DOI:10.3359/oz0703151

KONFERENCA *JAVAONE 2007*

V San Franciscu je od 8. do 11. maja potekala konferenca JavaOne. To je bila že 12. konferenca pod okriljem podjetja Sun. Glavno sporočilo letošnje konference je bilo: "Odprimo vrata novim možnostim" in res se je veliko govorilo o odprtokodnih rešitvah pri razvoju javinega okolja, ki skupaj s skriptnimi jeziki, odprtokodnimi programskimi rešitvami ter drugimi orodji Web 2.0 prevzema vodilno vlogo pri razvoju nove generacije aplikacij.

Konferenca je impresivna zaradi števila ljudi, ki se zberejo na enem mestu (več kot 15.000) in so tako ali drugače povezani z javino in drugimi spletnimi tehnologijami. Ob taki množici ljudi je seveda organizacija zelo zapletena, ki pa jo številno osebje izvaja zelo profesionalno, tako da se tudi novinec na konferenci počuti zelo domače. Podobno kot že prejšnje leto se je bilo treba tudi letos predhodno prijaviti na posamezna predavanja, s čimer so si organizatorji vsaj malo olajšali delo pri rezervaciji predavalnic.

Številne dogodke na konferenci lahko v splošnem razdelimo na: generalne dogodke, tehnične dogodke, laboratorije ter spremljajoči program (angl. *births-of-feather*), v katerega sodijo razne demonstracije, neformalni dogodki, razprave, okrogle mize ipd.

Ob tem poteka še sejem, na katerem se predstavljajo številni ponudniki aplikacijskih rešitev, orodij in drugih tehnologij, ki uporabljajo javino razvojno okolje.

Dan pred uradno konferenco je letos prvič potekal t. i. dan skupnosti (CommunityOne), kjer se je zbralo več kot 2500 razvijalcev, IT inženirjev in podjetnikov, ki so izmenjevali tehnične informacije o odprtokodnih projektih. Predstavili so naslednje projekte: GlassFish, NetBeans, Linux vs. Solaris, OpenJDK/Mobile & Embedded, Web 2.0 in OpenSolaris.

PROJEKT GLASSFISH

Projekt GlassFish, ki je začel delovati pod okriljem podjetja Sun in katerega glavni namen je razvoj aplikacijskega strežnika v skladu s standardom Java EE, je v tem letu dobro zaživel. V tednu konference je bila predstavljena verzija beta (V2) in nekaj novosti, kot je izboljšanje tehnologije gruč, podpora tehnologiji JBI (Java Business Interface), povezljivost s tehnologijo WBIT (Web Services Interoperability technology). Poleg teh so opazne še nekatere izboljšave: možnost uporabe različnih uporabniških profilov, varnost (tehnologija ECC in format JKS), spletni vsebnik je podprt s protokolom Apache, nov uporabniški vmesnik, podprt s tehnologijo AJAX, novosti pri specifikacijah (JSR 196, JSR 208, JSP, JAX-WS, JAXB …). Za GlassFish, V3, se napoveduje, da ne bo več izključno vezan na tehnologijo Java EE.

OPENJDK/MOBILE&EMBEDDED

Pred dobrega pol leta sta pod okriljem podjetja Sun začeli delovati spletni skupnosti OpenJDK in Mobile&Embedded. V teh skupnostih je mogoče sodelovati pri izpeljavi in vključevanju novih idej v javino razvojno okolje. Pri skupnosti OpenJDK gre za ideje v Java SE, medtem ko skupnost Mobile&Embedded temelji na Java ME. Uporaba te kode je omogočena z uporabo licence GPL, V2 (General public license).

Sun je v tem času skupnosti omogočil dostop do treh pomembnih programskih komponent (Java HotSpot technology, Java Programming Language Compiler in JavaHelp Software). Preko projekta OpenJDK je tako razvijalcem omogočen dostop do prevajalnika, spoznavanje novih značilnosti jezika, spoznavanje izgradnje JVM (Virtual Machine), posredovanje idej za izboljšano arhitekturo JVM, iskanje napak …

S temi odprtokodnimi projekti je tako omogočen vpliv na bodočo izvedbo JDK in s tem pomoč Sunu pri iskanju novih možnosti za razvoj jave.

Na uvodnem generalnem predavanju, kjer vodilni Sunovi ljudje predstavljajo smernice razvoja javinega okolja, je Rich Green poudaril vlogo komunikacije kot gonila celotne človeške skupnosti. V zadnjem letu je bilo prodanih več kot pol milijona mobilnih telefonov, da je razmerje v odnosu do osebnih računalnikov že 20 : 1.

Platform	Open Source Community	License	Code
Java SE Platform	OpenJDK	GPLv2	Nov. 2006 May 2007
Java ME Platform	Mobile and Embedded	GPLv2	Nov. 2006
Java EE Platform	GlassFish Project	CDDL/ GPLv2	June 2005

Tabela 1: Odprtokodne rešitve

V ta namen je predstavil novo "družino" proizvodov **Java FX**, v katero spadata **JavaFX Script** in **JavaFX Mobile**. JavaFX Script je skriptni jezik za enostavno kreiranje grafičnih vmesnikov (GUI), medtem ko JavaFX Mobile omogoča boljšo uporabo javinih aplikacij na mobilnih telefonih.

Prav tako je poudaril vlogo razvojnega orodja **NetBeans IDE 6.0**, ki vsebuje številne novosti:dinamično skriptiranje z javino tehnologijo, JRuby 1.0 in tehnologijo JavaScript, kreator za kreiranje grafičnih vmesnikov (GUI), ki zelo poenostavlja kreiranje dinamičnih uporabniških aplikacij, nov izboljšan urejevalnik in številni paketi, ki olajšajo delo pri razvoju aplikacij. Prav tako to orodje omogoča povezljivost z odprtokodno verzijo Jave OpenJDK. Dejal je, da je to orodje v tem trenutku primarno pri razvoju aplikacij za mobilne telefone, ki so bili nasploh v središču te uvodne predstavitve.

Na preostalih generalnih predavanjih so podjetja Oracle, Intel in Motorola predstavila svoje izkušnje pri uporabi javinega razvojnega okolja, medtem ko je zadnji dan konference James Gosling že tradicionalno podeljeval nagrade v "predstavitvi igrač", ki temeljijo na javinem okolju.

V nadaljevanju poročila so predstavljene nekatere podrobnosti, novosti in zanimivosti s predavanj, ki sva jih na tej konferenci obiskovala.

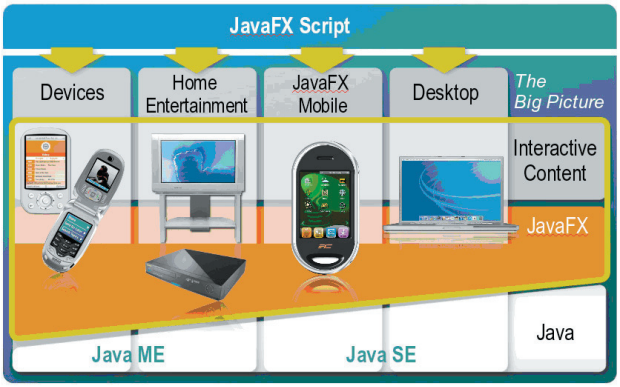
JAVA FX

JavaFX predstavlja družino proizvodov, ki temeljijo na računalniškem okolju jave in omogočajo njeno uporabo v mobilnih napravah, pri kreiranju namizij in grafičnih vmesnikov (GUI-jev), pri TV-komunikatorjih, tehnologiji Blu-ray ipd. Namen JaveFX je olajšanje razvoja aplikacij za te naprave in tehnologije. V to družino spadata proizvoda JavaFX Script in JavaFX Mobile. JavaFX Script je skriptni jezik, ki omogoča razvijalcem kreiranje zmogljivejših in uporabnejših aplikacij ter storitev za mobilne naprave, diske Blu-ray, TV-komunikatorje in namizja.

JavaFX Script uporablja sintakso za specifikacijo GUI-komponent, tako da je koda "bližje" grafičnemu vmesniku, prav tako omogoča enostavnejšo povezavo med podatki iz aplikacije in uporabniškim vmesnikom, mogoče ga je vključiti v različna razvojna okolja (Eclipse, NetBeans …), je statično tipiziran in strukturiran, tako da omogoča kreiranje zelo velikih aplikacij, prav tako omogoča kreiranje zelo kompleksnih grafičnih vmesnikov in prijaznejšo uporabo komponent Swing. V tem obdobju je preizkušanje kode v fazi alfa, je pa na voljo na spletišču java.net in bo izdana z dovoljenjem GPL 2.

JavaFX Mobile je ime za programsko opremo (v javinem okolju), ki podpira novo generacijo mobilnih naprav in zaobsega ogrodja, ki predstavljajo grafično, komunikacijsko in multimedijsko podporo tem napravam. Prav tako vključuje shranjevanje podatkov in programsko podporo telefoniji. Ta programska oprema je dostopna preko vgrajenih programskih razredov (Java API). Ideja JavaFX Mobile je pocenitev mobilnih naprav, kar bo omogočilo večjo dostopnost teh naprav in s tem večjo komunikacijo med ljudmi.

TEHNOLOGIJA DESKTOP JAVA



Slika 1: JavaFX Script

Na začetku predavanja so predstavili nekaj številk, ki govorijo o priljubljenosti Jave Desktop in javine tehnologije nasploh. Od januarja 2007 je bilo izvedenih več kot 50 milijonov namestitev javinega okolja, tako da ni presenetljivo, da že več kot 77 odstotkov vseh osebnih računalnikov uporablja javino tehnologijo.

Zato je eden izmed ciljev nove družine JavaFX še bolj približati to tehnologijo spletnim in grafičnim razvijalcem. JavaFX Script tako omogoča vljučitev različnih javinih razredov, kreiranje novih objektov, uporabo njihovih metod in javinih vmesnikov. Prav to dela komponente Swing in Java-2D prijaznejše in s tem lažje za kreiranje GUI-aplikacij. Na področju tehnologije JavaFX Mobile pa Sun načrtuje razvoj

namizja Desktop API, medtem ko prva verzija JavaFX vsebuje Swing in 2D komponente za mobilno tehnologijo.

Java SE, na kateri temelji JavaFX, pa za razvoj namizja omogoča: dostop do namizja API, uporabo razredov TrayIcon, izboljšan Look and Feel, podporo za Windows Visto, razred SwingWorker, projekt Matisse in Group Layout, sortiranje in filtriranje jTable, LCD-tekst, lastnosti Desktop AA text, Splash Screen …

Seveda pa še vedno obstajajo nekateri problemi, ki se jih razvijalci pri podjetju Sun zavedajo in se z njimi intenzivno ukvarjajo. Nekaj problemov: zagonski čas, namestitveni čas in procesi, detektiranje JRE.

Rešitve teh problemov bodo predstavljene v modernizirani verziji Jave SE 6, ki bo vsebovala: QuickStarter, Kernel, Deployment Toolkit, grafični pospeševalnik za Windows in Nimbus Look&Feel. V prihodnosti se prav tako pričakuje vgradnja 3D API-ja v javino okolje.

RUBY ON RAILS

Ruby je dinamično tipiziran objektno orientiran odprtokodni programski jezik, napisan v programskem jeziku C, ki teži k čim večji enostavnosti, večji produktivnosti in zabavi pri samem delu. Ruby on Rails je odprtokodno spletno ogrodje (napisano v jeziku ruby), pri katerem se teži k enostavnemu in hitremu programiranju spletnih aplikacij, neponavljanju kode (DRY), integriranemu testiranju, kar so tudi glavne prednosti pred razvojem aplikacij v javinem razvojnem okolju. Prav tako je značilna hitra rast skupnosti razvijalcev, kar seveda posledično prinese večjo bazo uporabnih primerov.

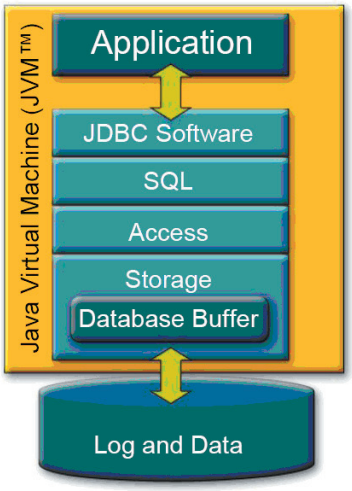
Jezik ruby v javinem okolju se imenuje JRuby, tako da je omogočena tudi povezava Ruby on Rails z javino tehnologijo. S tem je omogočen dostop do javinih aplikacijskih strežnikov, širša in skalabilnejša podpora bazam, enostavnejša povezljivost ogrodja s celotno arhitekturo, povezljivost z javinimi knjižnicami in servisi …

Skupaj s tehnologijo Java EE je tako omogočen dostop do baz, dostop do javinih API-jev in drugih tehnologij: Java Persistence API (JPA), Java Management Extensions (JMX), Enterprise JavaBeans (EJB), Java Message Service (JMS), API ter tehnologije SOAP, WSDL in SOA.

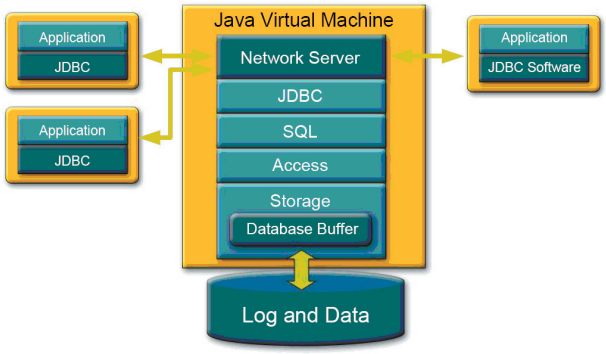
JAVA DB

Na kratko bomo opisali bazo Java DB, njeno uporabo in nastavitve za boljše delovanje in predstavili izboljšave v novejši verziji.

Java DB je odprtokodna relacijska baza,ki se razvija v skupnosti Apache Derby. Jedro baze temelji na entitetno-relacijski tehnologiji. Popolnoma je kompatibilna, razvita na tehnologiji java, skladno s standardoma JDBC in SQL. Vključena je v Javo JDK 6.0 in projekt GlassFish. Podprta je s programskimi orodji NetBeans, Sun Java Studio Enterprise, Eclipse … Zelo primerna je za javine aplikacije, ki potrebujejo izmenjavo podatkov z relacijskimi podatkovnimi bazami. Podpira uporabo večjega števila uporabnikov za delo s podatki. Poskrbljeno je za varnost (angl. *data encrypton, client authentication*). Baza podpira vgrajeno (angl. *embedded*) arhitekturo in arhitekturo odjemalec–strežnik (angl. *Client–Server*), kar prikazujeta slika 2 in slika 3.



Slika2: Prikaz arhitekture z vgrajeno bazo Java DB

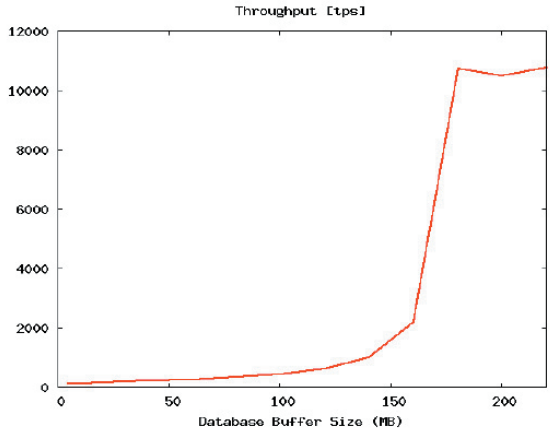


Slika 3: Prikaz arhitekture odjemalec–strežnik

Pravilni pristopi pri uporabi baze Java DB zagotavljajo optimalno delovanje baze. Ti pristopi so:

1. Za uporabnikove podatke in log-podatke je treba uporabiti dva fizična diska. S tem se poveča propustnost za dobrih 20 odstotkov.

2. Uporaba večjega pomnilnika za pogosteje uporabljene podatke dodobra poveča prepustnost. Slika 4 prikazuje prepustnost v odvisnosti od velikosti pomnilnika.



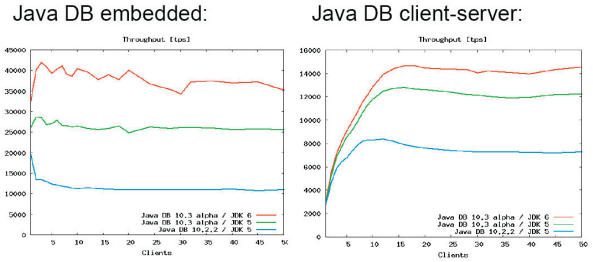
Slika 4: Prikaz pretoka pri uporabi večjega pomnilnika

3. Zaželena je uporaba Embedded Java DB, ker dosežemo bolj optimalno delovanje relacijske podatkovne baze. Zmanjšamo medprocesorske in strežniške komunikacije. Zmanjšamo obremenjevanje procesorja in razbremenimo strojno opremo. S tem posredno dosežemo boljšo skalabilnost (vse na enem računalniku). Če primerjamo Embedded Java DB in Client–Server Java DB, dosežemo z Embedded Java DB za dobrih 20 odstotkov boljšo prepustnost in za slabih 50 odstotkov manjšo obremenjenost procesorja.
4. Pri izvajanju vprašalnih stavkov je primernejša uporaba pripravljenih stavkov kot pa hkratno izdelovanje vprašalnih stavkov. S tem pristopom dodobra povečamo prepustnost podatkovne baze in zmanjšamo obremenitev procesorja.
5. Za iskanje podatkov v podatkovni bazi je primerno uporabiti indekse, s katerimi lahko omogočimo hitrejše iskanje. Z uporabo indeksov bomo tako pospešili delovanje podatkovne baze. Ravno tako je priporočljivo zapiranje objektov JDBC po njihovi uporabi. Zaželena je tudi uporaba transakcij in ne uporaba avtomatskega potrjevanja transakcij (angl. *auto-commit*).

V novejši verziji Java DB, 10.3 alpha, so dodane naslednje novosti:

1. V različici z vgrajeno bazo:
   - zmanjšana obremenitev procesorja,
   - zmanjšana obremenitev diska pri zapisovanju log-podatkov,
   - od 30–50 odstotkov povečana prepustnost.
2. V različici odjemalec–strežnik:
   - izboljšan tok za LOB-podatke.
3. Optimizacija SQL-a:
   - izboljšana optimizacija.

Na sliki 5 dva grafa prikazujeta različne baze med sabo.



Slika 5: Prikaz pretoka pri uporabi novejše verzije Java DB, 10.3 alpha

S pomočjo Java DB je zagotovljen dostop do podatkovnih baz z uporabo jave. Baza je vključena v Java EE in tako ni treba več uporabljati kakšne druge podatkovne baze, če želimo shranjevati podatke.

ECC

ECC – Elliptic Curve Cryptography je kriptografski sistem z javnim ključem (podobno kot RSA), ki temelji na diskretnem logaritmu eliptičnih krivulj. Reševanje teh logaritmov je zahtevnejše kot reševanje cikličnih grup, ki temeljijo na celih številih, so pa lahko ključi pri šifriranju z eliptično krivuljo krajši kot pri sistemih, ki temeljijo na celih številih.

Poenostavljeno povedano: obstaja matematični problem, da imamo eliptično krivuljo, na kateri poznamo točki P in Q in treba je najti koeficient k, tako da velja $Q = kP$.

RSA Key Size (bits)	ECC Key Size for equivalent security
1024	160
2048	224
3072	256
7680	384
15360	521

Tabela 2: Primerjava velikosti ključev ob enaki varnosti

Krivulje (vseh 25) so standardizirane s standardi NIST, SECG in ANSI, medtem ko algoritme označujemo z znakovnimi nizi. Več o tem je mogoče prebrati v dokumentu RFC 4492 (http://www.faqs.org/rfcs/rfc4492.html).

V tem trenutku ni preplaha za tehnologijo ECC, vendar pa se pričakuje, da bodo v prihodnosti na področju protokola SSL prevladale rešitve, ki bodo temeljile na tehnologiji ECC.

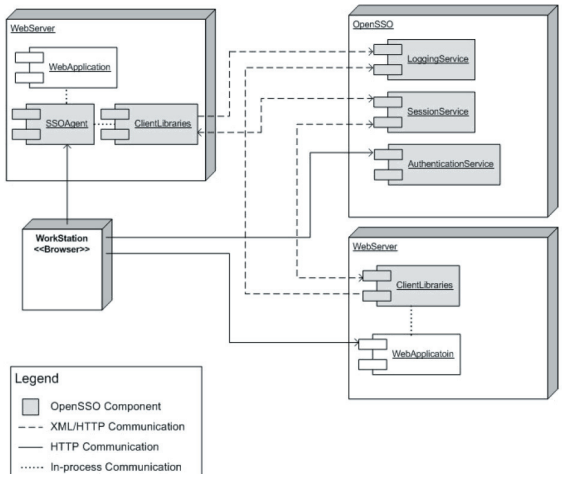
SSO

SSO – Single Sign-on predstavlja obliko enovite prijave in uporabe aplikacij. Prednosti tehnologije SSO so: enkratna prijava za dostop do različnih virov, večja produktivnost uporabnikov in razvijalcev, enostavnejša administracija. Seveda pa imajo rešitve SSO tudi svoje slabosti: potencialni vdor v sistem je tako z enkratno prijavo bistveno olajšan, obstajajo resne težave pri vgradnji rešitev SSO v obstoječe aplikacije.

Obstajajo tri glavne odprtokodne oblike rešitev SSO: OpenSSO, JOSSO in CAS.

OpenSSO

OpenSSO – Open Web Single Sign On je odprtokodni projekt, ki ga spodbuja Sun. Ta skrbi za infrastukturo javnodostopnih identifikacijskih servisov za spletne aplikacije.

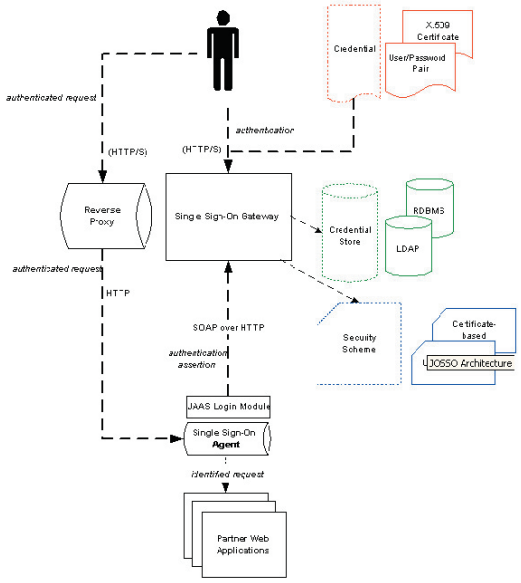


Slika 6: Arhitektura OpenSSO

JOSSO

JOSSO – Java Open Single Sign On sloni na Java Authentication and Authorization Service (JAAS), uporablja spletni strežnik z Apache Axis ter standarde Apache Struts ter JavaServer Page technology(JSP). Prav tako omogoča uporabo komponent Reverse Proxy, ki omogočajo n-slojno konfiguracijo SSO. Ta uporablja različne strategije za uporabo in shranjevanje uporabniških informacij, overitev v LDAP, podatkovnih baz, XML-datotek …

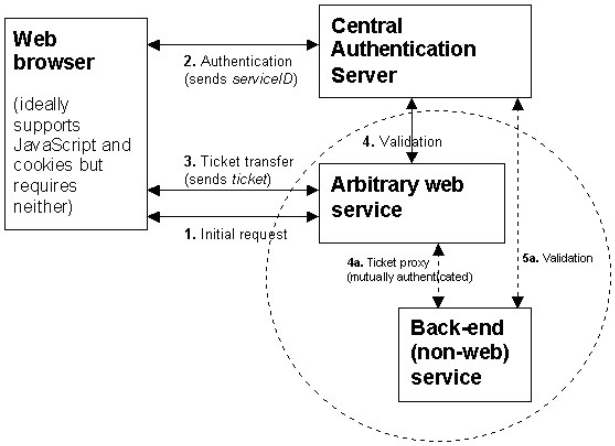
JOSSO je mogoče povezati z aplikacijskima strežnikoma TomCat in JBoss, seveda je pa z njim mogoče upravljati javine spletne aplikacije.



Slika 7: Arhitektura JOSSO

CAS

CAS – Central Authentication Service je odprt in dobro dokumentiran protokol, ki ga je mogoče povezati z javino tehnologijo, kot je .NET, PHP, Perl, Apache, uPortal … Ta protokol je tudi podprt z dokumentacijo, saj skupnost uporabnikov hitro narašča.



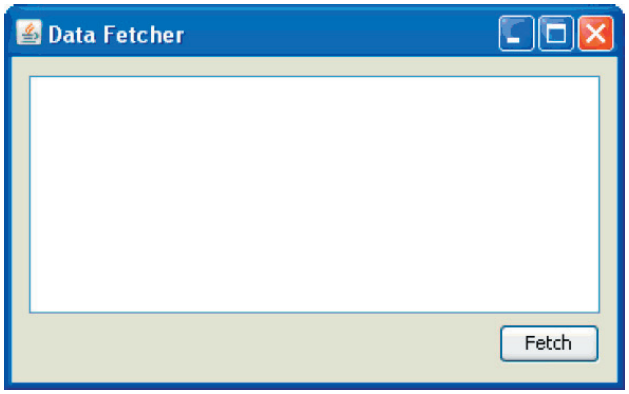
Slika 8: Arhitektura CAS

Vsi trije protokoli so odprtokodni, ideje in rešitve je mogoče najti na spletnih straneh projektov,[1] kjer je tudi urejena dokumentacija, uporabniški forumi in seznami e-naslovov uporabnikov.

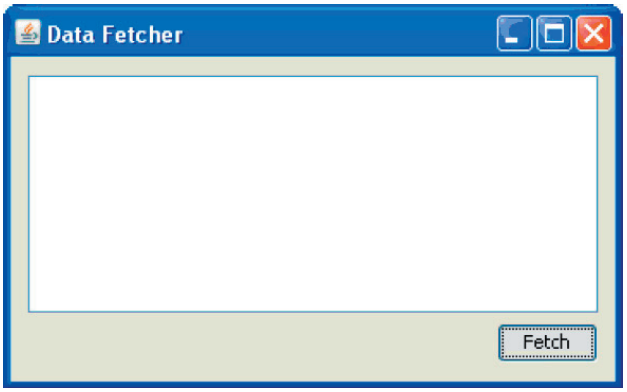
SWING Z UPORABO VEČNITNOSTI

S pomočjo večnitnosti postanejo aplikacije Swing dovzetne za različne akcije.

Uporabnika je treba informirati o izvajanju akcije, ki traja dalj časa. V takih primerih je zelo uporabno, da se za izvajanje akcije uporabi nova nit. Naslednje slike prikazujejo primere, kakšna je videti pravilno zastavljena aplikacija.



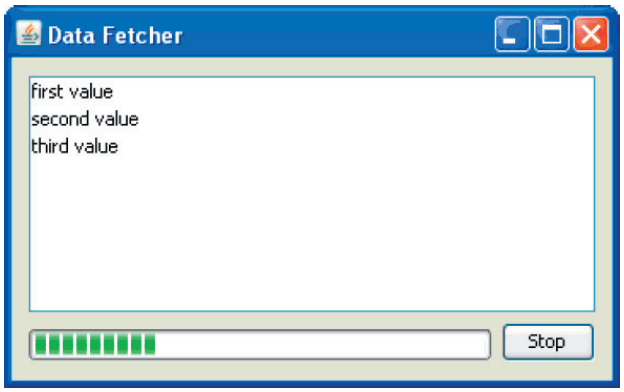
Slika 9: Prikaz okna pred akcijo Fetch



Slika 10: Prikaz okna po akciji Fetch po nekem času

Slika 9 prikazuje okno pred akcijo Fetch, ki potrebuje za svojo izvedbo dosti časa. Ko pritisnemo na tipko Fetch, se začne akcija izvajati. To zamrzne okno, kar prikazuje slika 10. V tem času uporabnik ne ve, kaj se s sistemom dogaja. V tem primeru bi bilo zelo koristno, če bi uporabniku prikazovali potek opravljanja različnih akcij, ki se v tem času izvajajo.

Zato je primerna uporaba aplikacije, ki je dovzetna za različne akcije. To lahko naredimo s pomočjo večnitnosti. Slika 11 prikazuje okno, ki uporablja večnitnost aplikacije Swing. Na tak način lahko dosežemo, da je uporabnik neprestano obveščen o delovanju sistema. Tako bo uporabnik ves čas vedel, kaj se s sistemom dogaja.



Slika 11: Prikaz okna po akciji Fetch po nekem času

V Javi 6 so ponudili razred SwingWorker, ki nam omogoča enostavno izgradnjo takšnega mehanizma.

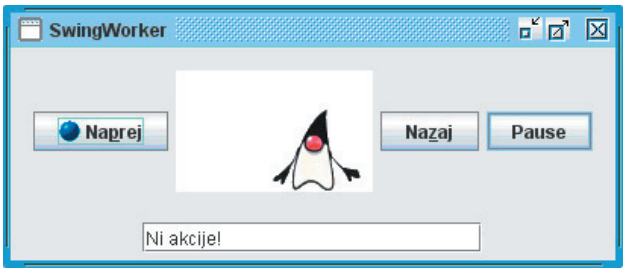
SwingWorker je abstrakten razred, ki ga je treba pred uporabo seveda uvesti. Pri SwingWorker<T, V> je treba definirati generična tipa <T, V>, pri čemer T predstavlja rezultat, ki ga SwingWorker vrne, V pa vmesne rezultate. Zelo pomembna metoda v SwingWorker-ju je metoda “doInBackground()”. V tej metodi je treba definirati vse akcije, ki se dogajajo v ozadju. S pomočjo metode “execute()” se zažene nit za SwingWorker. SwingWorker ravno tako zažene metodo “process()” in “done()”, s katerima se lahko kontrolirajo različne akcije.

```
private class Worker extends SwingWorker<String, Object> {

  public String doInBackground() {
    Object[] chunk = new String[5];
    for (int i=0; i<5; i++) chunk[i] = "To je chunk "+i+", ";
    //Object chunk = new String("argg");
    publish(chunk);
    pause(this);
    return "To je to!";
  }

  protected void done() {
    System.out.println("Konec"+" "+getState()+" "+
        "Is cancelled: "+isCancelled()+" "+"Is done: "+isDone());
    try {
      System.out.println("Result: "+get());
    } catch (InterruptedException e) {
      e.printStackTrace();
    } catch (ExecutionException e) {
      e.printStackTrace();
    }
  }

  protected void process(List<Object> chunks) {
    System.out.print("To je chunk: ");
    for (int i=0; i<chunks.size(); i++) {
      System.out.print(chunks.get(i)+" ");
    }
    System.out.println();
  }

}
```

Slika 12: Prikaz primera programa za SwingWorker

Primer na sliki 12 prikazuje uporabo vseh pomembnejših metod za SwingWorker. Program ne izvaja kakšnih pomembnejših funkcij, preprosto zažene kratko metodo, ki ima vlogo pri pisanju na konzolo in pisanju sporočila v

grafičnem vmesniku JTextField. Grafični vmesnik prikazuje slika 13.



Slika 13: Prikaz grafičnega vmesnika

Ob pritisku na tipko Pause se zažene SwingWorker z metodo "execute()". Zažene se dodatna nit, v kateri se izvajajo akcije. Istočasno lahko spreminjamo sličico s tipkama Naprej in Nazaj. V našem primeru se izvaja akcija v metodi "pause(this)", ki je prikazana na sliki 13.

```
private void pause(Worker worker) {
  for (int i=0; i<5; i++) {
    if (worker != null) {
    System.out.println("To je: "+i+" "+worker.getState()+" "+
          "Is cancelled: "+worker.isCancelled()+" "+"Is done: "+worker.isDone());
    this.textField.setText("Dela na akciji: "+String.valueOf(i+1));
    }
    try {
      Thread.sleep(1000);
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
  }
}
```

Slika 14: Prikaz metode "pause()"

Po končanem izvajanju se še izvede metoda "done()" in nato se še ustavi izvajanje dodatne niti.

JAVA RTS

V nadaljevanju bomo opisali, kako se Java RTS Real-Time System uporablja v časovno realnih sistemih v kombinaciji z operacijskim sistemom Solaris "real time". Spregovorili bomo tudi nekaj besed o sami filozofiji časovno realnih sistemov.

Pri sistemih, ki se morajo odvijati v realnem času, večina pomisli, da se mora neka akcija zgoditi zelo hitro. Hkrati povežemo zelo hitre akcije z zelo zmogljivimi računalniki. V realnosti pa temu ni tako. Pri sistemih v realnem času govorimo o akcijah, ki se bodo zgodile v vnaprej določenem in znanem času. Čas, v katerem se bodo akcije zgodile, je ravno tako pomemben, kot logični rezultat te akcije. Razmišljanje, da se mora neka akcija izvesti izredno hitro, je napačna.

Na tem mestu se ponudi tudi vprašanje, zakaj se mora Real-Time System integrirati v javo. Predvsem zaradi enostavnosti, saj želimo večino stvari napisati v programskem jeziku, ki je znan, enostaven, na visokem nivoju in napreden. V primerjavi s programskim jezikom, v katerem se programira na nižjem nivoju, je takšen nov koncept veliko lažji in preglednejši. Ravno tako ne potrebujemo dodatnega človeka, ki bo pisal kodo v programskem jeziku na nižjem nivoju.

Java RTS Real-Time System se lahko uporablja v vojski, telekomunikacijah, bankah in industriji. V vseh teh panogah zadosti vsem pogojem sistemov, ki morajo delovati v realnem času.

Za Java RTS Real-Time System je definiran standard JSR001, ki natančno definira obnašanje jave v sistemu z realnim časom. Java RTS 2.0 Real-Time System je implementirana v skladu z JSR001 in bazira na javinem okolju Java SE 5. Deluje na operacijskem sistemu Sun Solaris, tehnologiji SPARC in družini procesorjev x86/x64. Seveda mora zaradi zahteve realnega časa operacijski sistem Sun Solaris omogočati funkcije realnega časa. Poleg Java RTS Real-Time System je vključen tudi inovativni Real-Time Garbage Collector. Java RTS 2.0 deluje v enostavnem sistemu z eno ploščo in v kompleksnem strežniškem sistemu.

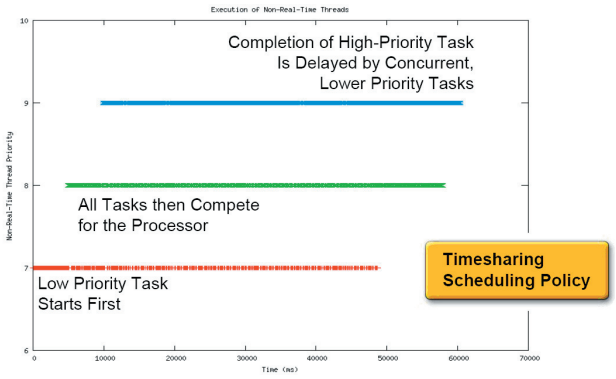
V naslednjih treh poglavjih bomo na kratko predstavili, kako se morajo sistemi v realnem času obnašati.

OBRAVNAVANJE NITI

Predpostavimo, da imamo tri naloge in naslednje pogoje:

- nizka prioriteta,
- srednja prioriteta, zažene se 5 sekund pozneje,
- visoka prioriteta, zažene se 5 sekund pozneje,
- vsaka naloga potrebuje 20 sekund za svoje delovanje,
- na razpolago je samo en procesor.

Na naslednji sliki bo prikazano časovno delovanje vsake naloge.



Slika 15: Prikaz treh nalog, ki se prepletajo

Iz slike 15 je lepo razvidno, da je visoko prioritetna naloga upočasnjena z nizko prioritetno nalogo. Pravilno izvajanje takšnih nalog bi bilo takšno, da bi se naloga z visoko prioriteto izvedla prva, nato pa bi se izvedle po prioriteti vse preostale naloge.

Pri zamenjavi določenih razredov, ki upoštevajo izvajanje prioritetnih nalog, se izvajanje nalog spremeni. To je vidno na sliki 16. Iz slike 15 in slike 17 je vidna razlika v izvajanju prioritetnih nalog.

- Replace:

```
Thread T = new java.lang.Thread();
T.setPriority(prio);
```

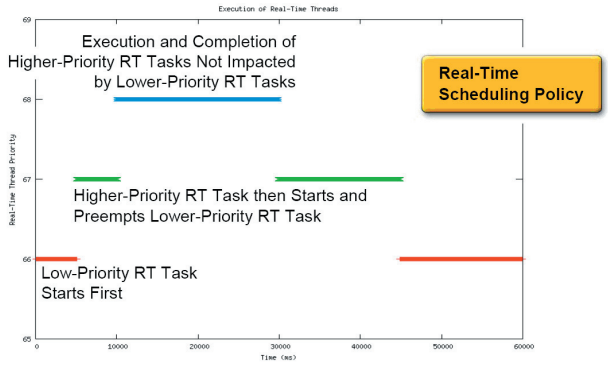
- With:

```
RealtimeThread RTT =
   new javax.realtime.RealtimeThread();
RTT.setSchedulingParameters(prioParms);
```

- Then…

Slika 16: Prikaz sprememb v programu

Po teh spremembah v programu se bo izvajanje nalog izvajalo na naslednji način.



Slika 17: Prikaz treh nalog, ki se prepletajo, v prioritetnem načinu delovanja

UPRAVLJANJE ASINHRONIH DOGODKOV

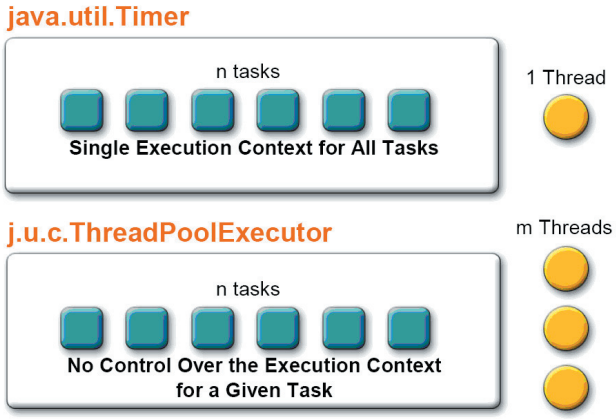
Večino obstoječih fizičnih sistemov ima dva načina asinhronega obnašanja:

- časovno proženje,
- dogodkovno proženje.

Koncept RTSJ ima naslednji način obnašanja:

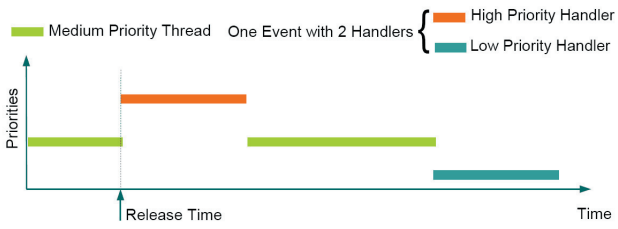
- za en dogodek se lahko uporabi več upravljavcev,
- izvajanje logike je razporejeno in dodeljeno več upravljavcem.

Slika 18 prikazuje filozofijo RTS pri upravljanju asinhronih dogodkov.



Slika 18: Asinhrono izvajanje v javinem okolju

Ko se v izvajanju sistema pojavi neki dogodek, se vsi pripadajoči upravljavci sprostijo. Dogodki se začnejo nato izvajati glede na upravljavčeve parametre. To delovanje prikazuje slika 19.

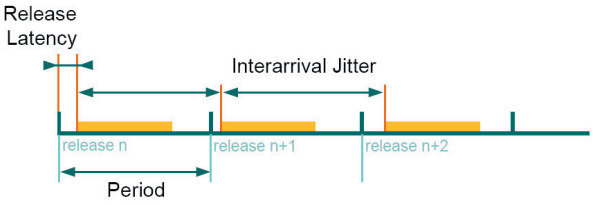


Slika 19: Prikaz koncepta "sprôsti in začni izvajati"

V Java SE RTS 2.0 lahko takšno izvajanje omogoča razred AsyncEventHandler. Za ta razred obstaja tudi družina razredov, ki izpolnjujejo še razne dodatne zahteve.

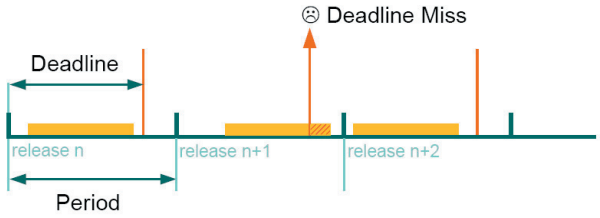
PERIODIČNO IZVAJANJE

Uporablja se predvsem za namene regulacij (closed-loop, PID controlers). Periodično izvajanje prikazuje slika 20.



Slika 20: Prikaz periodičnega izvajanja

Periodično izvajanje nadzorujemo s pomočjo razreda PeriodicParameters (start, period). Java RTS ima vgrajeno periodično izvajanje ter izvajanje spremljanja in nadzorovanja (angl. *monitoring*). Filozofijo le-tega prikazuje slika 21.



Slika21: Prikaz filozofije spremljanja in nadzorovanja

Periodi se določi končni rok. Če se določena naloga izvaja predolgo in se zastavljeni rok prekorači, se aktivira DeadlineMiss. Nato se morajo izvesti določene akcije. DeadlineMiss se določi na naslednji način.

- Deferred to a deadline miss handler

```
ReleaseParameters.setDeadlineMissHandler(
    AsyncEventHandler handler);
```

- Or, if no handler, performed by the thread itself

```
if (waitForNextPeriod() == false) {
    handle_deadline_miss();
}
```

Slika 22: Prikaz, kako se določi upravljavec napak pri prekoračitvi roka

Kot smo že omenili, sta za sistem v realnem času pomembna nadzor in čas. Opisali smo nekaj mehanizmov, ki nam to omogočajo. Z njihovo pomočjo je mogoče izvesti sistem, ki bo zanesljivo deloval v realnem času. Za pravilno izvajanje je obvezna uporaba operacijskega sistema, ki izvaja funkcije v realnem času, kot je na primer Sun RT-Solaris.

Opomba

1 Spletna stran OpenSSO im naslov https://opensso.dev.java.net/; JOSSO http://www.josso.org/ in CAS: http://www.ja-sig.org/products/cas/.

Anton Zorko, Martin Kostanjevec