**What is Repository Pattern in ASP.NET MVC?**

Repository pattern is usefult for decoupling entity operations form presentation,

which allows easy mocking and unit testing.

“The Repository will delegate to the appropriate infrastructure services to get the job

done. Encapsulating in the mechanisms of storage, retrieval and query is the most

basic feature of a Repository implementation”

“Most common queries should also be hard coded to the Repositories as methods.”

**New C# 3.0 Language Additions**

To make LINQ seamlessly integrate with C#, significant enhancements were needed for the C# language.

Virtually every significant enhancement to the C# language made in version 3.0 was made specifically

to support LINQ. While all of these features have merit on their own, it is really the sum of the parts

contributing to LINQ that makes C# 3.0 so noteworthy.

To truly understand much of the syntax of LINQ, it is necessary for me to cover some of the new

C# 3.0 language features before proceeding with the workings of the components of LINQ.

language additions:

• Lambda expressions

• Expression trees

• The keyword var, object and collection initialization, and anonymous types

• Extension methods

• Partial methods

• Query expressions

<https://www.codeproject.com/Articles/199060/Introducing-LINQ-Language-Integrated-Query>

### LINQ providers

LINQ providers are set of classes that take an LINQ query which generates method that executes an equivalent query against a particular data source.

The C#3.0 specification defines a Query Expression Pattern along with translation rules from a LINQ expression to an expression in a subset of C# 3.0 without LINQ expressions. The translation thus defined is actually un-typed, which, in addition to lambda expressions being interpretable as either delegates or expression trees, allows for a great degree of flexibility for libraries wishing to expose parts of their interface as LINQ expression clauses. For example, **LINQ to Objects** works on IEnumerable<T>s and with delegates, whereas **LINQ to SQL** makes use of the expression trees.

The **expression trees** are at the core of the LINQ extensibility mechanism, by which LINQ can be adapted for many data sources. The expression trees are handed over to LINQ Providers, which are data source-specific implementations that adapt the LINQ queries to be used with the data source. If they choose so, the LINQ Providers analyze the expression trees contained in a query in order to generate essential pieces needed for the execution of a query. This can be SQL fragments or any other completely different representation of code as further manipulatable data. LINQ comes with LINQ Providers for in-memory object collections, [Microsoft SQL Server](https://en.wikipedia.org/wiki/Microsoft_SQL_Server) databases, [ADO.NET](https://en.wikipedia.org/wiki/ADO.NET) datasets and XML documents. These different providers define the different flavors of LINQ:

#### LINQ to Objects

The LINQ to Objects provider is used for in-memory collections, using the local query execution engine of LINQ. The code generated by this provider refers to the implementation of the standard query operators as defined on the Sequence pattern and allows IEnumerable<T> collections to be queried locally. Current implementation of LINQ to Objects perform interface implementation checks to allow for fast membership tests, counts, and indexed lookup operations when they are supported by the runtime type of the IEnumerable.[[7]](https://en.wikipedia.org/wiki/Language_Integrated_Query#cite_note-Enumerable.ElementAt-7)[[8]](https://en.wikipedia.org/wiki/Language_Integrated_Query#cite_note-Enumerable.Contains-8)[[9]](https://en.wikipedia.org/wiki/Language_Integrated_Query#cite_note-Enumerable.Count-9)

#### LINQ to XML (formerly called XLINQ)

The LINQ to XML provider converts an XML document to a collection of XElement objects, which are then queried against using the local execution engine that is provided as a part of the implementation of the standard query operator.[[10]](https://en.wikipedia.org/wiki/Language_Integrated_Query#cite_note-10)

#### LINQ to SQL (formerly called DLINQ)

The LINQ to SQL provider allows LINQ to be used to query [Microsoft SQL Server](https://en.wikipedia.org/wiki/Microsoft_SQL_Server) databases, including [SQL Server Compact](https://en.wikipedia.org/wiki/SQL_Server_Compact) databases. Since SQL Server data may reside on a remote server, and because SQL Server has its own query engine, LINQ to SQL does not use the query engine of LINQ. Instead, it converts a LINQ query to a [SQL](https://en.wikipedia.org/wiki/SQL) query that is then sent to SQL Server for processing.[[11]](https://en.wikipedia.org/wiki/Language_Integrated_Query#cite_note-11) However, since SQL Server stores the data as [relational data](https://en.wikipedia.org/wiki/Relational_database) and LINQ works with data encapsulated in objects, the two representations must be [mapped](https://en.wikipedia.org/wiki/Object-Relational_mapping) to one another. For this reason, LINQ to SQL also defines a mapping framework. The mapping is done by defining classes that correspond to the tables in the database, and containing all or a subset of the columns in the table as data members.[[12]](https://en.wikipedia.org/wiki/Language_Integrated_Query#cite_note-ltos-12) The correspondence, along with other [relational model](https://en.wikipedia.org/wiki/Relational_model) attributes such as [primary keys](https://en.wikipedia.org/wiki/Primary_key), are specified using LINQ to SQL-defined [attributes](https://en.wikipedia.org/wiki/Attribute_(computing)). For example,

[Table(Name="Customers")]

**public** **class** **Customer**

{

[Column(IsPrimaryKey = true)]

**public** int CustID;

[Column]

**public** string CustName;

}

This class definition maps to a table named Customers and the two data members correspond to two columns. The classes must be defined before LINQ to SQL can be used. [Visual Studio 2008](https://en.wikipedia.org/wiki/Visual_Studio_2008) includes a mapping designer that can be used to create the mapping between the data schemas in the object as well as the relational domain. It can automatically create the corresponding classes from a [database schema](https://en.wikipedia.org/wiki/Database_schema), as well as allow manual editing to create a different view by using only a subset of the tables or columns in a table.[[12]](https://en.wikipedia.org/wiki/Language_Integrated_Query#cite_note-ltos-12)

The mapping is implemented by the DataContext that takes a connection string to the server, and can be used to generate a Table<T> where T is the type to which the database table will be mapped. The Table<T> encapsulates the data in the table, and implements the IQueryable<T> interface, so that the expression tree is created, which the LINQ to SQL provider handles. It converts the query into [T-SQL](https://en.wikipedia.org/wiki/T-SQL) and retrieves the result set from the database server. Since the processing happens at the database server, local methods, which are not defined as a part of the lambda expressions representing the predicates, cannot be used. However, it can use the [stored procedures](https://en.wikipedia.org/wiki/Stored_procedure) on the server. Any changes to the result set are tracked and can be submitted back to the database server.[[12]](https://en.wikipedia.org/wiki/Language_Integrated_Query#cite_note-ltos-12)

#### LINQ to DataSets

Since the LINQ to SQL provider (above) works only with [Microsoft SQL Server](https://en.wikipedia.org/wiki/Microsoft_SQL_Server) databases, in order to support any generic database, LINQ also includes the LINQ to DataSets. It uses ADO.NET to handle the communication with the database. Once the data is in ADO.NET Datasets, LINQ to DataSets execute queries against these datasets.[[13]](https://en.wikipedia.org/wiki/Language_Integrated_Query#cite_note-13)

**lambda expressions in LINQ?**

Lambda expression is referred as a unique function use to form delegates or expression tree types, where right side is the output and left side is the input to the method. For writing LINQ queries particularly, Lambda expression is used.

<https://www.guru99.com/comparison-between-web-services.html>

Async Await

## Task and Task<T>

Tasks are constructs used to implement what is known as the [Promise Model of Concurrency](https://en.wikipedia.org/wiki/Futures_and_promises). In short, they offer you a "promise" that work will be completed at a later point, letting you coordinate with the promise with a clean API.

* Task represents a single operation which does not return a value.
* Task<T> represents a single operation which returns a value of type T.

It’s important to reason about tasks as abstractions of work happening asynchronously, and not an abstraction over threading. By default, tasks execute on the current thread and delegate work to the Operating System, as appropriate. Optionally, tasks can be explicitly requested to run on a separate thread via the Task.Run API.

Tasks expose an API protocol for monitoring, waiting upon and accessing the result value (in the case of Task<T>) of a task.

**Async improves responsiveness**

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> RegisterAccount(RegisterViewModel viewModel)

{

await this.UserManager.AddToRoleAsync(user.Id, userRole);

await UserManager.ConfirmEmailAsync(user.Id, code);

await new MyMailer().ConfirmMail(userDetail.Email, userDetail.Name, "Confirm your account", callbackUrl, viewModel.Password).SendAsync();

}

public Task<string> GetHtmlAsync()

{

// Execution is synchronous here

var client = new HttpClient();

return client.GetStringAsync("http://www.dotnetfoundation.org");

}

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/>

<https://msdn.microsoft.com/en-us/magazine/dn802603.aspx?f=255&MSPPError=-2147217396>

[https://www.youtube.com/watch?v=drgsIO5M8ok](https://www.youtube.com/watch?v=drgsIO5M8ok&t=20s)

**Language Features**

<https://www.guru99.com/c-sharp-dot-net-version-history.html>

[**http://www.dotnetcurry.com/csharp/1411/csharp-favorite-features**](http://www.dotnetcurry.com/csharp/1411/csharp-favorite-features)

<https://www.codeproject.com/Articles/327916/C-Language-Features-From-C-to>

<https://www.codeproject.com/Articles/846566/What-s-New-in-Csharp-String-Interpolation>

<https://www.dotnettricks.com/learn/entityframework/difference-between-linq-to-sql-and-entity-framework>

<https://csharp.today/c-6-features-auto-property-initializers/>

<https://www.codeproject.com/Tips/900017/Null-Propagation-Operator-A-New-Feature-of-Csharp>

<https://www.c-sharpcorner.com/UploadFile/7ca517/exception-filter-in-C-Sharp-6-0/>

<https://www.codeproject.com/Articles/832189/List-vs-IEnumerable-vs-IQueryable-vs-ICollection-v>

<https://career.guru99.com/top-22-linq-interview-questions/>

**What is Lambda expressions , Action , Func and Predicate**

<https://www.youtube.com/watch?v=8o0O-vBS8W0&t=2s>

https://www.youtube.com/watch?v=1Q4I63-hKcY&t=670s

OOP

**WebAPI**

<https://www.codeproject.com/Articles/769140/Top-ASP-NET-Web-API-Interview-Questions>

<http://www.dotnetanalyst.com/FAQs/WebAPI>

HTML 5

<https://www.w3schools.com/html/html_elements.asp>

<https://www.w3schools.com/html/html5_intro.asp>

<https://www.w3schools.com/jquery/jquery_selectors.asp>

**Entity Framework**

DB First vs Code First

1. **Deployment Scripts** :If we have different environment then we need not to run db script for each and every environment at the time of deployment

2. **Better Versioning**: By (tfs or github or svn ) version control in db first is difficult to track any db Changes done by which team member as Whole edmx updated by single person at once.

(coz' versioning tool will show changes done by team members names who update edmx file ), but in code first we track who changes which table

3. **ViewModel not mandatory**: For validation if we want to use **data annotation** or we have some **derived properties** then we must have to use view Model in DB First as any changes to tt class will be reset whenever EDMX is updated.

\* (For Derived Properties in DB First we can use partial classes instead)

4: Data Model Schema might be different from Database Schema if **CodeFirst Entities created On Existing Database** by **Entity Framework Tools (i.e if a db is being used for more than 1 applications, Columns Nullability or length might be different from db in different projects. By changing Dataannotation and skipping Migration Script.)**

<https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/workflows/existing-database>

<https://channel9.msdn.com/blogs/ef/code-first-to-existing-database-ef6-1-onwards->