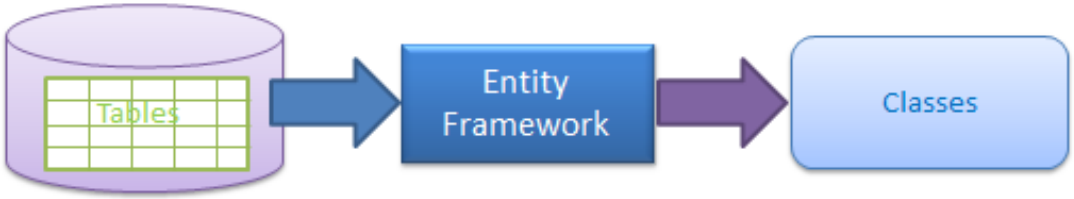
MVC 5 (Model-View-Controller)

Entity Framework (EF)

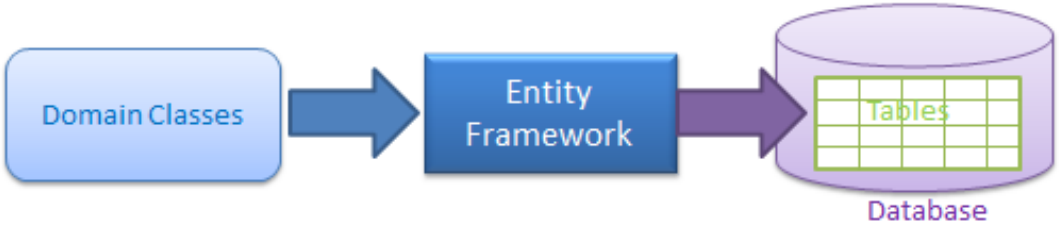
Što je Entity Framework

- O/RM (Object/Relational Mapping) radna okolina za pristup bazama podataka
- Mapiramo objekte iz baze te ih onda koristimo programirajući u .Net radnoj okolini, koristeći standardne .Net jezike (C#, Visual Basic)

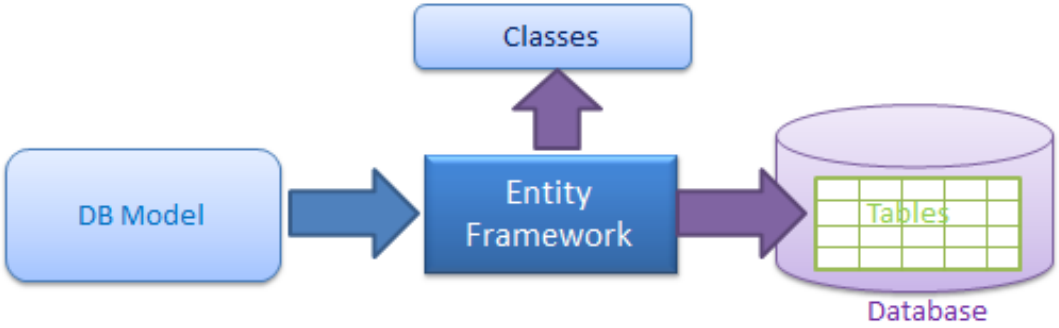
Mogućnosti EF-a



Database First - generiramo C# klase iz postojeće baze



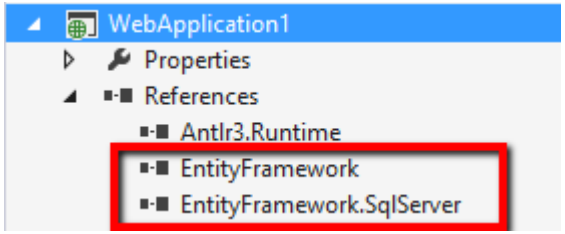
Code First - kreiramo bazu na osnovu C# klasa



Model First - kreiramo bazu i klase iz dizajniranog modela

Instaliranje EF-a

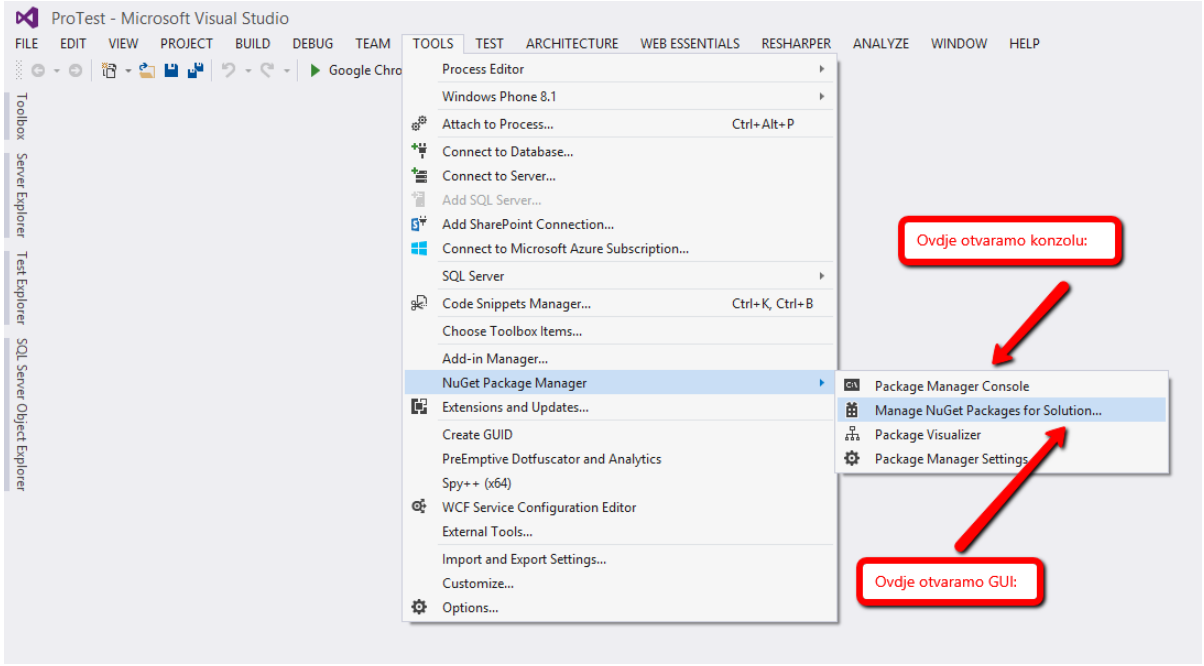
- Visual Studio 2013 defaultno instalira EF u okviru ASP.NET Web Application projektnog predloška, što možemo vidjeti po referencama novo kreiranog predloška:



- Ako želimo instalirati stariju ili noviju verziju EF-a nego što se defaultno nudi, moramo to sami napraviti

Nuget Package Manager

- Dodatak koji nam pomaže instalirati pakete poput Entity Frameworka
- Može se koristiti kroz GUI ili kroz komandnu liniju



Rad kroz komandnu liniju

- **Instalacija:**

Install-Package EntityFramework

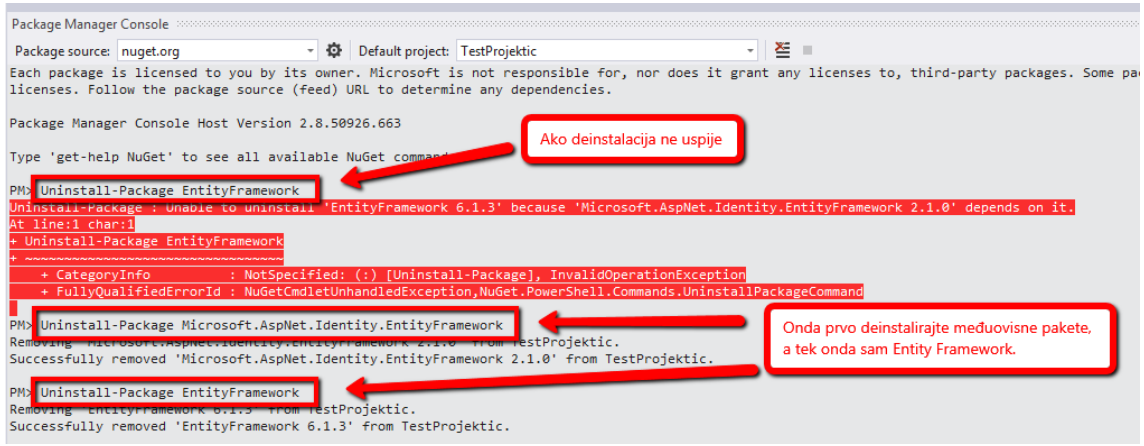
```
PM>  Install-Package EntityFramework
'EntityFramework 6.1.3' already installed.
Removing 'EntityFramework 6.1.1' from WebApplication5.
Successfully removed 'EntityFramework 6.1.1' from WebApplication5.
Adding 'EntityFramework 6.1.3' to WebApplication5.
Successfully added 'EntityFramework 6.1.3' to WebApplication5.

Type 'get-help EntityFramework' to see all available Entity Framework commands.

PM>
```

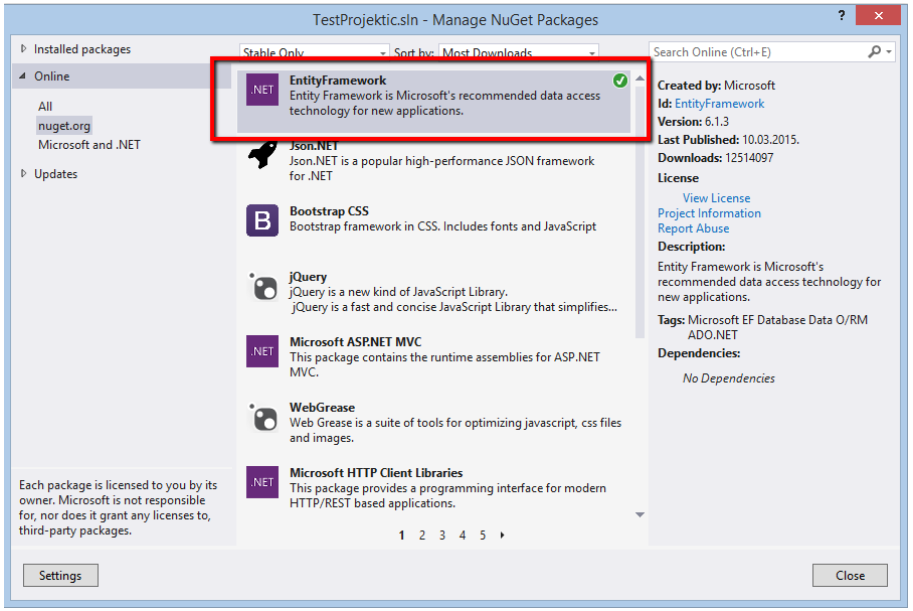
- **Deinstalacija:**

Uninstall-Package EntityFramework



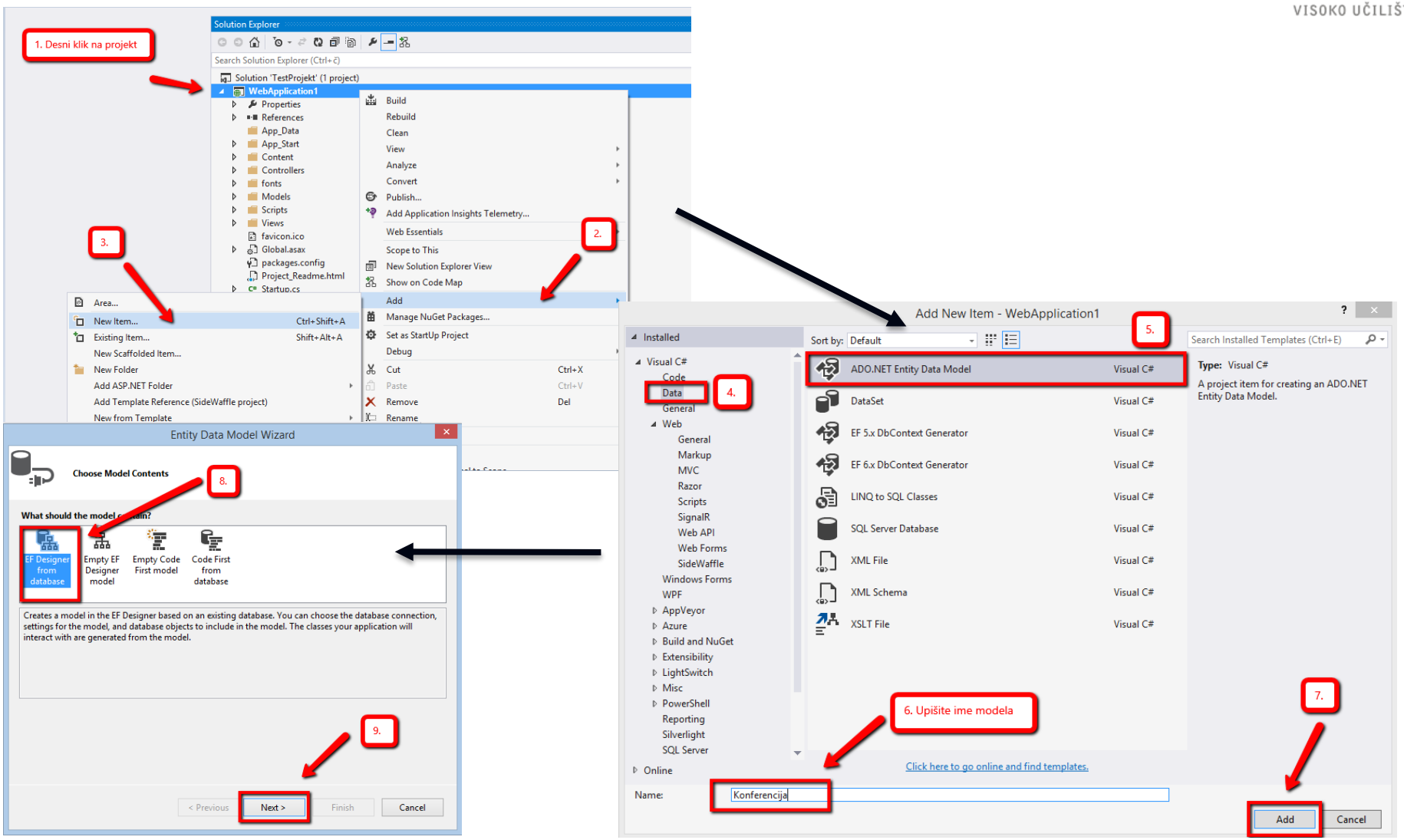
Rad kroz GUI

- Instalacija / deinstalacija se može obaviti i kroz GUI, gdje po tome da li se pored paketa vidi zelena strelica ili gumb 'Install' znamo da li je EF instaliran ili ne:



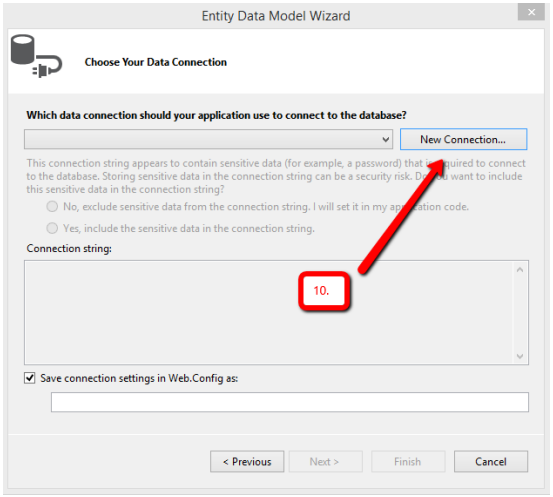
VERN'
VISOKO UČILIŠTE

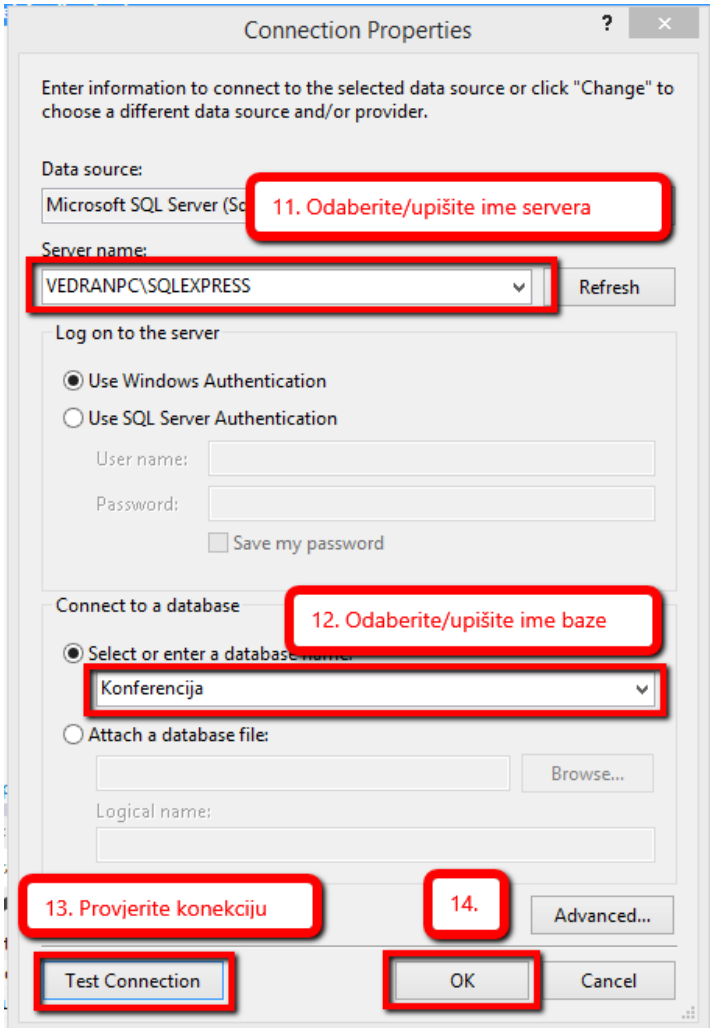
Kreiranje modela iz baze (I)

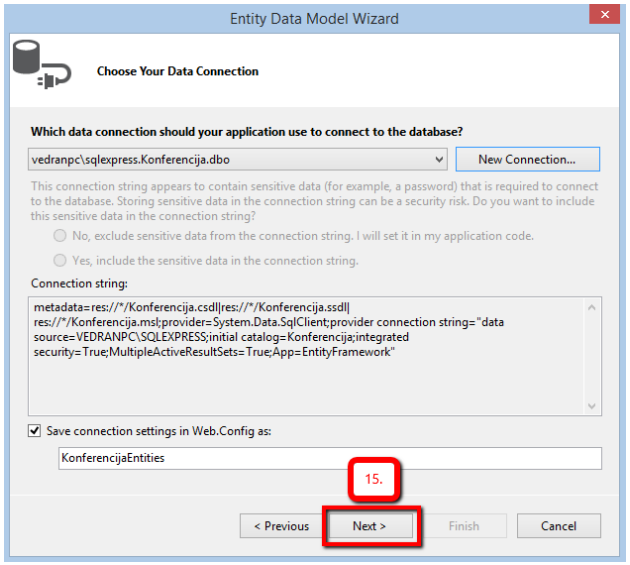


VERN'
VISOKO UČILIŠTE

Kreiranje modela iz baze (II)

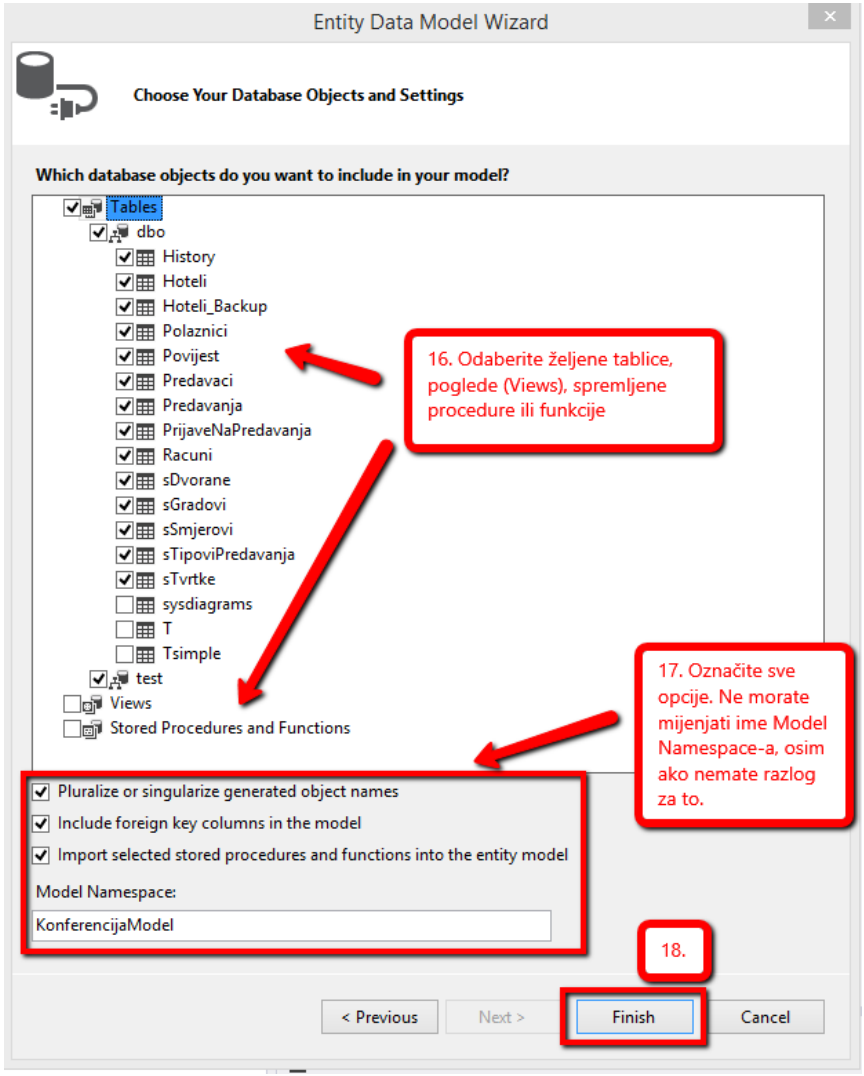


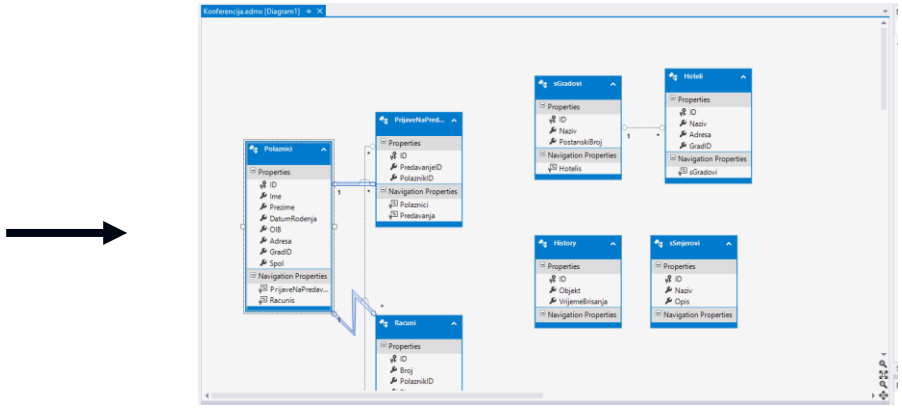




VERN'
VISOKO UČILIŠTE

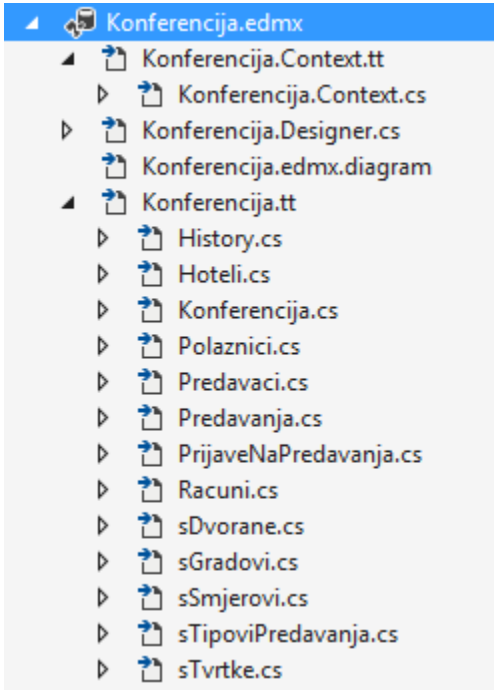
Kreiranje modela iz baze (III)





Što nam donosi model?

- Nakon što klikom na 'Save' spremimo promjene modela, generiraju se C# klase koje možemo koristiti za manipuliranje objektima i podacima unutar našeg projekta



- Entiteti EF-a su MODELI našeg MVC projekta

Validacija podataka modela

- MVC model kreiran od strane Entity Framework komponente:
  - Rukuje automatski greškama vezanim na tipove podataka
  - Regenerira se svaki put kada se projekt kompajlira.
  - Prilagođena logika će biti izgubljena zbog regeneriranja koda

... Ili drugim riječima imamo ...

- **PROBLEM**: primjenimo li validacijske atribute direktno na svojstva entiteta unutar generiranih klasa, slijedeći put kada modificiramo EF model i spremimo ga – **izgubiti ćemo sve što smo sami nadopisali** – jer će se sve klase ponovo regenerirati!

Kako onda proširiti model/entitet?

- Kako proširiti entitete?
- Koristimo: Partial Class
- Kreiramo odvojenu meta klasu u svrhu definiranja grešaka vezanih na poslovna pravila.

```
public partial class Blog
{

}
```

```
public class BlogMetadata
{
    Required(ErrorMessage="Naziv je obavezan")]
    public object Naziv { get; set; }

    Required(ErrorMessage="Opis je obavezan")]
    public object Opis { get; set; }
}
```

Primjer MetadataType klase

```
using System;
using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;
using System.Runtime.Serialization;

namespace DAL
{
    [DataContract]
    [MetadataType(typeof(TfsServerMetadata))]
    [Serializable]
    31 references | 0 authors | 0 changes
    public partial class TfsServer
    {

    }

    1 reference | 0 authors | 0 changes
    public partial class TfsServerMetadata
    {
        [Editable(false)]
        0 references | 0 authors | 0 changes
        public int Id { get; set; }

        [Required(ErrorMessage = "Name is required.")]
        [Display(Name = "Name")]
        0 references | 0 authors | 0 changes
        public string Name { get; set; }

        [Required(ErrorMessage = "Description is required.")]
        [Display(Name = "Description")]
        0 references | 0 authors | 0 changes
        public string Description { get; set; }
```