



CS230 - DISTRIBUIRANI SISTEMI

Umetanje konteksta i zavisnosti

Lekcija 09

PRIRUČNIK ZA STUDENTE

# CS230 - DISTRIBUIRANI SISTEMI

## Lekcija 09

### *UMETANJE KONTEKSTA I ZAVISNOSTI*

- ✓ Umetanje konteksta i zavisnosti
- ✓ Poglavlje 1: Uvod u CDI
- ✓ Poglavlje 2: Kvalifikatori
- ✓ Poglavlje 3: Stereotipovi
- ✓ Poglavlje 4: Tipovi povezivanja presretača
- ✓ Poglavlje 5: Vlastite CDI oblasti
- ✓ Poglavlje 6: Pokazni primer - Umetanje konteksta i zavisnosti
- ✓ Poglavlje 7: Individualne vežbe 9
- ✓ Poglavlje 8: Domaći zadatak 9
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

# ▼ Uvod

## UVOD

*CDI može biti upotrebljeno za pojednostavljivanje integracije različitih nivoa Java EE aplikacije.*

Umetanje konteksta i zavisnosti (**Contexts and Dependency Injection** - CDI) može biti upotrebljeno za pojednostavljivanje integracije različitih nivoa Java EE aplikacije. Na primer, CDI dozvoljava da se *zrno sesije* koristi kao *upravljano zrno*. Na taj način se može iskoristiti puna prednost primene EJB alata, poput transakcija, direktno u posmatranom upravljanoj zrnu.

Lekcija će se baviti analizom i diskusijom najsavremenijih tema iz ove oblasti, među kojima se ističu sledeće:

- **Uvod u CDI** - akcenat će biti na CDI API, povezivanju JSF komponenata i CDI zrna, oblastima zrna, zrnima kontrolera i tako dalje;
- **Kvalifikatori** - akcenat će biti na predstavljanju kvalifikatora, njihovom kreiranju i primeni;
- **Stereotipovi** - akcenat će biti na predstavljanju i kreiranju stereotipova, implementaciji stereotipova na CDI zrna;
- **Tipovi povezivanja presretača** - akcenat će biti na predstavljanju i kreiranju tipova povezivanja presretača, njihovom kodiranju, dodatnim podešavanjima kroz beans.xml datoteku, kreiranju odgovarajućih klasa kontrolera i primeni;
- **Vlastite CDI oblasti** - akcenat će biti na predstavljanju i kreiranju vlastitih CDI oblasti kao datoteka koje pišu i koriste framework programeri, više nego aplikativni programeri.

Oslanjajući se na navedene teme, sadržaj lekcije će biti pažljivo biran sa jakim akcentom na konkretne praktične primere koji će biti potpora analizi i diskusiji vezanim za ovu lekciju. Takođe, pažljivo će biti biran i sadržaj vežbi, pri čemu će izabrani zadaci u potpunosti zaokružiti izlaganje o CDI konceptima i tehnikama.

Savladavanjem ove lekcije student će u potpunosti razumeti i biti u stanju da koristi CDI koncepte i tehnike u Java EE aplikacijama. Nakon ove lekcije student će steći nova znanja i veštine u radu sa naprednim Java konceptima i principima, kao i naprednim Java EE veb aplikacijama.

# ▼ Poglavlje 1

## Uvod u CDI

### CDI API

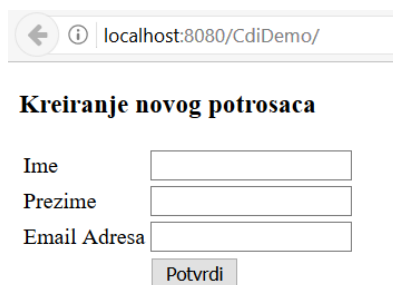
*JSF veb aplikacije koje koriste CDI veoma su slične JSF aplikacijama koje nisu koristile CDI.*

*JavaServer Faces* (JSF) veb aplikacije koje koriste CDI veoma su slične JSF aplikacijama koje u svojoj realizaciji nisu koristile *umetanje konteksta i zavisnosti*. Glavna razlika između ova dva pristupa ogleda se u primeni *CDI obeleženih zrna* umesto *JSF upravljanih zrna za model i kontrolere*. Ono što čini CDI aplikaciju lakšom za razvoj i održavanje su odlične mogućnosti primene umetanja zavisnosti primenom CDI API.

Kao i ostale JSF aplikacije, *CDI aplikacije* koriste fejslete (**facelets**) kao vlastitu tehnologiju pogleda (**views**).

Da bi problematika bila lakša za razumevanje i savladavanje, po već uspostavljenoj praksi, biće uveden konkretan primer koji će biti osnova za analizu i dalju diskusiju u lekciji.

Sledećom slikom i listingom je prikazana klasična JSF stranica koja koristi CDI:



localhost:8080/CdiDemo/

### Kreiranje novog potrosaca

Ime

Prezime

Email Adresa

Potvrdi

Slika 1.1 JSF stranica koja odgovara sledećem listingu [izvor: autor]

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Kreiranje novog potrosaca</title>
  </h:head>
  <h:body>
    <h:form>
      <h3>Kreiranje novog potrosaca</h3>
      <h:panelGrid columns="3">
```

```

        <h:outputLabel for="ime" value="Ime"/>
        <h:inputText id="firstName" value="#{potrosac.ime}"/>
        <h:message for="ime"/>
        <h:outputLabel for="prezime" value="Prezime"/>
        <h:inputText id="lastName" value="#{potrosac.
                                prezime}"/>

        <h:message for="prezime"/>
        <h:outputLabel for="email" value="Email Address"/>
        <h:inputText id="email" value="#{potrosac.email}"/>
        <h:message for="email"/>
        <h:panelGroup/>
        <h:commandButton value="Potvrđi"
                        action="#{potrosacController.
                                navigateToConfirmation}"/>

    </h:panelGrid>
</h:form>
</h:body>
</html>

```

## POVEZIVANJE JSF KOMPONENATA I CDI ZRNA

*JSF koristi Unified Expression Language za povezivanje sa osobinama i metodama CDI zrna.*

Ono što je moguće prvo primetiti, u kodu koji je priložen u prethodnoj sekciji, jeste da *JSF komponente* koriste unificirani jezik izraza - *Unified Expression Language* da se povežu sa osobinama i metodama CDI zrna.

U daljoj analizi, predstavlja se CDI zrno *Potrosac*, koje je priloženo sledećim listingom.

```

package com.metropolitan.cdideo;

import java.io.Serializable;
import javax.enterprise.context.RequestScoped;
import javax.inject.Named;

/**
 *
 * @author Vladimir Milicevic
 */
@Named
@RequestScoped
public class Potrosac implements Serializable {

    private String ime;
    private String prezime;
    private String email;

    public Potrosac() {
    }
}

```

```

public String getIme() {
    return ime;
}

public void setIme(String ime) {
    this.ime = ime;
}

public String getPrezime() {
    return prezime;
}

public void setPrezime(String prezime) {
    this.prezime = prezime;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}
}

```

U nastavku, sledi analiza priloženog listinga zrna *Potrosac.java*. Prvo što je moguće uočiti jeste prisutvo anotacije **@Named**. Klasa koja je obeležena ovom anotacijom predstavlja klasu CDI zrna. Po osnovnim podešavanjima, pogledati kod priložene *JSF* stranice, naziv zrna odgovara nazivu njegove klase sa razlikom što počinje malim slovom. Tako je, u konkretnom slučaju, naziv *CDI* zrna *potrosac* za kreiranu klasu CDI zrna *Potrosac*. Naravno da ovo ponašanje može da bude redefinisano. Ukoliko se želi da zrno nosi drugačiji naziv, taj naziv je moguće proslediti kao atribut **value** anotacije **@Named** na sledeći način:

```
@Named(value="zrnoPotrosac")
```

## METODE, OSOBINE I OBLASTI CDI ZRNA

*Metode i osobine CDI zrna dostupne su putem fejsleta.*

Metode i osobine *CDI zrna* dostupne su putem fejsleta na sličan način kao što je slučaj sa *JSF upravljanim zrnima*.

Kao i u slučaju *JSF upravljanih zrna*, postoji više oblasti u kojima mogu da funkcionišu CDI zrna. Ove oblasti su priložene i objašnjene sledećom tabelom.

Oblast	Anotacija	Opis
Request	@RequestScoped	Zrna iz ove oblasti dele se kroz trajanje jedinstvenog zahteva. Ovaj zahtev može da odgovara HTTP zahtevu, pozivu metode u EJB, pozivu veb servisa ili slanju JMS poruke zrnu vođenom porukama.
Session	@SessionScoped	Zrna iz ove oblasti dele se kroz sve zahteve koji su prisutni u HTTP sesiji. Svaki korisnik aplikacije dobija vlastitu instancu zrna iz oblasti session.
Application	@ApplicationScoped	Zrna iz ove oblasti žive tokom celokupnog živornog ciklusa aplikacije. Zrna se dele tokom korisničkih sesija.
Conversation	@ConversationScoped	Ova oblast može da obuhvati više zahteva i, obično, kraća je od oblasti Session.
Dependent	@Dependent	Zrna iz ove oblasti se ne dele. Svaki puta kada se umetne zrno ovog tipa, kreira se nova instanca.

Slika 1.2 Oblasti CDI zrna [izvor: autor]

Pretodna zrna su koristila oblast Request i bila su obeležena anotacijom @RequestScoped.

Prvo što je moguće primetiti da *CDI* zrna podržavaju primenu oblasti koje su ekvivalentne svim *JSF* oblastima. Međutim, ovde postoje i izvesna proširenja. *CDI* uvodi i dve dodatne oblasti: **Conversation** i **Dependent**. Prva oblast koja će biti pomenuta je **Conversation** i ona može da obuhvati više zahteva ali je kraća od oblasti **Session**. Za zrna iz oblasti **Dependent** važi da su nedeljiva i da prilikom umetanja svakog novog zrna dolazi do kreiranja nove instance.

## ZRNO KONTROLERA

*Analizu ove problematike je moguće dodatno proširiti predstavljanjem zrna kontrolera.*

Analizu ove problematike je moguće dodatno proširiti predstavljanjem *zrna kontrolera*. Ovo će biti drugo zrno u primeru koji je osnov izlaganja ovog dela lekcije. Zrno je predstavljeno sledećom klasom.

```
package com.metropolitan.cdideo.controller;

import com.metropolitan.cdideo.Potrosac;
import javax.enterprise.context.RequestScoped;
import javax.inject.Inject;
import javax.inject.Named;

/**
 *
 * @author Vladimir Milicevic
 */
```

```
@Named
@RequestScoped
public class PotrosacController {

    @Inject
    private Potrosac potrosac;

    public Potrosac getPotrosac() {
        return potrosac;
    }

    public void setPotrosac(Potrosac potrosac) {
        this.potrosac = potrosac;
    }

    public String navigateToConfirmation() {
//U realnoj aplikaciji
//korisnicki pdaci bi ovde bili sacuvani u bazi podataka.
        return "potvrda";
    }
}
```

U prethodnoj klasi, instanca klase *Potrosac* je umetnuta tokom vremena izvršavanja (*runtime*). Navedeno je realizovano primenom anotacije **@Inject**. Ova anotacija omogućava jednostavnu primenu *umetanja zavisnosti* u CDI aplikacijama. Pošto je klasa *Potrosac* obeležena anotacijom **@RequestScoped**, nova instanca klase *Potrosac* biće umetnuta sa svakim narednim zahtevom.

Metoda *navigateToConfirmation()*, priložene klase kontrolera, potiva se kada korisnik klikne na dugme *Potvrđi* na početnoj JSF stranici. Metoda *navigateToConfirmation()* funkcioniše na ekvivalentan način kao što bi funkcionisala metoda *JSF upravljanog zrna*. To znači da metoda vraća string i aplikacija vrši navigaciju ka stranici u zavisnosti od vrednosti pomenutog stringa. Kao što je bio slučaj, kada je diskusija bila bazirana na *JSF*, navedena stranica će imati ekstenziju *.xhtml* dok će njen naziv odgovarati povratnoj vrednosti metode *navigateToConfirmation()*.

## STRANICA ZA POTVRDU REZULTATA

*Pozivom metode `navigateToConfirmation()` vrši se navigacija ka stranica za potvrdu rezultata.*

Kao što je napomenuto, u prethodnom izlaganju, ukoliko ne dođe do objavljivanja nekog od izuzetaka, pozivom metode *navigateToConfirmation()* vrši se navigacija ka stranica za potvrdu rezultata. Ako se pogleda u kod klase kontrolera, moguće je primetiti da naziv ove stranice glasi *potvrda*, a ekstenzija njoj odgovarajuće datoteke *.xhtml*. Stranica je priložena sledećim listingom.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```



```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Uspesno!!!</title>
  </h:head>
  <h:body>
    Novi potrosac je uspesno kreiran.
    <h:panelGrid columns="2" border="1" cellpadding="0">
      <h:outputLabel for="ime" value="Ime"/>
      <h:outputText id="ime" value="#{potrosac.ime}"/>
      <h:outputLabel for="prezime" value="Prezime"/>
      <h:outputText id="prezime" value="#{potrosac.prezime}"/>
      <h:outputLabel for="email" value="Email Adresa"/>
      <h:outputText id="email" value="#{potrosac.email}"/>
    </h:panelGrid>
  </h:body>
</html>
```

Rezultati dobijeni unosom podataka u početnu stranicu *index.xhtml* (Slika 3) i prikazani na stranici *potvrda.xhtml* (Slika 4) ne odstupaju mnogo od izlaganja koje je bilo tema lekcija koje su u prvi plan stavljale *JSF*.

**Kreiranje novog potrosaca**

Ime: Vladimir

Prezime: Milicevic

Email Adresa: vladam.kg@gmail.com

Potvrdi

Slika 1.3 Unos podataka za novog potrosaca [izvor: autor]

Novi potrosac je uspesno kreiran.

Ime	Vladimir
Prezime	Milicevic
Email Adresa	vladam.kg@gmail.com

Slika 1.4 Prikazivanje podataka za novog potrosaca [izvor: autor]

Kao što je moguće primetiti CDI aplikacija funkcioniše na isti način kao JSF aplikacija uz izvesna proširenja i unapređenja. Prednosti primene CDI aplikacija se očitavaju u mogućnosti korišćenja dodatnih oblasti koje JSF ne podržava. Takođe, CDI dozvoljava da se zrna sesije koriste kao obeležena (CDI) zrna.

## ▼ Poglavlje 2

# Kvalifikatori

## UVODNA RAZMATRANJA O KVALIFIKATORIMA

*CDI obezbeđuje kvalifikatore za ukazivanje specifičnog tipa kojeg je neophodno umetnuti u kod .*

U nekim slučajevima, tip zrna koji je neophodno umetnuti u aktuelni kod predstavlja interfejs ili Java superklasa . Međutim, moguće je umetnuti i potklasu ili klasu koja implementira neki interfejs. U ovakvim slučajevima, CDI obezbeđuje kvalifikatore koji mogu biti upotrebljeni za ukazivanje specifičnog tipa kojeg je neophodno umetnuti u kod koji se razvija.

*CDI kvalifikator* je specifična anotacija predstavljena rezervisanom rečju **@Qualifier**. Ova anotacija može biti upotrebljena za obeležavanje specifične potklase ili nekog interfejsa.

Kao što je već ustanovljena praksa, u ovim materijalima, konkretan primer će nastaviti da daje podršku analizi. U tom svetlu će biti kreiran kvalifikator, pod nazivom *Premium*, za postojeće zrno potrošača. Implementacijom konkretne logike, premijum potrošači će dobiti izvesne pogodnosti koje obični potrošači nemaju. Te prednosti mogu biti, na primer, dodatni popusti prilikom kupovine.

Pre upuštanja u dalju analizu i demonstraciju, moguće je pogledati odgovarajući video materijal: *CDI Part one - Context and Dependency Injection*, kao veliku prednost poslednje verzije Java Enterprise platforme.

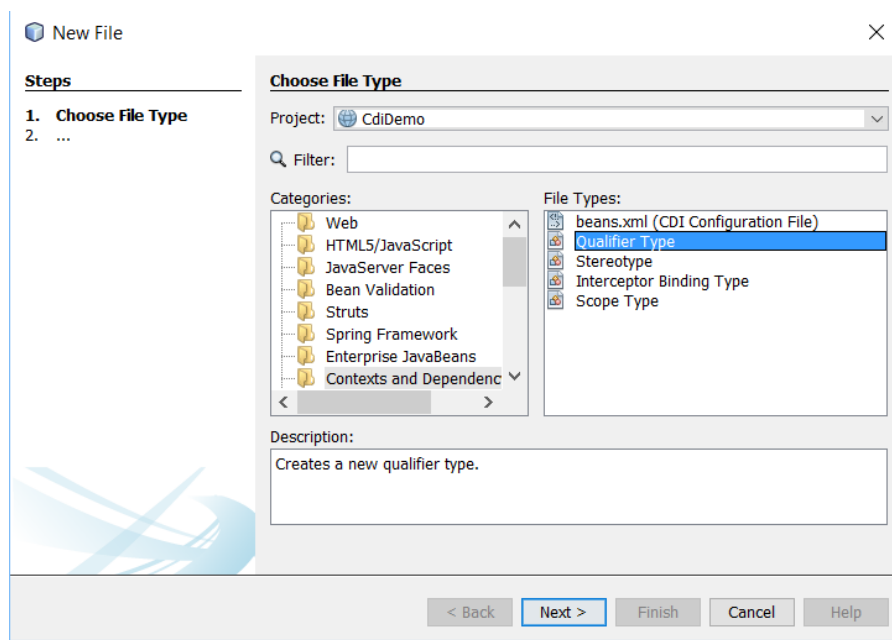
**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## KREIRANJE KVALIFIKATORA

*U narednom izlaganju akcenat će biti na kreiranju kvalifikatora i njihovoj primeni.*

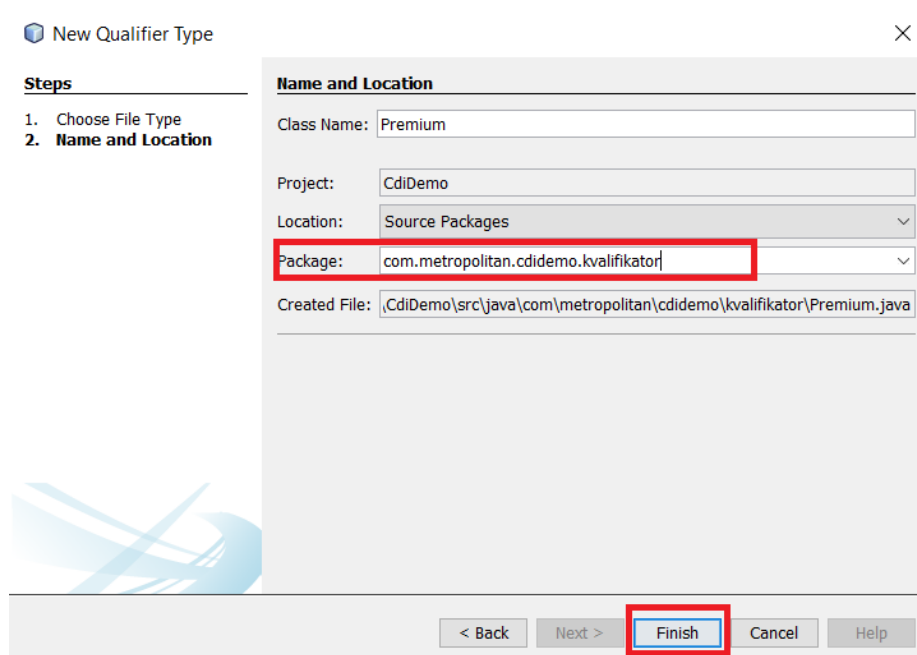
U narednom izlaganju akcenat će biti na kreiranju kvalifikatora i njihovoj primeni.

Slede instrukcije pomoću koji se u razvojnom okruženju kreiraju *kvalifikatori*. Za početak se bira *File| New File*, a zatim iz kategorije *Contexts and Dependency Injection* bira se tip datoteke *Qualifier Type* (pogledati Sliku 1).



Slika 2.1 Izbor tipa datoteke Qualifier Type [izvor: autor]

U nastavku, neophodno je za kvalifikatora izabrati pogodan naziv, kao i naziv paketa kojem će pripadati. Klikom na dugme **Next**, na prethodno prikazanom prozoru, otvoriće se novi prozor u kojem je moguće obaviti ovu aktivnost. Navedeno je prikazano sledećom slikom.

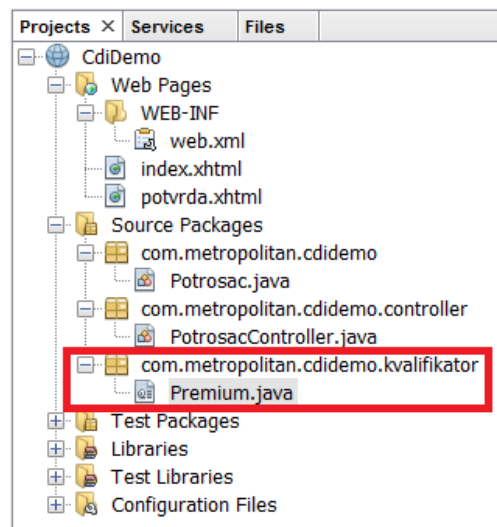


Slika 2.2 Završna podešavanja kvalifikatora [izvor: autor]

## KOD KVALIFIKATORA

*Praćenjem koraka, diktiranih NetBeans čarobnjakom, generisan je Java kod kvalifikatora.*

Kao što je moguće primetiti, praćenjem koraka, diktiranih NetBeans čarobnjakom, generisana je datoteka *kvalifikatora*. Sledećom slikom je prikazan položaj kreirane datoteke *Premium.java* u hijerarhiji projekta.



Slika 2.3 Položaj kreirane datoteke Premium.java u hijerarhiji projekta [izvor: autor]

Kada se pogleda sadržaj ove datoteke, moguće je primetiti da je ona snabdevena Java kodom koji uključuje primenu izvesnih anotacija. *Kvalifikator* je standardna *Java* anotacija označena sa *@Qualifier*. *Kvalifikatori*, tipično, poseduju mogućnost zadržavanja (retention) vremena izvršavanja i mogu da ciljaju na metode, polja, parametre ili tipove.

Sledećim listingom je priložen kod generisanog kvalifikatora *Premium*.

```
package com.metropolitan.cdideo.kvalifikator;

import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.PARAMETER;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import javax.inject.Qualifier;

/**
 *
 * @author Vladimir Milicevic
 */
@Qualifier
@Retention(RUNTIME)
@Target({METHOD, FIELD, PARAMETER, TYPE})
public @interface Premium {
}
```

## PRIMENA KVALIFIKATORA

*Kada je kvalifikator kreiran, moguće ga je upotrebiti za obeležavanje potklase ili interfejsa.*

U prethodnom izlaganju je pokazano kako se kvalifikator kreira. U nastavku, kada je kvalifikator kreiran, moguće ga je upotrebiti za obeležavanje potklase ili interfejsa koji se implementira.

Da bi ovo bilo obavljeno neophodno je, u aktuelnom primeru, kreirati potklasu koja će odgovarati premijum potrošaču i koja će biti obeležena kvalifikatorom `@Premium`. Klasa je priložena sledećim listingom.

```
package com.metropolitan.cdideo;

import com.metropolitan.cdideo.kvalifikator.Premium;
import javax.enterprise.context.RequestScoped;
import javax.inject.Named;
import com.metropolitan.cdideo.Potrosac;

/**
 *
 * @author Vladimir Milicevic
 */
@Named
@RequestScoped
@Premium
public class PremiumPotrosac extends Potrosac {

    private Integer kodPopusta;

    public Integer getKodPopusta() {
        return kodPopusta;
    }

    public void setKodPopusta(Integer kodPopusta) {
        this.kodPopusta = kodPopusta;
    }
}
```

U nastavku, pošto je obeležena specifična instanca koja se kvalifikuje, neophodno je primeniti i kvalifikatore u klijent kodu za specificiranje tačnog tipa zavisnosti koja je potrebna. Neophodno je kreirati kontroler za kreiranu klasu `PremiumPotrosac`. Ova klasa se kreira na dobro poznat način, kao standardna Java klasa, sa izabranim nazivom i paketom. Kod klase je priložen sledećim listingom.

```
package com.metropolitan.cdideo.controller;

import com.metropolitan.cdideo.Potrosac;
import com.metropolitan.cdideo.kvalifikator.Premium;
```

```
import com.metropolitan.cdideo.PremiumPotrosac;

import java.util.logging.Level;
import java.util.logging.Logger;
import javax.enterprise.context.RequestScoped;
import javax.inject.Inject;
import javax.inject.Named;

/**
 *
 * @author Vladimir Milicevic
 */
@Named
@RequestScoped
public class PremiumPotrosacController {

    private static final Logger logger = Logger.getLogger(
        PremiumPotrosacController.class.getName());

    @Inject
    @Premium
    private Potrosac potrosac;

    public String savePotrosac() {
        PremiumPotrosac premiumPotrosac
            = (PremiumPotrosac) potrosac;
        logger.log(Level.INFO, "Cuvaju se sledece informacije \n"
            + "{0} {1}, kod popusta = {2}",
            new Object[]{premiumPotrosac.getIme(),
                premiumPotrosac.getPrezime(),
                premiumPotrosac.getKodPopusta()});
        //U realnoj aplikaciji ovde bi isao kod kojim bi se
        //podaci potrosaca cuvali u bazi podataka
        return "premium_potrosac_potvrda";
    }
}
```

## PRIKAZIVANJE PODATAKA KAO REZULTATA PRIMENE KVALIFIKATORA

*Instanca klase **PremiumCustomer** je umetnuta u to polje obeleženo **@Premium** kvalifikatorom.*

Pošto je iskorišćena anotacija **@Premium** za obeležavanje polja potrošača (pogledati prethodni listing), instanca klase **PremiumCustomer** je umetnuta u to polje. Ovo je omogućeno iz razloga što je i navedena klasa obeležena **@Premium** kvalifikatorom.

U nastavku je neophodno kreirati stranice pod nazivom **premium\_potrosac.xhtml** i **premium\_potrosac\_potvrda.xhtml**, koje će uzimati i prikazati podatke unete za potrošača koji ima status premijum, respektivno. Naziv poslednje datoteke je vraćen kao povratna vrednost

metode kontrolera `public String saveCustomer()` (videti prethodno priloženi kod). Sledi listing datoteke `premium_potrosac.xhtml`.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Kreiranje novog PREMIUM potrosaca</title>
    </h:head>
    <h:body>
        <h:form>
            <h3>Kreiranje novog PREMIUM potrosaca</h3>
            <h:panelGrid columns="3">
                <h:outputLabel for="ime" value="Ime"/>
                <h:inputText id="ime"
                    value="#{premiumPotrosac.ime}"/>
                <h:message for="ime"/>
                <h:outputLabel for="prezime" value="Prezime"/>
                <h:inputText id="prezime"
                    value="#{premiumPotrosac.prezime}"/>
                <h:message for="prezime"/>
                <h:outputLabel for="email" value="Email Adresa"/>
                <h:inputText id="email"
                    value="#{premiumPotrosac.email}"/>
                <h:message for="email"/>
                <h:outputLabel for="kodPopusta" value="Kod popusta"/>
                <h:inputText id="discountCode"
                    value="#{premiumPotrosac.kodPopusta}"/>
                <h:message for="kodPopusta"/>
            </h:panelGrid>
            <h:panelGroup>
                <h:commandButton value="Potvrđi"
                    action="#{premiumPotrosacController.
                        savePotrosac}"/>
            </h:panelGroup>
        </h:form>
    </h:body>
</html>
```

Sledi listing datoteke `premium_potrosac_potvrda.xhtml`.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Uspesno!!!</title>
    </h:head>
    <h:body>
        Novi PREMIUM potrosac je uspesno generisan
        <h:panelGrid columns="2" border="1" cellpadding="0">
```

```
<h:outputLabel for="ime" value="Ime"/>
<h:outputText id="ime" value="#{premiumPotrosac.ime}"/>

<h:outputLabel for="prezime" value="Prezime"/>
<h:outputText id="prezime" value="#{premiumPotrosac.prezime}"/>

<h:outputLabel for="email" value="Email Adresa"/>
<h:outputText id="email" value="#{premiumPotrosac.email}"/>

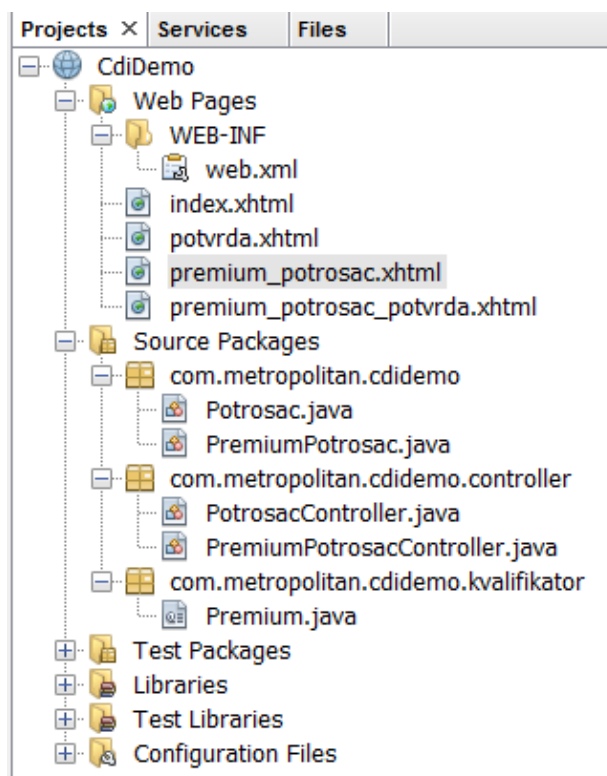
<h:outputLabel for="kodPopusta" value="Kod popusta"/>
<h:outputText id="kodPopusta" value="#{premiumPotrosac.kodPopusta}"/>

</h:panelGrid>
</h:body>
</html>
```

## DEMONSTACIJA PRIMERA

*Kada su sve datoteke na svom mestu, neophodno je demonstrirati urađeno.*

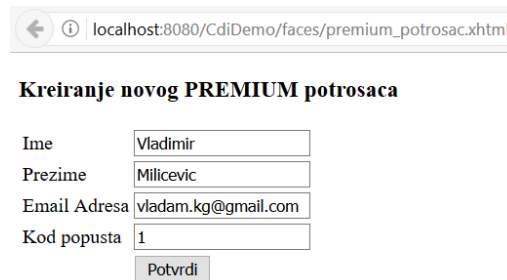
JSF stranice, koje su kreirane i demonstrirane na prethodnoj sekciji, su poslednje koje je bilo potrebno razviti i sa njima, a i prethodno kreiranim datotekama, projekat ima strukturu prikazanu sledećom slikom.



Slika 2.4 Trenutna struktura projekta [izvor: autor]



Projekat je neophodno prevesti i nakon toga je moguće uneti podatke za premijum potrošača, unosom linka, [http://localhost:8080/CdiDemo/faces/premium\\_potrosac.xhtml](http://localhost:8080/CdiDemo/faces/premium_potrosac.xhtml) u veb pregledač (sledeća slika).



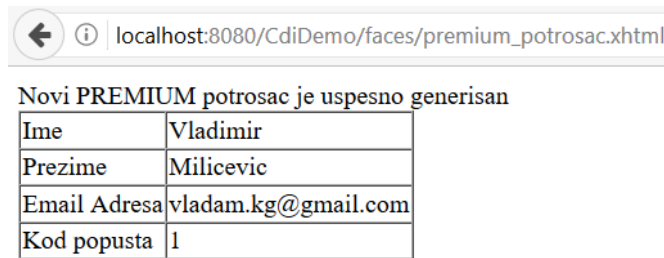
**Kreiranje novog PREMIUM potrosaca**

Ime	Vladimir
Prezime	Milicevic
Email Adresa	vladam.kg@gmail.com
Kod popusta	1

Potvrdi

Slika 2.5 Unos podataka za premijum potrošača [izvor: autor]

Klikom na dugme *Potvrdi*, ukoliko nije došlo do pojave izuzetaka, otvara se stranica [premium\\_potrosac\\_potvrda](#) koja prikazuje unete podatke za premijum potrošača (sledeća slika).



**Novi PREMIUM potrosac je uspesno generisan**

Ime	Vladimir
Prezime	Milicevic
Email Adresa	vladam.kg@gmail.com
Kod popusta	1

Slika 2.6 Prikaz podataka za premijum potrošača [izvor: autor]

## ▼ Poglavlje 3

# Stereotipovi

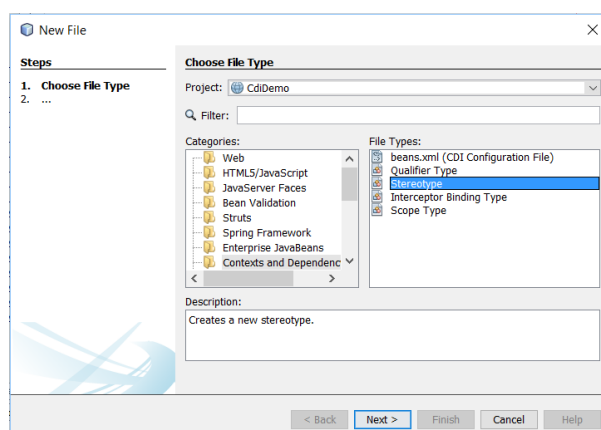
## KREIRANJE STEREOTIPA

*CDI stereotipovi dozvoljavaju kreiranje novih anotacija koje menjaju nekoliko CDI anotacija.*

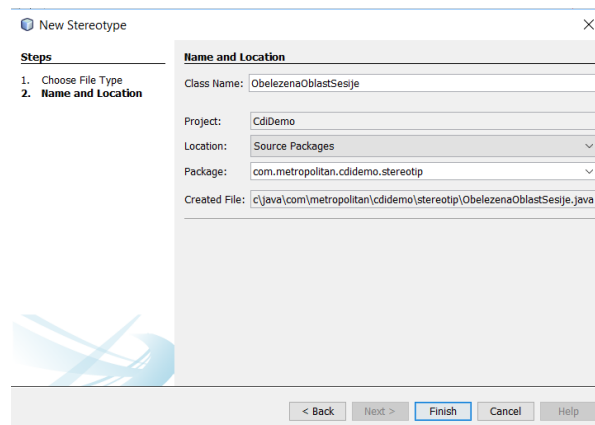
CDI stereotipovi dozvoljavaju kreiranje novih anotacija kojima je moguće zameniti grupu od nekoliko CDI anotacija. Na primer, ukoliko je neophodno izvršiti kreiranje nekoliko CDI zrna u okviru oblasti zahteva, biće neophodno primeniti dve konkretne anotacije na svako od zrna koje se kreira. U navedenom slučaju, radi se o anotacijama **@Named** i **@RequestScoped**. Umesto navedenih anotacija moguće je koristiti samo jednu anotaciju, označenu kao *stereotip*, pomoću koje je moguće obeležiti pomenuta zrna.

Za kreiranje CDI stereotipa, neophodno je koristiti neko savremeno razvojno Java okruženje. Neka to u ovom slučaju bude *NetBeans IDE* i neka se analiza i diskusija, u ovom delu lekcije, fokusira na proširenje i uvođenje novih funkcionalnosti u aktuelni primer. Dakle, primenom razvojnog okruženja, u aktuelnom projektu *CdiDemo* biće kreiran nov fajl iz kategorije *Contexts and Dependency Injection*, tipa *Stereotype*. Navedeno je prikazano Slikom 1.

Nakon obavljenog izbora, tipa datoteke koja se kreira, razvojno okruženje će obezbediti novi prozor u kojem je neophodno obaviti dopunska podešavanja stereotipa, a to je određivanje naziva datoteke, paketa kojem datoteka pripada i slično. Navedeno je prikazano Slikom 2.



Slika 3.1 Izbor kreiranja stereotipa [izvor: autor]



Slika 3.2 Inicijalna podešavanja datoteke stereotipa. [izvor: autor]

## KOD GENERISANOG STEREOTIPA I NJEGOVA MODIFIKACIJA

*Razvojno okruženje NetBeans je obavilo zadatak generisanja inicijalnog koda navedene datoteke.*

Kao što je moguće primetiti iz prethodnog izlaganja, kreirana je datoteka, pod nazivom *ObelezenaOblastZrna*, koja pripada Java paketu *com.metropolitan.cdidemo.stereotip*. Nakon obavljenih inicijalnih podešavanja ove datoteke, razvojno okruženje *NetBeans IDE* je obavilo zadatak generisanja inicijalnog koda navedene datoteke. Sledećim listingom je priložen kod datoteke *ObelezenaOblastZrna* koji je razvojno okruženje automatski generisalo.

```
package com.metropolitan.cdidemo.stereotip;

import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import javax.enterprise.inject.Stereotype;

/**
 *
 * @author Vladimir Milicevic
 */
@Stereotype
@Retention(RUNTIME)
@Target({METHOD, FIELD, TYPE})
public @interface ObelezenaOblastZrna {
}
```

U nastavku je neophodno izvršiti modifikaciju generisanog koda.

Da bi modifikacija bila obavljena, na traženi način, neophodno je obezbediti da anotacija kreiranog *stereotipa* menja više anotacija. U konkretnom slučaju, neophodno je da se ona koristi umesto anotacija *@Named* i *@RequestScoped*. U tom kontekstu, ovim anotacijama ne neophodno dopuniti prethodni kod. Prethodni kod datoteke *stereotipa*, nakon modifikacije, dobiće oblik koji odgovara kodu koji je priložen sledećim listingom.

```
package com.metropolitan.cdideo.stereotip;

import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import javax.enterprise.context.RequestScoped;
import javax.enterprise.inject.Stereotype;
import javax.inject.Named;

/**
 *
 * @author Vladimir Milicevic
 */
@Named
@RequestScoped
@Stereotype
@Retention(RUNTIME)
@Target({METHOD, FIELD, TYPE})
public @interface ObelezenaOblastZrna {
}
```

## IMPLEMENTACIJA STEREOTIPA NA CDI DATOTEKE PROJEKTA

*U svim CDI zrnima primera biće implementiran kreirani stereotip.*

Neophodno je krenuti redom, navigacijom kroz strukturu projekta *CdiDemo*, i obaviti izmenu para anotacija *@Named* i *@RequestScoped*, korisničkom stereotipnom anotacijom *@ObelezenaOblastZrna* koja je kreirana kao rezultat poslednje izmene u kodu odgovarajuće datoteke.

U navedenom svetlu, prvo će biti modifikovana datoteka *Potrosac.java* čiji je kod, nakon modifikacije koja je obrazložena u gora navedenom izlaganju, dobio oblik koji je priložen sledećim listingom.

```
package com.metropolitan.cdideo;

import java.io.Serializable;
import javax.enterprise.context.RequestScoped;
import javax.inject.Named;
import com.metropolitan.cdideo.stereotip.ObelezenaOblastZrna;
```

```
/**
 *
 * @author Vladimir Milicevic
 */
@ObelezenaOblastZrna
public class Potrosac implements Serializable {

    private String ime;
    private String prezime;
    private String email;

    public Potrosac() {
    }

    public String getIme() {
        return ime;
    }

    public void setIme(String ime) {
        this.ime = ime;
    }

    public String getPrezime() {
        return prezime;
    }

    public void setPrezime(String prezime) {
        this.prezime = prezime;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

Zatim, u skladu sa prethodnom diskusijom, biće modifikovana datoteka *PremiumPotrosac.java* čiji je kod, nakon modifikacije koja je obrazložena u gora navedenom izlaganju, dobio oblik koji je priložen sledećim listingom.

```
import com.metropolitan.cdideo.kvalifikator.Premium;
import javax.enterprise.context.RequestScoped;
import javax.inject.Named;
import com.metropolitan.cdideo.Potrosac;
import com.metropolitan.cdideo.stereotip.ObelezenaOblastZrna;

/**
 *
```

```
* @author Vladimir Milicevic
*/
@ObelezenaOblastZrna
@Premium
public class PremiumPotrosac extends Potrosac {

    private Integer kodPopusta;

    public Integer getKodPopusta() {
        return kodPopusta;
    }

    public void setKodPopusta(Integer kodPopusta) {
        this.kodPopusta = kodPopusta;
    }
}
```

Iz priloženih listinga je moguće primetiti da je primena anotacije @ObelezenaOblastZrna praćena `import` instrukcijom `import com.metropolitan.cdideo.stereotip.ObelezenaOblastZrna`.

## MODIFIKOVANA KLASA POTROSACCONTROLLER

*Korekcije će biti izvršene u svim CDI datotekama projekta.*

Korekcije koje su demonstrirane nad prethodnim dvema datotekama, biće primenjene i na preostale, a nakon toga će primer ponovo biti testiran sa ciljem provere uspešnosti obavljenog zadatka upotrebe *stereotipova*.

U skladu sa prethodnom diskusijom, biće modifikovana datoteka *PotrosacController.java* čiji je kod, nakon modifikacije koja je obrazložena u prethodno izlaganju, dobio oblik koji je priložen sledećim listingom.

```
package com.metropolitan.cdideo.controller;

import com.metropolitan.cdideo.Potrosac;
import javax.enterprise.context.RequestScoped;
import javax.inject.Inject;
import javax.inject.Named;
import com.metropolitan.cdideo.stereotip.ObelezenaOblastZrna;

/**
 *
 * @author Vladimir Milicevic
 */
@ObelezenaOblastZrna
public class PotrosacController {
```

```

@Inject
private Potrosac potrosac;

public Potrosac getPotrosac() {
    return potrosac;
}

public void setPotrosac(Potrosac potrosac) {
    this.potrosac = potrosac;
}

public String navigateToConfirmation() {
//In a real application we would
//Save customer data to the database here.
    return "potvrda";
}
}

```

Konačno, biće modifikovana datoteka *PremiumPotrosacController.java* čiji je kod, nakon modifikacije koja je obrazložena u prethodno izlaganju, dobio oblik koji je priložen sledećim listingom.

```

package com.metropolitan.cdideo.controller;

import com.metropolitan.cdideo.Potrosac;
import com.metropolitan.cdideo.kvalifikator.Premium;
import com.metropolitan.cdideo.PremiumPotrosac;

import java.util.logging.Level;
import java.util.logging.Logger;
import javax.enterprise.context.RequestScoped;
import javax.inject.Inject;
import javax.inject.Named;
import com.metropolitan.cdideo.stereotip.ObelezenaOblastZrna;

/**
 *
 * @author Vladimir Milicevic
 */
@ObelezenaOblastZrna
public class PremiumPotrosacController {

    private static final Logger logger = Logger.getLogger(
        PremiumPotrosacController.class.getName());

    @Inject
    @Premium
    private Potrosac potrosac;

    public String savePotrosac() {
        PremiumPotrosac premiumPotrosac
            = (PremiumPotrosac) potrosac;
        logger.log(Level.INFO, "Cuvaju se sledece informacije \n"

```

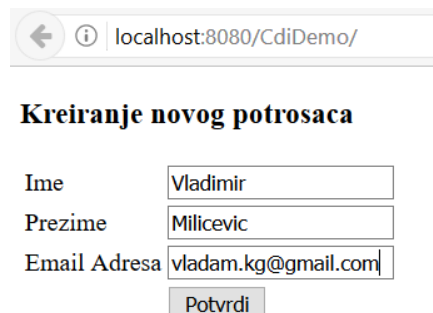
```
+ "{0} {1}, kod popusta = {2}",
new Object[]{premiumPotrosac.getIme(),
    premiumPotrosac.getPrezime(),
    premiumPotrosac.getKodPopusta()});
//If this was a real application, we would have code to save
//customer data to the database here.
return "premium_potrosac_potvrda";
}
}
```

## DEMONSTRACIJA PRIMERA SA STEREOTIPOVIMA

*Nakon obavljenih modifikacija, neophodno je ponovo testirati program.*

Nakon obavljenih korekcija, primer je kompletiran i kao takav zahteva testiranje sa ciljem provere da li je sav posao korektno obavljen. U razvojno okruženju *NetBeans IDE* vrši se prevođenje projekta da bi sve izmene u aplikaciji bile dostupne kada se program ponovo pokrene.

Prvo, testiranje će biti usmereno kao formi koja se nalazi na početnoj *index.xhtml* stranici. Stranica sa popunjenom formom, ima sledeći izgled.



← ⓘ localhost:8080/CdiDemo/

**Kreiranje novog potrosaca**

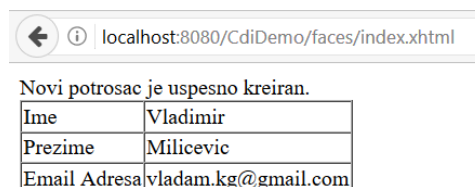
Ime

Prezime

Email Adresa

Slika 3.3 Popunjena forma na početnoj stranici [izvor: autor]

Klikom na dugme *Potvrdi*, otvara se stranica za prikazivanje rezultata.



← ⓘ localhost:8080/CdiDemo/faces/index.xhtml

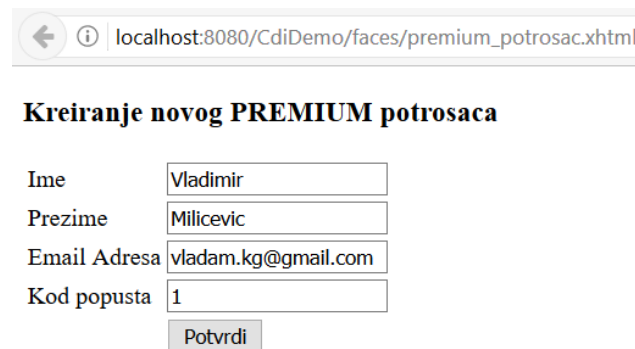
Novi potrosac je uspesno kreiran.

Ime	Vladimir
Prezime	Milicevic
Email Adresa	vladam.kg@gmail.com

Slika 3.4 Stranica za prikaz unetih podataka za potrošača [izvor: autor]

Dalje, biće proverene i modifikacije za CDI zrna premijum potrošača. Pozivom linka [http://localhost:8080/CdiDemo/faces/premium\\_potrosac.xhtml](http://localhost:8080/CdiDemo/faces/premium_potrosac.xhtml), u veb pregledaču, otvara se stranica sa formom za unos podataka kao na sledećoj slici.





Kreiranje novog PREMIUM potrosaca

Ime: Vladimir

Prezime: Milicevic

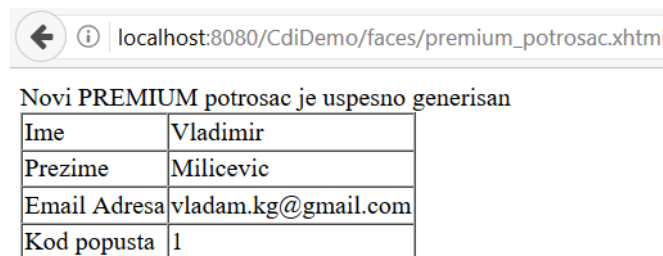
Email Adresa: vladam.kg@gmail.com

Kod popusta: 1

Potvrdi

Slika 3.5 Popunjena forma na stranici premium\_potrosac.xhtml [izvor: autor]

Klikom na dugme *Potvrdi*, otvara se stranica za prikazivanje rezultata.



Novi PREMIUM potrosac je uspesno generisan

Ime	Vladimir
Prezime	Milicevic
Email Adresa	vladam.kg@gmail.com
Kod popusta	1

Slika 3.6 Stranica za prikaz unetih podataka za premijum potrošača [izvor: autor]

## DOPUNSKA RAZMATRANJA

*Dodatna pažnja će biti posvećena datoteci  
PremiumPotrosacController.java.*

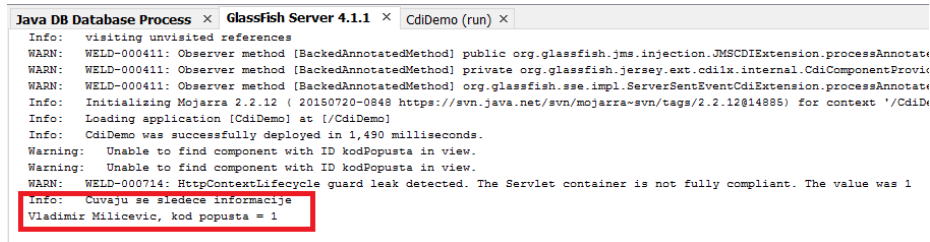
Još malo će biti vršena analiza tekućeg primera. Drugim rečima, dodatna pažnja će biti posvećena datoteci *PremiumPotrosacController.java*. Ako se pažljivo pogleda njen kod, moguće je primetiti da on poseduje i *Log* funkcionalnosti koje su ugrađene u metodu *savePotrosac()* ove klase. Klikom na dugme *Potvrdi*, forme sa stranice za unos podataka za premijum potrošača, dodatne informacije o kreiranom premijum korisniku će biti dostupne u prozoru koji odgovara aplikativnom serveru, *GlassFish* u ovom slučaju, i koje je moguće pročitati u razvojnom okruženju *NetBeans IDE*.

Sledećim listingom je izolovan kod metode *savePotrosac()* klase *PremiumPotrosacController.java*.

```
public String savePotrosac() {
    PremiumPotrosac premiumPotrosac
        = (PremiumPotrosac) potrosac;
    logger.log(Level.INFO, "Cuvaju se sledece informacije \n"
        + "{0} {1}, kod popusta = {2}",
        new Object[]{premiumPotrosac.getIme(),
            premiumPotrosac.getPrezime(),
            premiumPotrosac.getKodPopusta()});
    //If this was a real application, we would have code to save
```

```
//customer data to the database here.
    return "premium_potrosac_potvrda";
}
```

Dodatne informacije, proizvedene unetim podacima kao na Slici 5, u monitoru aplikativnog servera. a u okviru razvojnog okruženja *NetBeans IDE*, predstavljene su na način prikazan sledećom slikom.



Slika 3.7 GlassFish Server 4.1.1 monitor [izvor: autor]

Nakon demonstracije i dopunskih razmatranja potpuno je jasno da su svi zadaci sa kreiranjem i implementacijom stereotipova, u CDI aplikaciji primera, uspešno obavljeni.

## ▼ Poglavlje 4

# Tipovi povezivanja presretača

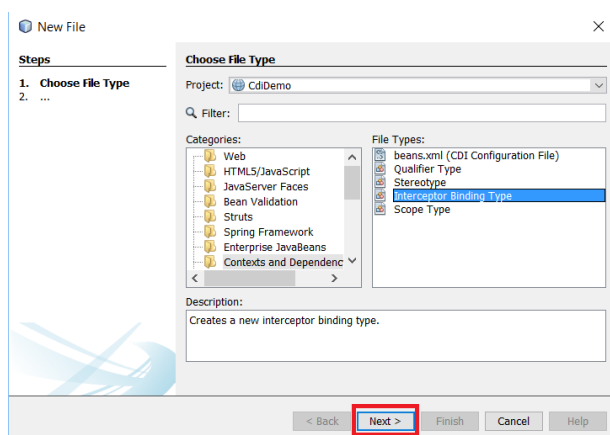
## KREIRANJE TIPRA ZA POVEZIVANJE PRESRETAČA

*CDI dozvoljava pisanje tipova za povezivanje (binding) presretača.*

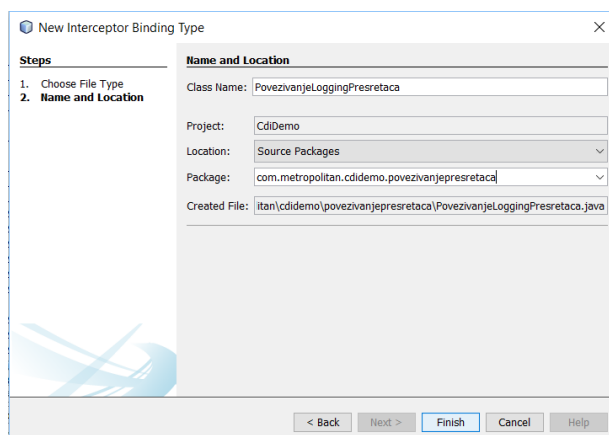
Jedna od glavnih prednosti *Java distribuiranih aplikacija* koja koriste *EJB zrna* je odlična podrška za implementaciju aspektno - orijentisanog programiranja preko koncepta presretača (*interceptors*). CDI dozvoljava pisanje tipova za povezivanje (*binding*) presretača. Na ovaj način je omogućeno povezivanje presretača sa zrnom pri čemu zrna ne moraju direktno da zavise (*dependency*) od presretača. Tipovi povezivanja presretača (*Interceptor binding types*) predstavljaju anotacije predstavljene pomoću rezervisane reči *@InterceptorBinding*.

Kreiranje *tipovi povezivanja presretača* u savremenim razvojnim okruženjima je veoma jednostavno. Posebno je ovaj zadatak pojednostavljen kada se koristi *NetBeans IDE* razvojno okruženje. U aktuelnom projektu, na dobro poznat način, bira se opcija za kreiranje nove datoteke. Zatim se iz kategorije *Contexts and Dependency Injection* bira tip datoteke *Interceptor Binding Type* (pogledati Sliku 1)

Nakon obavljenog izbora, tipa datoteke koja se kreira, razvojno okruženje će obezbediti novi prozor u kojem je neophodno obaviti dopunska podešavanja za *tip povezivanja presretača*, a to je određivanje naziva datoteke, paketa kojem datoteka pripada i slično. Navedeno je prikazano Slikom 2.



Slika 4.1 Izbor kreiranja tipa povezivanja presretača [izvor: autor]

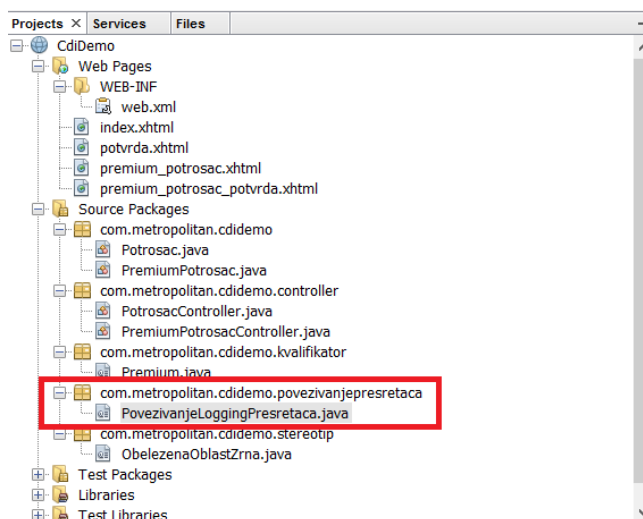


Slika 4.2 Inicijalna podešavanja datoteke tipa povezivanja presrećača [izvor: autor]

## KOD TIPA ZA POVEZIVANJE PRESRETAČA

*Nakon obavljenih inicijalni podešavanja, generisan je kod tipa za povezivanje presrećača.*

Kao što je moguće primetiti, a na osnovu prethodnog izlaganja, obavljena su inicijalna podešavanja datoteke *tipa za povezivanje presrećača*. Prvo je iskorišćen *NetBeans* čarobnjak da bi omogućio kreiranje nove datoteke ovog tipa, a zatim je u narednom prozoru dodeljen naziv ovoj datoteci, u konkretnom slučaju *PovezivanjeLoggingPresretaca.java*, kao i naziv paketa kojem će kreirana datoteka *tipa za povezivanje presrećača* pripadati. Paket u konkretnom slučaju nosi naziv *com.metropolitan.cdideo.povezivanjepresretaca*. Datoteka i paket su zauzeli svoje mesto u strukturi projekta, kao što je to prikazano sledećom slikom.



Slika 4.3 Datoteka i paket tipa za povezivanje presrećača u strukturi projekta [izvor: autor]

Klikom na dugme *Finish* (pogledati Sliku 2), prozora za podešavanje datoteke *tipa za povezivanje presrećača*, završava se proces inicijalnog podešavanje ove datoteke i kreira se njen inicijalni kod. U ovom slučaju, kod je potpuno funkcionalan i nema potrebe za dodavanjem novih instrukcija u okviru njega. Da bi kreirani kod mogao da bude upotrebljen

na pravi način, neophodno je obaviti kreiranje presrećača koji mora da bude obeležen anotacijom koja odgovara *tipu za povezivanje presrećača*.

Sledećim listingom priložen je kod koji je razvojno okruženje NetBeans IDE automatski generisalo, na osnovu inicijalnih podešavanja, u datoteci *PovezivanjeLoggingPresretaca.java*.

```
package com.metropolitan.cdideo.povezivanjepresretaca;

import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import java.lang.annotation.Inherited;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import javax.interceptor.InterceptorBinding;

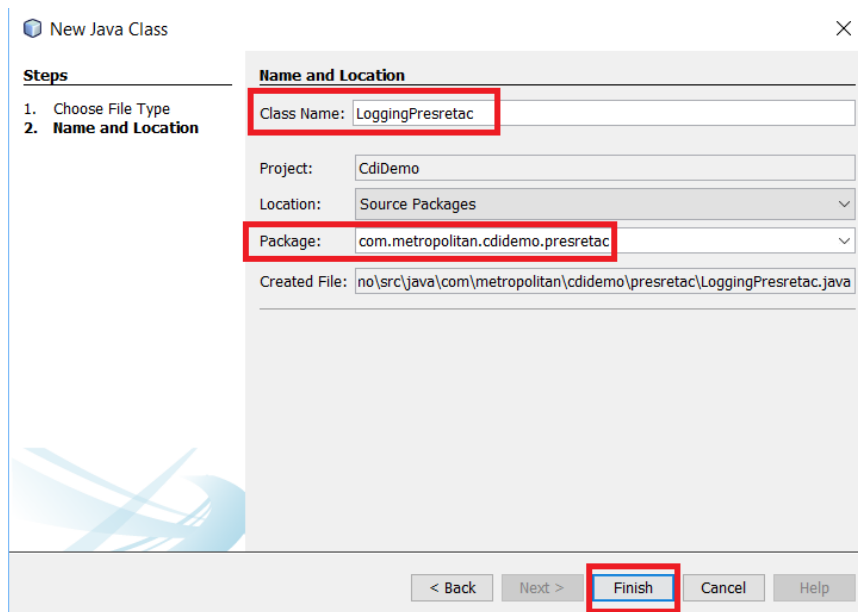
/**
 *
 * @author Vladimir Milicevic
 */
@Inherited
@InterceptorBinding
@Retention(RUNTIME)
@Target({METHOD, TYPE})
public @interface PovezivanjeLoggingPresretaca {
}
```

## KREIRANJE DATOTEKE PRESRETAČA

*Neophodno je u nastavku izvršiti kreiranje datoteke odgovarajućeg presrećača.*

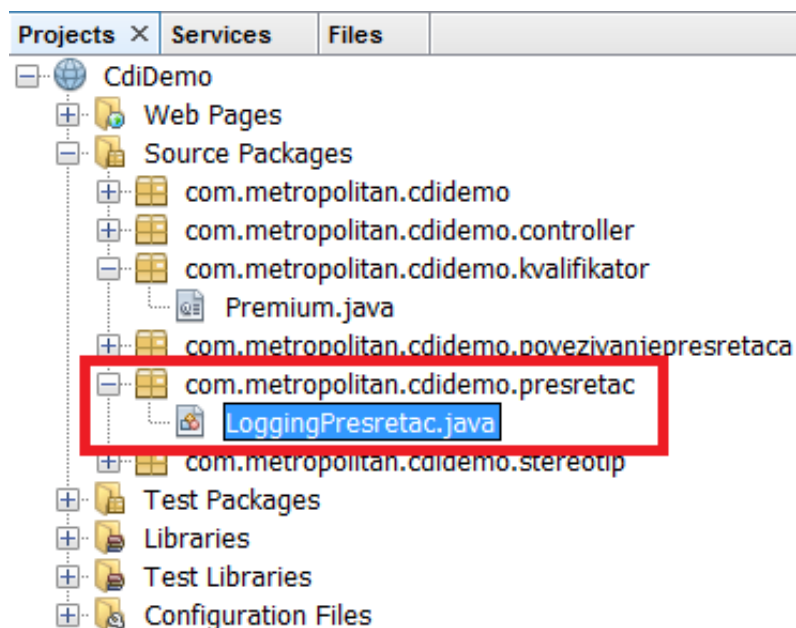
Kao što je istaknuto u prethodnom izlaganju, da bi kreirani kod mogao da bude upotrebljen na pravi način, neophodno je obaviti kreiranje *presrećača* koji mora da bude obeležen anotacijom koja odgovara *tipu za povezivanje presrećača*.

Datoteka može da bude kreirana prosto, kao obična Java klasa kojoj se na dobro poznat način, korišćenjem *NetBeans čarobnjaka*, dodeljuju naziv i paket kojem će pripadati. Navedeno je prikazano sledećom slikom.



Slika 4.4 Kreiranje datoteke presretača [izvor: autor]

Klikom na dugme *Finish*, *NetBeans* čarobnjaka, datoteka *presretača* i paket zauzimaju svoje mesto u strukturi projekta kao što je to prikazano sledećom slikom.



Slika 4.5 Datoteka presretača i paket u strukturi projekta [izvor: autor]

## KODIRANJE PRESRETAČA

*Klasu presretača je neophodno kodirati u narednom koraku.*

Sada je sve spremno da se datoteka *LoggingPresretac.java* popuni kodom koji će omogućiti upotrebu prethodno kreirane datoteke *tipa za povezivanje presretača*.

Prvo je neophodno primenom instrukcije **import** dodati kreirani *tip za povezivanje presretača* u kod presretača. Nakon toga sledi implementacija ostale logike *presretača*, a neposredno nakon primene anotacija *tipa za povezivanje presretača* **@PovezivanjeLoggingPresretacana** i presretača **@Interceptor**, na način priložen sledećim listingom.

```
package com.metropolitan.cdideo.presretac;

import com.metropolitan.cdideo.povezivanjepresretaca.PovezivanjeLoggingPresretaca;
import java.io.Serializable;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.interceptor.AroundInvoke;
import javax.interceptor.Interceptor;
import javax.interceptor.InvocationContext;

/**
 *
 * @author Vladimir Milicevic
 */
@PovezivanjeLoggingPresretaca
@Interceptor
public class LoggingPresretac implements Serializable {

    private static final Logger logger = Logger.getLogger(
        LoggingPresretac.class.getName());

    @AroundInvoke
    public Object logMethodCall(InvocationContext invocationContext)
        throws Exception {
        logger.log(Level.INFO, new StringBuilder("Ulazak ").append(
            invocationContext.getMethod().getName()).append(
                " metoda").toString());
        Object retVal = invocationContext.proceed();
        logger.log(Level.INFO, new StringBuilder("Izlazak ").append(
            invocationContext.getMethod().getName()).append(
                " metoda").toString());
        return retVal;
    }
}
```

## CDI KONFIGURACIONA DATOTEKA

*Neophodno je još malo podešavanja da bi tipovi povezivanja presretača radili na pravi način.*

Kao što je moguće primetiti iz prethodnog izlaganja, kreirana klasa *presretača*, osim što je obeležena anotacijom *tipa za povezivanje presretača*, predstavlja standardni *presretač* veoma sličan presretaču koji je korišćen za *EJB zrna sesije* (za više informacija pogledati

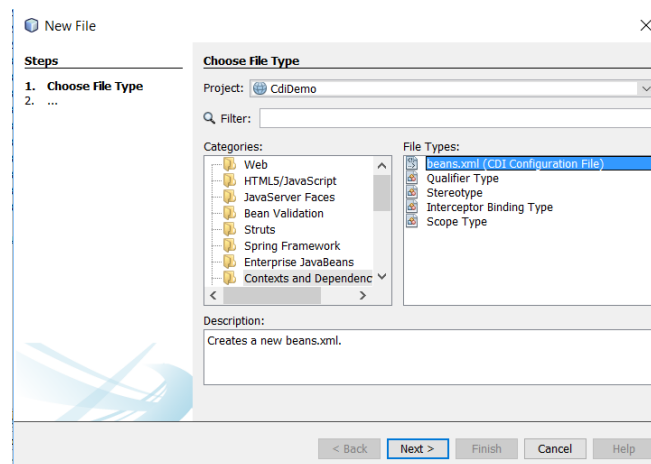
prethodnu lekciju). Ovde se analiza ne završava i potreban je još par koraka za njeno okončanje.

Sa ciljem da *tip za povezivanje presretača* funkcioniše na pravi način i kako se to od njega očekuje, neophodno je izvršiti u okviru kreiranog projekta, koji odgovara aktuelnom primeru, dopunska podešavanja. Navedeno podrazumeva kreiranje *CDI konfiguracione datoteke* koja će nositi zvaničan naziv *beans.xml*.

Datoteka se kreira na veoma jednostavan način. U razvojnom okruženju se bira dobro poznata opcija za kreiranje nove datoteke, a nakon toga, u otvorenom prozoru, bira se kategorija *Contexts and Dependency Injection*, nakon čega se bira tip datoteke *beans.xml* (*CDI Configuration File*).

Navedena aktivnost je ilustrovana dodavanjem sledeće slike. Klikom na dugme *Next*, otvara se prozor za potvrdu (*Finish*) kojim se završava proces kreiranja datoteke *beans.xml*.

Kada se kreira datoteka *beans.xml* u njoj ne neophodno registrovati presretača, kao u kodu koji je priložen odmah ispod Slike 6.



Slika 4.6 Kreiranje CDI konfiguracione datoteke [izvor: autor]

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/
xml/ns/javaee/beans_1_1.xsd"
       bean-discovery-mode="all">
  <interceptors>
    <class>
      com.metropolitan.cdidemo.presretac.LoggingPresretac
    </class>
  </interceptors>
</beans>
```



## KREIRANJE KONTROLERA

*Neophodno je dodati kontroler da bi tipovi povezivanja presretača radili na pravi način.*

Na samom kraju, neophodno je dodati kontroler da bi *tipovi povezivanja presretača* radili na pravi način. U konkretnom primeru neka to bude kontroler za premium potrošača, u konkretnom primeru, kojem odgovara klasa *PremiumPotrosacController.java*.

Dakle, sledeći korak koji je neophodno izvesti jeste obeležavanje klase kontrolera anotacijom *@PovezivanjeLoggingPresretaca* koja u konkretnom slučaju podrazumeva *tip povezivanja presretača*. Nakon toga je neophodno izvršiti ponovno prevođenje aplikacije da bi sve načinjene izmene u aplikaciji došle do izražaja i mogle da se koriste.

Sledećim listingom je priložen kod klase *PremiumPotrosacController.java* koji je modifikovan u skladu sa goe navedenim smernicama.

```
package com.metropolitan.cdideo.controller;

import com.metropolitan.cdideo.povezivanjepresretaca.PovezivanjeLoggingPresretaca;
import com.metropolitan.cdideo.Potrosac;
import com.metropolitan.cdideo.kvalifikator.Premium;
import com.metropolitan.cdideo.PremiumPotrosac;

import java.util.logging.Level;
import java.util.logging.Logger;
import javax.enterprise.context.RequestScoped;
import javax.inject.Inject;
import javax.inject.Named;
import com.metropolitan.cdideo.stereotip.ObelezenaOblastZrna;

/**
 *
 * @author Vladimir Milicevic
 */
@PovezivanjeLoggingPresretaca
@ObelezenaOblastZrna
public class PremiumPotrosacController {

    private static final Logger logger = Logger.getLogger(
        PremiumPotrosacController.class.getName());

    @Inject
    @Premium
    private Potrosac potrosac;

    public String savePotrosac() {
        PremiumPotrosac premiumPotrosac
            = (PremiumPotrosac) potrosac;
        logger.log(Level.INFO, "Cuvaju se sledece informacije \n"
            + "{0} {1}, kod popusta = {2}",
            new Object[]{premiumPotrosac.getIme(),
```

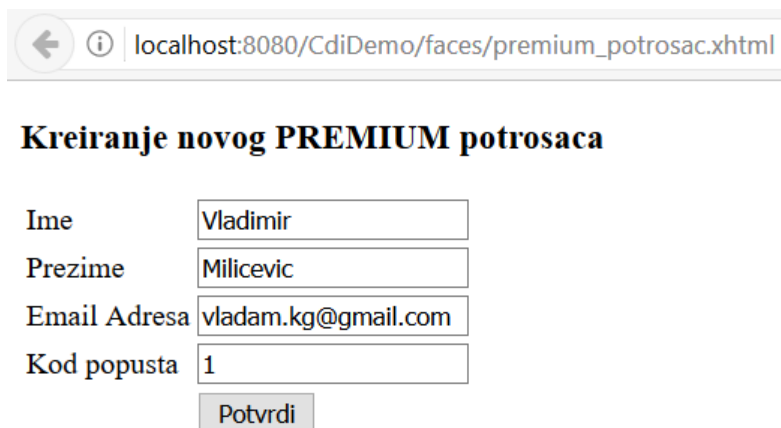
```
        premiumPotrosac.getPrezime(),  
        premiumPotrosac.getKodPopusta()});  
//If this was a real application, we would have code to save  
//customer data to the database here.  
    return "premium_potrosac_potvrda";  
}  
}
```

## TIPOVI POVEZIVANJA PRESRETAČA - DEMONSTRACIJA

*Neophodno je izvršiti testiranje kreiranog koda za tipove povezivanja presretača.*

Nakon kreiranih datoteka, i modifikacija pojedinih, na način prikazan u prethodnom izlaganju sve je spremno za upotrebu presretača u Java veb aplikaciji koja sve vreme služi kao podrška analizi i diskusiji u ovoj lekciji. Aplikacija će biti testirana za unos i prikazivanje podataka za premijum potrošača, a zatim će se obratiti pažnja na monitor *GlassFish* aplikativnog servera koji će priložiti dodatne informacije koje su zahtevane izvršavanjem metode *savePotrosac()* kontrolera *PremiumPotrosacController*.

Pomoću linka [http://localhost:8080/CdiDemo/faces/premium\\_potrosac.xhtml](http://localhost:8080/CdiDemo/faces/premium_potrosac.xhtml) odlazi se na stranicu za unos podataka o premijum potrošaču (videti sliku 7).



← ⓘ localhost:8080/CdiDemo/faces/premium\_potrosac.xhtml

### Kreiranje novog PREMIUM potrosaca

Ime	<input type="text" value="Vladimir"/>
Prezime	<input type="text" value="Milicevic"/>
Email Adresa	<input type="text" value="vladam.kg@gmail.com"/>
Kod popusta	<input type="text" value="1"/>
<input type="button" value="Potvrdi"/>	

Slika 4.7 Unos podataka o premijum potrošaču [izvor: autor]

Nakon unetih podataka, pomoću forme stranice sa Slike 7, vrši se klik na dugme *Potvrdi* i pokupljeni podaci sa forme se šalju na prikazivanje stranicom *premium\_potrosac.xhtml*. Rezultat izvršavanja ove stranice je prikazan sledećom slikom.



Ime	Vladimir
Prezime	Milicevic
Email Adresa	vladam.kg@gmail.com
Kod popusta	1

Slika 4.8 Podaci o generisanom premijum potrošaču [izvor: autor]

## TIPOVI POVEZIVANJA PRESRETAČA - DEMONSTRACIJA - DODATNA RAZMATRANJA

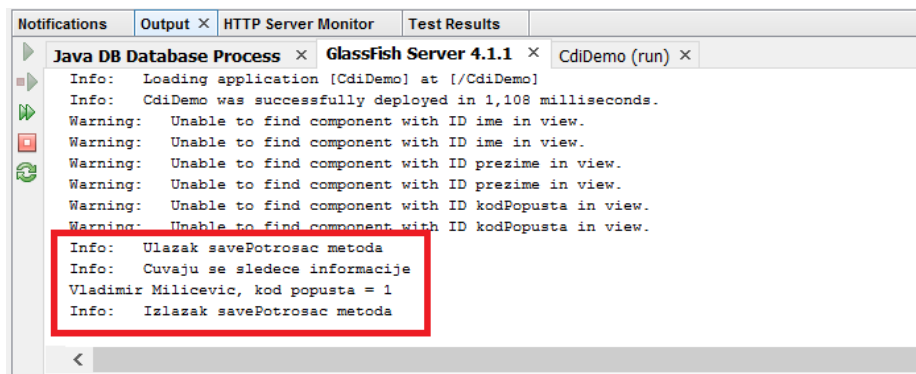
*Za kraj demonstracije je neophodno pozabaviti se metodom `savePotrosac()` kontrolera.*

Za kraj demonstracije je neophodno pozabaviti se metodom `savePotrosac()` kontrolera `PremiumPotrosacController`. Metoda poseduje **Log** instrukcije i prilaže dodatne informacije u monitoru `GlassFish` aplikativnog servera koje je moguće pročitati primenom razvojnog okruženja `NetBeans IDE`.

Sledećim listingom je izolovan kod metode `savePotrosac()` kontrolera `PremiumPotrosacController` koji je modifikovan u skladu sa diskusijom o primeni *tipova povezivanja presretnika*.

```
public String savePotrosac() {
    PremiumPotrosac premiumPotrosac
        = (PremiumPotrosac) potrosac;
    logger.log(Level.INFO, "Cuvaju se sledece informacije \n"
        + "{0} {1}, kod popusta = {2}",
        new Object[]{premiumPotrosac.getIme(),
            premiumPotrosac.getPrezime(),
            premiumPotrosac.getKodPopusta()});
    //If this was a real application, we would have code to save
    //customer data to the database here.
    return "premium_potrosac_potvrda";
}
```

Nakon unetih podataka, pomoću forme stranice sa Slike 7, vrši se klik na dugme *Potvrđi* i prikupljeni podaci sa forme se šalju na prikazivanje stranicom `premium_potrosac.xhtml` (videti Sliku 8), a **Log** informacije se šalju u monitor aplikativnog servera `GlassFish` razvojnog okruženja `NetBeans IDE` odakle mogu da budu pročitane. Rezultat izvršavanja ove stranice je prikazan sledećom slikom.



Slika 4.9 Informacije u GlassFish server monitoru [izvor: autor]

Ono što je moguće primetiti sa slike jeste da su linije ulaska i izlaska iz metode `savePotrosac()` dodate u log presretnačem, koje je indirektno pozvana tipom povezivanja presretnača.

Lekcija nastavlja analizu fokusirajući se na mogućnost kreiranja vlastitih (custom) oblasti `CDI` `zrna`.

## TIPOVI POVEZIVANJA PRESRETAČA - VIDEO MATERIJAL

*Diskusija o tipovima povezivanja presretnača biće zaokružena odgovarajućim video materijalom.*

Using Java EE Interceptors

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 5

# Vlastite CDI oblasti

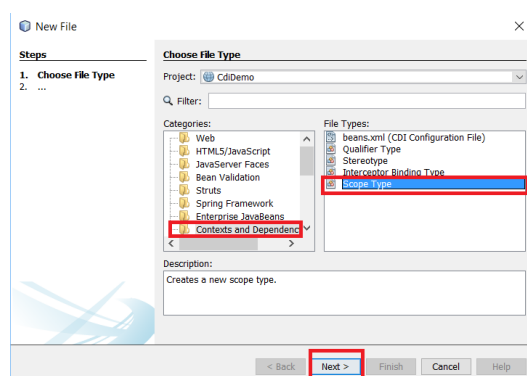
## KREIRANJE VLASTITIH CDI OBLASTI

*Savremena razvojna okruženja dozvoljavaju programerima kreiranje vlastitih CDI oblasti.*

Ono što je moguće primetiti iz svih prethodnih izlaganja jeste da najnovija verzija [Java EE](#) platforme, u kombinaciji sa savremenim razvojnim okruženjima, a jedno od njih je i [NetBeans IDE](#), pokazuje visok stepen fleksibilnosti kada je reč o razvoju savremenih veb i distribuiranih softverskih rešenja. To se posebno ogleda u visokom stepenu automatizacije procesa kreiranja programskog koda kroz korišćenje veoma pogodnih čarobnjaka ([wizards](#)) savremenih razvojnih okruženja.

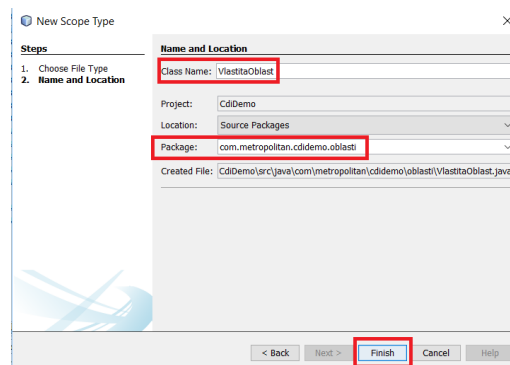
Na sličan način moguće je koristiti i upravljati [oblastima CDI zrna](#) ([CDI Scopes](#)). Pored već ugrađenih i podržanih [oblasti CDI zrna](#), savremena razvojna Java okruženja, kao direktna podrška najnovijoj generaciji [Java EE platforme](#), dozvoljavaju kreiranje i vlastitih ([custom](#)), korisnički prilagođenih, [oblasti CDI zrna](#). Ova funkcionalnost je, pre svega, namenjena programerima koji kreiraju okvir ([framework](#)) nad CDI zrnima, a manje aplikativnim programerima.

Zbog svega navedenog, razvojno okruženje [NetBeans IDE](#), kao jedno savremeno i moćno [Java](#) razvojno okruženje, obezbeđuje čarobnjaka ([wizard](#)) za kreiranje vlastitih ([custom](#)), korisnički prilagođenih, [oblasti CDI zrna](#). Proces započinje kao i u svim ostalim slučajevima kreiranja nove datoteke u Java projektima. Bira se opcija [File | New File](#). Nakon obavljenog navedenog izbora otvara se prozor pod nazivom [New File](#). U ovom prozoru iz kategorije [Contexts and Dependency Injection](#), bira se tip datoteke [Scope Type](#). Navedena aktivnost je ilustrovana sledećom slikom.



Slika 5.1 Početak kreiranja tipa datoteke Scope Type [izvor: autor]

Klikom na dugme *Next*, u prozoru sa Slike 1, otvara se nov prozor u kojem se, nakon izbor naziva datoteke i odgovarajućeg paketa, klikom na dugme Finish završavaju osnovna podešavanja ove datoteke (videti Sliku 2).



Slika 5.2 Izbor naziva datoteke oblasti zrna i paketa [izvor: autor]

## KOD DATOTEKE VLASTITE CDI OBLASTI

*U nastavku je neophodno prikazati kod koji je rezultat aktivnosti izvedenih u čarobnjaku.*

Iz prethodnog izlaganja moguće je primetiti kako savremeno razvojno okruženje *NetBeans IDE* nudi veoma elegantne čarobnjake za olakšavanje brojnih razvojnih zadataka nad *Java EE* platformom. Tako je i u slučaju kreiranja datoteka sa vlastitim CDI oblastima.

Ako se vrati pažnja na prethodnu sliku moguće je primetiti da je, nakon dodele naziva datoteci i izbora paketa kojem će ona pripadati, potrebno obaviti samo jedan zadatak, a to je klik na dugme *Finish*, nakon čega se generiše kod ove datoteke. Za naziv datoteke je izabran string *VlastitaOblast*, a za paket *com.metropolitan.cdideмо.oblasti*.

Kada se sve obavljeno i objašnjeno, neophodno je priložiti generisani kod datoteke *VlastitaOblast.java*. Kod je priložen sledećim listingom. **Listing pokazuje novu anotaciju @VlastitaOblast koja može biti primenjena na metode, polja i tipove, primenjuje se tokom vremena izvršavanja i definiše oblast (Scope).**

```
package com.metropolitan.cdideмо.oblasti;

import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import java.lang.annotation.Inherited;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import javax.inject.Scope;

/**
 *
 * @author Vladimir Milicevic
 */
```

```
@Inherited
@Scope // or @javax.enterprise.context.NormalScope
@Retention(RUNTIME)
@Target({METHOD, FIELD, TYPE})
public @interface VlastitaOblast {
}
```

Savladavanjem ove problematike studenti su razumeli i naučili kako je moguće kreirati vlastite CDI oblasti. Ovde je važno napomenuti pre svega, da je ovaj vid razvoja namenjen, pre svega, programerima koji kreiraju okvir (framework) nad CDI zrnima, a manje aplikativnim programerima koji će ga veoma retko koristiti.

## VLASTITE CDI OBLASTI - VIDEO MATERIJAL

*Izlaganje o vlastitim CDI oblastima biće zaokruženo prilaganjem odgovarajućeg video materijala.*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 6

# Pokazni primer - Umetanje konteksta i zavisnosti

## POSTAVKA (45 MIN)

### *Rešavanje zadataka sa CDI aplikacijama.*

Kreirati Java EE veb aplikaciju na osnovu sledećih zahteva:

1. Kreira se Java veb aplikacija za jednostavno registrovanje potrošača u veb prodavnici;
2. Na klijent strani se nalaze četiri JSF stranice: dve za unos podataka o potrošačima i dve za prikazivanje podataka koji su uneti;
3. Potrošač može da bude standardni i premijum ( veća lojalnost prilikom kupovine i veće pogodnosti);
4. Za svakog od navedenih tipova potrošača, kreirati klasu CDI zrna zajedno sa odgovarajućim kontrolerom;
5. Premijum potrošač je potklasa standardnog - koristiti kvalifikatore za kreiranje premijum potrošača;
6. Kreirana zrna su obeležena anotacijama @Named i @RequestScoped. Kreirati stereotipnu anotaciju koja će zameniti navedene anotacije;
7. Kreirati presretač i povezati ga sa jednim zrnom (na primer zrnom premijum presretača);

### REŠENJE:

Za početak je neophodno da se otvori razvojno okruženje NetBeans i da se kreira nov projekat pod nazivom Vezba9.

Prvo će biti kreirana front - end strana. Ovu stranu čine četiri datoteke sa ekstenzijom .xhtml. Dve datoteke sadrže i forme za unos podataka, dok preostale dve služe da se uneti podaci prikažu.

Sledi listing početne stranice pod nazivom index.xhtml.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Kreiranje novog potrosaca</title>
  </h:head>
```



```
<h:body>
  <h:form>
    <h3>Kreiranje novog potrosaca</h3>
    <h:panelGrid columns="3">
      <h:outputLabel for="ime" value="Ime"/>
      <h:inputText id="firstName" value="#{potrosac.ime}"/>
      <h:message for="ime"/>
      <h:outputLabel for="prezime" value="Prezime"/>
      <h:inputText id="lastName" value="#{potrosac.
                                prezime}"/>
      <h:message for="prezime"/>
      <h:outputLabel for="email" value="Email Adresa"/>
      <h:inputText id="email" value="#{potrosac.email}"/>
      <h:message for="email"/>
    </h:panelGrid>
    <h:panelGroup/>
    <h:commandButton value="Potvrđi"
                      action="#{potrosacController.
                                navigateToConfirmation}"/>
  </h:form>
</h:body>
</html>
```

## KREIRANJE .XHTML STRANICA

*Sledi prilaganje koda stranica koje su vidljive korisniku aplikacije.*

Podaci uneti na stranici index.xhtml, vidljivi su na stranici potvrda.xhtml. Sledi listing ove stranice.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Uspesno!!!</title>
  </h:head>
  <h:body>
    Novi potrosac je uspesno kreiran.
    <h:panelGrid columns="2" border="1" cellspacing="0">
      <h:outputLabel for="ime" value="Ime"/>
      <h:outputText id="ime" value="#{potrosac.
                                ime}"/>
      <h:outputLabel for="prezime" value="Prezime"/>
      <h:outputText id="prezime" value="#{potrosac.prezime}"/>
      <h:outputLabel for="email" value="Email Adresa"/>
      <h:outputText id="email" value="#{potrosac.email}"/>
    </h:panelGrid>
```

```
</h:body>
</html>
```

U veb prodavnici postoji i objekat premijum potrošač, pod objekat standardnog potrošača. Stranica za unos njegovih podataka se naziva premium\_potrosac.xhtml i sledi njen listing.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Kreiranje novog PREMIUM potrosaca</title>
  </h:head>
  <h:body>
    <h:form>
      <h3>Kreiranje novog PREMIUM potrosaca</h3>
      <h:panelGrid columns="3">
        <h:outputLabel for="ime" value="Ime"/>
        <h:inputText id="ime"
                     value="#{premiumPotrosac.ime}"/>
        <h:message for="ime"/>
        <h:outputLabel for="prezime" value="Prezime"/>
        <h:inputText id="prezime"
                     value="#{premiumPotrosac.prezime}"/>
        <h:message for="prezime"/>
        <h:outputLabel for="email" value="Email Adresa"/>
        <h:inputText id="email"
                     value="#{premiumPotrosac.email}"/>
        <h:message for="email"/>
        <h:outputLabel for="kodPopusta" value="Kod popusta"/>
        <h:inputText id="discountCode"
                     value="#{premiumPotrosac.kodPopusta}"/>
        <h:message for="kodPopusta"/>
        <h:panelGroup/>
        <h:commandButton value="Potvrđi"
                         action="#{premiumPotrosacController.
                               savePotrosac}"/>
      </h:panelGrid>
    </h:form>
  </h:body>
</html>
```

Podaci uneti na ovoj stranici, vidljivi su na stranici premium\_potrosac\_potvrda.xhtml čiji listing sledi:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Uspesno!!!</title>
  </h:head>
  <h:body>
```

```

Novi PREMIUM potrosac je uspesno generisan
<h:panelGrid columns="2" border="1" cellspacing="0">
  <h:outputLabel for="ime" value="Ime"/>
  <h:outputText id="ime" value="#{premiumPotrosac.ime}"/>

  <h:outputLabel for="prezime" value="Prezime"/>
  <h:outputText id="prezime" value="#{premiumPotrosac.prezime}"/>

  <h:outputLabel for="email" value="Email Adresa"/>
  <h:outputText id="email" value="#{premiumPotrosac.email}"/>

  <h:outputLabel for="kodPopusta" value="Kod popusta"/>
  <h:outputText id="kodPopusta" value="#{premiumPotrosac.kodPopusta}"/>

</h:panelGrid>
</h:body>
</html>

```

## KREIRANJE STEREOTIPA

*Kreira se anotacija koja će zameniti u kodu pojavljivanje većeg broja drugih anotacija.*

Pre nego što se kreiraju zrna i kontroleri, neophodno je kreirati i stereotipnu anotaciju, na osnovu uslova zadatka, koja će zameniti pojavljivanje dve anotacije, u odgovarajućim klasama, a to su @Named i @RequestScoped. U razvojnom okruženju, za kreirani projekat, kreirati nov paket i datoteku za stereotipnu anotaciju @ObelezenaOblastZrna kao u kodu koji je priložen ispod:

```

package com.metropolitan.cdideo.stereotip;

import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import javax.enterprise.context.RequestScoped;
import javax.enterprise.inject.Stereotype;
import javax.inject.Named;

/**
 *
 * @author Vladimir Milicevic
 */
@Named
@RequestScoped
@Stereotype
@Retention(RUNTIME)
@Target({METHOD, FIELD, TYPE})

```

```
public @interface ObelezenaOblastZrna {  
}
```

## IMPLEMENTACIJA PRESRETAČA

*Po uslovima zadatka neophodno je implementirati presretač.*

U ovom delu zadatka kreiraju se paketi i datoteke za presretača i odgovarajuću anotaciju presretača. Sledi listing datoteke kojom presretač dobija odgovarajuću anotaciju tj, tip za povezivanje presretača:

```
package com.metropolitan.cdideo.povezivanjepresretaca;  
  
import static java.lang.annotation.ElementType.TYPE;  
import static java.lang.annotation.ElementType.METHOD;  
import static java.lang.annotation.RetentionPolicy.RUNTIME;  
import java.lang.annotation.Inherited;  
import java.lang.annotation.Retention;  
import java.lang.annotation.Target;  
import javax.interceptor.InterceptorBinding;  
  
/**  
 *  
 * @author Vladimir Milicevic  
 */  
@Inherited  
@InterceptorBinding  
@Retention(RUNTIME)  
@Target({METHOD, TYPE})  
public @interface PovezivanjeLoggingPresretaca {  
}
```

Sledi kod presretača na koji je implementirana prethodno kreirana anotacija:

```
package com.metropolitan.cdideo.presretac;  
  
import com.metropolitan.cdideo.povezivanjepresretaca.PovezivanjeLoggingPresretaca;  
import java.io.Serializable;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import javax.interceptor.AroundInvoke;  
import javax.interceptor.Interceptor;  
import javax.interceptor.InvocationContext;  
  
/**  
 *  
 * @author Vladimir Milicevic  
 */  
@PovezivanjeLoggingPresretaca  
@Interceptor
```

```
public class LoggingPresretac implements Serializable {

    private static final Logger logger = Logger.getLogger(
        LoggingPresretac.class.getName());

    @AroundInvoke
    public Object logMethodCall(InvocationContext invocationContext)
        throws Exception {
        logger.log(Level.INFO, new StringBuilder("Ulazak ").append(
            invocationContext.getMethod().getName()).append(
                " metoda").toString());
        Object retVal = invocationContext.proceed();
        logger.log(Level.INFO, new StringBuilder("Izlazak ").append(
            invocationContext.getMethod().getName()).append(
                " metoda").toString());
        return retVal;
    }
}
```

## KREIRANJE CDI ZRNA

*Za svaki tip potrošača kreira se odgovarajuća klasa.*

Sledi kod klase (CDI zrna) za implementaciju standardnog potrošača. Na njega je primenjen kreirani stereotip.

```
package com.metropolitan.cdideo;

import java.io.Serializable;
import javax.enterprise.context.RequestScoped;
import javax.inject.Named;
import com.metropolitan.cdideo.stereotip.ObelezenaOblastZrna;

/**
 *
 * @author Vladimir Milicevic
 */
@ObelezenaOblastZrna
public class Potrosac implements Serializable {

    private String ime;
    private String prezime;
    private String email;

    public Potrosac() {
    }

    public String getIme() {
        return ime;
    }
}
```

```
}

public void setIme(String ime) {
    this.ime = ime;
}

public String getPrezime() {
    return prezime;
}

public void setPrezime(String prezime) {
    this.prezime = prezime;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}
}
```

Sledi kod klase (CDI zrna) za implementaciju premijum potrošača koji je potklasa prethodno kreirane klase. Na njega je primenjen kreirani stereotip.

```
package com.metropolitan.cdideo;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

import com.metropolitan.cdideo.kvalifikator.Premium;
import javax.enterprise.context.RequestScoped;
import javax.inject.Named;
import com.metropolitan.cdideo.Potrosac;
import com.metropolitan.cdideo.stereotip.ObelezenaOblastZrna;

/**
 *
 * @author Vladimir Milicevic
 */
@ObelezenaOblastZrna
@Premium
public class PremiumPotrosac extends Potrosac {

    private Integer kodPopusta;

    public Integer getKodPopusta() {
        return kodPopusta;
    }
}
```

```

    }

    public void setKodPopusta(Integer kodPopusta) {
        this.kodPopusta = kodPopusta;
    }
}

```

## KREIRANJE KVALIFIKATORA @PREMIUM

*Kvalifikator se primenjuje na prethodno kreiranu potklasu.*

Po uslovu zadatka, na poklasu PremiumPotrosac, klase Potrosac, primenjen je kvalifikator @Premium. Sledi njegov listing. Datoteka se kreira u posebnom paketu i sa nazivom koji je istaknut u listingu.

```

package com.metropolitan.cdideo.kvalifikator;

import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.PARAMETER;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import javax.inject.Qualifier;

/**
 *
 * @author Vladimir Milicevic
 */
@Qualifier
@Retention(RUNTIME)
@Target({METHOD, FIELD, PARAMETER, TYPE})
public @interface Premium {
}

```

## KREIRANJE KONTROLERA CDI ZRNA

*Primer se završava kreiranjem kontrolera za kreirana CDI zrna.*

Sledi listing koda koji se odnosi na datoteku PotrosacController.java:

```

package com.metropolitan.cdideo.controller;

import com.metropolitan.cdideo.Potrosac;
import javax.enterprise.context.RequestScoped;
import javax.inject.Inject;
import javax.inject.Named;

```

```
import com.metropolitan.cdideo.stereotip.ObelezenaOblastZrna;

/**
 *
 * @author Vladimir Milicevic
 */
@ObelezenaOblastZrna
public class PotrosacController {

    @Inject
    private Potrosac potrosac;

    public Potrosac getPotrosac() {
        return potrosac;
    }

    public void setPotrosac(Potrosac potrosac) {
        this.potrosac = potrosac;
    }

    public String navigateToConfirmation() {
//In a real application we would
//Save customer data to the database here.
        return "potvrda";
    }
}
```

Sledi listing koda koji se odnosi na datoteku PremiumPotrosacController.java:

```
package com.metropolitan.cdideo.controller;

import com.metropolitan.cdideo.povezivanjepresretaca.PovezivanjeLoggingPresretaca;
import com.metropolitan.cdideo.Potrosac;
import com.metropolitan.cdideo.kvalifikator.Premium;
import com.metropolitan.cdideo.PremiumPotrosac;

import java.util.logging.Level;
import java.util.logging.Logger;
import javax.enterprise.context.RequestScoped;
import javax.inject.Inject;
import javax.inject.Named;
import com.metropolitan.cdideo.stereotip.ObelezenaOblastZrna;

/**
 *
 * @author Vladimir Milicevic
 */
@PovezivanjeLoggingPresretaca
@ObelezenaOblastZrna
public class PremiumPotrosacController {

    private static final Logger logger = Logger.getLogger(
        PremiumPotrosacController.class.getName());
}
```



```
@Inject
@Premium
private Potrosac potrosac;

public String savePotrosac() {
    PremiumPotrosac premiumPotrosac
        = (PremiumPotrosac) potrosac;
    logger.log(Level.INFO, "Cuvaju se sledece informacije \n"
        + "{0} {1}, kod popusta = {2}",
        new Object[]{premiumPotrosac.getIme(),
            premiumPotrosac.getPrezime(),
            premiumPotrosac.getKodPopusta()});
    //If this was a real application, we would have code to save
    //customer data to the database here.
    return "premium_potrosac_potvrda";
}
}
```

## ▼ Poglavlje 7

### Individualne vežbe 9

#### INDIVIDUALNA VEŽBA (135 MIN)

*Pokušajte sami - kreirajte nove funkcionalnosti proširivanjem pokaznog primera.*

1. Kreirajte novu početnu stranicu na kojoj ćete birati između unosa podataka standardnog ili premijum potrošača;
2. Realizujte JSF stranice aplikacije primenom neke od poznatih JSF biblioteka komponenata;
3. Ostatak zadatka neka se bazira na zadatku urađenom na vežbama;
4. U slučaju nedoumice, konsultovati redom teme obrađene na predavanjima;

#### NAPOMENA:

1. Korisnik na početnoj stranici bira između dve opcije: standardni ili premijum potrošač.
2. Stranica je stilizovana nekom JSF bibliotekom komponenata (kao i sve ostale stranice aplikacije).
3. U slučaju izbora standardnog potrošača, navigacija teče ka stranicama i kontrolerima koji prihvataju i obrađuju informacije vezane za ovaj tip potrošača.
4. U suprotnom, navigacija teče ka stranicama i kontrolerima koji prihvataju i obrađuju informacije vezane za tip potrošača premijum.

Dokumentujte urađenu individualnu vežbu i kontaktirajte predmetnog asistenta.

## ▼ Poglavlje 8

### Domaći zadatak 9

#### ZADATAK 1 (120 MIN)

*Uradite domaći zadatak.*

Za izradu domaćeg zadatka koristiti kao uzor zadatak urađen na predavanjima i vežbama.

Kreirati Java EE veb aplikaciju na osnovu sledećih zahteva:

1. Kreira se Java veb aplikacija za jednostavno registrovanje studenata u informacionom sistemu fakulteta;
2. Na klijent strani se nalaze četiri JSF stranice: dve za unos podataka o studentima i dve za prikazivanje podataka koji su uneti;
3. Student može da bude tradicionalni i Internet ( Internet student nasleđuje tradicionalnog i ima dodatne osobine - osmisлити);
4. Za svakog od navedenih tipova studenata, kreirati klasu CDI zrna zajedno sa odgovarajućim kontrolerom;
5. Internet student je potklasa tradicionalnog - koristiti kvalifikatore za kreiranje Internet studenta;
6. Kreirana zrna su obeležena anotacijama @Named i @RequestScoped. Kreirati stereotipnu anotaciju koja će zameniti navedene anotacije;
7. Kreirati presretač i povezati ga sa jednim zrnom (na primer zrnom Internet studenta);
8. Dodati novu, petu JSF stranicu, koja ima zadatak da obezbedi korisniku izbor između unosa podataka za tradicionalnog ili Internet studenta (ovo će biti index.xhtml);
9. JSF stranicama dati lepši izgled primenom neke od poznatih JSF biblioteka komponenata;
10. Svaka od kreiranih JSF stranica ima kao pozadinu logo našeg Univerziteta;
11. Testirati aplikaciju i priložiti uz kod rešenja, slike koje pokazuju funkcionalnost kreirane aplikacije.

Nakon urađenog obaveznog zadatka, studenti dobijaju različite zadatke od predmetnog asistenta.

## ▼ Poglavlje 9

# Zaključak

## ZAKLJUČAK

*Lekcija se bavila temama iz oblasti umetanja konteksta i zavisnosti u Java EE aplikacijama.*

Lekcija se bavila temama iz oblasti umetanja konteksta i zavisnosti (**Contexts and Dependency Injection** - CDI). Pri tome je jasno navedeno da CDI može biti upotrebljeno za pojednostavljivanje integracije različitih nivoa Java EE aplikacije. Zo praktično znači da CDI dozvoljava da se *zrno sesije* koristi kao *upravljano zrno*. Na taj način se može iskoristiti puna prednost primene EJB alata, poput transakcija, direktno u posmatranom upravljanom zrnu.

Lekcija je posebnu pažnju stavila na bavljenje analizom i diskusijom najsavremenijih tema iz ove oblasti, među kojima su se istakle sledeće:

- **Uvod u CDI** - poseban akcenat je bio na CDI API, povezivanju JSF komponenata i CDI zrna, oblastima zrna, zrnima kontrolera i tako dalje;
- **Kvalifikatori** - u ovom delu lekcije akcenat je bio na predstavljanju kvalifikatora, njihovom kreiranju i primeni;
- **Stereotipovi** -u ovom delu lekcije akcenat je bio na predstavljanju i kreiranju stereotipova, implementaciji stereotipova na CDI zrna;
- **Tipovi povezivanja presretača** - akcenat je, dalje, bio na predstavljanju i kreiranju tipova povezivanja presretača, njihovom kodiranju, dodatnim podešavanjima kroz beans.xml datoteku, kreiranju odgovarajućih klasa kontrolera i primeni;
- **Vlastite CDI oblasti.** - konačno, akcenat je bio na predstavljanju i kreiranju vlastitih CDI oblasti kao datoteka koje pišu i koriste framework programeri, više nego aplikativni programeri.

Oslanjajući se na navedene teme, sadržaj lekcije je veoma pažljivo biran sa jakim akcentom na konkretne praktične primere koji su bili potpora analizi i diskusiji vezanim za ovu lekciju. Takođe, pažljivo je biran i sadržaj vežbi, pri čemu su izabrani zadaci u potpunosti zaokružili izlaganje o CDI konceptima i tehnikama.

Savladavanjem ove lekcije student je u potpunosti razumeo i u stanju je da koristi CDI koncepte i tehnike u Java EE aplikacijama. Nakon ove lekcije student je stekao nova znanja i veštine u radu sa naprednim Java konceptima i principima, kao i naprednim Java EE veb aplikacijama.

## ZAKLJUČAK - VIDEO MATERIJAL

*CDI : or how to extend Java EE.*

Izlaganje o konceptu Context and Dependency Injection, biće zaokružen jednim obimnim video materijalom.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## LITERATURA

*Za pripremu lekcije korišćena je najnovija literatura.*

1. Eric Jendrock, Ricardo Cervera-Navarro, Ian Evans, Kim Haase, William Markito. 2014. Java Platform, Enterprise Edition The Java EE Tutorial, Release 7, ORACLE
2. David R. Heffelfinger. 2015. Java EE7 Development With NetBeans 8, PACK Publishing
3. Yakov Fain. 2015. Java 8 programiranje, Kombib (Wiley)
4. Josh J. Ueneau. 2015. Java EE7 Recipes, Apress