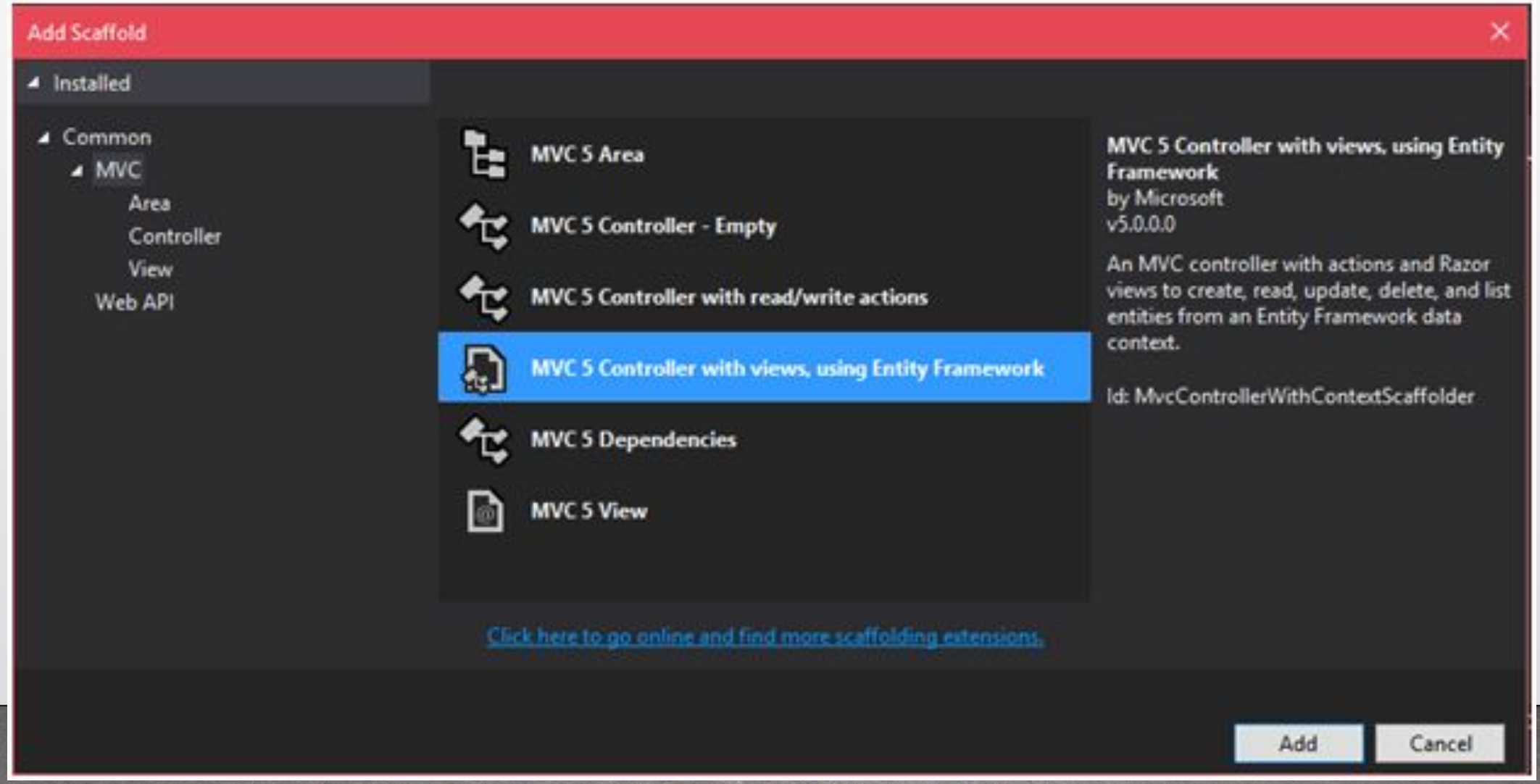
.NET MVC
Unit III

-JalpaPoriya

ASP.NET MVC Scaffolding

- It is a feature of ASP.NET that allows us to generate functional code rapidly.
- It is also known as code generator framework.
- It is pre-installed in Visual Studio 2013 and higher version.
- To create basic CRUD application, scaffolding is best choice. It reduces time amount and generate clean code

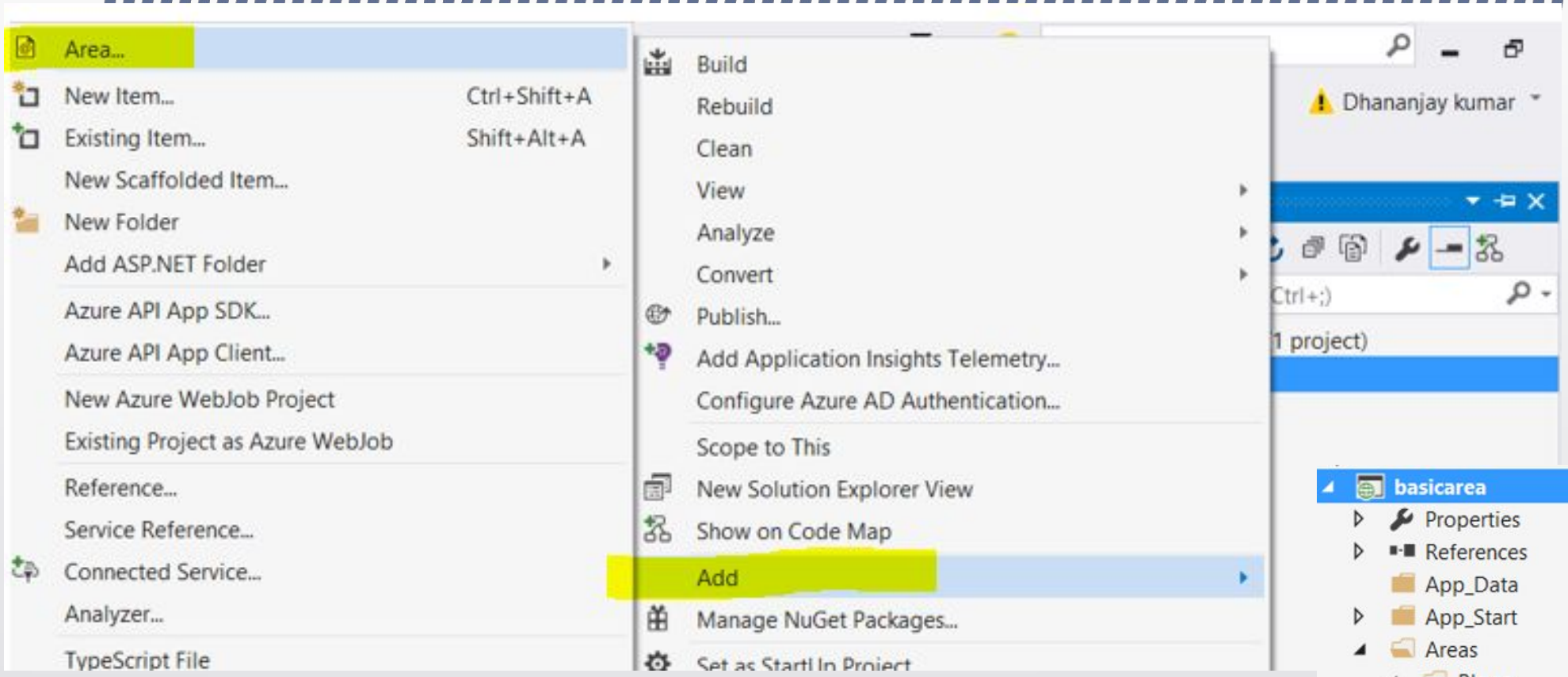
Executing the Scaffolding Template

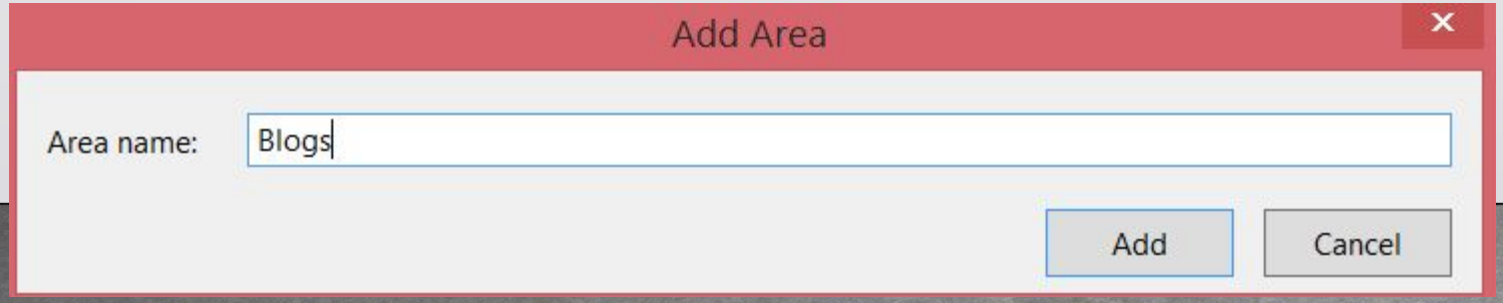


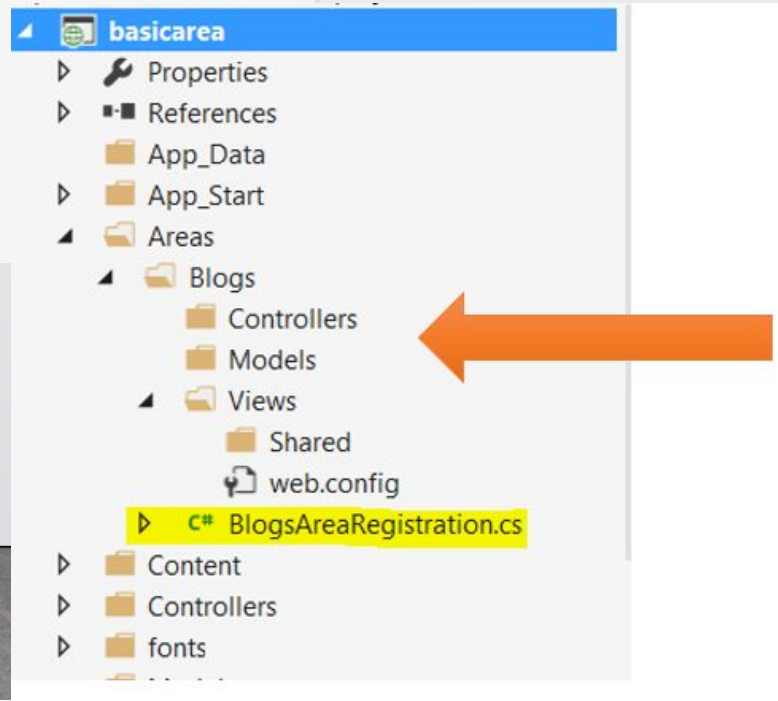
Introduction of AREA

- The main use of Areas are to physically partition web project in separate units.
- If you look into an ASP.NET MVC project, logical components like Model, Controller, and the View are kept physically in different folders, and ASP.NET MVC uses naming conventions to create the relationship between these components.

Creating Areas





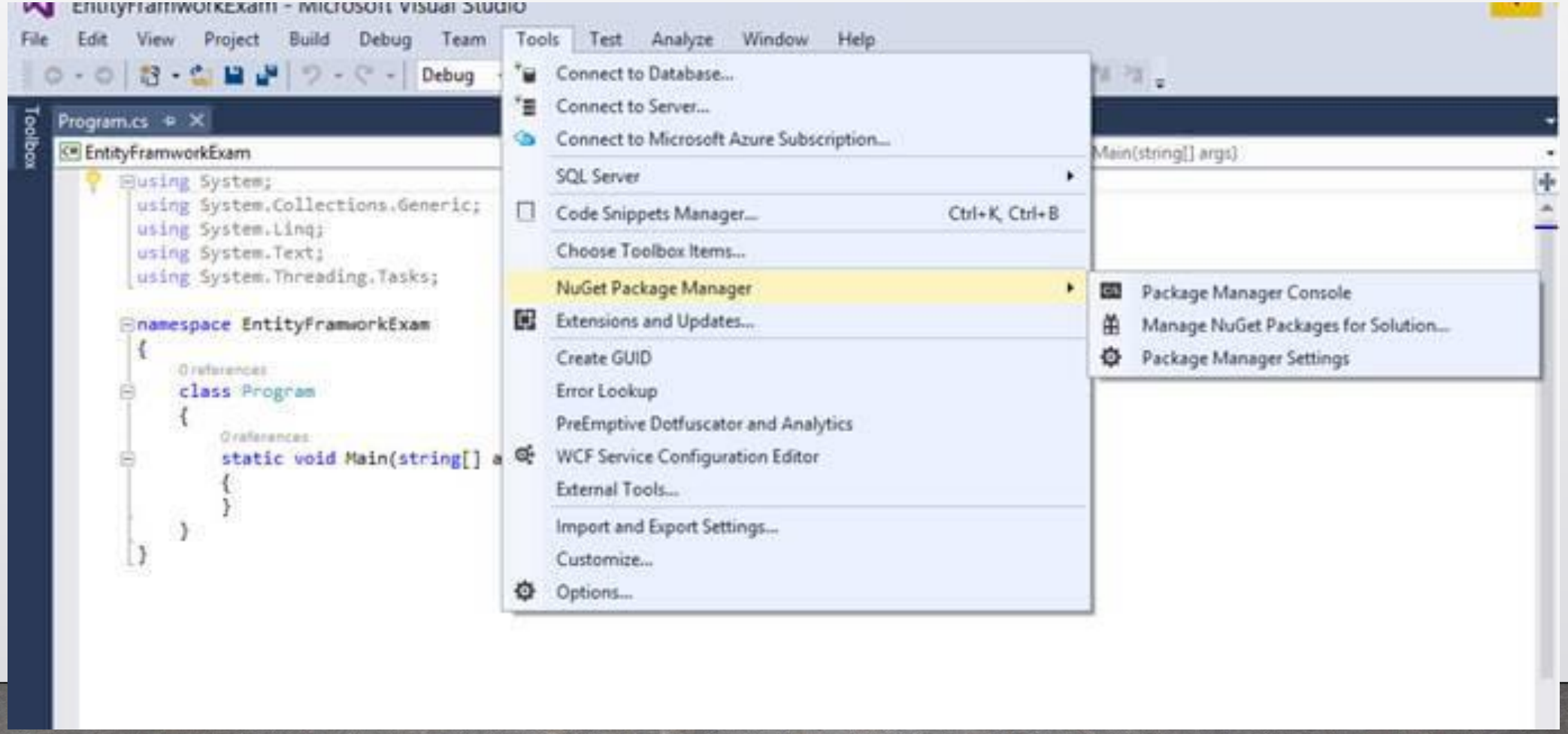


Register Areas in Global.asax File

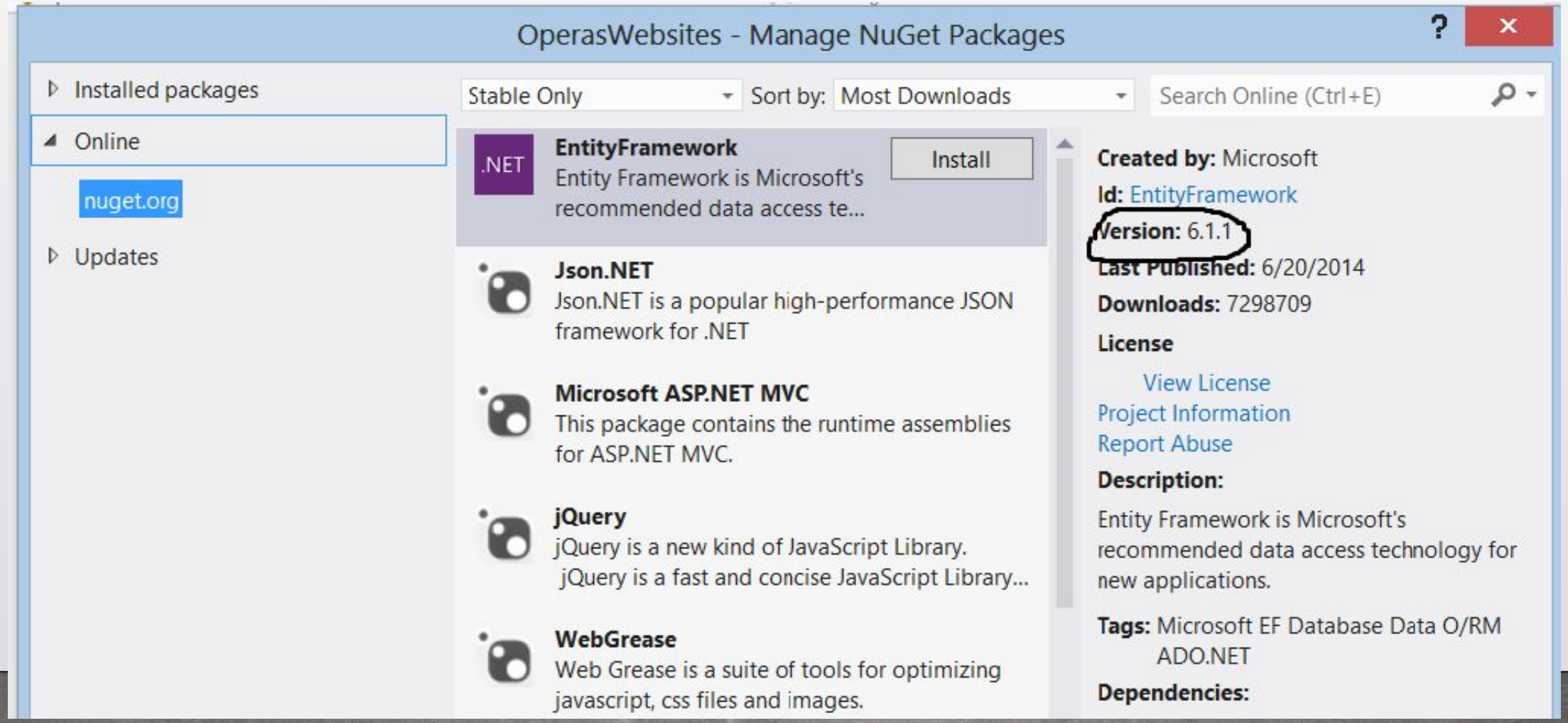
- As a last step to work with Areas, we need to verify whether the Areas are registered in the **App_Start** of the project or not.
- To do this, open global.asax and add the highlighted line of code below (if it's not there already):
- **Method Name:**
- AreaRegistration.RegisterAllArea();

Setting up Entity Framework

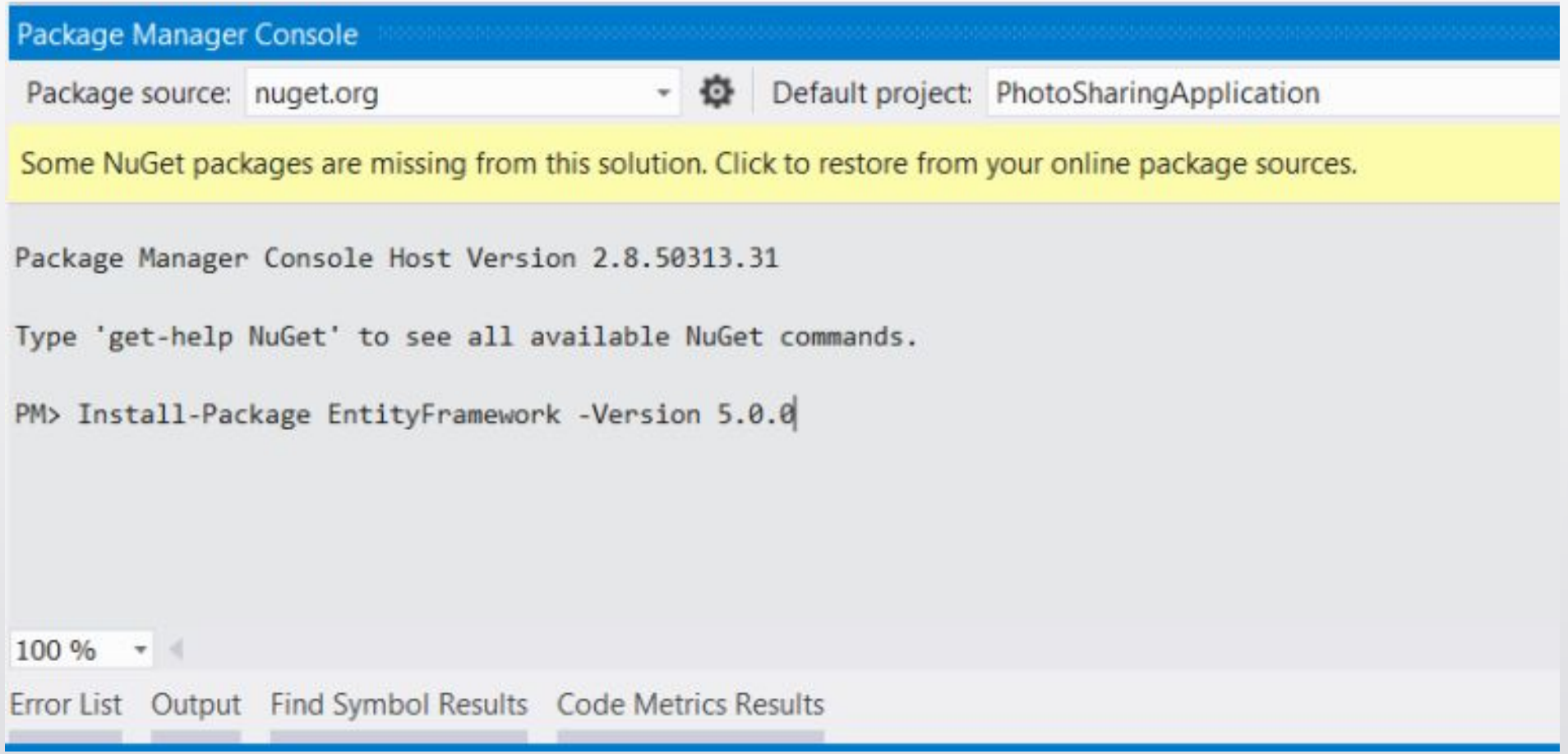
Step 1



Step 2


OperasWebsites - Manage NuGet Packages
Installed packages
Online
nuget.org
Updates
Stable Only
Sort by: Most Downloads
Search Online (Ctrl+E)
.NET
EntityFramework
Entity Framework is Microsoft's recommended data access te...
Install
Json.NET
Json.NET is a popular high-performance JSON framework for .NET
Microsoft ASP.NET MVC
This package contains the runtime assemblies for ASP.NET MVC.
jQuery
jQuery is a new kind of JavaScript Library.
jQuery is a fast and concise JavaScript Library...
WebGrease
Web Grease is a suite of tools for optimizing javascript, css files and images.
Created by: Microsoft
Id: EntityFramework
Version: 6.1.1
Last Published: 6/20/2014
Downloads: 7298709
License
View License
Project Information
Report Abuse
Description:
Entity Framework is Microsoft's recommended data access technology for new applications.
Tags: Microsoft EF Database Data O/RM ADO.NET
Dependencies:

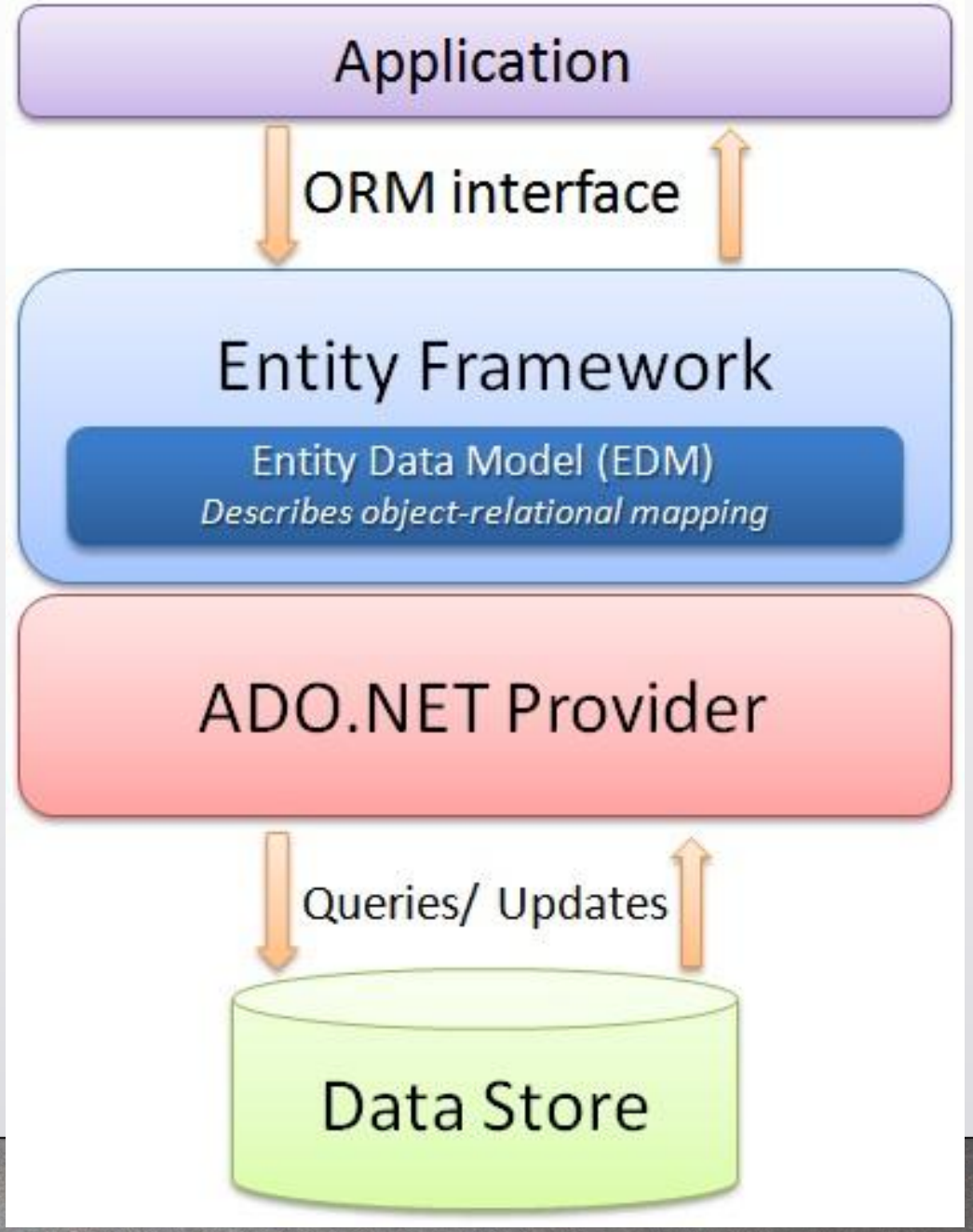
Step 3


Package Manager Console
Package source: nuget.org
Default project: PhotoSharingApplication
Some NuGet packages are missing from this solution. Click to restore from your online package sources.
Package Manager Console Host Version 2.8.50313.31
Type 'get-help NuGet' to see all available NuGet commands.
PM> Install-Package EntityFramework -Version 5.0.0
100 %
Error List
Output
Find Symbol Results
Code Metrics Results

Entity Framework

Entity Framework is an **Object Relational Mapper** (ORM). It basically generates business objects and entities according to the database tables and provides the mechanism for:

1. Performing basic CRUD operations.
2. Easily managing "1 to 1", "1 to many", and "many to many" relationships.
3. Ability to have inheritance relationships between entities.

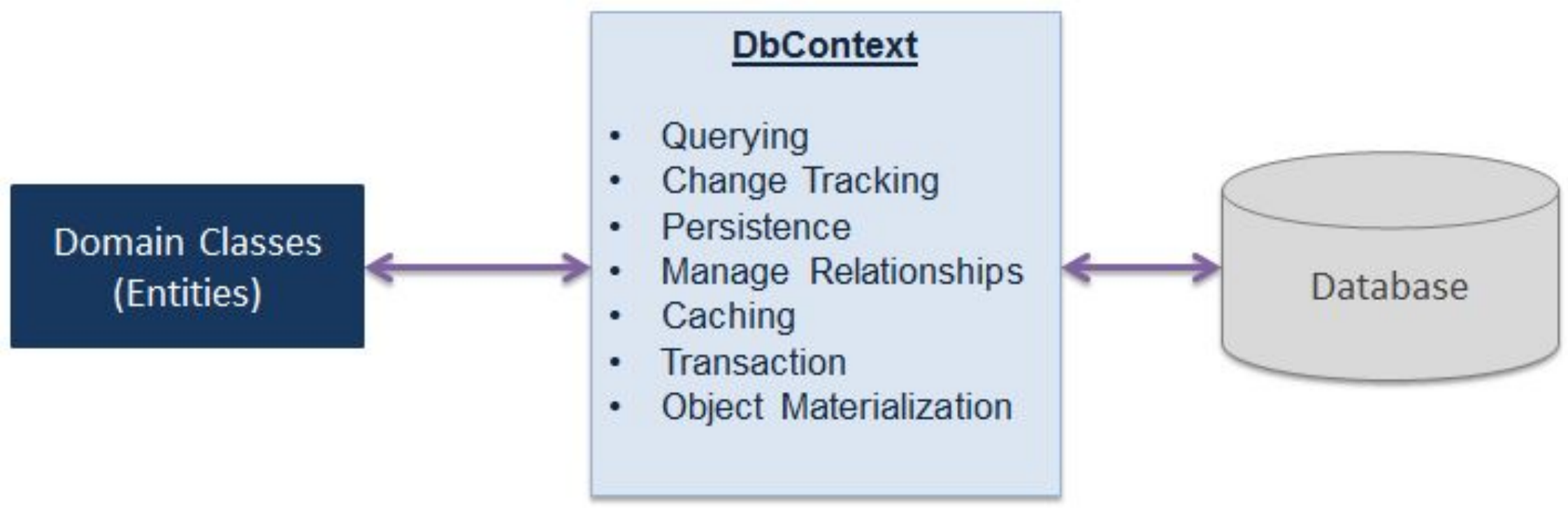


Features of ORM in entity framework

1. Map our database types to our code types
2. Avoid repetitive data access code
3. Access code automatically based on the data model class
4. Support a clean separation of concerns and independent development that allows parallel, simultaneous development of application
5. Easily reuse the data object
6. Application Maintainability

DbContext Class

DbContext is an important class in Entity Framework API. It is a bridge between your domain or entity classes and the database.



- **Querying:** Converts LINQ-to-Entities queries to SQL query and sends them to the database.
- **Change Tracking:** Keeps track of changes that occurred on the entities after querying from the database.
- **Persisting Data:** Performs the Insert, Update and Delete operations to the database, based on entity states.
- **Caching:** Provides first level caching by default. It stores the entities which have been retrieved during the life time of a context class.
- **Manage Relationship:** Manages relationships using API configurations in Code-First approach.
- **Object Materialization:** Converts raw data from the database into entity objects.

DbSet in Entity Framework 6

- The DbSet class represents an entity set that can be used for create, read, update, and delete operations.
- The context class (**derived from DbContext**) must include the DbSet type properties for the entities which map to database tables and views.

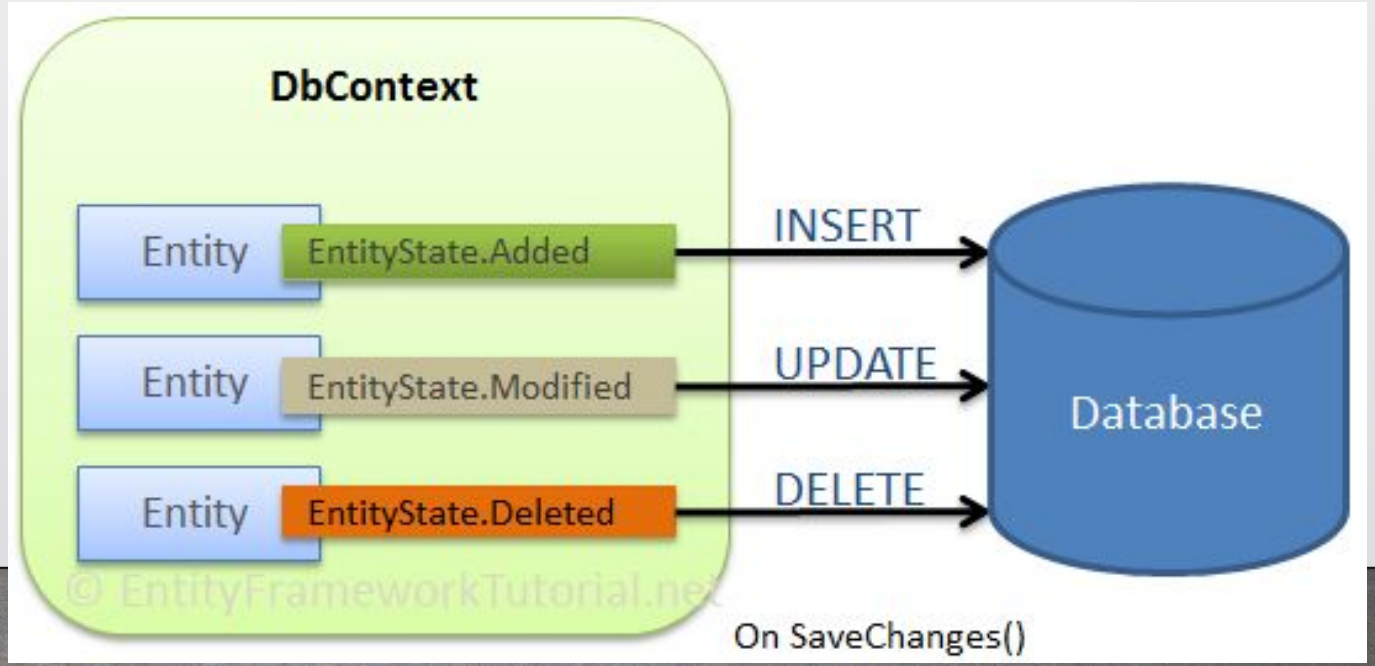
Example of DBSet

```
}

public virtual DbSet<Course> Courses { get; set; }
public virtual DbSet<Standard> Standards { get; set; }
public virtual DbSet<Student> Students { get; set; }
public virtual DbSet<StudentAddress> StudentAddresses { get; set; }
public virtual DbSet<Teacher> Teachers { get; set; }
public virtual DbSet<View_StudentCourse> View_StudentCourse { get; set; }
```

Method

- **SaveChanges**
  - Executes INSERT, UPDATE and DELETE commands to the database for the entities with Added, Modified and Deleted state.



- **Insert Data**
- Use the **DbSet.Add** method to add a new entity to a context (instance of DbContext), which will insert a new record in the database when you call the SaveChanges() method.
- Example: dbcontext.Students.Add(studentEntity)

- **Remove**
- Marks the given entity as Deleted. When the changes are saved, the entity is deleted from the database. The entity must exist in the context in some other state before this method is called.
- Example: dbcontext.Students.Remove(studentEntity);

- **Find(int)**
- Uses the primary key value to find an entity tracked by the context.
- If the entity is not in the context, then a query will be executed and evaluated against the data in the data source, and null is returned
- If the entity is not found in the context or in the data source. Returns entities that have been added to the context but have not yet been saved to the database.
- Example:
- Student studEntity = dbcontext.Students.Find(1);

Form Methods

HTTP provides methods for the action performed on a resource.

HTTP providers following mail verbs:

GET [HttpGet]

HttpGet data travels in URL only

http://localhost:111/Home/Display/1

http://localhost:111/Home/Display?StudId=1

POST [HttpPost]

HttpPost used to while we have to create new resource.

Data travers from Body (JSon), Header

PUT [HttpPut]

HttpPut is used while we have to update existing resource.

Data travers by URL or body

http://localhost:111/Home/AddRecord/1

Body - JSon

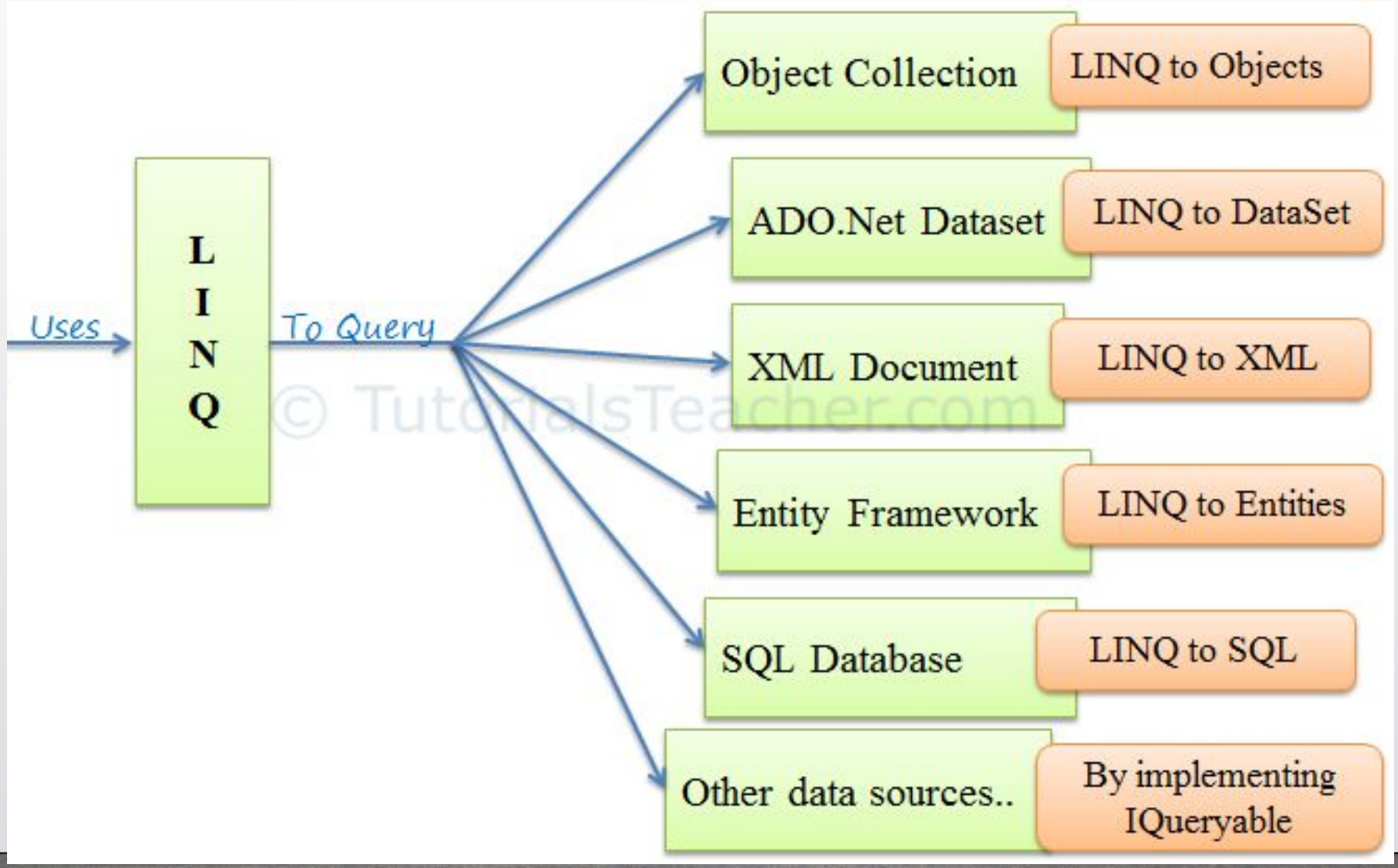
DELETE [HttpDelete]

HttpDelete used to delete existing resource.

http://localhost:111/Home/DeleteRecord/1

LINQ for MVC

- LINQ (Language Integrated Query) is uniform query syntax in C# and VB.NET to retrieve data from different sources and formats.
- It is integrated in C# or VB, thereby eliminating the mismatch between programming languages and databases, as well as providing a single querying interface for different types of data sources.


Uses
LINQ
To Query
Object Collection
LINQ to Objects
ADO.Net Dataset
LINQ to DataSet
XML Document
LINQ to XML
Entity Framework
LINQ to Entities
SQL Database
LINQ to SQL
Other data sources..
By implementing IQueryable

- LINQ queries return results as objects. It enables you to u
  object-oriented approach on the result set and not to w
  transforming different formats of results into objects.

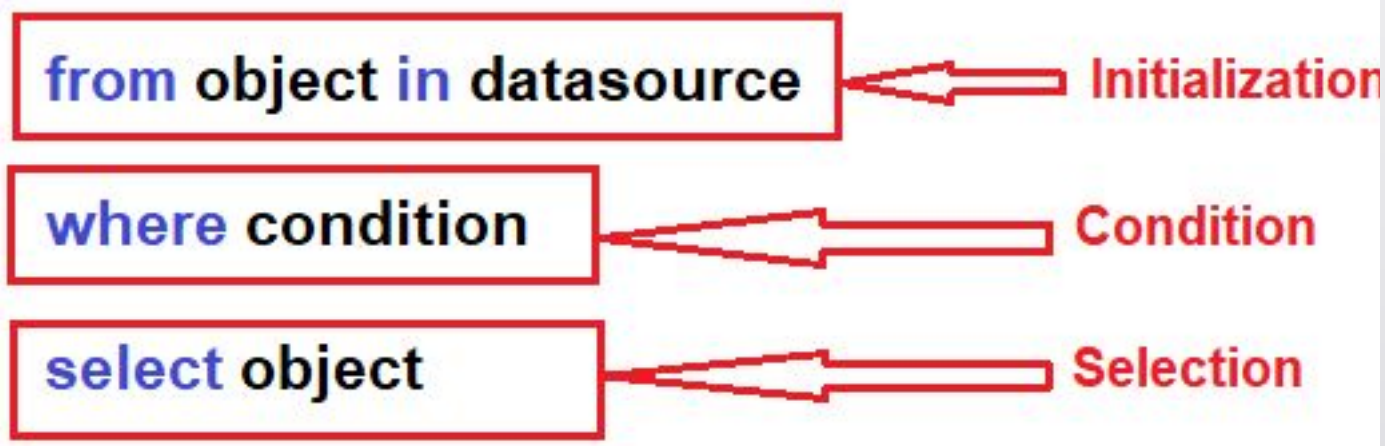
LINQ Query Syntax

- There are two basic ways to write a LINQ query to IEnumerable collection or IQueryable data sources.
- LINQ can be created with

1. Query Syntax
2. Method Syntax

LINQ Query Syntax:

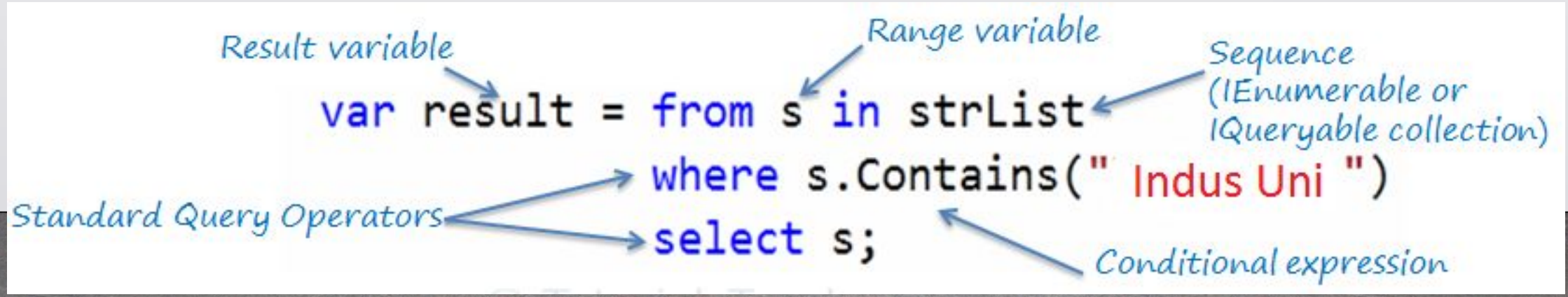
from **<range variable>** in **<Collection>**
<Standard Query Operators> <lambda expression>
<select or groupBy operator> <result formation>



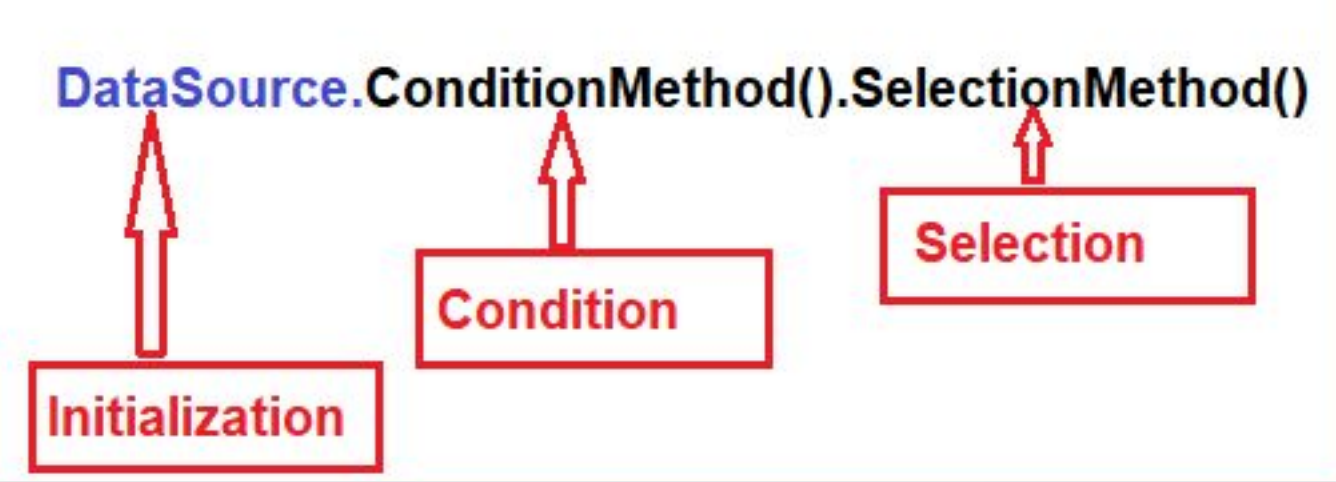
Example:

```
var teenAgerStudent = from s in studentList
                      where s.Age > 12 && s.Age < 20
                      select s;
```

```
var result = from s in stringList
             where s.Contains("Indus Uni")
             select s;
```



LINQ Method




```
var result = strList.Where(s => s.Contains("Tutorials"));
```


Extension method

Lambda expression

Standard Query Operators

Name	Example
Where	Query: from s in studentList **where** s.Age > 12 && s.Age < 20 select s.StudentName;
Where	Method: studentList.Where(s => s.Age > 12 && s.Age < 20);

orderby	Query: from s in studentList orderby s.StudentName ascending select s; from s in studentList orderby s.StudentName descending select s;

Orderby	Method:
	studentList.OrderBy(s => s.StudentName);
	studentList.OrderByDescending(s => s.StudentName);

ThenBy	Methods:
	studentList.OrderBy(s => s.StudentName).ThenBy(s => s.Age);
	studentList.OrderBy(s => s.StudentName).ThenByDescending(s => s.Age);

Group	Query: from s in studentList group s by s.Age; Method: studentList.GroupBy(s => s.Age);
Contains	Method: studentList.Contains(std); studentList.Name.Contains(“abc”
Average	studentList.Average(s => s.Age);
Count	studentList.Count(); studentList.Count(s => s.Age >= 18);
Max	studentList.Max(s => s.Age);
Sum	studentList.Sum(s => s.Age)

References

Web Site:
https://docs.microsoft.com/en-us/aspnet/mvc/
https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm
https://www.guru99.com/mvc-tutorial.html
https://en.wikipedia.org/wiki/ModelViewController
https://www.guru99.com/mvc-tutorial.html
https://www.geeksforgeeks.org/mvc-design-pattern/

Book:
Pro ASP.NET MVC 5.0