Havit.Business

[Framework 2](#_Toc165095984)

[Business objekty 2](#_Toc165095985)

[IdentityMap 2](#_Toc165095986)

[Model 2](#_Toc165095987)

[Query 2](#_Toc165095988)

[Dokumentace 2](#_Toc165095989)

[Generované třídy 2](#_Toc165095990)

[Konfigurace databáze 2](#_Toc165095991)

[Business objekt 2](#_Toc165095992)

[Kontruktor 2](#_Toc165095993)

[CreateObject 2](#_Toc165095994)

[GetObject 3](#_Toc165095995)

[Vlastnosti 3](#_Toc165095996)

[Inicializace 4](#_Toc165095997)

[Načítání objektu, ukládání objektu 4](#_Toc165095998)

[GetFirst, GetList 4](#_Toc165095999)

[CheckCostraints - kontrola business pravidel 4](#_Toc165096000)

[Cachování 4](#_Toc165096001)

[Kolekce business objektů 5](#_Toc165096002)

[Metoda LoadAll 5](#_Toc165096003)

[Metody Find, FindAll 5](#_Toc165096004)

# Framework

## Business objekty

* BusinessObjectBase
* ActiveRecordBusinessObjectBase
* BusinessObjectCollection

Základní objekty, které zajišťují práci s daty. Poskytují infrastrukturu tříd a metod pro funkčnost business vrstvy – načítání a ukládání objektů z/do databáze, apod.

## IdentityMap

Zajišťuje, aby byl každý objekt existoval nejvýše jednou. Zamezuje vzniku dvou různých instancí pro stejný objekt. Má charakter kontextu (podobně jako třída TransactionScope).

Vztahuje se jen na úplně načítané objekty – objekty načítané automaticky (načítání dat pro ghost objektu) a objekty načítané v kolekci (metoda LoadAll kolekce).

IdentityMap je nutné zakládat a rušit na „příslušném místě“ (např. global.asax).

## Model

Každá vygenerovaná třída obsahuje seznam svých vlastností a informaci o databázové struktuře dané třídy. Jde o statické vlastnosti ObjectInfo a Properties.

## Query

Obsahuje třídy a metody pro automatické skládání jednoduchých dotazů do databáze.

### Dokumentace

Viz [Dokumentace k HAVIT.NET Framework Extensions](HAVIT%20.NET%20Framework%20Extensions.chm)

# Generované třídy

Na základě databáze jsou generovány třídy business vrstvy.

## Konfigurace databáze

Viz <http://knowledge-base.havit.cz/navody-postupy/Business-Layer-Generator.aspx>

## Business objekt

Objekt může být načten úplně nebo částečně. K úplnému načtení dochází automaticky tehdy, když se načítají data a programátor neurčí, co se má načítat (Query.Properties). Částečně načítané objekty se neukládají do IdentityMap ani do cache.

### Kontruktor

Třídy neobsahují veřejný konstruktor (kontruktor je protected). Nelze proto vytvářet instance tříd business vrstvy pomocí klíčového slova new. K vytváření instancí slouží metody CreateObject a GetObject.

### CreateObject

Vytvoří novou instanci dané třídy.

Pokud jde o třídu, která je položkou kompozice jiné třídy (např. jde o položku faktury, která patří nějaké faktuře), přijímá metoda nadřazený objekt jako svůj parametr. Metoda automaticky nastaví předaný parametr do odpovídající vlastnosti a případně zařadí nový objekt do kolekce nadřazeného objektu. Např.

class PolozkaFaktury : ...

{

public PolozkaFaktury Create(Faktura faktura)

{

PolozkaFaktury result = new PolozkaFaktura() // vytvoří novou položku faktury

result.Faktura = faktura; // nastaví nadřazený objekt

faktura.Položky.Add(result);

// pokud existuje kolekce, pak je do ní nový objekt automaticky předán

return result;

}

}

### GetObject

Vrátí instanci daného objektu. Má jeden parametr – ID objektu.

Pokud jde o instanci stejné třídy se stejným ID, která je v identity map, vrátí se instance z identity map.

### Vlastnosti

Sloupců v databázi jsou vytvořeny vlastnosti v C# kódu. Pro sloupce, které jsou cizím klíčem do jiné tabulky se jako datový typ použije třída, která odpovídá cílové tabulce cizího klíče.

Hodnoty jsou evidovány v PropertyHolderech, pomocných objektech, které evidují, zda byla hodnota inicializována, změněna, apod.

Pokud uživatel objektu přistoupí k nějaké hodnotě vlastnosti objektu, který dosud nebyl načten (je ghostem), automaticky dojde k načtení hodnot objektu. Jinými slovy: Chceme-li hodnotu nenačteného objektu, objekt se načte.

Pokud uživatel objektu čte hodnotu neinicializované vlastnosti, je vyhozena výjimka. Jinými slovy: Čteme vlastnost, která nebyla z databáze načtena, jde o chybu.

PropertyHolder

* IsInicialized  
  Udává, zda je hodnota vlastnosti inicializována (načtena). True pro nové objekty.
* IsDirty  
  Udává, zda je hodnota vlastnosti změněna. True pro nové objekty.

CollectionPropertyHolder

* IsInicialized  
  Udává, zda je hodnota vlastnosti inicializována (načtena). True, pro nové objekty.
* IsDirty  
  Udává, zda je hodnota vlastnosti změněna, tj. zda je do kolekce přidán objekt nebo zda je nějaký objekt z kolekce odstraněn. False, pro nové objekty.

### Inicializace

Každý objekt je po vytvoření inicializován. Inicializují se PropertyHoldery vlastností, které nesou data.

Nové objekty inicializují hodnotu vlastností:

* Pokud má sloupec výchozí (default) hodnotu v databázi, použije se tato default hodnota (skalární hodnota, pro datum je možno použít funkci GETDATE()).
* Pokud sloupec v databázi nemá výchozí hodnotu, použije se výchozí hodnota pro daný typ (s ohledem na možnost uložení hodnoty null).

### Načítání objektu, ukládání objektu

Metody pro načítání a ukládání objektů automaticky vytváří generátor business vrstvy.

### GetFirst, GetList

Metody přijímají instanci třídy Query jako parametr.

Metody zajistí vytvoření (poskládání) databázového dotazu a jeho spuštění. Z vrácených dat jsou vytvořeny a vráceny objekty. GetList vrací kolekci objektů (nejsou-li vrácena žádná data, vrací prázdnou kolekci), GetFirst vrací první nalezený objekt nebo null, pokud žádný objekt splňující podmínky není nalezen.

Metody vrací:

* nenačtené objekty – ghosty – pokud Query.Properties obsahuje právě jeden sloupec a to Properties.ID dané třídy,
* plně načtené objekty, pokud parametr Query.Properties je prázdnou kolekcí,
* v ostatních případech částečně načtené objekty

Toto členění má vliv na identity map a cachování.

### CheckCostraints - kontrola business pravidel

Metoda slouží ke kontrole business pravidel ukládaných objektů. Pokud není splněno nějaké pravidlo, je vyhozena výjimka ConstraintViolationException. Metoda slouží k přepsání (override) a dopsání vlastních pravidel.

Standardně se testují jenom změněné hodnoty (u nového objektu jsou změněné všechny).

Příklad generovaného kódu:

if (\_NejakaVlastnostPropertyHolder.IsDirty

&& (\_NejakaVlastnostPropertyHolder.Value == null))

{

throw new ConstraintViolationException(this,

"Vlastnost \"NejakaVlastnost\" nesmí nabývat hodnoty null.");

}

### Cachování

Cachování objektů je nutné nastavit pomocí extended property nad tabulkou v databázi.

#### Cachování readonly objektů

Read only objekt je uložen do cache jako celek v metodě GetObject, instance jsou sdíleny thready.

Nepředpokládá se, že by readonly objekt odkazoval na jiný objekt, který není readonly.

#### Cachování non-readonly objektů

Objektu, který má být cachován, ale není readonly, jsou cachována zdrojová data, ze kterých je objekt inicializován (instance třídy DataRecord).

Data z cache se použijí:

* Při automatickém načtení hodnot objektu
* Při dočítání kolekcí pomocí metody LoadAll

Cachovaná data jsou z cache automaticky vyřazena při uložení objektu. Pozor, nikoliv při dokončení transakce – tj. může se stát, že objekt je ukládán do databáze vyhozen z cache, načten jiným threadem a uložen do cache a teprv poté je proveden commit. Potom data načtená jiným threadem a uložená do cache jsou původní data objektu!

## Kolekce business objektů

### Metoda LoadAll

Pokud se pracuje s objekty v kolekci po jednom (např. výpis v gridu), je každý nenačtený objekt (ghost) načítán z databáze jedním databázovým dotazem.

Tato metoda zajistí načtení všech nenačtených objektů (ghostů) v kolekci jedním databázovým dotazem.

Pokud je třída cachovaná, pokusí se nejprve nalézt objekt v cache. Pokud není objekt kolekce v cache, budou data načtena z databáze a do cache uložena.

### Metody Find, FindAll

Metody slouží k hledání objektů v kolekci, podstatné je, že obě metody nejprve zavolají LoadAll, aby načetly hodnoty objektů.