UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Igor Zirdum

Razvoj spletnega čarovnika v odprtokodnem ogrodju Vaadin

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: dr. Igor Rožanc

Ljubljana, 2012

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00242/2012
Datum: 02.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **IGOR ZIRDUM**

Naslov: **RAZVOJ SPLETNEGA ČAROVNIKA V ODPRTOKODNEM OGRODJU VAADIN**

DEVELOPMENT OF A WEB WIZARD IN THE VAADIN OPEN SOURCE FRAMEWORK

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V diplomski nalogi predstavite izdelavo spletnega čarovnika z uporabo odprtokodnega spletnega razvojnega ogrodja Vaadin, ki omogoča izdelavo ti. bogatih spletnih aplikacij (RIA). Čarovnik je vrsta uporabniškega vmesnika, ki z zaporedjem enostavnejših pogovornih oken vodi uporabnika pri pravilni izvedbi zahtevnejših vnosov. V okviru tega najprej predstavite uporabljene tehnologije, sam postopek izdelave in izgled pa prikažite na zgledu priprave zemljiškoknjižnega predloga v spletni aplikaciji elektronska zemljiška knjiga (eZK).

Mentor:

viš. pred. dr. Igor Rožanc

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a **Igor Zirdum**,
z vpisno številko **63050378**,
sem avtor/-ica diplomskega dela z naslovom:

Razvoj spletnega čarovnika v odprtokodnem ogrodju Vaadin

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)
  viš. pred. dr. Igor Rožanc

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne _____ Podpis avtorja/-ice: _____

Zahvala

Zahvalil bi se svojemu mentorju, dr. Igorju Rožancu, za pomoč in nasvete pri izdelavi diplomske naloge.

Zahvaljujem se tudi vsem sodelavcem pri podjetju SRC d.o.o., ter vsem drugim, ki so kakorkoli pomagali pri izdelavi diplomskega dela.

Posebna zahvala gre pa staršema, ki sta mi študij omogočila in v vseh teh letih stala za mano. Hvala!

Tako se je Vaadin zbudila iz prvega spanca,
lišaje ji je Seppo dal za jesti,
medico ji je dal za piti,
in se povzpel na njo in pričel pot.

Kazalo

Seznam slik

Uporabljene kratice

AJAX	ang. Asynchronous Javascript and Xml – asinhroni Javascript in XML
API	ang. Application Programming Interface – aplikacijski programski vmesnik
CRP	Centralni register prebivalstva - osrednja podatkovna baza z najosnovnejšimi podatki o prebivalstvu Slovenije
CSS	ang. cascading style sheets - podloge, predstavljene v obliki preprostega slogovnega jezika, ki skrbi za prezentacijo spletnih strani
DI	ang. Dependency Injection – načrtovalski vzorec, ki dovoljuje izbiro komponente v času izvajanja programa
EJB	ang. Enterprise Java Bean – strežniška javanska komponenta
EMŠO	Enotna matična številka občana
eZK	Elektronska zemljiška knjiga
GK	Glavna knjiga – evidenca o stvarnih pravicah na nepremičnini in pravnih dejstvih v zvezi z njo
GWT	Google Web Toolkit – Googlov nabor orodij za razvoj obogatenih spletnih aplikacij (RIA)
HTML	ang. HyperText Markup Language – označevalni jezik za izdelavo spletnih strani
IDE	ang. Integrated Development Environment – integrirano razvojno okolje
IOC	ang. Inversion of Control – obrat kontrole
JAR	ang. Java Archive – javanska knjižnica, ki združuje vsebinsko povezane razrede
Java EE	ang. Java Enterprise Edition – javanska platforma namenjena delovanju v zahtevnejših informacijskih sistemih
JPA	ang. Java Persistence API – javanski aplikacijski programski vmesnik za delo z relacijskimi podatki
JSP	ang. JavaServer Pages – tehnologija za izdelavo dinamičnih spletnih strani
JVM	ang. Java Virtual Machine – javanski navidezni stroj

MVC	ang. Model-View-Controller - model-pogled-krmilnik
OP-1	Osnovni pravni položaj
POM	ang. Project Object Model, konfiguracijska datoteka za Maven
POJO	ang. Plain Old Java Object – enostaven javanski objekt
PT-1	izvedena pravica - pravno dejstvo, ki omejuje lastninsko pravico (tj. osnovni pravni položaj nepremičnine)
RIA	ang. Rich Internet Application – obogatena spletna aplikacija
SWF	Spring Webflow – podprojekt ogrodja Spring za razvoj obogatenih spletnih aplikacij(RIA)
WAR	ang. Web Application Archive, spletna aplikacija stisnjena v arhiv
WFV	ang. Wizards for Vaadin – dodatek za ogrodje Vaadin
XML	ang. Extensible Markup Language – razširljiv označevalni jezik

Povzetek

Namen diplomske naloge je bil izdelati spletni čarovnik z uporabo AJAX tehnologij. To smo naredili na zgledu priprave zemljiškoknjižnega predloga v spletni aplikaciji eZK.

V uvodnem delu bomo na kratko analizirali obstoječo rešitev in v njej uporabljene tehnologije.

Nato bomo spoznali pomembnejša orodja in tehnologije, ki smo jih uporabili pri izdelavi svoje rešitve, pri čemer se bomo osredotočili na javanska orodja in komponente: ogrodje Vaadin in aplikacijski strežnik JBoss.

Osrednji del naloge zajema podrobnejši opis komponent. Pri tem so izpostavljena zanimivejša področja, ki so hkrati predstavljala največje izzive pri razvoju, predvsem uporaba Vaadin ogrodja.

Rezultat naloge je osnovna dokončana spletna aplikacija z vsemi zastavljenimi funkcijami za dokončanje izbranega postopka eZK, kar vključuje izbiro uporabnika, izbiro nepremičnine, izbiro osnovnih pravnih položajev ter dinamično formo za vpis nove izvedene pravice na izbrano nepremičnino.

Ob zaključku razvoja smo dosegli vse zastavljene cilje, poleg tega pa se je že med razvojem porodilo tudi več idej za možne izboljšave in nadgradnje, kar je mogoče z nekaj dodatnega dela enostavno izvesti glede na samo zasnovo aplikacije.

Ključne besede:

- čarovnik,
- spletna aplikacija,
- Vaadin
- JBoss
- zemljiška knjiga,

Summary

The thesis guides us through the necessary steps of developing a web wizard using the AJAX technologies. The preparation of a proposal in the web application eZK is the example on which the demonstration of wizard construction is performed.

In the introductory part we shortly analise the current solution and the technologies used.

Then we introduce the most important tools and technologies used in our project focusing on java based tools such as JBoss application server and Vaadin framework.

The main part of the thesis consists of a detailed explanation of the most interesting components with the highlight on the areas which were the greatest challenges in the developing process, e.e. the usage of the Vaadin framework.

The result of the thesis is a functioning web application with all the necessary steps needed to complete a procedure in the eZK application.

We conclude with some findings about our solution, additionally we present some ideas that came up during the developing process.

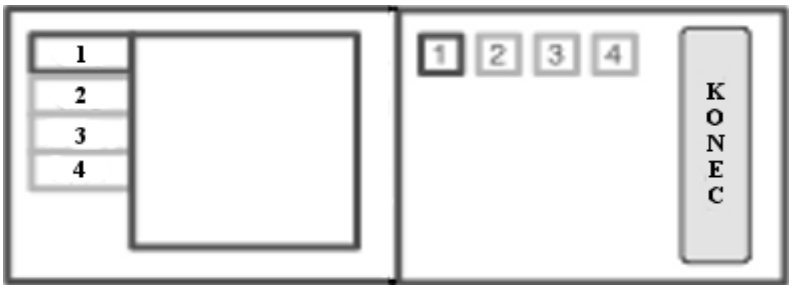
Key words:

- wizard,
- web application,
- Vaadin framework
- JBoss,
- land registry,

1 Uvod

Načrtovanje obogatene spletne aplikacije (ang. rich internet application – RIA) [1] je lahko preizkušnja za še tako izkušeno ekipo inženirjev. Največji izziv predstavlja oblikovanje odzivne in intuitivne izkušnje z združevanjem paradigm spletnih in namiznih okolij. Nekatere paradigme, ki obstajajo v namiznem okolju, so neprimerne za splet, medtem ko veliko znanih spletnih paradigm (npr. eksplicitno osveževanje strani) ni več potrebnih z uporabo RIA tehnologij (npr. AJAX [2]).

V svojem diplomskem delu smo se lotili priprave spletnega čarovnika. Čarovnik je tip uporabniškega vmesnika, katerega predstavlja zaporedje pogovornih oken, ki uporabnika popeljejo skozi vrsto točno določenih (ponavadi kompleksnih) korakov. Glede na tip obravnavanih podatkov je lahko delovni tok vodoraven ali navpičen [3] (slika 1)



Slika 1: Primer vodoravnega in navpičnega delovnega toka

Kot osnovo za naš čarovniški uporabniški vmesnik smo vzeli postopek priprave zemljiškoknjižnega predloga v elektronski zemljiški knjigi (eZK)[4].

eZK je kompleksen sistem, zgrajen iz treh osnovnih gradnikov:

- spletnega podportala eZK, preko katerega je možna oddaja zemljiškoknjižnih predlogov ter pridobivanje izpisov iz zemljiške knjige,
- namizne aplikacije `Dn vpisnik`, s katerim uporabniki na sodiščih izvajajo odločanje o vpisu za zemljiškoknjižne predloge, oddane preko Podportala eZK,
- `Glavne knjige` (GK), ki hrani podatke o vseh nepremičninah, pravicah ter njihovih imetnikih ter zajema celotno logiko vpisovanja za vse postopke.

Projekt izgradnje eZK je prinesel veliko izzivov. Eden od ključnih izzivov tega projekta je bilo obvladovanje zahtevne in precej obširne vsebine. S pomočjo eZK je namreč možno izvesti približno osemdeset različnih postopkov, pri čemer je vsak postopek sestavljen iz

spletnega dela, validacije ter vpisa v glavno knjigo. Pri svojem delu smo se osredotočili na izgradnjo enega postopka eZK (spletni del), kateri bi simuliral obnašanje postopka v obstoječi aplikaciji eZK.

2 Načrtovanje rešitve

Tipičen postopek, ki smo ga izbrali in razvili, sestavljajo torej sledeči koraki:

- vnos predlagatelja,
- izbira nepremičnine, ki bo predmet postopka,
- izbira osnovnih pravnih položajev (OP-1), ki so vezani na nepremičnino,
- vnos nove izvedene pravice, z vnosom ustreznih podatkov v vnosno formo, na podlagi tipa pravice,
- predogled vnesenih podatkov

Koraka za vnos predlagatelja in izbiro nepremičnine bomo realizirali kot iskalna obrazca, le da bosta uporabljala različen izvor podatkov. Obrazec za vnos predlagatelja bo iskal po registru CRP, iskalni pojem pa bo enotna matična številka občana (EMŠO). Obrazec za izbiro nepremičnine bo iskal po glavni knjigi, iskalni pojem pa bo `ID` (enoličen ključ) nepremičnine. Korak za izbiro OP-1 bomo realizirali kot izbirni obrazec, kjer bo uporabnik lahko izbiral med več OP-1, ki so vezane na nepremičnino. Korak za vnos izvedene pravice bomo realizirali kot dinamični obrazec z množico vnosnih polj, ki se bo spreminjala v odvisnosti na tip izvedene pravice. Med vnosnimi polji bomo razvili tudi šifrant, ki bo vseboval seznam svetovnih denarnih valut, z njegovo pomočjo pa bomo določili valuto enega izmed vnosnih polj.

Zadnji korak predstavlja tudi največji izziv, zato ga bomo v nadaljevanju podrobno opisali.

2.1 Analiza obstoječe rešitve

Portal eZK je razdeljen na tri dele:

Na zgornjem delu portala so uporabniku na voljo povezave, ki so skupne vsem opravilom: odjava, izbira vloge, izbira eZK-opravila itd. Na spodnjem levem delu portala je meni s povezavami do funkcionalnosti portala. Meni je odvisen od posameznega e-opravila. Meni za eZK-opravila ponuja dostop do funkcionalnosti za eZK podportal. Na spodnjem desnem delu se uporabniku prikaže vsebina, ki je odvisna od izbrane povezave.

2.1.1 Uporabljene tehnologije pri obstoječi rešitvi

Portal eZK temelji na MVC [5] arhitekturi, ki jasno ločuje aplikacijsko logiko od uporabniškega vmesnika. Pri implementaciji portala se uporabljajo naslednje tehnologije oziroma ogrodja:

- aplikacijski strežnik Jboss [6] za grafični vmesnik in storitveni strežnik.
- ogrodje Apache Tiles [7] za postavitev strukture portala. Apache Tiles omogoča po eni strani razdelitev spletne strani v fragmente, ki jih je mogoče vstaviti v spletno stran med izvajanjem, po drugi strani omogoča uporabo prikaznih vzorcev.
- ogrodje Spring [8] kot implementacijo MVC arhitekture in integracijsko ogrodje.
- Spring Webflow [9] za implementacijo vseh procesov oziroma postopkov, kjer se interakcija z uporabnikom izvaja v vnaprej določenih korakih. Vsi eZK postopki so izvedeni s pomočjo Spring Webflow.
- za dostop do baze podatkov se uporablja JPA (Hibernate) [10]

2.1.2 eZK postopki

Vsi eZK postopki so implementirani s pomočjo Spring Webflow-a. Postopki se definirajo v XML datoteki v vnaprej določenem formatu. Opis vsakega postopka vsebuje opis vseh korakov in prehodov med njimi. Spring Webflow podpira dedovanje tako, da skupne korake lahko definiramo v skupnem abstraktnem postopku, od katerega dedujemo.

Postopki in koraki so razviti s pomočjo gradnikov, ki so skupni vsem postopkom. To omogoča ponovno uporabo skupne funkcionalnosti. Gradniki so nastavljivi, tako da lahko uporabljamo isti gradnik z različnimi konfiguracijskimi podatki in pravili. Lahko določimo eno ali več akcij (na primer validacija podatkov), ki se kličejo med izvedbo operacij posameznega gradnika.

2.2 Uporabljena orodja in tehnologije

V tem poglavju so predstavljena orodja in tehnologije, ki smo jih uporabili pri izdelavi aplikacije. Pri razvoju aplikacije smo uporabljali orodja, ki so odprtokodna in prosto dostopna.

2.2.1 Java EE

Celoten sistem je spisan v programskem jeziku Java, različice 6.0 [11], zato za pogon (ang. runtime environment) potrebuje JVM (Java Virtual Machine) različice vsaj 1.6.

JavaEE je javansko razvojno okolje, ki se od običajne Jave razlikuje v tem, da vsebuje knjižnice, ki omogočajo razvoj modularnih, porazdeljenih aplikacij, ki tečejo na *aplikacijskem strežniku*. Dva najbolj znana komercialna strežnika sta WebLogic [12] in WebSphere [13], medtem ko pri odprtokodnih rešitvah prevladuje JBoss AS. Ker je bila aplikacija eZK razvita na aplikacijskem strežniku JBoss 5.1, smo želeli tudi tu storiti korak naprej in smo našo aplikacijo razvili na strežniku JBoss 7.1.

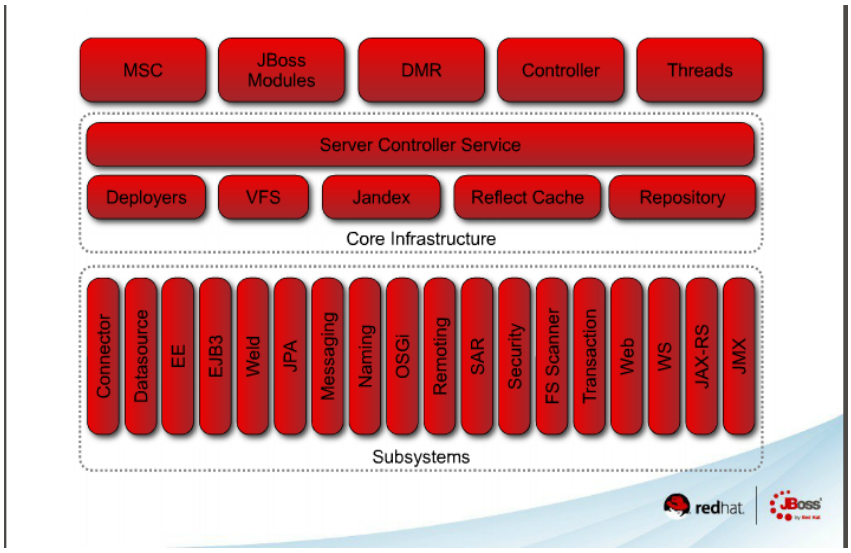
2.2.1.1 Servlet

Servlet je poseben javanski razred in ga poznamo le pri jeziku Java. Je javanski razred, ki razširi možnosti strežniških programov, ki delujejo po načelu zahteva-odgovor.

2.2.1.2 Aplikacijski strežnik JBoss

Aplikacijski strežnik Jboss [6] je zmogljiv odprtokodni aplikacijski strežnik, ki implementira Java EE platformo. Javanski aplikacijski strežnik standardizira arhitekturo aplikacijskega razvoja, kar stori tako, da definira različne *komponentne modele* (ang. component model) - standarde, katere lahko razvijalci uporabljajo za razvoj *komponent*. Te lahko potem z uporabo standardnega *postavitvenega modela* (ang. deployment model) namestijo na aplikacijski strežnik (npr. kot aplikacijo v `.war` obliki). Aplikacijski komponentni model vključuje standarde kot so EJB, JSP in servleti. Strežnik nudi podporo različnim storitvam, ki so na voljo aplikaciji, kot so:

- EJB 3.0
- JMS
- Quartz
- spletne storitve



Slika 2: Arhitektura aplikacijskega strežnika JBoss 7

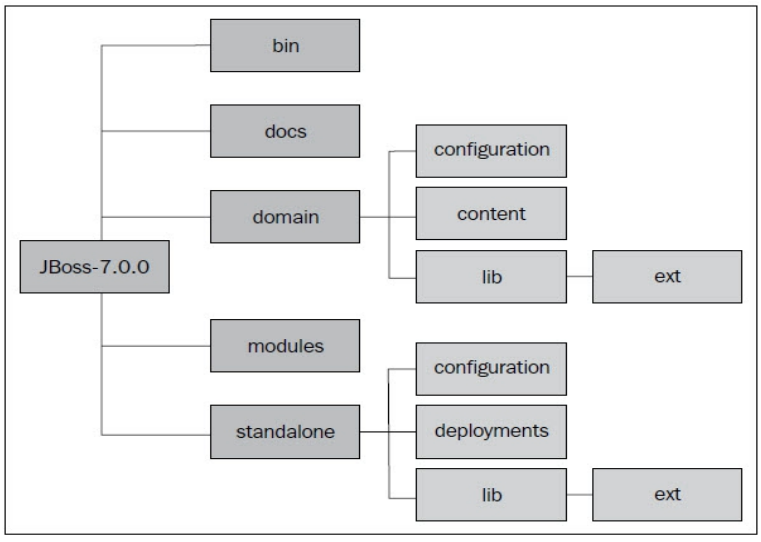
Ena poglavitnih sprememb glede na starejše verzije strežnika je, da vsaka knjižnica (`jar`) postane modul, ravno tako kot vse namestitve (`war`, `ear`, `jar`). Obstoječe module lahko z

svojo aplikacijo zvežemo preko datotek `jboss-deployment-structure.xml` (slika 3) ali `manifest.mf`, tako razvijalcu ni treba knjižnic, ki na strežniku že obstajajo, vključevati v svojo aplikacijo.

```
<jboss-deployment-structure xmlns="urn:jboss:deployment-structure:1.0">
  <deployment>
    <dependencies>
        <module name="org.apache.log4j" export="true"/>
        <module name="org.apache.commons.codec" export="true"/>
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

Slika 3:XML datoteka za dodajanje modulov k aplikaciji

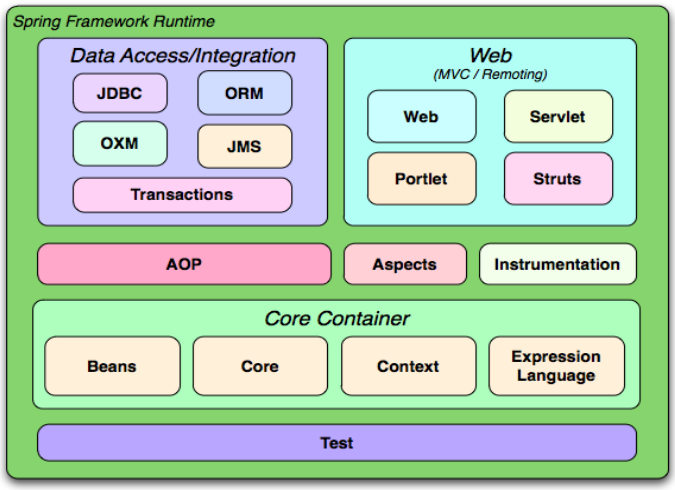
JBoss verzije 7 lahko poženemo v načinu `domain` ali `standalone`, kjer lahko v `domain` načinu upravljamo z več instancami aplikacijskega strežnika iz ene kontrolne točke. `Standalone` način je namenjen upravljanju ene instance strežnika in je zelo podoben delovanju starejših verzij strežnika. Konfiguracija se v tej verziji strežnika nahaja na enem mestu, v odvisnosti od tega kakšne vrste strežnika želimo upravljati, je to lahko datoteka `standalone.xml` ali `domain.xml`. Posamezna konfiguracijska datoteka se nahaja v podmapi `configuration` podmap `domain` ali `standalone`. Moduli imajo svojo posebno mapo `modules` v korenskem imeniku strežnika (slika 4).



Slika 4: Struktura map aplikacijskega strežnika JBoss 7

2.2.2 Ogrodje Spring

Spring [8] je odprtokodno spletno ogrodje za Java platformo. Sestavljajo ga številni različni moduli (slika 5). Pomemben del ogrodja Spring je Inversion of Control (IoC) [14] vsebnik, ki skrbi za kreiranje objektov, klicanje inicializacijskih metod ter nastavljanje objektov z medsebojnim povezovanjem (ang. dependency injection). Modul nastavljamo preko XML datotek, ki vsebujejo definicije zrn (ang. Beans), ki so potrebne za kreiranje Java objektov.



Slika 5: Prikaz modulov ogrodja Spring

Postopki na sistemu eZK so bili razviti s pomočjo razširitve za ogrodje Spring, ki se imenuje Spring Webflow (SWF). SWF je razširitev ogrodja Spring, ki je namenjeno upravljanju dialogov med strežnikom in odjemalcem. Skrbi za navigacijo med stranmi spletne aplikacije, konverzacijska stanja med njimi, pravila za navigacijo ter modularizacijo in ponovno uporabo že razvitih delov aplikacije.

2.2.3 Tehnologija Ajax

S kratico AJAX [2] poimenujemo skupek tehnologij, ki jih uporabljamo za izdelavo interaktivnih spletnih aplikacij. Glavna prednost uporabe AJAX-a je v tem, da lahko posodobimo vsebino trenutno prikazane spletne strani (oz. del nje) asinhrono. S tem omogočimo hitrejše delovanje aplikacij, saj se med odjemalcem in strežnikom pretaka manj podatkov. Istočasno pa uporabnikom omogočimo, da neprekinjeno interagirajo z aplikacijo, saj ni potrebno čakati da se po izvedeni akciji stran v celoti naloži. AJAX tehnologija se v spletnih aplikacijah uporablja pri:

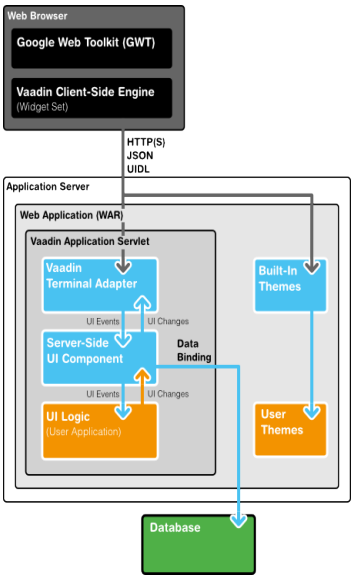
- preverjanju veljavnosti vnesenih podatkov na formah (v realnem času),
- ponujanju možnih vrednosti na podlagi delnega vnosa (ang. autocompletion) (npr. pri vnosu besedila v iskalnik),
- periodičnem osveževanju strani (npr. borzni indeksi) ter pridobivanju podatkov na zahtevo, kjer ni potrebno osveževanje celotne strani

2.2.4 Google Web Toolkit

Google Web Toolkit (GWT) [17] je odprtokodno orodje za razvoj spletnih aplikacij, ki temeljijo na tehnologiji AJAX. Z izjemo nekaterih knjižic je celotna izvorna koda v Javi. Pri razvoju uporabljamo sintakso in osnovne tipe programskega jezika Java z gradniki grafičnega vmesnika orodja GWT. GWT prevajalnik nato prevede javansko kodo v v kodo programskega jezika JavaScript.

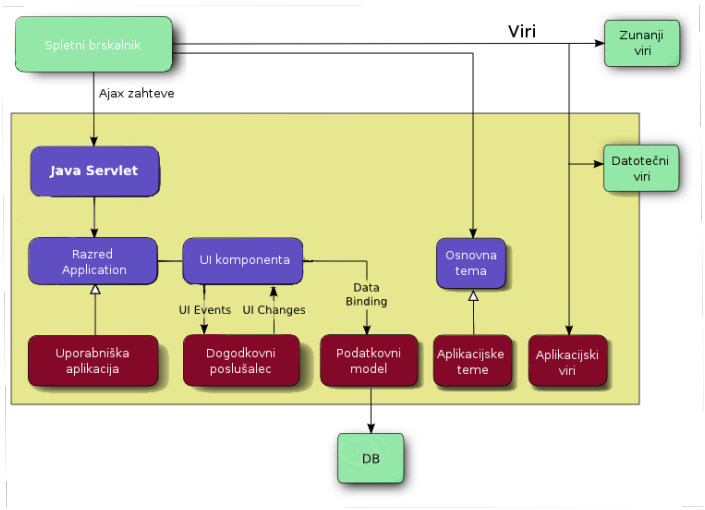
2.2.5 Ogrodje Vaadin

Vaadin [18] je odprtokodno aplikacijsko spletno ogrodje (ang. web framework) za gradnjo bogatih spletnih aplikacij (RIA). Ogrodje je sestavljeno iz spletnega aplikacijskega programskega vmesnika (ang. application programming interface) (API), številnih komponent uporabniškega vmesnika, tem za nadzor izgleda ter podatkovnega modela, ki dovoljuje vezanje komponent uporabniškega vmesnika neposredno na podatkovni izvor (slika 6). V ozadju se uporablja tudi končni vmesnik (ang. terminal adapter) za sprejemanje zahtev iz spletnih brskalnikov in generiranje odzivov z renderiranjem strani.



Slika 6: Arhitektura ogrodja Vaadin

Vsaka Vaadin aplikacija živi kot Java Servlet v Servlet vsebniku. Vstopna točka je razred, ki razširja komponento `Application` [19], ta kreira in upravlja z vsemi komponentami uporabniškega vmesnika. Uporabnikova interakcija z aplikacijo se obravnava preko dogodkovnih poslušalcev, poenostavljena pa preko povezave komponent neposredno s podatkovnim izvorom (npr. podatkovno bazo) (slika 6).



Slika 7:Arhitektura Vaadin aplikacije

Na odjemalčevi strani je ogrodje Vaadin zgrajeno okoli orodja GWT, ki je namenjeno razvoju AJAX aplikacij v jeziku Java. Programer piše kodo v Javi, ki je nato prevedena v HTML in Javascript preko GWT prevajalnika (ang. compiler).

2.2.5.1 Dodatek Wizards for Vaadin

Wizards for Vaadin (VFW) [20] je dodatek za Vaadin ogrodje s katerim je mogoče enostavno narediti čarovnika z več koraki. Za njegovo uporabo smo morali k našemu POM-u dodati Vaadinov repozitorij dodatkov (slika 8).

```
<repository>
  <id>vaadin-addons</id>
  <url>http://maven.vaadin.com/vaadin-addons</url>
</repository>
```

Slika 8:Delček XML za dodajanje Vaadin repozitorija

Sam dodatek VFW pa smo k projektni POM datoteki dodali z naslednjo odvisnostjo (slika 9)

```
<dependency>
  <groupId>org.vaadin.addons</groupId>
  <artifactId>wizards-for-vaadin</artifactId>
  <version>0.4.1</version>
</dependency>
```

Slika 9:Delček XML za dodajanje dodatka *Wizards for Vaadin*

Tipična izvedba čarovnika [21] nato izgleda takole (slika 10) :

```
// inicializacija Wizard-a(čarovnika)
Wizard carovnik = new Wizard();

// dodamo nekaj korakov ki implementirajo WizardStep interface
carovnik.addStep(new PrviKorak());
carovnik.addStep(new DrugiKorak());
carovnik.addStep(new TretjiKorak());
carovnik.addStep(new ZadnjiKorak());

// dodamo čarovnika na postavitev(angl. layout)
mainLayout.addComponent(wizard);
```

Slika 10:Tipična implementacija Wizards for Vaadin

2.2.6 Orodje TortoiseSVN

Pri razvoju aplikacije smo izvajali nadzor nad izvorno kodo z uporabo orodja TortoiseSVN [22]. Orodje deluje na platformno neodvisnem strežniku Apache Subversion [23]. Glavni značilnosti orodja sta vpogled v zgodovino programske kode in verzije projekta. Z vpogledom v zgodovino imamo na voljo pregled razvoja aplikacije, avtorjev posameznih segmentov kode, sprememb kode in možnost uporabe poljubnih starejših verzij v primeru, da smo v novi verziji kodo pokvarili. Vodenje inačic projekta poteka v povezavi z orodjem Apache Maven.

2.2.7 Orodje Maven

Apache Maven [24] je programsko orodje, ki se uporablja za gradnjo in upravljanje s projekti, ki so zasnovani v programskem jeziku Java, podpira pa tudi nekatere druge programske jezike, kot na primer C#, Scala in ruby. Princip gradnje projekta z orodjem Maven temelji na projektno objektnem modelu (ang. Project Object Model - POM) in naboru vtičev, ki so vgrajeni že v samem orodju. Nastavitve projekta so shranjene v enem ali več projektno objektnih modelih in sicer v datotekah `pom.xml`. Razvijalec ima vpogled v različico projekta, shranjene so odvisnosti med moduli oz. knjižnicami, omogoča nadzor nad Junit testi

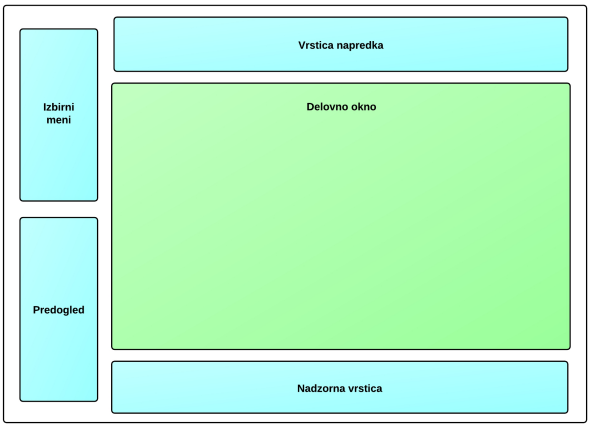
in drugo.

V tem delu je Maven uporabljen za hranjenje odvisnosti (ang. dependency) med moduli oz. knjižnicami in gradnjo aplikacije v `WAR` (ang. web application archive) datoteko. Gre za `JAR` datoteko s točno določeno strukturo, katero potem npr. JBoss aplikacijski strežnik sam razpozna, jo razširi in požene spletno aplikacijo.

3 Zasnova uporabniškega vmesnika

Uporabnik bo ob vsaki prvi interakciji z aplikacijo naletel na prazno delovno okno, izbirni meni z seznamom postopkov in oknom za predogled podatkov.

Ustrezni postopek poženemo z izbiro postopka v izbirnem meniju, ki se nato prikaže v delovnem oknu. Delovno okno smo v osnovi razdelili na tri dele, in sicer vrstico napredka, nadzorno vrstico in okno za prikaz trenutnega koraka. V vrstici napredka se prikazuje seznam korakov z trenutno izbranim korakom. V nadzorni vrstici se prikazujejo gumbi preko katerih uporabnik upravlja s postopkom: gumb za preklic postopka, gumba za naprej in nazaj ter gumb za zaključek postopka. Slika 11 prikazuje ogrodje uporabniškega vmesnika.



Slika 11: Zasnova uporabniškega vmesnika

Delovno okno se nadalje spreminja v odvisnosti od koraka v katerem se nahaja uporabnik. Iskalne obrazce smo razdelili na pol, kjer zgornja polovica deluje kot vnosni obrazec, spodnja polovica pa prikazuje ustrezne rezultate iskanja. Na koraku za izbiro pravnih položajev prikazujemo le potrditvena polja. Delovno okno za vnos izvedene pravice je kombinacija spustnega seznama in vnosnih polj.

Eden najpomembnejših delov vsakega postopka je korak za predogled, kjer uporabnik pred oddajo predloga pregleda vnesene podatke na postopku, v naši aplikaciji pa smo ta korak premestili v glavno okno, kjer bodo vneseni podatki vedno na ogled uporabniku.

4 Razvoj aplikativne rešitve

Za pridobitev podatkov in dostop do aplikacijske baze želimo uporabiti javanske razrede iz obstoječih aplikacij. Za tvorbo in upravljanje teh razredov je v času izvajanja aplikacije odgovorno ogrodje Spring, zato smo morali pripeljati Springovo *aplikacijsko okolje* (ang. applicaction context) [25] v Vaadin aplikacijo.

To smo storili tako, da smo ob inicializaciji Vaadin servleta obenem inicializirali še Springovo aplikacijsko okolje. Za kreiranje hierarhije POJO objektov smo uporabili infrastrukturo Springa, ki omogoča iskanje in kreiranje javanskih razredov anotiranih z `@Service`, `@Component` ali `@Repository` [26]. Springov mehanizem kontekstnega iskanja aktiviramo s pomočjo elementa v springovi kontekstni datoteki XML z imenom `component-scan`. Nad objekti, ki so definirani za skeniranje, se sproži tudi mehanizem *vpeljave odvisnosti* (ang. dependency injection) [27].

V naši aplikaciji tako kreirane POJO instance pridobimo preko klica metode `ApplicationContext.getBean(String beanName)` [28]. Getter metode za pridobitev na tak način kreiranih POJO objektov smo združili v razredu `VaadinBeanSupport` (slika 12).

```
public class VaadinBeanSupport {

  public static final String BEAN_NAME_CRP_DELEGATOR  = "crpDelegator";
  public static final String BEAN_NAME_PRS_DELEGATOR  = "prsDelegator";
  public static final String BEAN_NAME_GK_DELEGATOR   = "gkservice";
  public static final String BEAN_NAME_SIF_DELEGATOR  = "sifrantServiceClient";
  public static final String BEAN_NAME_CFG_PROVIDER   = "configurationProviderServiceClient";

  public static ICRPDelegator getCRPDelegator(){
    return (ICRPDelegator)AppContext.getApplicationContext().getBean(BEAN_NAME_CRP_DELEGATOR);
  }
```

Slika 12: Primer getterja za pridobljen POJO objekt

4.1 Postopek

Glavni razred naše aplikacije smo poimenovali `VaadinApp` in tako kot v vsaki Vaadin aplikaciji tudi ta razširja razred `Application`. Ta razred vzpostavi osnovno Vaadin infrastrukturo, ki omogoča zunanje odkrivanje in manipulacijo uporabnika, oken in tem ter zagon in zaustavitev aplikacije. Poleg tega ta razred implementira še vmesnik `ItemClickListener`, ki nam omogoča, da lovimo uporabnikove klike po aplikaciji. Ob inicializaciji razreda inicializiramo še glavno okno aplikacije (slika 12).

Vaadin ločuje izgled uporabniškega vmesnika od poslovne logike z uporabo tem. Teme lahko vključujejo CSS predloge, HTML postavitve po meri in vse potrebne grafične datoteke. V

```
@Configurable
@org.springframework.stereotype.Component(value = "vaadinApp")
@Scope("session")
public class VaadinApp extends Application implements ItemClickListener {
  private static final String THEME_NAME_RUNO = "runo";
  private PostopekView postopekView;
  private MainWindow mainWindow;

  @Override
  public void init() {
    setTheme(THEME_NAME_RUNO);
    buildMainLayout();
  }

  private void buildMainLayout() {
    mainWindow = new MainWindow();
    mainWindow.getTree().addListener((ItemClickListener)this);
    setMainWindow(mainWindow);
  }
```

Slika 13: Programska koda, ki inicializira Vaadin aplikacijo

naši aplikaciji smo nastavili prednaloženo temo Runo za prijetnejši izgled (slika 13).

Programska logika, ki skrbi za prikazovanje postopka, se nahaja v razredu `PostopekView`, ki razširja Vaadin element `Panel` in implementira vmesnik `WizardProgressListener` (slika 14), ki ponuja metode na podlagi katerih npr. vemo, kdaj je čarovnik preklican ali končan. Razred `PostopekView` hrani tudi instanco razreda `Wizard`, kateri je implementacija našega čarovnika. Koraki našega postopka so implementacija vmesnika `WizardStep`.

```
public class PostopekView extends Panel implements WizardProgressListener {
  private static final long serialVersionUID = -3231448423614834379L;

  private FirstStep firstStep;
  private SecondStep secondStep;
  private ThirdStep thirdStep;
  private FourthStep fourthStep;

  protected Wizard wizard;
  private WizardProgressBar progressBar;
  private VerticalLayout layout;

  public PostopekView() {
    firstStep = new FirstStep(this);
    secondStep = new SecondStep(this);
    thirdStep = new ThirdStep(this);
    fourthStep = new FourthStep();
    layout = new VerticalLayout();
    layout.setSizeFull();
    wizard = new Wizard();
    wizard.setUriFragmentEnabled(true);
    progressBar = new WizardProgressBar(wizard);
    wizard.addStep(firstStep);
    wizard.addStep(secondStep);
    wizard.addStep(thirdStep);
    wizard.addStep(fourthStep);
    wizard.addListener((WizardProgressListener)this);
    layout.addComponent(wizard);
    setSizeFull();
    setContent(layout);
  }
```

Slika 14: Programska koda za prikaz postopka

4.2 Vnosni obrazci

Kot rečeno so vnosni obrazci razviti s pomočjo Vaadin komponent, sam postopek pa sloni na Vaadin dodatku Wizards for Vaadin.

4.2.1 Vnos predlagatelja

Poslovna logika, ki skrbi za prikazovanje koraka za vnos predlagatelja v prvem koraku našega postopka se nahaja v razredu `S001Layout`. Ta razred deduje iz razreda `VerticalSplitPanel`, ki poskrbi za razdelitev zaslona po horizontali. Nadalje ta razred združuje razreda `S001Panel` in `S001ResultsPanel`. `S001Panel` skrbi za iskanje po registru CRP, deduje pa iz razreda `FormLayout`. `S001ResultsPanel` skrbi za prikazovanje rezultatov iskanja. Oba razreda preprosto nastavimo na `S001Layout` z metodama `setFirstComponent` in `setSecondComponent` (slika 15).

```
public class S001Layout extends VerticalSplitPanel {

  private S001Panel searchPanel;
  private S001ResultsPanel searchResultsPanel;

  public S001Layout() {
    searchPanel = new S001Panel();
    searchResultsPanel = new S001ResultsPanel();

    searchPanel.setHeight(100, Sizeable.UNITS_PERCENTAGE);
    searchResultsPanel.setHeight(100,Sizeable.UNITS_PERCENTAGE);

    setFirstComponent(searchPanel);
    setSecondComponent(searchResultsPanel);
    setSplitPosition(50);
  }
```

Slika 15: Uporaba metod `setFirstComponent` in `setSecondComponent` objekta `VerticalSplitPanel`

4.2.2 Vnos nepremičnine

Korak za vnos nepremičnine je v osnovi podoben koraku za vnos predlagatelja. Ravno tako imamo razred, ki razširja komponento `VerticalSplitPanel` (slika 16), na njem pa razreda, ki razširjata komponento `Panel`, kjer eden skrbi za vnos podatkov, drugi pa za prikazovanje rezultatov iskanja.

```
public class S002Layout extends VerticalSplitPanel {
  private static final Log log = LogFactory.getLog(S002Layout.class);
  private static final long serialVersionUID = 4461134507385545133L;

  private S002Panel searchPanel;
  private S002ResultsPanel searchResultsPanel;

  public S002Layout() {
    log.debug("init::s002 layout");
    searchPanel = new S002Panel();
    searchResultsPanel = new S002ResultsPanel();

    searchPanel.setHeight(100, Sizeable.UNITS_PERCENTAGE);
    searchResultsPanel.setHeight(100,Sizeable.UNITS_PERCENTAGE);
    setFirstComponent(searchPanel);
    setSecondComponent(searchResultsPanel);
    setSplitPosition(50);
  }
```

Slika 16: Razred za prikaz koraka izbire nepremičnine

Sledi korak za izbiro osnovnih pravnih položajev.

4.2.3 Izbira osnovnih pravnih položajev

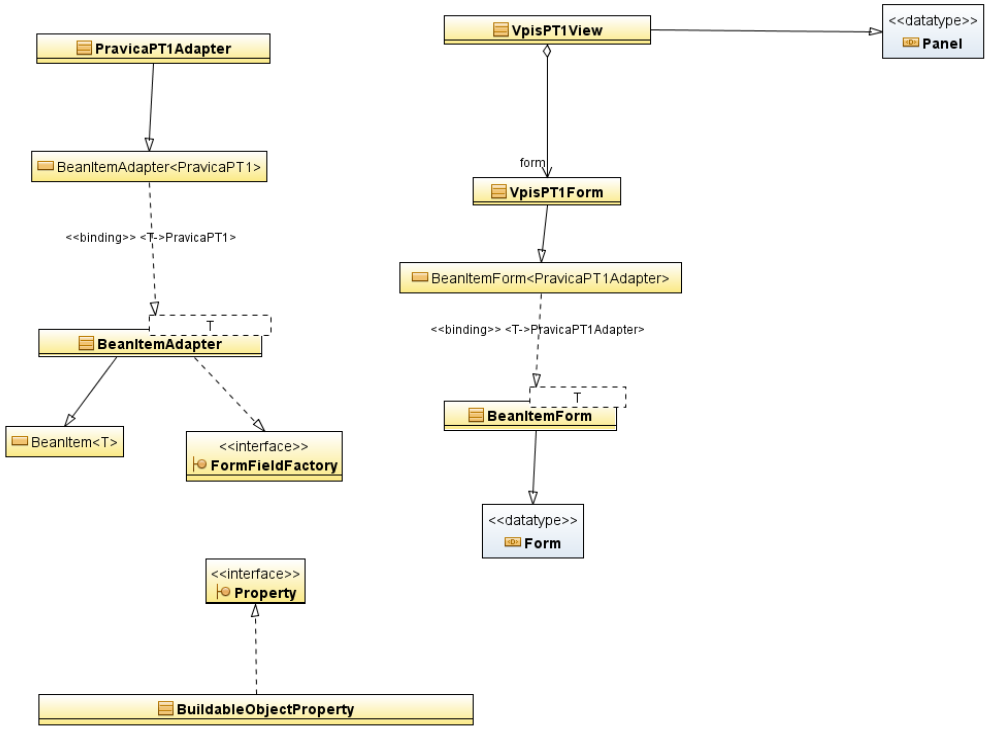
Korak za izbiro osnovnega pravnega položaja se nahaja v razredu `S003View`, ki razširja Vaadin komponento `Panel`. Za prikaz potrditvenih polj, ki predstavljajo posamezne položaje pa smo izbrali komponento `OptionGroup` (slika 17).

```
public class S003View extends Panel {
  private OptionGroup pravice;

  public S003View() {
    setCaption("izbira polozajev");
    setSizeFull();
    pravice = new OptionGroup("izberite polozaje");
    pravice.setMultiSelect(true);
    addComponent(pravice);
  }

  public OptionGroup getPravice() {
    return pravice;
  }
}
```

Slika 17: Uporaba `OptionGroup` komponente

Sledi vpis nove izvedene pravice.

4.2.4 Vnos izvedene pravice



Slika 18: Diagram razredov za dinamično formo

Korak za vpis pravice je v osnovi razširitev komponente `Form`, na njej pa se nahaja spustni seznam (komponenta `ComboBox`) z možnimi tipi pravice ter množico vnosnih polj tipa `Field`. `Form` komponenta omogoča enostavno tvorbo obrazcev, kjer so polja lahko generirana avtomatično in z podatkovnim virom vezanim na obrazec. `Field` je vmesnik za vse vnosne Vaadin komponente. `BeanItem` razred omogoča, da so lahko podatkovni viri tudi POJO objekti. Na sliki 18 vidimo diagram razredov, ki smo jih uporabili pri gradnji forme.

Vnosna polja smo želeli nastaviti pred inicializacijo, zato smo z abstraktnim razredom `BeanItemAdapter` [29] razširili razred `BeanItem` (ki je ovojni razred za dodajanje `Item` vmesnika h kateremukoli javanskemu zrnu) in redefinirali metode `getItemProperty`, `addItemProperty`, in `removeItemProperty`. Naša implementacija metode `GetItemProperty` vrača našo implementacijo razreda `Property` po imenu `BuildableObjectProperty`.

V razredu `BuildableObjectProperty` (slika 19) hranimo konfiguracijo za posamezno

vnosno polje, implementira pa vmesnik `Property` in posreduje klice na metode tega vmesnika preko reference na objekt pridobljen v konstruktorju.

```
public class BuildableObjectProperty implements Property {

  private static final long serialVersionUID = -2112159691038583343L;
  private Property property;
  private String displayName;
  private int width = 200;
  private boolean radio = false;
  private boolean richText = false;
  private boolean required = false;
  private Container dataSource;
  private String itemCaptionPropertyId;
  private boolean multi;
  private List<Validator> validators;
  private List<Object> options = new ArrayList<Object>();

  public BuildableObjectProperty(Property property) {
    this.property = property;
  }

  @Override
  public String toString() {
    return property.toString();
  }

    @Override
  public Class<?> getType() {
    return property.getType();
  }

    @Override
  public Object getValue() {
    return property.getValue();
  }
```

Slika 19: Razred za hranjenje konfiguracije vnosnega polja

Nadalje smo v razredu `BuildableObjectProperty` definirali lastnosti, ki jih lahko ima vnosno polje na formi, npr. naziv in širino. Te lastnosti nato nastavimo v metodi `initializeItemProps` (slika 20) razreda `PravicaPT1Adapter`, ki razširja razred `BeanItemAdapter`.

```
@Override
protected void initializeItemProps() {
  getItemProperty(DODATNI_OPIS).setDisplayName("dodatni opis").setRichText(true);
  getItemProperty(ZNESEK_TERJATVE).setDisplayName("znesek terjatve").addValidator(new DoubleValidator("Znesek mora biti cifra"));
  getItemProperty(OBRESTI).setDisplayName("obresti").addValidator(new DoubleValidator("Znesek mora biti cifra"));
  getItemProperty(VALORIZACIJA).setDisplayName("valorizacija");
  getItemProperty(POSTOPEK_ORGAN).setDisplayName("postopek organ");
  getItemProperty(POSTOPEK_OPR_ST).setDisplayName("postopek opr. št.");
  getItemProperty(VALUTA_TERJATVE).setDisplayName("valuta terjatve");

  List<? extends ISifrantElt> valute = VaadinBeanSupport.getSifDelegator().getSifranti(KonstDbSifranti.SIF_WS_VALUTE);
  BeanItemContainer<ISifrantElt> objects = new BeanItemContainer<ISifrantElt>(ISifrantElt.class, valute);

  getItemProperty(VALUTA_TERJATVE).setMulti(true);
  getItemProperty(VALUTA_TERJATVE).setDataSource(objects);
  getItemProperty(VALUTA_TERJATVE).setItemCaptionPropertyId("naziv");
  getItemProperty(VALUTA_TERJATVE).setWidth(200);

}
```

Slika 20: Implementacija metode za nastavitev vnosnih polj

Razred `BeanItemAdapter` implementira še vmesnik `FormFieldFactory`, ta pa ponuja le metodo `createField` (slika 21). V njej smo inicializirali z `BuildableObjectProperty` objekti prednastavljene `Field` objekte.

```
@Override
public Field createField(Item item, Object propertyId, Component uiContext) {
  BuildableObjectProperty property = (BuildableObjectProperty) item.getItemProperty(propertyId);

  Field field = null;

  if (property.isMulti()) {
    if (property.isRadio() == false) {
      ComboBox comboBox = new ComboBox();
      comboBox.setNullSelectionAllowed(false);
      comboBox.setNewItemsAllowed(false);

      if(property.getDataSource() != null) {
        comboBox.setContainerDataSource(property.getDataSource());
        comboBox.setItemCaptionPropertyId(property.getItemCaptionPropertyId());

      } else if(CollectionUtils.isNotEmpty(property.getOptions())){
        for (Object option : property.getOptions()) {
          comboBox.addItem(option);
        }
      }
      comboBox.setImmediate(true);
      comboBox.setNullSelectionAllowed(false);
      field = comboBox;
    } else {
      OptionGroup optionGroup = new OptionGroup();
      for (Object option : property.getOptions()) {
        optionGroup.addItem(option);
      }

      field = optionGroup;
    }
  } else {

    if (property.getType() == Date.class) {
      PopupDateField dateField = new PopupDateField();
      dateField.setResolution(PopupDateField.RESOLUTION_DAY);
      field = dateField;
    } else if (property.isRichText()) {
      TextArea richTextArea = new TextArea();
      richTextArea.setNullRepresentation("");
      field = richTextArea;
    } else {
      TextField textField = new TextField();
      textField.setNullRepresentation("");
      field = textField;
    }
  }
```

Slika 21: Implementacija metode za tvorbo vnosnih polj

Ob izbiri vrednosti v spustnem seznamu se sproži asinhroni klic metode `ValueChange` v formi, ki osveži nabor vnosnih polj, katere so na voljo.

To stori tako, da glede na izbrano vrednost iz baze pridobi objekt s konfiguracijskimi podatki, nato pa iterira skozi polja iz adapterja na naši formi in objekt z konfiguracijo tega polja pridobi z klicem metode `getVpisPodatkov().getClass().getDeclaredField (naziv polja)`. Lastnosti vnosnega polja (kot sta prikaz polja ali obveznost vpisa) smo nato prilagodili z metodo `setDisplayRule`.

```
typeSelect.addListener(new Property.ValueChangeListener() {

  @Override
  public void valueChange(Property.ValueChangeEvent event) {
    VaadinWebUtils.showNotificationMessage(getWindow(), "izbrano: " + event.getProperty());

    Integer valueId = (Integer) event.getProperty().getValue();

    PT1Config config = VaadinBeanSupport.getCFGProvider().getConfigForPravicaPt1(valueId);

    for (Object prop : getItemDataSource().getItemPropertyIds()) {
      Field field = getField(prop);
      try {
        java.lang.reflect.Field tipVpisaField = config.getVpisPodatkov().getClass().getDeclaredField((String) prop);

        if (tipVpisaField.getName().equals(prop)) {
          try {
            tipVpisaField.setAccessible(true);
            TipVpisa tipVpisa = (TipVpisa) tipVpisaField.get(config.getVpisPodatkov());
            setDisplayRule(tipVpisa, field);
          } catch (IllegalArgumentException ex) {
            log.error(ex);
          } catch (IllegalAccessException ex) {
            log.error(ex);
          }
        }
      } catch (NoSuchFieldException ex) {
        log.error(ex);
      } catch (SecurityException ex) {
        log.error(ex);
      }
    }
  }
});
```

Slika 22: Implementacija metode za osvežitev nabora vnosnih polj

V razredu `PravicaPT1Adapter` hranimo konstante z imeni vseh vnosnih polj, ki se lahko vpišejo. Imena teh polj in nastavitvene podatke smo povezali skupaj preko *reflekcije* (ang. reflection) (slika 22). Reflekcija je mehanizem preko katerega Java izpostavi lastnosti razreda med njegovim izvajanjem, s čimer dovoli Java programom oštevilčenje in dostop do metod, polj in konstruktorjev tega razreda.

4.3 Predogled

Okno za predogled smo si zamislili kot del glavnega menija, na njem pa hranimo osnovne podatke,ki jih je uporabnik vnesel skozi postopek. Razred `MenuLayout` razširja komponento `VerticalLayout`, nanj pa z metodo `setFirstComponent` nastavimo objekt `NavigationTree`, ki hrani seznam postopkov ki so na voljo.

```
public class MenuLayout extends VerticalLayout {

  private NavigationTree tree = new NavigationTree();
  private PredogledPanel predogledPanel = new PredogledPanel();
  private VerticalSplitPanel menuSplitPanel;

  public MenuLayout() {
    menuSplitPanel = new VerticalSplitPanel();
    this.setSizeFull();
    this.addComponent(menuSplitPanel);
    this.setExpandRatio(menuSplitPanel, 1);

    menuSplitPanel.setFirstComponent(tree);
    menuSplitPanel.setSecondComponent(predogledPanel);
  }
```

Slika 23: Razred za prikaz menija

Z metodo `setSecondComponent` pa nastavimo objekt `PredogledPanel`, ki hrani podatke, ki smo jih pridobili skozi postopek.

```
public class PredogledPanel extends Panel {

  private Label emso;
  private Label nepremicnina;
  private Label polozaji;
  private Label pravice;

  public PredogledPanel() {
    addComponent(new Label("PREDOGLED"));
    emso = new Label();
    emso.setCaption("emso");
    nepremicnina = new Label();
    nepremicnina.setCaption("nepremicnina");
    polozaji = new Label();
    polozaji.setCaption("polozaji");
    pravice = new Label();
    pravice.setCaption("pravice");
    this.addComponent(emso);
    this.addComponent(nepremicnina);
    this.addComponent(polozaji);
    this.addComponent(pravice);
  }
```

Slika 24: Razred za prikaz predogleda

4.4 Analiza

Po koncu razvoja lahko sklenemo, da je delo z ogrodjem Vaadin pozitivna izkušnja. Dejstvo da temelji na ogrodju GWT pomeni, da je veliko funkcionalnosti, ki nam jih ponuja, že prirojenih in je potrebno programerju razumeti le sintakso programiranja v samem ogrodju.

Razvita aplikacija je že nekaj časa nameščena v razvojnem okolju eZK in prvi odzivi uporabnikov so nad uporabniško izkušnjo in funckionalnostmi, ki jih aplikacija ponuja pozitivni. Morebitne kritike se nanašajo le na funkcionalnosti, ki v dejanski aplikaciji že obstajajo, pa nam je zmanjkalo časa da bi jih razvili tudi v naši aplikaciji.

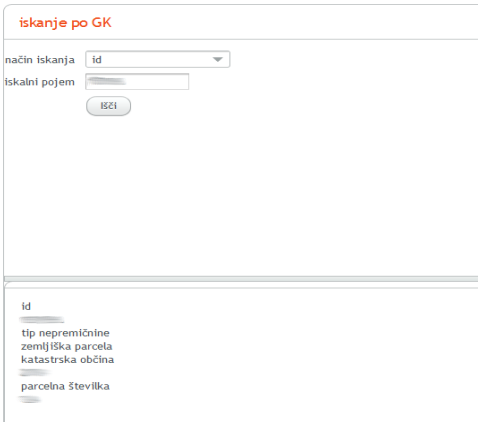
5 Primer uporabe aplikacije

Prvi korak, ki ga moramo izvesti za pripravo predloga je izbira predlagatelja. Kot predlagatelja lahko izberemo le fizično osebo. V ta namen lahko kot iskalni kriterij uporabimo številko EMŠO.

Kot rezultat iskanja se nam izpišejo osnovni podatki izbrane osebe (slika 25). Na ta način se prepričamo, da je to res oseba, ki jo želimo izbrati kot predlagatelja ZK predloga.

Slika 25: Korak za izbiro predlagatelja

Nato v navigacijski vrstici izberemo `Naprej`, da prikličemo naslednji korak oziroma izbiro nepremičnine. Forma za iskanje nepremičnine je v osnovi enaka kot pri iskanju predlagatelja. V iskalno polje vpišemo `ID` nepremičnine, ki naj bo predmet zemnljiškoknjižnega predloga. Kot rezultat iskanja se izpišejo osnovni podatki o izbrani nepremičnini (slika 26).

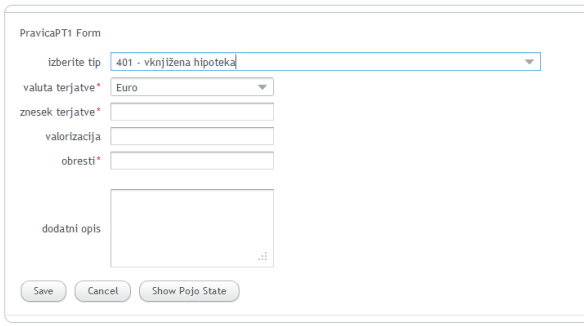


Slika 26: Korak za izbiro nepremičnine

Ko se prepričamo da smo poiskali pravo nepremičnino, izberemo `Naprej`.

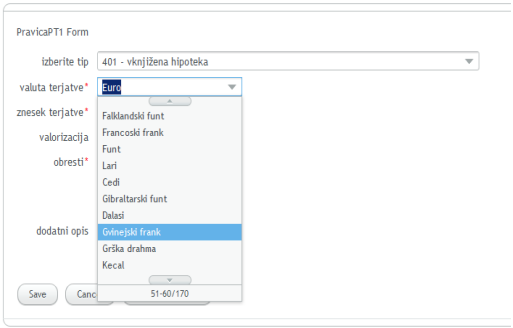
Slika 27:Izbira položajev

Na vsako nepremičnino je v osnovi vezanih 1 : n osnovnih pravnih položajev, zato je naslednji korak izbira položajev na katere želimo vezati novo izvedeno pravico (slika 27). Po izbiri položajev izberemo `Naprej`.



Slika 28:Korak za vpis izvedene pravice

Na formi za vnos nove izvedene pravice v šifrantu tipov pravic izberemo željeni tip (slika 28). Na podlagi izbranega tipa pravice se prikažejo vnosna polja, ki jih potrebujemo za vpis pravice. Polje za vnos zneska terjatve ima na voljo tudi šifrant svetovnih denarnih valut (slika 29), tako kot nekatera druga polja pa je validirano glede na zahtevan (numeričen) tip podatka. Nekatera polja so za vpis pravice potrebna in so ustrezno označena z zvezdico (slika 28).



Slika 29: Izbira iz šifranta valut

Ko smo zaključili z vnosom podatkov, izberemo `Zakljuci`.

6 Sklepne ugotovitve

V diplomskem delu je opisan razvoj spletnega čarovnika z odprtokodnim ogrodjem Vaadin. Razvoj same rešitve je bil zelo zanimiv, saj razvoja aplikacij z ogrodjem Vaadin pred tem nismo poznali. Zahvaljujoč predznanju programiranja v programskem jeziku Java pa je bil sam proces učenja ogrodja zelo hiter. Poleg tega je naloga zajemala še namestitev in nastavljanje aplikacijskega strežnika.

Uspešno nam je uspelo razviti vse zadane funkcionalnosti, ki jih najdemo v tipičnem eZK postopku. Z aplikacijo lahko uporabnik doda predlagatelja iz CRP registra, izbere nepremičnino, izbere osnovne pravne položaje ter na koncu v dinamični formi vpiše podatke za novo izvedeno pravico. Aplikacija je tudi razširljiva, saj si lahko zamislimo nov korak k postopku in ga enostavno realiziramo. Znanje, ki smo ga pridobili pri izdelavi diplomske naloge, bomo s pridom uporabili na bodočih projektih.

Da bi aplikacija imela uporabno vrednost, bi morali iz tako sestavljenega predloga sestaviti XML datoteko, s čimer bi lahko implementirali funkcionalnost oddaje predloga. Tako sestavljen predlog bi lahko zapisali v bazo. Zaenkrat v aplikaciji ni implementirana nobena vrsta prijave v sistem, zaradi večje varnosti bi bilo bolje da bi v morebitnem nadaljnem razvoju implementirali prijavo z uporabniškim imenom in geslom, še bolje pa z digitalnim certifikatom, tako kot v dejanski eZK aplikaciji.

Izdelana rešitev lahko služi kot pilotna aplikacija za druge projekte, predvsem take, kjer je potrebno razviti različne vrste postopkov.

7 Literatura

[1] Wikipedia contributors. Rich internet application. Wikimedia Foundation, Inc. Dostopno na http://en.wikipedia.org/wiki/Rich_Internet_application

[2] Wikipedia contributors. Ajax. Wikimedia Foundation, Inc. Dostopno na http://en.wikipedia.org/wiki/Ajax_(programming)

[3] Rich internet application screen design. Dostopno na http://uxmag.com/articles/rich-internet-application-screen-design

[4] E- zemljiška knjiga. Dostopno na https:/evlozisce.sodisce.si

[5] Wikipedia contributors. Model-view-controller. Wikimedia Foundation, Inc. Dostopno na http://en.wikipedia.org/wiki/Model-view-controller

[6] JBoss AS. Dostopno na http://www.jboss.org

[7] Apache Tiles. Dostopno na http://tiles.apache.org

[8] Spring Framework. Dostopno na http://www.springsource.org/

[9] Spring Webflow. Dostopno na http://www.springsource.org/spring-web-flow

[10] Hibernate. Dostopno na http://www.hibernate.org/

[11] Wikipedia contributors. Java Development Kit. Dostopno na http://en.wikipedia.org/wiki/Java_Development_Kit

[12] Oracle Weblogic Server. Dostopno na http://www.oracle.com/technetwork/middleware/weblogic/overview/index.html

[13] WebSphere software. Dostopno na http://www-01.ibom.com/software/websphere/

[14] Wikipedia contributors. Inversion of control. Wikimedia Foundation, Inc. Dostopno na http://en.wikipedia.org/wiki/Inversion_of_control

[15] J.Jamae, P.Johnson, »JBoss in Action«, Greenwich, CT, Manning, 2011

[16] F. Marchioni, »JBoss AS 7 Configuration, Deployment, and Administration«, Birmingham, UK, Packt Publishing, 2011

[17] Google Web Toolkit. Dostopno na https://developers.google.com/web-toolkit/

[18] Vaadin Framework. Dostopno na http://vaadin.com

[19] M. Gronroos, Book of Vaadin: 4th Edition, Vaadin Ltd, 2010

[20] Wizards for Vaadin.
Dostopno na https://vaadin.com/directory#addon/wizards-for-vaadin

[21] Wizards for Vaadin manual.
Dostopno na http://code.google.com/p/wizards-for-vaadin/wiki/GettingStarted

[22] TortoiseSVN. Dostopno na http://tortoisesvn.net/

[23] Apache Subversion. Dostopno na http://subversion.tigris.org/

[24] Apache Maven. Dostopno na http://maven.apache.org/

[25] C.Ho, R. Harrop, »Pro Spring 3«, Apress,2012

[26] Spring framework reference documentation. Dostopno na http://static.springsource.org/spring/docs/

[27] Wikipedia contributors. Dependency injection. Wikimedia Foundation, Inc.
Dostopno na http://en.wikipedia.org/wiki/Dependency_injection

[28] Access the Spring-ApplicationContext from everywhere in your application. Dostopno na http://blog.jdevelop.eu/2008/07/06/access-the-spring-applicationcontext-from-everywhere-in-your-application/

[29] Easy form creation with builder patterns in Vaadin. Dostopno na
http://www.aceevo.com/easy-form-creation-with-builder-patterns-in-vaadin/

[30] Interno gradivo, projekt E-zemljiška knjiga - SRC d.o.o.