

BAZE PODATAKA [PROJEKTI ZADATAK: SISTEM ZA IZDAVANJE KREDITA U BANCIMA]

OPIS FUNKCIONALNOSTI

Uvod

U sklopu ovog dijela projektnog zadatka neophodno je dati opis svih formi projekta sa opisom svih njihovih funkcionalnosti na sljedeći način:

- Dati izgled forme sa svim kontrolama (U slučaju da se neke od kontrola skrivaju pri izvršenju programa neophodno je tu formu prikazati više puta da bi se istakle sve te funkcionalnosti);
- Opisati svaku od funkcionalnosti na formi i po potrebi ubaciti određeni broj dijagrama kako bi se te funkcionalnosti bolje opisale;
- Za svaku od funkcionalnosti dati određeni dio programskog koda koji izvršava tu funkcionalnost;
- Dostaviti testne primjere. (Potrebno je precizno i iscrpno objasniti navedene primjere i logiku upotrijebljenih primjera);
- Mogući i potencijani problemi kod upotrebe iste. Opisati potencijalne probleme koje očekujemo kod upotrebe date forme i način kako će dati problemi biti prevaziđeni.

Sistem za izdavanje kredita u banci, kako je zamišljeno u ovom projektnom rješenju, sastoji se od dva podsistema. Prvi od njih (koji je namjenjen uposlenicima banke) biće rađen kao desktop aplikacija i služiće kao aplikacija kadrovskoj službi. Drugi dio sistema koji će biti realizovan upotrebom web tehnologija biće na raspolaganju i uposlenicima banke, ali i korisnicima usluga web bankarstva. U sklopu ovog dijela projekta daćemo opis formi desktop aplikacije, kao i usluga i funkcionalnosti koje će korisnicima i uposlenicima biti ponuđene upotrebom web baziranog podsistema.

Web bazirani podsistem

U prvom dijelu projektnog zadatka već je naglašeno da će se cjelokupan sistem raditi upotrebom Oracleove ADF tehnologije. Desktop aplikacija biće rađena upotrebom ADF/Swing tehnologije a web prezentacija i cjelokupan podsistem za korisničku podršku upotrebom ADF Faces tehnologije. Stoga smo se odlučili koristiti Oracle JDeveloper 11 kao razvojno okruženje.

Također, u prvom dijelu projekta u poglavljima 2 i 3 naglašeno koje funkcionalnosti i koje komponente će imati web bazirani podsistem (sistem za korisničku podršku kako stoji u spomenutim poglavljima). Na ovom mjestu će se na detaljan način opisati ovaj podsistem, kako je definisano u uvodu ovog dokumenta, te će se detaljno proći kroz sam razvoj podsistema i njegovih komponenti.

Zahtjevi podsistema

Na ovom mjestu potrebno je detaljno objasniti koji su razlozi za razvijanje web baziranog sistema, koji su to zahtjevi koje on mora ispuniti, koje će funkcionalnosti realizovati i kako korisnicima garantovati siguran sistem?

Kao prvi razlog za razvijanje ovog sistema javila se potreba za postojanjem web prezentacije banke kao mogućnost predstavljanja poslovanja, odnosno prezentiranja proizvodnog programa proizvoda i usluga na način velikog dometa i mogućnosti. Kao logičan slijed, nakon web prezentacije na internetu, dolazi potreba za razvojem web aplikacija koje će imati namjenu da obezbjede podršku poslovnim partnerima i korisnicima. Na kraju, kao glavni cilj, dolazi postojanje integritetnog poslovnog informacionog sistema koji će korisniku pružati sve neophodne informacije, pružati niz aplikacija kao „sistema“ korisničke podrške, sistema koji će preuzeti glavnu riječ u pogledu poslovanja banke sa njenim klijentima i sistema koji će pružati sigurnost obavljanja poslovnih transakcija.

Usluge koje banka pruža putem web podsistema

Takav jedan podsistem korisnicima bi trebao da omogućiti:

- Svim posjetiocima web prezentacije banke dostupnost svih informacija vezanih za vrste kredita, kao i način njihove otplate.
- Online registracija korisnika banke u cilju pružanja usluga registrovanim korisnicima.
- Registrovanim korisnicima banke mogućnost online podnošenja zahtjeva za kredit.
- Za svaki odobreni kredit kreiranje jedinstvenog otplatnog plana.
- Online otplata kredita, tj. uplaćivanje rata kredita putem web aplikacija.
- Upisnicima banke potrebno je omogućiti pregled svih dospjelih zahtjeva za kredit, kao i daljih akcija nad tim zahtjevima (odbijanje/odobravanje zahtjeva)

- Pružanje usluga monitoring timu: mogućnost pregleda trenutnog stanja u banci. Stanja koje se odnosi na efikasnost poslovanja banke: pregled aktivnih kredita, uvid u otplaćivanje kredita kako bi se moglo ustanoviti da li klijenti banke ispunjavaju obaveze na koje su pristali sklapanjem ugovora sa bankom onog trenutka kada im je dodijeljen kredit.

Kako će neki od ovih zahtjeva biti realizovani i koje će se sve funkcionalnosti realizovati prilikom ispunjavanja ovih zahtjeva vidjećemo u nastavku ovog dokumenta.

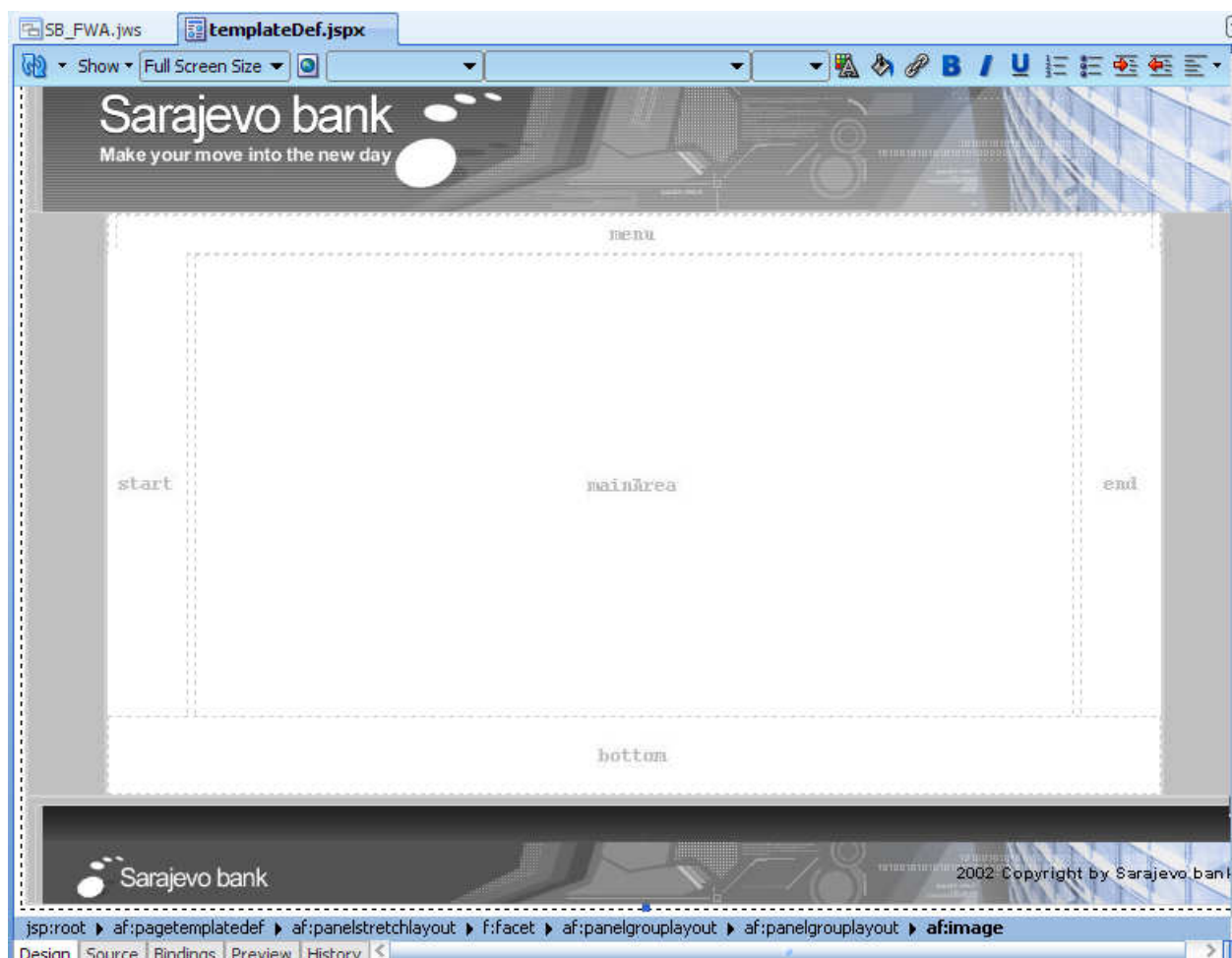
Dostupnost informacija 24/7

Dostupnost svih informacija 24 sata 7 dana sedmično podrazumijeva postojanje adekvatne web prezentacije banke na kojoj će svi posjetioči (registrovani članovi ili obični posjetioči) moći dobiti sve neophodne informacije o uslugama koje banka pruža svojim klijentima. Usluge koje banka pruža klijentima odnose se na proces kreditiranja. Tako će klijentima biti dostupne informacije o tri kreditne linije koje su predstavljene sljedećom tabelom:

Tabela 1. Kreditne linije banke

No.	Naziv	Opis	Namjena	Min iznos	Max iznos	Period otplate	Obaveznost postojanja žiranta	Rata
1.	SB Standardni							
2.	SB Silver							
3.	SB Gold							

Kako će sam web bazirani podsistem biti sačinjen od niza web stranica, potrebno je napraviti šablon (template) kojeg će sve stranice slijediti, u cilju postizanja jedinstvenog dizajnerskog rješenja sistema. U tu svrhu smo kreirali JSF Page Template kojeg će sve ostale stranice u pogledu dizajna slijediti. Template ima dvije editabilne regije (dva faceta koja će biti specifična za svaku stranicu). To su: *menu* – služi kao prostor stranice na kojem će biti smještene menu komponente, navigacija te eventualno neko dugme koje će realizovati neku funkcionalnost; i *mainArea* – dio na kojem će se učitavati i prikazivati sadržaj stranice. Izgled šablona unutar razvojnog okruženja prikazan je slikom 91:



Slika 91. Izgled šablona kojeg će sve stranice naslijediti, tj. kojeg će sve stranice u pogledu dizajna naslijediti.

Većina stranica u sistemu koristiće isti menu bar, tj. istu navigaciju koja će nas na jednostavan način usmjeravati ka ostalim stranicama sistema. HTML kod koji realizuje takav jedan meni bar priložen je u nastavku.

```
<af:menuBar>
  <af:commandMenuItem text="Novosti"/>
  <af:spacer width="10" height="10"/>
  <af:menu text="O nama">
    <af:commandMenuItem text="O nama"/>
    <af:commandMenuItem text="Izvestaj o poslovanju"/>
    <af:commandMenuItem text="Posao"/>
  </af:menu>
  <af:spacer width="10" height="10"/>
  <af:menu text="Usluge">
    <af:commandMenuItem text="SB Standard"/>
  </af:menu>
</af:menuBar>
```

```

    <af:commandMenuitem text="SB Silver"/>
    <af:commandMenuitem text="SB Gold"/>
  </af:menu>
  <af:spacer width="10" height="10"/>
  <af:commandMenuitem text="Lokacije"/>
  <af:spacer width="10" height="10"/>
  <af:commandMenuitem text="Kontakt"/>
  <af:spacer width="10" height="10"/>
  <af:menu text="Registrovani korisnici">
    <af:commandMenuitem text="Login"/>
    <af:commandMenuitem text="Postani clan"/>
  </af:menu>
</af:menuBar>

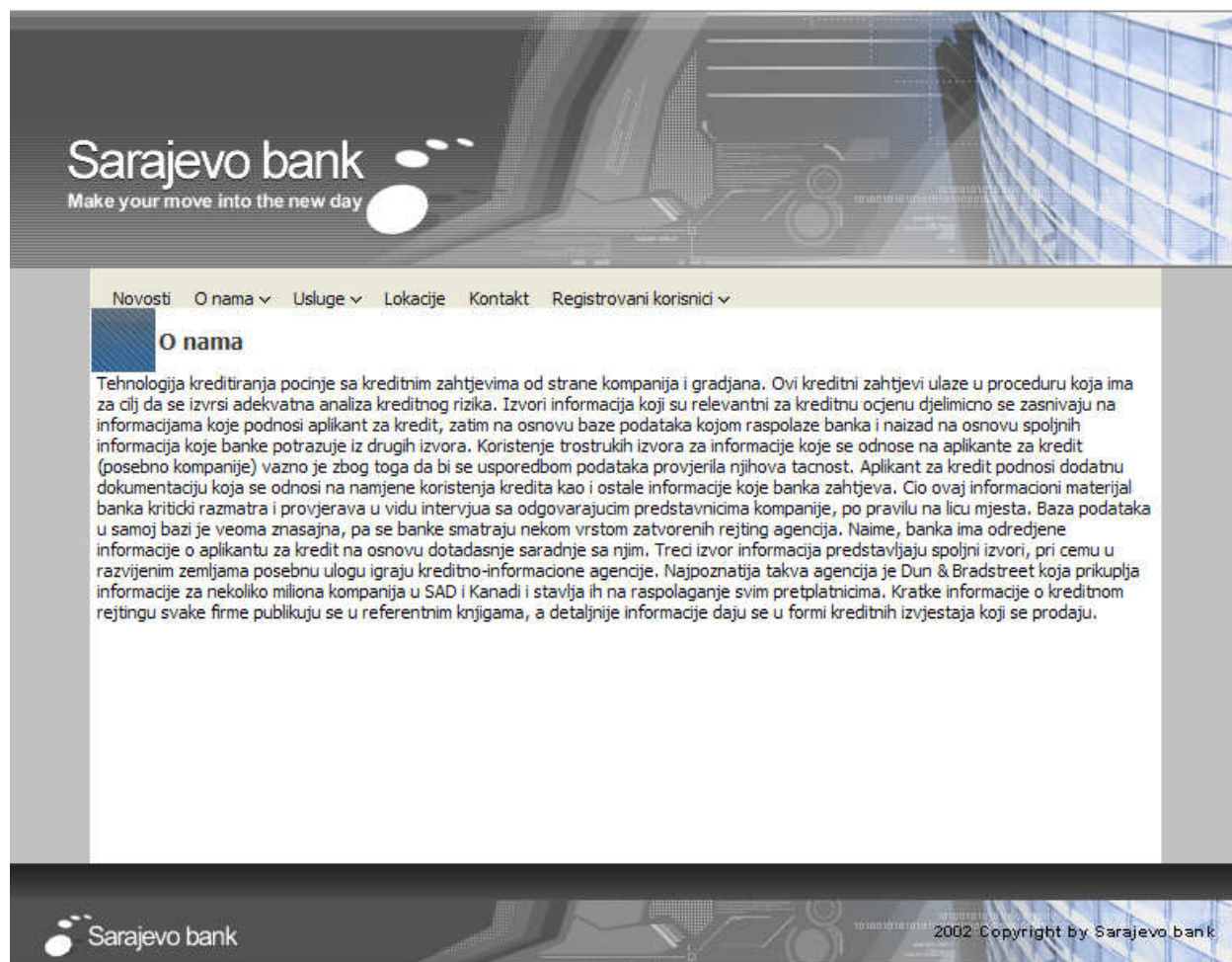
```

Stranice koje će realizovati zahtjev sistema o dostupnosti informacija su: **novosti.jspx** – stranica služi za prikaz najnovijih informacija vezano za način poslovanja banke (nove pogodnosti u liniji kreditiranja), otvaranje novih poslovnica itd; **o_nama.jspx** – stranica na kojoj će biti pružene osnovne informacije o banci (historijat banke itd); **poslovanje.jspx** – na ovoj stranici će se nalaziti izvještaj o poslovanju banke (vizija i misija banke, statistički podaci o broju klijenata, broju ostvarenih prava na kredit itd); **posao.jspx** – stranica na kojoj će biti objavljeni konkursi za posao u banci; Meni „**Usluge**“ će imati opcije za pristup informacijama o tri kreditne linije banke koje su navedene u tabeli 1. Stoga će se kreirati i tri stranice istog naziva kao i kreditne linije na kojima će posjetioc moći dobiti sve neophodne informacije vezane za tu kreditnu liniju, a registrovani članovi banke će moći i podnijeti zahtjev za dobivanje kredita. Stranica **lokacije.jspx** namijenjena je za grafički prikaz lokacija poslovnica banke. I na kraju stranica **kontakt.jspx** će pružati informacije pomoću kojih klijenti ili potencijalni klijenti mogu ostvariti kontakt sa bankom.

Prije nego što krenemo sa izradom spomenutih stranica neophodno je da za našu aplikaciju napravimo model. Model gradimo na osnovu „ADF Buisness Tier“-a koji nam omogućava kreiranje komponenti iz tabela koje postoje u našoj bazi podataka. Dakle, neophodno je da kreiramo komponente koje će uzimati podatke iz baze podataka, jer stranice poput **novosti.jspx**, te stranice koje se odnose na usluge banke sve informacije koje budu prikazivale uzimaće direktno iz baze podataka. Kreiranje modela podrazumijeva kreiranje Entity i View objekata. Entity objekti nam omogućavaju da čuvamo u kešu podatke iz baze, te da pratimo ažuriranja u bazi. Osim ovih mogućnosti, entity objekti su pogodni za rad zbog svojstva dodavanja validacije na polja objekta (odnosno na polja iz tabele, jer entity objekti mapiraju tabele 1:1). Entity objekti nam omogućavaju manipulaciju podacima (ažuriranje, brisanje, dodavanje novih slogova), dok View objekti omogućavaju samo prikaz sadržaja tabele.

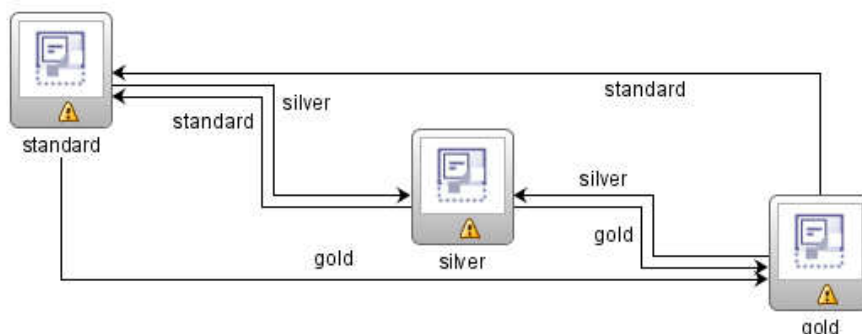
Izrada stranice „o_nama.jspx“

Prva od stranica koja će se realizovati je stranica **o_nama.jspx**. Na ovoj stranici neće biti prikazivani podaci koji se dobivaju iz baze podataka, već će samo sadržavati neke tekstualne podatke. Važno je napomenuti da se za svaku stranicu koju kreiramo, također, kreira i backing java datoteka koja nam omogućava da za naše stranice pišemo java kod ukoliko je neke stvari potrebno isprogramirati (npr. Unutar backing java datoteke možemo prosljeđivati parametre pripremljenim view objektima itd). Stranica je prikazana na slici broj 92.



Slika 92. Stranica `o_nama.jspx`

Na slici broj 92 možemo uočiti menu bar koji će služiti kao navigacija do određenih stranica koje stoje na raspolaganju posjetiocima web prezentacije banke. Da bi menu bar funkcionisao, tj. da bi odabirom neke od njegovih opcija zaista bili preusmjereni na odgovarajuću stranicu potrebno je da kreiramo *task flow*. Tok zadatka kreiramo unutar datoteke `adfc-config.xml` ili možemo kreirati nove tokove.



Slika 93. ADF Task flow: "usluge_flow" koji prikazuje navigaciju izmedju stranica: standard.jspx, silver.jspx i gold.jspx. Na linijama koje spajaju stranice napisanu su nazivi akcija koje će se pozivati prilikom odabira opcija menu bara.

Izrada stranica za kreditne linije

Sljedeća stranica koju ćemo realizovati je jedna od stranica koja nam daje informacije o kreditnoj liniji SB Standard. Dakle, kreiramo stranicu standard.jspx. Osim što će sadržavati osnovne podatke o ovoj kreditnoj liniji, stranica će također nuditi opciju posjetiocu da napravi zahtjev za kredit. Zahtjev za kredit mogu podnositi samo registrovani članovi banke, tako da nakon odabira opcije za podnošenje zahtjeva za kredit, korisnik mora biti preusmjeren na stranicu za login (ukoliko već nije prijavljen). Iz navedenog možemo zaključiti da web bazirani sistem ima stranice koje su public (dostupne svima) i one koje su zaštićene, te se za rad s njima moramo prijaviti na sistem sa korisničkim nalogom kojeg korisnik dobije kada potpisuje prvi put ugovor sa bankom.

Postavlja se pitanje kako obezbjediti ovaj nivo sigurnosti i zaštititi određeni broj stranica? U ovu svrhu možemo koristiti ADF Security i u nastavku ćemo nakon što objasnimo izradu web stranica kreditnih linija objasniti koji vid sigurnosti ćemo koristiti i kako on funkcioniše.

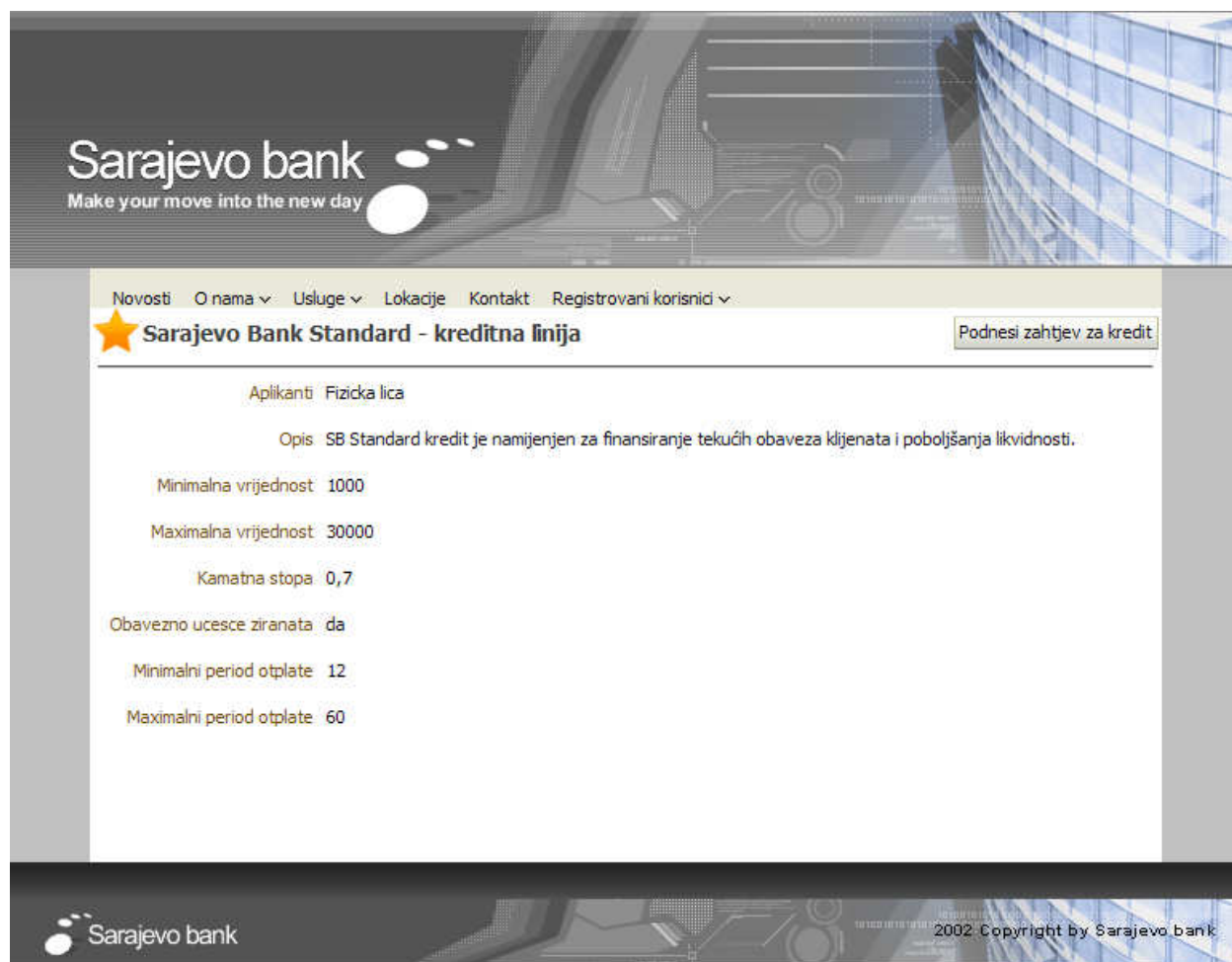
Stranica standard.jspx će prikazivati podatke koji su smješteni u tabeli *tipovi_kredita* stoga moramo napraviti View objekat u našem modelu koji će dohvatati podatke vezano za tip kredita „SB Standard“. SQL upit za pogled dat je u nastavku:

```

SELECT tip_aplikanta, opis, min_vrijednost,
       max_vrijednost, kamata, ziranti, min_period_otplate,
       max_period_otplate
FROM tipovi_kredita WHERE tip_kredita = 'SB Standard'

```

View smo nazvali SBStandard i dodali smo ga u model naše aplikacije. Također, kako je namjena ovog pogleda samo da prikazuje podatke iz tabele isti smo napravili kao read-only. Zatim smo na stranicu dodali kreirani view u obliku ADF Read Only forme. Na stranicu smo dodali i dugme koje će realizovati funkciju podnošenja zahtjeva. Ova funkcija podrazumijeva otvaranje nove stranice na kojoj će korisnik unijeti sve neophodne podatke vezano za kredit (vremenski period otplate, željeni iznos itd), naravno ukoliko je korisnik prijavljen na sistem. Ukoliko korisnik nije prijavljen na sistem, doći do automatskog preuzmjerenja korisnika na stranicu za login. Stranica standard.jspx je prikazana na slici 94.



Slika 94. Stranica *standard.jspx* – prikaz osnovnih informacija o kreditnoj liniji SB Standard. Stranica sadrži link (dugme) za podnošenje zahtjeva za ovu kreditnu liniju.

Isti je postupak za kreiranje stranice druge dvije kreditne linije, te ćemo u nastavku samo napisati sql upit koji smo koristili za pravljanje potrebnih view objekata i priložiti slike stranica.

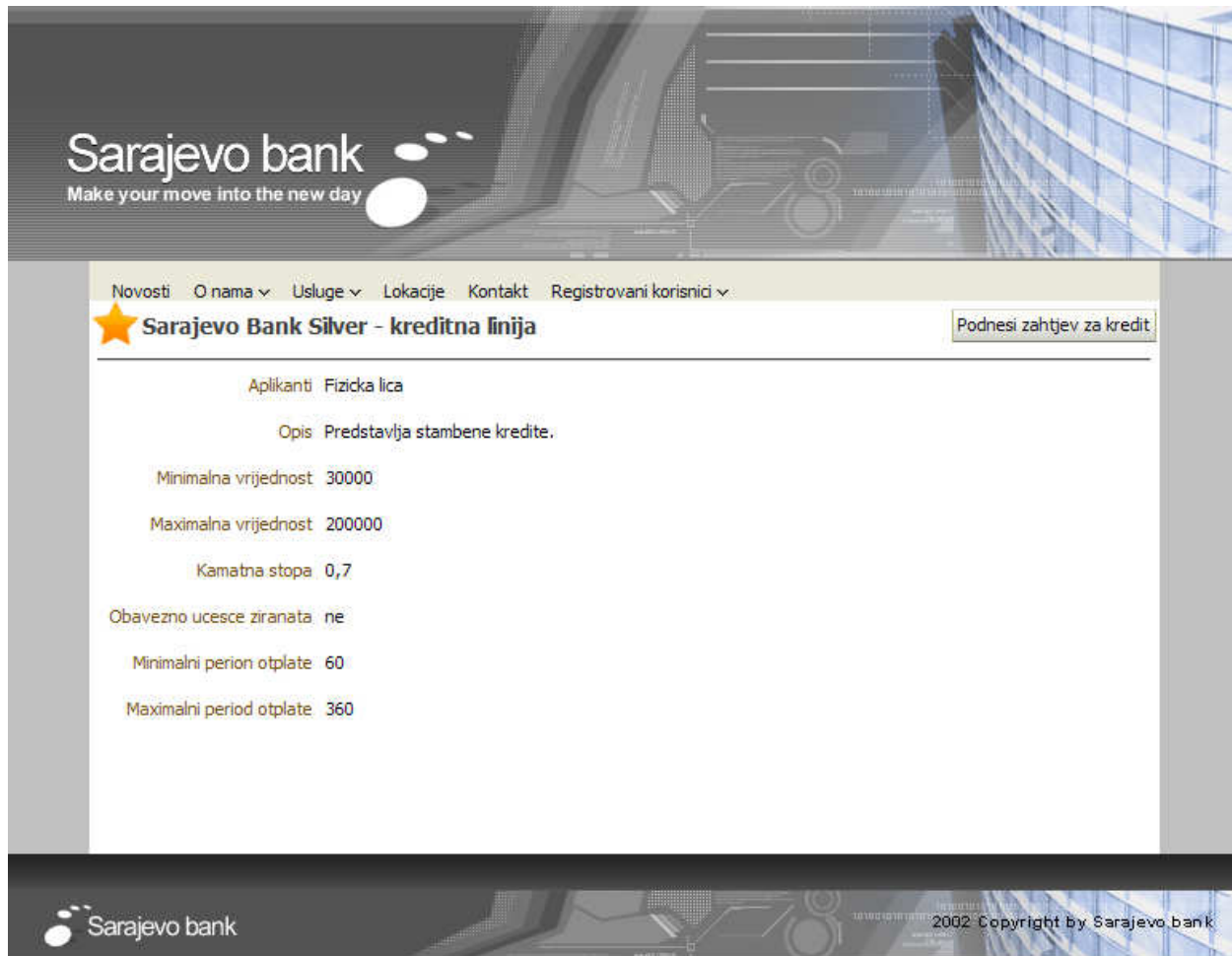
Sql upit za kreiranje pogleda SBSilver:

```
SELECT tip_aplikanta, opis, min_vrijednost,
       max_vrijednost, kamata, ziranti, min_period_otplate,
       max_period_otplate
FROM tipovi_kredita WHERE tip_kredita = 'SB Srebreći'
```

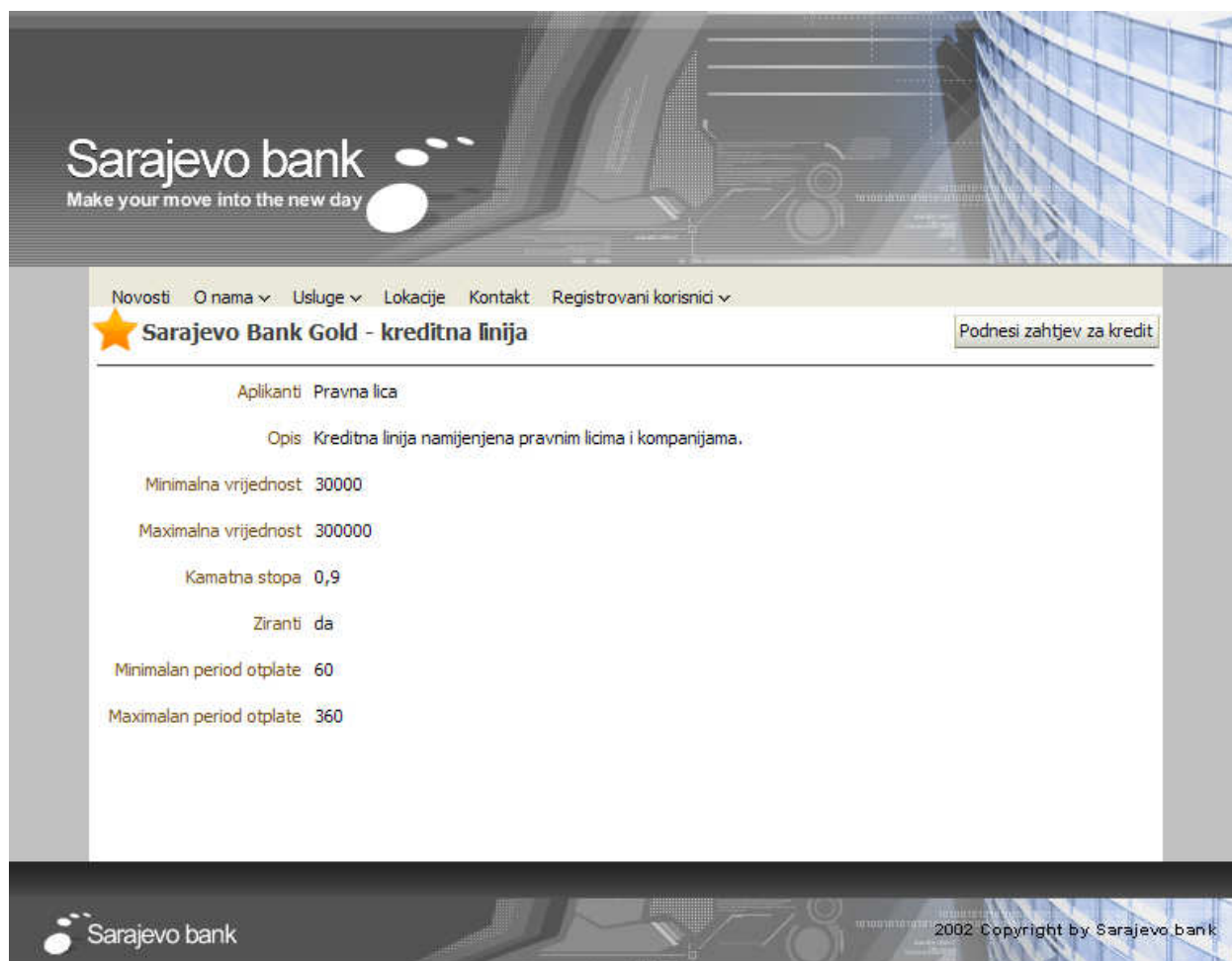
Te sql upit za kreiranje pogleda SBGold:

```
SELECT tip_aplikanta, opis, min_vrijednost,
```

```
max_vrijednost, kamata, ziranti, min_period_otplate,  
max_period_otplate  
FROM tipovi_kredita WHERE tip_kredita = 'SB Zlatni'
```



Slika 95. Stranica silver.jspx – osnovni podaci o kreditnoj liniji SB Srebrei i mogućnost apliciranja za istu.



Slika 96. Stranica *gold.jspx* – prikaz informacija o kreditnoj liniji SB Zlatni kao i mogućnost podnošenja zahtjeva za istu.

Kako podnošenje zahtjeva podrazumijeva login korisnika na sistem, o čemu je bilo riječi i ranije, to ćemo u nastavku dokumenta objasniti šta je to ADF Security i šta nam sve omogućava.

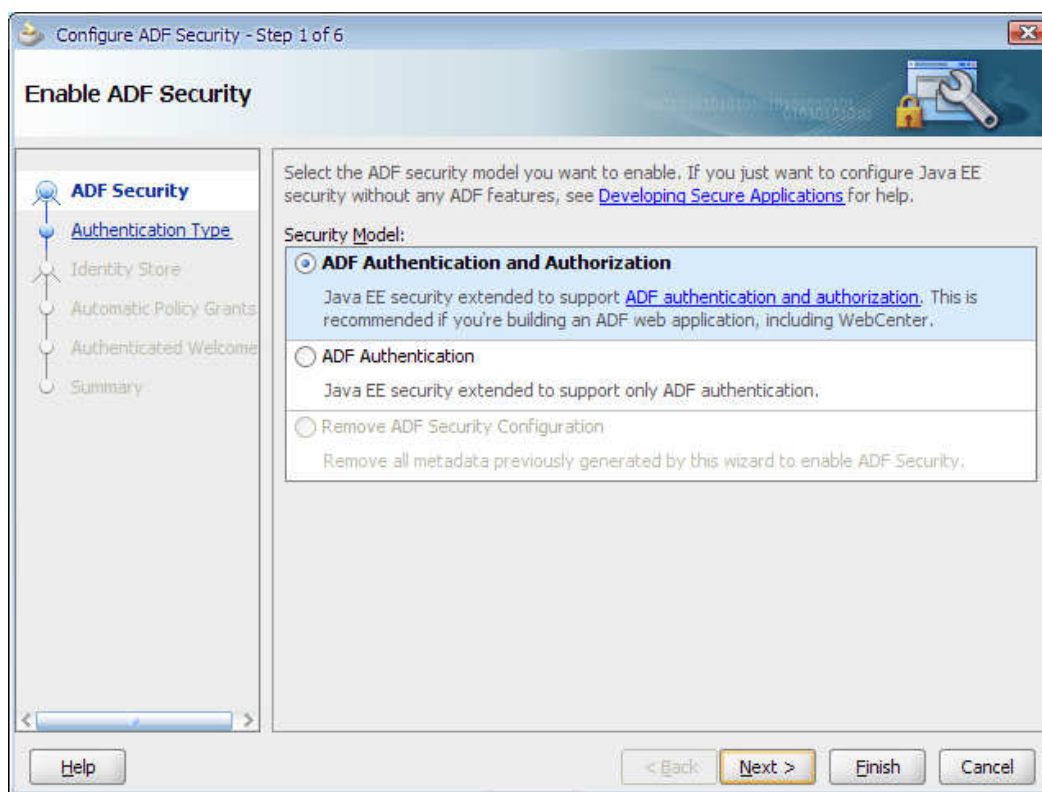
ADF Security

Oracle ADF Security nam dopušta da kreiramo police pristupa za različite resurse aplikacije. Tako na primjer, možemo upotrebom različitih polica definisati koja polica može pristupiti kojem toku poslova (task flow). Police su sačuvane u sigurnosnoj datoteci koja se zove `jazn-data.xml`.

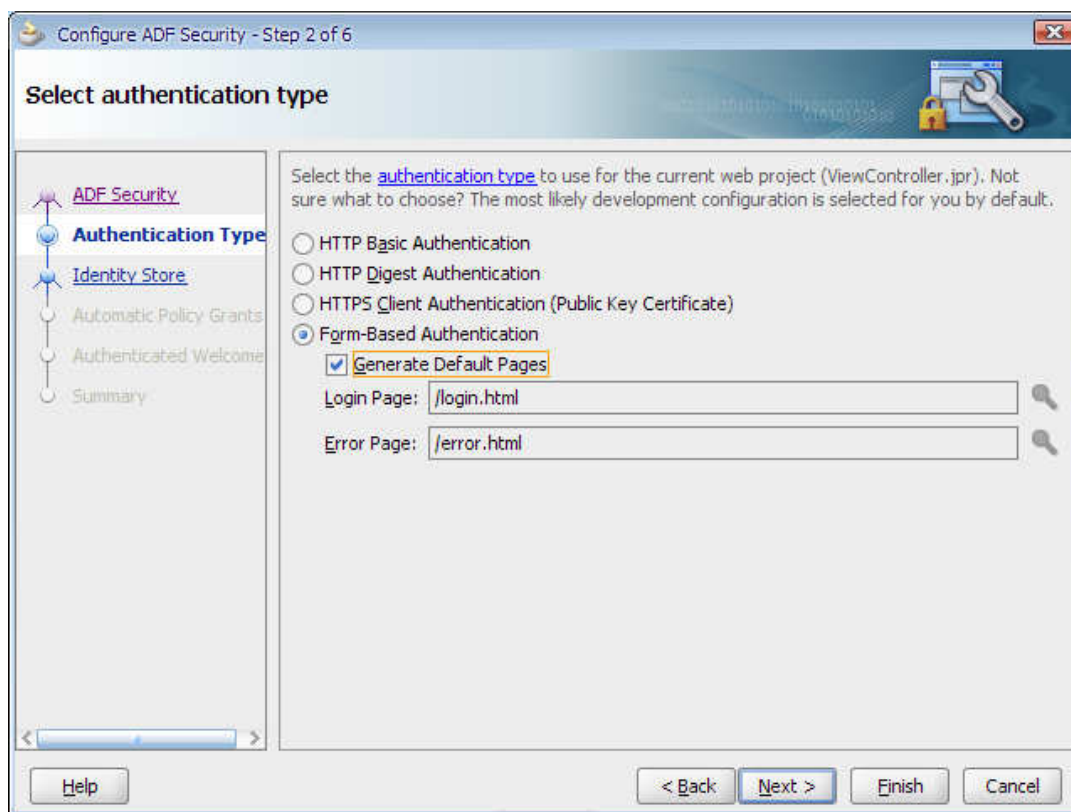
JDeveloper i ADF Security nam dopuštaju da omogućimo autentikaciju i autorizaciju zasebno, tako da imamo dvije opcije:

- (1) Omogućiti ADF autentikacijski sevrlet sa nametanjem ADF autorizacije. Nametanje autorizacije podrazumijeva definisanje prava pristupa web aplikaciji dodjeljujući pristupne dozvole određenim rolama aplikacije.
- (2) Omogućiti ADF autentikaciju koja nalaže kreiranje sigurnosnih ograničenja za stranice. I jednom kada se korisnik loguje sve stranice sistema su mu dostupne.

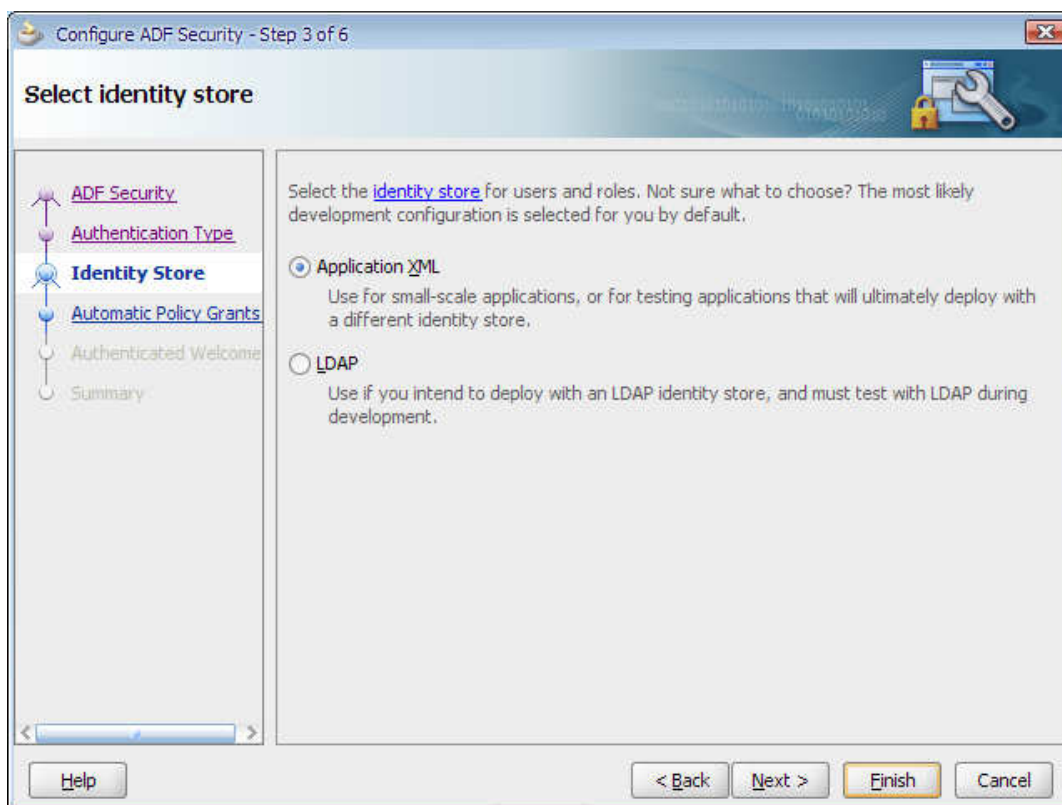
Kako je web bazirani podsistem namijenjen i klijentima, ali i uposlenicima banke onda ćemo koristiti autentikaciju sa enforsiranom autorizacijom kako bi mogli definisati prava pristupa nad određenim skupovima resursa (stranica). Koraci za omogućavanje sigurnosti prikazani su narednim slikama.



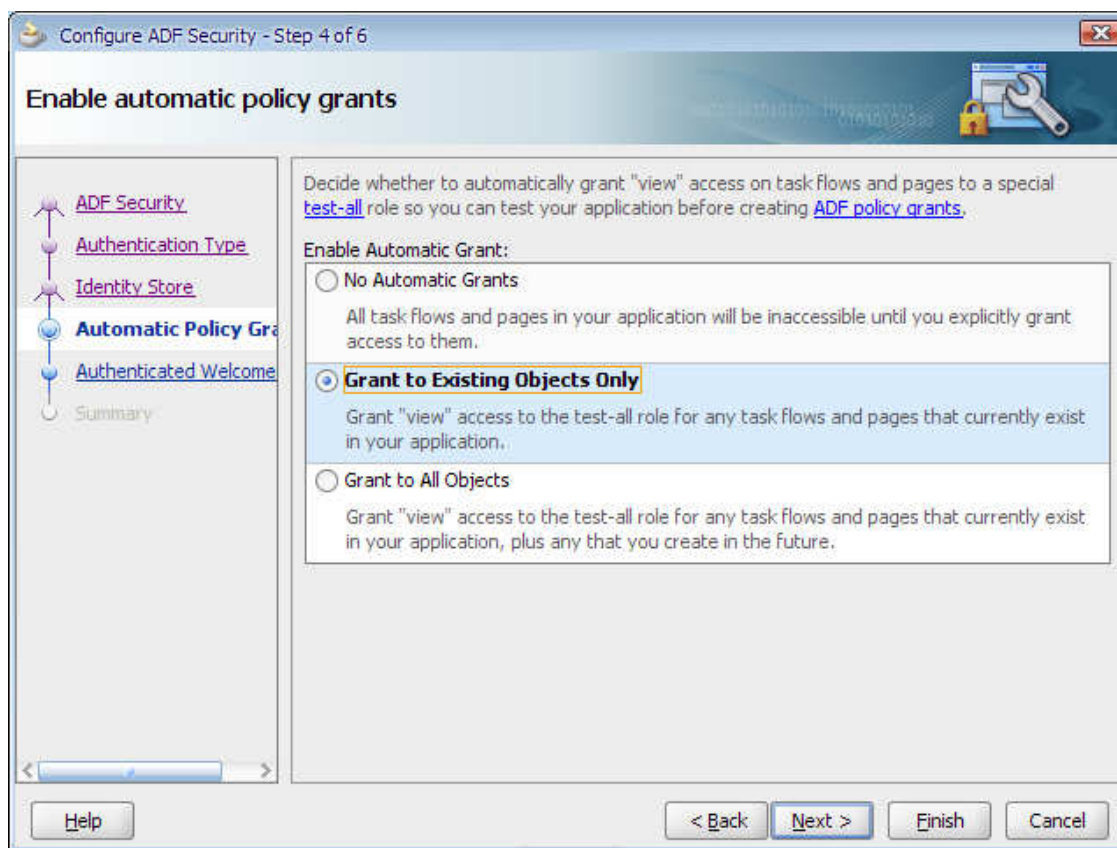
Slika 97. Korak (1): odabir vrste sigurnosti. Mi se odlučujemo za autentikaciju sa enforsiranom autorizacijom kako različitim korisnicima omogućili pristup različitim skupovima resursa.



Slika 98. Korak (2): Odabir tipa autentikacije. Dva najčešća tipa su HTTP Basic i Form-Based. Prvi tip predstavlja način da se korisnik loguje na sistem kroz dijalog okvir kojeg mu browser ponudi, dok drugi tip podrazumijeva login kroz formu koja se nalazi na definisanoj stranici (u ovom slučaju mi smo koristili defaultne stranice).

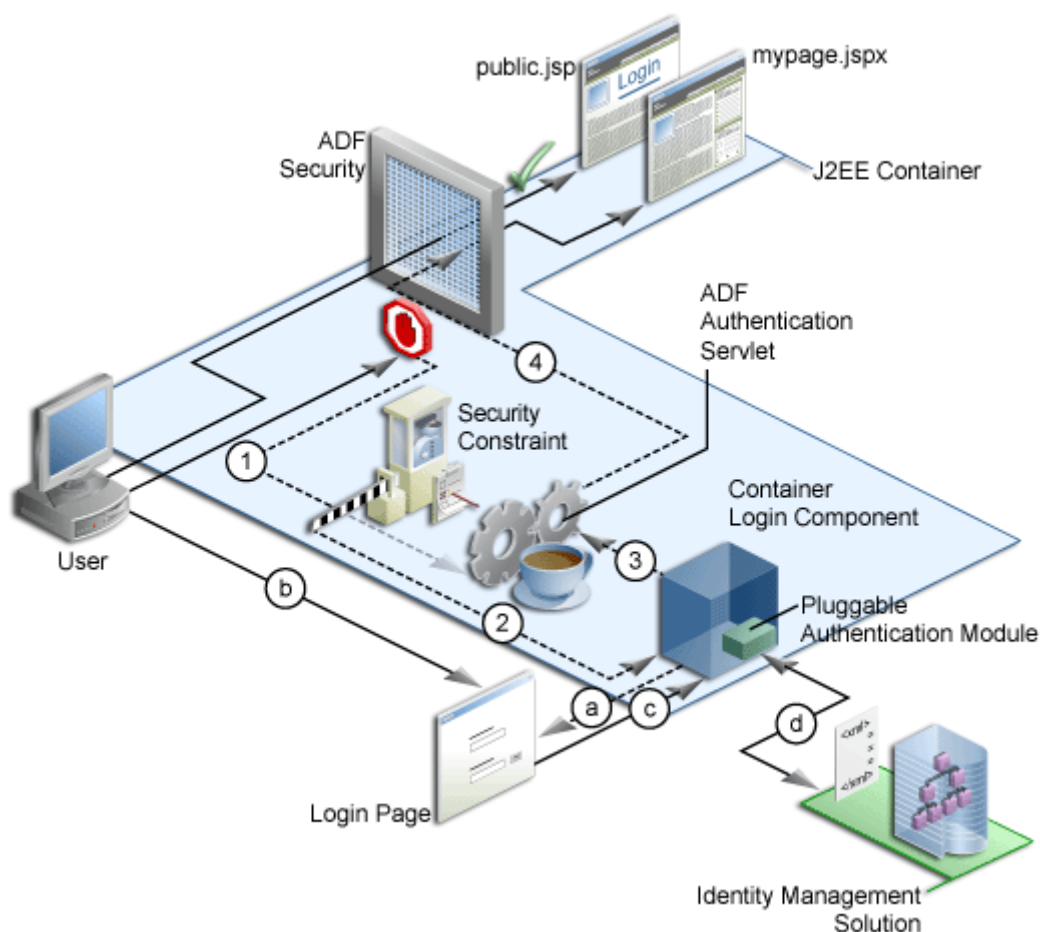


Slika 99. Korak (3): odabir načina pohranjivanja sigurnosnih podataka (misli se na korisničke naloge). U našem slučaju odlučeno je da to bude XML datoteka u kojoj će biti pohranjeni korisnički akaunti i lozinke, kao i dozvole za pristup određenim stranicama.



Slika 100. Korak (4): odabir automatske police dozvola: omogućićemo pristup svim do sada kreiranim stranicama ali ne i onim koje ćemo kreirati u budućnosti. Ovo je zadnji korak i pritiskom na dugme „Finish“ kreirat će se sve neophodne datoteke koje će nam obezbjediti mehanizam sigurnosti u našem web baziranom podsistemu.

Pogledajmo sada kako radi ADF Security. Na slici 101 prikazan je proces autentikacije kada korisnik pokušava pristupiti stranici koja sadrži ADF resurse bez prethodne prijave na sistem. Implicitno se inicira autentikacija jer je korisnik pokušao pristupiti stranici koja nije javna, tj. za nju postoji određena dozvola pristupa.



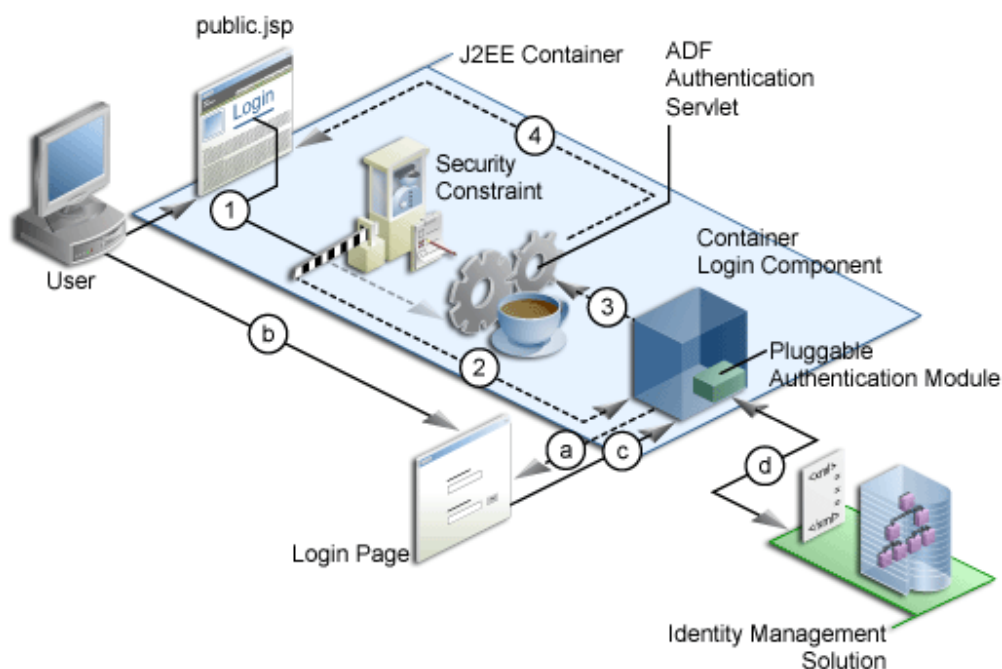
Slika 101. *Implicitno inicirana autentikacija [1].*

Implicitna autentikacija podrazumijeva da resurs nema dozvolu za *anonymous-role* rolu, da korisnik nije prethodno autentikovao, te da je metoda autentikacije form-bazirana. U tom slučaju proces se odvija sljedećim tokom:

- (1) Kada se zahtjeva web stranica (ili ADF Task flow), filter ADF bindings servlet-a preusmjerava zahtjev Oracle autentikacijskom servletu pohađujući logičku operaciju koja je pokrenula login.
- (2) ADF autentikacijski servlet ima postavljeno Java EE sigurnosno ograničenje koje prouzrokuje pokretanje mehanizma autentikacije.
 - (a) Prikaz forme za login.
 - (b) Korisnik unosi neophodne podatke u login formu.
 - (c) Klikom na dugme forme korisnik šalje formu `j_security_check()` metodi.
 - (d) Java EE container autentikuje korisnika koristeći autentikacijski modul.
- (3) Nakon uspješne autentikacije, container preusmjerava korisnika na servlet koji je prouzrokovao autentikaciju – u ovom slučaju ADF autentikacijski servlet.

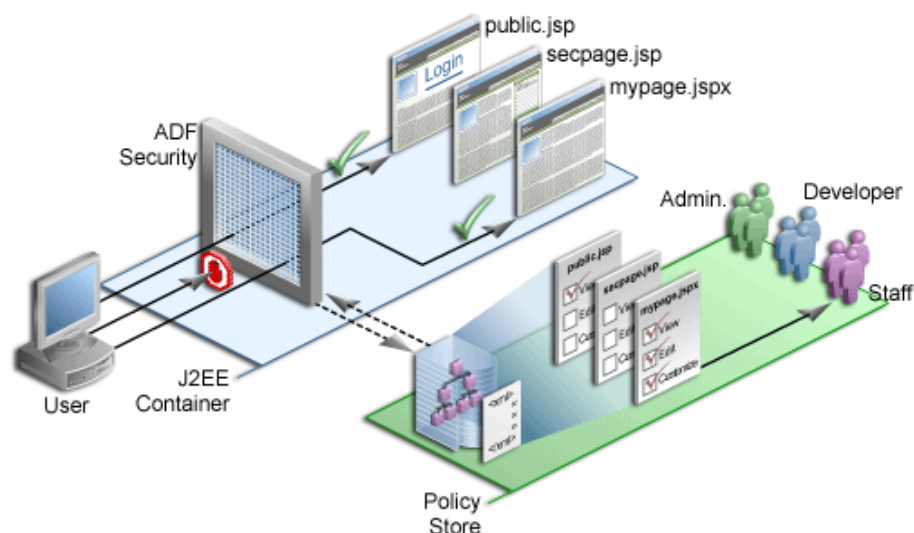
- (4) ADF autentikacijski servlet preusmjerava korisnika na željeni resurs (na slici na stranicu mypage.jsp).

Slika 102 pokazuje slučaj kada korisnik biva autentikovao koristeći login link na javnoj stranici.



Slika 102. *Eksplcitna autentikacija [1]: u ovom scenariju korisnik nakon uspješne autentikacija biva preusmjeren na istu stranicu na kojoj je i potražio opciju za login, ali sada sa dozvolama koje su pripisane njegovom akauntu.*

U slučaju kada koristimo i autorizaciju nakon autentikacije provjerava se da li korisnik dozvolu za pristup resursu koji pripada nekom skupu resursa. Ukoliko nema dozvolu sistem će korisnika preusmjeriti na error stranicu.



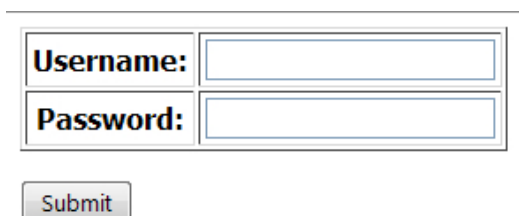
Slika 103. Autorizacija [1]: Pokušaj pristupa stranici koja se nalazi u skupu resursa koji su definisani određenom rolom u aplikaciji/sistemu.

Sada kada smo objasnili kako se odvijaju procesi autentikacije i autorizacije neophodno je definisati korisnike i role u web aplikaciji. U narednoj tabeli prikazane su role kao i opis njihovih permisija:

Tabela 2. Role u aplikaciji.

Rola	Opis
admin-users	Korisnici tipa admin moći će pristupati stranicama vezano za dodavanje i upravljanje novostima.
ks-users	Ova rola namjenjena je kreditnim službenicima banke, kojima će se omogućiti pristup stranicama i resursima koji će im omogućavati da: pregledaju prispjele zahtjeve za kredit, odobravanje kredita, pregled aktivnih kredita koje su oni odobrili, te pregled svih aktivnih kredita (ove zadnje dvije opcije odnose se na proces monitoringa).
kk-users	Namjenjena korisnicima banke kojima će biti omogućeno podnošenje zahtjeva za kredit, pregled dosadašnje saradnje sa bankom, pregled aktivnih otplata, pregled otplatnog plana itd.

Role i korisnici se dodaju u jazn-data.xml datoteci i ovde taj korak nije predstavljen². Za testiranje ove funkcionalnosti stavili smo da samo admin korisnici mogu pristupiti stanici gold.jspx. pokušaj da se otvori ova stranica bez prethodne autentikacije rezultovao je preusmjeravanjem korisnika na login stranicu.



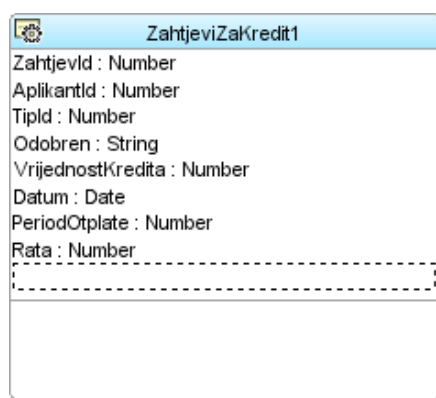
The image shows a simple login form. It consists of two input fields: one labeled 'Username:' and another labeled 'Password:'. Below these fields is a button labeled 'Submit'.

Slika 104. Forma za korisničku prijavu na sistem: redirekcija korisnika na login stranicu nakon pokušaja otvaranja stranice bez prethodne autentikacije.

Podnošenje zahtjeva za kredit

Nakon što smo dodali ADF Security možemo preći na dio koji će realizovati funkcionalnost podnošenja zahtjeva za kredit. Dakle, kao što je već prethodno rečeno, samo registrovani članovi mogu podnositi zahtjev za kredit (tj. oni koji pripadaju roli *kk-users*). Podnošenje zahtjeva za kredit namjeravamo ostvariti na dva načina. Prvi način je pritiskom na dugme za podnošenje zahtjeva na sztanicama kreditnih linija, a drugi način će biti podnošenje zahtjeva kroz opcije koje će korisniku biti ponuđene nakon prijave na sistem kroz opciju **Login** iz menu bara bilo od koje prethodno opisanih stranica.

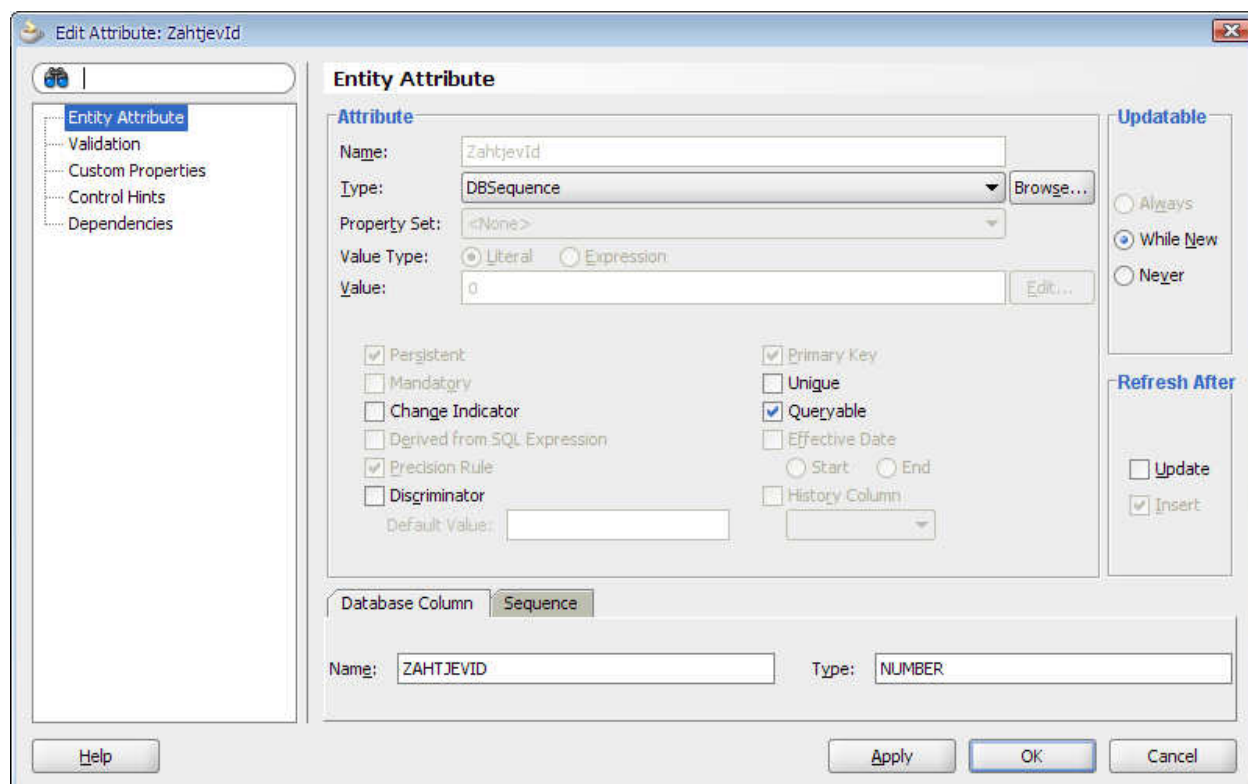
Za sada fokusiraćemo se na prvi način. Pogledajmo najprije koji to podaci sve trebaju biti unijeti da bi se kreirao jedan zahtjev za kredit.



The image shows a table titled 'ZahtjeviZaKredit1'. The table has the following fields: 'Zahtjevid : Number', 'ApikantId : Number', 'TipId : Number', 'Odobren : String', 'VrijednostKredita : Number', 'Datum : Date', 'PeriodOplate : Number', and 'Rata : Number'. Below the table is a dashed line and a large empty rectangular area.

Slika 105. Tabela „ZAHTJEVI_ZA_KREDIT“.

² Za više informacija o rolama i dodavanju korisnika pogledati poglavlje 28. **Adding Security to a Fusion Web Application** [1].



Slika 106. Postavljanje vrijednosti primarnog ključa na sekvencu iz baze podataka.

U trenutku kada se kreira nova instanca entity objekta aplikacija će u polje `ZAHTJEV_ID` upisati neku vrijednost (u ovom slučaju to je vrijednost -2) kako bi se na formi popunilo polje koje je obavezno za unos. Kada kliknemo na *commit* dugme u tabelu će se ustvari upisati vrijednost sekvence.

Podnošenje zahtjeva za kredit

Id zahtjeva -2

Podnosioc zahtjeva za kredit

Tip kredita 3

* Vrijednost

* Period otplate

* Iznos mjesečne rate

Slika 107. Prikaz forme za podnošenje zahtjeva za kredit. Uočimo vrijednost -2 kod polja „Id zahtjeva“.

Treće polje tabele `ZAHTJEVI_ZA_KREDIT` možemo dobiti na osnovu korisničkog naloga korisnika koji se prethodno morao autentikovati da bi podnio zahtjev za kredit. Informaciju o korisničkom imenu možemo dobiti na osnovu session scope varijable `userName`. Potrebno je samo pozvati metodu `getUserName()` `SessionScope` instance. Ali ovako ćemo dobiti samo korisničko ime, nama treba ustvari

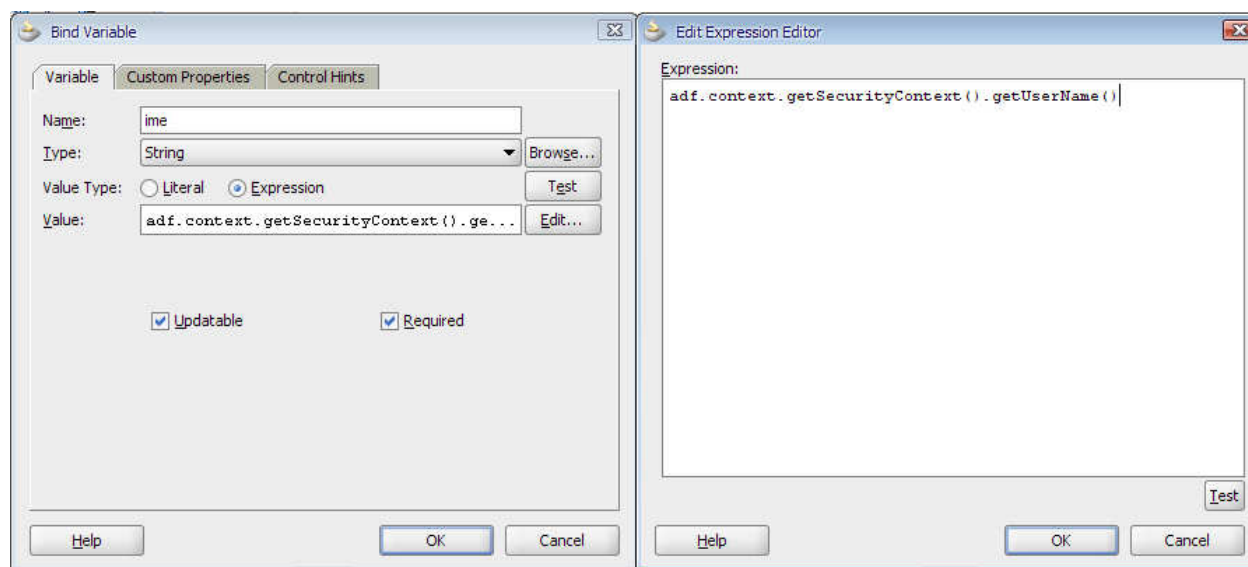
polje *aplikant_id* iz tabele *KORISNICKI_RACUNI* tako da ćemo ovu vrijednost morati dohvatiti iz baze i proslijediti je formi za kreiranje zahtjeva. U tu svrhu kreirali smo novi view objekat kojeg smo nazvali *DohvatildAplikanta*. Sql upit ovog view objekta je sljedeći:

```
SELECT aplikant_id
FROM KORISNICKI_RACUNI
WHERE korisnicko_ime = :ime
```

gdje je **ime** takozvana bind varijabla i ovaj view objekat očekuje od aplikacije da mu proslijedi vrijednost bind varijable kako bi on mogao vratiti rezultat select naredbe. U JDeveloperu možemo odrediti i vrijednost bind varijabli u trenutku izvršavanja aplikacije upotrebom **Groovy** skriptnog jezika. I mi u našem slučaju varijablu **ime** moramo postaviti na vrijednost koju je korisnik unio kao *userName* prilikom autentikacije. Podataka o korisničkom imenu možemo dobiti iz instance objekta *SecurityContext*-a. Groovy izraz za dohvaćanje korisničkog imena je:

```
adf.context.getSecurityContext().getUserName()
```

Postavljanje vrijednosti bind varijable prikazano je i na narednoj slici.



Slika 108. Postavljanje vrijednosti bind varijable view objekta upotrebom Groovy skriptnog jezika.

Sada kada možemo dobiti vrijednost *aplikant_id* potrebno je istu dodijeliti polju forme za kreiranje zahtjeva za kredit. To možemo uraditi na sljedeći način:

- (1) Dodat ćemo na stranicu view *DohvatildAplikanta* ali ćemo postaviti da se on ne prikazuje na stranici, jer nam je potreban samo za dohvaćanje vrijednosti.

- (2) U backing bean datoteci stranice kreirat ćemo novu varijablu `int idAplikanta` za koju ćemo generisati getter i setter metodu na sljedeći način:

```
public void setIdAplikanta(Integer idAplikanta) {
    this.idAplikanta = idAplikanta;
}

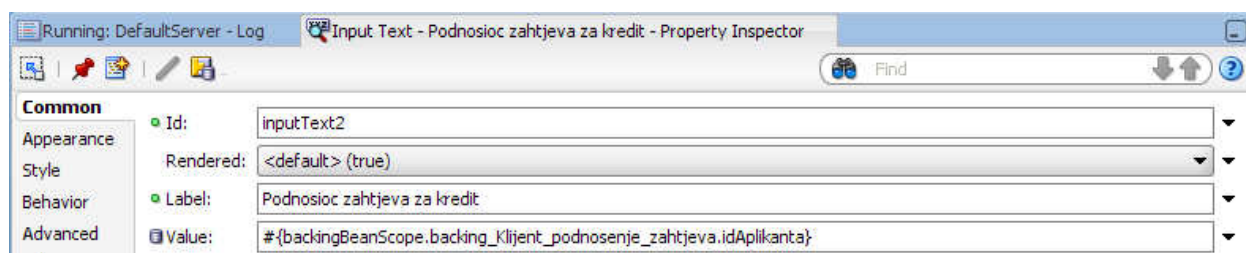
public Integer getIdAplikanta() {
    return Integer.parseInt(outputText1.getValue().toString());
}
```

gdje je `outputText1.getValue()` vrijednost koju vraća view *DohvatIdAplikanta*.

- (3) Za polje koje predstavlja Id Aplikanta u formi za podnošenje zahtjeva value ćemo postaviti na:

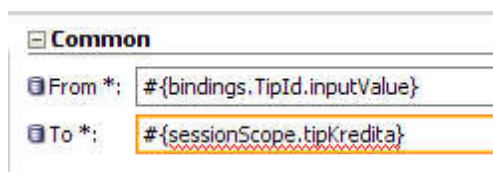
```
#{backingBeanScope.backing_Klijent_podnosenje_zahhtjeva.idAplikanta}
```

što je prikazano narednom slikom



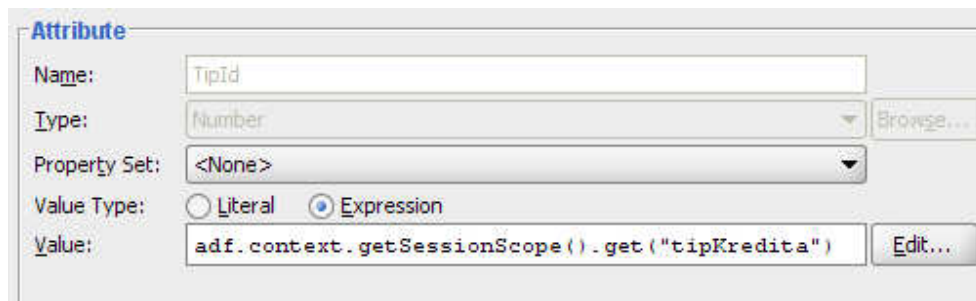
Slika 109. Postavljanje vrijednosti „podnosioc zahtjeva za kredit“.

Sljedeće polje tabele `ZAHTJEVI_ZA_KREDIT` je polje *odobren* koje ima već podrazumijevanu vrijednost *ne*. Također, polje *datum* je postavljeno na *today*. Sljedeća tri polja koja su važna su *vrijednost_kredita*, *period_otplate* i *rata*. Ono što je zajedničko za ova tri polja je da se vrijednost polja *rata* određuje na osnovu polja *vrijednost_kredita*, *period_otplate*, ali i vrijednosti kamatne stope koju imamo u tabeli `TIP_KREDITA`. Da bismo na stranici za podnošenje zahtjeva za kredit imali vrijednost kamatne stope za onaj kredit na čijoj smo stranici i inicirali podnošenje zahtjeva za kredit potrebno je da uradimo sljedeće: na stranicama koje se odnose na kreditne linije (4, 5, 6) na dugme koje se zove „Podnesi zahtjev za kredit“ neophodno je dodati odgovarajuće Action Listener-e kako bi određene vrijednosti sa forme smjestili u Session Scope varijable. Primjer ovog postupka prikazan je na slici 110.



Slika 110. Property polja action listenera dodanog na button – odredili smo da se vrijednost iz polja TipId forme proslijedi drugoj stranici kroz session scope varijablu tipKredita.

Tako smo odlučili proslijediti sljedeće varijable: tipKredita(koristimo kako bi znali za koji to tip kredita korisnik aplicira), kamatnaStopa (koristimo da izračunamo vrijednost rate). U nastavku slijedi primjer postavljanja polja TIP_ID na vrijednost session scope varijable tipKredita. Potrebno je editovati ovo polje view objekta *PodnosenjeZahtjeva* tako da se njegova vrijednost odredi pomoću Groovy skriptnog jezika kako je to prikazano na sljedećoj slici:



Slika 111. Postavljanje vrijednosti atributa view objekta na vrijednost session scope varijable.

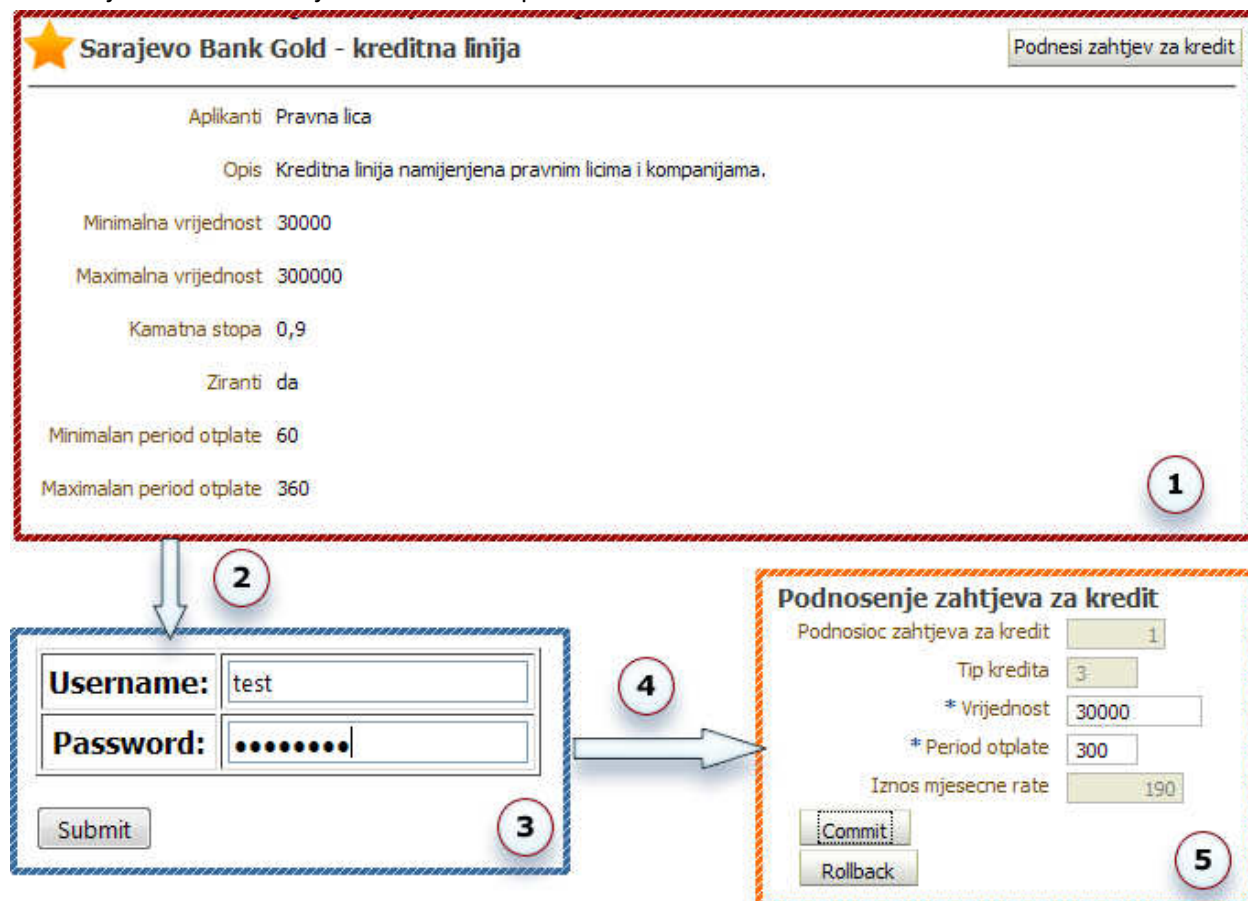
Već je naglašeno kako će se određivati polje *rata*. I to polje na formi neće se moći unositi već će se ono izračunavati pomoću sljedeće metode:

```
public Double getVrijednostRate() {
    if (inputText4.getValue() == null || inputText5.getValue() == null) {
        return 0.;
    }
    Double suma = Double.parseDouble(inputText4.getValue().toString())*(1 +
    Double.parseDouble(outputLabel1.getValue().toString()));
    return suma/Double.parseDouble(inputText5.getValue().toString());
}
```

Važno je napomenuti da smo ovde iskoristili osobinu ADF komponenti da se ponašaju kao Ajax komponente. Naime za komponente koje će sadržavati informacije o vrijednosti kredita, te periodu otplate postavili smo osobinu AutoSubmit na **true** čime smo rekli da će se postaviti vrijednosti za ove dvije komponente bez da se uradi commit transakcije. Također, morali smo za komponentu koja sadrži

vrijednost rade dodati *partialTrigger* i dodati prethodne dvije komponente kako bi naglasili da promjena vrijednosti bilo koje te dvije komponente utiče da se mijenja i komponenta koja sadrži ratu.

Na kraju kao rezultat cjelokupnog rada dobili smo implementaciju funkcionalnosti podnošenja zahtjeva za kredit čiji se tok izvršavanja slikovito može predstaviti slikom 112.



Slika 112. (1) otvaramo stranicu *gold.jspx*; (2) klikom na dugme „Podnesi zahtjev za kredit“ dešava se implicitna autentikacija i automatski korisnik biva preusmjeren na *login.html* jer stranica za podnošenje zahtjeva sadrži resurse kojima može pristupiti samo korisnik sa *kk-users* rolom; (3) unošenje korisničkog imena i lozinke; (4) u slučaju validne autentikacije korisniku se otvara stranica *podnosenje_zahhtjeva.jspx*; (5) forma za podnošenje zahtjeva za kredit – polja o podnosiocu zahtjeva za kredit, tipu odabranog kredita i visini mjesečne rate su Read Only jer se ta polja generišu na osnovu *session scope* varijabli, *securityContext* varijabli ili upotrebom neke metode.

```
# { backingBeanScope.backing_Klijent_podnosenje_zahhtjeva.vrijednostRate }
```

```
# { backingBeanScope.backing_Klijent_podnosenje_zahhtjeva.idAplikanta }
```