Računarstvo i informatika

Katedra za računarstvo
Elektronski fakultet u Nišu

Sistemi baza podataka

NHibernate

Letnji semestar 2015

NHibernate

- NHibernate manipulacija objektima se bazira na korišećenju metoda koje implementira ISession interfejs.
  - Interfejs za rad sa NHibernate sesijama
  - Obezbeđuje metode za osnovne CRUD operacije
  - Persistance manger pošto obezbeđuje mehanizme za perzistenciju objekata
  - Jednostavno kreiranje i uništavanje sesije (ne zahteva puno resursa)
  - Nije thread safe. Nhibenate sesija se koristi samo u okviru jedne niti
  - Odgovara konekciju kod ADO.NET biblioteke

NHibernate

- Krieranje perzistentnog objekta.
- Prilikom kreiranja perzistentnog objekta, objektu se dodelju vrednost primarnog ključa prema mehanizmu koji je definisan mapiranjem.

```
ISession s = DataLayer.GetSession();


Entiteti.Prodavnica p = new Entiteti.Prodavnica();

p.Naziv = "Emmi Shop VII";
p.RadniDan = "08-20";
p.Subota = "08-14";
p.Nedelja = "Ne radi";

s.Save(p);

s.Flush();
s.Close();
```

NHibernate

- Učitavanje perzistenog objekta na osnovu poznate vrednosti primarnog ključa.
- Metoda ***Load*** generiše izuzetak ukoliko ne postoji vrsta sa zadatom vrednošću primarnog ključa.

```
ISession s = DataLayer.GetSession();

//Ucitavaju se podaci o prodavnici za zadatim brojem
Prodavnica.Entiteti.Prodavnica p = s.Load<Prodavnica.Entiteti.Prodavnica>(61);

MessageBox.Show(p.Naziv);

s.Close();
```

NHibernate

- Metoda ***Get*** se koristi u situacijama kada korisnik nije siguran da postoji podataka sa zadatom vrednošću primarnog ključa.

```
ISession s = DataLayer.GetSession();

Odeljenje o = s.Get<Odeljenje>(100);

if (o != null)
{
    MessageBox.Show(o.PripadaProdavnici.Naziv);
}
else
{
    MessageBox.Show("Ne postoji odeljenje sa zadatim identifikatorom");
}


s.Close();
```

NHibernate

- Metoda ***Refresh*** omogućava da se sadržaj objekta ponovo učita iz baze podataka u bilo kom trenutku.
- Pri tome se ponovo učitavaju i podaci u svim kolekcijama koje pripadaju objektu.
- Korisno u situacijama kada postoji više konkurentnih korisnika koji mogu da promene stanje objekta.

```
ISession s = DataLayer.GetSession();

Odeljenje o = s.Get<Odeljenje>(55);

//obrada podataka o odeljenju

s.Refresh(o);

s.Close();
```

NHibernate

- Metoda ***CreateQuery*** omogućava kreiranje upita za pretraživanje podataka.
- Upiti se zadaju korišćenjem HQL jezika
- Ekvivalent ***SELECT * FROM <ImeTabele>*** upita:

```
ISession s = DataLayer.GetSession();

IQuery q = s.CreateQuery("from Odeljenje");

IList<Odeljenje> odeljenja = q.List<Odeljenje>();

foreach(Odeljenje o in odeljenja)
{
    MessageBox.Show(o.Lokacija);
}

s.Close();
```

NHibernate

- Upit sa WHERE klauzulom

```
ISession s = DataLayer.GetSession();

//Odeljenja koja nemaju info pult
IQuery q = s.CreateQuery("from Odeljenje as o where o.InfoPult = 'Ne'");

IList<Odeljenje> odeljenja = q.List<Odeljenje>();

foreach (Odeljenje o in odeljenja)
{
    MessageBox.Show(o.Id + " " + o.Lokacija);
}

s.Close();
```

NHibernate

- Parametrizovani upit

```
ISession s = DataLayer.GetSession();

//Paramterizovani upit
IQuery q = s.CreateQuery("from Odeljenje as o where o.InfoPult = ? and o.BrojKasa >= ?");
q.SetParameter(0, "Ne");
q.SetInt32(1, 1);

IList<Odeljenje> odeljenja = q.List<Odeljenje>();

foreach (Odeljenje o in odeljenja)
{
    MessageBox.Show(o.Id + " " + o.Lokacija);
}

s.Close();
```

NHibernate

- Upit sa imenovanim parametrima (mogu se navoditi u bilo kom redosledu, mogu se koristiti više puta u upitu)
- Parametri se zadaju u obliku ***:ime*** .

```
ISession s = DataLayer.GetSession();

//Paramterizovani upit
IQuery q = s.CreateQuery("select o.PripadaProdavnici from Odeljenje as o "
                         + " where o.InfoPult = :pult and o.BrojKasa >= :kase");
q.SetString("pult", "Ne");
q.SetInt32("kase", 1);

IList<Entiteti.Prodavnica> prodavnice = q.List<Entiteti.Prodavnica>();

foreach (Entiteti.Prodavnica p in prodavnice)
{
    MessageBox.Show(p.Naziv);
}

s.Close();
```

NHibernate

```
ISession s = DataLayer.GetSession();

//Paramterizovani upit
IQuery q = s.CreateQuery("select o.PripadaProdavnici from Odeljenje as o "
                         + " where o.InfoPult = :pult and o.BrojKasa >= :kase "
                         + " and o.PripadaProdavnici.RadniDan = :radni_dan");
q.SetString("pult", "Ne");
q.SetInt32("kase", 1);
q.SetString("radni_dan", "07-20");

IList<Entiteti.Prodavnica> prodavnice = q.List<Entiteti.Prodavnica>();

foreach (Entiteti.Prodavnica p in prodavnice)
{
    MessageBox.Show(p.Naziv);
}

s.Close();
```

NHibernate

- Ukoliko se očekuje da upit vrati veliki broj rezultata preporučuje se korišćenje **Enumerable** metode.
- Metoda Enumerable vraća **System.Collections.IEnumerable** interfejs.
- Metoda funkcioniše tako da se inicijalno vraćaju samo identifikatori objekata koji zadovoljavaju uslove, a nakon toga uz pomoć iteratora se objekti učitavaju na zahtev.
- Metoda Enumerable daje bolej rezultate i u situacijama kada se očekuje postojanje velikog broja duplikata u podacima. Metoda se tada u velikoj meri oslanja na keš sesije, pa se jednom učitani objekti ne učitavaju ponovo.

NHibernate

```
ISession s = DataLayer.GetSession();

IQuery q = s.CreateQuery("from Odeljenje");

IEnumerable<Odeljenje> odeljenja = q.Enumerable<Odeljenje>();

foreach (Odeljenje o in odeljenja)
{
    if (o.InfoPult == "Ne")
        break;
}

s.Close();
```

NHIbernate

- Upiti koji vraćaju samo jednu vrednost (skalarnu vrednost ili instancu objekta)

```
ISession s = DataLayer.GetSession();

IQuery q = s.CreateQuery("select sum(o.BrojKasa) from Odeljenje o ");

//za slucaj da upit vraca samo jednu vrednost
Int64 kase = q.UniqueResult<Int64>();

MessageBox.Show(kase.ToString());

s.Close();
```

NHibernate

```
ISession s = DataLayer.GetSession();

IQuery q = s.CreateQuery("select o from Odeljenje o where o.Id = 55 ");

//za slucaj da upit vraca samo jednu vrednost
Odeljenje o = q.UniqueResult<Odeljenje>();

MessageBox.Show(o.Lokacija);

s.Close();
```

NHibernate

- Upiti koji vraćaju više vrednosti (skalarnih i/ili objekata)

```
ISession s = DataLayer.GetSession();

IQuery q = s.CreateQuery("select o.Lokacija, sum(o.BrojKasa), count(o) from Odeljenje o "
                         + " group by o.Lokacija ");

//za slucaj da upit vraca visestruku vrednost
IList<object[]> result = q.List<object[]>();

foreach (object[] r in result)
{
    string tip = (string)r[0];
    Int64 kase = (Int64)r[1];
    long broj = (long)r[2];

    MessageBox.Show(tip + " " + broj.ToString() + " " + kase.ToString());
}

s.Close();
```

NHibernate

- Definisanje prve rezultujuće vrste i maksimalnog broja rezultujućih vrsta

```
ISession s = DataLayer.GetSession();

IQuery q = s.CreateQuery("from Odeljenje");
q.SetFirstResult(10);
q.SetMaxResults(10);

IList<Odeljenje> odeljenja = q.List<Odeljenje>();

foreach (Odeljenje o in odeljenja)
{
    MessageBox.Show(o.Id.ToString());
}

s.Close();
```

NHibernate

- HQL predstavlja moćan jezik za kreiranje upit. Glavni nedostatak je što se upit pišu kao stringovi čiju ispravnost je nemoguće proveriti pre izvršavanja upita.
- Nhibernate kao alternativu nudi korišćenje ***ICriteria*** interfejsa.

```
ISession s = DataLayer.GetSession();

ICriteria c = s.CreateCriteria<Odeljenje>();

c.Add(Expression.Ge("BrojKasa", 1));
c.Add(Expression.Eq("InfoPult", "Ne"));

IList<Odeljenje> odeljenja = c.List<Odeljenje>();

foreach(Odeljenje o in odeljenja)
{
    MessageBox.Show(o.Id.ToString());
}

s.Close();
```

NHibernate

- QueryOver kao alternativa korišćenju NHibernate criteria izraza

```
ISession s = DataLayer.GetSession();

IList<Odeljenje> odeljenja = s.QueryOver<Odeljenje>()
                                .Where(x => x.BrojKasa >= 1)
                                .Where(x => x.InfoPult == "Ne")
                                .List<Odeljenje>();

foreach (Odeljenje o in odeljenja)
{
    MessageBox.Show(o.Id.ToString());
}

s.Close();
```

NHibernate

- **L**anguage **IN**tegrated **Q**uery (LINQ)
- LINQ predstavlja skup proširenja .NET Framework okruženja koja obuhvataju operacije za rad sa upitima, skupovima i transformacijama.
- Osnovni cilj je da se sintaksa .NET jezika proširi podrškom za kreiranje upita.
- Osnovna namena je da se u okviru .NET programskih jetika olakša pristup i integracija informacija van OO okruženja (relacione baze podataka i XML dokumenti).

NHibernate

```
public static List<string> people = new List<string>()
{
  "Granville", "John", "Rachel", "Betty",  "Chandler", "Ross",
"Monica"
};


 static void Main(string[] args)
 {
   IEnumerable<string> query = from p in people select p;

   foreach (string person in query)
   {
       Console.WriteLine(person);
   }
 }
```

NHibernate

```
static void Main(string[] args)
 {
   IEnumerable<string> query = from p in people
                               where p.Length > 5
                               order by p
                                select p.ToUpper();

   foreach (string person in query)
   {
       Console.WriteLine(person);
   }
 }
```

NHibernate

- NHibernate LINQ NuGet paket

```
PM> Install-Package NHibernate.Linq
```

- Podrška:

```
using System.Linq;
using NHibernate.Linq;
```

NHibernate

```
ISession s = DataLayer.GetSession();

IList<Odeljenje> odeljenja = (from o in s.Query<Odeljenje>()
                              where (o.BrojKasa >= 1 && o.InfoPult == "Ne")
                              select o).ToList<Odeljenje>();

foreach (Odeljenje o in odeljenja)
{
    MessageBox.Show(o.Id.ToString());
}


s.Close();
```

NHibernate

```
ISession s = DataLayer.GetSession();

IEnumerable<Proizvod> proizvodi = from p in s.Query<Proizvod>()
                                  where (p.Tip == "LUTKE" || p.Tip == "DODACI ZA LUTKE")
                                  orderby p.Tip, p.Naziv.Length
                                  select p;

foreach (Proizvod p in proizvodi)
{
    MessageBox.Show(p.Naziv);
}


s.Close();
```

NHibernate

```
ISession s = DataLayer.GetSession();

IEnumerable<Proizvod> proizvodi = s.Query<Proizvod>()
                                   .Where(p => (p.Tip == "LUTKE" || p.Tip == "DODACI ZA LUTKE"))
                                   .OrderBy(p => p.Tip).ThenBy(p => p.Naziv.Length)
                                   .Select(p => p);

foreach (Proizvod p in proizvodi)
{
    MessageBox.Show(p.Naziv);
}


s.Close();
```

NHibernate

- Izvršavanje Native SQL naredbi

```
ISession s = DataLayer.GetSession();

ISQLQuery q = s.CreateSQLQuery("SELECT O.* FROM ODELJENJE O");
q.AddEntity(typeof(Odeljenje));


IList<Odeljenje> odeljenja = q.List<Odeljenje>();

foreach (Odeljenje o in odeljenja)
{
    MessageBox.Show(o.Id.ToString());
}

s.Close();
```

NHibernate

- Sve izmene nad objektima, koje se izvrše u toku trjanja sesije u kojoj je objekat učitan, automatski će biti zapamćene u bazu podataka kada se izvrši tzv. ***Flush*** sesije.

```
ISession s = DataLayer.GetSession();

Odeljenje o = s.Load<Odeljenje>(54);
o.BrojKasa = 5;

s.Flush();

s.Close();
```

- U praksi se često javlja potreba da se objekat učita u jednoj sesiji, da se zatim koristi nezavisno od sesije i da se na kraju snimi u potpuno novoj sesiji.
- U tu svrhu se koristi metoda ***Update***.

NHibernate

```
ISession s = DataLayer.GetSession();

Odeljenje o = s.Load<Odeljenje>(54);

//originalna sesija se zatvara i raskida se veza izmedju objekta i sesije
s.Close();

//objekat se modifikuje potpuno nezavisno od sesije
o.BrojKasa = 10;

//otvara se nova sesija
ISession s1 = DataLayer.GetSession();

//poziva se Update i objekat se povezuje sa novom sesijom
s1.Update(o);

s1.Flush();
s1.Close();
```

NHibernate

- Ukoliko je u drugoj sesiji već učitan objekat sa istim identifikatorom generisaće se izuzetak prilikom poziva metode Update.
- Metoda ***SaveOrUpdate*** detektuje da li se radi o potpuno novoj instanci ili se radi o već postojećem objektu. Za nove objekte efekat je isti ka da je pozvana metoda Save a za postojeće objekte efekat je isti kao da je pozvana metoda Update.
- Za razlikovanje novih i postojećih instanci koristi se vrednost identifikatora. Podrazumevano se uzima da je kod novih objekata vrednost identifikatora 0 ali se mapiranjem može specificirati drugačija logika.

```
Id(x => x.Id, "ID").GeneratedBy.TriggerIdentity().UnsavedValue(-1);
```

NHibernate

- Update i SaveOrUpdate se koriste u sledećem scenariju:

1. Aplikacija učitava objekat u prvoj sesiji
2. Objekat se prosleđuje najčešće UI sloju
3. UI prikazuje objekat i dozvoljava korisniku da ga menja
4. UI prosleđuje modifikovani objekat sloju za rad sa podacima
5. Otvara se druga sesija i objekat se snima korišćenjem Update ili SaveOrUpdate metode

NHibernate

- SaveOrUpdate funkcioniše na sledeći način:
  - Ukoliko objekat pripada sesiji i nije modifikovan ne radi ništa
  - Ukoliko nije definisana vrednost identifikatora poziva se metoda Save
  - Ukoliko vrednost identifikatora odgovara vrednsoti definisanoj u UnsavedValue poziva se metoda Save
  - Identifikator objekta je definisan ali u sesiji već postoji objekat sa tim identifikatorom generiše se izuzetak
  - Poziva se Update
- Da bi se izbeglo generisanje izuzetka u slučaju da već postoji objekat sa istim identifikatorom, može korisitit metoda **Merge**.
- Metoda Merge kopira vrednosti objekta u objekat)koji je vezan za sesiju) sa istim identifikatorom. Ukoliko takav objekat ne postoji u sesiji učitava se iz baze podataka.

NHibernate

- Metoda ***Delete*** obezbeđuje brisanje podatka iz baze podataka.

```
ISession s = DataLayer.GetSession();

Odeljenje o = s.Load<Odeljenje>(54);

//brise se objekat iz baze ali ne i instanca objekta u memroiji
s.Delete(o);
//s.Delete("from Odeljenje");

s.Flush();
s.Close();
```

NHibernate

- Podrška za rad sa transakcijama

```
ISession s = DataLayer.GetSession();

Odeljenje o = s.Load<Odeljenje>(52);

ITransaction t = s.BeginTransaction();

s.Delete(o);

//t.Commit();
t.Rollback();

s.Close();
```

NHibernate

- Periodično Nhibernate sesija obavlja sinhronizaciju objekata u svom kešu sa objektima u bazi podataka.
- Sinhronizacija se obavlja u sledećim trenucima:

1. Prilikom pozivanja Get, CreateQuery/List, Enumerable() metoda
2. poziv NHibernate.ITransaction.Commit() metode
3. Poziv ISession.Flush metode

NHibernate

- Zatvaranje sesije:

1. Flush sesije
2. Commit transakcije
3. Close sesije
4. Obrada izuzetaka

- Ukoliko se koristi transakcija nije neophodno eksplicitno pozvati Flush metodu. Metode Commit i Rollback obezbeđuju automatski flush proces za sesiju koja se zatvara.