Bindings

Nakon prethodne lekcije u kojoj su iznete osnovne osobine Java FX propertija, lekcija koja sledi biće posvećena pojmu bindinga. Naime, Java FX propertiji su specijalni propertiji koji imaju mogućnost da emituju događaje o svom unutrašnjem stanju, što je prikazano u lekciji za nama. Druga bitna osobina Java FX propertija je mogućnost bindinga.

Šta je binding?

U programiranju pojam binding se veoma često može čuti u različitim oblastima i kontekstima. Jednostavno rečeno, binding definiše relaciju između elemenata koji nose određene podatke tako što vrši njihovu sinhronizaciju. Ovo je naročito korisno kod razvoja aplikacija sa grafičkim korisničkim okruženjem, gde se binding veoma često koristi za sinhronizovanje objekata modela i kontrola UI-ja.

Binding je najlakše razumeti na primeru jednog matematičkog izraza. Kao izraz dat je:

```
x = y + z;
```

Jasno je da će se prilikom promene vrednosti y ili z i vrednost x neminovno promeniti. Stoga se može zaključiti da u prikazanom primeru postoji binding između x i zbira y i z. Svaki put kada dođe do promena nekog od sabiraka, ta promena se mora adekvatno preslikati i na x. Sličan ovome je i način funkcionisanja Java FX bindinga.

Kako Java FX binding funkcioniše?

U prethodnom primeru za y i z se može reći da su zavisnosti, a da je x bindovan za sumu ove dve vrednosti.

Kod Java FX-a, uspešno funkcionisanje bindinga se oslanja na jedan od eventa opisan u prethodnoj lekciji. Reč je o *invalidation* eventu. Java FX binding zapravo dodaje slušače za ovaj događaj na sve zavisnosti. Kada se vrednost bilo koje od zavisnosti promeni, binding postaje nevalidan. Ovo je znak da se pri sledećoj situaciji u kojoj se zatraži vrednost povezane (bindovane) strane mora obaviti njena rekalkulacija pre konačnog isporučivanja. Iz ovoga se može zaključiti da Java FX binding koristi odloženo kalkulisanje vrednosti propertija.

Strogo matematički primer sa početka ove lekcije sada će biti pretočen u specifičan Java FX kod korišćenjem bindinga.

```
package main;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;

public class Main {

    public static void main(String[] args) {

        IntegerProperty x = new SimpleIntegerProperty(10);
        IntegerProperty y = new SimpleIntegerProperty(20);
        IntegerProperty z = new SimpleIntegerProperty(60);

        z.bind(x.add(y));

        System.out.println("Initial value of z = " + z.get());

        System.out.println("changing dependencies...");
        x.set(15);
        y.set(19);
        System.out.println("After changing x and y, z = " + z.get());

        System.out.println("unbinding...");
        z.unbind();

        System.out.println("changing dependencies...");
        x.set(100);
        y.set(200);
        System.out.println("After unbinding z " + z.get());

    }

}
```

Sa prve tri linije koda unutar main metode izvršeno je instanciranje Java FX propertija. Metodom bind izvršeno je bindovanje vrednosti promenljive z i sume vrednosti x i y.

Napomena

U prikazanom primeru neophodno je koristiti metodu add, s obzirom na to da se sabiraju vrednosti objektnih promenljivih.

Prvom linijom za ispisivanje vrednosti u konzolu prikazuje se inicijalna vrednost promenljive z koja je 30, što je zbir vrednosti 10 i 20.

Nakon ovoga, vrednosti x i y se menjaju, i to na 15 i 19, respektivno. Nakon ovih promena, vrednost promenljive z se opet ispisuje u konzoli, gde se može zaključiti da je ta vrednost adekvatno pratila promenu vrednosti x i y.

Sledeći korak jeste raskidanje binding povezanosti i to metodom unbind, nad instancom Java FX propertija z. Nakon raskida binding ugovora, sinhronizacija vrednosti se više neće odvijati, što se može videti u poslednjoj promeni vrednosti x i y prikazanoj na kraju primera. Nakon pomenute promene, z ostaje na staroj vrednosti, odnosno ne prati promenu vrednosti zbira x i y.

Unidirectional i Bidirectional Binding

Binding može imati smer. On se odnosi na smer propagiranja promena. Java FX podržava dva tipa bindinga u zavisnosti od smera propagiranja:

- unidirectional binding i
- bidirectional binding.

Unidirectional binding funkcioniše samo u jednom smeru, kao što je to bio slučaj u prethodnom primeru. Promene na zavisnostima se propagiraju na povezani property, ali ne i obrnuto.

Bidirectional binding funkcioniše u oba smera, odnosno promene na zavisnostima se preslikavaju na povezani property, ali i obrnuto.

Neke od glavnih osobina unidirectional bindinga su sledeće:

- povezanost se postiže metodom bind,
- moguće je povezati property i kompleksni izraz,
- nije moguće promeniti samostalno vrednost povezanog propertija; takav pokušaj će dovesti do izbacivanja Runtime Exceptiona,
- ukoliko je potrebno promeniti vrednost povezanog propertija, neophodno je raskinuti povezanosti, pa tek onda samostalno promeniti vrednost,
- jedan property može imati definisan samo jedan unidirectional binding i
- definisanje povezanosti (binding) za property koji već ima definisan binding raskida prethodnu povezanost.

Neke od glavnih osobina bidirectional bindinga su sledeće:

- povezanost se postiže metodom bindBidirectional(),
- moguće je povezati samo property i property, odnosno nije moguće povezati property i izraz; ukoliko se property promeni, ne može se utvrditi na koji način bi se adekvatno promenio izraz, stoga ova vrsta dvosmerne povezanosti nije podržana,
- moguće je ovakvu povezanost kreirati samo između propertija istog tipa,
- vrednosti povezanih propertija mogu biti ručno promenjene,
- moguće je postavljati više ovakvih bindinga na jedan property i
- uklanja se metodom unbindBidirectional().

Sledeći primer ilustruje upotrebu bidirectional bindinga.

```
package main;

      import javafx.beans.property.IntegerProperty;
      import javafx.beans.property.SimpleIntegerProperty;

      public class Main {

          public static void main(String[] args) {

              IntegerProperty x = new SimpleIntegerProperty(1);
              IntegerProperty y = new SimpleIntegerProperty(2);
```

```
        IntegerProperty z = new SimpleIntegerProperty(3);

        System.out.println("Before binding:");
        System.out.println("x=" + x.get() + ", y=" + y.get() + ", z="
+ z.get());

        x.bindBidirectional(y);
        System.out.println("After binding x to y:");
        System.out.println("x=" + x.get() + ", y=" + y.get() + ", z="
+ z.get());

        x.bindBidirectional(z);
        System.out.println("After binding x to z:");
        System.out.println("x=" + x.get() + ", y=" + y.get() + ", z="
+ z.get());

        System.out.println("After changing z:");
        z.set(19);
        System.out.println("x=" + x.get() + ", y=" + y.get() + ", z="
+ z.get());

        x.unbindBidirectional(y);
        x.unbindBidirectional(z);
        System.out.println("After unbinding and changing them
separately:");
        x.set(100);
        y.set(200);
        z.set(300);
        System.out.println("x=" + x.get() + ", y=" + y.get() + ", z="
+ z.get());

    }

}
```

Prethodni primer ilustruje bidirectional binding između tri propertija. Property x je povezan sa y, ali i sa z. U prikazanom primeru moguće je steći uvid u tok promene vrednosti, postepeno.

Prvo se vrši povezivanje propertija x sa y. S obzirom na to da svi propetiji imaju početne vrednosti (1, 2, 3), x dobija vrednost 2, što je vrednost y propertija.

Sledeće povezivanje definiše povezanost između propertija x i z. Ovom povezanošću x dobija vrednost propertija z, što je 3. Pošto je reč o bidirectional bindingu, promenom vrednosti x promeniće se vrednost y, tako da će sva tri propertija imati istu vrednost, a to je 3.

Fluent API

U nastavku lekcije biće prikazane osnovne funkcionalnosti. Naziv Fluent API duguje mogućnosti nadovezivanja metoda (*method chaining*). Reč je o mogućnosti pisanja poziva metoda jednog za drugim. Ovu funkcionalnost je najbolje upoznati na primeru. Evo kako bi izgledala dva poziva metode add.

```
x.add(y);
x.add(z);
```

Korišćenjem Fluent API-ja upravo prikazani kod bi se mogao napisati na sledeći način:

```
x.add(y).add(z);
```

U dosadašnjem toku kursa ova metoda add je već korišćena nekoliko puta. Tada verovatno niste primetili da ova metoda ima povratnu vrednost Number Binding. Ovaj interfejs, baš kao Number Expression interfejs koji sadrži upravo pomenutu metodu add() deo su Fluent API-ja. Number Binding je tip koji predstavlja bilo koju numeričku bindovanu vrednost, dok interfejs Number Expression grupiše sve metode koje se koriste za kreiranje bindinga u Fluent API stilu.

Uzimajući u obzir sve izrečeno, kreiranje bindinga za sumu vrednosti dva Java FX propertija bi izgledalo ovako:

```
IntegerProperty x = new SimpleIntegerProperty(100);
IntegerProperty y = new SimpleIntegerProperty(200);

NumberBinding sum = x.add(y);
int value = sum.intValue();
```

U upravo prikazanom primeru za preuzimanje vrednosti iz instance tipa Number Binding koristi se metoda int Value. Pored ove metode, postoje i metode za preuzimanje vrednosti drugih numeričkih tipova. One su:

- double double Value()
- float float Value()
- int intValue()
- long longValue()

Kada bi se u prikazanom primeru koristili propertiji tipa Double Property, kao metoda za preuzimanje vrednosti koristila bi se double Value.

Mogući su i slučajevi sabiranja vrednosti Java FX propertija koji nisu istog tipa, kao što su na primer Double Property i Integer Property. U tom slučaju, najbolja varijanta jeste korišćenje metode double Value. Naravno, moguće je koristiti i metodu int Value, kada bi došlo do gubitka preciznosti i zaokruživanja vrednosti na prvu celobrojnu nižu vrednost.

```
DoubleProperty x = new SimpleDoubleProperty(200.65);
IntegerProperty y = new SimpleIntegerProperty(100);

    NumberBinding sum = x.add(y);
    int value = sum.intValue();

  System.out.println(value);
```

Prikazani kod proizvodi rezultat 300.

Pored ove metode add, Number Expression interfejs sadrži i mnoge druge srodne metode. One su prikazane u narednoj tabeli.

Metoda	Povratna vrednost	Opis
add()	Number Binding	Ovo su metode koje predstavljaju osnovne aritmetičke operacije i koje stvaraju nove instance tipa Number Binding.
subtract()		
multiply()		
divide()		
greaterThan()	Boolean Binding	Ova grupa metoda koristi se za poređenje vrednosti propertija, te zbog toga kao povratnu vrednost i imaju novu instancu tipa Boolean Binding.
greaterThanOrEqualTo()		
isEqualTo()		
isNotEqualTo()		
lessThan()		
lessThanOrEqualTo()		
negate()	Number Binding	Rezultuje negacijom.
asString()	String Binding	Ovo je metoda koja vrednost bindinga pretvara u String objekat.

Tabela 4.1

Iz prikazane tabele moguće je videti da pored bindinga tipa Number Binding, koji je prikazan u dosadašnjem toku lekcije postoje i drugi tipovi bindinga.

Sledeći primer ilustruje upotrebu bindinga za automatsko proračunavanje površine kruga u zavisnosti od radiusa.

```
package main;

import javafx.beans.binding.StringExpression;
import javafx.beans.property.DoubleProperty;
import javafx.beans.property.SimpleDoubleProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;

public class Main {

    public static void main(String[] args) {

      DoubleProperty radius = new SimpleDoubleProperty(9.0);
      DoubleProperty area = new SimpleDoubleProperty(0);
      StringProperty initStr = new SimpleStringProperty("Radius
     is ");

       area.bind(radius.multiply(radius).multiply(Math.PI));

      StringExpression desc = initStr.concat(radius.asString())
               .concat(", Area is ")
               .concat(area.asString("%.2f"));
      System.out.println(desc.getValue());

       radius.set(16.0);
       System.out.println(desc.getValue());

    }
}
```

U prikazanom primeru Java FX Double Property area je bindovan na izraz za proračunavanje površinu kruga. Dalje je iskorišćen String Expression tip za formiranje tekstualne poruke koja će biti emitovana korisniku, a sve to korišćenjem nadovezivanja metoda Fluent API-ja. Za nadovezivanje vrednosti propertija korišćena je metoda concat. Property init Str je tipa String Property tako da nema potrebe za njegovim pretvaranjem u string, kao što je to slučaj sa preostala dva propertija za čije se dobijanje tekstualne reprezentacije koristi metoda as String. Metoda as String može da primi parametar koji definiše format string vrednosti. U primeru je proračunata vrednost površine prilikom pretvaranja u tekstualnu reprezentaciju zaokružena na dve decimale.

Na kraju, binding je moguće koristiti i na korisnički definisanim tipovima. Ukoliko kao korisnički tip postoji klasa Person i dva njene instance, utvrđivanje jednakosti te dve instance i bindovanje dobijene vrednosti je moguće postići na sledeći način:

```
package main;

import javafx.beans.binding.BooleanBinding;
import javafx.beans.property.ObjectProperty;
import javafx.beans.property.SimpleObjectProperty;

public class Main {

    public static void main(String[] args) {

        Person p1 = new Person("John");
        Person p2 = new Person("Ben");
        ObjectProperty<Person> person1 = new
SimpleObjectProperty<>(p1);
        ObjectProperty<Person> person2 = new
SimpleObjectProperty<>(p2);

        BooleanBinding isEqual = person1.isEqualTo(person2);
        System.out.println(isEqual.get());
        person2.set(p1);
        System.out.println(isEqual.get());

    }

}
```

U prikazanom primeru instancirana su dva objekta tipa Person. Prilično je jasno da su to dve posebne instance, tako da objekti nisu jednaki. Nakon instanciranja objekata, instancirana su i dva Java FX propertija tipa Object Property i njima su kao vrednosti prosleđene instance kreiranih objekata.

Korišćenjem metode isEqualTo utvrđuje se jednakost Java FX propertija, a rezultat se smešta u instance tipa Boolean Binding. Vrednost ove instance se preuzima metodom get.

Prva linija koja prikazuje poruku u okviru konzole svakako će emitovati false na izlaz. Ali nakon ove linije za vrednosti drugog Java FX Object Propertyja postavljana je vrednost prvog, tako da sada ovi propertiji upućuju na istu referencu. Stoga, poslednja linija koda, koja prikazuje poruku u okviru konzole, nesumnjivo emituje vrednosti true.

Bindings

Priča o bindingu ne može da prođe bez pominjanja pomoćne klase Bindings koja omogućava definisanje jednostavnih bindinga, korišćenjem više od 150 statičkih metoda koje izlaže. Praktično, sve što je do sada urađeno korišćenjem Fluent API-ja može biti urađeno i korišćenjem Bindings klase.

Neke od metoda ove klase prikazane su u tabeli:

Metoda	Opis
add()	Obavljanje aritmetičke operacije i kreiranje bindinga.
subtract()	
multiple()	
divide()	
bindBidirectional()	Kreiranje i poništavanje dvosmernog bindinga između dva propertija.
unbindBidirectional()	
concat()	Nadovezivanje stringova i kreiranje bindinga.
createXXXBinding()	Kreiranje specifičnog bindinga u odnosu na tip, gde XXX predstavlja neki od tipova Boolean, Double, Float, Integer, String, ili Object.
equal()	Izračunavanje jednakosti i kreiranje bindinga tipa Boolean Binding.
notEqual()	
greaterThan()	
greaterThanOrEqual()	
lessThan()	
lessThanOrEqual()	
isNotNull	Poređenje vrednosti sa null, a zatim kreiranje Bindinga tipa Boolean Binding sa dobijenom vrednošću.
isNull	
max()	Određivanje maksimuma ili minimuma dva prosleđena argumenta, a zatim kreiranje bindinga.
min()	

Tabela 4.2

Posebno je zanimljiva upotreba metoda select XXX(), gde XXX predstavlja neki od specifičnih tipova Boolean, Double, Float, Integer, String, ili Object. Ova metoda se koristi za kreiranje bindinga ugneždenih propertija. Ugneždeni propertiji su oni koji se već nalaze unutar nekog specifičnog Java FX propetija.

Ovakva situacija se može dočarati sledećim primerom. Klasa Address je klasa koja sadrži podatke o adresi. Klasa Person može sadržati svojstvo tipa Address, koje bi simbolizovalo adresu osobe. Evo klase Address:

```
      public class Address {
      private StringProperty zip = new SimpleStringProperty("36106");
      public StringProperty zipProperty() {
      return zip;
      }
}
```

Klasa Person bi mogla da izgleda ovako:

```
public class Person {
private ObjectProperty<Address> addr = new
SimpleObjectProperty(new Address());
public ObjectProperty<Address> addrProperty() {
return addr;
}
}
```

Na nekom mestu u kodu, gde bi se javila potreba za upotrebom Person instance, properti bi mogao da izgleda ovako:

```
ObjectProperty<Person> p = new SimpleObjectProperty(new
Person());
```

Ali šta ako je potrebno neki specifični properti povezati sa propertijem koji simbolizuje adresu unutar ovakvog upravo kreiranog propertija? U takvim situacijama se koristi metoda select XXX.

```
StringBinding zipBinding = Bindings.selectString(p, "addr");
```

U prikazanoj liniji koda definisan je binding na ugneždeni properti unutar propetija čija je instanca p. Naziv propertija za koji je definisan binding je addr, a u definiciji klase Person se može videti da je to naziv propertija koji predstavlja adresu.

Moguće je čak otići i korak dalje i izvršiti binding na properti unutar addr propetija:

```
StringBinding zipBinding = Bindings.selectString(p, "addr",
"zip");
```

Sada je izvršen binding na property zip, unutar propertija addr, unutar instance propertija p.

Pitanje

Na koji event Java FX propertija se oslanja binding?

a) invalidation
b) change

Kod Java FX-a, uspešno funkcionisanje bindinga se oslanja na invalidation event. Java FX binding zapravo dodaje slušače za ovaj događaj na sve zavisnosti.

Rezime

- Binding definiše relaciju između elemenata koji nose određene podatke, tako što vrši njihovu sinhronizaciju.
- Binding se veoma često koristi za sinhronizovanje objekata modela i kontrola UI-ja, što je naročito korisno kod razvoja aplikacija sa grafičkim korisničkim okruženjem.
- Kod Java FX-a, uspešno funkcionisanje bindinga se oslanja na invalidation event.
- Java FX binding koristi odloženo kalkulisanje vrednosti propertija.
- Java FX podržava dva tipa binding u zavisnosti od smera propagiranja.
- Undirectional binding funkcioniše samo u jednom smeru.
- Bidirectional binding funkcioniše u oba smera, odnosno promene na zavisnostima se preslikavaju na povezani properti, ali i obrnuto.
- Fluent API sadrži brojne metode u nekoliko interfejsa i klasa koje umnogome olakšavaju binding u Java FX-u.