja poslužitelja i postavljanja veličine odgovora. Drugo sredstvo *measure-result* služi za slanje i vizualizaciju rezultata mjerenja u pregledniku.

Dijagram tijeka procesa mjerenja dan je na slici 5.1. Na samom početku skripta pokreće testni poslužitelj i ReverseHTTP poslužitelj. Nakon toga pokreće Web preglednik koji dohvaća aplikaciju za mjerenje s testnog poslužitelja. Testni poslužitelj služi za omogućavanje korištenja WebSocketa jer skripte pokrenute izravno s datotečnog sustava nemaju dozvolu otvaranja WebSocket veza. Nakon pokretanja ap-



Slika 5.1: Dijagram tijeka sustava za mjerenje

likacije, skripta pregledniku šalje kontrolnu poruku *ready* koja u pregledniku pokreće provjeru stanja poslužitelja. Ukoliko je preglednik spreman za početak testiranja, počinje inicijalizacija mjerenja. Skripta prvo pomoću alata za mjerenje radnih svojstava poslužitelja *ab* [8] pokreće testno mjerenje s ciljem završne provjere prije početka mjerenja. Nakon uspješne provjere, skripta pregledniku šalje podatke o cjelokupnom mjerenju u svrhu inicijaliziranja vizualizacije rezultata mjerenja.

Mjerenje se izvršava u dvostrukoj petlji gdje prva petlja u svakoj iteraciji dvostruko povećava veličinu odgovora aplikacije, a ugrađena petlja povečava broj konkurentnih zahtjeva za 100. Mijenjanjem postavki, moguće je izvršiti mjerenje s drugačijim brojem iteracija, početnom veličinom zahtjeva, korakom povećanja broja konkurentnih zahtjeva i ponavljanja zahtjeva.

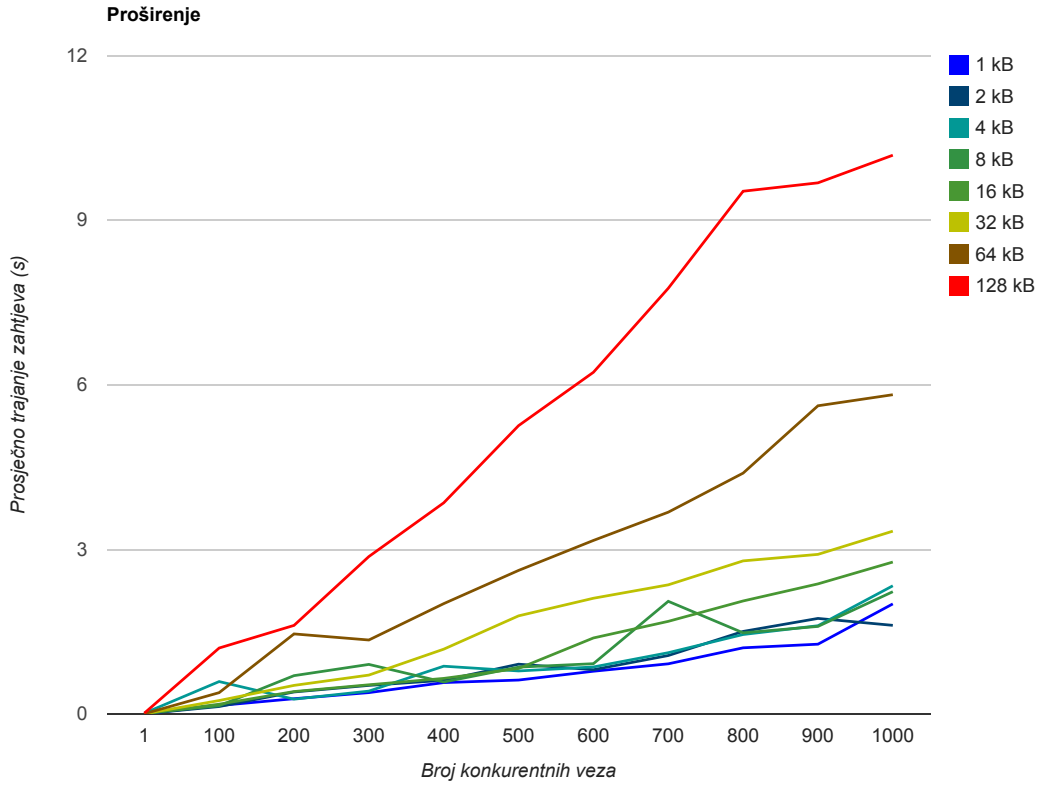
Izlaz alata *ab* šalje se izravno pregledniku koji ga parsira i iz njega izvlači trajanje obrade što sprema u unutarnje strukture podataka. Na kraju jedne iteracije mjerenja, skripta šalje kontrolnu poruku *end* koja označava završetak izvođenja alata ab. Preglednik u tom trenutku provjerava da li je primio očekivan broj rezultata. Ako nije, mjerenje se ponavlja. Odbacivanje zahtjeva je moguće zbog preopterećenja poslužitelja. Tijekom mjerenja, znalo se dogoditi da HTTP poslužitelj ugrađen u preglednik

zagubi neke zahtjeve, ili da u potpunosti prestane posluživati zahtjeve. U tom se slučaju se ponovo inicijalizira proširenje, pa se mjerenje ponavlja. Tijekom mjerenja nije se dogodilo da ReverseHTTP usluga zagubi zahtjeve.

Nakon što preglednik dobije očekivan broj rezultata mjerenja, ti se podaci u stvarnom vremenu vizualiziraju korištenjem Google visualization programskog sučelja [10]

5.2. Rezultati mjerenja

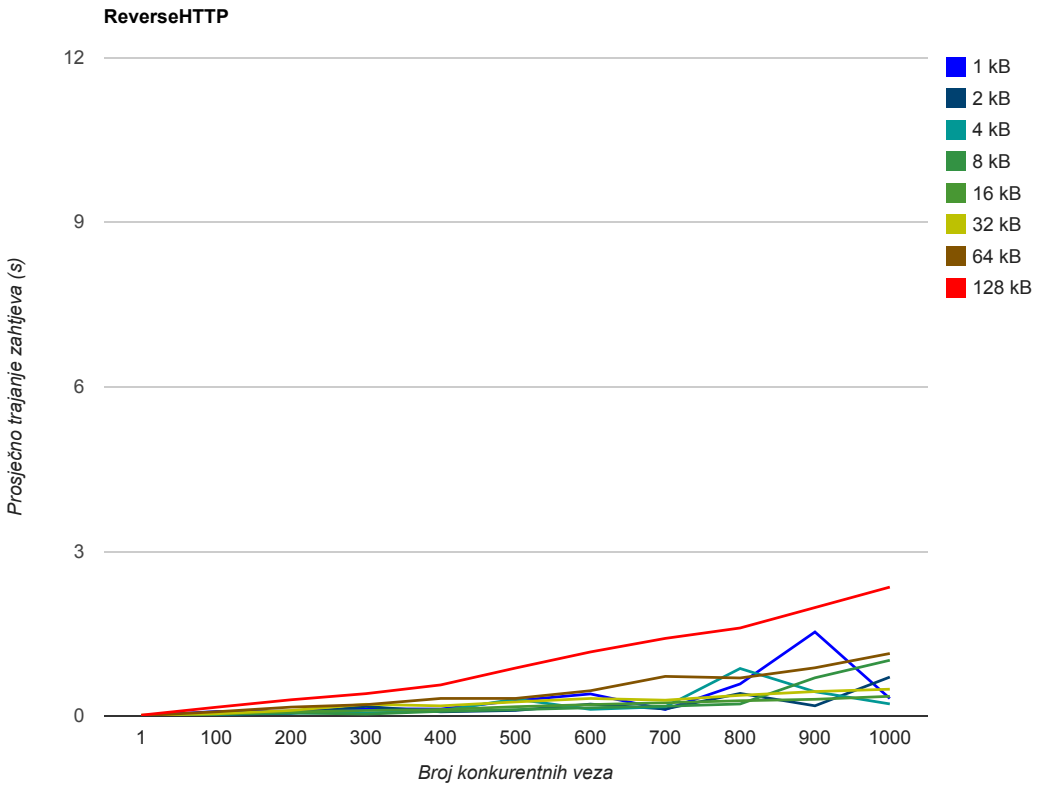
Mjerenjem se dobivaju sva trajanja zahtjeva u različitim scenarijima mjerenja. Svaka iteracija mjerenja korištenjem alata *ab* šalje *C* broj konkurentnih zahtjeva, a sustav za pristup korisničkim procesima u Web pregledniku vraća odgovor veličine *S*. Za oba načina pristupa procesima u pregledniku grafovima za svaku iteraciju mjerenja prikazujemo prosječno vrijeme obrade zahtjeva i standardnu devijaciju.



Slika 5.2: Prosječno trajanje obrade zahtjeva proširenja Web preglednika

Graf na slici 5.2 prikazuje prosječno trajanje zahtjeva korištenjem proširenja preglednika u ovisnosti o veličini odgovora i broju konkurentnih veza. Vidi se da pros-

ječno trajanje zahtjeva uz manja odstupanja linearno raste s brojem konkurentnih zahtjeva i veličinom odgovora što je očekivano ponašanje ugrađenog poslužitelja.



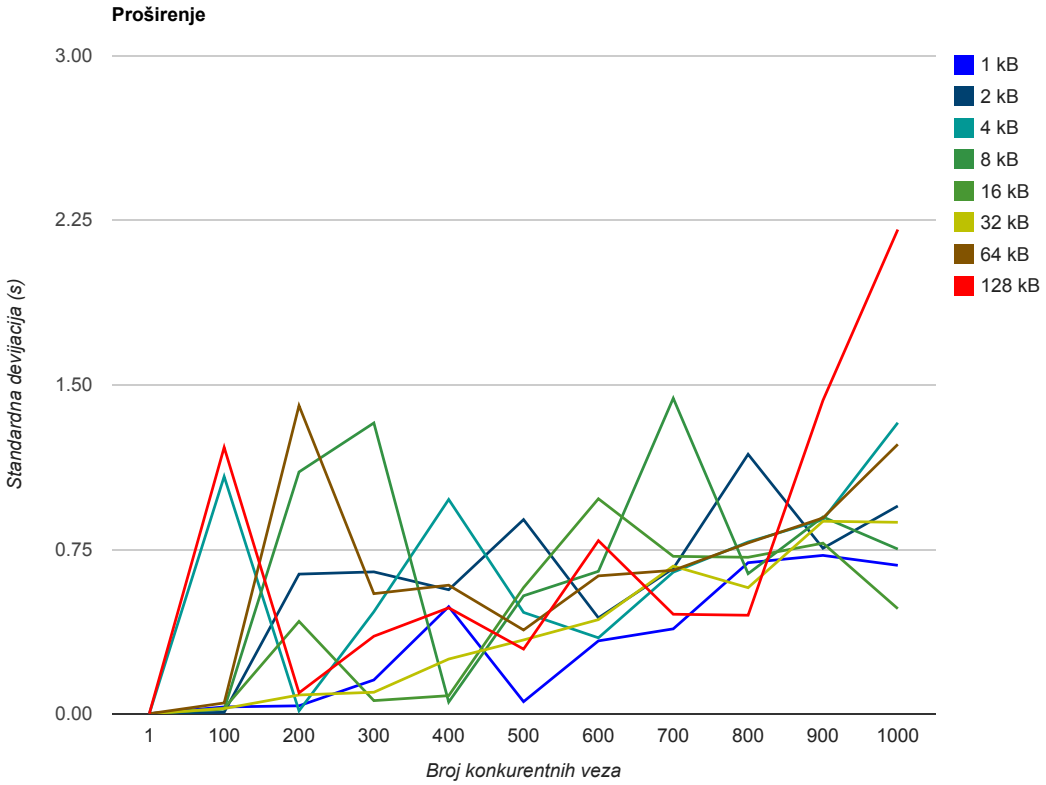
Slika 5.3: Prosječno trajanje obrade zahtjeva ReverseHTTP posrednika

Slika 5.3 prikazuje isti graf kao i 5.2, samo za pristup korištenjem ReverseHTTP usluge. Vidljivo je da i prilikom korištenja ReverseHTTP usluge trajanje obrade skoro linearno prati veličinu odgovora i broj konkurentnih zahtjeva, ali to je trajanje zamjetno manje nego u slučaju korištenja proširenja. Razlog tome vrlo je vjerojatno činjenica da ugrađeni poslužitelj nije optimiran za brzinu budući da mu je glavna namjena izvođenje testova jedinki. Ugrađeni poslužitelj ostvaren je u programskom jeziku JavaScript koji ne podržava višedretvenost, a nije ni poznat po velikoj brzini. Također, važno je primjetiti da se ugrađeni poslužitelj ne koristi optimalno zbog prilagodbe poslužitelja izvođenju u DOM okruženju. Rezultati mjerenja bili bi nešto bolji da se kod svakog zahtjeva ne prepisuju sva zaglavlja paketa, i da se podacima zahtjeva pristupa preko tokova podataka.

ReverseHTTP usluga ostvarena je u programskom jeziku Java korištenjem Jetty Web poslužitelja. Jetty podržava višedretvenost, pa može usporedno posluživati više zahtjeva. Ipak, zbog jednodretvenosti Web preglednika kojem se delegira obrada za-

htjeva, u testovima ta višedretvenost ne dolazi do izražaja. Prednost višedretvenog rada došao bi do izražaja kad bi se na poslužitelj prijavilo više Web preglednika, ali takav je test teško izvršiti zbog ograničenja resursa na lokalnom računalu, a testiranje u mrežnom okruženju unosi brojne nekonzistencije u mjerenje.

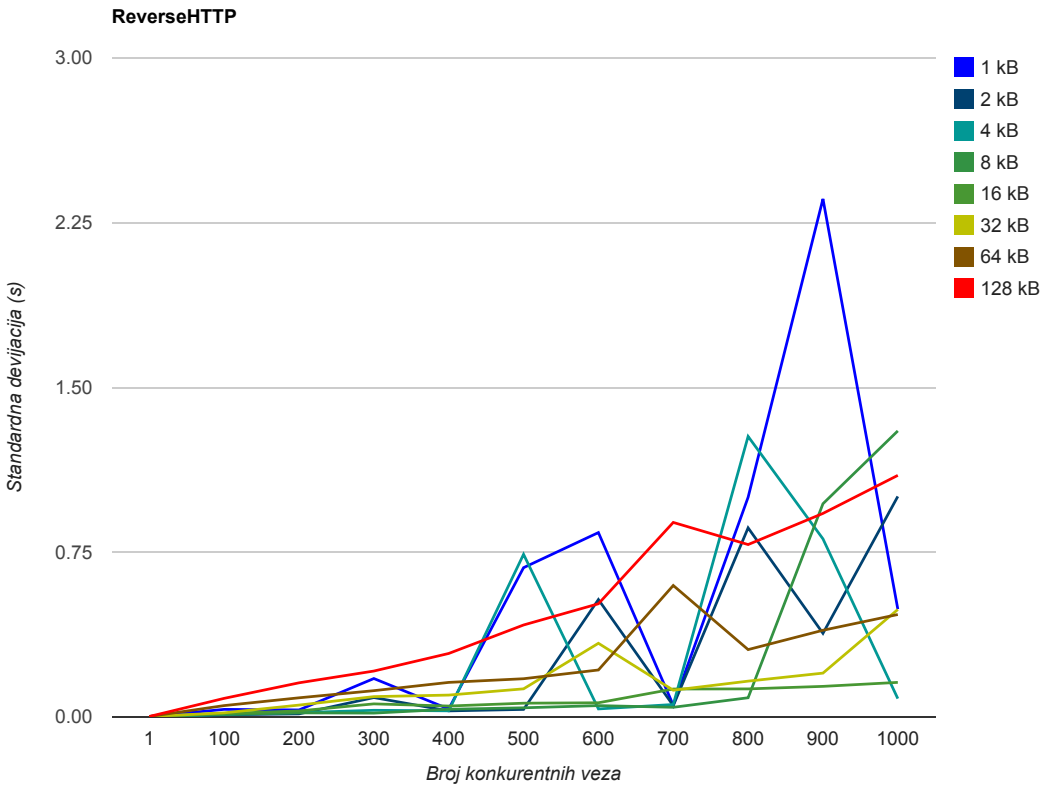
Na grafu se može primjetiti blago nelinearno povećanje trajanja obrade zahtjeva što se može pripisati višedretvenosti Jetty poslužitelja. Zbog usporednog posluživanja više zahtjeva, dolazi do opterećenja poslužitelja. Iako se u pregledniku zahtjevi obrađuju slijedno, ReverseHTTP poslužitelj počet će obradu više usporednih zahtjeva čiji će se konteksti spremiti u memoriju i tako usporiti poslužitelj. S grafa je jasno vidljivo da s povećanjem veličine poruke nelinearnost postaje zamjetnija. S druge strane, zbog potpuno slijedne obrade zahtjeva, kod ugrađenog poslužitelja nelinearnost je manje zamjetna.



Slika 5.4: Standardna devijacija trajanja obrade zahtjeva proširenja Web preglednika

Slika 5.4 prikazuje standardnu devijaciju trajanja obrade zahtjeva prikazanog na slici 5.2. Standardna devijacija daje nam uvid u razinu determinantnosti sustava, odnosno mogućnost pretpostavke trajanja obrade zahtjeva uz poznato opterećenje poslužitelja. Budući da ugrađeni poslužitelj poslužuje zahtjeve potpuno slijedno, ne mogu

postojati dva usporedna zahtjeva koja će imati isto trajanje obrade zahtjeva, već bi njihovo trajanje obrade trebalo linearno rasti u ovisnosti o poretku reda čekanja. Uz navedenu pretpostavku, standardna devijacija bi u idealnom slučaju trebala rasti linearno s brojem konkurentnih veza i veličine odgovora. Sa slike mogu se primjetiti velika odstupanja od idealnog slučaja što ukazuje na nederminantnost ugrađenog poslužitelja. Također je vidljivo da standardna devijacija kod najveće veličine zahtjeva i velikog broja konkurentnih veza skače u ekstrem što ukazuje na preopterećenje poslužitelja. Na preopterećenje također ukazuje činjenica da se poslužitelj počeo gušiti, pa čak je nekoliko puta i prestao raditi kod tog opterećenja, pa nije imalo smisla povećavati veličinu odgovora i broj konkurentnih veza.



Slika 5.5: Standardna devijacija trajanja obrade zahtjeva ReverseHTTP posrednika

Slika 5.5 prikazuje standardnu devijaciju trajanja obrade zahtjeva korištenjem ReverseHTTP poslužitelja. Sa slike vidljivo je puno stabilnije ponašanje nego kod ugrađenog poslužitelja. Ipak, na par mjesta zabilježena su značajnija odstupanja koja se mogu pripisati Javinom čistaču memorije (eng. *garbage collector*) budući da sustav koristi veliku količinu memorije koja se odmah nakon korištenja odbacuje. Ta bi se odstupanja mogla otkloniti izravnim pozivanjem čistača memorije prije svakog pozi-

vanja alata *ab*.

Kod manjeg broja konkurentnih zahtjeva, standardna devijacija kod korištenja ReverseHTTP poslužitelja osjetno je stabilnija od ugrađenog poslužitelja. Uzrok ovakvih rezultata leži u tehnologiji ostvarenja poslužitelja, višedretvenosti i učinkovitijem radu s memorijom.

6. Primjer korištenja

Sustav za ostvarivanje pristupa korisničkim procesima u Web pregledniku može se iskoristiti u brojne svrhe a u ovom poglavlju su nabrojane samo neke od njih.

Prva i osnovna ideja ostvarivanja pristupa korisničkim procesima u Web pregledniku je posluživanje sredstva iz okruženja Web stranice u Web pregledniku na Webu. Jedna od mogućih primjena jest posluživanje korisničkog sadržaja kao što su tekst, slike, datoteke ili sredstva korisničkog računala kao što su Web kamera ili mikrofon. U današnje vrijeme ubrzanog razvoja Web standarda, Web preglednicima je na raspolaganju sve više računalnih sredstava. Standard HTML5 uvodi nova DOM sučelja kao što su *Device*, *Stream*, *ConnectionPeer*, *File*, *WebGL* koja Web pregledniku omogućavaju izravan pristup programskim sučeljima i sredstvima operacijskog sustava koja su nekad bila rezervirana za Desktop aplikacije. Na primjer, pomoću *Device* programskog sučelja, Web stranice mogu dobiti izravan pristup uređajima povezanim na korisničko računalo kao što je mikrofon, kamera ili pisač, sučelje *WebGL* u novijim preglednicima omogućava izravan pristup sredstvima grafičke kartice, a *File* sučelje omogućuje ograničen pristup sustavu datoteka korisničkog operacijskog sustava.

Navedena proširenja DOM okruženja stvaraju platformu za izvođenje naprednih aplikacija unutar Web preglednika, pa tako dolazi do izražaja potreba za omogućavanje pristupa tim aplikacijama. HTML5 standard, osim što proširuje mogućnosti Web stranica, širi tu naprednu Web platformu na velik broj različitih uređaja kao što su pametni telefoni televizori i tablet računala.

Sustav za pristup korisničkim procesima u Web pregledniku može se iskoristiti za zaobilaženje fizičkih nedostataka takvih uređaja kod korištenja Web aplikacija. Jedan od nedostataka takvih uređaja je ograničenje ulaza i veličina radnog prostora, zbog kojih je teško ostvariti napredne funkcije koje nude Web preglednici na osobnim računalima. Na primjer, mobilne verzije Web preglednika ne sadrže razvojne alate koji su vrlo korisni kod testiranja i razvoja aplikacija. Ovim se sustavom, uz pretpostavku da uređaj ima podršku za tehnologije potrebne za korištenje sustava, korisnik može povezati na Web aplikaciju koja se izvršava na mobilnom telefonu s osobnog računala

i tako si olakšati razvoj aplikacija prilagođenih takvim uređajima.

Druga jednostavna primjena sustava je ostvarenje RPC sustava za upravljanje procesima u Web stranici ili pregledniku korisnika. Jedan takav jednostavan sustav ostvaren je u sklopu sustava za mjerenje gdje se koristi za sinkronizaciju BASH skripte i aplikacije za mjerenje u Web pregledniku, te za slanje rezultata aplikaciji. Tijekom mjerenja, u stvarnom se vremenu crtaju grafovi koji vizualiziraju rezultate. Za omogućavanje snimanja tih grafova na datotečni sustav, također je korišten sustav za pristup korisničkim procesima u Web pregledniku. Grafovi su crtani korištenjem Google visualization programskog sučelja koje za crtanje grafova koristi SVG standard. Graf u SVG formatu, ugrađen je u HTML stranice, pa ga je nemoguće jednostavno snimiti na datotečni sustav jer nema URL adresu već je dinamički sadržaj u DOM strukturi. Sustav za pristup korisničkim procesima u Web pregledniku iskorišten je za posluživanje tog dinamičkog sredstva DOM strukture kao zasebno HTTP sredstvo, te je tako omogućeno snimanje dinamički generiranih grafova na datotečni sustav korisničkog računala.

Ideja o posluživanju dinamičkog sadržaja Web stranice može se proširiti dodavanjem podrške za DOM događaje. Tako bi se omogućilo upravljanje Web aplikacijom na daljinu što bi moglo naći primjenu kod modernih televizora gdje bi se moglo ostvariti daljinsko upravljanje aplikacijama koje se izvršavaju na televiziji. Kao daljinski upravljač, mogli bi se koristiti pametni telefoni ili, danas sve popularnija, tablet računala.

Središnji motiv specifikacije ReverseHTTP jest ostvarivanje korisničkih Web Hookova. Kontcept Web Hook objašnjen je u poglavlju 2.5. Mogućnost prijave Web Hook sredstva u Web pregledniku omogućila bi nadograđivanje postojećih Web aplikacija i osiguralo veći stupanj međudjelovanja korisnika i poslužitelja što bi kao posljedicu imalo bolje korisničko iskustvo i veću razinu interakcije korisnika s aplikacijom. Web preglednik bi bio u mogućnosti bez velike potrošnje računalnih i mrežnih resursa izvršavati velik broj naprednih Web aplikacija s osvježavanjem stanja u stvarnom vremenu.

6.1. Demo aplikacija

U ovom poglavlju dan je primjer jednostavne Web aplikacije koja koristi ujedinjeno sučelje za pristup korisničkim procesima u Web pregledniku i poslužuje jednostavno sredstvo koje vraća odgovor s tijelo sadržaja *demo*.

Prvi korak kod korištenja sustava je inicijalizacija ujedinjenog sučelja. Nakon toga stvara se instanca aplikacije koja se prijavljuje na ujedinjeno sučelje. Na kraju se

aplikaciji prijavljuje povratna funkcija za obradu zahtjeva.

```
BrowserAccess.init();
var app = new Application();
BrowserAccess.registerApplication('app', app);
app.addResource('/bla/*',
  function(request, response) {
    response.bodyText = 'demo';
    response.headers['Content-Length'] =
    response.bodyText.length;
    response.send();
  }
);
```

Prijavljenom resursu zatim je moguće pristupiti na poddomeni domene poslužitelja na putanji */bla* i svim podputanjama te putanje.

7. Zaključak

Web preglednici su trenutno područje velikog istraživanja i napretka. Moderni Web preglednici su za red veličine brži od onih od prije nekoliko godina, a HTML5 standard obogaćuje Web kao platformu za razvoj bogatih aplikacija kakve prije samo nekoliko godina nisu bile zamislive na Webu. Ideja omogućavanja pristupa aplikacijama na takvoj naprednoj platformi s vremenom počinje sve više dobivati na smislu.

U sklopu ovog rada, ostvaren je sustav za pristup korisničkim procesima u Web pregledniku koji koristi dva načina pristupa Web pregledniku, pomoću proširenja Web preglednika i upotrebom ReverseHTTP posredničkog poslužitelja. Prvi način je izravan jer ugrađuje HTTP poslužitelj izravno u preglednik. Drugi način je posredovan jer se za dodjelu URL adrese sredstvima u Web pregledniku koristi adresni prostor posredničkog ReverseHTTP poslužitelja. Mjerenjem je pokazana razlika dvaju pristupa. Izravan pristup pokazao je loša radna svojstva u odnosu na posredovani, ali u stvarnom radu gdje bi posrednici posluživali sredstva s velikog broja prijavljenih preglednika, do izražaja bi došle prednosti izravnog pristupa.

Ostvarena ReverseHTTP usluga pokazala je vrlo dobra radna svojstva, ali u sustavu s velikim brojem preglednika koji poslužuju sredstva, posrednik bi bio usko grlo sustava. Bilo bi potrebno uvesti velik broj posredničkih poslužitelja da bi se osigurala dobra radna svojstva sustava. S druge strane, izravan pristup osigurava bolje svojstvo razmjernog rasta, ali i lošija radna svojstva. Taj se problem može riješiti ugrađivanjem boljeg poslužitelja u Web preglednik.

U okviru rada ostvareno je jedinstveno sučelje za ostvarivanje izravnog i posredovanog pristupa Web pregledniku. Sučelje je proširivo novim ostvarenjima ReverseHTTP preglednicima, a uz manje preinake, moguće ga je koristiti i s drugim proširenjima Web preglednika, iako po mojem saznanju slična proširenja ne postoje.

Ostvareni sustav može se iskoristiti u razne svrhe, a samo neke od njih spomenute su u ovom radu. Sustav bi primjenu mogao naći velikom broju postojećih Web aplikacija, a teško je predvidjeti moguće primjene u budućnosti zbog velike brzine razvoja Weba kao platforme bogatih aplikacija.

LITERATURA

[1] Python web server gateway interface v1.0, Prosinac 2003. URL http://www.python.org/dev/peps/pep-0333/.

[2] Xpconnect, Listopad 2007. URL https://developer.mozilla.org/en/XPConnect.

[3] Http server for unit tests, Studeni 2010. URL https://developer.mozilla.org/En/Httpd.js/HTTP_server_for_unit_tests.

[4] Html living standard, peer-to-peer connections, Ožujak 2011. URL http://www.whatwg.org/specs/web-apps/current-work/multipage/commands.html#peer-to-peer-connections.

[5] Html living standard, the device element, Ožujak 2011. URL http://www.whatwg.org/specs/web-apps/current-work/multipage/commands.html#devices.

[6] Uniform resource identifer (uri) schemes, Veljača 2011. URL http://www.iana.org/assignments/uri-schemes.html.

[7] Html living standard, stream api, Ožujak 2011. URL http://www.whatwg.org/specs/web-apps/current-work/multipage/commands.html#stream-api.

[8] ab - apache http server benchmarking tool, 2011. URL http://httpd.apache.org/docs/2.0/programs/ab.html.

[9] Beyond html5 - peer-to-peer conversational video, January 2011. URL https://labs.ericsson.com/developer-community/blog/beyond-html5-peer-peer-conversational-video.

[10] Google visualization api reference, 2011. URL http://code.google.com/apis/chart/interactive/docs/reference.html.

[11] Xpcom, Siječanj 2011. URL https://developer.mozilla.org/en/XPCOM.

[12] K.Coar D.Robinson. The common gateway interface (cgi) version 1.1, listopad 2004. URL http://www.ietf.org/rfc/rfc3875.txt.

[13] R. Fielding et al. Hypertext transfer protocol – http/1.1, Lipanj 1999. URL http://www.ietf.org/rfc/rfc2616.txt.

[14] Tony Garnock-Jones. Reverse http, 2009. URL http://www.reversehttp.net/reverse-http-spec.html.

[15] Jesse James Garrett. Ajax: A new approach to web applications, Veljača 2005. URL http://www.adaptivepath.com/ideas/essays/archives/000385.php.

[16] Ian Hickson. Html5, a vocabulary and associated apis for html and xhtml, Travanj 2010. URL http://dev.w3.org/html5/spec/Overview.html.

[17] Ian Hickson. The web socket protocol, Svibanj 2010. URL http://tools.ietf.org/id/draft-hixie-thewebsocketprotocol-76.txt.

[18] P. Francis K. Egevang. The ip network address translator (nat), Svibanj 1994. URL http://www.ietf.org/rfc/rfc1631.txt.

[19] R. Moats. Urn syntax, Svibanj 1997. URL http://www.ietf.org/rfc/rfc2141.txt.

[20] P. Mockapetris. Domain names - implementation and specification, Studeni 1987. URL http://www.ietf.org/rfc/rfc1035.txt.

[21] Paul Osman. What is a webhook, 2010. URL http://wiki.webhooks.org/.

[22] S. Lawrence R. Khare. Upgrading to tls within http/1.1, Svibanj 2000. URL http://www.ietf.org/rfc/rfc2817.txt.

[23] Thomas Robinson. Jsgi specification, v0.2, 2009. URL http://jackjs.org/jsgi-spec.html.

[24] Michael Mealling Ron Daniel Jr. Urc scenarios and requirements, Studeni 1994. URL http://tools.ietf.org/id/draft-ietf-uri-urc-req-00.txt.

[25] Alex Russell. Comet: Low latency data for the browser, Ožujak 2006. URL http://infrequently.org/2006/03/comet-low-latency-data-for-the-browser/.

[26] Sheppy. Same origin policy for javascript, Lipanj 2010. URL https://developer.mozilla.org/en/Same_origin_policy_for_JavaScript.

[27] L. Masinter T. Berners-Lee, R. Fielding. Uniform resource identifier (uri): Generic syntax, Siječanj 2001. URL http://www.ietf.org/rfc/rfc3986.

[28] M. McCahil T. Berners-Lee, L. Masinter. Uniform resource locators, Prosinac 1994. URL http://www.ietf.org/rfc/rfc1738.txt.

[29] W3C/IETF URI Planning Interest Group. Uris, urls, and urns: Clarifications and recommendations 1.0, Rujan 2001. URL http://www.w3.org/TR/uri-clarification.

[30] Anne van Kesteren. Cross-origin resource sharing, Ožujak 2009. URL http://www.w3.org/TR/access-control/.

Sustav za izravnu i posredovanu komunikaciju s procesima u Web pregledniku putem HTTP protokola

Sažetak

U okviru ovog rada proučena je arhitektura Weba i mogućnosti Web platforme za razvoj bogatih aplikacija. Razmatrana je mogućnost ostvarenja pristupa kontekstu Web aplikacije te moguće dobrobiti takvog pristupa. Ostvareno je jedinstveno sučelje za omogućavanje izravnog i posredovanog pristupa korisničkim procesima u Web pregledniku. Izravan pristup ostvaren je kao proširenje Web preglednika Mozilla Firefox, a posredovani kao ReverseHTTP usluga. Jedinstveno sučelje ostvareno je kao JavaScript knjižnica.

Ostvareni sustav može se koristiti za posluživanje sredstava iz konteksta Web aplikacija. Ubrzanim razvojem Weba platforme, posluživanje sredstava iz konteksta Web aplikacije postaje sve smislenije. Uvođenjem tehnologija kao što su *Device* i *File API* proširuju se mogućnosti Web aplikacija, a ostvarenje pristupa tim novim sredstvima s Weba dodatno bi obogatio Web platformu.

Ostvareni sustav može naći primjenu u velikom broju postojećih Web aplikacija, a veću bi primjenu mogao naći u budućnosti zbog ubrzanog razvoja Web platforme.

Ključne riječi: Web, Web preglednici, JavaScript, HTML5, WebSocket, Web Hook, HTTP, Mozilla Firefox, ReverseHTTP

System for direct and proxied communication with Web browser processes using the HTTP protocol

Abstract

In this thesis Web architecture and possibilities of Web platform for rich application development are researched. Possibilities of implementating access to Web application context are considered, and potential benefits of such implementation are given. A unified interface for enabling direct and proxied access to user processes within a Web browser is implemented. Direct access is implemented as a Mozilla Firefox extension, and proxied as a ReverseHTTP service. The unified interface is implemented as a JavaScript library.

The implemented system can be used for serving resources from within the context of Web applications. With rapid progress of Web platform, serving resources from within Web application gains meaning. Introduction of technologies such as *Device* and *File API* expands possibilities of Web applications and implementation of access to these new resources from the Web would further enrich Web platform.

Implemented system can be used in large number of existing Web applications, but in the future, with rapid expansion of Web platform, this system could find even greater application.

Keywords: Web, Web browsers, JavaScript, HTML5, WebSocket, Web Hook, HTTP, Mozilla Firefox, ReverseHTTP