infoart
inovacija. znanje. tehnologija.

Bolji JPA uz najbolje prakse ADF-a

Vjekoslav-Leonard Prčić, mag. ing. rač.

JPA

- Java Persistence API
  - relacijske baze podataka
  - programsko sučelje
  - različite implementacije
  - Java Persistence Query Language (JPQL)

- Problemi
  - novi jezik za pristup bazi podataka
  - otežano pisanje složenih upita
    - JPQL radi samo nad entitetima
    - ne postoji “View” sloj

Entity

```
@Entity
@Table(schema = "HR", name = "REGIONS")
@NamedQueries({
 @NamedQuery(
  name = "Regions.findAll",
  query = "SELECT r FROM Regions r")
})
public class Regions implements Serializable {
 private static final long serialVersionUID = 1L;
 @Id
 @Basic(optional = false)
 @NotNull
 @Column(name = "REGION_ID")
 private BigDecimal regionId;
 @Size(max = 25)
 @Column(name = "REGION_NAME")
 private String regionName;

 public Regions() { }

 public BigDecimal getRegionId() {
  return regionId;
 }

 public void setRegionId(BigDecimal regionId) {
  this.regionId = regionId;
 }

 /* constructors, getters, setters,
 hashCode, equals, toString */
}
```

Tablica u bazi preslikava se u Java klasu
- *@Entity* – oznaka JPA entitet-a
- *@Table* – mapiranje na tablicu

Polja u tablici postaju privatna polja u klasi
- *@Column* – mapiranje na polje u tablici

Restrikcije nad poljima se preslikavaju u validacijske anotacije
- *@NotNull*, *@Size*, …

Nedostatci
- Ne postoji nadklasa koja bi sadržavala standardne metode
- Vrijednosti izložene direktno preko gettera/settera

Entity

```
@Entity
@Table(schema = "HR", name = "REGIONS")
@NamedQueries({
 @NamedQuery(
  name = "Regions.findAll",
  query = "SELECT r FROM Regions r")
})
public class Regions extends AbstractEntity<BigDecimal> {
 @Id
 @Basic(optional = false)
 @NotNull
 @Column(name = "REGION_ID")
 @Access(AccessType.PROPERTY)
 private BigDecimal regionId;
 @Size(max = 25)
 @Column(name = "REGION_NAME")
 @Access(AccessType.PROPERTY)
 private String regionName;

 public Regions() { }

 public BigDecimal getRegionId() {
  return (BigDecimal)getAttribute("regionId").getValue();
 }

 public void setRegionId(BigDecimal regionId) {
  getAttribute("regionId").setValue(regionId);
 }

 /* constructors, getters, setters */
}
```

Getter/*Setter* metode prolaze kroz „atribute”

- *@Access(AccessType.PROPERTY)*
- Pojedine klase atributa NE sadrže vrijednosti, ali ih mogu vratiti i postaviti
- Atributi sadrže inicijalne vrijednosti i *default*-ne vrijednosti
- Moguća kontrola postavljanja vrijednosti
- *Readonly* atributi

Klasa *AbstractEntity<T>* sadrži standardni kôd

- Sadrži sve atribute što omogućava implementaciju generičkih metoda
- *shallowEquals* – provjera podatkovne jednakosti
- *getDefault* – vraća novi entitet iste vrste, ali s *default*-nim vrijednostima atributa
- *getInitial* – vraća novi entitet iste vrste, ali s inicijalnim vrijednostima atributa
- Drži stanje cjelokupnog entiteta
- Zna vratiti *primary key* atribut

View

```
select
 {r},
 row_number() over (order by r.region_id) as region_rbn
from regions r
where r.name like nvl(:NamePrefix, '') || '%'
```

```
@View(row = RegionsViewRow.class)
public class RegionsView extends AbstractView<RegionsViewRow>
{

 @Variable("NamePrefix")
 private String namePrefix;

 public VariableAttribute<String> getNamePrefix() {
  return getVariable("namePrefix");
 }

 public static class RegionsViewRow extends AbstractViewRow {
  @EntityAttribute("R")
  private Regions region;

  @ValueAttribute("REGION_RBN")
  private BigDecimal regionRbn;

  public RegionsViewRow(RegionsView view) {
   super(view);
  }
 }
}
```

Svaki *view* ima pridružen SQL

- Moguće ga je definirati u *@View* anotaciji, ili u zasebnoj SQL datoteci
- Nema potrebe za učenjem novog jezika
- Moguće je jednostavno kopiranje SQL *query*-a u aplikaciju za pristup bazi podataka i vidjeti njegov rezultat

Klasa *AbstractView* sadrži

- Kolekciju instanci klase *AbstractViewRow*
- Atribute za varijable
- Metode za dohvaćanje podataka, specifikaciju filtera i redoslijeda podataka

Klasa *AbstractViewRow* sadrži

- Atribute za sve vrijednosti koje sadrži
- *Ključ* za pristup pojedinom atributu sastoji se od imena polja entiteta te imena polja atributa u tom entitetu
- Ukoliko se radi o vrijednosti bez pripadajućeg entiteta, koristi se *„this”*

View

```
select
 {r},
 '<test name>' as test_name
from regions r
```

```
@View(row = RegionsViewRow.class)
public class RegionsView extends AbstractView<RegionsViewRow>
{

 public static class RegionsViewRow extends AbstractViewRow {
  @EntityAttribute("R")
  @EntityAttributeMappings(
   @EntityAttributeMapping(
    column = "TEST_NAME",
    field = "region_name"
   )
  )
  private Regions region;

  public RegionsViewRow(RegionsView view) {
   super(view);
  }
 }
}
```

Obavezna izgradnja *query*-a

- Za sve definirane entitete u *Row* klasi, generira se popis njihovih stupaca te se sva pojavljivanja niza znakova *{__alias__}* s njime zamjenjuju
  - Eliminira potrebu za dodavanjem stupaca u *query* kada se entitet proširi s novim poljem iz tablice

Moguće je definirati proizvoljna mapiranja stupaca iz *query*-a na atribute entiteta

- Ovo omogućava jednostavno generiranje predložaka za nove unose iz postojećih (kada je potrebno kopirati postojeći redak, ali dati korisniku na izbor promjenu nekih vrijednosti)

Attribute

```
abstract public class AbstractAttribute<T> {
 /* … */

 protected abstract T getValueInternal();

 public final T getValue() {
  GetEvent event = null;
  if (!AbstractAttribute.isEmpty(getValueListeners)) {
   event = new GetEvent(this.getValueInternal(), this);
  }

  AbstractAttribute.notifyBefore(getValueListeners, event);
  try {
   T value = getValueInternal();
   if (!this.initialized) {
    this.defaultValue = value;
   }
   return value;
  } finally {
   AbstractAttribute.notifyAfter(getValueListeners, event);
  }
 }

 /* … */
}
```

Postavljanje i dohvaćanje vrijednosti atributa

- Svaki atribut sadrži listu *get* i *set listener-a*
- Moguće je dodati logiku prije ili poslije *get*/*set* akcije

Za atribut se smatra da ima *default*-nu vrijednost ukoliko je poziv metode *get* prethodio pozivu metode *set*

- Kao *default*-na vrijednost uzima se vrijednost koja se nalazi u atributu u trenutku poziva *get* metode

Svaki atribut ima inicijalnu vrijednost

- Kao inicijalna vrijednost uzima se vrijednost koja se prva postavi kroz *set* metodu

Default-na i inicijalna vrijednost se nalaze u instanci atributa

- Istovjetni model *listener*-a postoji i za postavljanje, odnosno dohvaćanje *default*-ne i inicijalne vrijednosti

Attribute

```
abstract public class AbstractAttribute<T> {
 /* … */

 protected abstract void setValueInternal(T value);

 public final void setValue(T newValue) {
  T oldValue = this.getValueInternal();

  SetEvent event = null;
  if (!AbstractAttribute.isEmpty(setValueListeners)) {
   event = new SetEvent(oldValue, newValue, this);
  }

  AbstractAttribute.notifyBefore(setValueListeners, event);
  try {
   this.setValueInternal(newValue);
   if (!this.isInitialized()) {
    this.setInitial(newValue);
   } else {
    this.setChanged(!ObjectUtils.equals(oldValue, newValue));
   }
  } finally {
   AbstractAttribute.notifyAfter(setValueListeners, event);
  }
 }

 /* … */
}
```

Postavljanje i dohvaćanje vrijednosti atributa

- Svaki atribut sadrži listu *get* i *set listener-a*
- Moguće je dodati logiku prije ili poslije *get*/*set* akcije

Za atribut se smatra da ima *default*-nu vrijednost ukoliko je poziv metode *get* prethodio pozivu metode *set*

- Kao *default*-na vrijednost uzima se vrijednost koja se nalazi u atributu u trenutku poziva *get* metode

Svaki atribut ima inicijalnu vrijednost

- Kao inicijalna vrijednost uzima se vrijednost koja se prva postavi kroz *set* metodu

Default-na i inicijalna vrijednost se nalaze u instanci atributa

- Istovjetni model *listener*-a postoji i za postavljanje, odnosno dohvaćanje *default*-ne i inicijalne vrijednosti

JSF // Controller

```
abstract public class Controller {

 @EJB protected ApplicationModule am;
 @EJB protected DBContext dbc;
 protected BindingManager bm;

 /* ... */
}
```

Klasa *Controller* je osnova svake stranice
- Model sadrži dvije specifične podvrste *Controller* klase
- *ViewController* koristi se na stranicama za prikaz podataka
- *EditController* koristi se na stranicama za unos podataka
- Nije nužno koristiti niti jednu vrstu kontrolera

ApplicationModule klasa koristi se za grupiranje instanci *AbstractView* klasa koje se koriste na stranici
- Ne nužno za prikaz podataka
- Svaka *AbstractView* instanca ima jedinstveno ime u *ApplicationModule*-u te zna u kojem se AM-u nalazi

DBContext klasa predstavlja pojednostavljenu transakciju baze podataka
- Moguće je dodati instancu *AbstractEntity* klase u *DBContext*
- Pozivom funkcije *DBContext.commit()*, model provjerava integritet podataka prije spremanja redaka u bazu te javlja grešku ukoliko je isti narušen
- Dodavanjem više instanci *AbstractEntity* klasa, sve će instance biti pohranjene
  - Moguće je definirati *dependency* podataka iz određenih atributa predanih instanci *AbstractEntity*-a te će se takvi podaci kopirati nakon što se redci unesu
    - Popunjavanje vrijednosti *FK* ključa

BindingManager klasa sadrži organizirane *listener*-e pojedinih atributa
- Korištenjem atributa moguće je automatski preusmjeriti podatak iz jednog u drugi

Primefaces // Composite components

Jednostavno mapiranje vrijednosti preko atributa
- Kompozitne komponente uzimaju atribut kao ulazni parametar
- Dinamičko generiranje stupaca tablica
- Moguće definirati izvan aplikacije
  - Omogućava izmjenu prikazanih stupaca bez potrebe za *deploy*-em aplikacije

Korištenjem atributa varijabli iz *AbstractView*-a i *BindingManager*-a moguće je jednostavno proslijediti odabranu vrijednost iz *frontend*-a u model

LOV komponente koriste posebnu vrstu *AbstractView* klase
- Odabrani redak predstavljen je atributom
- Jednostavno prosljeđivanje identifikatora odabranog retka u odredišni atribut

Automatsko dohvaćanje *converter*-a

Automatska validacija

Automatsko dohvaćanje labele

Primefaces // Bundle

```
@Entity
@Table(schema = "HR", name = "REGIONS")
/* ... */
@BundleDefinition(RegionsEntityBundle.class)
public class Regions extends AbstractEntity<BigDecimal> {
 @Id
 /* … */
 @Access(AccessType.PROPERTY)
 @Column(name = "REGION_ID")
 private BigDecimal regionId;
 /* … */
 @Column(name = "REGION_NAME")
 @Access(AccessType.PROPERTY)
 private String regionName;

 /* … */
}
```

```
{
    "Regions": {
        "attributes": {
            "regionName": {
                "label": "Naziv regije"
            }
        }
    }
}
```

Bundle klasa definira opisne podatke atributa potrebne za njegov prikaz

- Minimalno miješanje modela i prezentacijskog sloja
- Moguće definirati opise (*label*), formate *converter*-a, smije li se atribut moći mijenjati od strane korisnika…
- Vrijednosti vezane za atribute v*iew*-a mogu biti naslijeđene od originalnog entiteta

Primefaces // Bundle

```
@View(row = RegionsViewRow.class)
@BundleDefinition(RegionsViewBundle.class)
public class RegionsView extends AbstractView<RegionsViewRow>
{
 @Variable("NamePrefix")
 private String namePrefix;

 public static class RegionsViewRow extends AbstractViewRow {
  @EntityAttribute("R")
  private Regions region;

  @ValueAttribute("REGION_RBN")
  private BigDecimal regionRbn;

  /* … */
 }
}
```

```
{
    "RegionsView": {
        "attributes": {
            "this": {
                "regionRbn": {
                    "label": "Redni broj regije"
                }
            }
        },
        "variables": {
            "namePrefix": {
                "label": "Početak imena regije"
            }
        }
    }
}
```

Bundle klasa definira opisne podatke atributa potrebne za njegov prikaz

- Minimalno miješanje modela i prezentacijskog sloja
- Moguće definirati opise (*label*), formate *converter*-a, smije li se atribut moći mijenjati od strane korisnika…
- Vrijednosti vezane za atribute v*iew*-a mogu biti naslijeđene od originalnog entiteta

infoart

inovacija. znanje. tehnologija.

Infoart d.o.o.
Lastovska 23
HR - 10000 Zagreb

Tel: +385 1 2334 754
Fax:+385 1 2303 707

www.infoart.hr