

Sadržaj

1. PHP: Hypertext preprocessor	1
2. Poslužiteljske tehnologije Weba	3
3. Sjednice i praćenje stanja aplikacije	9
4. Mrežno programiranje	14
5. Raspodijeljeni računalni sustavi	18
6. REST i Web API-ji	29
7. Klijentske tehnologije Weba.....	34
8. Sigurnost.....	41
9. Besplatnost, sloboda, otvorenost u računarstvu	47
KAO ZAKLJUČAK.....	50
PRIMJERI:.....	50

1. PHP: Hypertext preprocessor

Logika programiranja

- PHP datoteka = filter – prihvaća ulazne podatke i generira rezultat
- Primjereno Webu- ne postoji trajna interakcija s korisnikom i ne postoji implicitno očuvanje stanja aplikacije(aplikacija ≠ proces)
- Aplikacija razdvojena na niz cjelina- sve podatke potrebno eksplicitno prenijeti od jedne cjeline do druge

Skriptni jezik opće namjene, otvorenog koda, pogodan za razvoj Weba i dostupan na svim platformama, jednostavan za učenje -> jednostavan jezik za brzo stvaranje dinamičkih stranica Weba

Sintaksa slična C-u, vliki broj funkcija, podržan kod ISP-ova (Internet Service Provider). (PHP 5 - aktualna verzija)

Pokretanje PHP-a:

- Izvodi se na poslužitelju
- Više načina rada:
 - o Kao dio poslužitelja Weba(modul)
 - o Kao CGI poveznik kojeg poslužitelj Weba poziva
 - o Iz naredbene linije CLI(Command Line Interface)
- U osnovi radi kao filter- preuzima datoteku s programom i generira podatke kao rezultat rada
- Namijenjen prvenstveno dinamičkim stranicama Weba
- Moguća izrada klasičnih aplikacija

Osnovna svojstva:

- Programski kod- uključen u HTML ili XML; samostalan programski kod
- Blokovi PHP koda označeni oznakama:
 - o `<?php.....?>`
 - o `<script language="php">....</script>` -> ova dva načina su preporučena
 - o `<?.....?>`, `<?=.....?>`
 - o `<%.....%>`, `<%=.....%>` -> ovisno o postavkama
- Sve izvan oznaka se ignorira
- Komentari:
 - o Više redova- `/*....*/`
 - o Jedan redak- `//.....` ILI `#.....`
- Proceduralna i objektno orijentirana paradigma(funkcije i program; PHP5-proširena OO funkcionalnost)
- Tipovi podataka:
 - o Boolean, integer, float(double precision), string(8-bitni znakovi; UTF-8, ISO8859-2, Win1250)-podrška za Unicode znakove-> funkcije `mb_*(mb_strlen)-multibyte`
 - o Složeni tipovi: array, object
 - o Posebni tipovi: resource(pokazuju na vanjske resurse), NULL
- Imena varijabli- kao u C-u-> počinje sa slovom ili '_', slijedi proizvoljan broj slova, brojeva ili _
- Pristupanje varijabli- `$ime_var`; PHP interno određuje tip varijable prema kontekstu
- Skup operatora: sličan C-u(`!=`, `==`, `>=`, `<=`, `===`); `===` operator identiteta->true ako su varijable jednake vrijednosti i istog tipa

Što sve može- API:

- Skup ugrađenih funkcija se može proširiti
- Velik broj postojećih proširenja(extension)
- Podrška za: rad sa stringovima, datotekama, XML, baze podataka, Flash, PDF, LDAP, HTTP, FTP, SMTP, SOAP, SSL, SSH, ZIP, RAR, Zlib...
- PHP kao podloga za izgradnju:
 - o Sustava- stranica Weba, portala Weba, Web 2.0 sustava(Facebook), CMS sustava
 - o Alata- sustav predložaka(Smarty), framework biblioteka
- PHP kao podloga za dogradnju sustavima cacheiranja, optimizatorima

Prednosti i mane:

- Najčešće se izvodi kao modul poslužitelja Weba(nema pokretanja dodatnog procesa poveznika, manje zauzeće resursa, jednostavniji život administratora)
- Koristi mehanizme prijenosa podataka ugrađene u HTTP
- Koristi mehanizme opisane CGI-jem(poslužitelj postavlja varijable okoline kao za CGI, prijenos podataka između poslužitelja i modula nevidljiv korisniku)
- U slučaju potrebe moguć rad i kao CGI poveznik
- PHP prvenstveno namijenjen izradi dinamičkih stranica:
 - o automatsko parsiranje upita i podataka dobivenih HTTP-om u varijable PHP-a
 - o moguće miješanje PHP i HTML koda
 - o automatsko generiranje rezultata prema HTTP normi

- globalne varijable definirane unaprijed(\$_ENV-varijable okruženja; \$_SERVER- podaci o poslužitelju; \$_REQUEST, \$_GET, \$_POST- podaci iz zahtjeva)

U odnosu na druge programske jezike:

Prednosti	Mane
Jednostavnost	Sporost(kod se ne prevodi nego tumači u 2 faze)
Popularnost	Nedosljednost API-ja
Mogućnost proširenja	Nedostatci jezika(nekonzistentna podrška za Unicode, nepotpuni OO model..)
Fokusiran na Web	

2. Poslužiteljske tehnologije Web-a

Procesni modeli poslužitelja Web-a:

1. Jednoprocesni poslužitelj

- Poslužitelj se sastoji od samo jednog trajno aktivnog procesa(unutar procesa aktivna jedna dretva)
- Istovremeno može imati aktivnu samo jednu vezu s klijentom(preglednikom)
- Ostalih N zahtjeva za vezom čekaju u redu FIFO
- Prekobrojni zahtjevi se automatski odbacuju
- Nedostaci:
 - o Greška u implementaciji može terminirati proces, a samim time i cijeli poslužitelj
 - o Vrlo slaba skalabilnost s obzirom na broj zahtjeva
- Jednostavna izvedba, mali zahtjevi na memoriju i procesnu snagu računala poslužitelja

2. Pokretanje procesa na zahtjev

- *Proces upravljač* zaprima zahtjeve za vezom od strane preglednika
- Po zaprimanju zahtjeva proces upravljač:
 - o Stvara novi proces-*proces poslužitelj*
 - o Prosljeđuje vezu s preglednikom novostvorenom procesu
 - o Nastavlja čekati na nove zahtjeve za vezu
- Proces poslužitelj:
 - o Preuzima vezu s preglednikom
 - o Prihvaća zahtjev za sadržajem
 - o Vraća sadržaj pregledniku
 - o Zatvara vezu ili čeka na nove zahtjeve(jednokratna/trajna veza)
 - o Nakon zatvaranja veze proces se terminira
- Nedostaci:
 - o Mora postojati ograničenje na broj istovremeno aktivnih procesa
 - o Stvaranje novog procesa vrlo je zahtjevno i „skupo“
 - o Skalabilnost s obzirom na naglo povećanje zahtjeva je slaba
- Pokretanje novog procesa omogućuje opsluživanje više preglednika istovremeno
- Proces upravljač uvijek je spreman na obradu novog zahtjeva za vezom

- Eventualna greška terminira samo proces unutar kojeg se obrada izvodi, ne i čitavi poslužitelj

3. Pokretanje dretve na zahtjev

- Samo jedan *proces poslužitelj*, unutar poslužitelja više *dretvi*
- Dretva upravljač:
 - o Zaprima zahtjeve za vezom
 - o Stvara nove *dretve radnike*
 - o Prosljeđuje vezu novoj dretvi
 - o Nastavlja zaprimanje zahtjeva za vezom
- Dretva radnik:
 - o Zaprima zahtjeve od preglednika
 - o Vraća sadržaj pregledniku
 - o Zatvara vezu ili nastavlja komunikaciju(s obzirom na vrstu veze)
 - o Terminiranje dretve
- Stvaranje nove dretve zahtjevno, ali manje nego stvaranje novog procesa
- Nedostaci:
 - o Broj istovremeno aktivnih dretvi mora biti ograničen
 - o KomPLICIRANJA implementacija poslužitelja, opreznost s dodatnim modulima(„Thread safe“)
 - o Nema izolacije obrade zahtjeva- pogreška terminira čitav poslužitelj

4. Bazen procesa

- Na početku rada poslužitelja stvara se veći broj procesa poslužitelja(eng. process pool)
- Procesi u bazenu su u neaktivnom stanju
- Proces upravljač;
 - o Prihvaća vezu
 - o Aktivira jedan od procesa poslužitelja
 - o Prosljeđuje mu vezu s preglednikom
- Nakon zatvaranja veze s preglednikom proces se vraća u neaktivno stanje
- Procesi već postoje- nema naglog zahtjeva za zauzećem resursa
- Procesi obrađuju konačan broj procesa, nakon toga se terminiraju
 - o Dio mehanizma upravlja veličinom bazena procesa
 - o Sprječavanje moguće zauzeće memorije uzrokovano greškom u implementaciji
- Upravljanjem brojem procesa:
 - o Inicijalni broj pokrenutih procesa
 - o Minimalan i maksimalan broj neaktivnih procesa u bazenu
 - o Maksimalan broj procesa poslužitelja u sustavu
 - o Politika stvaranja i uništavanja procesa(kojom brzinom, koji procesi?)
- Bazen procesa omogućava:
 - o Brzo prihvaćanje veze s preglednikom aktiviranjem postojećeg procesa
 - o Brzu reakciju sustava na povećanje frekvencije zahtjeva za vezom
 - o Postupno oslobađanje resursa računala s obzirom na smanjenje broja zahtjeva

5. Bazen dretvi

- Slično kao bazen procesa
- Upravljačka dretva:
 - o Zaprima zahtjev za vezom
 - o Prosljeđuje dretvi razniku u bazenu dretvi

- Dretva radnik:
 - o Obrađuje zahtjeve preglednika
 - o Zatvara vezu i vraća se u neaktivno stanje
- Broj dretvi u bazenu varijabilan ili konstantan, ne utječe na količinu zauzete memorije

6. Dva procesa i bazen dretvi

- *Proces upravljač:*
 - o Prihvaća zahtjeve za vezom
 - o Prihvaćene veze proslijeđuje dretvama u procesu poslužitelju
- *Proces poslužitelj:*
 - o Sadrži bazen dretvi
 - o Prihvaća veze, obrađuje ih unutar dretve, vraća rezultate
 - o zatvara vezu, vraća dretvu u neaktivno stanje
- Odvajanje procesa upravljača od dretvi radnika zbog otpornosti sustava na greške- kritična greška terminirat će proces i sve sadržane dretve. Upravljač može ponovno pokrenuti proces poslužitelj

7. Hibridni model

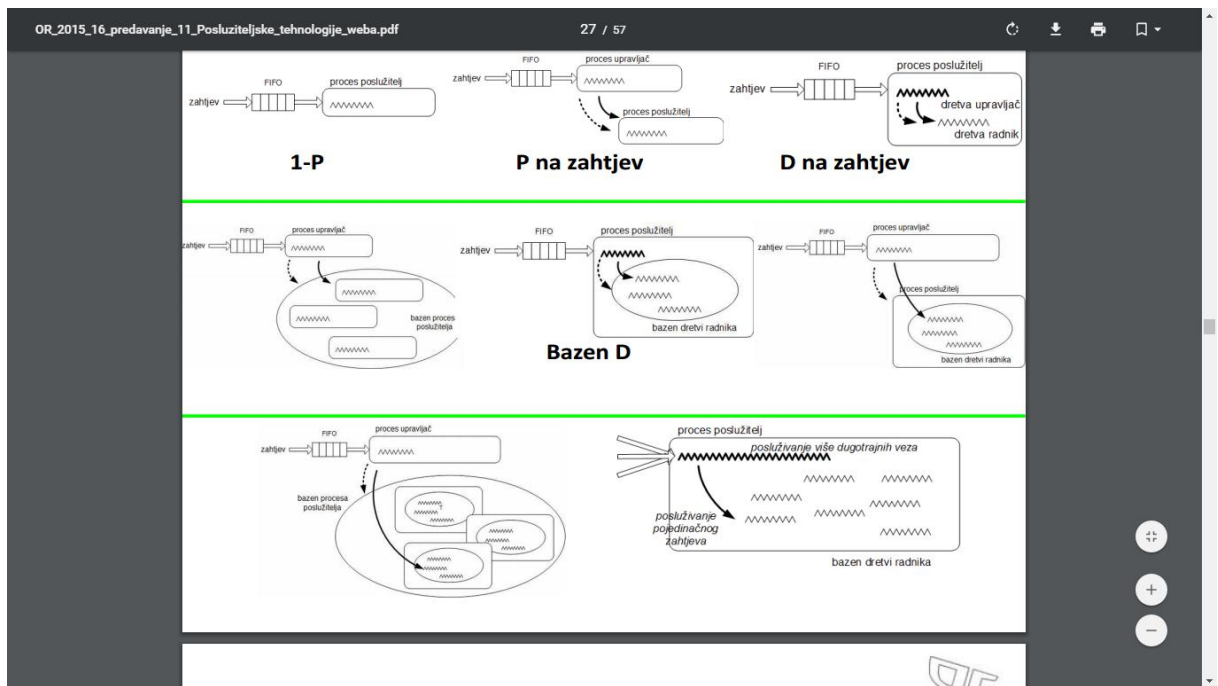
- Proces upravljač i bazen procesa poslužitelja
- Broj dretvi unutar pojedinog procesa konstantan i isti za sve procese u bazenu
- Proces upravljač drži održava broj neaktivnih dretvi između min. i max. Dopuštenog -> upravljanje veličinom globalnog bazena dretvi mijenjanjem broja procesa poslužitelja
- Procesi imaju ograničenje maksimalno dopuštenog broja posluženih veza nakon kojeg se terminiraju(nakon što sve dretve u procesu prijeđu u neaktivno stanje)
- Hibridni model je:
 - o Kompromis između potrošnje resursa i robusnosti sustava
 - o Kod velikog broja zahtjeva ima bolje karakteristike od čisto procesnog modela

Problem posluživanja trajnih veza

- Svaka trajna veza veže jedan proces ili dretvu za vrijeme trajanja veze
- Dugotrajne veze povećavaju efikasnost prijenosa(HTTP 1.1)->stajalište klijenta
- Efikasnost korištenja dugotrajne veze je mala->stajalište poslužitelja
- Problem nastaje kada preglednici s dugotrajnim vezama niske iskoristivosti zauzimaju vezu preglednicima koji čekaju

Model asinkronog posluživanja:

- Unutar procesa poslužitelja skupovi dretvi posebne namjene i dretvi radnika
- Dretve posebne namjene održavaju trajne veze s više preglednika istovremeno
- Dolaskom pojedinog zahtjeva preko trajne veze, zahtjev se proslijeđuje dretvi radniku na obradu
- Dretva radnik obrađuje zahtjev i vraća se u neaktivno stanje, **ne raskidajući vezu**



Poslužitelji Web-a podržavaju više procesnih modela- primjer Apache

- Multi-processing modules(MPM):
 - o MPM prefork(bazen procesa bez dretvi)
 - o MPM worker(hibridni višedretveni višeprocesni poslužitelj)
 - o **MPM event**(asinkrono posluživanje)->ova 3 su UNIX/LINUX
 - o MPM winnt(dva procesa i bazen dretvi)->MS Windows

Proširenja funkcionalnosti poslužitelja:

- Vanjski izvršni programi proširuju funkcionalnost poslužitelja Web-a
 - o Komunikacija između procesa poslužitelja i programa nekim od mehanizama međuprocenjske komunikacije(IPC)
 - o Sučelje između PP¹ i programa mora biti dobro definirano
 - o Poslužitelj može, ali i ne mora imati kontrolu nad vanjskim programom
 - o Komunikacija **zahtjev-odgovor**
 - o Prenosivost ovisi o jeziku implementacije: prevedeni programi teško prenosivi, interpretirani programi/skripte u ovisnosti o raspoloživosti interpretera na platformi
- Model CGI:
 - o Proces poslužitelj pokreće dodatni proces vanjskog programa
 - o Proces se pokreće za svaki pojedini zahtjev
 - o Nakon obrade zahtjeva proces se terminira
 - o Vrlo neučinkovit model i loša skalabilnost s obzirom na broj zahtjeva
 - o Vanjski program je izoliran od poslužitelja

¹ Proes poslužitelj

- Veze prema drugim resursima moraju se uspostavljati za svako pokretanje vanjskog programa(npr. veza prema bazi podataka)->vrlo neučinkovito
- Ne postoji način upravljanja brojem pokrenutih instanci vanjskog programa
- U slučaju korištenja poslužitelja Weba temeljenog na dretvama dodatni problem zbog stvaranja kopija procesa(fork())
- Rješenje:
 - Korištenje vanjskog poslužitelja za stvaranje procesa vanjskog programa, indirektna komunikacija procesa poslužitelja s vanjskim programom uz posredovanje *CGI'd poslužitelja* (CGI daemon)->ostaje problem jednokratnog korištenja procesa vanjskog programa
 - Proces poslužitelj i dalje nema kontrolu nad životnim ciklusom vanjskog programa, no *FastCGI poslužitelj* ima->pokreće i zaustavlja rad vanjskog programa
 - Proces poslužitelj putem FastCGI poslužitelja traži uslugu vanjskog programa, proslijeđuje podatke o okolini i sadržaj zahtjeva
 - FastCGI poslužitelj i proces poslužitelj mogu biti na istom ili različitom računalu
 - Ako vanjski program nije pokrenut FastCGI ga pokreće te mu proslijeđuje zahtjev
 - Nakon obrade zahtjeva i vraćanja sadržaja, vanjski program NE prestaje s radom, već neaktivan čeka novi zahtjev od strane FastCGI poslužitelja

Skriptni jezici na poslužitelju

- interpretiran programski kod kao proširenje funkcionalnosti poslužitelja Weba
- Slabije performanse no vanjski izvršni programi
- Brži razvoj i lakše održavanje sustava manje složenosti
- Vrste skriptnih jezika po orijentiranosti na Web:
 - Jezici opće namjene(Perl, Python, Ruby...)
 - Jezici prilagođeni za stvaranje dinamičkih stranica Weba(PHP...)
 - Specijalizirani jezici i okoline za Web(ASP, ColdFusion, JSP...)

Samostalni interpreteri

- Pokreću se kao vanjski programi korištenjem CGI ili srodnog mehanizma
- Iste prednosti i nedostaci kao vanjski izvršni programi
-

Ugrađeni(in-process) interpreteri

- Interpreter ugrađen u izvršni kod poslužitelja
 - Statički(preveden s ostatkom moda poslužitelja)
 - Dinamički(na zahtjev učitani u proces poslužitelja)
- Bolje performanse od samostalnih interpretera
- Obrada nije odvojena od ostalih zahtjeva u slučaju višedretvenog poslužitelja

Odvajanje logike i prezentacije

Logika je zadužena za dohvat podataka i obradu, a prezentacija za oblikovanje izgleda podataka na strani korisnika.

Razlozi za odvajanje logike i prezentacije:

- Odvajanje različitih tipova poslova i dodjeljivanje različitim profilima stručnjaka u razvojnom timu
- Jednostavnija promjena prezentacije, bez zadiranja u logiku
- Jednostavan razvoj više prezentacija istih podataka

Načini odvajanja logike i prezentacije:

- korištenje mehanizma predložaka
- potiskivanje aplikacijske logike u niže slojeve aplikacije

Predlošci

- Sadrže statički dio stranica i posebne oznake
- Prilikom zahtjeva za stranicom koja je temeljena na predlošku, poslovna logika dohvaća i obrađuje podatke, te rezultatima obrade zamjenjuje oznake unutar predložaka
- Tako stvorena stranica vraća se pregledniku kao rezultat upita
- Npr. Smarty predlošci za PHP

Modularnost poslužitelja Weba

- Jezgra(poput jezgre OS-a) i standardni skup modula(definira ponašanje poslužitelja za standardne zahtjeve preglednika)
- Dodatni moduli- podešavanje rada poslužitelja ili dodavanje specifične funkcionalnosti
 - o Mogu biti uključeni unutar poslužitelja(statički- uvijek prisutni; dinamički- učitavaju se i deaktiviraju po potrebi)
 - o Mogu definirati filtre i generatore sadržaja
- Generički model obrade zahtjeva unutar poslužitelja:
 - o Preglednikov zahtjev prolazi niz ulaznih filtara(npr. autorizacija pristupa ili dekompresija sadržaja)
 - o Određuje se relevantan generator sadržaja
 - o Generira se sadržaj odgovora
 - o Odgovor prolazi niz izlaznih filtara(npr. šifriranje i kompresija sadržaja)
- Implementacija modula je u potpunosti ovisna o ciljanom poslužitelju
 - o Mora koristiti odgovarajući API poslužitelja Weba
 - o Mora koristiti programski jezik platforme u kojem je poslužitelj razvijen
 - o Vezan je za računalne platforme na koje je poslužitelj prenesen
- Performanse modula su dobre jer predstavlja sastavni dio poslužitelja
- Nema izolacije prema poslužitelju->greška terminira proces, moguće i poslužitelj
- Primjeri:
 - o Apache- sučelje SAPI
 - o Microsoft IIS- sučelje ISAPI

Aplikacijski poslužitelji

- Poslužitelji Weba optimirani za posluživanje statičkog sadržaja
- Aplikacijski poslužitelji namjenjeni pružanju radne okoline razvijenim proširenjima-programskim komponentama
 - o Na poslužitelj se postavljaju programske komponente koje implementiraju funkcionalnost aplikacije ili široko primjenjive funkcionalnosti iskoristive u više aplikacija->funkcionalnost poslužitelja + postojeće komponente + nove aplikacijski specifične komponente
- Aplikacijski poslužitelj transparentno upravlja procesnim modelom, pristupom resursima...
- Komunikacijski protokoli: najčešće korišteni protokol HTTP
- Aplikacijski poslužitelji:
 - o sadrže poslovnu logiku aplikacije ili grupe aplikacija
 - o Isporučuju podatke poslužiteljima Weba
 - o Isporučuju podatke i pružaju usluge samostalnim aplikacijama
 - o Oslanjaju se na druge sustave za ostvarenje potpune funkcionalnosti aplikacija
- Poslužitelji Weba:
 - o Sadrže prezentacijsku logiku aplikacije Weba
 - o Isporučuju statične sadržaje
 - o Stvaraju prikaz podataka isporučenih od aplikacijskih poslužitelja, temeljen na HTML-u, i prosljeđuju ga klijentima
- Izvršno okruženje aplikacijskih poslužitelja:
 - o Skup podržanih usluga ovisan o složenosti aplikacijskog poslužitelja(npr. podrška povezivanju bazama podataka, podrška raspodijeljenim transakcijama, sigurnost, upravljanje resursima, skalabilnost, robusnost, asinkrona komunikacija korištenjem redova poruka...)
- Neki poznatiji ap. Posl.: Java EE(WebSphere, Jboss, Tomcat), PHP(appserver.io), .NET(.NET framework, Mono)

3. Sjednice i praćenje stanja aplikacije

Sustavi i pamćenje stanja: procesi s pamćenjem stanja, procesi bez pamćenja stanja

Proces posluživanja stranica/resursa jednostavan je proces bez pamćenja stanja(resurs1=GET(URI1))

HTTP-komunikacijski protokol bez stanja(stateless), transakcije dohvata međusobno neovisne

Upravljanje stanjem aplikacije:

- Stanje aplikacije na razini čitave aplikacije
- Stanje aplikacije na razini pojedinog korisnika aplikacije
- Stanje aplikacije na razini pojedine sjednice

Mehanizmi praćenja stanja i označavanja sjednica

Sjednica(session)- slijed logički povezanih transakcija između klijenta i poslužitelja unutar konačnog vremenskog perioda

- Početak sjednice- određen npr. zahtjevom klijenta prema poslužitelju nakon duljeg vremenskog perioda neaktivnosti
- Trajanje sjednice- slijed logički povezanih transakcija između klijenta i poslužitelja
- Završetak sjednice- određen npr. prestankom rada klijenta

Identifikator sjednice(session token)

- Jednoznačno određuje sjednicu
- Pridijeljen svakoj transakciji koja pripada sjednici

Sjednice se u pravilu koriste za označavanje transakcija autoriziranog korisnika aplikacije Weba

Prenošenje stanja protokolom HTTP:

Mehanizmi prenošenja podataka o stanju:

- 1) Skrivena polja(hidden fields)
- 2) Prepisivanje URL-a(URL Rewriting)
- 3) Kolačići(cookies)

Mjesto pohrane podataka o stanju/sjednici: na strani klijenta ili na strani poslužitelja

Skrivena polja

Sadržaj HTML stranice uključuje podatak o stanju/sjednici->skriveno polje unutar obrasca.

Slanjem sadržaja obrasca šalje se i sadržaj skrivenog polja.

Prednosti:

- Neovisnost o pregledniku
- Ne mogu se onemogućiti na klijentu
- Jednostavnost korištenja, performanse

Nedostaci:

- Lako dostupni
- Prenose se kod svake transakcije, u oba smjera
- Zahtijevaju korištenje obrazaca
- Zahtijevaju korištenje dinamički generiranih HTML stranica
- Rade samo s tekstualnim podacima

Prepisivanje URL-a

Poveznice na stranici uključuju podatak o sjednici.

Prepisivanje URL-a je mehanizam automatizirane promjene URL-a dolaznog zahtjeva na poslužitelj Weba, unutar ulaznog niza filtara poslužitelja i mehanizam dodavanja informacija unutar URL-ova poveznica HTML stranice dostavljene pregledniku.

Podaci su parovi **ime-vrijednost**, npr. <http://www.fer.unizg.hr/predmet/or?sid=234a3f0cc7>

Prednosti:

- Potpuna neovisnost o klijentu
- Ne može se onemogućiti na klijentu
- Jednostavna implementacija

Nedostaci:

- Podaci se prenose unutar polja upita URI-ja
- Moguće koristiti vrlo ograničenu količinu podataka
- Podaci dio većine URI-ja na stranici
- Potrebna dodatna funkcionalnost kod implementacije(ekstrakcija podataka o sjednici iz polja URI-ja)
- Smanjena čitljivost URI-ja
- Moraju se koristiti dinamičke stranice

Kolačići

Mehanizam razmjene male količine slobodno definiranih podataka(<= 4kB) između klijenta i poslužitelja unutar svake transakcije protokola HTTP

Sadržaj kolačića je par **ime-vrijednost**

Meta-podaci: domena, put, rok valjanosti, ograničenje pristupa, ograničenje na sigurnost prosljeđivanja

Stvaranje/promjena kolačića:

- kolačići pohranjeni na strani klijenta(unutar preglednika Weba)
- poslužitelj definira sadržaj i svojstva kolačića te ih uključuje u zaglavlje odgovora protokola HTTP
- klijent prihvaća kolačić i pohranjuje ga u lokalnom spremištu te ga pri sljedećem zahtjevu šalje poslužitelju
- ako je kolačić istog imena već bio definiran prethodna definicija se zamjenjuje novom
- kolačić se može stvoriti/promijeniti lokalno na strani preglednika(pomoću JavaScript Koda)

- poslužitelj određuje:
 - o par ime-vrijednost(obavezno)
 - o domenu(ako nije definirana podrazumijeva se kvalificirano ime poslužitelja)
 - o put(ako nije definiran podrazumijeva se put URI-ja resursa oji se dohvaća)
 - o rok valjanosti(nije obavezan)
 - o ograničenje pristupa(nije obavezno)
 - o ograničenje na sigurnost prosljeđivanja(nije obavezno)
- Prosljeđivanje kolačića:
 - o klijent kod slanja svakog zahtjeva za resursom nekom poslužitelju pretražuje lokalno spremište kolačića
 - o unutar zahtjeva se prosljeđuju svi kolačići koji zadovoljavaju uvjete
 - o prosljeđivani kolačići koriste samo par ime=vrijednost, meta podaci o kolačiću se ne prosljeđuju poslužitelju
- trajnost kolačića:
 - o trajni- imaju definirano vrijeme isteka valjanosti nakon kojeg se brišu
 - o privremeni- nemaju definirano vrijeme isteka valjanosti nego se brišu nakon prestanka rada klijenta
 - o moguće „ručno“ brisanje kolačića od strane korisnika preglednika ili promjenom sadržaja kolačića(postavljanje roka valjanosti koji je već prošao)
- ograničenje pristupa:
 - o udaljeni pristup kolačiću- poslužitelji, mehanizam prosljeđivanja
 - o lokalni pristup- klijent, programski, JavaScript
 - o ako je postavljeno svojstvo ograničenja pristupa(HttpOnly) lokalni pristup nije moguć
 - o ako je postavljeno ograničenje pristupa na sigurne kanale kolačić neće moći biti poslan preko npr. HTTP-a
- sigurnost kolačića:
 - o informacije unutar kolačića vidljive pri prijenosu nesigurnim kanalom
 - o moguća krađa podataka iz kolačića prisluškivanjem
 - o slanje kolačića poslužitelju van definiranog dosega kolačića korištenjem JavaScripta(„cookie scripting“, „session cookie theft“, „session hijacking“)
 - o maliciozna promjena sadržaja kolačića na pregledniku(„cookie poisoning“)
 - o praćenje navika i ponašanja korisnika
 - o EU direktive glede korištenja kolačića(korisnik mora imati mogućnost odbijanja pohrane kolačića)

Sjednice nad protokolom HTTP

Ostvarenje sjednice- općeniti postupak ostvarenja sjednice:

- 1) korisnik se uspješno prijavljuje u sustav
- 2) poslužitelj stvara zapis u tablici aktivnih sjednica
 - a. dodjeljuje jedinstveni identifikator sjednice
 - b. povezuje sjednicu s poznatim korisnikom sustava
- 3) poslužitelj šalje klijentu privremeni kolačić s jedinstvenim identifikatorom sjednice
- 4) klijent unutar svakog zahtjeva za resursom prosljeđuje poslužitelju privremeni kolačić

- 5) poslužitelj kod svakog zahtjeva provjerava valjanost identifikatora u tablici aktivnih sjednica (tablica je zbog potrebne brzine pristupa najčešće smještena u radnoj memoriji poslužitelja)
- 6) sjednica prestaje bit aktivna i briše se iz tablice:
 - a. nakon isteka vremenskog perioda bez zahtjeva od strane klijenta
 - b. nakon eksplicitne odjave korisnika iz sustav

Pohrana stanja aplikacije

Pohrana stanja aplikacije na klijentu:

- podaci u potpunosti pohranjeni na klijentu
 - o smanjeni zahtjevi na resurse poslužitelja
 - o skalabilnost s obzirom na broj korisnika
 - o veća mogućnost gubitka podataka
- podaci se u cijelosti proslijeđuju poslužitelju kod svakog zahtjeva za resursom
 - o garantira se cjelovitost podataka
 - o povećava količinu prenošenih podataka
- primjeri korištenja: preferencije korisnika, trajni podaci o identitetu korisnika (sigurnosno nekritični)

Pohrana stanja aplikacije na poslužitelju:

- podaci u potpunosti pohranjeni na poslužitelju
 - o pohranjeni u trajnom spremištu na strani poslužitelja
 - o neovisnost korisnika o korištenom klijentu
 - o identifikacija korisnika oslanja se na aktivnu sjednicu (privremeni kolačić)

Hibridni pristup:

- identifikator korisnika pohranjen na klijentu (dugotrajni kolačić)
- ostali podaci pohranjeni u trajnom spremištu na strani poslužitelja
- ostali podaci se dohvaćaju na osnovu identifikatora korisnika
- podaci o stanju dostupni i bez aktivne sjednice
- svojstva:
 - o smanjena mogućnost gubitka podataka
 - o manja količina prenošenih podataka u transakcijama
 - o povećani zahtjevi na resurse poslužitelja za smještaj podataka

PHP i sjednice

- identifikator PHPSESSID
- spremanje identifikatora sjednice u datoteke
- podaci u globalnoj varijabli \$_SESSION
- podržano upravljanje sjednice kolačićima ili automatskim prepisivanjem URL-a

Višestruki poslužitelji

- sustavi s velikim brojem korisnika i posjeta(veći broj poslužitelja)
- stanje pohranjeno na klijentu(neovisno o korištenom poslužitelju)
- stanje pohranjeno na poslužitelju- u memoriji, dijeljenom datotečnom sustavu, dijeljenoj bazi podataka, sjednički afinitet(zahtjevi se usmjeravaju istom poslužitelju)

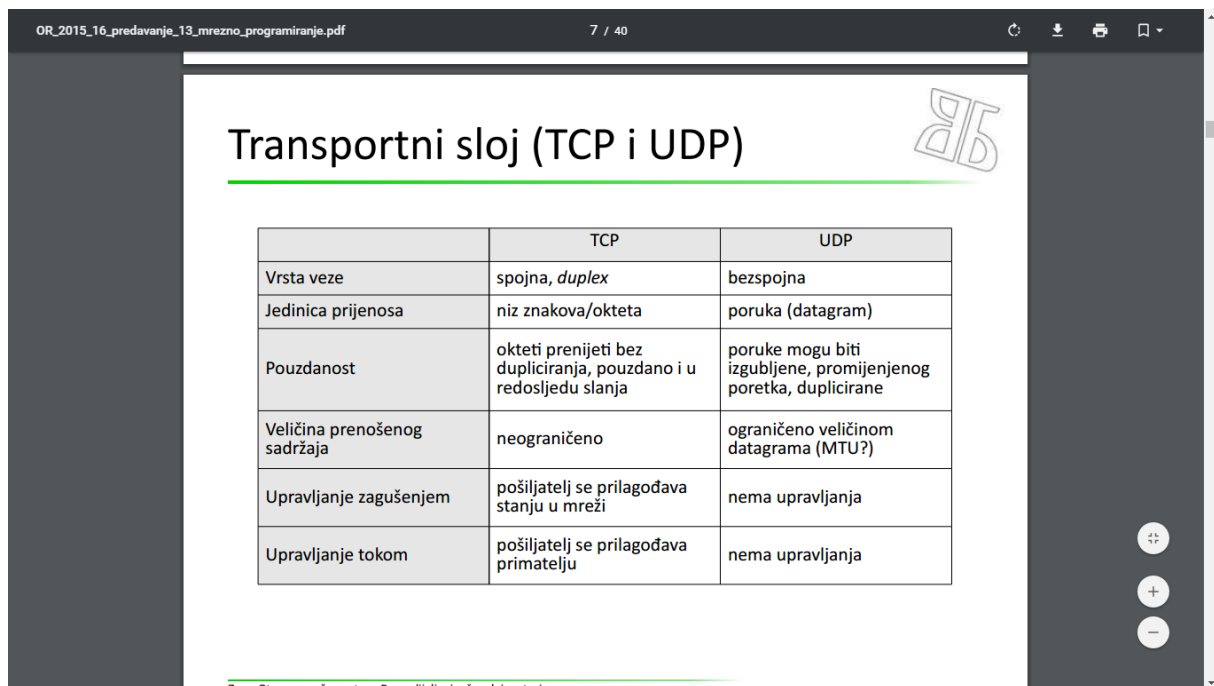
4. Mrežno programiranje

Za uspješno izvođenje komunikacija udaljenih procesa potrebno je:

- 1) locirati procese
- 2) ostvariti komunikacijski kanal
- 3) koristiti zajednički jezik komunikacije

TCP/IP stog protokola

- Sloj veze(podataka i fizički sloj), sloj interneta(IP), sloj transporta(TCP, UDP), sloj aplikacije(HTTP, FTP, SMTP, POP, RTP)



	TCP	UDP
Vrsta veze	spojna, <i>duplex</i>	bezspojna
Jedinica prijenosa	niz znakova/okteta	poruka (datagram)
Pouzdanost	okteti prenijeti bez dupliciranja, pouzdano i u redosljedu slanja	poruke mogu biti izgubljene, promijenjenog poretka, duplicirane
Veličina prenošenog sadržaja	neograničeno	ograničeno veličinom datagrama (MTU?)
Upravljanje zagušenjem	pošiljalac se prilagođava stanju u mreži	nema upravljanja
Upravljanje tokom	pošiljalac se prilagođava primatelju	nema upravljanja

Stvaranje TCP veze:

Razmjena 3 SYN/ACK paketa (SYN seq=x; SYN-ACK ack=x+1, seq=y; ACK ack=y+1, seq=x+1)

Proces učenja pošiljalca optimalnoj brzini slanja paketa. Opcionalni keep-alive paketi tijekom neaktivne veze. Nije poželjno učestalo stvaranje i raskidanje veze

Utičnice

- Utičnica(socket)- programska apstrakcija krajnje točke komunikacije
- Jedinstveno određena **IP adresom, portom i protokolom**
- Aplikacijski kod piše podatke u i čita podatke iz utičnice

Klijent:

- TCP: inicira vezu prema poslužitelju
- UDP: izravno šalje datagrame poslužitelju
- Mora prethodno poznavati adresu procesa poslužitelja

Poslužitelj:

- TCP: osluškuje zahtjeve za uspostavom veze
 - TCP: ostvaruje komunikacijski kanal s klijentom i razmjenjuje podatke dvosmjernim kanalom
 - UDP: zaprima datagrame i odgovara klijentu
- Standardne usluge imaju alocirane portove(npr. HTTP- port 80).
- Portovi <1024 dostupni procesima s posebnim ovlastima. Portovi >=1024 dostupni svim procesima

Programsko sučelje utičnica

- Općenit mehanizam međuprocenke komunikacije između procesa na istom ili različitim računalima
- Berkeley Sockets API: 4.2 BSD UNIX (1983)
 - o API- apstrakcija mrežnih priključnica, jezik C
 - o Licencirano od 1989.- AT&T
 - o De facto standard
 - o Ekvivalent API postoje za većinu programskih jezika

OR_2015_16_predavanje_13_mrežno_programiranje.pdf 16 / 40

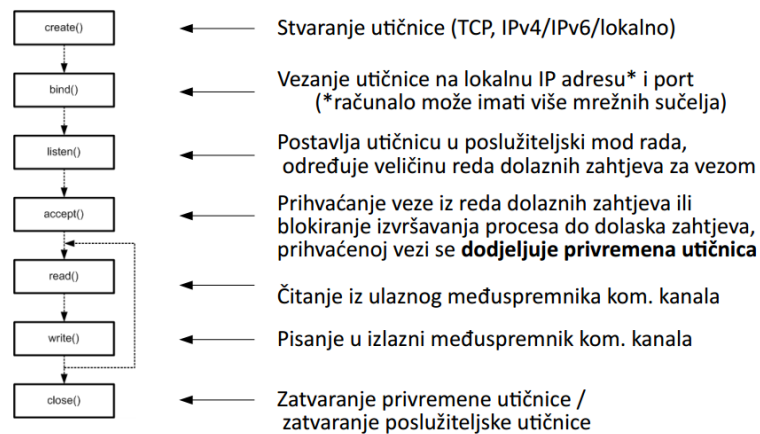
Klijent – spojna veza

```
graph TD; A[create()] --> B[connect()]; B --> C[write()]; C --> D[read()]; D --> E[close()];
```

- ← Stvaranje utičnice (TCP, IPv4/IPv6/lokalno)
- ← Spajanje na poslužiteljevu utičnicu (adresa utičnice)
[utičnici se dodjeljuje privremeni port]
- ← Pisanje u izlazni međuspremnik kom. kanala
- ← Čitanje iz ulaznog međuspremnika kom. kanala
- ← Zatvaranje klijentske utičnice

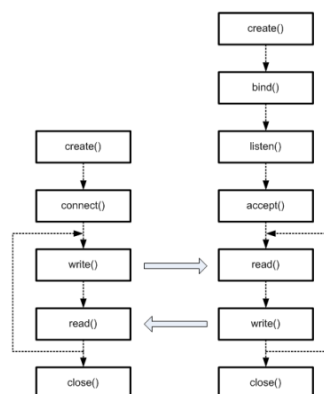
16 Otvoreno računarstvo - Raspodijeljeni računalni sustavi

Poslužitelj – spojna veza



17 Otvoreno računarstvo - Raspodijeljeni računalni sustavi

Interakcija klijenta i poslužitelja



18 Otvoreno računarstvo - Raspodijeljeni računalni sustavi

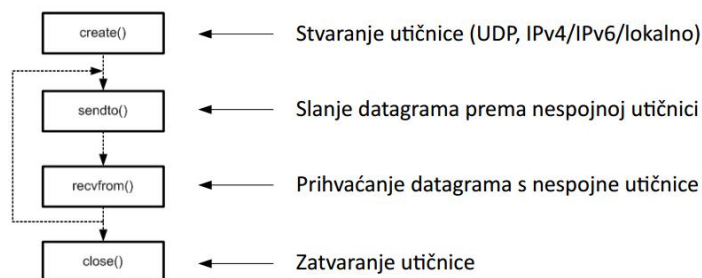
Procesni model na strani poslužitelja

- Određeno akcijom nakon zaprimanja nove veze
- Sinkroni model komunikacije
 - o Povezani procesi su sinkronizirani
 - o 1:1 odnos klijenta i izvršnog konteksta na poslužitelju
- Blokirajući i neblokirajući mod rada utičnica

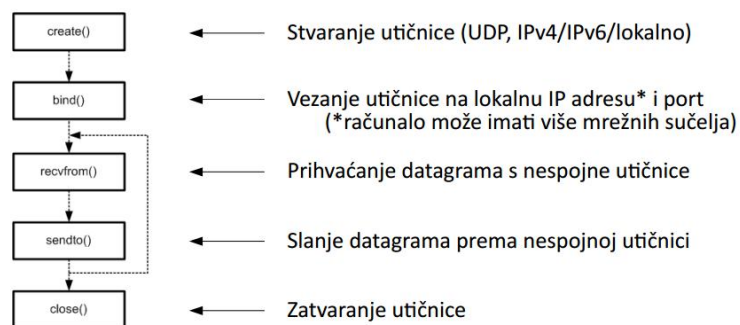
Čitanje podataka iz spojne veze

- Klijent šalje sadržaj u komunikacijski kanal
- Na strani poslužitelja naredba *recv()* čita sadržaj dolaznog međuspremnika

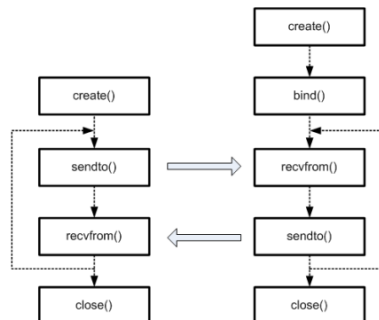
Klijent – bezspojna veza



Poslužitelj – bezspojna veza



Interakcija klijenta i poslužitelja



Procesni model na strani poslužitelja

- Nema trajnih veza s klijentskim procesima
 - Svaki prispjeli datagram je neovisan, može biti iz različitog procesa
 - Datagrami se mogu obrađivati u zasebnim dretvama

5. Raspodijeljeni računalni sustavi

"Sustavi u kojem programske i sklopovske komponente umreženih računala komuniciraju i međusobno usklađuju aktivnosti isključivo razmjenom poruka" (Coulouris i dr., 2012)

Osnovni izazovi: **paralelizam** izvođenja elemenata sustava(nužna koordinacija), **nepostojanje globalnog sata**(komunikacija isključivo porukama), **neovisnost grešaka**(mjesto greške, detekcija, oporavak)

Svojstva raspodijeljenih sustava:

- 1) Heterogenost komponenti
- 2) Otvorenost
- 3) Sigurnost
- 4) Skalabilnost
- 5) Pogreške u radu
- 6) Paralelizam
- 7) Transparentnost
- 8) Kvaliteta usluga

Heterogenost komponenti

Suradnja komponenti raspodijeljenog sustava unatoč heterogenosti na razinama komunikacijskih mreža, računalnog sklopovlja, operacijskih sustava, programskih jezika korištenih u izgradnji aplikacija, implementacija programskih komponenti sustava.

Npr. poredak zapisa okteta(Big/little endian, network byte order), zapis struktura podataka, kodiranje znakov, razlike u jezicima implementacije, raspoloživa sučelja za ostvarenje mrežne komunikacije.

Metode skrivanja heterogenosti: komunikacijski protokoli, middleware.

Postelov zakon(princip robusnosti):

„Budi konzervativan u onome što činiš(šalješ), budi liberalan u onome što prihvaćaš od drugih.“

Otvorenost raspodijeljenih sustava

Određena je mogućnošću **dodavanja novih** usluga i njihovim **jednostavnim korištenjem**.

Temelji:

- Normirani komunikacijski protokoli
- Javno objavljena ili normirana sučelja za pristup dijeljenih resursima

Otvoreni raspodijeljeni sustavi su građeni od heterogenih elemenata, ali svaki od tih elemenata mora biti sukladna kotišrenim normama.

Sigurnost

Povjerljivost(čuvanje resursa od neovlaštenog pristupa), Integritet(čuvanje od promjena ili uništavanja), Raspoloživost(čuvanje od zapreka pristupa resursima).

Skalabilnost

Učinkovitost sustava bez obzira na značajno povećanje posluživanih resursa i/ili broja posluživanja.

- Skalabilnost s obzirom na cijenu posluživanja(nadogradnja opreme zbog povećanja zahtjeva za resurse)->prihvatljivo povećanje cijene posluživanja resursa $\leq O(n)$
- Skalabilnost s obzirom na performanse(smanjenje performansi posluživanja uz povećanje posluživanih resursa i istu opremu, prihvatljivo $\leq O(\log(n))$)
- Ograničenost resursa(konačna)-npr Ipv4 adrese
- Izbjegavanje uvođenja uskih grla sustava- raspodijeljena rješenja umjesto centraliziranih

Pogreške u radu

Pogreške u raspodijeljenom sustavu su djelomične-dio komponenata nastavlja raditi

Najčešće vrste pogrešaka su u procesima(komponentama) sustava i u mrežnoj komunikaciji

Problematika pogrešaka u raspodijeljenom sustavu:

- Detekcija pogreške(mjesto, uzrok)
- Posljedice pogreške
- Toleriranje pogrešaka
- Oporavak od pogreške

Paralelizam

Komponente raspodijeljenog sustava izvršavaju se paralelno. Zahtjevi za resursom mogu prispjeti od više klijenata istovremeno->problem usklađivanja pristupa resursu

Transparentnost

Prikaz sustava kao cjeline umjesto kao skupa samostalnih komponenata.

- Transparentnost pristupa-jednak skup operacija za pristup lokalnim i udaljenim resursima
- Transparentnost lokacije- skrivena prava lokacija resursa(URL vs IP adresa)
 - o Transparentnost pristupa i lokacije zajedno čine mrežnu transparentnost
- Transparentnost paralelizma- očuvanje konzistencije dijeljenog resursa
- Transparentnost replikacije resursa
- Transparentnost pogrešaka- skrivanje postupka oporavka
- Transparentnost mobilnosti- promjena lokacije resursa ne utječe na način pristupa resursu
- Transparentnost performansi- prilagodba novom stanju(npr. broju korisnika) zbog očuvanja performansi sustava
- Transparentnost skalabilnosti- proširenje sustava moguće bez utjecaja na strukturu sustava, postojeće aplikacije i korištene algoritme

Arhitekture raspodijeljenih sustava

Struktura sustava s obzirom na gradivne komponente sustava i njihove odnose

Ključna pitanja:

- 1) Entiteti raspodijeljenog sustava
- 2) Komunikacijska paradigma
- 3) Uloge i odgovornost entiteta
- 4) Preslikavanje entiteta na fizičke komponente sustava

Entiteti raspodijeljenog sustava

Perspektiva sustava: procesi(niti), čvorovi

Perspektiva programa: objekti, komponente, web usluge

Komunikacijske paradigme

- Izravna komunikacija
 - o Jedan entitet korisnik i jedan entitet pružatelj usluge(odnos 1:1)
 - o Korisnik svjestan pružatelja, najčešće i pružatelj svjestan korisnika->**prostorna sprega(space coupling)**
 - o Oba entiteta aktivna tijekom komunikacije-> **vremenska sprega(time coupling)**
- Neizravna komunikacija
 - o Nije potrebna izravna sprega entiteta pružatelja usluge i korisnika
 - o Moguće korištenje trećeg, posredničkog, entiteta za razdvajanje korisnika i pružatelja usluge(odnos 1:n, m:1 ili n:m)

Mehanizmi izravne komunikacije

- Međuprocena komunikacija
 - o Komunikacija niske razine-primitive operacijskog sustava za slanje poruka, utičnice(Berkeley sockets API), višesmjerno odašiljanje(multicast)
- Udaljeni pozivi
 - o Zahtjev-odgovor(request-response/reply)
 - o Pozivi udaljenih procedura(Remote Procedure Call-RPC)
 - o Poziv udaljenih metoda(Remote Method Invocation-RMI)

Paradigma udaljenih poziva

Usluga upravlja skupom povezanih resursa, omogućava dostup resursima korisnicima i aplikacijama

- Usluga ispisa
- Usluga pristupa udaljenom datotečnom sustavu
- Usluga dohvata stranica Weba
- Aplikacijski specifična usluga

Pristup funkcionalnosti usluge putem sučelja(interface)- skup čvrsto definiranih operacija(read, write, get, put, post, delete, head...)

Udaljeni poziv je poziv operacije nad resursom->postoji vremenska i prostorna sprega klijenda i poslužitelja

Zahtjev-odgovor

Izveden „nad“ komunikacijskim kanalom za slanje poruka.

Jednostavan, učinkovit, ali primitivan mehanizam

Zahtjev sadrži pozivanu operaciju i argumente

Odgovor sadrži rezultate operacije nad resursom

Poziv udaljene procedure

Poslužitelj nudi sučelje prema resursima u obliku skupa procedura

Jezik za opis sučelja

- naziv, argumenti, povratna vrijednost
- Prevodi se u kod za stranu klijenda i poslužitelja

Sustav neovisan o platformi-kodiranje komunikacije: XDR, XML, JSON

Sakrivena složenost udaljene komunikacije i kodiranja podataka

Na strani klijenta:

- Klijent poziva procedure lokalno, unutar svog procesa
- Lociranje poslužitelja i sučelja
- Enkodiranje argumenata poziva
- Prosljeđivanje poziva poslužitelju
- Identificiranje rezultata obrade
- Dekodiranje povratne vrijednosti
- Vraćanje klijentu

Na strani poslužitelja:

- Zaprimanje zahtjeva za pozivom procedure
- Dekodiranje argumenata poziva
- Pozivanje implementacije procedure i primanje rezultata obrade
- Kodiranje povratne vrijednosti
- Slanje rezultata pozivatelju

Poziv udaljene metode

RPC u objektnom svijetu.

Opis sučelja koji implementira niz metoda- eksplicitni ili implicitni opisi; naziv metode, argumenti, povratna vrijednost

Iz opisa se stvaraju programski entiteti:

Strana klijenta: objektni zastupnici(proxy objects)

- Transparentnost za korisnika
 - Poziv se prosljeđuje metodi udaljenog objekta
 - Argument i povratna vrijednost može biti referenca udaljenog objekta
- Strana poslužitelja: skeleton
- „prazna“ implementacija metoda u udaljenom slučaju
 - Priprema parametara i prosljeđivanje pravom lokalnoj metodi
 - Vraćanje rezultata proxy objektu klijenta

Mehanizmi neizravne komunikacije

- 1) Grupna komunikacija
- 2) Objavi-pretplati
- 3) Redovi poruka
- 4) Raspodijeljena memorija
- 5) Prostori podataka

Grupna komunikacija

Apstrakcija grupne komunikacije- pošiljalatelj šalje poruku grupi primatelja, pošiljalatelju nisu poznati pojedini članovi grupe, svi aktivni članovi grupe primaju odaslanu poruku

Grupa primatelja:

- Apstrakcija grupe temelji se na jedinstvenom identifikatoru grupe
- Entitet se pridružuje grupi
- Član grupe može napustiti grupu

Sprege između pošiljalatelja i članova grupe: vremenska sprema postoji, prostorna (najčešće) NE postoji

Objavi-pretplati

Veći broj pošiljalatelja i primatelja

Sustav temeljen na događajima/porukama

Temelji se na posredničkom entitetu koji:

- Zaprima pretplate na događaje od zainteresiranih entiteta- postoje filtri pretplate
- Zaprima brisanja pretplata
- Prihvaća poruke objavljene od entiteta objavitelja i proslijeđuje ih svim pretplaćenim entitetima(ako je zadovoljen filter pretplate)

Sprema između pretplatnika i objavitelja: vremenska postoji, prostorna NE postoji

Jedinstveni protokol komunikacije objavitelja i pretplatitelja definiran protokolom prema posredničkom entitetu.

Redovi poruka

Posrednički entitet sadrži trajne redove poruka.

Entitet može biti vlasnik jednog ili više redova poruka

Entitet pošiljalatelj:

- Postavlja poruku u red poruka ciljnog entiteta(komunikacija 1:1)
- Posrednički entitet sigurno sprema poruku

Entitet vlasnik:

- Provjerava postojanje poruka u redu poruka
- Zaprima poruku iz reda poruka(poruka se briše iz reda poruka)

Entiteti koji komuniciraju ne moraju biti istovremeno aktivni->vremenska sprega NE postoji

Prostorna sprega postoji.

Jedinstveni protokol komunikacije entiteta definitan protokolom prema posredničkom entitetu.

Raspodijeljena dijeljena memorija

Temelji se na mehanizmu međuprocenske komunikacije->između procesa istog računala

Lokalne kopije dijeljene memorije na svakom računalu.

Propagacija promjena iz lokalne kopije u lokalne kopije svih ostalih računala->problem koherencije

Sprega između računala: prostorna NE postoji, vremenska postoji

Prostori podataka

Posrednički entitet pruža prostor(trajnu memoriju-touple space) koja služi čuvanju strukturiranih podataka(tuples)

Entiteti proizvođači- postavljaju podatke u prostor

Entiteti potrošači- čitaju(podaci ostaju u prostoru) ili vade podatke iz prostora

Sprega između proizvođača i potrošača: vremenska NE postoji, prostorna NE postoji (Dropbox?)

Uloge i odgovornosti entiteta

Klijent zahtijeva uslugu, a poslužitelj mora isporučiti uslugu- jedan entitet može imati obje uloge(u složenijem sustavu).

Svi entiteti su ravnopravni(peer-to-peer)-isti program, iste uloge; dio tereta preseljen s davatelja usluge na korisnike usluge.

Skaliranje resursa usluge sukladno trenutnom broju korisnika->veze između entiteta sukladno potrebama aplikacije

Raspodjela opterećenja po aktivnim članovima(mrežni promet, podaci, obrada) i redundancija podataka i funkcionalnosti.

Raspodjela entiteta na čvorove mreže

Logički entiteti mapiraju se na stvarne izvedbene resurse.

Višestruka računala poslužitelji-dijeljenje komponenti usluge na više računala poslužitelja; repliciranje jedinstvene usluge na više računala poslužitelja.

Pokretni programi:

- dobavljeni kod proširuje funkcionalnost klijenta
- lokalna ili udaljena interakcija
- problem sigurnosti
- java applet- dobavljanje izvršnog koda na stranu klijenta
- JavaScript- dobavljanje koda u preglednik Web

Pokretni agenti:

- Prenošnje i koda i stanja
- Samostalno kretanje između grupe računala i lokalno obavljanje zadataka
- Smanjenje korištenja mreže, ubrzavanje operacija

Modeli slanja/primanja podataka:

- Povlačenje podataka(pull)
 - o Zahtjev za podacima entitetu koji ih posjeduje
 - o Najčešće klijent zahtijeva podatke od poslužitelja
 - o Tipičan primjer- request-response protokoli(HTTP)
 - o Priroda veze- većinom povremena(trajanje dohvata)
- Guranje podataka(push)
 - o Guranje podataka od strane klijenta(npr slanje e-pošte, messenger-i...)- veza većinom privremena
 - o Guranje podataka od strane poslužitelja(FTP, HTTP push...)- veza je trajnija(problem kod velikog broja klijenata)
- Prozivanje(polling)
 - o Klijent periodički uspostavlja vezu s poslužiteljem i projerava dostupnost podataka
 - o Podaci se dohvaćaju povlačenjem
 - o Nema trajne veze kao kod „čistog“ guranja
 - o Značajno opterećenje mreže i poslužitelja kod velikog broja klijenata
 - o Primjeri: guranje e-pošte na klijenta(POP, IMAP), RSS feeds

Arhitekturni obrasci

Slojevit arhitektura- elementi slojevitog modela: slojevi(razine apstrakcije, rješavaju neovisne zadatke) i konektori(protokoli interakcije između susjednih slojeva)

Hijerarhijska organizacija slojeva- interakcija samo između susjednih slojeva, udaljeni slojevi „skriveni“

Prednosti:

- Sloj obavlja točno određenu ulogu
- Slojevi slabo povezani konektorima
- Neovisnost o implementaciji, slojevi jednostavno zamjenjivi
- Protokoli interakcije se moraju strogo poštivati

Nedostaci:

- Smanjena učinkovitost sustava
- Skupa promjena protokola interakcije
- Ponekad teško identificirati jasno odijeljene slojeve

Monolitna aplikacija

Jedan proces izvođen u okviru operacijskog sustava jednog računala

Slojevi čine jedinstven izvedbeni sloj aplikacije(logički (i fizički?) odvojene biblioteke funkcija)

Konektori između slojeva- skup funkcija vidljivih iz susjednog sloja->komunikacija pozivima funkcija susjednog sloja.

Višeprocesna aplikacija

Dva ili više procesa izvođenih na jednom ili više računala

Izvođenje na više računala- raspodijeljena aplikacija

Neki(ili svi) slojevi izolirani unutar zasebnih procesa

Konektori između slojeva-mehanizmi međuprocesne komunikacije->komunikacijski protokol

Slojevi aplikacije

Tipična arhitektura aplikacije sastoji se od tri sloja:

- Sloj prezentacije(GUI)
- Sloj aplikacijske logike
- Sloj podataka

Monolitna arhitektura

Svi funkcionalni slojevi unutar procesa izvođenog na jednom računalu

Dodatni alati potrebni za omogućavanje grupnog rada

Dvoslojna arhitektura

Funkcionirajući slojevi grupirani u dva zasebna arhitekturna sloja(2-tier), tj. procesa

Npr. klijentska+poslužiteljska aplikacija->sloj aplikacijske logike i sloj prezentacije+ sloj podataka

Može biti i drugačije!

Tanki i debeli klijent

Debeli klijent(fat client):

- Sadrži slojeve prezentacije i aplikacijske logike
- Zahtijeva veću snagu obrade računala domaćina i veću količinu podataka prenošenih mrežom

Tanki klijent(thin client):

- Sadrži samo sloj prezentacije
- Manja snaga obrade, manja količina prenošenih podataka

Troslojna arhitektura

Funkcionalni slojevi grupirani u tri zasebna arh. sloja(3-tier), tj. procesa

- Sloj prezentacije- klijentska aplikacija
- Sredni sloj- sloj aplikacijske logike
- Sloj podataka- baza podataka

Višeslojna arhitektura

Sadrži višestruke aplikacijske poslužitelje i/ili baze podataka

-ravnomjernija raspodjela opterećenja-> potrebna veća propusnost komunikacijske infrastrukture

Karakteristike arhitektura

Parametri procjene: snaga obrade računala, kapacitet spremišta podataka, propusnost komunikacijske infrastrukture, prilagodljivost, robusnost sustava, cijena izgradnje sustava, cijena održavanja sustava

Aplikacijski protokoli

Jezik sporazumijevanja komunicirajućih entiteta.

Binarni ili tekstualni format?

Binarni- kompaktniji, manje opterećenje mreže, manje opterećenje računala, nečitak za čovjeka

Tekstni- dulji, veće opterećenje mreže i računala, lakše praćenje i ispravljanje grešaka

Svaki od entiteta igra ulogu definiranu protokolom, a uloga određuje ponašanje tijekom konverzacije.

Mehanizmi protokola(RFC 3117- On the Design of Application Protocols):

- 1) Uokvirenje poruka(framing)
- 2) Kodiranje sadržaja(encoding)
- 3) Izvještavanje o stanju(reporting)
- 4) Asinkronost konverzacija(asynchrony)
- 5) Vjerodostojnost(authentication)
- 6) Zaštita podataka(privacy)

Uokvirenje poruka

- Jednostavna detekcija kraja jednorednih poruka->problem detekcije kraja duljih poruka
- Tri osnovne metode uokvirenja poruka: umetanjem okteta(octet stuffing), brojanjem okteta(octet counting), uništavanjem veze(connection blasting)

Uokvirenje umetanjem okteta:

- Npr. Poruka se terminira retkom u kojem se nalazi samo točka
- Primjer: prenošenje sadržaja e-pošte u SMTP
- Prednost- u trenutku početka prenošenja poruke pošiljatelju ne mora biti poznat njen čitav sadržaj
- Mana- sporo, dodatna obrada poruke i na pošiljatelju i na primatelju, nije pogodno za binarne podatke

Uokvirenje brojanjem okteta:

- Npr. dohvata e-pošte IMAP klijentom
- Prije početka slanja poruke pošiljatelj primatelju šalje duljinu poruke u oktetima
- Prednost- brzina, minimalna obrada kod slanja i primanja
- Mana- čitava poruka mora biti raspoloživa prije slanja

Uokvirenje uništavanjem veze:

- Stvaranje nove veze za prijenos jedne poruke
- Npr. korištenje u FTP protokolu
- Potrebno vrijeme za prenošenje podataka o parametrima nove veze(host, port), za otvaranje nove veze
- Pogodno za dulje(binarnе) datoteke
- Za manje datoteke neučinkovito

Kodiranje sadržaja poruke

Poruka se sastoji od zaglavlja i tijela(MIME)

Zaglavlje- jedan ili više redaka s atributima(opis prenošenih podataka)- prazan redak terminira zaglavlje

Tijelo poruke sadrži podatke- u sirovom obliku ili kodirane prije transporta, dekodirane nakon transporta

Reprezentacija stanja

Mehanizam prenošenja rezultata naredbe i stanja sustava na udaljenoj strani(većinom poslužitelju):

- Uspješno izvedene naredbe
- Trajne ili privremene greške
- Ostala stanja konverzacije

Brojke namijenjene programu, tekst čovjeku

Stanja usluge

- Usluge bez očuvanja stanja(stateless)
 - o Svaka akcija neovisna o prethodnim akcijama
 - o Jednostavne usluge poput request-response protokola
- Usluge s očuvanjem stanja(statefull)
 - o Rezultat akcije ovisi o prethodnim akcijama
 - o **Kontekst očuvanja stanja:**
 - Kontekst veze- npr. FTP->radno kazalo na udaljenom računalu
 - Kontekst klijenta- npr. stanje sandučića e-pošte
 - Globalni kontekst- npr. sadržaj tablice baze podataka

Asinkronost

Način obrade naredaba unutar jedne konverzacije:

- Slijedno-ne može se zaprimiti nova naredba dok izvođenje prethodne nije završeno
- Protočna struktura naredaba- poslužitelj prihvata naredbe i pohranjuje ih u FIFO strukturu, izvodi ih slijedno
- Paralelno izvršavanje- naredbe se prihvataju u FIFO i paralelno izvršavaju(u ovisnosti o broju raspoloživih niti)

6. REST i Web API-ji

REST ili **RE**presentational **S**tate **T**ransfer-Roy Fielding, 2000.g., doktorska disertacija

Stil programske arhitekture za izgradnju raspodijeljenih sustava

Sustavi koji prate REST principe-„RESTful“: otvoreni, skalabilni, nadogradivi, jednostavni

REST-principi

Svi resursi dijele uniformno sučelje za prijenos stanja između klijenata i resursa pomoću:

- Jedinstevne identifikacije resursa
- Upravljanje resursima pomoću reprezentacija
- Samo-opisnih poruka
- Poveznica na druge resurse unutar pojedinog resursa(HATEOAS-Hypermedia As The Engine Of Application State)

Protokol komunikacije:

- **Klijentsko-poslužiteljski**
- Interakcija je **bez pamćenja sustava**
- Mogući su posrednici- **slojevit sustav**

Resursi se mogu pohraniti u priručnu memoriju.

Kod-na-zahtjev->poslužitelji mogu slanjem koda na klijenta privremeno proširiti funkcionalnost klijenta

Web je sustav raspodijeljenog hipermedija-arhitekturne komponente: URI, HTTP, HTML, XML-ostale tehnologije Weba nasljeđuju osnovne komponente

Značajke Weba kao sustava:

- Otvorenost-otvoren prema novim tehnologijama
- Skalabilnost- proširiv, bez uskih grla
- Nadogradivost- evoluirati i nadograđuje se
- Jednostavnost- preživljava na osnovnim postavkama

Web je RESTful:

- Resursi se definiraju pomoću URI-ja, tj. pristupa se njihovim reprezentacijama. Moraju imati naziv(resursi) jer inače kao da ih i nema. Najčešći tip resursa je dokument.
- Stanja su predstavljena sadržajem koji se prenosi-protokol je bez pamćenja stanja

REST tehnologije

URI su najčešći odabir za **imenice**(resurse)

Metode HTTP-a su najčešći odabir za **glagole**(GET, POST, DELETE, PUT). HTTP je najuspješniji RESTful protokol.

Ugrađeni *caching* u protokol.

Autentikacija korištenjem HTTP autentikacijskih protokola

Sigurni prijenos podataka pomoću HTTPS(HTTP over TLS/SSL)

XML i JSON su najčešći odabir za tipove podataka. Zabranjeni kolačići(ugl.)->zbog prijenosa stanja.

Resurs je sve što je dovoljno važno da se može referencirati. Svaki resurs mora imati svoj URI(**adresirljivost**).

Reprezentacija resursa:

- Računalno čitljiv dokument koji sadrži informacije o resursu
- Može postojati više od jedne reprezentacije određenog resursa

HTTP zahtjev traži resurs.

HTTP odgovor vraća reprezentaciju resursa.

Mreža resursa- naglasak preseljen s poslužitelja na klijenta.

Stanje aplikacije- „na kojoj stranici/resursu se nalaziš?“

Stanje resursa- stanje na poslužitelju

Promjena stanja-automat->svaki HTTP odgovor u sebi sadrži prijedlog „sljedećih koraka“, tj. budućih HTTP zahtjeva(slično kao labirint)

Hipermedij

Informacije za upravljanje aplikacijom unutar prikaza informacija.

Zadaće hipermedijskih elemenata:

- Upućuje klijenta kako oblikovati HTTP zahtjev, koju metodu i URI koristiti, koje parametre poslati
- Nagovještaju oblik HTTP-odgovora, zaglavlja, podatke koji će biti vraćeni
- Predlažu kako će klijent koristiti odgovor u daljnjem procesu rada

HATEOAS je temelj RESTful API-ja

Tko su klijenti Web API-ja? Nisu samo preglednici Weba(najčešće skripte, agenti, mobiteli , tableti..)

4 razine zrelosti API-ja(Richardson's maturity model):

0) The Swamp of POX(Plain Old XML)

- Uporaba HTTP-zahtjeva kao tunela za poziv udaljenih procedura
- Metode GET i POST
- Upućivanje upita na jednu lokaciju
- Proizvoljno generiranje URI-ja
- Poruke zapakirane u proizvoljni XML ili JSON

1) Resursi

- Imenice
- Više mjesta kojima se upućuju zahtjevi
 - o Razdvajanje jednog velikog servisa u više malih
 - o Put URI-ja označava pojedini resurs

2) Glagoli- metode HTTP-a

- Metoda označava vrstu radnje koju želimo obaviti
- GET, POST, PUT, DELETE, (PATCH)
- Slične situacije obavljamo istim glagolom(npr. dodavanje komentara i dodavanje poruke)
- Pravilna uporaba HTTP kodova rezultata(status codes)

3) Hipermedijske kontrole(upravljački elementi)

- Konačno „pravi“ REST
- HATEOAS
 - o Poveznice koje se nalaze u dobivenom odgovoru pružaju upute o sljedećim koracima
 - o Klijent svojim odabirom mijenja stanje aplikacije
- Samo-opisne poruke, smanjuje se utjecaj dokumentacije

Trenutno stanje puno je bolje nego prije nekoliko godina zbog:

- Razumijevanja resursa i njihove reprezentacije
- Imenovanja resursa i URI-ja
- Pravilne uporabe pojedinih metoda HTTP-a

Metode HTTP-a

GET- dohvaćanje reprezentacije resursa(lokalizacija uadana u URI-ju)

POST- u užem smislu:stvaranje novog resursa

- u širem smislu: različite promjene, zapisivanje na poslužitelj..
- lokaciju određuje poslužitelj
- HTTP odgovori:201-created; 202- accepted

PUT- zamjenjuje stanje resursa novim resursom opisanim u danoj reprezentaciji

- može služiti i za dodavanje novog resursa
- stavlja se na lokaciju točno određenu URI-jem(ne određuje poslužitelj)
- HTTP odgovori: 200-OK; 204-No Content

PATCH- promjena dijela jednog podatka

- Dodatak HTTP-u->(još) nije službena metoda

DELETE- brisanje resursa na definiranoj lokaciji

Svojstva zahtjeva i metoda

Nullpotentni zahtjevi- sigurna metoda, ne mijenja stanje poslužitelja(npr. GET)

Idempotentni zahtjevi- višestruko slanje istog zahtjeva NE mijenja stanje (npr. DELETE, PUT)

Svaka sigurna metoda je i idempotentna(obrat ne vrijedi)

Formati za poruke API-ja

Internet media type(bivši MIME)

Podjela na razini zapisa podataka- JSON i XML. XML ima podršku za hipermedij no sve je manje prisutan(prevladava JSON)

JSON

Problem s običnim JSON-om jer:

- Nema hipermedijsku podršku
- Nema koncepta „veze“ ili nečeg sličnog
- URI je samo „obični“ niz znakova
- Implementirana semantika aplikacije ne može se ponovno koristiti za drugi API jer nema standarda, svaki novi API se parsira ispočetka

Dokumentacija API-ja

Zahtjev REST-a: self-describing messages(slično kao Content-Type)

Semantika protokola- odgovara na pitanje „kamo dalje?“, koje HTTP zahtjeve je moguće uputiti?

Semantika aplikacije- odgovara na pitanje „Što podaci znače?“

Primjer: odgovor Twitter API-ja

Content-Type:application/json

Nema uputa ni o semantici protokola ni o semantici aplikacije

Treba čitati API-dokumentaciju- profil->kolekcija poveznica i semantičkih oznaka(uz opis svake oznake)

Upravljanje verzijama API-ja

Problematično jer su API-ji napravljeni za klijente, a implementacija klijenata je mnogo

Kompatibilnost s prethodnim verzijama- ako nova verzija „skrši“ staru, stara treba ostati aktivna

Razdvajanje verzija->najčešće po URL-u pristupne točke:

- U imenu poslužitelja: „http://api-v1.mojsite.com“ „http://api-v2.mojsite.com“
- U putu do početne točke:“http://api.mojsite.com/v1/““ http://api.mojsite.com/v2/“

Prestanak rada- pristojno je dati obećanja

Primjeri koraka:

- 1. Korak- proglasiti verziju zastarjelom, ali još uvijek će raditi
- 2. Korak- obavijestiti da prestaje krpanje rupa
- 3. Korak- objaviti rok prestanka rada
- 4. Korak(možda)- dodatni period & isključiti API (HTTP code 410(Gone)+ objašnjenje+ novi link)

Kako HTTP pomaže RESTful API-jima?

Posluživanje više reprezentacija resursa, pregovaranje o sadržaju, caching, uporaba API-ja generira mnoge upite, zaglavlja u zahtjevu i odgovoru HTTP-a, autentifikacija

Prednosti RESTful API-ja:

- Fleksibilnost sustava(lakša promjena)
- Samo-dokumentiranje
- Korak u smjeru ponovne uporabe API-klijenata

7. Klijentske tehnologije Web-a

Razlika Web-a XX. i XXI. Stoljeća:

- Dynamic HTML(DHTML)
- Skup tehnologija za dinamičku promjenu stranica Web-a na strani klijenta((X)HTML-struktura, CSS-razmještaj i izgled, JavaScript-upravljanje elementima, DOM-model strukture)

DHTML služi za promjenu izgleda-svojstava elemenata stranice nakon početnog prikazivanja stranice. Sve se i dalje događa u pregledniku(nema komunikacije s poslužiteljem)

Povijest:

Na početku nije bila moguća nikakva promjena nakon prikazivanja stranice.

Početak DHTML-a- podrška preglednika IE4- nekompatibilnost među preglednicima, različite implementacije DOM-a, JavaScript

W3C norme- počinju ublažavati nekompatibilnost

Danas- znatno promijenjen u odnosu na početni DHTML. DOM scripting-pristup elementima putem modela DOM

Za promjenu stranice potrebno je moći: pristupiti elementima stranice, promijeniti svojstva elemenata, dodati/ukloniti elemente, reagirati na korisničke događaje u sučelju

Pristupanje elementima

Preglednik sadrži objektni model prikazivane stranice- HTML DOM.

Za točnu izgradnju i korištenje DOM-a potreban je valjani (X)HTML

Pronalaženje elemenata prema- elementima, atributima ili brojanjem(npr. 4. Odlomak)

JavaScript

Skriptni jezik ugrađen u preglednik. Sintaksa inspirirana jezicima C i Java.

Omogućuje dinamičke promjene elemenata stranice i reagiranje na događaje.

Izvodi se unutar preglednika Web-a-> sigurnosna ograničenja u radu(ne može pisati po disku, osim cookiesa)

Interpretira se, ne prevodi!

Implementacije: JavaScript, Jscript(Microsoft), ActionScript(Adobe)

Svojstva jezika

- JSON tipovi+ nekoliko dodatnih
- Varijable nemaju deklarirane tipove
- Upravljanje tokom programa- kao u C/PHP-> for, for.. in, do..while, while, if, else, switch
- Upravljanje greškama- try...catch, throws

Uključivanje vanjskih datoteka- spremanje skriptata u cache, mogu se uključiti i skripte na drugim domenama.

Korištenje DOM-a kroz JavaScript

The screenshot shows a presentation slide titled "Korištenje DOM-a kroz JavaScript". The slide is part of a PDF document named "OR_2015_16_Predavanje_16_Klijentske_tehnologije_Web-a.pdf", page 17 of 48. The slide content is as follows:

- Pristupanje elementima (W3C)
 - jednom elementu s definiranim atributom *id*
 - `document.getElementById()`
 - svim elementima s definiranim atributom *name*
 - `document.getElementsByName()`
 - svim elementima iste oznake
 - `document.getElementsByTagName()`
 - roditelju nekog elementa (XML DOM)
 - `document.getElementById().parentNode`
 - kolekcijama elemenata
 - `document.images[], document.forms[]`

At the bottom of the slide, there is a small footer: "17 Otvoreno računarstvo - Klijentske tehnologije Web-a".

Promjena svojstva elemenata (I)



- Dinamička promjena svojstva elemenata

- promjena izgleda

```
document.getElementById('naslov').style.color = 'red';
```

- promjena vidljivosti (izgleda)

```
document.getElementById('naslov').style.visibility = 'hidden';
```

```
document.getElementById('naslov').style.display = 'none';
```

- promjena sadržaja elementa (Text pod-element !)

```
document.getElementById('naslov').innerHTML = 'Novi naslov';
```

- promjena naziva gumba (atributa)

```
document.getElementById('gumb').value = 'Novi';
```

Dodavanje/uklanjanje čvorova



- Za rad s čvorovima koriste se metode JavaScripta za upravljanje modelom DOM, poput:

- **createElement** – stvaranje novog elementa
 - **createAttribute** – stvaranje novog atributa
 - **createTextNode** – stvaranje novog tekstualnog čvora

- **appendChild** – dodavanje elementa postojećem
 - **removeChild** – uklanjanje čvora djeteta
 - **replaceChild** – zamjena čvorova
 - **removeAttribute** – uklanjanje atributa



Primjer dodavanja elementa

- Dodavanje teksta na stranicu
 - Izravni upis – zastarjeli, neispravan način, ne ulazi u stablo
~~`document.write("<p>Novi tekst</p>")`~~
- Dodavanje novog elementa u DOM


```
var element = document.createElement("p");
var tekst = document.createTextNode("Novi tekst");
element.appendChild(tekst);
document.getElementsByTagName("body").item(0).
appendChild(element);
```
- DOM način je kompliciraniji, no omogućuje pozicioniranje elemenata i kasniju uporabu elementa

Reakcija na događaje

Većina funkcionalnosti vezana uz događaje(events).

Izvori događaja:

- HTML DOM elementi(korisnik)
- BOM(Browser Object Model) elementi(preglednik)
- Vremenski okidani događaji (JavaScript)

HTML DOM događaji se odnose na elemente- odlomci teksta, poveznice, gumbi, obrasci..

HTML DOM događaje okidaju: miš(onclick, onmouseover...), tipkovnica(onkeypress, onkeyup...), HTML objekti i obrasci(onload, onsubmit, onfocus...)

Registracija događaja

Porevizanje događaja na nekom elementu s naredbama/funkcijama koje treba izvesti:

- Inline(unutar HTML-a)->ne razdvaja JS od HTML-a- ``
- Uobičajeni(unutar JS koda)-> razdvaja logiku od prezentacije, za razliku od inline-
`element.onclick=promijeni;`
- W3C DOM level 2-> dodatne, napredne mogućnosti- više listenera za događaj-
`element.addEventListener('click', promijeni, false)`

Ako JS nije uključen:

- Graceful degradation- što više funkcionalnosti izvesti na drugi način
- Progressive enhancement- pisati osnovne funkcionalnosti kao da JS neće biti podržan
- U svakom slučaju obavijestiti korisnika o smanjenoj funkcionalnosti

AJAX

Interaktivnost stranice Weba s izvorima podataka- izravna komunikacija JS programa i vanjsih izvora informacija omogućila bi: izbjegavanje učitavanja čitave stranice, komunikaciju na zahtjev/događaj na stranici, ali i otvorila niz pitanja sigurnosti!

Asynchronous JavaScript XML- skup klijentskih tehnologija Weba

Komunikacija JS koda i poslužitelja- korištenje postojeće funkcionalnosti preglednika za komunikaciju protokolom HTTP(npr. asinkroni dohvat slika s učitavane HTML stranice)

Asinkroni način komunikacije:

- Nema čekanja na završetak komunikacije s poslužiteljem
- Očuvana interaktivnost stranice
- Složenija izvedba programa

Sinkroni način komunikacije:

- Čekanje na završetak komunikacije
- Gubitak interaktivnosti stranice
- Jednostavnija izvedba programa

Primjene AJAX-a:

- Provjera podataka u obrascima
- Automatsko nadopunjavanje teksta u polju
- Dohvat podskupa informacija s obzirom na dinamički definirane uvjete
- Osvježavanje podataka(u stvarnom vremenu)
- Dohvat dijelova mapa obzirom na lokaciju i rezoluciju mape
- Web 2.0 aplikacije

Kroaci u izvođenju AJAX zahtjeva

- 1) Događaj pokrene izvođenje funkcije rukovatelja događajem(event handler function)
- 2) Funkcija rukovatelj stvori XMLHttpRequest objekt i inicijalizira ga- postavi način rada, URL, metodu HTTP-a, parametre poziva, zaglavlja... i callback funkciju
- 3) Funkcija rukovatelj pokreće zahtjev(sinkroni ili asinkroni)
- 4) Na svaku promjenu stanja komunikacije, XMLHttpRequest objekt poziva callback funkciju i proslijeđuje trenutno stanje i rezultat komunikacije
- 5) Nakon detektiranog završetka komunikacije, callback funkcija obrađuje konačan rezultat komunikacije i mijenja DOM stranice

Stanja dohvata

XMLHttpRequest.readyState:

- 0: zahtjev nije inicijaliziran
- 1: ostvarena veza s poslužiteljem
- 2: zahtjev prihvaćen
- 3: zahtjev se obrađuje
- 4: zahtjev završio, odgovor je spreman (ili greška)

XMLHttpRequest.status:

- HTTP kod statusa zahtjeva

AJAX problemi

Mnogo koda pisanog u JavaScriptu- manja prilagodba preglednicima i potrebno testiranje

Integracija s preglednikom Web- neprirodna funkcionalnost gumba „back“ i „refresh“; problemi spremanja stranice u Bookmarks

Korisnik ne očekuje promjenu podataka na dijelu stranice niti čekanje na prijenos podataka- potrebna vizualna upozorenja („učitavam“)

Problemi ograničena zbog sigurnosti

Pristup izvorima podataka

Stranice Web- sadržaj, prezentacija i izvršni kod

Želimo interakciju stranica aktivnih na istome pregledniku->problem sigurnosti zbog politike istog izvora

Izvor URI-ja= <protokol, sjedište, port> npr. <http, www.fer.hr, 80>

Politika istog izvora:

- Interakcija stranica unutar preglednika samo između stranica istih izvora
- Mrežna komunikacija samo između stranice i izvorišnog sjedišta (sadržaj domene ima sva prava pristupa sadržajima unutar te domene; sadržaj van domene nema prava pristupa)

Politiku provodi preglednik Web

Politika istog izvora ne primjenjuje se na komunikaciju tijekom dohvata elemenata stranice

Metode pristupa poslužiteljima izvan domene: poslužitelji posrednici, JSONP, CORS, različiti dodaci...

Poslužitelj-posrednik->AJAX zahtjevi sa stranice (klijenta) upućuju se samo prema izvorišnom poslužitelju

Poslužitelj-posrednik->zaprima zahtjeve, prosljeđuje ih ciljnim poslužiteljima, zaprima odgovore i prosljeđuje ih AJAX klijentu

Dva modela interakcije AJAX klijenta i posrednika:

- Klijent upućuje zahtjev i čeka sadržaj od posrednika
- Klijent upućuje zahtjev i odspaja se. Klijent se naknadno spaja i provjerava dostupnost sadržaja te povlači sadržaj, ako je on spreman

JSON with padding-zaobilaženje politike istog izvora korištenjem `<script>` elementa za izvršavanje zahtjeva prema poslužitelju van domene dokumenta->JavaScript u HTML DOM stranice „injektira“ `<script>` element koji pokazuje na vanjski izvor->preglednik postavlja GET zahtjev za dohvat sadržaja skripte s naznačenog URL-a->poslužitelj vrati sadržaj tipa `application/javascript`->preglednik izvrši dohvaćeni JS kod

Callback JS funkcija-sadržaj dohvaćenog koda- poziv callback funkcije prethodno definirane unutar postojećeg JS koda stranice-argument funkcije je JSON objekt

Callback funkcija obrađuje objekt, a poslužitelj mora: zaprimiti argumente URL-a dohvata podataka i generirati ispravan JS kod- ime callback funkcije

Vrste odgovora poslužitelja

Po definiciji JSON- u stvari odgovor poslužitelja je JS kod

Sigurnost- potencijalni problem sigurnosti- odgovor nije omeđen samo na jedan poziv funkcije->isti mehanizam komunikacije mogu koristiti maliciozni programi(npr. prosljeđivanje ukradenih podataka van domene); dohvaćeni kod postaje dio iste domene kao i stranica->vrijede ograničenja s obzirom na politiku istog izvora

CORS

Cross-origin resource sharing je W3C recommendation

Mehanizam dogovaranja preglednika i poslužitelja o omogućavanju komunikacije stranice i poslužitelja van iste domene- dodatna HTTP zaglavlja(`Origin`-zahtjev i `Access-Control-Allow-Origin` – odgovor) i `cross-domain XMLHttpRequest`

Cross-site scripting ranivost- ideja je injektiranje koda unutar HTML-a na poslužitelju(kratkoročno ili dugoročno)->korisnik učitava kod kao dio dokumenta->kod iskorištava prava unutar domene

8. Sigurnost

Sigurnosni zahtjevi-ciljevi:

- Povjerljivost, tajnost(confidentiality, secrecy)
 - o Očuvanje tajnosti poruke- treba biti razumljiva samo pošiljatelju i primatelju
- Cjelovitost, očuvanost(integrity)
 - o Sadržaj poruke ne smije se mijenjati prilikom prijenosa(promjene treba moći primijetiti)
- Izvornost, ovjera(authenticity)
 - o Sposobnost određivanja izvornosti i mogućnost otkrivanja promjene sugovornika
- Neporicljivost(nonrepudiation)
 - o Nemogućnost poricanja slanja poslane poruke
- Dostupnost(availability)
 - o Osiguranje dostupnosti usluge aktivnim sprječavanjem napada
- Kontrola pristupa(access control)
 - o Sposobnost dodjele ili zabrane prava pristupa i korištenja resursa na pouzdan način

Osnovni pojmovi

Kriptologija je znanost koja se bavi izučavanjem i definiranjem metoda za zaštitu informacija (šifriranjem) i izučavanjem i pronalaženjem metoda za otkrivanje šifriranih informacija (dekriptiranjem).

Kriptografija- umijeće čuvanja tajnih informacija

Kriptoanaliza- umijeće otkrivanja tajnih informacija

Ključ(key)- informacija koja se koristi u postupku kriptiranja i/ili dekriptiranja i jednoznačno određuje postupak kriptiranja i/ili dekriptiranja

Šifra(cypher, cipher)- par algoritama koji se koriste za pretvorbu iz izvornog u kriptirani oblik i obratno. Katkad može imati i značenje ključa.

Kod(code)- zamjena jedinice izvornog teksta kodnom riječju. Bilo koja metoda skrivanja izvornog značenja.

Aktivni napadi

Lažno predstavljanje- korisnika(impersonation) ili usluge(phishing)

Ubacivanje u komunikaciju- man in the middle

Uskraćivanje usluge- denial of service (DOS/DDOS)

Napad lažnim porukama- ponavljanjem poruka(replay attack) zamjenom poruka(substitution attack)

Pasivni napadi

Prisluškivanje- eavesdropping, tapping

Pogađanje ključeva ili lozinki- napad grubom silom(brute force attack), napad rječnikom(dictionary attack), napad odabranim porukama(chosen cipher/plain-text), kriptanaliza, statističke metode

Metode zaštite

Zaštita na razini:

- Sustava- arhitektura mreže, vatrozid, antivirusna zaštita, sigurnosni alati
- Komunikacijskog kanala- secure socket layers(SSL), IPSEC, kriptiranje komunikacije
- Poruke- digitalni potpis, digitalna omotnica

Algoritmi

Tajni algoritmi- neprikladni za ozbiljnu primjenu

Javni algoritmi:

- Algoritmi sažetka(digest, hash)- digitalni otisak prsta
 - o Cjelovitost, (izvornost)
- Algoritmi s ključem- tajni ključ(secret key)- simetrični algoritmi(šifriranje blokova ili šifriranje toka); javni ključ(public key)- asimetrični algoritmi
 - o Povjerljivost, (cjelovitost), (izvornost)
- Steganografija- digitalni vođeni žig(watermarking)
 - o Povjerljivost

Algoritmi sažetka

Prevode sadržaj poruke u jedinstveni sažetak

Funkcija generiranja sažetka- jednosmjerna(gubitak informacija)

- Prevodi izvorni tekst u sažetak fiksne duljine(različiti izvorni tekstovi mogu imati iste sažetke)
- Nije moguće odrediti koje dvije poruke imaju isti sažetak
- Generirani sadržaj treba sličiti slučajno generiranim podacima
- Sažetak poruke odgovara digitalnom otisku prsta poruke
- Algoritmi: MD5, SHA-1, SHA-3(dolazi- lel...)

Algoritmi s tajnim ključem

Isti ključ za kriptiranje i dekriptiranje- simetrični(tajni ključ, dijeljeni ključ)

Sigurnost ovisi o ključu i mehanizmu dogovora ključa između sugovornika

Blokovske šifre- najčešće, ulaz u funkciju->blok podataka stalne duljine

Šifre toka- ulaz u funkciju->bit po bit iz toka podataka koji se šifrira

Gradivni blokovi- supstitucijske i permutacijske kutije

Brz rad algoritma za sklopovske implementacije

Simetrični algoritmi

Tajni(dijeljeni) ključ- problem sigurnog prenošenja poruke prevodimo u problem sigurnog prenošenja ključa->dogovor dviju strana o ključu putem sigurnog kanala

Algoritmi: AES(Rijndael), Serpent, Twofish, Blowfish, IDEA, 3DES, DES

Algoritmi s javnim ključem

Različiti ključevi za šifriranje i dešifriranje- asimetrični(tajni ključ i javni ključ)

Temeljeni na NP teškim matematičkim problemima.

Sigurnost ovisi o odabranom problemu, ključu(duljini) i zaštiti tajnog ključa

Kako izgraditi algoritam?

- Uzeti težak problem s posebnim slučajem koji se može riješiti u polinomnoj složenosti
- Šifriranje- pretvoriti poruku u poseban slučaj problema, zatim javnim ključem pretvoriti jednostavan problem u težak
- Dešifriranje- korištenjem privatnog ključa pretvoriti težak problem u jednostavan i riješiti ga
- Primjeri problema- faktorizacija brojeva, diskretni logaritmi, eliptičke krivulje

Simetrični algoritmi

Prednosti:

- Velika raznolikost algoritama
- Brzina

Mane:

- Distribucija ključeva (za N sudionika potrebno $n*(n-1)/2$ ključeva) i problem razmjene ključeva

Asimetrični algoritmi

Prednosti:

- Distribucija ključeva- javni ključ se može slobodno dijeliti

Mane:

- Sporost(velika složenost i složena implementacija)

Primjena algoritama

Šifriranje podataka(npr. na disku)- cjelovitost, tajnost

Digitalni potpis- ovjera izvornosti, cjelovitost, neporicljivost

Digitalna omotnica- ovjera izvornosti, cjelovitost, neporicljivost, tajnost

Ključevi

Što sve može biti ključ?

- Kratki- lozinke, OTP(one time password), token
- Dugi- sažetak, certifikat
- Biometrijski- otisak prsta ili dlana, uzorak šarenice, glas, DNA

Sigurnost ovisi o kvaliteti ključa.

Ključevi za simetrične kriptosustave-pseudoslučajni brojevi->važna kvaliteta generatora slučajnih brojeva

Ključevi za asimetrične sustave- posebna svojstva(npr. umnožak 2 velika prosta broja)

Pohrana ključeva- sigurnosne norme zahtijevaju pohranu unutar uređaja- pametne kartice, krypto uređaji. Ključ ne može i ne smije napustiti uređaj(pokušaj otvaranja uređaja rezultira uništenjem ključa)

Digitalni potpis

- Pošiljalatelj generira sažetak poruke S
- Pošiljalatelj šifrira sažetak ključem Pk(privatni ključ)
- Pošiljalatelj dodaje šifrirani sažetak na poruku
- Primaatelj javnim ključem Jk dešifrira sažetak(ovjera i neporicljivost)
- Primaatelj generira sažetak primljene poruke S'.
- Ako je S=S' primljena poruka je istovjetna originalu(očuvanost)

Digitalna omotnica

- Pošiljalatelj šifrira poruku simetričnim algoritmom ključem K
- Pošiljalatelj šifrira ključ K asimetričnim algoritmom(javnim ključem Jk)
- Primaatelj ključem P dešifrira ključ K
- Primaatelj ključem K dešifrira poruku(tajnost)

Zakon o električkom potpisu 2002. Osigurava zakonsku istovjetnost naprednog elektroničkog potpisa s ručnim potpisom, odnosno potpisom i pečatom. Razrađuje zakonske pod-akte za definiranje uloge države u procesu te definira tko može postati CA(Certificate Authority)

Elektronički potpis- skup podataka u elektroničkom obliku koji su pridruženi ili su logički povezani s drugim podacima u elektroničkom obliku i koji služe za identifikaciju potpisnika i vjerodostojnosti potpisanoga elektroničkog dokumenta. (Ovjera i očuvanost)

Napredan elektronički potpis- pouzdano jamči identitet potpisnika i:

- Povezan je isključivo s potpisnikom
- Nedvojbeno identificira potpisnika
- Nastaje korištenjem sredstava kojima potpisnik može samostalno upravljati i koja su isključivo pod nadzorom potpisnika
- Sadržava izravnu povezanost s podacima na koje se odnosi i to na način koji nedvojbeno omogućava uvid u bilo koju izmjenu izvornih podataka(neporicljivost, ovjera, očuvanost)

Potreba za središnjim autoritetom koji ovjerava naš potpis simetričnim ključem.

Komunikacija s tim autoritetom zaštićena je simetričnom kriptografijom

Autoitet označava vrijeme primitka poruka(zaštita od napada ponavljanjem poruka)

Središnji autoritet može čitati sve poruke i ovjerava svaku poruku. On čuva veliku količinu tajnih informacija.

Za asimetrični ključ nešto drugačija situacija:

Poruku potpisujemo našim tajnim ključem, a sugovornik provjerava potpis našim javnim ključem.

Nema potrebe za središnjim autoritetom koji provjerava svaku poruku-> kako znamo da je javni ključ sugovornika baš njegov?

Potvrda o valjanosti ključa=certifikat->središnji autoritet jamči ispravnost ključa

Certifikat je potvrda u elektroničkom obliku koja povezuje podatke za verificiranje elektroničkog potpisa s nekom osobom i potvrđuje identitet te osobe

Certifikat je potvrda o vezi između identiteta i javnog ključa. Javan je i sadrži: identifikaciju izdavatelja i subjekta, oznaku algoritma potpisa i javni ključ, razdoblje važenja, potpis.

CA- certificate authority- izdaje certifikate->pravna ili fizička osoba koja izdaje certifikate ili daje druge usluge povezane s elektroničkim potpisima

Moguće ostvarenje hijerarhije CA->lanac povjerenja, staza certificiranja(od korijenskog do našeg CA).

CA održava mehanizme provjere valjanosti certifikata- provjera potpisa i održavanje popisa povučenih certifikata(CRL- certificate revocation list).

Potpisnik koji je izgubio certifikat ili mu je on otuđen mora o tome odmah obavijestiti davatelja usluge certificiranja. Davatelj usluge provodi uvid u postupak opoziva certifikata i dalje postupa po utvrđenim pravilima opoziva izdatih certifikata.

CRL ne uključuje certifikate kojima je istekao rok valjanosti.

Provjera valjanosti certifikata- provjera digitalnih potpisa certifikata, provjera razdoblja valjanosti certifikata, provjera popisa povučenih certifikata

Kompromitirani CA unosi veliku štetu- cijela hijerarhija od tog CA na niže postaje nevažeća

Norme- Public Key Cryptography Standards(PKCS), X.509, FIPS(Federal Information Processing Standards), **W3C**(struktura XML-a s digitalnim potpisom)

Osnove sigurnosti na internetu

Razine zaštite su onoliko visoke(jake, skupe) koliko je ono što se čuva vrijedno.

Obično se rade zaštite na nekoliko razina- sustavi, aplikacije, komunikacija

Zaštita sustava

Onemogućavanje upada u mrežu(firewall ili DMZ-demilitarizirana zona)

Onemogućavanje stražnjih vrata(backdoor)- zaštita i kontrola WLAN konekcija, zabrana korištenja modemskih priključaka, kontrola uporabe LAN priključaka

Praćenje neuobičajenih i potencijalno štetnih mrežnih aktivnosti-uporaba antivirusnih alata ili posebnih sigurnosnih alata

Zaštita aplikacije

Onemogućavanje stražnjih vrata-zaštita javnih servisa, provjera sigurnosti svih dijelova aplikacije, otvaranje sučelja samo prema poznatim klijentima, autentikacija svih klijenata

Praćenje neuobičajenih i štetnih aktivnosti- zapis svih aktivnosti

Sigurnosna testiranja aplikacije- simulacije namjernih napada, bolje spriječiti nego liječiti

Sigurnost komunikacije

Autentikacija korisnika, autorizacija korisnika za skup akcija, zaštita poruka od čitanja i mijenjanja, zaštita pristupa komunikacijskom kanalu i zaštita od čitanja podataka s komunikacijskog kanala

Najčešće sigurnosne tehnologije aplikacija Weba:

- Zaštita komunikacijskog kanala kriptiranjem- HTTPS
- Autentikacija korisnika- lozinka, token, certifikat
- Autentikacija klijenta- certifikat na strani klijenta
- Autentikacija poslužitelja- certifikat na strani poslužitelja

HTTPS je URI shema- default vrata broj 442(umjesto 80)

Enkripcija između HTTP i TCP sloja temeljena na poznatim kriptografskim protokolima(SSL) i korištenje certifikata->onemogućava niz napada raznih tipova

9. Besplatnost, sloboda, otvorenost u računarstvu

Tri vrste programske podrške:

- Vlasnički programi(proprietary)
- Besplatni programi(freeware)
- Slobodni(open source programi)(free software)

Vlasnički programi

- Zatvoreni kod- programski kod skriven od javnosti i konkurencije
- Zaštita od- umnožavanja i reverse engineeringa
- Zatvoreni timovi programera
- Isporuka izvršnih datoteka, bez source code-a

Besplatni program

- I dalje vlasničke programska podrška- All rights reserved(sva prava pridržana)
- Besplatno korištenje
- Izvršne datoteke, izvorni kod nije dostupan
- Autor može:
 - o Prestati razvijati programsku podršku
 - o Početi je naplaćivati
 - o Odlučiti koje norme implementirati
 - o Upravljati izvornim kodom kako želi

Slobodni programi

- „Free as in free speech, not as in free lunch“
- Poanta nije cijena, već sloboda korištenja
- Kultura otvorenosti
- Programska podrška koju možete koristiti, pregledavati, mijenjati, distribuirati
- Programska podrška se smije „Prodavati“- dijeliti uz naknadu(neograničeno veliku)

4 slobode slobodnih programa:

- 0) Sloboda pokretanja programa za sve namjene
- 1) Sloboda uvida u rad programa i prilagodbe potrebama- potreban pristup source code-u
- 2) Sloboda širenja preslika programa
- 3) Sloboda poboljšanja i dijeljenja poboljšane inačice drugima

Programi otvorenog koda

- Open source

Drugačiji od slobodne programske podrške po načinu pristupa- naglasak na različite ideje

Open source: praktična pitanja; free software: etička pozadina

Gotovo svi programi otvorenog koda su i slobodni(i obratno)

Licencije programa

License- official or legal permission to do or own a specified thing

Ugovori između autora i korisnika programa- definiraju prava uporabe

Licencije otvorenog koda obično na korisnika prenose i prava vlasništva dotične kopije.

Mnoštvo sličnih licencija

Licencije programa otvorenog koda:

- GNU General Purpose License(GNU GPL)
 - o Daje korisniku prava slobodne programske podrške
 - o Promjene programa i njihova redistribucija zadržavaju GPL licenciju
- BSD License
 - o Manje restriktivna licencija
 - o Moguća promjena licencije izmjenjenog koda
- Apache Software License
- MIT License
- Mozilla Public License
- Itd.

Otvoreni kod je dobar jer:

Implementacija međunarodnih normi- alati su korišteni po želji, no zbog međunarodnih normi podaci moraju biti međusobno izmjenjivi

Suradnja na projektu- brzi pronalasci i ispravci grešaka, velika baza korisnika, uzajamna pomoć, dugi životni vijek programskog proizvoda

Povećana sigurnost korisnika- tajne ne postoje, izvorni kod je dostupan

Početna investicija za licencije iznosi 0kn, no novac je usmjeren na instalaciju i prilagodbu sustava kupcima, podršku, održavanje, obrazovanje korisnika, certificiranje

Vlade, javne organizacije, mediji i fondovi preferiraju otvoreni kod jer novac od usluga ostaje u zemlji, a vraća se kroz porez.

Pri razvoju treba misliti na „obične“ korisnike->osnovni oblik zapisa mora podržati potrebe svih

Licenciranje „neprogramskog“ dijela- Creative Commons licencije

„All rights reserved“-c

„No rights reserved“- Public Domain

Temeljena na autorskim pravima

Odricanje nekih prava u korist općeg dobra-> „Some rights reserved“

Autor sam odabire „jačinu“ licence-nema posrednika u procesu licenciranja.

„slobodno smijete dijeliti, umnožavati, distribuirati i javnosti priopćavati djelo“, Ali...

Parametri licence

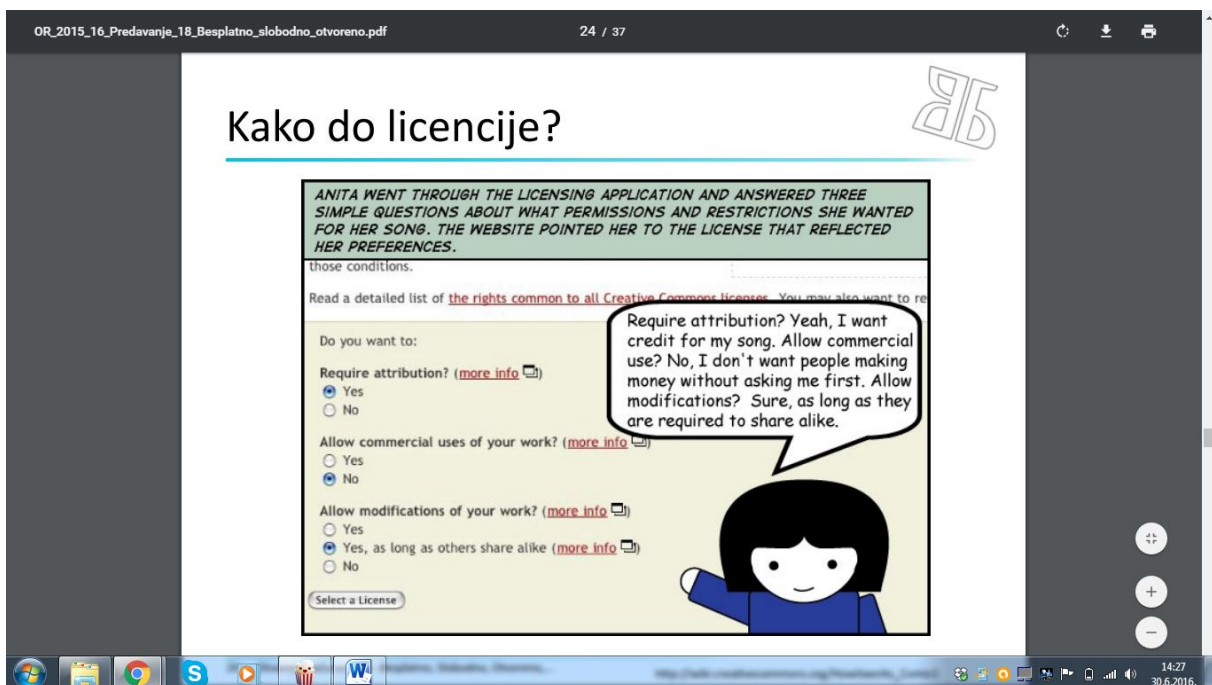
Attribution- imenovanje

Noncommercial- nekomercijalno

No derivative Works- bez prerada

Share alike- dijeli pod istim uvjetima

Moguće različite kombinacije ovih parametara, moguće pojedinačno odstupanje od definirane licence, uz dopuštenje autora



Rezultat izrade licence je čitljiv oblik za: ljude, pravnike i računala(XML)

KAO ZAKLJUČAK

Otvoreno računarstvo je:

- Sloboda u razmišljanju
- Način razmišljanja i pristupa problemu
- Neovisnost o proizvođačima
- Može se graditi, poboljšavati, prilagođavati, surađivati
- Može se svekoliko pomagati korisniku kojem je namijenjeno i olakšati mu rješavanje problema

OR temelji se na normama i otvara nove probleme oko sigurnosti

Dobar inženjer- kritički razmišlja, razmatra alternative, nepristrano odlučuje, **argumentirano stoji iza svojih odluka**

Otvorenost znači:

- Prenosivost(portability)
- Podesivost(scalability)
- Suradnja(interoperability)
- Dostupnost(accessibility)
- Prilagodljivost(adaptability)

PRIMJERI:

PHP- https://www.fer.unizg.hr/download/repository/OR_2015_16_Predavanje_10_PHP.pdf

(Predavanje br. 10; slajd 26)

Kolačići(cookies) u PHPu-

https://www.fer.unizg.hr/download/repository/OR_2015_16_predavanje_12_sjednice.pdf

(predavanje br.12; slajd 25)

Utičnice u PHP-u i primjeri-

https://www.fer.unizg.hr/download/repository/OR_2015_16_predavanje_13_mrežno_programiranje.pdf (predavanje br.13, slajd 27)