UDC: 004.6 Info M: str. 34-43

KOMPARATIVNA ANALIZA MOJARRA I APACHE MYFACES TEHNOLOGIJA I NAJKORIŠĆENIJIH JAVA SERVER FACES BIBLIOTEKA KOMPONENATA
COMPARATIVE ANALYSIS OF MOJARRA AND APACHE MYFACES TECHNOLOGY AND MOST WIDELY USED JAVA SERVER FACES COMPONENT LIBRARIES

Veljko Bogosavljević, Siniša Vlajić, Dušan Savić, Ilija Antović, Miloš Milić

REZIME: Predmet istraživanja ovog rada jeste komparativna analiza Oracle Mojarra i Apache MyFaces tehnologija, kao i analiza najznačajnijih i najkorišćenijih JavaServer Faces (JSF) biblioteka komponenata. U ovom radu, JSF će se koristiti u dva različita konteksta, i to: 1) u kontekstu specifikacije za razvoj Java WEB aplikacija, i 2) u kontekstu Java WEB okvira. U kontekstu specifikacije, u radu će biti dat hronološki prikaz različitih verzija JSF specifikacije, kao i komparativna analiza dve najpoznatije implementacije JSF specifikacije: Oracle Mojarra i Apache MyFaces. U kontekstu Java WEB okvira, JSF će se posmatrati kroz skup različitih biblioteka komponenti za razvoj Java WEB korisničkog interfejsa. U skladu sa time biće dat pregled i analiza najznačajnijih JSF biblioteka komponenata. Analiza praktične primene implementacija JSF specifikacije i različitih komponenti, kao i njihova komparativna analiza, zasnovana je na studijskom primeru.

KLJUČNE REČI: JavaServer Faces (JSF), JSF Okvir, JSF Specifikacija, JSF Implementacije Oracle Mojarra i Apache MyFaces, JSF Biblioteke Komponenata

ABSTRACT: Subject of this paper's study is comparative analysis of Oracle Mojarra and Apache MyFaces technologies, and an analysis of the most important and most frequently used JavaServer Faces (JSF) component libraries. In this paper, the JSF will be used in two different contexts, namely: 1) in the context of the specification for development of Java WEB applications, and 2) in the context of Java WEB framework. In the context of the specification, this paper provides a chronological overview of the different versions of the JSF specification, as well as a comparative analysis of the two most popular JSF implementations: Oracle Mojarra and Apache MyFaces. In the context of Java WEB framework, JSF will be observed through a different set of component libraries for developing Java WEB user interface. In accordance with that, we will give an overview and analysis of the major JSF library components. Analysis of practical application of the JSF implementations and analysis of practical application of the various components, as well as their comparative analysis, is based on a study case.

KEY WORDS: JavaServer Faces (JSF), JSF Framework, JSF Specification, JSF Implementations Oracle Mojarra i Apache MyFaces, JSF Component Library

1. UVOD

JavaServer Faces [JSR 314] bi se mogao definisati kao okvir (*framework*) kojim su definisani modeli komponenata korisničkog interfejsa *WEB* aplikacija baziranih na *Java* tehnologijama. *JavaServer Faces* deo je *Java EE* platforme [JSR 151] i jedan od najpopularnijih i najčešće korišćenih okvira za izradu grafičkog korisničkog interfejsa *WEB* aplikacija baziranih na *Java* tehnologijama. *JavaServer Faces* u velikoj meri pojednostavljuje razvoj sofisticiranog i komplikovanog korisničkog interfejsa WEB aplikacija, omogućava razvoj aplikacija bez previše briga o detaljima *HTTP* protokola, a dizajnerima bez programerskog znanja lakši rad sa korisničkim interfejsom. Iako je većina *WEB* aplikacija razvijana korišćenjem *HTML*-a kao *markup* jezika, *JavaServer Faces* nije ograničen *HTML*-om niti bilo kojim drugim *markup* jezikom. *Renderi* (*renderers*) su odvojeni od komponenti korisničkog interfejsa i kontrolišu *markup* koji se šalje klijentu, pa tako ista komponenta korisničkog interfejsa u kombinaciji sa različitim *renderima* može da proizvede različite izlaze, na primer *HTML* ili *WML* elemente, odnosno, *JavaServer Faces* aplikacije će raditi dobro čak i na pregledačima (*browsers*) namenjenim mobilnim uređajima. Kao i većina drugih tehnologija, *JavaServer Faces* poseduje sopstveni skup termina koji formiraju konceptualnu osnovu. U cilju boljeg razumevanja, u nastavku će biti pokriveni ključni koncepti i termini, ali takođe i predstavljene veze između njih.

Komponente korisničkog interfejsa (*UI Components*) - Postojan objekat koji se čuva na serveru, pruža specifične funkcionalnosti za potrebe interakcije sa krajnjim korisnikom. Imaju svoja svojstva, metode i događaje. Organizovane su u formi stabla komponenata koje se prikazuju na određenoj strani.

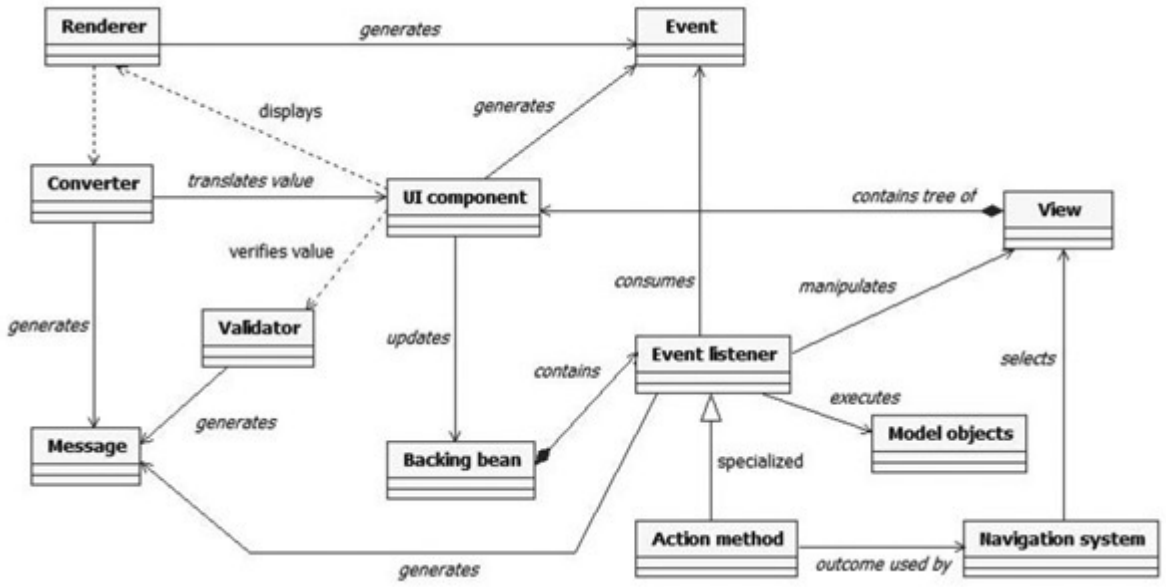
Renderi (*Renderers*) - Odgovorni su za prikaz komponenata korisničkog interfejsa. Mogu biti dizajnirani da rade sa jednom ili više komponenata korisničkog interfejsa, a jedna komponenta korisničkog interfejsa može biti vezana za više različitih rendera.

Validatori (*Validators*) - Odgovorni za proveru korisnički unetih vrednosti i osiguravanje da su one prihvatljive. Jedan ili više validatora mogu biti vezani za jednu komponentu korisničkog interfejsa.

Managed Beans - Specijalizovane klase koje primaju vrednosti komponenata korisničkog interfejsa i implementiraju metode koje se pozivaju prilikom nekog događaja.

Konverteri (*Converters*) - Konvertuju vrednosti komponenata iz tekstalnog zapisa u odgovarajući tip, ali i vrednosti iz određenog tipa u odgovarajući tekstualni zapis za prikaz. Jedan konverter može biti vezan za jednu komponentu korisničkog interfejsa.

Događaji i osluškivači (*Events and Listeners*) - *JavaServer Faces* koristi proveren *JavaBeans* model događaja i osluškivača. Komponente korisničkog interfejsa generišu određene događaje, a osluškivači mogu te događaje obrađivati.



Slika 1: *UML dijagram klasa koji predstavlja vezu ključnih JavaServer Faces koncepata*[1]

Poruke (*Messages*) - Povratne informacije koje se prikazuju korisniku. Bilo koji deo aplikacije može generisati određene povratne informacije kao što je na primer poruka o grešci koja se može dalje prikazati korisniku.

Navigacija (*Navigation*) - Sposobnost prelaska sa jedne strane na drugu. *JavaServer Faces* poseduje moćan sistem za navigaciju koji je integrisan sa specijalizovanim osluškivačima događaja.

Dalje, potrebno je sagledati način na koji su gore pomenuti koncepti povezani. Na *slici 1* prikazan je *UML* dijagram klasa na kome je prikazan svaki od koncepata i njegove veze.

Komponente korisničkog interfejsa predstavljaju samu suštinu *JavaServer Faces*-a, a najveća prednost *JavaServer Faces* ogleda se u tome što sam koncept na kome je baziran *JavaServer Faces* dozvoljava kreiranje korisnički prilagođenih komponenti korisničkog interfejsa koji se mogu dalje ponovo koristiti, što utiče na povećanje produktivnosti i omogućava doslednost prilikom izrade *Java WEB* aplikacija. Imajući u vidu sve prethodno navedeno, kao i to da se vremenom *JavaServer Faces* konstantno razvijao i unapređivao novim verzijama, da je deo *Java EE* platforme, mnogi vendori su implementirali *JavaServer Faces* specifikaciju i ponudili je širokoj programerskoj zajednici na korišćenje. Takođe, razvojem *JavaServer Faces* okvira pojavljivale su se različite bilbioteke komponenata sa različitim mogućnostima i karakteristikama. Upravo ova raznovrsnost implementacija *JavaServer Faces* specifikacije i pre svega biblioteka komponenata dovela je do toga da u širokoj ponudi treba napraviti razliku i utvrditi mogućnosti, prednosti i nedostatke jednih u odnosu na druge.

1 *Slika je preuzeta iz knjige JavaServer Faces in Action, Kito D. Mann*

2. IMPLEMENTACIJE JAVASERVER FACES SPECIFIKACIJE

JavaServer Faces možemo posmatrati i definisati i kao specifikaciju za izradu korisničkih interfejsa *WEB* aplikacija baziranih na komponentama, odnosno kao specifikaciju sa implementacijama koje nude različiti vendori.. Prva zvanična verzija *JavaServer Faces* specifikacije objavljena je marta 2004. godine. Podržavala je servlete (*v 2.3*) i bila je veoma čvrsto vezana za JSP, ali je takođe imala poprilično dosta grešaka i nedostataka, pa je ubrzo poboljšana novom verzijom 1.1 čija je specifikacija objavljena u maju iste godine i glavni cilj verzije 1.1 je upravo bilo ispravljanje uočenih grešaka i nedostataka. Kao takva usvojena je i uključena u *Java Enterprise Edition* platformu (verzija 1.4). Dve godine kasnije objavljena je verzija 1.2 koja donosi prva značajna poboljšanja. Pored značajnih poboljšanja u osnovi sistema i *API*-ju, od ove verzije *JavaServer Faces* nije više bio tako čvrsto vezan za *JavaServer Pages*. Upravo se zato kao najznačajnije poboljšanje u odnosu na prethodnu verziju izdvaja način obrade prikaza korisničkog intefejsa. On se više ne oslanja na *JavaServer Pages*, a takođe je omogućeno ugnježdavanje izvornih *HTML* tagova u *JavaServer Faces* stranicu. Ova verzija donosi i poboljšanu podršku *AJAX*-u, integraciju sa *JSTL* tagovima, vezu poruka sistema sa komponentama, uvodi korišćenje *XML* šeme za definisanje konfiguracionih fajlova umesto *DTD*-a, poboljšanja po pitanju bezbednosti kao i ispravku uočenih grešaka kao što je problem sa dugmetom za povratak kod aplikacija sa više prikazanih prozora. *JavaServer Faces* 2.0 predstavlja drugu bitnu verziju *JavaServer Faces* specifikacije i objavljena je u julu 2009. godine. U ovoj verziji načinjen je veliki broj tehničkih i funkcionalnih promena. Uvedene su anotacije, *Fa-*

celets tagovi, parametrizovani tipovi, podrška za *GET* zahtev, unapređena je primena *AJAX*-a. Takođe, uveden je novi opseg, odnosno opseg prikaza korisničkog interfejsa - *ViewScope*. Verzija *JavaServer Faces* 2.0 usvojena je i uključena u *Java Enterprise Edition* platformu verziju 6. Verzija *JavaServer Faces* 2.1 donosi manje promene u specifikaciji, dok trenutna verzija 2.2 uvodi nove koncepte kao što su *stateless view*, *faces flows,* podršku za *HTML5* i tako dalje. *JavaServer Faces* implementacije implementiraju *Java Server Faces API* specifikaciju i u najmanju ruku sadrže sve standardne komponente za prikaz osnovnih *HTML* elemenata. Postoje dve glavne *JavaServer Faces* implementacije koje se široko koriste i to su ***Oracle Mojarra*** *[Mojarra]* i ***Apache MyFaces*** *[Apache MyFaces]*. U tehničkom smislu ne postoje velike razlike između ove dve implementacije, jer se i jedna i druga pridržavaju *Java Server Faces API* specifikacije. Razlike su prisutne u robustnosti ovih implementacija, dostupnosti dokumentacije, nivou podrške, održavanju u vidu brzine poboljšanja, otklanjanju grešaka, objavljivanju novih verzija. Međutim, razlike u performansama između ove dve implementacije takođe postoje i one zavise od broja *JavaServer Faces* komponenata u stablu, od toga da li se stanje čuva na klijentskoj ili serverskoj strani i drugih parametara.

Apache MyFaces predstavlja implementaciju *JavaServer Faces* specifikacije dostupnu svima na korišćenje (*open source*). *Apache MyFaces* je razvijan pod okriljem *Apache Software Foundation*[2] i sastoji se od nekoliko podprojekata koji su u vezi sa *JavaServer Faces* tehnologijom. Inicijalno je razvijan kao implementacija *JavaServer Faces* specifikacije, a kasnije je proširen nizom biblioteka komponenata, da bi se zatim podelio na *Core* implementaciju, koja u suštini i predstavlja implementaciju *JavaServer Faces* specifikacije, i podprojekte usmerene na razvoj novih komponenata, proširenje funkcionalnosti postojećih, kao i rešavanje specifičnih potreba kao što je na primer povezivanje aplikacije sa bazom podataka. Dakle, *Apache MyFaces Core* predstavlja samu implementaciju *JavaServer Faces* specifikaciju i polaznu tačku na osnovu koje je započeo razvoj samog *Apache MyFaces* projekta. *Apache MyFaces Core* razvijao se u skladu sa *JavaServer Faces* specifikacijom, pa tako trenutno postoji pet verzija: *Core JSF-1.1, Core JSF-1.2, Core JSF-2.0, Core JSF-2.1* i *Core JSF-2.2* i svaka od verzije predstavlja implementaciju odgovarajuće *JavaServer Faces* specifikacije. *Apache MyFaces Core* sastoji se od dva podmodula:

- *API* podmodul implementira sve klase koje su definisane *JavaServer Faces* specifikacijom
- *Impl* podmodul sadrži klase koje su neophodne za rad sa *JavaServer Faces* okvirom, kao što su render klase za sve standardne *JavaServer Faces* komponente

Od verzije *Apache MyFaces Core 2.0,* osim navedena dva podmodula, uključeni su i drugi podmoduli koji omogućavaju uspešnu integraciju *Apache MyFaces Core* implementacije i ostalih *Apache MyFaces* projekata:

- Interni podmodul koji sadrži klase koje su zajedničke za druge *Apache MyFaces* projekte kao što su *Trinidad*[3], *Orchestra*[4] i tako dalje.
- Podmodul koji sadrži paket koji uključuje biblioteke kojim su definisani *API* podmodul i *Impl* podmodul u jedan *.jar* dokument u kome se nalazi i *OSGi MANIFEST.MF.*

Ponašanje *Apache MyFaces* implementacije može se definisati na osnovu dostupnih parametara konteksta od strane same implementacije, a koji se mogu naći na zvaničnom *Apache MyFaces* sajtu. Prefiks parametara konteksta *Apache MyFaces Core* implementacije je „*org.apache.myfaces*“, a zatim sledi naziv parametra. Generalno gledano, ovi parametri podeljeni su u nekoliko grupa, odnosno u konfiguracione parametre, parametre za podešavanje *Expression Language,* renderovanje resurasa, čuvanje stanja, konverziju i validaciju, obradu prikaza i druge. U dokument *web.xml JavaServer Faces* projekta dodaju se definisanjem elemenata `<param-name>` i `<param-value>` koji predstavljaju naziv i vrednost parametara, unutar elementa `<context-param>`:

```
<context-param>
  <param-name>org.apache.myfaces.SOME_ PARAM</param-name>
  <param-value>someValue</param-value>
</context-param>
```

Oracle Mojarra implementacija prvenstveno je razvijana kao *Sun JavaServer Faces Reference Implementation,* odnosno *Sun JSF RI* i to ime nosila je sve do verzije *1.2*. Sa obzirom da se *Sun JSF RI* implementacija razvijala pod okriljem *Glassfish* projekta, od pomenute verzije naziv implementacije promenjen je u skladu sa nazivom projekta pod čijim okriljem je razvijana, odnosno u *Mojarra* (vrsta ribe). Kasnije, kada je *Sun* preuzet od strane *Oracle,* naziv *Sun Mojarra* promenjen je u sadašnji *Oracle Mojarra.* Kao i prethodno opisana implementacija, *Oracle Mojarra* implementacija razvijala se u skladu sa razvojem i objavljivanjem novih *JavaServer Faces* specifikacija, tako da je trenutna verzija *Oracle Mojarra* implementacije *2.2,* a pored nje postoje još i verzije *Oracle Mojarra 1.1, Oracle Mojarra 1.2, Oracle Mojarra 2.0* i *Oracle Mojarra 2.1. Oracle Mojarra* implementacija dostupna je u okviru biblioteke sa *.jar* ekstenzijom *javax.faces.jar,* ili kao dve biblioteke sa *.jar* ekstenzijom u kojima je odvojen *API* i implementacija, *jsf-api.jar* i *jsf-impl.jar.* Ove biblioteke potrebno je uključiti u *JavaServer Faces* projekat ukoliko aplikacioni server nije isporučen sa *Oracle Mojarra* implementacijom. Prefiks parametara konteksta *Oracle Mojarra* implementacije je „*com.sun.faces*“, i oni mogu biti podeljeni u grupe parametara konteksta koji su zaduženi za definisanje i podešavanje obrade resurasa, čuvanja stanja, konverzacije i validacije i drugo. Kao i kod *Apache MyFaces* implementacije, parametri konteksta *Oracle Mojarra* implementacije dodaju se u dokumentu *web.xml* definisanjem elemenata `<param-name>` i `<param-value>` koji predstavljaju naziv i vrednost parametara, unutar elementa `<context-param>`:

2 https://www.apache.org/foundation/

3 https://myfaces.apache.org/trinidad/

4 http://myfaces.apache.org/orchestra/

```
<context-param>
  <param-name>com.sun.faces.SOME_      PARAM</param-name>
  <param-value>someValue</param-value>
</context-param>
```

2.1. *Komparativna analiza i poređenje perfomansi*

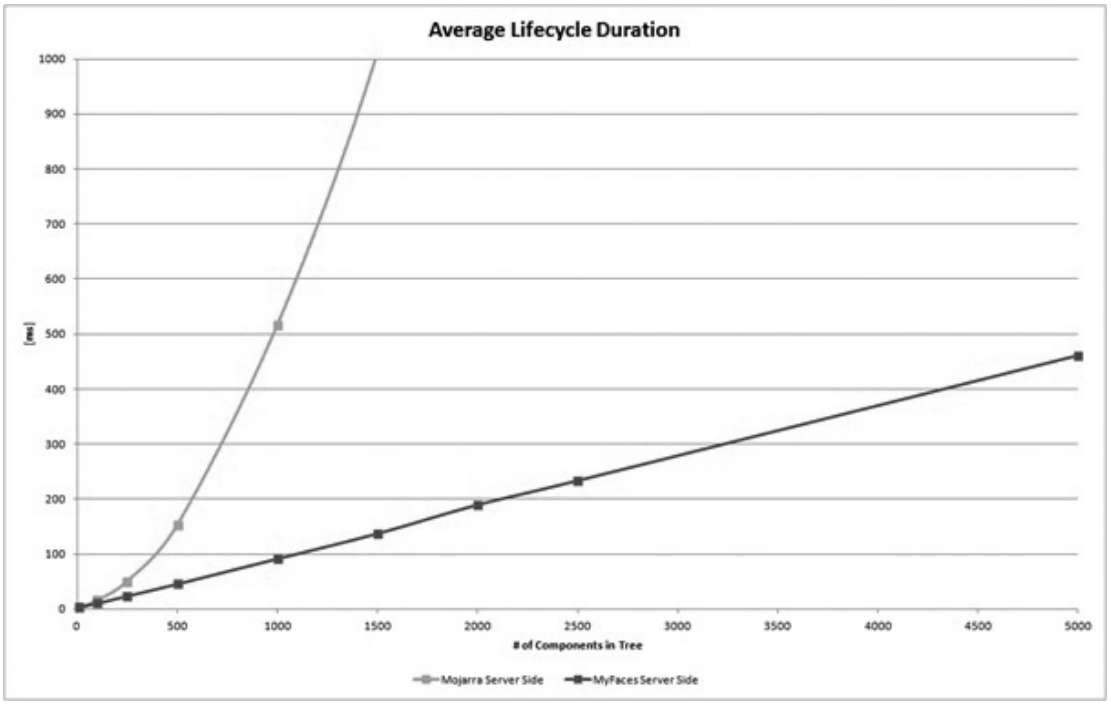
Razlike u performansama između *Apache MyFaces* i *Oracle Mojarra* implementacija postoje i one mogu zavisiti od broja *JavaServer Faces* komponenata u stablu, kao i od toga da li se stanje čuva na klijentskoj ili serverskoj strani. Naredni deo ima zadatak da odgovori na pitanje kakav uticaj ima veličina *JavaServer Faces* stabla komponenata i odabrani metod čuvanja stanja na performanse same *JavaServer Faces* aplikacije u zavisnosti od implementacije. U tu svrhu, vreme izvršenja životnog ciklusa mereno je za različit broj komponenata u stablu i metod čuvanja stanja, za svaku od ove dve implementacije, i na taj način dobijeni rezultati interpretirani su u svrhu poređena ponašanja i performansi ovih implementacija za svaki skup jednakog broja komponenata u stablu. Dakle, veličine koje je potrebno izmeriti za svaku od dve implamentacije određene su metodama čuvanja stanja prikaza sa različitim brojem komponenata u stablu za svaki od metoda čuvanja stanja prikaza koji iznosi 10, 100, 250, 500, 1000, 2000, 2500 i 5000. Očekuje se da će različite metode čuvanja stanja prikaza imati uticaj na vreme izvršenja životnog ciklusa. Takođe se očekuje da će uticaj na vreme izvršavanja životnog ciklusa imati i broj komponenata u stablu komponenata, odnosno, da će sa povećanjem broja komponenata u stablu rasti i vreme izvršavanja životnog ciklusa. Kako i *Oracle Mojarra* i *Apache MyFaces* implementiraju istu specifikaciju, očekuje se da će se obe implementacije ponašati slično pod istim uslovima. Kako bi se sagledao uticaj navedenih parametara na *JavaServer Faces* aplikaciju i njene implementacije, uslovi pod kojima je vršeno testiranje morali su biti u skladu sa sledećim zahtevima:

- Testiranje mora biti zasnovano na istom kodu, odnosno istoj aplikaciji
- Zamena *JavaServer Faces* implementacija ne sme uticati na kod test aplikacije
- Merenje mora biti zasnovano na različitom broju komponenata u stablu
- Merenja se moraju obaviti za svaki od metoda čuvanja stanja prikaza
- Sva merenja moraju biti uporediva

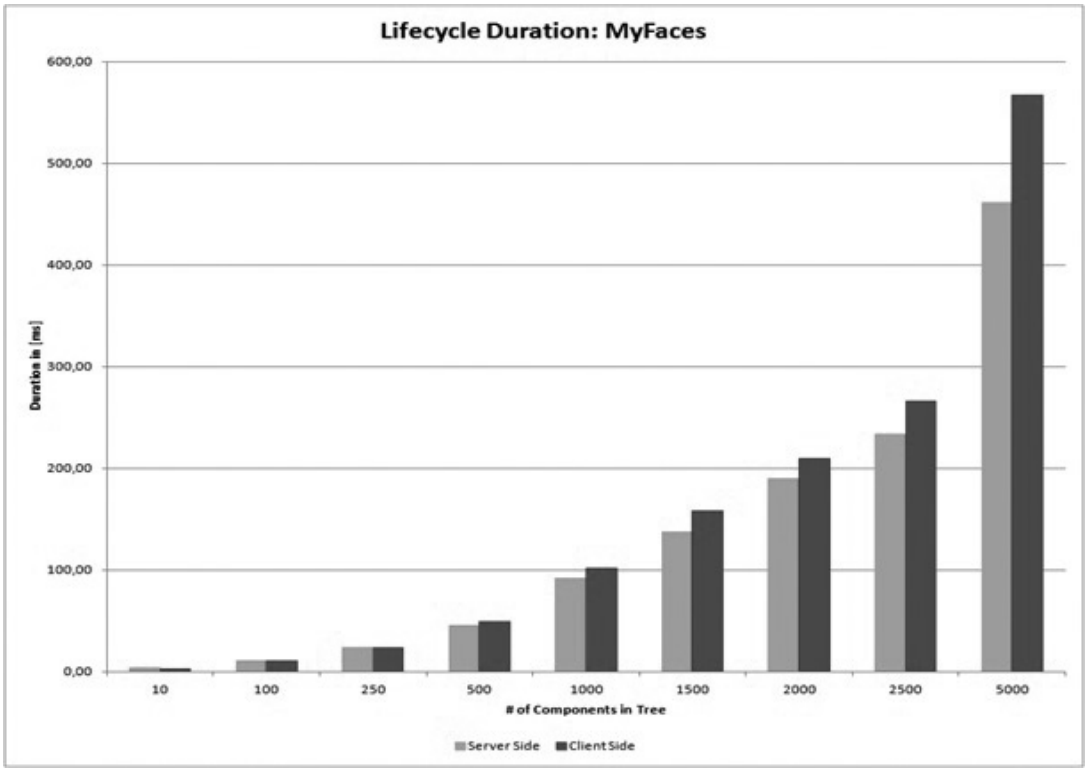
Kako bi se omogućilo direktno poređenje između implementacija i individualnih konfiguracija, definisane su sledeće varijacije:

- Različite implementacije: *Oracle Mojarra* i *Apache MyFaces*
- Različiti metodi čuvanja stanja: čuvanje stanja prikaza na klijentskoj i čuvanje stanja prikaza na serverskoj strani
- Različit broj komponenata u stablu komponenata: 10, 100, 250, 500, 1000, 2000, 2500 i 5000

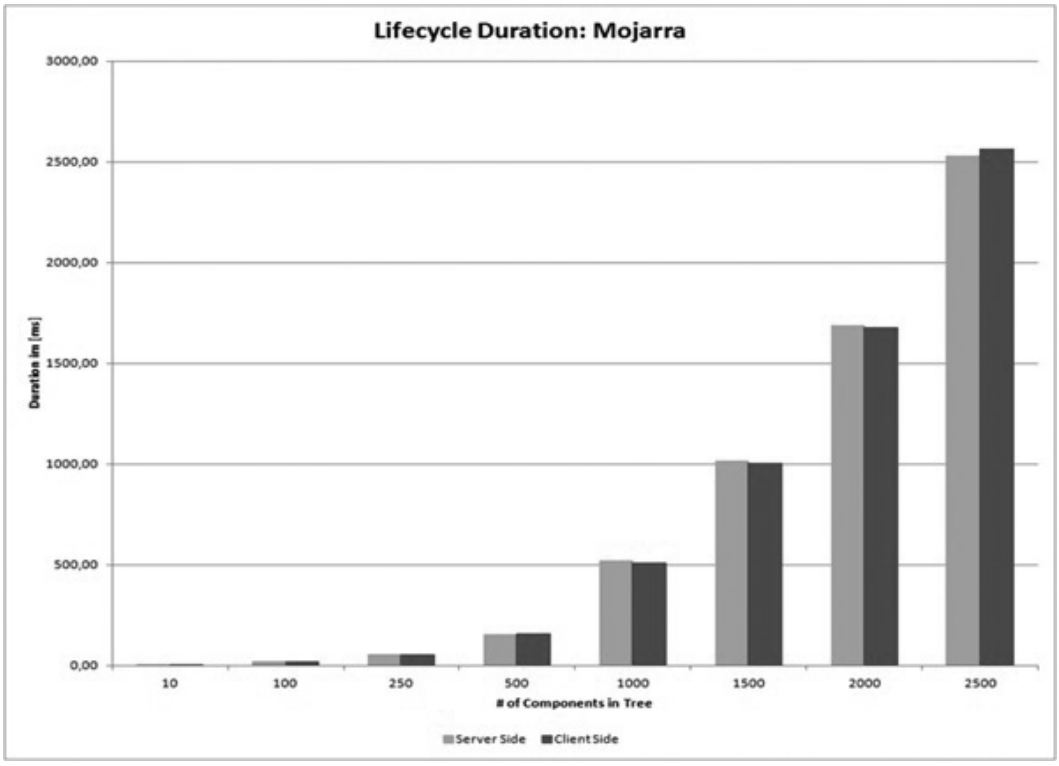
Kako bi rezultati bili reprezentativni, merenja su se ponavljala po nekoliko puta, a zatim je kao krajnji rezultat uzimana njihova srednja vrednost. Od alata korišćen je *Apache JMeter,* pomoću koga su simulirani zahtevi od strane klijenta. Vremena izvršenja *JavaServer Faces* životnih ciklusa merena su pomoću *JProfiler* alata.



Slika 2: *Trajanje JavaServer Faces životnog cikusa u zavisnosti od broja komponenata u stablu* (Slika je preuzeta sa http://blog.oio.de)



Slika 3: *Trajanje JavaServer Faces životnog cikusa u zavisnosti od broja komponenata u stablu i metoda čuvanja stanja za Apache MyFaces implementaciju* (Slika je preuzeta sa http://blog.oio.de)



Slika 4: *Trajanje JavaServer Faces životnog cikusa u zavisnosti od broja komponenata u stablu i metoda čuvanja stanja za Oracle Mojarra implementaciju* (Slika je preuzeta sa http://blog.oio.de)

Nasuprot očekivanjima da će se obe implementacije isto ponašati u zavisnosti od broja komponenta u stablu, vreme izvršenja životnog ciklusa razlikovalo je se u značajnoj meri, kao što je prikazano na grafiku niže. U oba slučaja stanje je čuvano na serverskoj strani.

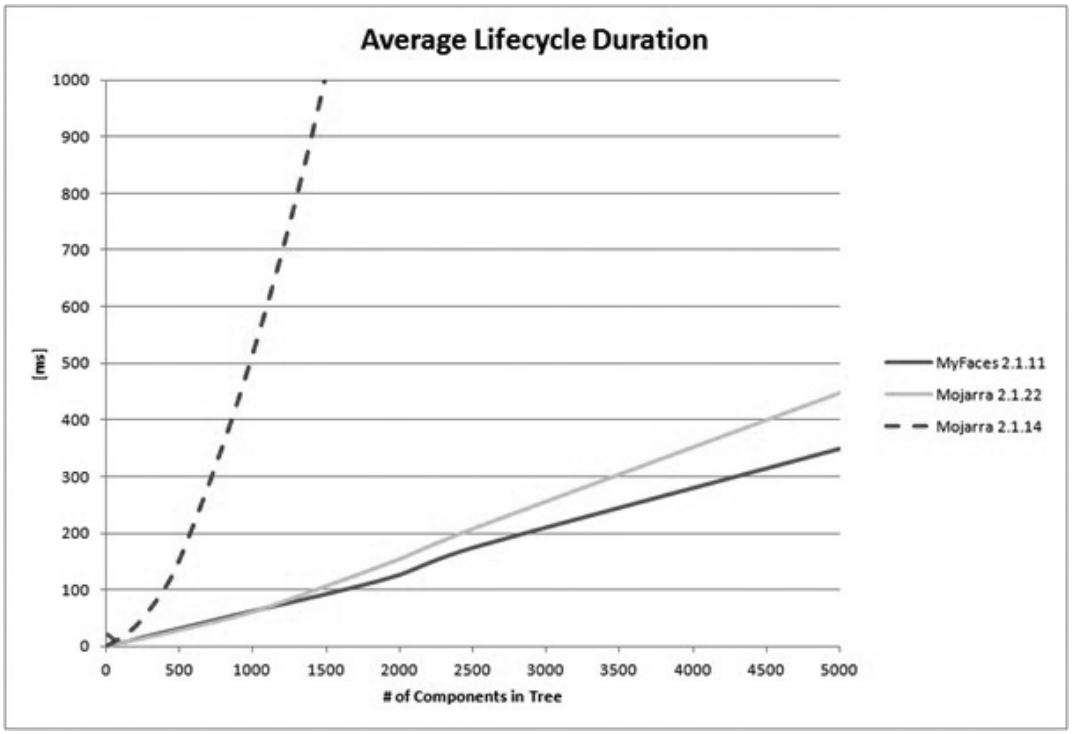
Dok trajanje životnog ciklusa kod *Apache MyFaces* implementacije linearno raste u zavisnosti od broja komponenata, kod *Oracle Mojarra* implementacije trajanje životnog ciklusa raste eksponencijalno u zavisnosti od broja komponenata i ova implementacija se pokazala kao značajno sporija u odnosu na *Apache MyFaces*. Za aplikacije sa velikim brojem komponenata, sa ove tačke gledišta, preporučljivije je koristiti *Apache MyFaces* implementaciju. Ponašanje implementacija u zavisnosti od metoda čuvanja stanja prikazano je na sledećim dijagramima.

Kao što je i očekivano, vreme izvršavanja životnog ciklusa veće je kada se stanje čuva na klijentskoj strani nego kada se stanje prikaza čuva na serverskoj strani. Ovo je posledica činjenice da je prilikom čuvanja stanja na klijentskoj strani potrebno dodatno vreme da se stanje na klijentu serijalizuje i enkodira, a zatim na serveru deserijalizuje i dekodira.

Za razliku od *Apache MyFaces* implementacije, *Oracle Mojarra* implementacija ponaša se gotovo isto bez obzira koji metod čuvanja stanja je odabran. Međutim, razlika u trajanju životnog ciklusa između *JavaServer Faces* implementacija *Oracle Mojarra* i *Apache MyFaces* je i dalje ogromna bez obzira koji metod za čuvanje stanja prikaza je odabran. *Apache MyFaces* implementaciji je bilo potrebno *255,7 ms* za izvršavanje životnog ciklusa kada se u stablu nalazilo 2500 komponenti, a stanje je čuvano na serverskoj strani, dok je sa istim parametrima *Oracle Mojarra* implementaciji bilo potrebno skoro deset puta više vremena, odnosno *2527,94 ms*. Na osnovu izloženih podataka možemo zaključiti da:

- Broj komponenti u stablu ima direktan uticaj na performanse same aplikacije koje se ogledaju u vremenu izvršavanja životnog ciklusa *JavaServer Faces* aplikacije
- *Apache MyFaces* implementacija obrađuje *JavaServer Faces* životni ciklus znatno brže nego *Oracle Mojarra* implementacija, a ova razlika je značajno primetna kada se u stablu komponenata nalazi veliki broj komponenata
- Linearni rast vremena izvršenja *JavaServer Faces* životnog ciklusa omogućava pouzdanije skaliranje očekivanog vremena izvršenja *JavaServer Faces* životnog ciklusa u zavisnosti od broja komponenata
- Uzimajući sve u obzir, *Apache MyFaces* implementacija u svim aspektima pružila je ekvivalentne ili bolje perfomanse od *Oracle Mojarra* implementacije

Oracle Mojarra predstavlja podrazumevanu *JavaServer Faces* implementaciju koja se isporučuje sa mnogim serverima, kao što je to slučaj sa *Glassfish* serverom, i veoma se često koristi prilikom razvoja *JavaServer Faces* aplikacija. Imajući u vidu prethodne testove, ona se pokazala kao znatno sporija od *Apache MyFaces* implementacije *JavaServer Faces* specifikacije, što je bio povod za poboljšanje *Oracle Mojara* implementacije i redukciju vremena potrebnog za izvršavanje *JavaServer Faces* životnog ciklusa, pa samim tim, nove verzije su donosile i nova unapređenja, a od *Oracle Mojarra* verzije *2.1.22*, vreme izvršavanja životnog ciklusa je značajno smanjeno i približno jedanko vremenu izvršavanja životnog ciklusa *Apache MyFaces* implementacije, kao što je prikazano niže na grafiku.



Slika 5: *Trajanje životnog ciklusa kod unapređene verzije Oracle Mojarra implementacije* (Slika je preuzeta sa http://blog.oio.de)

Kao što se može videti iz priloženog grafika, vreme izvršavanja životnog ciklusa *Oracle Mojarra* implementacije jednako je, pa čak i manje od vremena izvršavanja životnog ciklusa *Apache MyFaces* implementacije kada se prikaz sastoji od stabla komponenata koje ne broji više od 1200 komponenti. Tek nakon 1200 komponenti u stablu, *Apache MyFaces* preuzima prednost u odnosu na *Oracle Mojarra* implementaciju u vidu vremena potrebnog za izvršavanje *JavaServer Faces* životnog ciklusa, ali ta prednost nije više toliko značajna. Međutim, bitno je da sada vreme izvršavanja *JavaServer Faces* životnog ciklusa kod *Oracle Mojarra* implementacije ne raste eksponencijalno u zavisnosti od broja komponenti koje sadrži određeni prikaz, već linearno. Ako uzmemo u obzir osobinu *Oracle Mojarra* implementacije da je vreme izvršenja *JavaServer Faces* životnog ciklusa skoro identično bez obzira koji je metod čuvanja stanja odabran, svaka verzija *Oracle Mojarra* implementacije počev od pomenute verzije *2.1.22* preporučljivija je prilikom razvoja *JavaServer Faces* aplikacija sa relativno malim brojem komponenata, odnosno do 1200 komponenata, bez obzira koji je metod čuvanja stanja odabran.

Ako sagledamo praktičnu primenu pomenutih implementacija, iako i *Oracle Mojarra* i *Apache MyFaces* implementacija prate *JavaServer Faces* specifikaciju, postoje određene razlike prilikom korišćenja ove dve implementacije. Pre svega, iako obe implementiraju iste metode, često se dešava da metode koje se koriste u ovim implementacijama nemaju iste nazive svojih parametara, pa tako na primer metoda `validate` *JavaServer Faces* validatora *Oracle Mojarra* implementacije ima sledeći potpis:

```
public void validate(FacesContext context,
       UIComponent component,
       Object value) throws ValidatorException
```

, dok ista metoda kod *Apache MyFaces* ima sledeći potpis:

```
public void validate(FacesContext fc,
       UIComponent uic, Object o)
       throws ValidatorException
```

Dalje, *Apache MyFaces* implementacija *JavaServer Faces* specifikacije nije se pokazala kompatibilna prilikom korišćenja *CDI*[5] *Bean*-ova i anotacija, pa samim tim, prilikom implementacije poslovne logike, izbor je spao na *Managed Bean*-ove. Takođe, ove dve implementacije često koriste i različite pomoćne biblioteke i klase za implementaciju sličnih ili istih funkcionalnosti. Takođe, svi *Managed Bean*-ovi kod *Apache MyFaces* implementacije u obavezi su da implementiraju *Serializable* interfejs, dok kod *Oracle Mojarra* implementacije to nije slučaj. Validacija polja takođe je ispoljila različito ponašanje kod ove dve implementacije. Razlike između ove dve implementacije ogledaju se i u dostupnim parametrima konteksta. Ipak, i pored određenih razlika, uzimajući sve u obzir može se reći da su obe implementacije u skladu sa *JavaServer Faces* specifikacijom uz određene varijacije. Međutim, baš upravo zbog tih razlika, migracije projekata, naročito onih kompleksnih, sa jednu na drugu implementaciju nije tako lako izvesti, pa prilikom razvoja *JavaServer Faces* aplikacija preporučljivo je odabrati jednu implementaciju i držati se nje, a samtramo da je najbolje koristiti onu implementaciju koja je sadržana u *Servlet* kontejneru koji se koristi. Ako sagledamo *HTML markup* koji generišu navedene *JavaServer Faces* implementacije, uočeno je da ne postoje velike razlike između njih, već da obe generišu *HTML markup* za komponente i delove stranica na isti način.

3. JAVASERVER FACES BIBLIOTEKE KOMPONENATA

Sa druge strane, *JavaServer Faces[Bergsten, 2004], [Mann, 2005], [Geary& Horstmann, 2010]* predstavlja okvir za izradu korisničkog interfejsa *WEB* aplikacija baziranih na *Java* tehnologiji i deo je *Java EE* platforme. Svojim karakteristikama i mogućnostima primamio je mnoge velike vendore da izvrše implementaciju *Java Server Faces* specifikacije, a programerskim zajednicama i drugim zainteresovanim stranama da prošire i nadograde postojeće implementacije nadogradnjom, proširivanjem i stvaranjem novih komponenata korisničkog interfejsa čime su nastajale *JavaServer Faces* biblioteke komponenata koje su danas široko dostupne na korišćenje. Dakle, *JavaServer Faces* biblioteke komponenata se nadovezuju na osnovnu implementaciju i nadograđuju je poboljšanim *AJAX* funkcionalnostima, dodavanjem novih šablona, proširivanjem komponenata itd. Trenutno postoji veliki broj dostupnih *JavaServer Faces* biblioteka komponenata. Neke od njih sadrže veće mogućnosti za izradu šablona i stilizovanje, neke imaju izraženije *AJAX* funkcionalnosti a neke su bogatije komponentama. Najkorišćenije biblioteke komponenata su ***PrimeFaces***[6]***, RichFaces*** *[Lee, 2008]*[7] i ***ICEFaces*** *[Eschen, 2008]*[8]. Sve navedene biblioteke komponenata kroz uspešan razvoj i unapređivanje dostigle su svoju zrelost, i kao posledica toga danas su uspešno usvojene i široko se koriste prilikom izrade *JavaServer Faces WEB* aplikacija. Međutim, nisu sve biblioteke istih karakteristika, niti pružaju iste mogućnosti. Navedene biblioteke komponenata biće sagledane sa više aspekata kako bi se uočile njihove prednosti i mane u odnosu na druge i po čemu se to one izdvajaju.

3.1. *Komparativna analiza najčešće upotrebljavanih biblioteka komponenata*

Što se tiče dostupnosti komponenata, *RichFaces* biblioteka komponenata sadrži 39 osnovnih komponenata koje su podeljene u određene grupe komponenata. Broj novih komponenata nije rastao značajno sa objavljivanjem novih verzija, pa se može zaključiti da je akcenat prilikom razvoja *RichFaces* biblioteke uglavnom bio na nadogradnji i proširivanju postojećih. Ipak, *RichFaces* se isporučuje sa *Component Develop-*

5 *Contexts and Dependency Injection Bean - fleksibilniji model Managed Bean-a kojim se upravlja od strane aplikacionog servera*

6 *http://primefaces.org*

7 *http://richfaces.jboss.org*

8 *http://www.icesoft.org*

ment Kit, vrstom alata za razvoj novih korisnički definisanih komponenata sa ugrađenom *AJAX* podrškom, što bi trebalo da kompezuje nedostatak ugrađenih komponenti koje sadrži *RichFaces* biblioteka komponenata. Ipak, ostaje otvoreno pitanje koliko je korisnik u skladu sa svojim programerskim znanjem i poznavanjem *JavaServer Faces* tehnologije u stanju da koristi ovaj alat za razvoj svojih komponenti. *ICEFaces* biblioteka komponenata sadrži oko 70 komponenata u svojoj *ICE* biblioteci, kao i preko 40 komponenata u svojoj *ACE* bibloteci, koje predstavljaju komponente kreirane i dizajnirane za razvoj *WEB* aplikacija novije generacije. Ove komponente koriste spoj tehnika za renderovanje na klijentskoj i serverskoj strani kako bi se korisnicima obezbedio bogat korisnički interfejs uz brzu obradu poslatih zahteva. *PrimeFaces* biblioteka komponenata sadrži najveći skup dostupnih komponenata, čak 117. Pored osnovnih komponenata (komponente za unos korisničkih ulaza, dugmad, veze i slično), dostupan je i veliki broj naprednih komponenata, kao što su *HTMLEditor,* komponente za izradu i prikaz grafika i tako dalje. *PrimeFaces* biblioteka komponenata u pozadini koristi *JQuery,* iskorišćavajući sve njegove dobre osobine u vidu *widget*-a, dodataka, *AJAX* interakcije. Kompatibilnost između komponenti je jako dobra, a komponente je veoma lako stilizovati i menjati im izgled. *PrimeFaces* biblioteka komponenata pruža preko 25 ugrađenih tema za stilizovanje komponenti.

Drugi veoma bitan aspekt jeste lakoća korišćenja biblioteka komponenata i uključivanje istih u projekat. Često nepravilno konfigurisanje biblioteka komponenata može da prouzrokuje greške u radu, stoga je neophodno voditi računa kako i na koji način se biblioteke komponenata uključuju u projekat. *RichFaces* biblioteka komponenata ne sadrži nikakv tutorijal, odnosno konkretna uputstva kako ovu biblioteku integrisati sa razvojnim okruženjem. Neki zaključci mogu se izvući iz zvanične dokumentacije, koja je uglavnom fokusirana na *Maven*-u, koji predstavlja široko korišćen standard za uključivanje biblioteka i dokumenata treće strane u projekat, ali nije jedini. Uključivanje *RichFaces* biblioteke komponenata u projekat zahteva osnovnu biblioteku, *UI* biblioteku i još tri obavezne biblioteke. Takođe, postoje i opcioni dokumenti u vidu biblioteka sa *.jar* ekstenzijom koji se mogu uključiti u projekat, koje se odnose na *Bean* validaciju, *JMS*[9] i *CDI* integraciju. *ICEFaces* biblioteka komponenata takođe ne sadrži nikakva uputstva za uključivanje iste u projekat. Uključivanje biblioteke u projekat svodi se na dodavanje dokumenata u vidu biblioteka sa *.jar* ekstenzijom koje sadrže skup željenih komponenti kao i određenih dodatnih dokumenata u vidu biblioteka sa *.jar* ekstenzijom u cilju boljeg procesuiranja i obrade dodatih komponenata. *PrimeFaces* biblioteka komponenata sadrži jasna uputstva kako je uključiti u projekat. Sve što je potrebno uraditi jeste preuzimanje sa zvaničnog *PrimeFaces* sajta odgovarajućeg dokumenta u vidu biblioteke sa *.jar* ekstenzijom i dodavanje u projekat. Neka razvojna okruženja čak dolaze sa integrisanom *PrimeFaces* bibliotekom, kao što je *NetBeans*, pa je prilikom kreiranja projekta potrebno samo odabrati ovu biblioteku.

9 *Java Message Service*

Dokumentacija predstavlja takođe veoma bitan aspekt. RichFaces biblioteka komponenata pruža online korisnički vodič koji se ažurira sa svakom novom verzijom zadržavajući istu formu. Međutim, bliža uputstva o tome kako se kreira aplikacija i koriste RichFaces komponente ne postoje. ICEFaces biblioteka komponenata pruža jako dobru i sveobuhvatnu dokumentaciju koja uključuje veliki broj primera, uputstava, pa čak i u video formi. Sa druge strane, za pristup odeđenim materijalima potrebno je registrovati se na njihovom zvaničnom sajtu. PrimeFaces biblioteka komponenata pruža najpragmatičniji pristup dokumentovanju, jer uputstvo za korišćenje ove biblioteke ujedno predstavlja i knjigu o PrimeFaces biblioteci komponenata i integrisano je sa svim primerima na njihovom zvaničnom sajtu, tako da je na jednom mesto dostupno uputstvo za korišćenje određene PrimeFaces komponente, dokumentacija, odnosno njen deo koji se odnosi na datu komponentu, kao i određeni dodatni resursi ukoliko postoje.

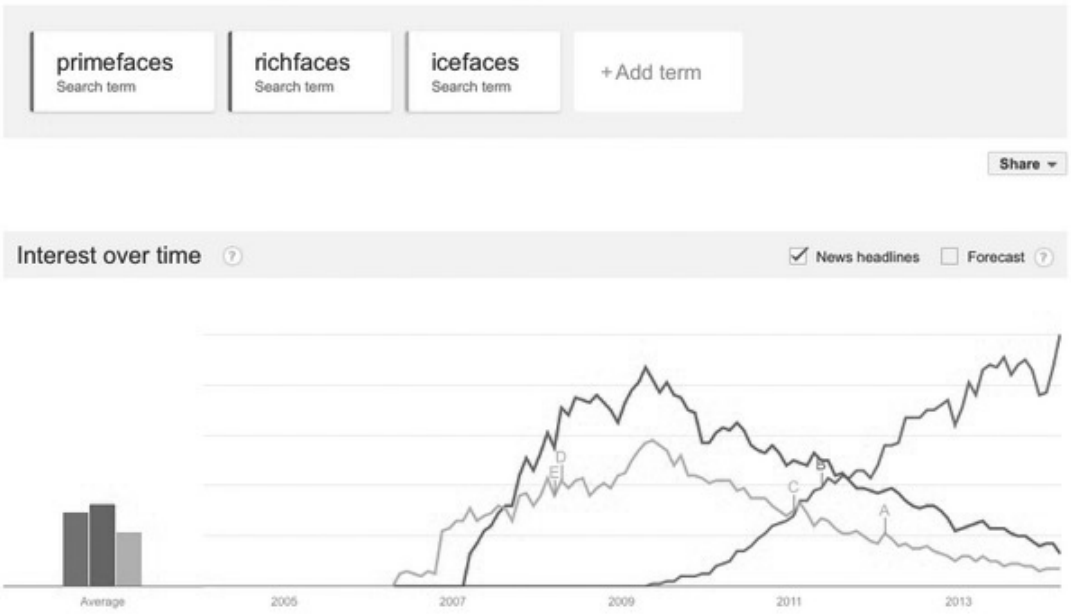
Korišćenjem navedenih biblioteka komponenata uočene su mnoge greške i nedostaci, ali i potrebe za novim kompenentama, koje predstavljaju otvorena pitanja svake od biblioteke komponenata. Ova otvorena pitanja nije jednostavno uporediti, jer ona nisu ista ni za jednu biblioteku komponenata, niti se rešavaju istom brzinom. Međutim, sva otvorena pitanja imaju svoj prioritet, a veći broj otvorenih pitanja najvećeg prioriteta ukazuje da se greške otklanjaju nedovoljnom brzinom, odnosno da postoji više bitnijih propusta. Broj otvorenih pitanja najveći je u *RichFaces* biblioteci komponenata, a 64% procenata označena su kao najvišeg prioriteta. Sa druge strane, *ICEFaces* biblioteka ima nešto manje otvorenih pitanja, ali su čak 93% označena kao otvorena pitanja najvišeg prioriteta. *PrimeFaces* biblioteka komponenata vodi svoja otvorena pitanja pomoću *google* koda, i njihov broj je znatno manji nego kod prethodne dve navedene biblioteke komponenata, oko 128 u odnosu na hiljade otvorenih pitanja kod *RichFaces* i *ICEFaces* biblioteka komponenata.

U cilju upoređivanja perfomasni, sprovedeno je testiranje u kome je izvršeno slanje forme na server koja se sastoji od jedne tabele od 100 redova i 5 kolona sa ćelijama popunjenim nasumično generisanim tekstulanim zapisom za svaku od navedene tri biblioteke komponenti. Merenja su izvršena pomoću *Apache ab stress tool* alata, a na osnovu 5000 poslatih zahteva, dobijeni su rezultati prikazani niže u tabeli.

Tabela 1: *Performanse JavaServer Faces biblioteka komponenata*

	RichFaces	ICEFaces	PrimeFaces
Veličina dokumenta (u bajtovima)	47442 bytes	76072 bytes	53904 bytes
Broj zahteva u sekundi (srednja vrednost)	28.59	11.12	33.46
Vreme potrebno za obradu jednog zahteva (milisekunde)	174.869 ms	899.281 ms	149.422 ms
Brzina prenosa (Kbytes/sec)	1329.59 Kbytes/sec	829.41 Kbytes/sec	1767.06 Kbytes/sec

Iz tabele možemo videti da je, po gotovo svim parametrima, najbolje rezultate postigla *PrimeFaces* biblioteka komponenata, zatim *RichFaces* biblioteka komponenata, a kao



Slika 6: *Trend korišćenja JavaServer Faces biblioteka komponenata*

najlošija po performansama pokazala se *ICEFaces* biblioteka komponenata, uglavnom zbog toga što je korišćenjem ove biblioteke generisana najveća *HTML* stranica.

Ako sagledamo trend korišćenja ovih biblioteka, *RichFaces* i *ICEFaces* biblioteka komponenata intezivno su korišćenje od 2006. godine do 2009. godine, dok se pojavom *PrimeFaces* bibilioteke komponenata 2009. godine indeks upotrebe prethodno navedene dve biblioteke konstantno nalazi u padu, a danas *PrimeFaces* biblioteka komponenata daleko je u prednosti u odnosu na svoje konkurente, a indeks upotrebe iste i dalje raste.

Ako sagledamo praktičnu primenu pomenutih biblioteka komponenata, upravo jednostavnost *PrimeFaces* biblioteke komponenata doprinela je da se komponente ove biblioteke savršeno uklapaju sa bilo kojom od implementacija *JavaServer Faces* specifikacije i ponašaju u skladu sa definisanom specifikacijom. Komponente ove biblioteke u potpunosti su se ponašale na očekivan način, a tokom rada nisu proizvodile nikakve nusefekte. Veliki izbor dostupnih komponenata *PrimeFaces* biblioteke učinio je razvoj aplikacije još lakšim, a dobro dokumentovani primeri značajno su olakšali implementaciju ovih komponenata u projekat. Korišćenjem *PrimeFaces* biblioteke komponenata gotovo da nema slučaja korišćenja koji se ne može pokriti komponentama iz ove biblioteke. Još jedna velika prednost ove biblioteke je ta što pruža jako veliku fleksibilnost svojih komponenata, pa se takve komponente mogu u velikoj meri na različite načine prilagođavati aplikaciji. Korišćenje komponenata *PrimeFaces* biblioteke u kombinaciji sa standardnim *JavaServer Faces* komponentama ne predstavlja nikakav problem, pa tako ugnježdavanje i uporedno korišćenje *PrimeFaces* i standardnih *JavaServer Faces* komponenata neće dovesti do nikakvih disfunkcionalnosti, već će samo pomoći u bržem i efikasnijem razvoju aplikacije. *RichFaces* biblioteka komponenata poseduje relativno mali broj komponenata koji ne predstavlja toliko značajno i bitno poboljšanje u odnosu na standardni *JavaServer Faces* skup komponenata. Komponente za prihvatanje korisničkih ulaza su prisutne u malom broju varijacija, što za aplikacije koje rade sa većim brojem složenih podataka može predstavljati problem. Napredne komponente prisutne su jako malom broju, i uglavnom se odnose na rad sa podacima, kao što su tabele, liste i drugo, izradu stabala i *drag and drop* komponente. Uputstva za korišćenje komponenti postoje, ali su često površna i uglavnom obuhvataju uputstva za implementaciju komponenti na samoj stranici, bez ikakvih dodatnih kodova logike koja se izvršava u pozadini. U toku rada sa ovim komponentama primećeno je da se često nisu ponašale u skladu sa *JavaServer Faces* specifikacijom, a implementacija *AJAX*-a predstavljala je bitan problem. Integracija sa *JavaServer Faces* ugrađenim komponentama takođe je predstavljala problem, pa dolazimo do zaključka da komponente *RichFaces* biblioteke nisu dizajnirane za korišćenje u kombinaciji sa drugim komponentama, već je za njihovu implementaciju preporučljivo koristiti komponente iz same *RichFaces* biblioteke. *ICEFaces* biblioteka komponenata, kao i *PrimeFaces* biblioteka, sadrži relativno veliki broj komponenata podeljenih u *ICEFaces Advanced (ACE)* i *ICEFaces ICE* komponente. Za svaku komponentu ponuđeno je uputstvo za korišćenje, kao i kod poslovne logike koji je implementiran u pozadini na jedan vrlo jasan i koncizan način. Takođe, za svaku komponentu, pored uputstva, dostupan je direktno i deo dokumentacije koji se odnosti na datu komponentu. Moglo bi se reći da su broj i mogućnosti *ICEFaces* komponenti zadovoljavajući i za izradu kompleksnijih *JavaServer Faces* aplikacija, a lakši rad sa njima omogućavaju i jako dobro definisana i struktuirana uputstva. Međutim, kada je reč o njihovoj primeni u praksi, situacija je nešto drugačija. *ICEFaces* komponente nisu se često ponašale u skladu sa uputstvima, dokumentacijom i očekivanim ponašanjem definisanim *JavaServer Faces* specifikacijom.

Tokom razvoja aplikacije javljale su se razne prepreke u njihovom korišćenju i često su se pokazale kao nekompatibilne sa standardnim *JavaServer Faces* komponentama, ali i sa drugim komponentama iz svoje biblioteke. Upotreba ovih komponenti često je dovodila do određenih nusefekata i disfunkcionalnosti, pa je samim tim razvoj aplikacije korišćenjem komponenata iz *ICEFaces* biblioteke bio prilično otežan, a pun potencijal ovih komponenata nije mogao biti iskorišćen u punoj meri.

4. ZAKLJUČAK

JavaServer Faces danas predstavlja široko upotrebljavan i jako popularan okvir za razvoj *Java WEB* aplikacija i kao takav usvojen je kao sastavni deo *Java EE* platforme od 2003. godine. Komponente korisničkog interfejsa predstavljaju samu suštinu *JavaServer Faces*, a najveća prednost *JavaServer Faces* ogleda se u tome što sam koncept na kome je *JavaServer Faces* baziran dozvoljava kreiranje korisnički prilagođenih komponenti koji se mogu dalje ponovo koristiti, što utiče na povećanje produktivnosti i omogućava doslednost prilikom izrade *Java WEB* aplikacija. Mnogi vendori su implementirali *JavaServer Faces* specifikaciju i ponudili je širokoj programerskoj zajednici na korišćenje. Takođe, razvojem *JavaServer Faces* okvira pojavljivale su se različite bilbioteke komponenata sa različitim mogućnostima i karakteristikama. Upravo ova raznovrsnost implementacija *JavaServer Faces* specifikacije i biblioteka komponenata dovela je do toga da u širokoj ponudi treba napraviti razliku i utvrditi mogućnosti, prednosti i nedostatke jednih u odnosu na druge. Postoje dve glavne *JavaServer Faces* implementacije koje se široko koriste - *Apache MyFaces* i *Oracle Mojarra*. U tehničkom smislu ne postoje velike razlike između ove dve implementacije, međutim, razlike su prisutne u robustnosti, preformansama, dostupnosti dokumentacije i tako dalje. Perfomanse su u početku bile na strani *Apache MyFaces* implementacije, međutim, razvojem i unapređivanjem *Oracle Mojarra* implementacije, od verzije *2.1.22* ove razlike više nisu toliko izražene, čak su za relativno mali broj komponenata u stablu, odnosno do 1200, na strani *Oracle Mojarra* implementacije, koja danas predstavlja jednu zrelu i stabilnu implementaciju koja se isporučuje sa mnogim *Servlet* kontejnerima. Dalje, trenutno postoji veliki broj dostupnih *JavaServer Faces* biblioteka komponenata. Neke od njih sadrže veće mogućnosti za izradu šablona i stilizovanje, neke imaju izraženije *AJAX* funkcionalnosti a neke su bogatije komponentama. Najkorišćenije biblioteke komponenata danas su *PrimeFaces, RichFaces* i *ICEFaces*. Od navedene tri biblioteke, svojim komponentama, dokumentacijom, lakoćom upotrebe, ali i po mnogim drugim parametrima izdvojila se *PrimeFaces* biblioteka komponenata, koja je danas integrisana i u neka razvojna okruženja kao što je *NetBeans IDE*. *PrimeFaces* biblioteka komponenata sadrži najveći broj komponenata dostupnih na korišćenje, kako osnovnih, tako i naprednih komponenata, veoma se lako integriše sa razvojnim okruženjem, ima jako dobro obrađene primere i dostupnu dokumentaciju, a po performansama i trendu korišćenja daleko je ispred ostale dve biblioteke komponenata. Ono što je najvažnije, korišćenje ovih komponenata prilikom razvoja *JavaServer Faces* aplikacija ne izaziva nikakve disfunkcionalnosti, a komponente se ponašaju uvek u skladu sa specifikacijom.

5. LITERATURA

[1] [Apache MyFaces] Apache MyFaces Software Foundation, *http://myfaces.apache.org;* Dan posete: Oktobar 2014

[2] [Bergsten, 2004] Hans Bergsten, „JavaServer Faces“, O'Reilly Media, Sebastopol 2004

[3] [Geary& Horstmann, 2010] David Geary, Cay Horstmann, „Core JavaServer Faces“, Oracle, Redwood Shores 2010

[4] [Eschen, 2008] Rainer Eschen, „ICEfaces 1.8: Next Generation Enterprise Web Development“, Packt Publishing, Birmingham 2009

[5] [JSR 314] JSF specification, https://jcp.org/aboutJava/communityprocess/final/jsr314/;

[6] [JSR 151] J2EE specification, https://www.jcp.org/en/jsr/detail?id=151

[7] [Lee, 2008] Jason Lee, „Practical RichFaces“, Apress, New York 2008

[8] [Mann, 2005] Kito D. Mann, „JavaServer Faces in Action“, Manning Publications, Greenwich 2005

[9] [Mojarra] Mojarra JavaServer Faces, *https://javaserverfaces.java.net;* Dan posete: September 2014

M.Sc. Veljko Bogosavljević, Bakson LTD - Programer
Kontakt: veljko.bogosavljevic@gmail.com
Oblasti interesovanja: Baze podataka, razvoj mobilnih aplikacija, Java tehnologije i razvoj Java aplikacija

Dr Siniša Vlajić, Univerzitet u Beogradu, Fakultet organizacionih nauka
Kontakt: vlajic@fon.bg.ac.rs
Oblasti interesovanja: Softverski proces, održavanje softvera, formalizacija softverskih paterna

Mr Dušan Savić, Univerzitet u Beogradu, Fakultet organizacionih nauka
Kontakt: dules@fon.bg.ac.rs
Oblasti interesovanja: Modelovanje i meta-modelovanje, softverski zahtevi, softverski paterni, domenski specifični jezici, razvoj Java web aplikacija

Mr Ilija Antović, Univerzitet u Beogradu, Fakultet organizacionih nauka
Kontakt: ilijaa@fon.bg.ac.rs
Oblasti interesovanja: Automatizacija razvoja korisničkog interfejsa, softverski zahtevi, softverski paterni, generatori koda

M.Sc. Miloš Milić, Univerzitet u Beogradu, Fakultet organizacionih nauka
Kontakt: mmilic@fon.bg.ac.rs
Oblasti interesovanja: Softversko inženjerstvo, kvalitet softvera, projektovanje softvera