Primena Spring alata u razvoju Java web aplikacije za zaposlene na fakultetu

Uroš P. Romić i Igor D. Manić

***Sadržaj* — U radu je opisana nova verzija aplikacije za zaposlene Elektrotehničkog fakulteta, bazirana na aktuelnim Java web tehnologijama otvorenog koda, kao što su Spring, JSF, Hibernate, iBatis i dr. U okviru rada su navedeni razlozi zbog kojih su izabrana navedena softverska rešenja, opisana je višeslojna arhitektura aplikacije sa Spring framework-om kao osnovom, kao i integracija različitih Java web tehnologija u funkcionalnu celinu. Posebna pažnja je posvećena opisu rešenja sigurnosti aplikacije, kao i modela za komunikaciju sa drugim informacionim sistemima u okviru fakulteta.**

***Ključne reči* — distribuirano programiranje, Facelets, Http Invoker, IoC Container, JSF, RPC, Spring, Spring Security.**

I. Uvod

NA Elektrotehničkom fakultetu (ETF) u Beogradu već duži niz godina zaposleni koriste web bazirane servise za pristup i unos različitih vrsta informacija [1]. Ovi servisi se oslanjaju na informacioni sistem DOSITEJ i njegove podsisteme – EVIDES (praćenje i organizovanje nastavnog procesa na fakultetu) i FIMES (kadrovska evidencija, finansijsko i materijalno poslovanje fakulteta). Prethodna verzija je realizovana kao Java web aplikacija, sa primenjenom Model 2 arhitekturom. U aplikaciji se, pored osnovnih Java servleta i JSP (Java Server Pages) tehnologije, u srednjem sloju za realizaciju poslovne logike koristi Apache Struts framework, verzija 1.3. U prezentacionom sloju pored već pominjanih JSP strana koristi se i JSTL (JSP Standard Tag Library) biblioteka, kao i Struts-ove biblioteke tagova. Za povezivanje sa bazom podataka koristi se Apache iBatis tehnologija. Aplikacija je povezana sa fakultetskom bazom podataka, kao i bazom podataka biblioteke ETF. Aplikativni server radi pod Linux operativnim sistemom, sa Apache HTTP serverom i Tomcat 5.5 servlet container-om. Nova verzija informacionog sistema fakulteta se bazira na open source rešenjima - Java web tehnologijama baziranim na Spring tehnologiji i PostgreSQL bazi podataka. Primena novih tehnologija je uslovila i razvoj nove verzije web aplikacije za zaposlene (u daljem tekstu eZaposleni), koja je tema ovog rada.

Namena rada je da prikaže mogućnost izabranih Java web tehnologija da odgovore na složene zahteve, kao i da predstavi jednu kombinaciju tih alata i strukturu aplikacije, kao polaznu osnovu za razvoj informacionog sistema, koji će pratiti aktuelne tokove u oblasti Java web tehnologija i biti u stanju da se efikasno prilagođava njihovom daljem razvoju. U radu su navedeni razlozi koji su uticali na izbor najpogodnijih softverskih rešenja za aplikaciju eZaposleni. Dat je opis funkcionalnosti i arhitekture aplikacije, pregled korišćenih tehnologija sa posebnim osvrtom na Spring framework, a potom i detaljniji prikaz primenjenih Spring rešenja u oblasti sigurnosti i komunikacije sa drugim sistemima putem distribuiranog programiranja.

U. P. Romić, Elektrotehnički fakultet u Beogradu, Srbija (telefon: +381-64-1995495, +381-11-3218436; e mail: uros.romic@etf.rs).

I. D. Manić, Elektrotehnički fakultet u Beogradu, Srbija (telefon: +381-64-3400529, +381-11-3218436; e mail: igor.manic@etf.rs).

II. Razlozi za primenu novih rešenja u daljem razvoju aplikacije za zaposlene

Iako je prethodna verzija aplikacije za zaposlene zadovoljavajuće ispunjavala svoju namenu, korišćene verzije Java web tehnologija, navedene u uvodu, stare su već nekoliko godina, i za (u skorijoj budućnosti neminovan) prelazak na novije verzije bilo bi potrebno slično vreme i trud kao i u slučaju razvoja baziranog na novim tehnologijama. Kao primer može poslužiti primena nove verzije Apache Struts2 frameworka, koja se značajno razlikuje od do sada korišćene 1.3 verzije. Osim toga, prelaskom sistema FIMES i EVIDES na aktuelne Java web tehnologije sa Spring framework-om kao osnovom strukture, pomenuta saradnja aplikacije sa tim sistemima bila bi otežana. Naime, prelaskom ovih sistema na nove verzije, baze podataka su razdvojene, radi veće samostalnosti svakog od sistema, i nije moguće koristiti dosadašnji model, koji se sastojao u komunikaciji aplikacija na nivou baze podataka. Takođe je bilo potrebno obezbediti mogućnost da aplikacija za zaposlene, u slučaju potrebe, odnosno zahteva korisnika, može da radi i sa samo jednim od sistema FIMES ili EVIDES, tj. da bude podeljena na međusobno nezavisne grupe modula, koji bi mogli da se izvršavaju i nezavisno i međusobno povezani, što u prethodnoj verziji nije bio slučaj. U pogledu saradnje između sistema, rešenje koje je izabrano kao optimalno je da se za dobijanje podataka iz FIMES i EVIDES aplikacija koristi distribuirano programiranje, na osnovu izabranog RPC modela (Remote Procedure Call – udaljeni pozivi metoda).

Prilikom izbora rešenja za primenu u poslovnoj logici aplikacije, obavljena je analiza postojećih tehnologija u toj oblasti. Kao potencijalne tehnologije, sa adekvatnom podrškom open source zajednice, uočene su Java tehnologije bazirane na Spring framework-u, PHP tehnologije, kao i Adobe Flex tehnologije. Pored navedenih, u obzir je uzeta i .NET platforma, međutim,

ova platforma u određenim segmentima nema adekvatnu open source podršku, niti portabilnost. Od Adobe Flex tehnologije se odustalo zbog činjenice da bi se njome pokrio samo jedan od slojeva sistema, za čije prihvatanje od strane razvojnog tima bi bilo potrošeno određeno vreme, a opet bi se došlo u situaciju da se većina slojeva mora razviti u nekoj od drugih tehnologija (slojevi za pristup bazi podataka, za udaljene pozive i za izveštavanje). Od PHP tehnologije se odustalo zbog neadekvatne podrške najbitnijim aspektima enterprise sistema (transakciona obrada podataka, složenije mogućnosti autentifikacije i autorizacije korisnika itd.). Spring framework izabran je kao rešenje koje omogućava lake zamene pojedinih elemenata aplikacije, bez izmene izvornog koda, što je u skladu sa pomenutom potrebom za modularnošću sistema. Takođe, Spring obezbeđuje podršku za druge tehnologije (npr. Hibernate, iBatis, JSF) i njihovo lakše konfigurisanje, kao i sopstveno rešenje za oblast sigurnosti aplikacije. Osim toga, podrška za distribuirano programiranje u okviru Spring framework-a u potpunosti odgovara prethodno navedenim zahtevima. Zbog svega navedenog, izabran je razvoj nove web aplikacije, koja bi zadržala funkcionalnosti prethodne, i unapredila ih modernim rešenjima zahvaljujući Spring tehnologiji.

III. KORIŠĆENE TEHNOLOGIJE I RAZVOJNO OKRUŽENJE

Osnovu novog sistema čini Spring framework, verzija 3. Osnovne mogućnosti koje se koriste su inverzija kontrole (Inversion of Control – IoC) putem „ubrizgavanja“ zavisnosti (Dependency Injection – DI) [2]. DI predstavlja šablon po kojem se vrednosti atributa nekog objekta, prilikom njegovog kreiranja, postavljaju spolja, od strane drugog objekta. U slučaju Spring tehnologije, objekat koji realizuje DI naziva se Spring IoC Container, i realizovan je u dve varijante - interfejsima BeanFactory i ApplicationContext. Informacije o klasama koje treba instancirati, o njihovom opsegu i međusobnoj zavisnosti smeštaju se u obliku meta-podataka, u XML konfiguracione fajlove. Instanciranje IoC Container-a u slučaju Java web aplikacije obavlja se dodavanjem odgovarajuće XML konfiguracije u web deskriptor (web.xml fajl). Osnovna pogodnost koju pruža DI, jeste tzv. loose coupling [3], tj. mogućnost da zavisnosti među objektima budu definisane preko interfejsa, a da se zatim, pomoću IoC Container-a i meta-podataka, koriste različite implementacije tih interfejsa.

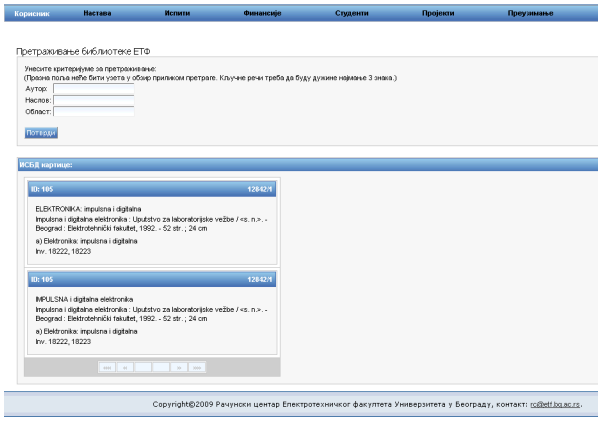
U eZaposleni aplikaciji primenjen je ApplicationContext za povezivanje zavisnih objekata koji se nalaze u susednim slojevima aplikacije. Osim korišćenja DI-a za povezivanja klasa u aplikaciji, upotrebljena je i ugrađena podrška Spring framework-a za druge alate, kao što su Hibernate ili iBatis, koja omogućava lakše konfigurisanje tih alata. Opet se upotrebljavaju DI šabloni i XML konfiguracija, bez dodatnog Java koda. Takođe se koristi i aspektno orijentisano programiranje (Aspect Oriented Programming – AOP), pre svega za upravljanje transakcijama. Od dodatnih Spring rešenja, koriste se Spring Web Flow i Spring MVC framework, kao i HTTP Invoker (u primeni distribuiranog programiranja) i Spring Security (za oblast sigurnosti).

Kao baza podataka koristi se PostgreSQL 8.4., a za komunikaciju sa bazom podataka aplikacije, kao i sa drugim bazama podataka (npr. biblioteka ETF), koriste se alati Hibernate i Apache iBatis. Kao rešenje za prezentacioni sloj aplikacije izabrana je JSF (Java Server Faces) tehnologija (verzija 1.2 Mojarra referentne JSF implementacije), obogaćena JBoss RichFaces dodatnom bibliotekom komponenti. Facelets framework izabran je kao view tehnologija. Facelets unapređuje razvoj u odnosu na JSP, kao standardnu JSF view tehnologiju, zahvaljujući potpunoj kompatibilnosi sa JSF-om i komponentama za kreiranje šablona [4]. Za upravljanje navigacijom između strana, koristi se Spring WebFlow. Time je obezbeđena podrška za složenije slučajeve korišćenja koji obuhvataju kretanje kroz nekoliko strana po tačno utvrđenom redosledu.

Kao razvojno okruženje koriste se Eclipse IDE, verzija 3.5 (Galileo), sa podrškom za razvoj J2EE aplikacija. Ovo okruženje izabrano je zbog širokih mogućnosti konfiguracije i dobre podrške u vidu dodataka (plugin-ova) za druge alate koji se koriste za razvoj aplikacije. Za upravljanje strukturom aplikacije i potrebnim bibliotekama koristi se Maven (verzija 2.1) i M2Eclipse plugin za Eclipse. Kontrola verzija obavlja se pomoću Subversion sistema i Subclipse plugin-a. Podršku za lakši pregled i validaciju Spring Beans i Spring Web Flow XML konfiguracija obezbeđuje Spring IDE plugin. Da bi se ubrzao razvoj JSF strana, bio je potreban plugin koji bi imao efikasan vizuelni editor, s obzirom na to da ugrađeni editor za web strane nema mogućnost prikaza pojedinih JSF komponenti. Izabran je JBoss Tools plugin, koji je potpuno kompatibilan sa korišćenim RichFaces i Ajax4JSF komponentama, a podržava i standardne JSF, kao i Facelets komponente. Kao razvojni aplikacioni server koristi se Apache Tomcat, verzija 6, čiji se rad kontroliše iz samog razvojnog okruženja.

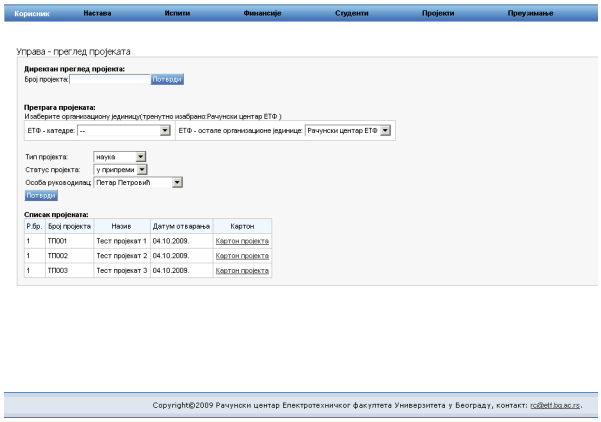
IV. FUNKCIONALNOSTI I ARHITEKTURA APLIKACIJE

Aplikacija eZaposleni, kao i prethodna verzija aplikacije za zaposlene, obuhvata pre svega grupe funkcionalnosti vezane za održavanje nastave i ispita, kao i radne obaveze i primanja zaposlenih. Funkcionalnosti vezane za nastavu i ispite koriste podatke EVIDES aplikacije i metode koje realizuju ove funkcionalnosti komuniciraju sa EVIDES-om u okviru servisnog sloja. Ova grupa funkcionalnosti obuhvata unos ocena, pregled ispitnih prijava, pregled podataka o studentima (lični podaci, ispiti, urađene obaveze itd.), pregled rasporeda časova i rasporeda ispita u ispitnom roku i dr. Funkcionalnosti vezane za radne obaveze zaposlenih obuhvataju unos i pregled podataka o aktivnostima koje utiču na varijabilni deo primanja: broj održanih časova, dežurstava, pregledanih ispitnih i drugih radova i sl. Ove funkcionalnosti oslanjaju se na podatke i iz FIMES i iz EVIDES aplikacija, i neophodna je komunikacija sa oba sistema.



Sl. 1. Primeri korisničkog interfejsa aplikacije – strana za pretraživanje biblioteke ETF.

Funkcionalnosti vezane za primanja zaposlenih obuhvataju pregled isplata zaposlenima po različitim osnovama, za zadat vremenski period, i koriste podatke informacionog sistema FIMES. Osim opisanih funkcionalnosti, aplikacija eZaposleni obuhvata i različita obaveštavanja zaposlenih putem elektonske pošte, prijavu kvarova tehničkoj podršci, prijavu za MSDNAA program, pretraživanje fakultetske biblioteke (Sl. 1.), kao i posebnu grupu funkcionalnosti koje su namenjene upravi fakulteta (Dekan, prodekani itd.) (Sl. 2.).



Sl. 2. Primeri korisničkog interfejsa aplikacije – strana za pregled podataka o projektima za upravu fakulteta.

Posebna pažnja tokom razvoja aplikacije posvećena je višeslojnoj arhitekturi, radi očuvanja nezavisnosti i paralelnog razvoja pojedinih slojeva aplikacije, kao i modularnosti aplikacije, koja omogućava podelu na osnovu sličnih grupa funkcionalnosti. eZaposleni je podeljena na sledeće slojeve: **domain**, **DAO** (Data Access Object), **service**, **controller**, **view**. Slojevi DAO i service dodatno su podeljeni na implementacioni i API deo, u skladu sa korišćenom Spring tehnologijom. API delovi ovih slojeva sadrže interfejse sa potpisima metoda, koje „vidi“ ostatak aplikacije. Implementacioni deo sadrži jednu ili više implementacija svakog od interfejsa iz API dela, pri čemu se, u zavisnosti od potrebe (testiranje, promena tehnologije i sl.), korišćena implementacija može promeniti u Spring XML definicijama, bez promene Java izvornog koda. **Domain sloj** sadrži Java bean-ove koji odgovaraju tabelama u bazi podataka ili funkcionalnostima poslovne logike aplikacije. Ovakvo preslikavanje podataka iz baze omogućava efikasnije korišćenje Hibernate i iBatis alata. **DAO sloj** sadrži klase koje obavljaju osnovne operacije čitanja i ažuriranja baze podataka. Opisana podela na implementacioni i API deo, u DAO sloju se ogleda u korišćenju Hibernate i iBatis implementacija postojećih interfejsa. **Servisni (service) sloj** ima jedan skup implementacija i predstavlja „most“ između funkcionalnosti sistema dostupnih korisniku i DAO sloja, s obzirom na to da izvršavanje većine funkcionalnosti zahteva pozivanje više DAO metoda, kao i upravljanje transakcijama. Ovaj sloj takođe služi za povezivanje sa drugim sistemima, pozivanjem udaljenih procedura, što je detaljnije opisano u posebnom poglavlju. Prezentacioni deo aplikacije, okrenut ka korisnicima, razvijen je korišćenjem JSF tehnologije. Podeljen je na controller i view slojeve. **Sloj kontrolera** sadrži skup standardnih JSF JavaBean objekata, koji predstavljaju podršku komponentama korisničkog interfejsa. U okviru njih se obavlja navigacija između strana, validacija podataka unetih od strane korisnika, i prenos podataka između view i service sloja. **View sloj** predstavlja korisnički interfejs, izgrađen od standardnih JSF i dodatnih JBoss-ovih RichFaces i a4j komponenti. Web strane poštuju XHTML sintaksu, kao preporučeni standard od strane W3C (World Wide Web Consortium) [5] i koriste šablone kreirane pomoću Facelets framework-a.

Modularnost aplikacije je ostvarena podelom na pakete na osnovu grupa sličnih funkcionalnosti. Posebno je značajno je da su i view komponente, odnosno JSF strane, smeštene u pakete tj. direktorijume u izvornom Java kodu, što olakšava dodavanje ili uklanjanje pojedinih modula i njihovo pakovanje u JAR biblioteke. Meta-podaci za Spring IoC Container podeljeni su tako da svi slojevi u pojedinim modulima imaju svoje XML konfiguracione fajlove, koji su takođe smešteni u odgovarajuće pakete.

V. SPRING REŠENJA U OBLASTI SIGURNOSTI I KOMUNIKACIJE SA DRUGIM SISTEMIMA

S obzirom na to da je u pitanju web aplikacija sa tankim klijentom, koja treba da bude dostupna sa bilo koje lokacije na Internetu, posebna pažnja posvećena je sigurnosti aplikacije, odnosno autentifikaciji i autorizaciji korisnika. Kao rešenje koje zadovoljava sve postavljene zahteve po pitanju sigurnosti, a u isto vreme je i deo Spring tehnologije, izabran je Spring Security projekat. Konfigurisanje Spring Security-a obavlja se definisanjem odgovarajućih klasa u XML fajlovima u okviru aplikacije, koje Spring–ov ApplicationContext učitava zajedno sa ostalim Spring XML definicijama. Dodatnu pogodnost, uvedenu u Spring Security od verzije 2.0, predstavlja tzv. Namespace konfiguracija, koja uz pomoć unapred definisanih XML komponenti omogućava jednostavno podešavanje sigurnosnih opcija - u nekoliko Spring definicija. Ipak, mora se napomenuti da je za svako naprednije konfigurisanje sigurnosnih elemenata potrebno koristiti klasičan način definisanja bean-ova, koji daje veću

slobodu u izboru klasa za realizaciju pojedinih funkcionalnosti. U aplikaciji eZaposleni korišćeni su elementi Namespace konfiguracije onoliko koliko su mogli da odgovore na zahteve, dok su dodatna podešavanja izvršena korišćenjem standardnih bean definicija. Podešavanje sigurnosnih opcija odvojeno je u poseban XML konfiguracioni fajl. Namespace konfiguracija obuhvatila je definisanje parametara osnovnog <http> elementa, kao i njegovih podelemenata – definisanje login forme, šablona koji određuju koje će korisničke role imati pravo pristupa pojedinim URL-ovima aplikacije, ograničavanje broja sesija po korisniku i sl. Dodatna podešavanja, realizovana standardnim bean definicijama, odnose se na definisanje odgovarajućih klasa za povezivanje sa bazom podataka i dohvatanje podataka o korisnicima (korisničko ime, lozinka, role), kao i definisanje enkripcije lozinki (SHA-256 sa korisničkim imenom kao salt-om). Ostavljena je mogućnost daljeg proširivanja konfiguracija klasama koje će odlučivati o mogućnosti pristupa aplikaciji na osnovu proizvoljnih parametara, kao što je npr. IP adresa korisnika.

Pomenuto je da je jedan od najvažnijih zahteva koji su postavljeni, efikasna realizacija komunikacije putem pozivanja udaljenih servisa sa postojećim i budućim verzijama fakultetskih informacionih sistema. S obzirom na to da se i eZaposleni i druge aplikacije oslanjaju na Spring tehnologije, za njihovu komunikaciju je izabran jedan od sledećih RPC modela za koje Spring biblioteke obezbeđuju odgovarajuću podršku: RMI, Hessian/Burlap, HTTP Invoker ili JAX-RPC/SOAP [3]. Izabran je HTTP Invoker model, koji omogućava udaljene pozive putem HTTP-a i koji koristi standardnu Javu serijalizaciju objekata. HTTP komunikacija je važan uslov, imajući u vidu sigurnosna ograničenja fakultetske mreže ETF koja bi predstavljala problem za standardne RMI servise. Takođe, Spring daje potpunu podršku za implementaciju ovog modela kroz XML konfiguracije. HTTP Invoker model je, na „klijentskoj“ strani aplikacije, realizovan deklarisanjem bean-a u servisnom sloju aplikacije, koji referencira Spring klasu HttpInvokerProxyFactoryBean. U okviru deklaracije inicijalizovani su atributi ServiceURL i ServiceInterface. ServiceURL predstavlja URL na koji se odaziva aplikacija koja pruža servis. ServiceInterface je ime interfejsa sa potpisima servisnih metoda. Deklarisani HttpInvokerProxyFactoryBean ubačen je kao atribut u sve ostale servisne bean-ove kojima je potrebno dohvatanje podataka iz udaljenih sistema, pri čemu se tretira kao standardni Spring bean i u okviru aplikacije nisu potrebna dodatna podešavanja u vezi sa HTTP Invoker-om. Na „serverskoj“ strani, u aplikaciji koja pruža servis, HTTP Invoker model realizovan je deklarisanjem bean-a tipa HttpInvokerServiceExporter, sa atributima service i serviceInterface. Prvi atribut referencira bean sa servisnim metodama, a drugi interfejs koji service bean implementira. Da bi se zahtevi HTTP Invoker klijenta preveli u pozive servisnih metoda, definisan je i odgovarajući servlet u web.xml deskriptoru aplikacije koja pruža servis. Ime servleta treba da bude identično imenu HttpInvokerServiceExporter bean-a. URL šablon servleta određuje na koji će URL aplikacija pružati servis, odnosno koji će ServiceURL atribut koristiti HTTP Invoker klijent.

VI. ZAKLJUČAK

Rešenja primenjena u aplikaciji eZaposleni izabrana su sa namerom da ne samo olakšaju komunikaciju i prilagođavanje sistemima u okruženju, već i da budu međusobno kompatibilna i da se, gde je moguće, efikasno dopunjuju. Time bi trebalo da bude olakšan dalji razvoj sistema u pravcu dodavanja novih funkcionalnosti kao što je dinamičko upravljanje korisničkim privilegijama uz pomoć Spring Security-a ili višejezička podrška. Takođe, olakšan je i prelazak na nove verzije korišćenih alata ili potpuna zamena tehnologija u pojedinim slojevima aplikacije. Izabrano razvojno okruženje omogućava brzo prilagođavanje sistema drugačijim potrebama, tako da može da se upotrebi kao uzorak za niz sličnih web aplikacija. Na kraju, treba napomenuti da su sve korišćene tehnologije besplatne, što ovakav sistem čini pogodnim za studentske i istraživačke projekte.

LITERATURA

[1] Računski centar Elektrotehničkog fakulteta, Fakultetski servisi (2009), http://rc.etf.rs/?p=web_fakultetski_servisi

[2] Rod Johnson, Juergen Hoeller, Keith Donald and others (2009), *Spring Java Application Framework Reference Documentation*, http://www.springsource.org/documentation

[3] C. Walls, R. Breidenbach, “Spring in Action”, 2nd Ed, Greenwich, Manning Publications Co, 2008.

[4] B. Aranda, Z. Wadia, “Facelets Essentials: Guide to JavaServer Faces View Definition Framework”, Apress, Inc., United States, 2008.

[5] W3C (2001), *XHTML™ 1.1 - Module-based XHTML*, http://www.w3.org/TR/xhtml11/

ABSTRACT

Change of technology in the surrounding systems (towards open source Java web solutions) set new requirements for the web application used by employees of the School of Electrical Engineering in Belgrade. Regarding present conditions, a new web application was chosen as the most practical solution. To achieve that, several Java web frameworks (such as Spring, JSF, Hibernate, iBatis, Facelets etc.) were combined together into advanced system, that provides rapid development (using Eclipse IDE), high modularity and configurability (thanks to Spring framework) and modern presentation layer components provided by JBoss RichFaces. The web application architecture consists of several layers: domain, DAO, service, controller and view. Security issues are resolved by using Spring Security framework, which provides simple Spring-based configuration, but yet some advanced features. Distributed computing was made possible by using Spring's RPC Model - Http Invoker. Future development should include easy addition of new modules and improving security features by, for example, enabling dynamic user role management.

USING SPRING-BASED TOOLS IN DEVELOPMENT OF JAVA WEB APPLICATION FOR FACULTY EMPLOYEES

Uroš P. Romić and Igor D. Manić