POJEDNOSTAVLJENJE ODRŽAVANJA ORACLE FORMS APLIKACIJA SMJEŠTANJEM PREZENTACIONE LOGIKE U BAZU PODATAKA

SIMPLIFICATION IN MAINTAINING ORACLE FORMS APPLICATIONS STORING PRESENTATION LOGIC IN THE DATABASE

Jasmin Heljić
JP Elektroprivreda BiH d.d. – Sarajevo
Vilsonovo šetalište 15, Sarajevo, Bosna i Hercegovina
Telefon: +38761157445
E-mail: j.heljic@epbih.ba

Emina Krešťalica
JP Elektroprivreda BiH d.d. – Sarajevo
Vilsonovo šetalište 15, Sarajevo, Bosna i Hercegovina
Telefon: +38761818875
E-mail: em.krestalica@epbih.ba

SAŽETAK

Čest slučaj tokom održavanja aplikacija razvijenih u Oracle Forms je izmjena korisničkog interfejsa na način da je potrebno prekomponovati ili redizajnirati prikaz određenih polja na formi, nametnuti ograničenja na poljima, zamijeniti pozicije i sl. U ovom članku govorimo o smještanju prezentacione logike u bazu podataka u cilju jednostavnijeg održavanja korisničkog interfejsa na primjeru aplikacija razvijenih u JP Elektroprivreda BiH u Oracle Forms razvojnom alatu.

ABSTRACT

During maintaining Oracle Forms applications we are often faced with requests related to user interface changes in a way to redesign presentation of certain fields on the form, impose fields constraints, change fields positions etc. In this article we talk about some examples from JP Elektroprivreda BiH (Public Enterprise Electric Utility of Bosnia and Herzegovina) about storing presentation logic in the database in order to simplify maintenance process.

UVOD

Na početku članka je potrebno naglasiti jednu bitnu karakteristiku razvoja softvera: softverski proizvod se ne može posmatrati na isti način kako se posmatra materijalni proizvod. Filozofija razvoja softvera se u početku bazirala na pretpostavci da je razvoj softvera jasno definisan proces analogan fizičkoj aktivnosti poput izgradnje zgrada ili mostova, što je za posljedicu imalo tretiranje softverskog proizvoda analogno materijalnom. Vrijeme je pokazalo da je takav pristup pogrešan. Softverski proizvod u čijoj osnovi se nalazi sublimirano znanje o određenim procesima ili pojavama podložan je promjenama kako tokom razvoja tako i nakon toga, dok s druge strane materijalni proizvod poput građevine ostaje statičan. Prije same izgradnje napravi se detaljan plan, koji tačno definiše kako će izgledati finalni proizvod i kako treba da teče cijeli proces razvoja, što uključuje niz dokumenata, skica, matematičkih proračuna, dijagrama, a nakon toga dolazi izgradnja koja se izvodi striktno prema napravljenom planu [1].

Jedna od karakteristika razvoja softvera je da programi zahtijevaju održavanje. Preciznije rečeno studije o razvoju softvera ukazuju da za većinu programa, plaćanje programera da održavaju softver nakon što je razvijen čini između 80 i 90 posto ukupnih troškova [2]. Vrlo je teško precizirati šta održavanje softvera predstavlja. Sama ideja na prvu zvuči bizarno. Ako se o njoj razmišlja u terminima održavanja materijalnih dobara, naprimjer automobila ili mosta, održavanje se radi kada se nešto pokvari - u slučaju dotrajalog materijala, pokvarenog dijela, pregorenih kablova i sl. Ni jedna od

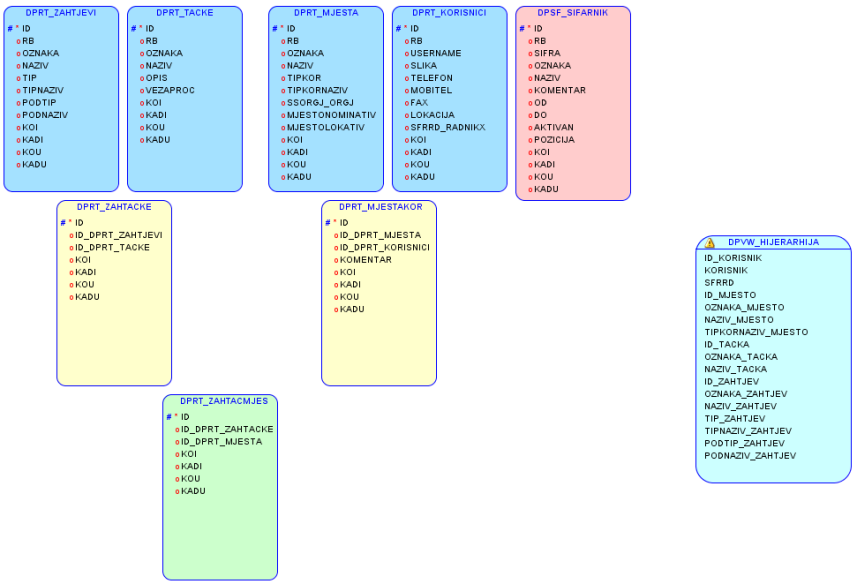
navedenih situacija se ne može primijeniti na softver. Stalno korištenje softvera neće uzrokovati njegovu dotrajalost kao što je slučaj sa materijalnim proizvodima.

Softver je potrebno održavati iz dva principijelna razloga. Čak i nakon priličnog testiranja, ili u nekim slučajevima nakon godina produkcijskog korištenja, bagovi i dalje mogu opstati u originalnom kodu. Debagiranje je esencijalni dio softverskog održavanja, međutim nije najvažniji. Mnogo važniji dio, u pogledu učešća u ukupnom trošku programskog održavanja, je poboljšanje programskih karakteristika. Programi se pišu kako bi se koristili. U jednu ruku softver obavlja zadatak brže nego što bi ga korisnik sam obavljao, a u drugu ne radi sve što bi korisnik htio da radi. Nakon određenog perioda korištenja, korisnik dolazi s novim zahtjevima: poboljšati određeni segment softvera, napraviti novu funkcionalnost, napraviti korisniji prikaz podataka, povezati softver s nekim drugim softverom koji se koristi u istom radnom okruženju, ubrzati program, dodati nove karakteristike itd. Pošto je softver poprilično fleksibilan dobavljači mogu odgovoriti na postavljene zahtjeve. U svakom slučaju, bilo da se radi o debagiranju ili dodavanju novih funkcionalanosti potrebni su stručni ljudi i vrijeme za takav posao. Softversko održavanje je teško zbog činjenice da originalni kôd najčešće pišu jedni programeri, a održavaju drugi. Softverski inžinjering potencira pisanje kôda koji će biti razumljiv drugim programerima. Nažalost u praksi se često dešava obrnuta situacija: nerazumljiv kôd kojeg održavaju programeri koji ga nisu pisali. Dobar stil pisanja kôda je kritičan za efikasno održavanje softvera.

1. INFORMACIONI SISTEM U KOMPANIJI

Informacioni sistemi u velikim korporacijama vremenom postanu složeni i najčešće heterogeni, što dodatno usložnjava održavanje. Naredni problem koji je prisutan u vezi održavanja informacionih sistema je raspoloživost ljudskih resursa. Primjetan je manjak visoko kvalifikovanog kadra u ovoj branši na tlu bivše Jugoslavije, te je stoga veoma poželjno optimizirati proces softverskog održavanja.

Oracle tehnologije se često upotrebljavaju u velikim korporacijama u BiH, što je slučaj i sa JP Elektroprivreda BiH d.d. – Sarajevo. Kao dugogodišnji developeri u Oracle tehnologijama (sa naglaskom na Oracle Forms i Reports) došli smo do zaključka da i minimalna optimizacija u konstrukciji i arhitekturi aplikacija predstavlja značajnu uštedu u kasnijem održavanju. Informacioni sistemi u JP Elektroprivreda BiH d.d. – Sarajevo većinom imaju za cilj da ubrzaju ili optimiziraju poslovne procese koji imaju sličnu sekvencijalnu strukturu, te se na osnovu ove karakteristike mogu kreirati određeni generički šabloni transkacija. Na ovu činjenicu je utjecala i organizaciona šema koja je dosta robusna i predvidiva, hijerarhijski uređena i jednostavna za relaciono opisivanje.

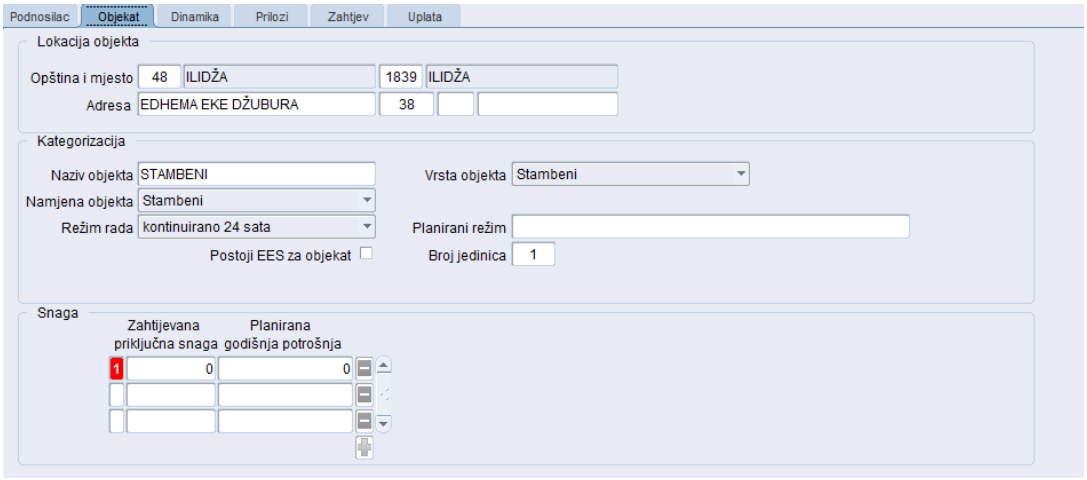


Slika 1. Primjer logičkog modela aplikacije predstavljen u Oracle SQL Developer Data Modeleru

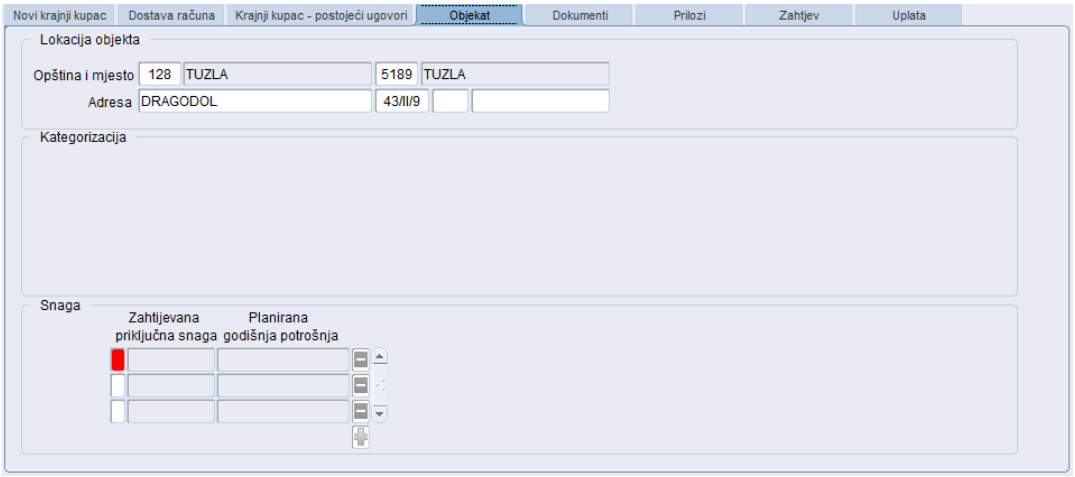
Kako bismo pojednostavili održavanje formi namijenjenih započinjanju poslovnih procesa iz iste grupe/kategorije pristupilo se detaljnoj analizi modela podataka i identificiranju zajedničkih komponenti, što nas je dovelo do zaključka da se projektovanje aplikacija iz iste grupe treba posmatrati sveobuhvatno. Takav pristup je rezultirao kreiranjem generičke osnove čije komponente su predstavljene matematičkim relacijama u bazi podataka.

2. KORISNIČKI INTERFEJS

Značajno pojednostavljenje održavanja korisničkog interfejsa moguće je postići kreiranjem generičkih formi koje mogu biti iskorištene u različitim poslovnim procesima na način da se ponašanje komponenti na tim formama definiše programabilno. Jedan od primjera takvih formi su ulazi za proces evidencije različitih tipova zahtjeva na prijemnim tačkama. U JP Elektroprivreda BiH d.d. – Sarajevo postoji mnogo kategorija poslovnih procesa koji imaju sličan ulaz. Naprimjer u djelatnosti distribucije se podnose različiti tipovi zahtjeva iz segmenta izdavanja elektroenergetskih saglasnosti u postupku priključenja kupaca na elektrodistributivnu mrežu. Podnosilac zahtjeva je dužan obezbijediti podatke prema definisanom formularu koji je primjeren njegovom tipu zahtjeva prema Distributeru. Dio podataka je obavezan.[3]

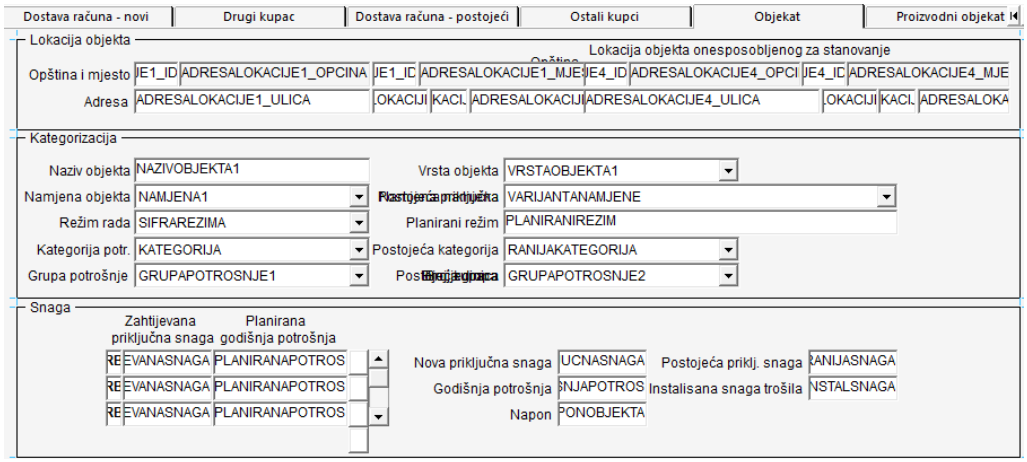


Slika 2. Primjer ulazne forme po zahtjevu za izdavanje prethodne elektroenergetske saglasnosti



Slika 3. Primjer ulazne forme po zahtjevu za raskid postojećeg i sklapanje novog ugovora

Kako se vidi iz prethodnog primjera, ulazna forma za dva različita tipa zahtjeva u sklopu iste kategorije mijenja svoj oblik prikazivanjem ili sakrivanjem određenih itema. Ovo se može proširiti i sa drugim vidovima transformacije itema (pomjeranje na zadate koordinate, promjena veličine, manipulacija vizuelnim identitetom). To je omogućeno kreiranjem jedinstvene forme sa svim potrebnim ulaznim komponentama čije ponašanje je uslovljeno vrstom zahtjeva.



Slika 4. Izgled ulazne forme za evidenciju zahtjeva u Oracle Forms

Iz prethodnog primjera se vidi da oblik ulazne forme za prvi tip zahtjeva sadrži iteme iz segmetna kategorije zahtjeva (naziv objekta, vrsta, namjena i sl.) za razliku od oblika forme za drugi tip zahtjev gdje se ovi itemi uopšte ne prikazuju. Također po prvom tipu zahtjeva blok u koji se unose podaci o snazi je omogućen, dok je po drugom tipu zahtjeva onemogućen.

3. SETOVANJE SVOJSTAVA ITEMA

Korištenjem built-in procedure SET_ITEM_PROPERTY možemo modifikovati sve instance itema u bloku promjenom specifičnog svojstva. Polimorfizam procedure SET_ITEM_PROPERTY omogućava različite načine korištenja procedure u zavisnosti od svojstva koga prosljeđujemo i potreba samog kodiranja. Identifikaciju itema možemo uraditi navođenjem ID-a ili naziva.

```
SET_ITEM_PROPERTY(item_id ITEM, property NUMBER, value VARCHAR2)
SET_ITEM_PROPERTY(item_id ITEM, property NUMBER, x NUMBER)
SET_ITEM_PROPERTY(item_id ITEM, property NUMBER, x NUMBER, y NUMBER)
SET_ITEM_PROPERTY(item_name VARCHAR2, property NUMBER, value VARCHAR2)
SET_ITEM_PROPERTY(item_name VARCHAR2, property NUMBER, x NUMBER)
SET_ITEM_PROPERTY(item_name VARCHAR2, property NUMBER, x NUMBER, y NUMBER)
```

Slika 5. Sintaksa SET_ITEM_PROPERTY built-in procedure u Oracle Forms

Pomoću SET_ITEM_PROPERTY built-in procedure možemo upravljati velikim brojem svojstava itema. Neka od tih svojstava su svojstva koja se odnose na izgled fonta (FONT_NAME, FONT_SIZE, FONT_SPACING, FONT_STYLE, FONT_WEIGHT), prompt itema (PROMPT_DISPLAY_STYLE, PROMPT_FONT_NAME, PROMPT_FOREGROUND_COLOR), manipulaciju itemom (ENABLED, UPDATE_ALLOWED, UPDATE_NULL, UPDATE_COLUMN), navigaciju, veličinu, poziciju itd.

4. POHRANJIVANJE PONAŠANJA ITEMA U BAZI PODATAKA

Jedan od načina da pojednostavimo manipulaciju svojstvima je pomoću tabele sastavljene od kolona koje predstavaljaju svojstva i unesnih vrijednosti koje definišemo u zavisnosti od vrste zahtjeva/procesa [PROCESS_TYPE]. Primjer definicije tabele svojstava je dat u nastavku.

```
Name                          Type
----------------------------  --------------------
ID                            NUMBER
PROCESS_TYPE                  VARCHAR2(64)
FORM                          VARCHAR2(64)
BLOCK                         VARCHAR2(64)
ITEM                          VARCHAR2(64)
ENABLED                       VARCHAR2(64)
REQUIRED                      VARCHAR2(64)
ALIGNMENT                     VARCHAR2(64)
AUTO_HINT                     VARCHAR2(64)
AUTO_SKIP                     VARCHAR2(64)
BACKGROUND_COLOR              VARCHAR2(64)
CASE_RESTRICTION              VARCHAR2(64)
FORMAT_MASK                   VARCHAR2(64)
HINT_TEXT                     VARCHAR2(64)
LABEL                         VARCHAR2(64)
NEXT_NAVIGATION_ITEM          VARCHAR2(64)
PREVIOUS_NAVIGATION_ITEM      VARCHAR2(64)
VISIBLE                       VARCHAR2(64)
PROMPT_TEXT                   VARCHAR2(64)
PROMPT_TEXT_ALIGNMENT         VARCHAR2(64)
X_POSITION                    NUMBER
Y_POSITION                    NUMBER
```

Slika 6. Primjer tabele za pohranjivanje vrijednosti svojstava itema

Na slici 7. je dat primjer sadržaja tabele za primjere ulaznih formi prikazanih na slikama 2 i 3. Na primjeru itema NAZIVOBJEKTA1 vidimo da se njegovo ponašanje za različite vrste zahtjeva (kolona PROCESS_TYPE) postavlja unosom dozvoljenih vrijednosti za određena svojstva itema.

ID	PROCESS_TYPE	FORMA	ITEM	ENABLED	REQUIRED	VISIBLE	ALIGNMENT	WIDTH	HEIGHT	AUTO_HINT	AUTO_SKIP	BACKGROUND_COLOR	HINT_TEXT	LABEL	NEXT_NAVIGATION_ITEM
4708	Z06	ZAHTJEVI	NAZIVOBJEKTA1	TRUE	FALSE	TRUE	(null)	420	(null)	(null)	(null)	(null)	(null)	(null)	ADRESALOKACIJE1_ID
1711	Z01	ZAHTJEVI	NAZIVOBJEKTA1	TRUE	FALSE	TRUE	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)
6481	Z15	ZAHTJEVI	NAZIVOBJEKTA1	FALSE	FALSE	FALSE	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)

Slika 7. Sadržaj tabele na primjeru jednog itema

Prilikom pokretanja forme možemo učitati vrijednosti iz baze podataka o vrijednostima svojstava itema koji se nalaze na formi u zavisnosti od vrste zahtjeva (procesa). Jedan od načina da to uradimo jeste da kreiramo proceduru za setovanje koju pozivamo iz triggera WHEN-NEW-FORM-INSTANCE. Potpis procedure može izgledati kao na slici 8.

```
procedureSetForm(process_type VARCHAR2)
```

Slika 8. Potpis procedure za setovanje svojstava itema na formi u zavisnosti od vrste procesa

Prvi dio procedure podrazumijeva dobavljanje vrijednosti svojstava za sve iteme koji se nalaze na odabranoj formi i smještanje u kursor. U opisu tabele sa slike 6. možemo uočiti kolonu ITEM u koju smještamo naziv itema sa forme. Nakon dobavljanja podataka iz tabele, petljom prolazimo kroz kursor i čitamo vrijednosti svojstava za svaki od itema sa forme. Strukturu tabele možemo granulirati prema potrebama, naprimjer dodavanjem bloka za slučaj da se radi o formi sa

više blokova. U tom slučaju prosljeđivanje parametara u proceduru SET_ITEM_PROPERTY koja vrši manipulaciju nad itemom za odabrano svojstvo može izgledati kao na slici 9.

```
SET_ITEM_PROPERTY(rec1.block||'.'||rec1.item, ENABLED, PROPERTY_TRUE);
```

Slika 9. Prosljeđivanje itema iz specifičnog bloka u SET_ITEM_PROPERTY čitanjem rekorda rec1 kursora

5. ZAKLJUČAK

Ideja opisana u članku je jednostavna i odnosi se na pojednostavljenje održavanja formi koje se mogu strukturirati na uopšten način za poslovne procese iz iste i/ili slične kategorije. Prezentaciona logika odnosno ponašanje itema na formi se umjesto na aplikativnom nivou može pohraniti u bazu podataka tj. tabele koje sadrže vrijednosti svojstava itema. Prilikom instanciranja formi definisane vrijednosti svojstava itema se dobavljaju u aplikativne strukture. Mehanizam za setovanje itema (koji može biti kreiran u vidu jedne ili više procedura) koristi dobavljene vrijednosti i vrši setovanje itema. Na ovaj način izmjene ponašanja itema ne moramo raditi direknto u kodu, nego u bazi podataka jednostavnim unošenjem vrijednosti u predefinisane kolone.

LITERATURA

[1] Boris Mešetović. "Razvoj softvera na dijeti". INFO magazin (2006)
[2] Eric S. Roberts. *The Art & Science of Java - An Introduction to Computer Science*. Prentice Hall, 2007, pp. 4-14.
[3] Proceduralni okvir za postupak priključenja kupaca i proizvođača na ED mrežu u JP Elektroprivreda BiH d.d. - Sarajevo