Oblikovanje informacijskog sustava za upravljanje stambenim zgradama vođeno domenom primjene

Željko Tepšić

Mentor: Prof.dr.sc. Nikola Bogunović
Mentor IN2: mr.sc. Ninoslav Čerkez

Sadržaj

- **Uvod**
- **Razvoj vođen domenom (DDD)**
- **Perzistencija modela domene**
- **Domena upravljanja stambenim zgradama**
- **Implementacija**
  - Domenski sloj
  - Infrastrukturni sloj
  - Aplikacijski i prezentacijski sloj (ASP.NET MVC)
- **Zaključak**

Uvod (1)

- Programska potpora je instrument stvoren radi rješavanja kompleksnih problema modernog života.
  - Mora biti praktična i korisna
- U IS stvarni svijet predstavljamo objektima koji su imenovani i oblikovani prema konceptima iz stvarnog svijeta.
- Programska potpora potječe iz domene te je time i usko povezana s domenom.
- Potrebno je dizajnirati model domene.
  - Model domene čini spregu podataka i poslovnih procesa, posjeduje višestruke vrijednosti atributa i kompleksnu mrežu asocijacija te koristi nasljeđivanje, različite strategije i ostale oblikovne obrasce.

Uvod (2)

- Zadatak diplomskog rada je primjenom postupaka oblikovanja vođenog domenom primjene specificirati, modelirati i ostvariti IS za upravljanje stambenim zgradama.
- IS ostvaren je kao web primjenski program korištenjem MVC obrasca i relacijske baze podataka.
- Za perzistenciju modela domene korišten je ORM NHibernate.
- Implementacija IS-a: C# i .NET

Razvoj vođen domenom
(Domain – Driven Design)

- DDD je pristup za razvoj kompleksne programske potpore koji duboko povezuje implementaciju sa razvijajućim modelom jezgre poslovnih koncepata.
- **Glavni temelji DDD-a su:**
  - Glavni fokus projekta postavlja se na jezgru domene i domensku logiku.
  - Složeni dizajn temelji se na modelu
    - Model – driven, a ne data – driven
  - Uspostavljanje kreativne kolaboracije između eksperata domene i razvojnika radi što bližeg približavanja konceptualnoj srži problema.
    - Sveprisutni jezik

Entiteti i vrijednosni objekti

- Entitet
  - Objekt koji je primarno određen svojim identitetom naziva se entitet.
  - Životni ciklus entiteta može radikalno promijeniti formu i sadržaj objekta, ali identitet mora biti očuvan.
  - Definicija identiteta proizlazi iz domene.
- Vrijednosni objekt
  - Objekt koji predstavlja opisni aspekt domene bez konceptualnog identiteta zove se vrijednosni objekt.
  - Vrijednosni objekti moraju biti nepromjenjivi (engl. *immutable*).

Servisi

- Servis je operacija ponuđena preko sučelja koje postoji samostalno u modelu, bez enkapsulacije stanja.
- Dobar servis ima tri karakteristike:
  - Operacija se odnosi samo na koncept iz domene koji nije prirodni dio entiteta ili vrijednosnog objekta.
  - Sučelje je definirano u okvirima drugih elemenata u domeni.
  - Operacije koje servis sadrži moraju biti bez stanja.

Agregati

- Agregat je skup povezanih objekata koje tretiramo kao jedinku prilikom promjene podataka.
- Pravila agregata:
  - Korijenski entitet ima globalni identitet i odgovoran je za provjeravanje invarijanti.
  - Entiteti unutar granice imaju samo lokalni identitet, jedinstven jedino unutar agregata.
  - Ništa izvan agregata ne može držati referencu na objekt unutar granice. Korijen može predati referencu na unutarnje entitete samo na privremeno korištenje.
  - Iz baze je moguće dobiti samo korijenske entitete.
  - Objekti unutar agregata mogu držati referencu na korijene drugih agregata.
  - Prilikom brisanja agregata, operacija brisanja mora odjednom obrisati sve elemente agregata.

Tvornice

- Stvaranje objekata može biti složena operacija.
- Programski element čija je odgovornost stvaranje drugih objekata naziva se tvornica.
- Tvornica enkapsulira znanje potrebno za stvaranje kompleksnih objekata ili agregata.
- Dva osnovna zahtjeva za implementaciju:
  - Svaka operacija stvaranja je atomarna i provodi sve invarijante.
  - Tvornica treba biti ostvarena prema apstrakcijama.

Repozitoriji

- Repozitorij je mehanizam za enkapsulaciju ponašanja spremanja, dohvaćanja i pretraživanja koji emulira kolekciju objekata.
- Repozitorij je potrebno ostvariti kroz dobro poznato globalno sučelje.
- Repozitoriji se definiraju samo za korijenske agregate koji u stvarnosti trebaju direktan pristup.

Perzistencija modela domene

- U OO aplikacijama, perzistencija omogućuje objektu da nadživi procese ili aplikaciju koja ga je kreirala.
- Aplikacija sa modelom domene ne radi direktno sa relacijskom reprezentacijom poslovnih entiteta, već isključivo preko OO modela poslovnih entiteta.
- Poslovna logika se nikada ne izvršava u bazi podataka, kao spremljene SQL procedure, već je ostvarena modelom domene.
- Fundamentalne nekompatibilnosti među paradigmama:
  - Problem identiteta
  - Problem granularnosti
  - Problem asocijacija
  - Problem nasljeđivanja i polimorfizma

Objektno – relacijsko mapiranje

- Pomoću ORM moguće je stvoriti translacijski sloj koji će jednostavno transformirati objekte u relacijske podatke i obratno.
- Objektno relacijsko mapiranje je automatizirana perzistencija objekata aplikacije u tablice relacijske baze podataka, koristeći meta podatke koji opisuju preslikavanje između objekata i baze podataka.
- NHibernate je besplatno (*open source*) rješenje objektno – relacijskog mapiranja za .NET platformu.
- NHibernate automatizira mnoge ponavljajuće zadatke kodiranja, ali znanje tehnologije perzistencije mora postojati i izvan NHibernatea.

Implementacija modela domene za NH (1)

- Nije potrebno naslijediti ili implementirati bazne razrede ili sučelja.
- Model domene mora biti implementiran primjenom POCO **(P**lain **O**ld **C**LR **O**bject**)** modela programiranja.
  - Objekti implementirani isključivo .NET-om te ne ovise o vanjskim knjižnicama.
- Za ostvarenje *lijenog dohvaćanja* potrebno je zadovoljiti:
  - Razred mora imati definiran defaultni (ne privatan) konstruktor.
  - Razred ne smije biti zaključan za nasljeđivanje (engl. *sealed*).
  - Sve javne metode i sva javna svojstva moraju biti virtualna.
  - Ne smiju postojati javne članske varijable.

Implementacija modela domene za NH (2)

- Kolekcije moraju biti prema van predstavljene preko sučelja.
- Upravljanje asocijacijama prepušteno je programskom kodu unutar POCO objekata.
- Definiranje meta podataka o mapiranju
  - Specificiraju preslikavanje između razreda i tablica, svojstava i stupaca, asocijacija i stranih ključeva, .NET i SQL tipova.
  - Moguće pomoću .NET atributa ili pomoću XML datoteka.

Životni ciklus NH objekata

- Tranzijentni objekti – objekti koji nisu perzistirani
- Perzistentni objekti – objekti koji su povezani sa bazom podataka
- Odvojeni objekti – objekti koji su izgubili vezu sa upraviteljem perzistencije (ISession)
- NH posjeduje upravitelja perzistencije ISession.
- Sjednica se stvara na početku svake jedinice posla i traje do njezinog kraja.
- ISession jamči da za vrijeme trajanja sjednice postoji točno jedan objekt koji predstavlja neki redak u tablici baze podataka.

Upravljanje stambenim zgradama (1)

- Sudionici upravljanja su suvlasnici i upravitelj.
- Upravitelj upravlja zgradom, održava ju i prikuplja pričuvu za zgradu.
- Izvor prihoda kojim se osigurava i ostvaruje briga za stambenu zgradu jest pričuva.
- Potpuno uređen suvlasnički odnos u nekoj nekretnini postoji kada se točno utvrdi tko je vlasnik kojega dijela zgrade – etažiranje.

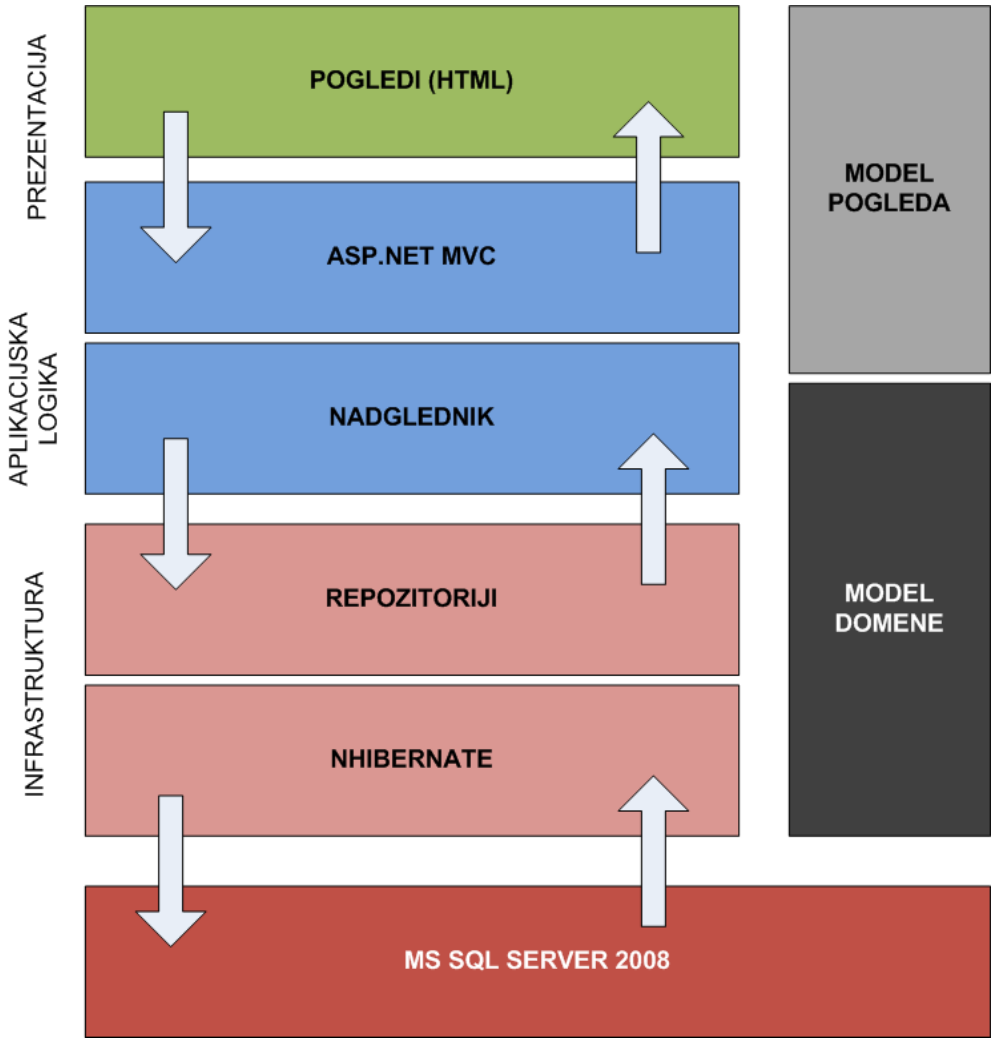
Upravljanje stambenim zgradama (2)

- Svi poslovi koje suvlasnici poduzimaju na zgradi imaju karakter redovne i izvanredne uprave.
- O redovnoj upravi suvlasnici odlučuju većinom glasova.
  - Odluka se smatra donesenom kada se za nju izjasne suvlasnici koji zajedno imaju većinu suvlasničkih dijelova.
- Dok je za izvanrednu upravu potrebna suglasnost svih suvlasnika.

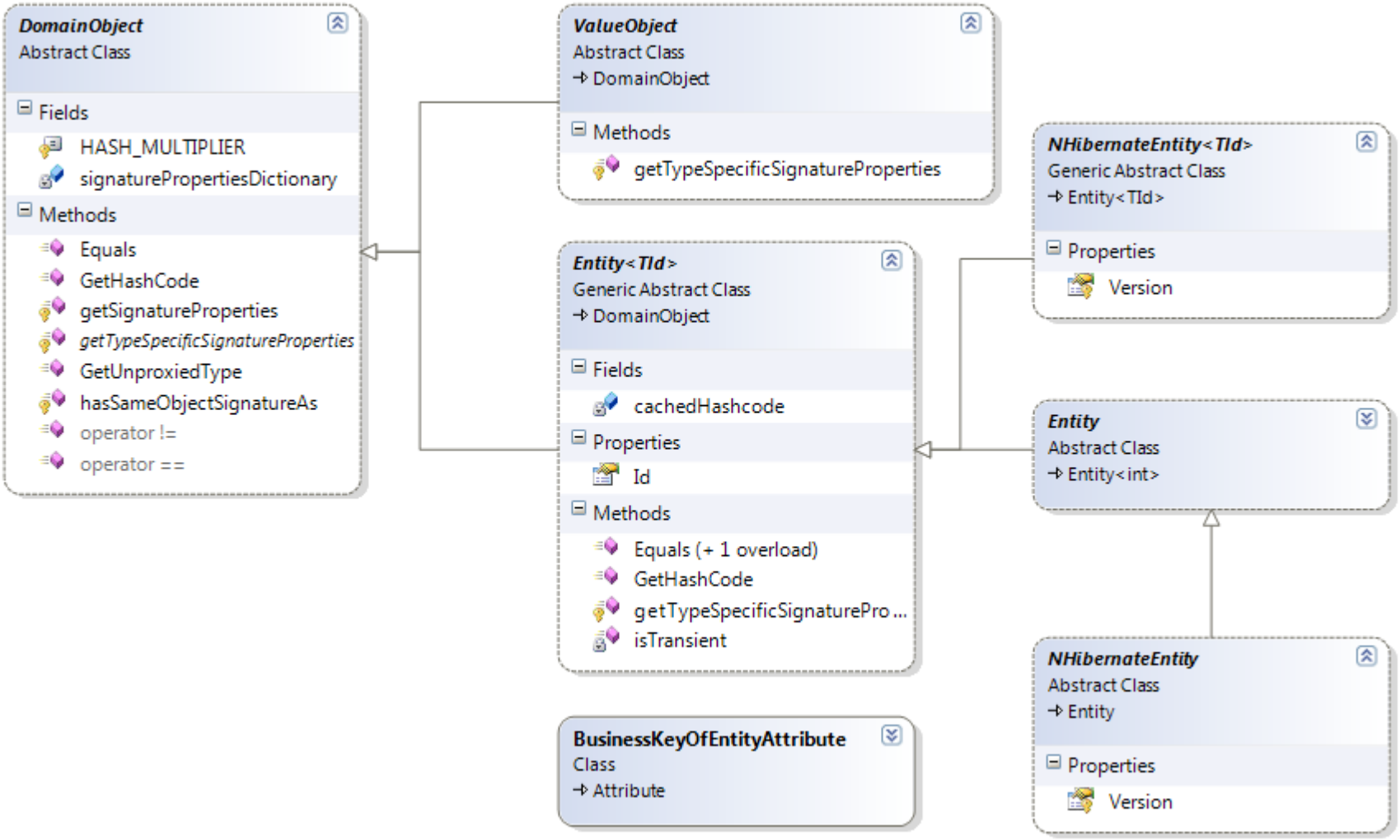
Upravljanje stambenim zgradama (3)

- Suvlasnici mogu prijaviti kvarove upravitelju.
- Upravitelj za prijavljene kvarove angažira svoje izvođače radova za sanaciju kvarova.
- Predstavnik suvlasnika mora potvrditi da je posao sanacije kvara obavljen.
- Predstavnik suvlasnika mora odobriti svako plaćanje iz pričuve upravitelju.

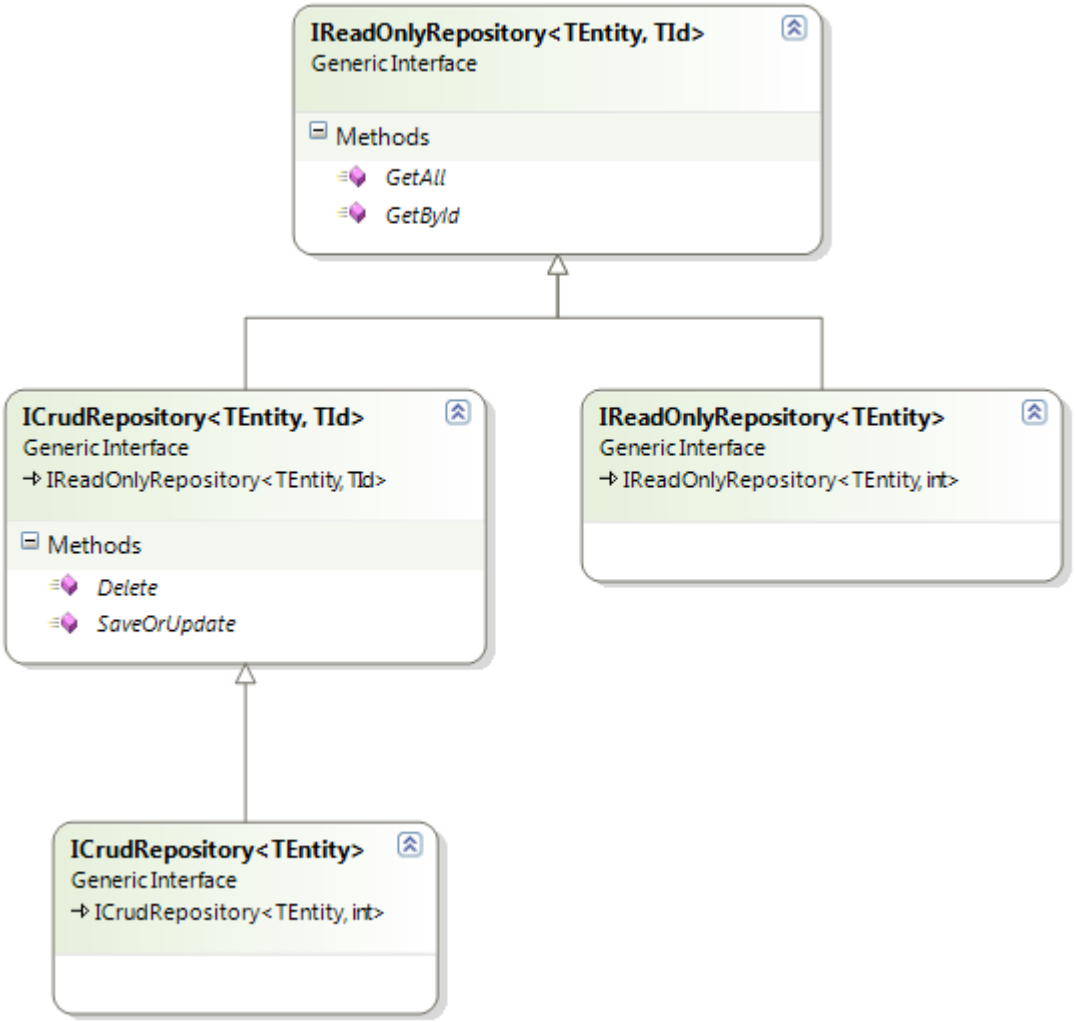
Implementacija



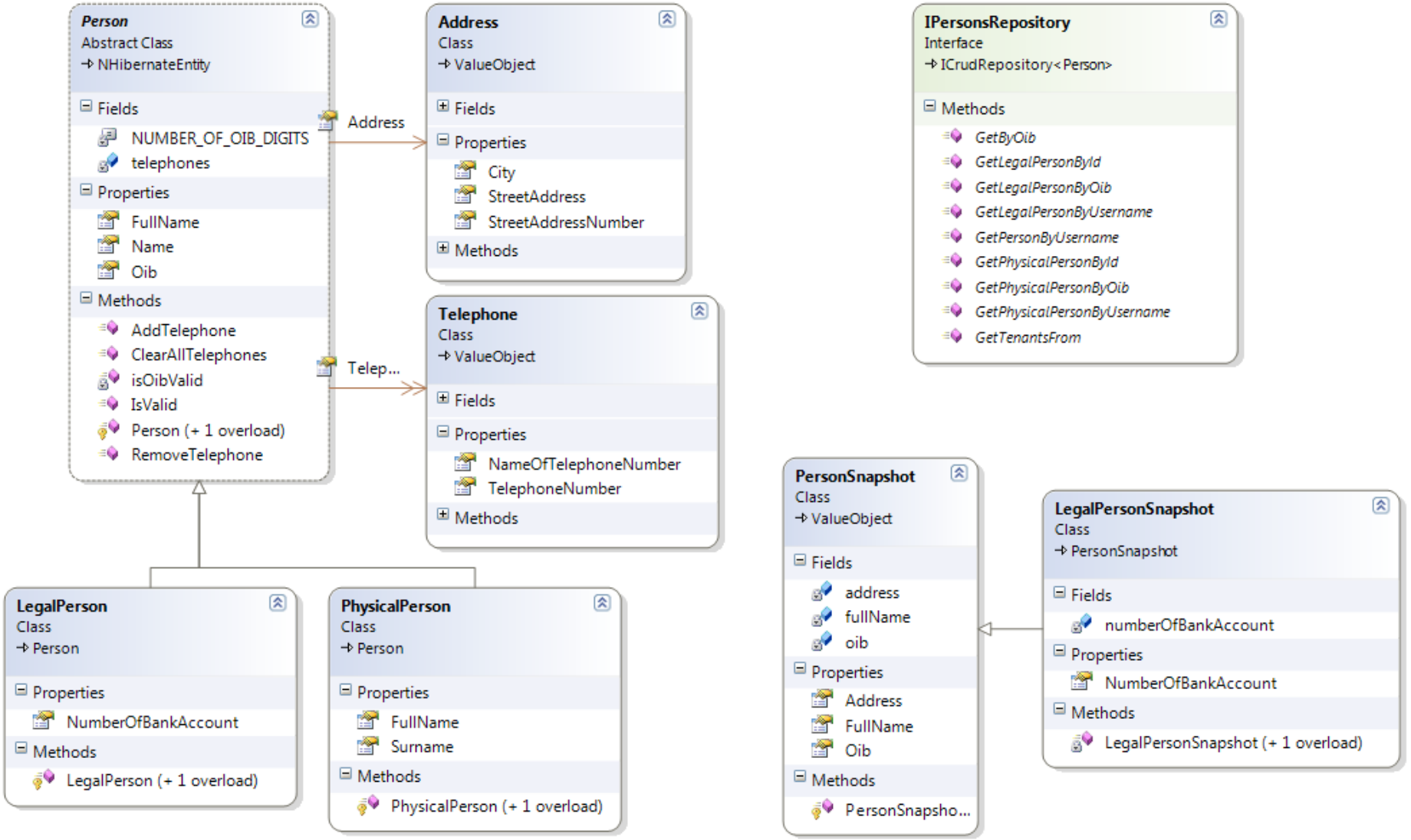
Implementacija – Apstrakcije (1)



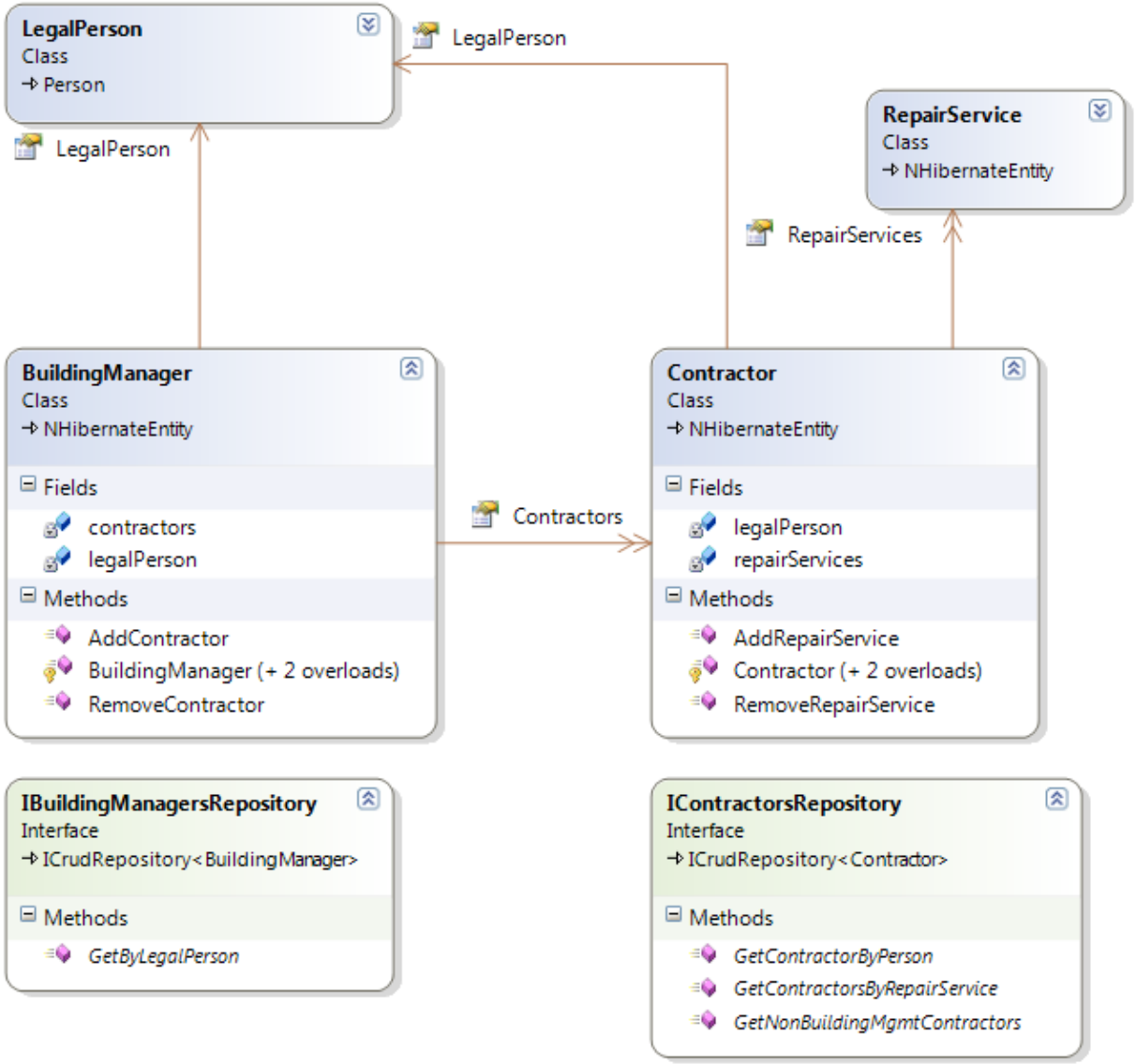
Implementacija – Apstrakcije (2)



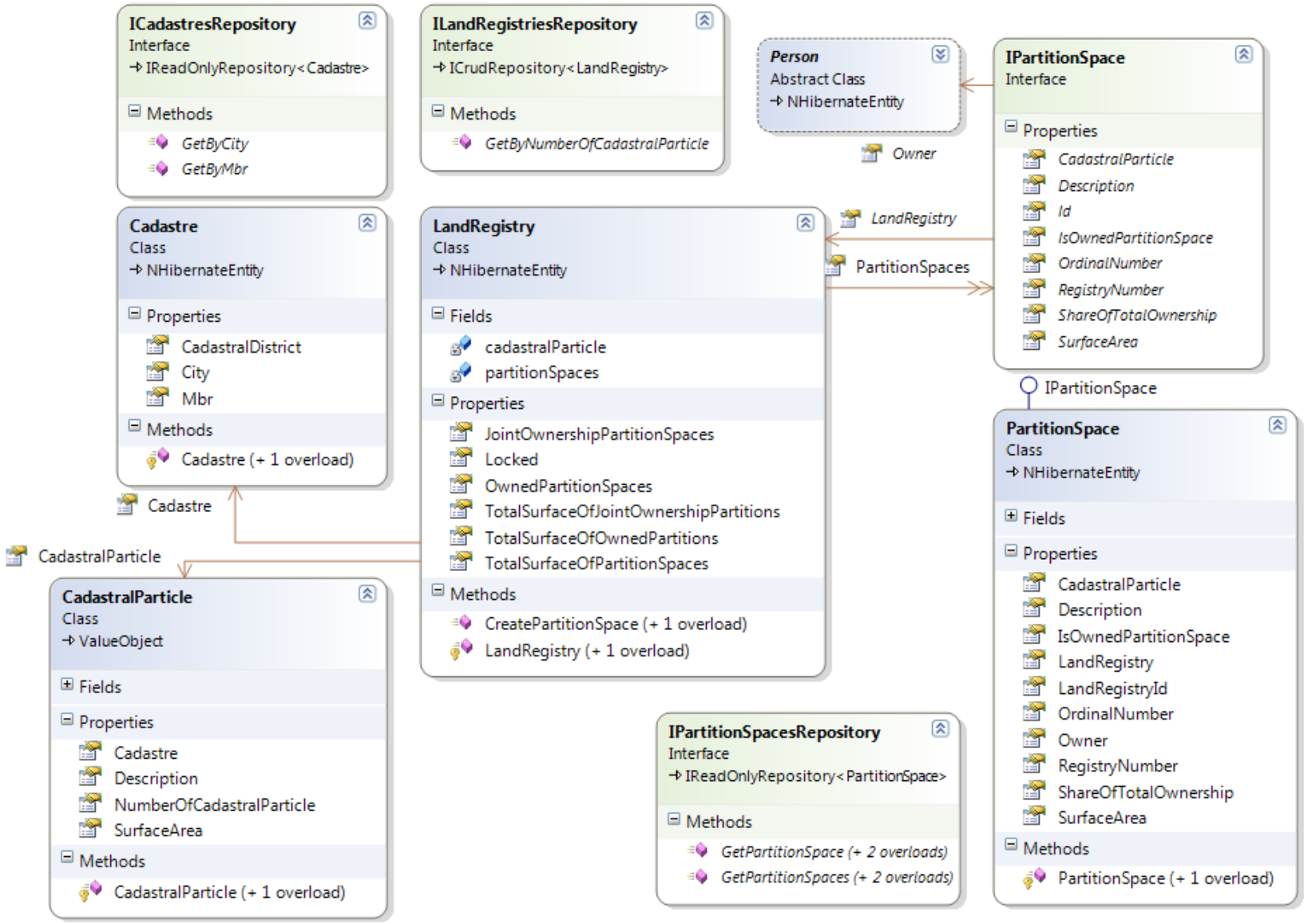
Implementacija – Osobe i uloge (1)



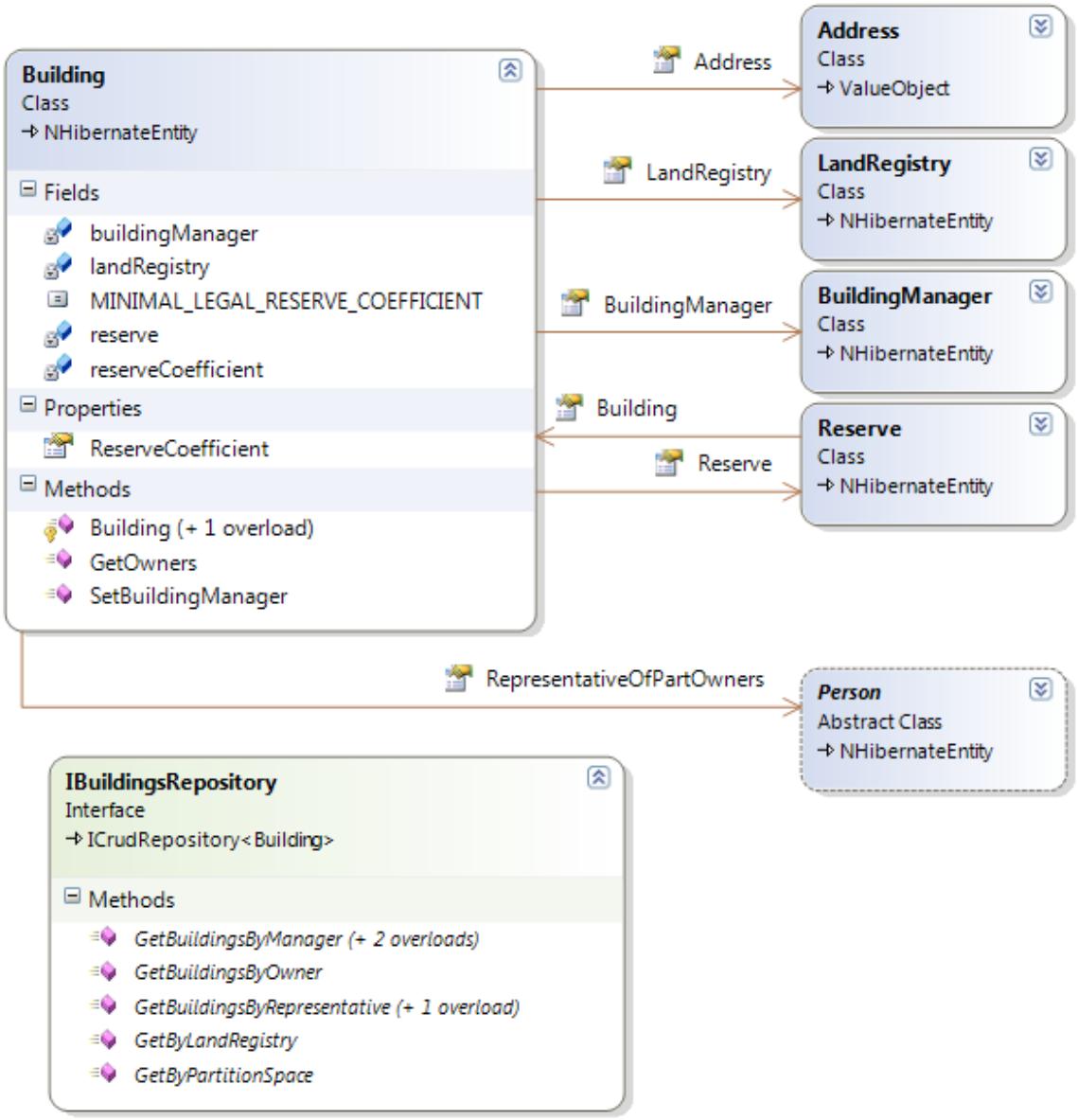
Implementacija – Osobe i uloge (2)



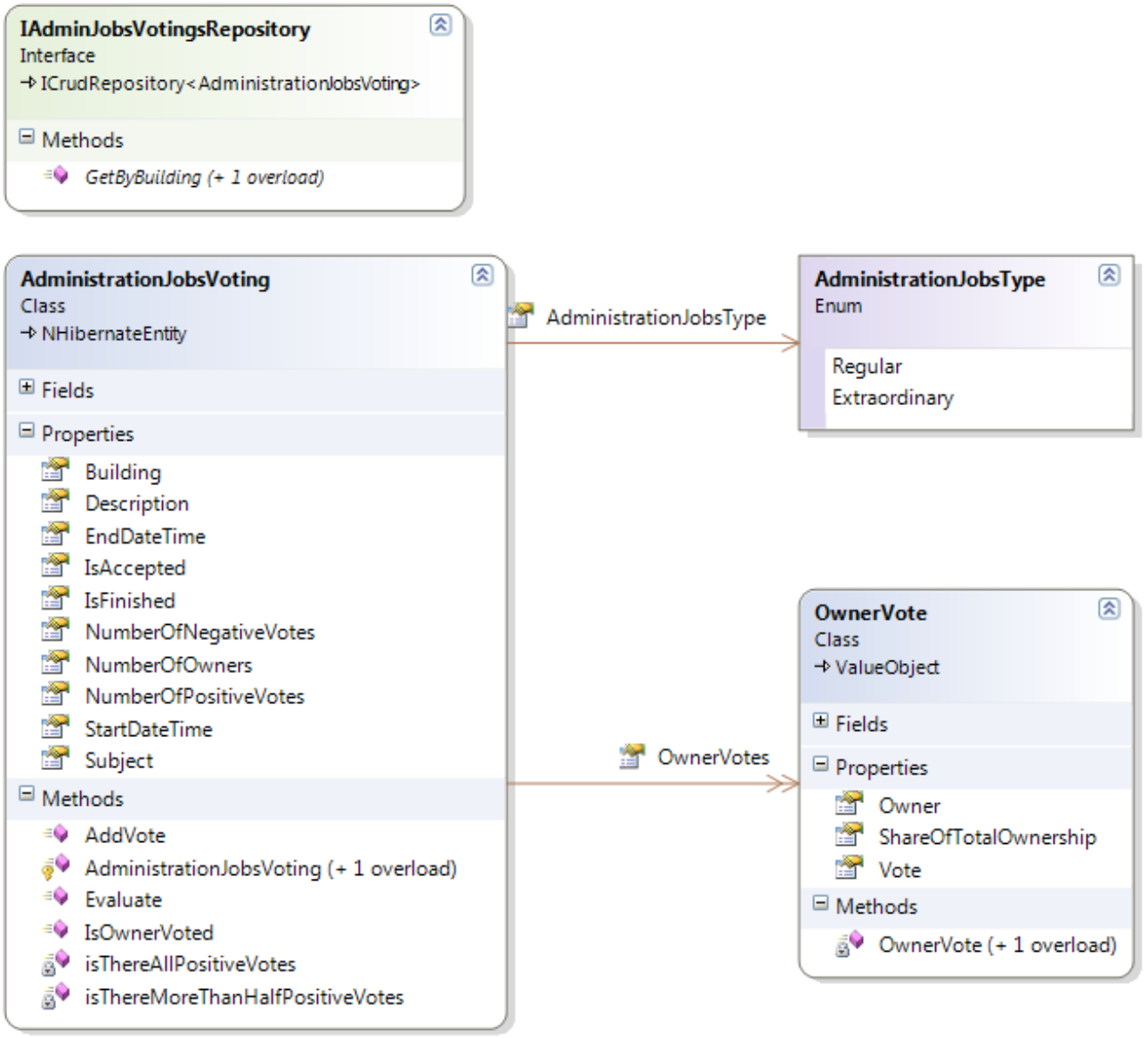
Implementacija – Zakonodavstvo



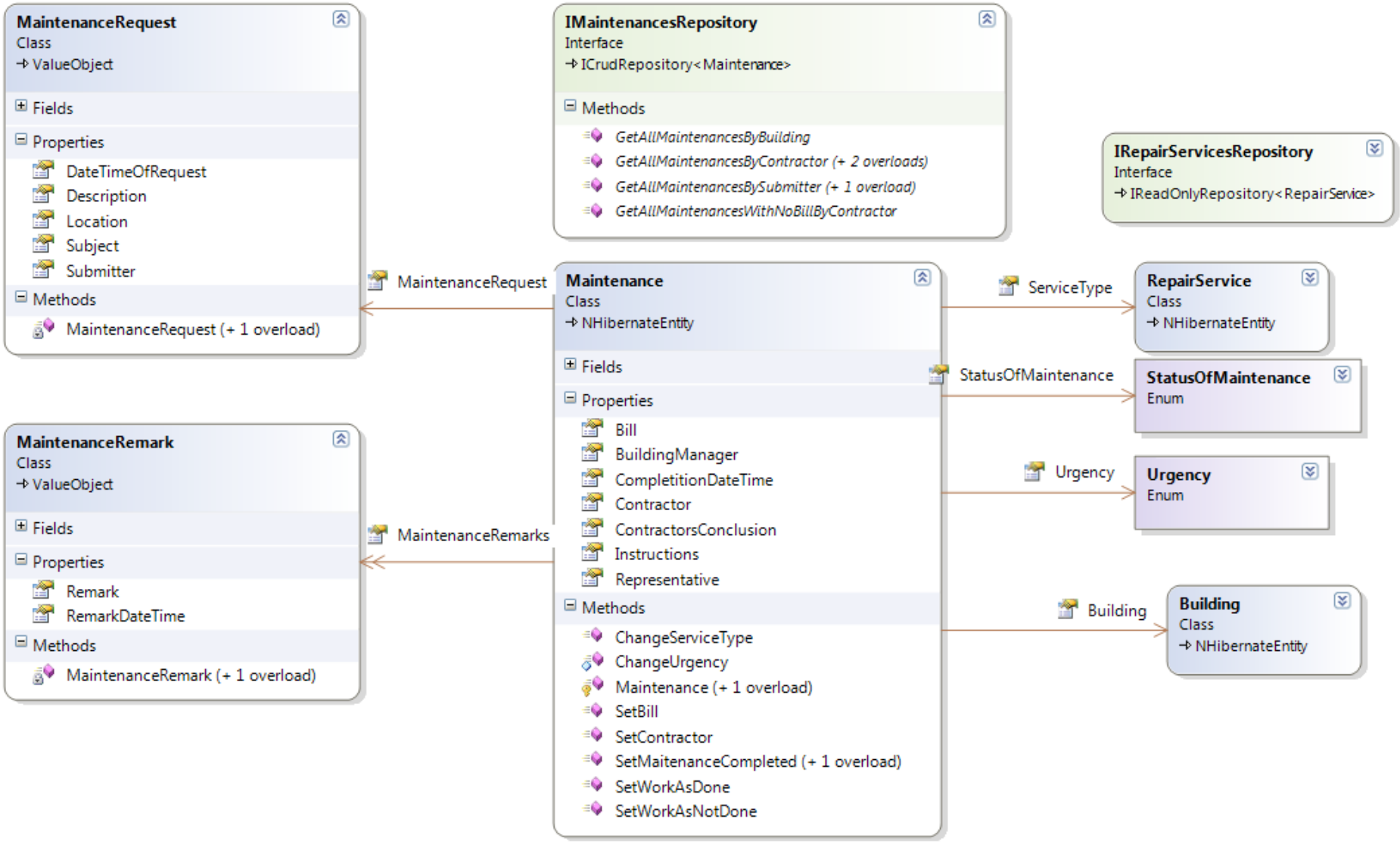
Implementacija – Upravljanje zgradom (1)



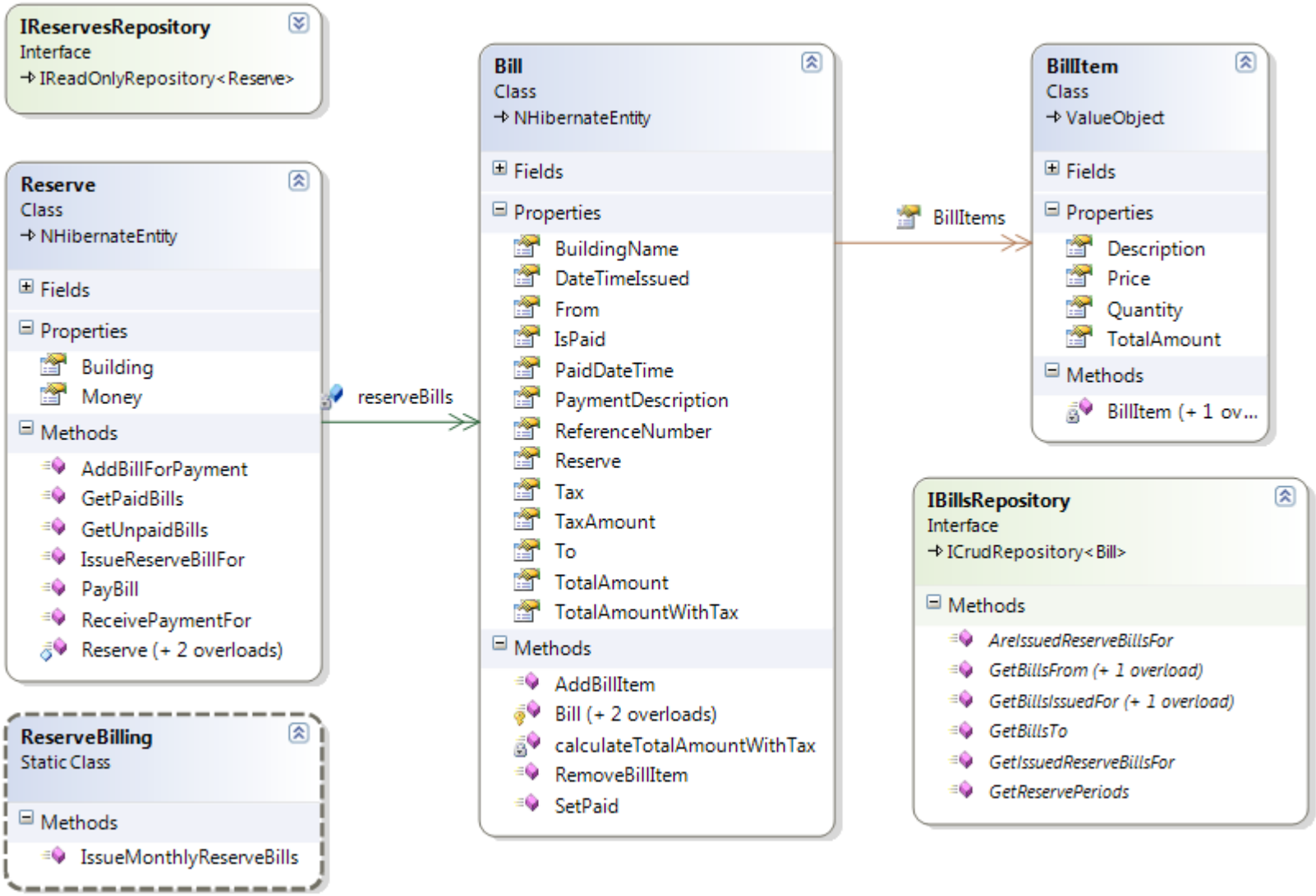
Implementacija – Upravljanje zgradom (2)



Implementacija – Prijava kvara



Implementacija – Financije

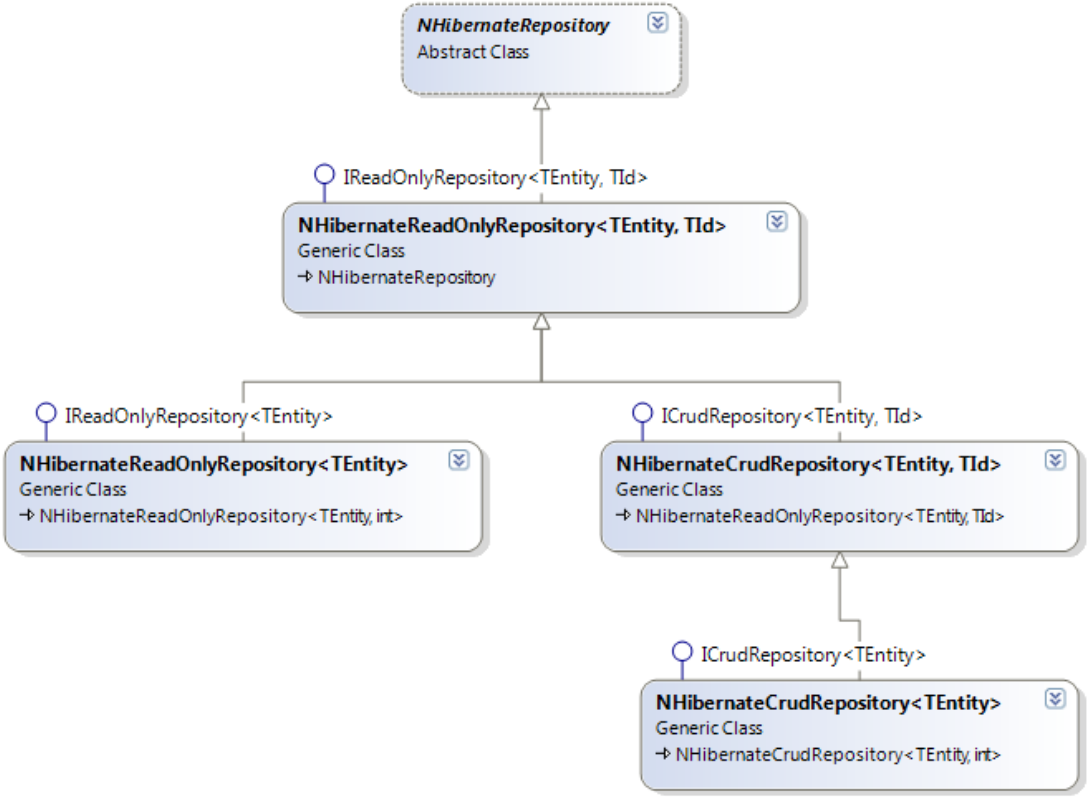


Infrastruktura – NH sjednice i transakcije

- NH prepušta upravljanje sjednicama i transakcijama aplikaciji.
- NH značajka kontekstualne sjednice omogućuje da se sjednica asocira sa specifičnim djelokrugom aplikacije koja aproksimira jedinicu posla.
  - Tipično u web aplikaciji sjednica se asocira sa http zahtjevom.
- Cilj je otvaranje sjednice i transakcije što više odgoditi
  - U ovome slučaju to je Akcija nadglednika.
- Implementiran `NHibernateTransactionAttribute`
  - Dekorira akciju nadglednika.
- Stvaranje sjednice je jeftina operacija, stvaranje tvornice sjednice je skupa operacija
  - `NHibernateSessionProvider` sigleton objekt

Infrastruktura – NH Repozitoriji

- Implementacija na temelju sučelja iz domenskog sloja.
- Svaki upit na bazu izvodi se u kontekstu NH sjednice.
- Temeljem osnovnih NH implementacija repozitorija ostvareni domenski specifični repozitoriji.
- Testiranje ostvareno pomoću SQLite baze podataka (baza podataka u memoriji)



Infrastruktura – Autentifikacija i autorizacija

- Temelji se na ASP.NET programskim knjižnicama.
- `Forms Authentication` za autentifikaciju.
- `Membership` – korisnički računi.
- `Role` – uloge.
- **Problem**: postojeća implementacija – duboko povezana sa Microsoft SQL Serverom, koristi čisti SQL i već predefinirane podatkovne sheme
- Onečišćuje model domene i otežava korištenje objektno – relacijske tehnologije.
- **Rješenje:** Ostvaren model domene (entiteti, agregati, repozitoriji).
- Temeljem sučelja i apstraktnih razreda Membership i Role ostvareno je rješenje koje radi sa NHibernateom.

Aplikacijski i prezentacijski sloj (1)

- Interakcija korisnika i IS obavlja se posredovanjem nadglednika.
- Svi nadglednici definiraju ovisnosti o repozitorijima i aplikacijskim servisima preko parametara (sučelja) konstruktora (engl. *dependency injection)*.
- Odgovornost za instanciranje repozitorija i aplikacijskih servisa predana je izvan dosega samog nadglednika (engl. *inversion of controll*).
  - U tu svrhu korištenja je programska knjižnica Ninject za MVC.
- Akcije nadglednika obavljaju poslovne operacije isključivo sa elementima domene.

Aplikacijski i prezentacijski sloj (2)

- Elementi modela domene često puta su previše kompleksni za korištenje u prezentaciji.
- Problem koji bi se javio prilikom direktnog korištenja elemenata domene u prezentaciji je doseg NH sjednice i lijeno dohvaćanje.
- Za svaki pogled definiran je jedan poseban razred koji sadrži sve informacije koje će se prezentirati u pogledu.
  - Sadrži samo podatke, bez ponašanja (engl. data transfer object)
  - Preslikava samo potrebne podatke iz modela domene.
  - Umjesto ručnog preslikavanja koristi se AutoMapper.
    - Potrebno je definirati konfiguraciju i ostalo se sve obavlja automatski.
- Pogledi prikazuju informacije korisnicima IS-a (HTML).

Zaključak (1)

- DDD nije uvijek najbolje rješenje za ostvarenje IS-a. Koristi se:
  - Kada postoji značajna složenost poslovnih procesa i fokus na dobro definiran poslovni model
  - Kada postoji suradnja sa ekspertima domene.
  - Za IS za koje je poznato da će se dalje razvijati i imati dug životni vijek.
  - Moguće je koristiti DDD i u jednostavnijim projektima samo je pitanje da li je to isplativo zbog složenosti razvoja.
- DDD zahtjeva veliku disciplinu prilikom razvijanja od strane razvojnika.
- Na početku sporo napredovanje u razvoju, kasnije korist u obliku lakog održavanja, proširenja sustava i u ispravnome radu.

Zaključak (2)

- Prednosti DDD-a:
  - Visoka kohezija.
  - Mali stupanj međuovisnosti u aplikacijskom kodu.
  - Jednostavno testiranje
  - Izoliranost poslovne logike od ostalih slojeva
- Premda DDD zastupa *Persistence ignorance* potrebno je razmišljati i o implementacijskim detaljima, odnosno performansama.
  - Repozitoriji (korijenski agregati - performanse) vs. kolekcije kao članovi agregata (u nekim slučajevima podskup kolekcije)
  - Repozitoriji trebaju osim konceptima iz domene omogućiti specificiranje načina dohvaćanja (eager ili lazy)

Literatura

1. *Eric Evans: Domain-Driven Design: Tackling Complexity in the Heart of Software, Addison Wesley, 2003.*
2. *Jimmy Nilsson: Applying Domain-Driven Design And Patterns: With Examples in C# and .NET, Addison Wesley, 2006.*
3. *Martin Fowler: Patterns of Enterprise Application Architecture, Addison Wesley, 2002.*
4. *Pierre Henri Kuaté, Tobin Harris, Christian Bauer, Gavin King: NHibernate in Action, Manning, 2009.*
5. *Jason Dentler: NHibernate 3.0 Cookbook, Packt Publishing, 2010.*
6. *Steven Sanderson: Pro ASP.NET MVC 2 Framework, Apress, 2010.*