9: Yield keyword

We can use this collections and generics for IEnumerables.

11. about Delegates

Delegate is a type which holds the method(s) reference in an object. It is also referred to as a type safe function pointer.

* Delegates are reference types
* Delegates are type safe
* Delegate in C# is similar to a function pointer in C or C++
* A delegate is a class that encapsulates a method signature. Although it can be used in any context, it often serves as the basis for the event-handling model in C# but can be used in a context removed from event handling (e.g. passing a method to a method through a delegate parameter).

Incorrect about delegates:

* Only one method can be called using delegate
* Delegates are object oriented.

14. Method to be used for adding functionality for existing classes

* Containment (OOP concept)
* Inheritance (OOP concept)
* Extension Methods (C# 3.0 concept)

**MVC**

16: Order of execution multiple filters in MVC

* Authorization filter
* Action filter
* Result filter (Response)
* Exception filter

17: Role of routing in MVC framework.

* Routing maps an URL to physical file or controller Class
* Route Reg must be done RouteConfig class under App\_Start Folder.
* We are able restrict allowed values for specific paremeter
* A Route Contains a URL pattern is after the domain name

18: Partial View

* Reusable view
* You can use the partial view in the layout view, as well as other content views.

19: Rest API’s built with ASP.NET Web API.

20: ASP.NET Web API Framework

21: Transactions on Entity framework 6

**DbContext.Database.BeginTransaction**

This method returns a DbContextTransaction object. The BeginTransaction method has two overloads, one has no argument and the other accepts an explicit Isolation Level.

22: db.SaveChanges()

23: Lazy and Eager loading on Entity framework

Entity Framework defaults to **lazy** loading and allows you to override it when necessary.

That is, I think, a good thing -- but not enough of a good thing to qualify as the "best." Fundamentally, lazy loading means that the child objects at the end of a navigation property aren't retrieved unless you explicitly work with the navigation property in your code. If you don't need those child objects, then this is the behavior that you want because it reduces the amount of data retrieved from the database when you retrieve the parent object.

24: Approaches available in Entity frameworks.

* Model First Approach:
* Object First Approach:
* Database First Approach:

25: Syntax for Code First Approach:

public class Blog

{

public int BlogId { get; set; }

public string Name { get; set; }

public virtual List<Post> Posts { get; set; }

}

To save details for blog name into blog table.

using (var db = new BloggingContext())

{

// Create and save a new Blog

Console.Write("Enter a name for a new Blog: ");

var name = Console.ReadLine();

var blog = new Blog { Name = name };

db.Blogs.Add(blog);

db.SaveChanges();

}