**jana**

* [Logout](http://janagaraj.in/AccountLog/LogOff)

Interview question and answers Add New

# 2.Advantage of MVC?

Advantage:

--Separation of concern. Separation of concern means we divide the application Model, Control   and View.

--We can easily maintain our application because of separation of concern.

--In the same time we can split many developers work at a time. It  will not affects  one developer work to another developer work.

--It supports TTD (test-driven development). We can create an application with unit test. We can write won test case.

--Latest version of MVC Support default responsive web site and mobile templates.

--We can create own view engine. It is syntax is very easy compare to traditional view engine.

Disadvantage:

--Cannot see design page preview like .aspx page. Every time want to run then see the design.

--Understanding flow of application is very hard one. It is little bit of complex to implement and not suitable for small level applications.

--It's deployment is little bit hard one.

  Edit  Delete

# 3.MVC lifecycle?

-->User Request-->Routing-->(MVC handler)-->Controller initialization -->ActionExecution-->View Result-->(View engine)--> View -->Response

  Edit  Delete

# 4.what is routing and types?

In .NET technology, we have seen lot of different ways to hit some specific resource or URI. In ASP.NET, we use Url Rewriting where as in ASP.NET MVC, there are two more options, first one is Convention Based Routing and other one is Attribute Routing.

1. **Convention based routing** - to define this type of routing, we call MapRoute method and set its unique name, url pattern and specify some default values.
2. **Attribute based routing** - to define this type of routing, we specify the Route attribute in the action method of the controller.
3. routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");
4. routes.IgnoreRoute("favicon.ico");
5. routes.IgnoreRoute("Content/img/dotnet-tutorial.ico");
6. routes.MapRoute("LogOff", "logOff", **new**
7. {
8. controller = "Account", action = "logoff"
9. });

**Routes:** It is nothing but only a table which is a collection of routes defined in route table. When someone hits some url in the browser, application first check the existing routes table and match with routes.  
**IgnoreRoute:**It is also a collection of url that should be ignored by application.  
**MapRoute:**It is used to add new route into the route table.

table.  
  
  
  
Attribute Routing  
ASP.NET MVC also supports new type of routing that is called Attribute Routing. In this, attribute participate to route url to particular action.  
**To enable attribute routing, we need to use MapMvcAttributeRoutes() in the RouteConfig**

1. **public** **class** RouteConfig
2. {
3. **public** **static** **void** RegisterRoutes(RouteCollection routes)
4. {
5. routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");
6. // Attribute Routing
7. routes.MapMvcAttributeRoutes();
8. // Convention Based Routing
9. routes.MapRoute("LogOff", "logOff", **new**
10. {

https://www.c-sharpcorner.com/UploadFile/8a67c0/choose-attribute-routing-instead-of-convention-based-routing/

  Edit  Delete

# 5.MVC Filters

|  |  |  |
| --- | --- | --- |
| Filter Type | Interface | Description |
| Authentication | IAuthenticationFilter | These are Runs, before any other filters or the action method. |
| Authorization | IAuthorizationFilter | These Runs first, before any other filters or the action method. |
| Action | IActionFilter | These Runs before and after the action method. |
| Result | IResultFilter | Runs before and after the action result is executed. |
| Exception | IExceptionFilter | Runs only if another filter, the action method, or the action resultthrows an exception. |

There are five types of Filters in ASP.NET MVC 5:

1. **Authentication Filters**  
   Authentication filter runs before any other filter or action method. Authentication confirms that you are a valid or invalid user. Action filters implements the IAuthenticationFilter interface.
2. **Authorization Filters**  
   The AuthorizeAttribute and RequireHttpsAttribute are the examples of Authorization Filters. Authorization Filters are responsible for checking User Access; these implement the IAuthorizationFilterinterface in the framework. These filters used to implement authentication and authorization for controller actions. For example, the Authorize filter is an example of an Authorization filter.
3. **Action Filters**   
     
   Action Filter is an attribute that you can apply to a controller action or an entire controller. This filter will be called before and after the action starts executing and after the action has executed.  
     
   Action filters implement the IActionFilter interface that have two methods OnActionExecuting andOnActionExecuted. OnActionExecuting runs before the Action and gives an opportunity to cancel the Action call. These filters contain logic that is executed before and after a controller action executes, you can use an action filter, for instance, to modify the view data that a controller action returns.
4. **Result Filters**  
   The OutputCacheAttribute class is an example of Result Filters. These implement the IResultFilter interface which like the IActionFilter has OnResultExecuting and OnResultExecuted. These filters contains logic that is executed before and after a view result is executed. Like if you want to modify a view result right before the view is rendered to the browser.
5. **ExceptionFilters**  
   The HandleErrorAttribute class is an example of ExceptionFilters. These implement the IExceptionFilter interface and they execute if there are any unhandled exceptions thrown during the execution pipeline. These filters can be used as an exception filter to handle errors raised by either your controller actions or controller action results.

You can override the methods in your controller class if you want.

https://www.c-sharpcorner.com/article/filters-in-Asp-Net-mvc-5-0-part-twelve/

  Edit  Delete

# what is Action Result?

| **Result Class** | **Description** | **Base Controller Method** |
| --- | --- | --- |
| ViewResult | Represents HTML and markup. | View() |
| EmptyResult | Represents No response. |  |
| ContentResult | Represents string literal. | Content() |
| FileContentResult,FileResult, FilePathResult, FileStreamResult | Represents the content of a file | File() |
| JavaScriptResult | Represent a JavaScript script. | JavaScript() |
| JsonResult | Represent JSON that can be used in AJAX | Json() |
| RedirectResult | Represents a redirection to a new URL | Redirect() |
| RedirectToRouteResult | Represent another action of same or other controller | RedirectToRoute() |
| PartialViewResult | Returns HTML | PartialView() |
| HttpUnauthorizedResult | Returns HTTP 403 status |  |

  Edit  Delete

**View Bag vs View Data vs Temp Data**

# Comparison: ViewModel vs ViewData vs ViewBag vs TempData vs Session

| # | ViewModel | ViewData | ViewBag | TempData | Session |
| --- | --- | --- | --- | --- | --- |
| 1 | Is a class. It is a model specific for rendering a view. | Is a key-value dictionary derived from ViewDataDictionary. | Is a Dynamic property. It is a wrapper around ViewData. | Is a key-value dictionary derived from TempDataDictionary. | Is a key-value dictionary derived from TempDataDictionary. |
| 2 | Strongly typed class. So, no need for type-casting. | Un-typed. So, needs type-casting for complex data. | Type casting is not required. | Un-typed: Needs type-casting for complex data type. | Un-typed: Needs type-casting and null checking. |
| 3 | Represents only the data from a model required for rendering the view. | Used to pass data between controller and view. | Used to pass data between controller and view. | Used to pass data between requests. I.e. it helps to pass data from one controller to another controller. | Used to store a small amount of data for the duration of the user visiting the website. |
| 4 | The lifespan is only for the current request. | The lifespan is only for the current request. | The lifespan is only for the current request. | Lifespan is for the current and subsequent request. The lifespan of TempData can be increased beyond the first redirection using TempData.Keep() method. | lifespan of a session persists till it is forcefully destroyed by the server or the user. |
| 5 | On redirection the ViewModel object will be destroyed. | On redirection, the value in the ViewData becomes Null. | On redirection, the value in the ViewData becomes Null. | The data stored in TempData persists only during redirection. | The data stored in Session persists during any number of redirection. |
| 6 | Provides compile-time error checking and Intellisense support. | Does not provide compile-time error checking. | Does not provide compile-time error checking. | Does not provide compile-time error checking. | Does not provide compile-time error checking. |
| 7 | It is safe to use ViewModel in webfarm (server cluster) environment as they are not dependent on session. | ViewData is safe to use in webfarm environment as they are not dependent on session. | It is safe to use ViewBag in webfarm environment as they are not dependent on session. | TempData is not reliable in webfarm with cluster of servers as the TempData uses ASP.NET Session for storage. The workaround is to set Session State Mode to Out-of-Process and make the data stored in the TempData serializable. | Session is not reliable in web farm as they are stored on server’s memory. In a webfarm scenario, if a session is created by a server and the return request goes to another server in the cluster, then the session will be lost. The workaround is to set Session State Mode to Out-of-Process. |
| 8 | Usage:  (a) Display data in a table with master child relationship. (b) Reports with aggregate and summary. (c) Paging data in a table. (d) Flexible with very simple as well as highly complex data from multiple data-source. | Usage:  (a) To pass a list of data to render a drop down list. (b) To pass small amount of data to be rendered in the view. (c) Not ideal for complex data involving multiple data sources. | Usage:  (a) To pass a list of data to render a drop down list. (b) To pass small amount of data to be rendered in the view. (c) Not ideal for complex data involving multiple data sources. | Usage:  (a) Useful for storing one time messages like error message and validation messages. (b) Used in scenarios to pass small data from one action to another action or one controller call to another controller call. | Usage:  (a) To check whether the user is logged in to the website. (b) To store the user’s permission information. |

  Edit  Delete

# what is session Management and how we maintain the sessions and Types?

ASP.NET **session state** supports several **storage options** for session variables. Each option is identified as a session-state **mode** type. There are four **mode types** or just **modes**. In-Process mode, State Server mode, SQL Server mode, Custom mode and Off mode. These are **modes**. In-Process mode uses memory as **session storage**. State Server mode uses state-server as **session storage**. And so on. Besides, when a mode is specified by way of web.config, some additional parameters are also required like connectionstring, timeout etc. One of these parameters is "cookieless" for which the default value is "AutoDetect". If specified as "true", it will embed the ID in url. (As explained by @naivists). Cookieless option can be used with any mode.

There are three kinds of session, and they are listed as follows

1. Inprocess.
2. Outprocess.
3. Sql server session.

where they are stored.

1. inproc - default stored in web.config.
2. outproc - stored in server side.
3. Sql server - stored in database.

You have following types of session management in asp.net which you can define in your web.config file

Session mode="inproc"...means the session will be stored on the webserver within your application session mode="outproc"....means session will be stored on the server outside your application session mode="stateserver"...means session will be stored in a temporary memory in the database session mode="sqlserver"...means session will be stored in the database permanently.

  Edit  Delete

# what is HTML helper and Uses?

Using the HTML Helper class, we can create HTML Controls programmatically. HTML Helpers are used in View to render HTML content. HTML Helpers (mostly) is a method that returns a string. It is not mandatory to use HTML Helper classes for building an ASP.NET MVC application. We can build an ASP.NET MVC application without using them, but HTML Helpers helps in the rapid development of a view. HTML Helpers are more lightweight as compared to ASP.NET Web Form controls as they do not use ViewState and do not have event models.

HTML Helpers are categorized into three types:

1. Inline HTML Helpers
2. Built-in HTML Helpers
3. Custom HTML Helpers

In this Article, we will cover Built-In HTML Helpers. We will see Inline and Custom HTML Helpers in the upcoming article of this series.

Built-in HTML Helpers are further divided into three categories:

1. Standard HTML Helpers
2. Strongly Typed HTML Helpers
3. Templated HTML Helpers

| Standard HtmlHelper | **Strogly Typed HtmlHelpers** | **Templated HTML Helpers** |
| --- | --- | --- |
| Html.ActionLink |  | Html.Display |
| Html.TextBox | Html.TextBoxFor | Html.DisplayFor |
| Html.TextArea | Html.TextAreaFor | Html.Editor |
| Html.CheckBox | Html.CheckBoxFor | Html.EditorFor |
| Html.RadioButton | Html.RadioButtonFor |  |
| Html.DropDownList | Html.DropDownListFor |  |
| Html.ListBox | Html.ListBoxFor |  |
| Html.Hidden | Html.HiddenFor |  |
| Password | Html.PasswordFor |  |
| Html.Display | Html.DisplayFor |  |
| Html.Label | Html.LabelFor |  |
| Html.Editor | Html.EditorFor |  |

  Edit  Delete

# what is Area?

Area allows us to partition large application into smaller units where each unit contains separate MVC folder structure, same as default MVC folder structure. For example, large enterprise application may have different modules like admin, finance, HR, marketing etc. So an Area can contain separate MVC folder structure for all these modules as shown below.

 protected void Application\_Start()

        {

            AreaRegistration.RegisterAllAreas();

            GlobalConfiguration.Configure(WebApiConfig.Register);

            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

            RouteConfig.RegisterRoutes(RouteTable.Routes);

            BundleConfig.RegisterBundles(BundleTable.Bundles);

            //int sr = Convert.ToInt32("sss");

        }

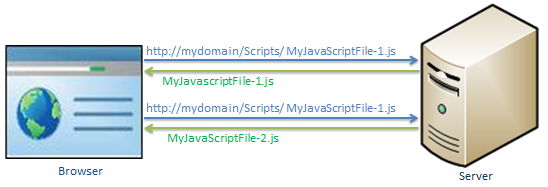
  Edit  Delete

# Bundling and Minification?

# Bundling

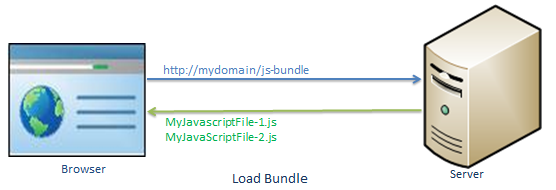
Bundling and minification techniques were introduced in MVC 4 to improve request load time. Bundling allow us to load the bunch of static files from the server into one http request.

The following figure illustrates the bundling techniques:

[](https://www.tutorialsteacher.com/Content/images/mvc/bundling-1.png)Load script files in separate requests

In the above figure, browser sends two separate requests to load two different JavaScript file MyJavaScriptFile-1.js and MyJavaScriptFile-2.js.

Bundling technique in MVC 4 allows us to load more than one JavaScript file, MyJavaScriptFile-1.js and MyJavaScriptFile-2.js in one request as shown below.

[](https://www.tutorialsteacher.com/Content/images/mvc/bundling-2.png)

# Minification

Minification technique optimizes script or css file size by removing unnecessary white space and comments and shortening variable names to one character.

For example, consider following JavaScript function.

Example: JavaScript

sayHello = function(name){

//this is comment

var msg = "Hello" + name;

alert(msg);

}

The above JavaScript will be optimized and minimized into following script.

Example: Minified JavaScript

sayHello=function(n){var t="Hello"+n;alert(t)}

As you can see above, it has removed unnecessary white space, comments and also shortening variable names to reduce the characters which in turn will reduce the size of JavaScript file.

Bundling and minification impacts on the loading of the page, it loads page faster by minimizing size of the file and number of requests.

# Bundle Types

MVC 5 includes following bundle classes in System.web.Optimization namespace:

**ScriptBundle**: ScriptBundle is responsible for JavaScript minification of single or multiple script files.

**StyleBundle**: StyleBundle is responsible for CSS minification of single or multiple style sheet files.

**DynamicFolderBundle**: Represents a Bundle object that ASP.NET creates from a folder that contains files of the same type.

All the above bundle classes are included in *System.Web.Optimization.Bundle* namespace and derived from [Bundle class](http://msdn.microsoft.com/en-us/library/system.web.optimization.bundle(v=vs.110).aspx).

Learn about ScriptBundle in the next section.

https://www.tutorialsteacher.com/Content/images/bulb-glow.png Points to Remember :

1. Bundling and Minification minimize static script or css files loading time therby minimize page loading time.
2. MVC framework provides ScriptBundle, StyleBundle and DynamicFolderBundle classes.
3. ScriptBundle does minification of JavaScript files.
4. StyleBundle does minification of CSS files.

 public class BundleConfig

    {

        // For more information on bundling, visit http://go.microsoft.com/fwlink/?LinkId=301862

        public static void RegisterBundles(BundleCollection bundles)

        {

            bundles.Add(new ScriptBundle("~/bundles/jquery").Include(

                        "~/Scripts/jquery-{version}.js"));

            // Use the development version of Modernizr to develop with and learn from. Then, when you're

            // ready for production, use the build tool at http://modernizr.com to pick only the tests you need.

            bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(

                        "~/Scripts/modernizr-\*"));

            bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(

                      "~/Scripts/bootstrap.js",

                      "~/Scripts/respond.js"));

            bundles.Add(new StyleBundle("~/Content/css").Include(

                      "~/Content/bootstrap.css",

                      "~/Content/site.css"));

        }

    }

  @Styles.Render("~/Content/css")

    @Scripts.Render("~/bundles/modernizr")

  Edit  Delete

# Partial view and Types?

Partial view is a reusable view, which can be used as a child view in multiple other views. It eliminates duplicate coding by reusing same partial view in multiple places. You can use the partial view in the layout view, as well as other content views.

@Html.Partial() helper method renders the specified partial view. It accept partial view name as a string parameter and returns MvcHtmlString. It returns html string so you have a chance of modifing the html before rendering.

@Html.RenderPartial() The RenderPartial helper method is same as the Partial method except that it returns void and writes resulted html of a specified partial view into a http response stream directly.

@Html.RenderAction() The RenderAction helper method invokes a specified controller and action and renders the result as a partial view. The specified Action method should return PartialViewResult using the Partial() method.--- returns Void

@Html.Action() The RenderAction helper method invokes a specified controller and action and renders the result as a partial view. The specified Action method should return PartialViewResult using the Partial() method.---returns MvcHtmlString

  Edit  Delete

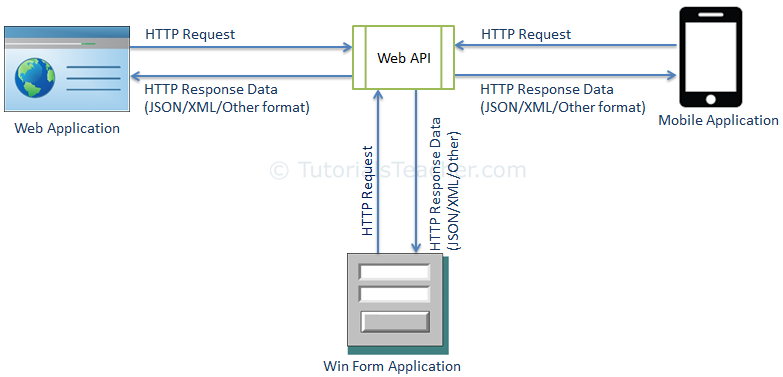
# what is web API? why we go for web API?

ASP.NET Web API is a framework for building HTTP services that can be consumed by a broad range of clients including browsers, mobiles, iphone and tablets. It is very similar to ASP.NET MVC since it contains the MVC features such as routing, controllers, action results, filter, model binders, IOC container or dependency injection.

  Edit  Delete

API (Application Programing Interface)

The ASP.NET Web API is an extensible framework for building HTTP based services that can be accessed in different applications on different platforms such as web, windows, mobile etc. It works more or less the same way as ASP.NET MVC web application except that it sends data as a response instead of html view. It is like a webservice or WCF service but the exception is that it only supports HTTP protocol.

[](https://www.tutorialsteacher.com/Content/images/webapi/webapi-overview.png)Web API

# ASP.NET Web API Characteristics

1. ASP.NET Web API is an ideal platform for building RESTful services.
2. ASP.NET Web API is built on top of ASP.NET and supports ASP.NET request/response pipeline
3. ASP.NET Web API maps HTTP verbs to method names.
4. ASP.NET Web API supports different formats of response data. Built-in support for JSON, XML, BSON format.
5. ASP.NET Web API can be hosted in IIS, Self-hosted or other web server that supports .NET 4.0+.
6. ASP.NET Web API framework includes new HttpClient to communicate with Web API server. HttpClient can be used in ASP.MVC server side, Windows Form application, Console application or other apps.

| **Web API Version** | **Supported .NET Framework** | **Coincides with** | **Supported in** |
| --- | --- | --- | --- |
| Web API 1.0 | .NET Framework 4.0 | ASP.NET MVC 4 | VS 2010 |
| Web API 2 - Current | .NET Framework 4.5 | ASP.NET MVC 5 | VS 2012, 2013 |

# ASP.NET Web API vs WCF

| **Web API** | **WCF** |
| --- | --- |
| Open source and ships with .NET framework. | Ships with .NET framework |
| Supports only HTTP protocol. | Supports HTTP, TCP, UDP and custom transport protocol. |
| Maps http verbs to methods | Uses attributes based programming model. |
| Uses routing and controller concept similar to ASP.NET MVC. | Uses Service, Operation and Data contracts. |
| Does not support Reliable Messaging and transaction. | Supports Reliable Messaging and Transactions. |
| Web API can be configured using HttpConfiguration class but not in web.config. | Uses web.config and attributes to configure a service. |
| Ideal for building RESTful services. | Supports RESTful services but with limitations. |

# When to choose WCF?

* + Choose WCF if you use .NET Framework 3.5. Web API does not support .NET 3.5 or below.
  + Choose WCF if your service needs to support multiple protocols such as HTTP, TCP, Named pipe.
  + Choose WCF if you want to build service with WS-\* standards like Reliable Messaging, Transactions, Message Security.
  + Choose WCF if you want to use Request-Reply, One Way, and Duplex message exchange patterns.

# When to choose ASP.NET Web API?

* + Choose Web API if you are using .NET framework 4.0 or above.
  + Choose Web API if you want to build a service that supports only HTTP protocol.
  + Choose Web API to build RESTful HTTP based services.
  + Choose Web API if you are familiar with ASP.NET MVC.

# Web API Features

1. Supports convention-based CRUD actions, since it works with HTTP verbs GET,POST,PUT and DELETE.
2. Responses have an Accept header and HTTP status code.
3. Supports multiple text formats like XML, JSON etc. or you can use your custom MediaTypeFormatter.
4. May accepts and generates the content which may not be object oriented like images, PDF files etc.
5. Automatic support for OData. Hence by placing the new [Queryable] attribute on a controller method that returns IQueryable, clients can use the method for OData query composition.
6. Supports Self-hosting or IIS Hosting.
7. Supports the [ASP.NET MVC features](https://www.dotnettricks.com/learn/mvc) such as routing, controllers, action results, filter, model binders, IOC container or dependency injection.

Register in global.asax.cs

 protected void Application\_Start()

        {

            AreaRegistration.RegisterAllAreas();

            GlobalConfiguration.Configure(WebApiConfig.Register);

            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

            RouteConfig.RegisterRoutes(RouteTable.Routes);

            BundleConfig.RegisterBundles(BundleTable.Bundles);

        }

Webapi configured in HttpConfiguration class

 public static class WebApiConfig

    {

        public static void Register(HttpConfiguration config)

        {

            // Web API configuration and services

            // Configure Web API to use only bearer token authentication.

            config.SuppressDefaultHostAuthentication();

            config.Filters.Add(new HostAuthenticationFilter(OAuthDefaults.AuthenticationType));

            // Web API routes

            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(

                name: "DefaultApi",

                routeTemplate: "api/{controller}/{id}",

                defaults: new { id = RouteParameter.Optional }

            );

        }

    }

  Edit  Delete

# Routing in Web API and Type?

1.Convention based routing - to define this type of routing, we call HttpRoute method and set its unique name, url pattern and specify some default values.

2.Attribute based routing - to define this type of routing, we specify the Route attribute in the action method of the controller.

public static class WebApiConfig

{

public static void Register(HttpConfiguration config)

{

// Attribute routing.

config.MapHttpAttributeRoutes();

// Convention-based routing.

config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

}

}

protected void Application\_Start()

{

// Pass a delegate to the Configure method.

GlobalConfiguration.Configure(WebApiConfig.Register);

}

  Edit  Delete

# what is the result type of Web API?

Action Method Return Type

In the previous section, you learned about parameter binding with Web API action method. Here, you will learn about the return types of action methods which in turn will be embedded in the Web API response sent to the client.

The Web API action method can have following return types.

1. Void
2. Primitive type or Complex type
3. HttpResponseMessage
4. IHttpActionResult

# Void

It's not necessary that all action methods must return something. It can have void return type.

For example, consider the following Delete action method that just deletes the student from the data source and returns nothing.

Example: Void Return Type

public class StudentController : ApiController

{

public void Delete(int id)

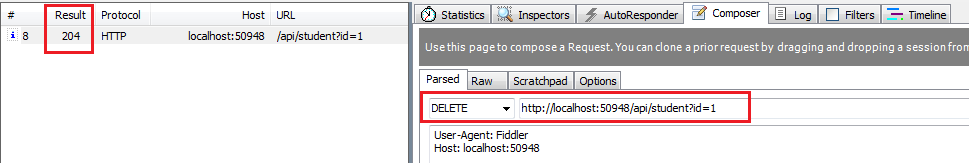
{

DeleteStudentFromDB(id);

}

}

As you can see above Delete action method returns void. It will send 204 "No Content" status code as a response when you send HTTP DELETE request as shown below.

[](https://www.tutorialsteacher.com/Content/images/webapi/void-response-status.png)Void Response Status

# Primitive or Complex Type

An action method can return primitive or other custom complex types as other normal methods.

Consider the following Get action methods.

Example: Primitive or Complex Return Type

public class Student

{

public int Id { get; set; }

public string Name { get; set; }

}

public class StudentController : ApiController

{

public int GetId(string name)

{

int id = GetStudentId(name);

return id;

}

public Student GetStudent(int id)

{

var student = GetStudentFromDB(id);

return student;

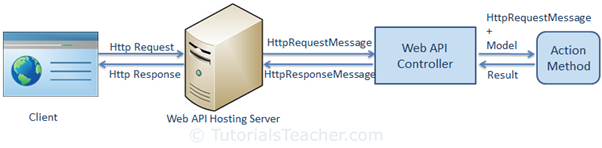
}

}

As you can see above, GetId action method returns an integer and GetStudent action method returns a Student type.

# HttpResponseMessage

Web API controller always returns an object of HttpResponseMessage to the hosting infrastructure. The following figure illustrates the overall Web API request/response pipeline.

[](https://www.tutorialsteacher.com/Content/images/webapi/webapi-request-pipeline.png)Web API Request Pipeline

Visit [Web API HTTP Message Life Cycle Poster](http://www.asp.net/media/4071077/aspnet-web-api-poster.pdf) for more details.

As you can see in the above figure, the Web API controller returns HttpResponseMessage object. You can also create and return an object of HttpResponseMessage directly from an action method.

The advantage of sending HttpResponseMessage from an action method is that you can configure a response your way. You can set the status code, content or error message (if any) as per your requirement.

Example: Return HttpResponseMessage

public HttpResponseMessage Get(int id)

{

Student stud = GetStudentFromDB(id);

if (stud == null) {

return Request.CreateResponse(HttpStatusCode.NotFound, id);

}

return Request.CreateResponse(HttpStatusCode.OK, stud);

}

In the above action method, if there is no student with specified id in the DB then it will return HTTP 404 Not Found status code, otherwise it will return 200 OK status with student data.

# IHttpActionResult

The *IHttpActionResult* was introduced in Web API 2 (.NET 4.5). An action method in Web API 2 can return an implementation of IHttpActionResult class which is more or less similar to ActionResult class in ASP.NET MVC.

You can create your own class that implements IHttpActionResult or use various methods of ApiController class that returns an object that implement the IHttpActionResult.

Example: Return IHttpActionResult Type using Ok() and NotFound() Methods

public IHttpActionResult Get(int id)

{

Student stud = GetStudentFromDB(id);

if (stud == null)

{

return NotFound();

}

return Ok(stud);

}

In the above example, if student with specified id does not exists in the database then it will return response with the status code 404 otherwise it sends student data with status code 200 as a response. As you can see, we don't have to write much code because NotFound() and Ok() method does it all for us.

The following table lists all the methods of ApiController class that returns an object of a class that implements IHttpActionResult interface.

| **ApiController Method** | **Description** |
| --- | --- |
| BadRequest() | Creates a BadRequestResult object with status code 400. |
| Conflict() | Creates a ConflictResult object with status code 409. |
| Content() | Creates a NegotiatedContentResult with the specified status code and data. |
| Created() | Creates a CreatedNegotiatedContentResult with status code 201 Created. |
| CreatedAtRoute() | Creates a CreatedAtRouteNegotiatedContentResult with status code 201 created. |
| InternalServerError() | Creates an InternalServerErrorResult with status code 500 Internal server error. |
| NotFound() | Creates a NotFoundResult with status code404. |
| Ok() | Creates an OkResult with status code 200. |
| Redirect() | Creates a RedirectResult with status code 302. |
| RedirectToRoute() | Creates a RedirectToRouteResult with status code 302. |
| ResponseMessage() | Creates a ResponseMessageResult with the specified HttpResponseMessage. |
| StatusCode() | Creates a StatusCodeResult with the specified http status code. |
| Unauthorized() | Creates an UnauthorizedResult with status code 401. |

     https://www.tutorialsteacher.com/webapi/action-method-return-type-in-web-api

  Edit  Delete

# Web API Request/Response Data Formats?

Here, you will learn how Web API handles different formats of request and response data.

# Media Type

Media type (aka MIME type) specifies the format of the data as type/subtype e.g. text/html, text/xml, application/json, image/jpeg etc.

In HTTP request, MIME type is specified in the request header using **Accept** and **Content-Type** attribute. The Accept header attribute specifies the format of response data which the client expects and the Content-Type header attribute specifies the format of the data in the request body so that receiver can parse it into appropriate format.

For example, if a client wants response data in JSON format then it will send following GET HTTP request with Accept header to the Web API.

HTTP GET Request:

GET http://localhost:60464/api/student HTTP/1.1

User-Agent: Fiddler

Host: localhost:1234

**Accept: application/json**

The same way, if a client includes JSON data in the request body to send it to the receiver then it will send following POST HTTP request with Content-Type header with JSON data in the body.

HTTP POST Request:

POST http://localhost:60464/api/student?age=15 HTTP/1.1

User-Agent: Fiddler

Host: localhost:60464

**Content-Type: application/json**

Content-Length: 13

{

id:1,

name:'Steve'

}

Web API converts request data into CLR object and also serialize CLR object into response data based on Accept and Content-Type headers. Web API includes built-in support for JSON, XML, BSON, and form-urlencoded data. It means it automatically converts request/response data into these formats OOB (out-of the box).

Example: Post Action Method

public class Student

{

public int Id { get; set; }

public string Name { get; set; }

}

public class StudentController : ApiController

{

public Student Post(Student student)

{

// save student into db

var insertedStudent = SaveStudent(student);

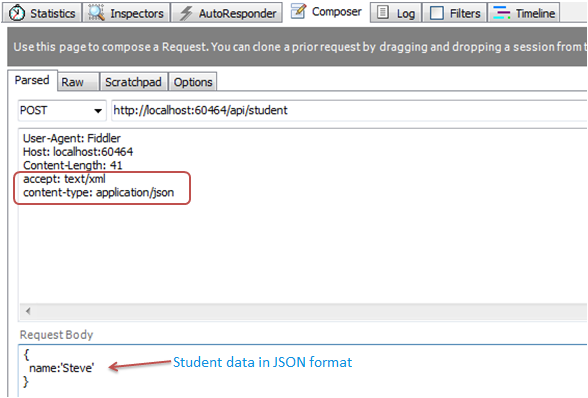
return insertedStudent;

}

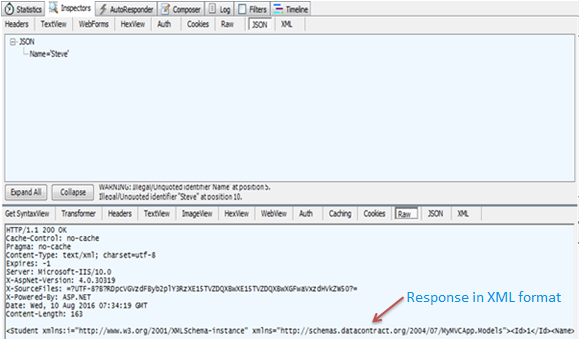
}

As you can see above, the Post() action method accepts Student type parameter, saves that student into DB and returns inserted student with generated id. The above Web API handles HTTP POST request with JSON or XML data and parses it to a Student object based on Content-Type header value and the same way it converts insertedStudent object into JSON or XML based on Accept header value.

The following figure illustrates HTTP POST request in fiddler.

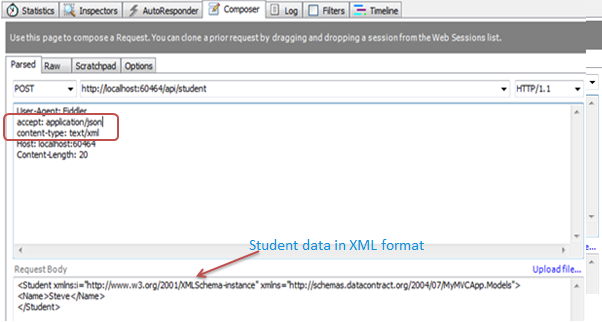
[](https://www.tutorialsteacher.com/Content/images/webapi/webapi-req-response1.png)Request-Response Data Format

In the above figure, Accept header specifies that it expects response data in XML format and Content-Type specifies that the student data into request body is in the JSON format. The following is the response upon execution of the above request.

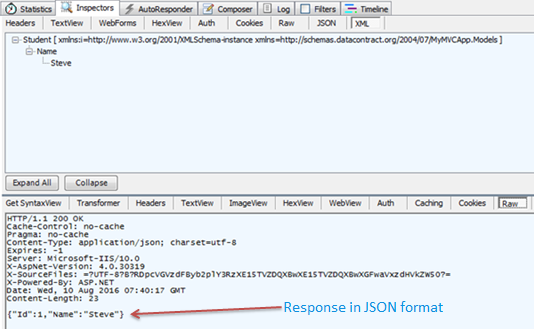
[](https://www.tutorialsteacher.com/Content/images/webapi/webapi-req-response2.png)Request-Response Data Format

The same way, you can specify different request & response format using accept and content-type headers and Web API will handle them without any additional changes.

The following HTTP POST request sends data in XML format and receives data in JSON format.

[](https://www.tutorialsteacher.com/Content/images/webapi/fiddler-req2.png)Web API Request

The above HTTP POST request will get the following response upon execution.

[](https://www.tutorialsteacher.com/Content/images/webapi/fiddler-response2.png)Web API Response

Thus, Web API handles JSON and XML data by default. Learn how Web API formats request/response data using formatters in the next section.

  Edit  Delete

# ASP.NET Web API: Media-Type Formatters?

         As you have seen in the previous section that Web API handles JSON and XML formats based on Accept and Content-Type headers. But, how does it handle these different formats? The answer is: By using Media-Type formatters.

Media type formatters are classes responsible for serializing request/response data so that Web API can understand the request data format and send data in the format which client expects.

Web API includes following built-in media type formatters.

| **Media Type Formatter Class** | **MIME Type** | **Description** |
| --- | --- | --- |
| JsonMediaTypeFormatter | application/json, text/json | Handles JSON format |
| XmlMediaTypeFormatter | application/xml, text/json | Handles XML format |
| FormUrlEncodedMediaTypeFormatter | application/x-www-form-urlencoded | Handles HTML form URL-encoded data |
| JQueryMvcFormUrlEncodedFormatter | application/x-www-form-urlencoded | Handles model-bound HTML form URL-encoded data |

Retrieve Built-in Media Type Formatters

As mentioned Web API includes above listed media type formatter classes by default. However, you can also add, remove or change the order of formatters.

The following example demonstrates HTTP Get method that returns all built-in formatter classes.

Example: Retrieve Built-in Formatters in C#

public class FormattersController : ApiController

{

public IEnumerable<string> Get()

{

IList<string> formatters = new List<string>();

foreach (var item in GlobalConfiguration.Configuration.Formatters)

{

formatters.Add(item.ToString());

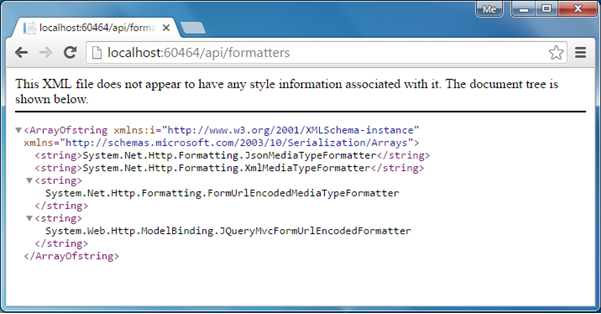
}

return formatters.AsEnumerable<string>();

}

}

In the above example, GlobalConfiguration.Configuration.Formatters returns MediaTypeFormatterCollection that includes all the formatter classes. The above example returns names of all the formatter classes as shown below.

[](https://www.tutorialsteacher.com/Content/images/webapi/formatters1.png)Built-in Media-Type Formatters

Alternatively, MediaTypeFormatterCollection class defines convenience properties that provide direct access to three of the four built-in media type formatters. The following example demonstrates retrieving media type formatters using MediaTypeFormatterCollection's properties.

Example: Retrieve Built-in Formatters in C#

public class FormattersController : ApiController

{

public IEnumerable<string> Get()

{

IList<string> formatters = new List<string>();

formatters.Add(GlobalConfiguration.Configuration.Formatters.JsonFormatter.GetType().FullName);

formatters.Add(GlobalConfiguration.Configuration.Formatters.XmlFormatter.GetType().FullName);

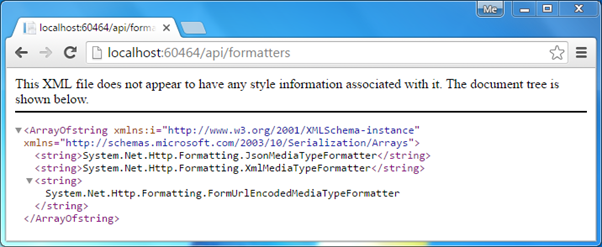
formatters.Add(GlobalConfiguration.Configuration.Formatters.FormUrlEncodedFormatter.GetType().FullName);

return formatters.AsEnumerable<string>();

}

}

The above example returns following response to the browser.

[](https://www.tutorialsteacher.com/Content/images/webapi/formatters2.png)Media-Type Formatters

BSON Formatter

Web API also supports BSON format. As the name suggests, BSON is binary JSON, it is a binary-encoded serialization of JSON-like documents. Currently there is very little support for BSON and no JavaScript implementation is available for clients running in browsers. This means that it is not possible to retrieve and automatically parse BSON data to JavaScript objects.

Web API includes built-in formatter class BsonMediaTypeFormatter for BSON but it is **disabled** by default. Learn more about BSON support in Web API [here](http://www.asp.net/web-api/overview/formats-and-model-binding/bson-support-in-web-api-21).

JSON Formatter

As mentioned above, Web API includes JsonMediaTypeFormatter class that handles JSON format. The JsonMediaTypeFormatter converts JSON data in an HTTP request into CLR objects (object in C# or VB.NET) and also converts CLR objects into JSON format that is embeded within HTTP response.

Internally, JsonMediaTypeFormatter uses third-party open source library called [Json.NET](http://json.codeplex.com/) to perform serialization.

Configure JSON Serialization

JSON formatter can be configured in WebApiConfig class. The JsonMediaTypeFormatter class includes various properties and methods using which you can customize JSON serialization. For example, Web API writes JSON property names with PascalCase by default. To write JSON property names with camelCase, set the CamelCasePropertyNamesContractResolver on the serializer settings as shown below.

Example: Customize JSON Serialization in C#

public static class WebApiConfig

{

public static void Register(HttpConfiguration config)

{

config.MapHttpAttributeRoutes();

config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

// configure json formatter

JsonMediaTypeFormatter jsonFormatter = config.Formatters.JsonFormatter;

jsonFormatter.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver();

}

}

XML Formatter

The XmlMediaTypeFormatter class is responsible for serializing model objects into XML data. It uses System.Runtime.DataContractSerializer class to generate XML data.

var responseObject = new HttpResponseMessage();

                responseObject.Headers.AcceptRanges.Add("bytes");

                responseObject.Content = new StreamContent(stream);

                responseObject.Content.Headers.ContentType = new MediaTypeHeaderValue("text/csv");

                responseObject.Content.Headers.ContentDisposition = new ContentDispositionHeaderValue("attachment")

                {

                    FileName = fileInfo.Name

                };

  Edit  Delete

**what is REST and RESTful service?**

  Edit  Delete

**7.CRUD oprations?**

  Edit  Delete

**How we handle the exception in Web API?**

  Edit  Delete

**How you pass the Data MVC to Web API?**

  Edit  Delete

**Defind POST,PUT,DELETE,GET and PATCH?**

  Edit  Delete

**Action Method Return Type in Web API?**

  Edit  Delete

**Action Method Return Type in Web API?**

  Edit  Delete

**Ninject in Web API?**

  Edit  Delete

**what is status code?**

  Edit  Delete

**what is EF?**

ADO.NET Entity Framework is an Object/Relational Mapping (ORM) framework that enables developers to work with relational database. It enabling developers to deal with data as objects and properties.Using the Entity Framework, developers issue queries [using LINQ](https://www.dotnettricks.com/learn/linq), then retrieve and manipulate data as strongly typed objects using C# or VB.Net.

The Entity Framework’s ORM implementation also provides services like change tracking, identity resolution, lazy loading, and query translation so that developers can focus on their application-specific business logic rather than the data access fundamentals. ADO.NET Entity Framework is targeted for developing Enterprise applications which can support MS SQL databases as well as other databases like as Oracle, DB2, MySql and many more. To learn the entity framework first we need to know what is orm. I am going to give the overview of ORM.

**ntroduction**  
The Entity Framework provides three approaches to create an entity model and each one has their own pros and cons.

1. Database First
2. Model First
3. Code First

**Database First**  
The Database First approach enables us to create an entity model from the existing database. This approach helps us to reduce the amount of code that we need to write. The following procedure will create an entity model using the Database First approach.  
  
**Step 1**

Create the ADO.Net entity data model using the "Generate from Database" option.  
  
  
  
**Step 2**

Select an existing connection or create new connection from "Choose Your Data Connection" windows and after this select database object (such as table, view and Stored Procedure) that are required into the project.  
  
  
  
This will generate with selected entities and [association](http://www.c-sharpcorner.com/UploadFile/ff2f08/association-in-entity-framework/) (relationship among entities).  
  
  
  
To update the model (in case of a database change), right-click on the model and select the “Update Model from Database” option and follow from Step 2.  
  
  
  
**Advantages**

* This approach is very popular when we have an existing database
* The EDM wizard creates the entities and their relationships automatically
* Extensibility is possible using a partial class or T4 templates
* We need to write less code and put less effort into creating the entity model
* Easy update if any changes are made to the database

**Disadvantages**

* Very difficult to maintain a complex model
* Model customization is very difficult. In other words, if we don't have a foreign key in the database and we need an association in the conceptual layer then we must create it manually and it is very difficult to maintain when we update the model from the database.

**Model First**  
In this approach, model classes and their relation is created first using the ORM designer and the physical database will be generated using this model. The Model First approach means we create a diagram of the entity and relation that will be converted automatically into a code model.  
  
The following procedure will create an entity model using the Model First approach.  
  
**Step 1**

Create the ADO.net entity data model using the "Empty EF Designer Model" option.  
  
  
  
**Step 2**

Now create the entity. Step 1 created the empty entity model. To add a new entity just right-click on the diagram and select “Add new” and the entity option. Fill in the entity name and database table name and primary key information.  
  
  
  
**Step 3**

Now to add properties to the entity.  
  
  
  
**Step 4**

Add associations (relation among entities) if any.  
  
  
  
**Step 5**

Once the entity model design is completed, we can generate the database from this model using the "Generate Database from model" context menu option. Here we can select any existing database connection or create a new database connection to create the database from a model.  
  
  
  
Step 5 will generate the DDL script and this generated file will be added to the solution as a script file.  
  
  
  
  
  
**Advantages**

* Visual designer to create a database schema
* Model diagram can be easily updated when the database changes
* Entity Framework generates the Code and database script
* Extensibility is possible using a partial class

**Disadvantages**

* When we change the model and generate SQL to sync the database then this will always result in data loss because the tables are dropped first.
* We don't have much control on entities and database.
* Requires a good knowledge of Entity Framework to update the model and database

**Code First**  
The Code First approach enables us to create a model and their relation using classes and then create the database from these classes. It enables us to work with the Entity Framework in an object-oriented manner. Here we need not worry about the database structure.  
  
As said earlier, we need to create a class that represents the database table. As an example I have created the two classes Employee and EmployeeDetails.

1. public partial class Employee
2. {
3. public int EmployeeId { get; set; }
5. [StringLength(10)]
6. public string Code { get; set; }
8. [StringLength(50)]
9. public string Name { get; set; }
11. public virtual EmployeeDetail EmployeeDetail { get; set; }
12. }
14. public partial class EmployeeDetail
15. {
16. [Key]
17. [DatabaseGenerated(DatabaseGeneratedOption.None)]
18. public int EmployeeId { get; set; }
20. [StringLength(25)]
21. public string PhoneNumber { get; set; }
23. [StringLength(255)]
24. public string EmailAddress { get; set; }
26. public virtual Employee Employee { get; set; }
27. }

The next step is to create a DbContext class and define a DbSet properties type of entity classes that are represented as a table in the database.

1. public partial class CodeFirst : DbContext
2. {
3. public CodeFirst() : base("name=CodeFirst")
4. {
5. }
7. public virtual DbSet<Employee> Employees { get; set; }
8. public virtual DbSet<EmployeeDetail> EmployeeDetails { get; set; }
10. protected override void OnModelCreating(DbModelBuilder modelBuilder)
11. {
12. }
13. }

Now if we want to create the database from the model then open the Package Manager Console and follow the migration procedure as shown below.  
  
  
  
**Step A: Enable Migration**  
  
enable-migrations -ContextTypeName “Conext class type with namespace” -MigrationsDirectory:”Migration directory”  
  
**Example**  
PM> enable-migrations -ContextTypeName   
EntityFrameworkApproaches.CodeFirst.CodeFirst -MigrationsDirectory:CodeFirst  
  
  
  
**Step B: Add Migration**  
Add-Migration -configuration –DbContext –Migrations –Configuration “Class-with-Namespaces” -Migrations- “Name”  
  
**Example**  
PM> Add-Migration -configuration EntityFrameworkApproaches.CodeFirst.Configuration InitialEntities  
  
  
  
**Step C: Update database**  
Update-Database -configuration –DbContext – Migrations “Configuration Class withNamespaces” –Verbose  
  
**Example**  
PM> Update-Database -configuration:EntityFrameworkApproaches.CodeFirst.Configuration -Verbose  
  
  
  
**Advantages**

* There is full control of the model from the code. There is no EDMX/designer and no auto-generated code
* It supports database migrations, so it is very easy to sync various databases
* We have more customization options and more control
* We can also use Code First to map our model to an existing database; to learn more click here

**Disadvantages**

* It is very difficult to maintain a database compared to a visual design tool
* Requires a good knowledge of C# and data annotation

**Comparing Approaches**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Entity Framework Approaches** | | |
| **Code First** | **Model First** | **Database First** |
| Support ORM designer tool (visual creation of data model) | No.  No EDMX support hence there are no any support of visual creation ofdata model. | Yes. | Yes. |
| Generates code and database scripts | Yes. | Yes. | NA  In this approach, we already have database and from the database wecreate model. |
| Extensible through partial classes | NA  In this approach, all are POCO classes, | Yes. | Yes. |
| Full control over model from code | Yes. | No. | No. |
| Manual changes to the database are possible? | Yes. | Yes. | Yes. |
| Easy to modify the model? | Yes. | Yes. | Yes. |
| An existing database can be used? | Yes.  To know more click [here](http://www.c-sharpcorner.com/UploadFile/ff2f08/entity-framework-code-first-migrations-with-the-existing-da/). | NA | Yes. |

**Selecting Right Approach**  
Definitely the development approach depends upon the project situation. The following diagram may help us to select the correct approach for your project. As in this diagram, if we already have domain classes, the Code First approach is best suited for our application. The same as if we have a database, Database First is a good option. If we don't have model classes and a database and require a visual entity designer tool then Model First is best suited.  
  
  
 **My opinion**  
It definitely depends on the situation what approach is more suitable for a project / application but after comparing three approaches, I still prefer Code First approaches. I have a couple of reasons for that.

* Code First supports database migrations.
* Much simpler to synchronize databases among developers and different versions of the application.
* More control over the Model. In other words, we have more control over how the entities and associations are created and how they work.

Summary   
Entity Framework provides three approaches to create and maintain entity models. Each one has their own pros and cons. Depending upon the situation, we can select one of the best suited approaches for our application.

  Edit  Delete

# OOPS concepts?

1.Class--Classes are blueprints for Object.

2.Object--Objects are instance of classes

3.Abstraction

4.Encapsulation

5.Polymorphism

6.Inheritance

# Abstraction

Abstraction allows us to represent complex real world in simplest manner. It is process of identifying the relevant qualities and behaviors an object should possess, in other word represent the necessary feature without representing the back ground details. Abstraction is a process of hiding work style of an object and showing only those information which are required to understand the object. Abstraction means putting all the variables and methods in a class which are necessary.

# Encapsulation

It is a process of hiding all the internal details of an object from the outside real world. The word Encapsulation, like Enclosing into the capsule. It restrict client from seeing its internal view where behavior of the abstraction is implemented. In Encapsulation, generally to hide data making it private and expose public property to access those data from outer world. Encapsulation is a method for protecting data from unwanted access or alteration. Encapsulation is the mechanism by which Abstraction is implemented.

# Difference between Abstraction and Encapsulation

Abstraction is a process. It is the act of identifying the relevant qualities and behaviors an object should possess. Encapsulation is the mechanism by which the abstraction is implemented.

|  |  |
| --- | --- |
| Abstraction | Encapsulation |
| Abstraction solves the problem in the design level. | Encapsulation solves the problem in the implementation level. |
| Abstraction is used for hiding the unwanted data and giving onlyrelevant data. | Encapsulation is hiding the code and data into a single unit toprotect the data from outer world. |
| Abstraction is set focus on the object instead of how it does it. | Encapsulation means hiding the internal details or mechanics of howan object does something. |
| Abstraction is outer layout in terms of design.  For Example: - Outer Look of a iPhone, like it has a display screen. | Encapsulation is inner layout in terms of implementation. For Example: - Inner Implementation detail of a iPhone, how DisplayScreen are connect with each other using circuits |

C# Polymorphism

The term "Polymorphism" is the combination of "poly" + "morphs" which means many forms. It is a greek word. In object-oriented programming, we use 3 main concepts: inheritance, encapsulation and polymorphism.

There are two types of polymorphism in C#: compile time polymorphism and runtime polymorphism. Compile time polymorphism is achieved by method overloading and operator overloading in C#. It is also known as static binding or early binding. Runtime polymorphism in achieved by method overriding which is also known as dynamic binding or late binding.

# C# Runtime Polymorphism Example

Let's see a simple example of runtime polymorphism in C#.

1. using System;
2. public class Animal{
3. public virtual void eat(){
4. Console.WriteLine("eating...");
5. }
6. }
7. public class Dog: Animal
8. {
9. public override void eat()
10. {
11. Console.WriteLine("eating bread...");
12. }
14. }
15. public class TestPolymorphism
16. {
17. public static void Main()
18. {
19. Animal a= new Dog();
20. a.eat();
21. }
22. }

Output:

eating bread...

# C# Runtime Polymorphism Example 2

Let's see a another example of runtime polymorphism in C# where we are having two derived classes.

1. using System;
2. public class Shape{
3. public virtual void draw(){
4. Console.WriteLine("drawing...");
5. }
6. }
7. public class Rectangle: Shape
8. {
9. public override void draw()
10. {
11. Console.WriteLine("drawing rectangle...");
12. }
14. }
15. public class Circle : Shape
16. {
17. public override void draw()
18. {
19. Console.WriteLine("drawing circle...");
20. }
22. }
23. public class TestPolymorphism
24. {
25. public static void Main()
26. {
27. Shape s;
28. s = new Shape();
29. s.draw();
30. s = new Rectangle();
31. s.draw();
32. s = new Circle();
33. s.draw();
35. }
36. }

Output:

drawing...

drawing rectangle...

drawing circle...

# Runtime Polymorphism with Data Members

Runtime Polymorphism can't be achieved by data members in C#. Let's see an example where we are accessing the field by reference variable which refers to the instance of derived class.

1. using System;
2. public class Animal{
3. public string color = "white";
5. }
6. public class Dog: Animal
7. {
8. public string color = "black";
9. }
10. public class TestSealed
11. {
12. public static void Main()
13. {
14. Animal d = new Dog();
15. Console.WriteLine(d.color);
17. }
18. }

Output:

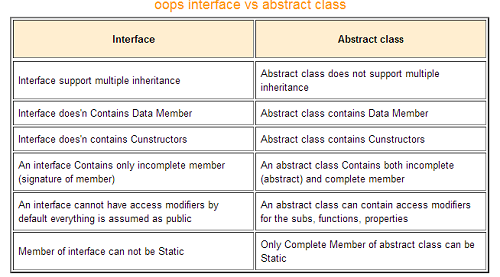
white

  Edit  Delete

# Differents between interface and Abstract methods?

# What is an Abstract Class?

An abstract class is a special type of class which acts as a base of other classes and cannot be instantiated. The implementation logic of an abstract class is provided by its derived classes. To make a class abstract, the “abstract” modifier is used which means some missing implementation needs to be implemented in the class derived from it. It contains both abstract and non-abstract members. An abstract class is intended to provide basic functionality which can be further shared and overridden by multiple derived classes. It is useful to avoid any kind of code duplication. They look very much like interfaces but with added functionality.

[](http://cdn.differencebetween.net/wp-content/uploads/2017/11/Difference-between-Abstract-Class-and-Interface-in-C.png)

# What is an Interface?

An interface, on the other hand, is not a class which contains only the signature of functionality. It’s a pattern with no implementation. Conceptually speaking, it’s just the definition of methods which contains only the declaration of members. It’s an empty shell which does not contain the implementation of its members. It’s like an abstract base class which only contains abstract members such as methods, events, indexers, properties, etc. It cannot be instantiated directly and its members can be implemented by any class. Additionally, multiple interfaces can be implemented by a class, however, a class can only inherit a single class.

# Abstract Class vs. Interface : difference between Abstract Class and Interface in C#

1. **Multiple Inheritance** – A class can only use one abstract class, hence multiple inheritance is not supported. An interface, on the other hand, can support multiple inheritance, which means a class can inherit any number of inheritances.
2. **Definition**  of **Abstract Class and Interface in C#**– An abstract class is a special type of class which may contain definition with no implementation. The implementation logic is provided by its derived classes. It can have abstract as well as non-abstract methods. An interface, on the other hand, is just a pattern which cannot do anything. Technically, it’s just an empty shell.
3. **Implementation** – An abstract class can contain both definition and its implementation. It’s an incomplete class which cannot be instantiated. An interface can only have the signature of the functionality without any code.
4. **Access Modifiers** – An abstract class can have several access modifiers such as subs, functions, properties, etc, while an interface is not allowed to have access modifiers and all methods must be implicitly defined as public.
5. **Homogeneity** – An abstract class is used for implementations of the same type, behavior, and status, while an interface is used for implementations that share only method signatures.
6. **Declaration** – An abstract class acts as a base class for all other classes so it can declare or use any variable while an interface is not allowed to declare any variables.
7. **Constructor Declaration** – While an abstract class can have[constructor declaration, an interface cannot have constructor declaration](http://www.differencebetween.net/technology/difference-between-constructor-and-method/).
8. **Core vs. Peripheral** – An abstract class is used to define the core identity of a class and can be used for objects of the same data type. An interface, on the other hand, is used to define the peripheral ability of a class.
9. **Rigid vs. Supple** – An abstract class is more supple in terms of functionality, at least from a developer’s perspective, while an interface is more rigid.

**Abstract Class vs. Interface: Table form**

|  |  |
| --- | --- |
| **Abstract Class** | **Interface** |
| An abstract class can have all access modifiers for member declaration of functions, subs, and properties. | An interface is not allowed to have access modifiers for member declaration of functions, subs, properties, etc. All the members are assumed as implicitly public. |
| A class can at the most use only one abstract class. | A class may inherit any number of interfaces. |
| Multiple inheritance is not supported in abstract class. | An interface may support multiple inheritance. |
| Technically, it’s a class which may or may not contain both definition and its implementation. | An interface can only have the signature of functionality so it’s basically an empty shell. |
| An abstract class can contain members. consts, defined methods, and method stubs. | An interface can only contain methods and consts. |
| It defines the core identity of a class and is used to describe objects of the same data type. | It is used to describe the peripheral abilities of a class. |
| Members of an abstract class cannot be static unless it’s a Complete Member. | Members of an interface cannot be static. |
| It’s ideal for implementations of the same kind and common behavior. | It’s better to use interfaces if several implementations share only method signatures. |
| It can have constructor declaration. | [It cannot have constructor declaration](http://www.differencebetween.net/technology/difference-between-constructor-and-destructor/). |
| An abstract class has pre-defined fields and constrants. | Fields cannot be defined in interfaces. |
| It can have both abstract and non-abstract methods. | It can only have abstract methods. |

Read more: [Difference between Abstract Class and Interface in C# - Difference Between | Difference Between | Difference between Abstract Class vs Interface in C# -http://www.differencebetween.net/technology/difference-between-abstract-class-and-interface-in-c/#ixzz5kUDfAHAO](http://www.differencebetween.net/technology/difference-between-abstract-class-and-interface-in-c/#ixzz5kUDfAHAO)

  Edit  Delete

**Boxing and UnBoxing?**

int i = 123;      // a value type

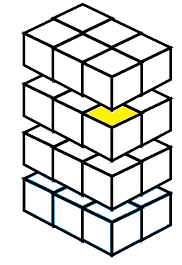
object o = i;     // boxing -- convert value type to Reference type is boxing

int j = (int)o;   // unboxing --convert Reference type to value type is unboxing--using cast

  Edit  Delete

**Differents between String and String Builder?**

string and StringBuilder

****A string instance is immutable. Immutable means once we create a string object we cannot modify the value of the string Object in the memory. Any operation that appears to modify the string, it will discard the old value and it will create new instance in memory to hold the new value.

The System.Text.StringBuilder is mutable, that means once we create StringBuilder object we can perform any operation that appears to change the value without creating new instance for every time. It can be modified in any way and it doesn't require creation of new instance.

String Example

string colors;

colors += "red";

colors += "blue";

colors += "green";

****In the above code string color will alter 3 times, each time the code perfom a string operation (+=). That mean 3 new string created in the memory. When you perform repeated operation to a string, the overhead associated with creating a new String object can be costly.

# StringBuilder Example

StringBuilder sb = new StringBuilder("");

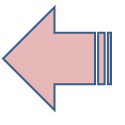
sb.Append("red");

sb.Append("blue");

sb.Append("green ");

string colors = sb.ToString();

In the above code the StringBuilder object will alter 3 times, each time the code attempt a StringBuilder operation without creating a new object. That means, using the StringBuilder class can boost performance when concatenating many strings together in a loop.

But immutable objects have some advantages also, such as they can be used across threads without fearing synchronization problems. On the other hand, when initializing a StringBuilder, you are going down in performance. Also many actions that you do with string can't be done with StringBuilder object.

  Edit  Delete

# what are the Access specifiers and Explain?

**Introduction**  
Access modifiers and specifiers are keywords (private, public, internal, protected and protected internal) to specify the accessibility of a type and its members.  
  
C# has 5 access specifier or access modifier keywords; those are private, public, internal, protected and protected Internal.  
  
Usage of Access Specifiers **private:**limits the accessibility of a member to within the defined type, for example if a variable or a functions is being created in a ClassA and declared as private then another ClassB can't access that.  
  
**public:**has no limits, any members or types defined as public can be accessed within the class, assembly even outside the assembly. Most DLLs are known to be produced by public class and members written in a .cs file.  
  
**internal:**internal plays an important role when you want your class members to be accessible within the assembly. An assembly is the produced .dll or .exe from your .NET Language code (C#). Hence, if you have a C# project that has ClassA, ClassB and ClassC then any internal type and members will become accessible across the classes with in the assembly.  
  
**protected:**plays a role only when inheritance is used. In other words any protected type or member becomes accessible when a child is inherited by the parent. In other cases (when no inheritance) protected members and types are not visible.  
  
**Protected internal:**is a combination of protected and internal both. A protected internal will be accessible within the assembly due to its internal flavor and also via inheritance due to its protected flavor.

|  |  |
| --- | --- |
| Modifier | Description |
| public | There are no restrictions on accessing public members. |
| private | Access is limited to within the class definition. This is the default access modifier type if none is formally specified |
| protected | Access is limited to within the class definition and any class that inherits from the class |
| internal | Access is limited exclusively to classes defined within the current project assembly |
| protected internal | Access is limited to the current assembly and types derived from the containing class. All members in current project and all members in derived class can access the variables. |
| private protected | Access is limited to the containing class or types derived from the containing class within the current assembly. |

  Edit  Delete

# readonly vs constant vs static in c#?

# Const

A variable declared as const (read: constant) must be assigned a value at declaration, and this value may not then change at a later time.

public const string ConnectionString = "YourConnectionString";

The value in a const variable is what's called a "compile-time" value, and is immutable (which means [it does not change over the life of the program](https://msdn.microsoft.com/en-us/library/ms173119.aspx)).

Only [primitive or "built-in" C# types](https://msdn.microsoft.com/en-us/library/ya5y69ds.aspx) (e.g. int, string, double) are allowed to be declared const. Therefore, you cannot write either of these:

public const DateTime DeclarationOfIndependence = new DateTime(1776,7,4);

public const MyClass MyValue = new Class() {Name = "TestName"};

You want to use const when you have a variable whose value will not change, ever, during the time your application is being used. Further, any variable declared as const will also, implicitly, be declared static.

But that begs the question: what does static do?

# Static

A static member (variable, method, etc) belongs to the type of an object rather than to an instance of that type. Hence, if we declare this:

public class MyClass

{

public static string MyMethod() { ... }

}

We must call this method like this:

var result = MyClass.MyMethod();

We will NOT be able to make calls like this:

var myClass = new MyClass();

var result = myClass.MyMethod(); //Will not compile

Now there's only one keyword left to define: readonly.

# Readonly

A readonly field is one where assignment to that field can only occur as part of the declaration of the class or in a constructor.

public class TestClass

{

public readonly string ConnectionString = "TestConnection";

public TestClass()

{

ConnectionString = "DifferentConnection";

}

public void TestMethod ()

{

ConnectionString = "NewConnection";//Will not compile

}

}

This means that a readonly variable can have different values for different constructors in the same class.

  Edit  Delete

# what is the different between Ref and Out?

# Introduction

The keywords ref and out are used to pass arguments within a method or function. Both indicate that an argument/parameter is passed by reference. By default parameters are passed to a method by value. By using these keywords (ref and out) we can pass a parameter by reference.

# Ref Keyword

The ref keyword passes arguments by reference. It means any changes made to this argument in the method will be reflected in that variable when control returns to the calling method.

**Example code**

1. public static string GetNextName(ref int id)
2. {
3. string returnText = "Next-" + id.ToString();
4. id += 1;
5. return returnText;
6. }
7. static void Main(string[] args)
8. {
9. int i = 1;
10. Console.WriteLine("Previous value of integer i:" + i.ToString());
11. string test = GetNextName(ref i);
12. Console.WriteLine("Current value of integer i:" + i.ToString());
13. }

**Output**

# Out Keyword

The out keyword passes arguments by reference. This is very similar to the ref keyword.

**Example Code**

1. public static string GetNextNameByOut(out int id)
2. {
3. id = 1;
4. string returnText = "Next-" + id.ToString();
5. return returnText;
6. }
7. static void Main(string[] args)
8. {
9. int i = 0;
10. Console.WriteLine("Previous value of integer i:" + i.ToString());
11. string test = GetNextNameByOut(out i);
12. Console.WriteLine("Current value of integer i:" + i.ToString());
13. }

**Output**

# Ref Vs Out

|  |  |
| --- | --- |
| Ref | Out |
| The parameter or argument must be initialized first before it is passed to ref. | It is not compulsory to initialize a parameter or argument before it is passed to an out. |
| It is not required to assign or initialize the value of a parameter (which is passed by ref) before returning to the calling method. | A called method is required to assign or initialize a value of a parameter (which is passed to an out) before returning to the calling method. |
| Passing a parameter value by Ref is useful when the called method is also needed to modify the pass parameter. | Declaring a parameter to an out method is useful when multiple values need to be returned from a function or method. |
| It is not compulsory to initialize a parameter value before using it in a calling method. | A parameter value must be initialized within the calling method before its use. |
| When we use REF, data can be passed bi-directionally. | When we use OUT data is passed only in a unidirectional way (from the called method to the caller method). |
| Both ref and out are treated differently at run time and they are treated the same at compile time. | |
| Properties are not variables, therefore it cannot be passed as an out or ref parameter. | |

# Ref / Out keyword and method Overloading

Both ref and out are treated differently at run time and they are treated the same at compile time, so methods cannot be overloaded if one method takes an argument as ref and the other takes an argument as an out.

**Example code**

1. public static string GetNextName(ref int id)
2. {
3. string returnText = "Next-" + id.ToString();
4. id += 1;
5. return returnText;
6. }
7. public static string GetNextName(out int id)
8. {
9. id = 1;
10. string returnText = "Next-" + id.ToString();
11. return returnText;
12. }

**Output** when the code is compiled:

However, method overloading can be possible when one method takes a ref or out argument and the other takes the same argument without ref or out.

**Example Code**

1. public static string GetNextName(int id)
2. {
3. string returnText = "Next-" + id.ToString();
4. id += 1;
5. return returnText;
6. }
7. public static string GetNextName(ref int id)
8. {
9. string returnText = "Next-" + id.ToString();
10. id += 1;
11. return returnText;
12. }

  Edit  Delete

# what is Partial Class?

**Partial class** is a concept where a single class can be split into 2 or more files. This feature is useful when a class consists of a large number of members (functions, properties and so on).  
  
In such cases functions can be developed in one file and properties in another.

Advantages:

1. It avoids programming confusion (in other words better readability).
2. Multiple programmers can work with the same class using different files.
3. Even though multiple files are used to develop the class all such files should have a common class name.

**Note:**

* When compilation is done all the files will be compiled into a single file.
* When the **CLR** executes the program it doesn't differentiate between a normal class or partial class.

*Foldername : Console* **Filename: partial1.cs**

1. using System;
2. partial class A
3. {
4. public void Add(int x,int y)
5. {
6. Console.WriteLine("sum is {0}",(x+y));
7. }
8. }

**Filename: partial2.cs**

1. using System;
2. partial class A
3. {
4. public void Substract(int x,int y)
5. {
6. Console.WriteLine("Difference is {0}", (x-y));
7. }
8. }
10. class Demo
11. {
12. public static void Main()
13. {
14. A obj=new A();
15. obj.Add(7,3);
16. obj.Substract(15,12);
17. }
18. }

**Note**

  Edit  Delete

# what is Partial Class?

**Partial class** is a concept where a single class can be split into 2 or more files. This feature is useful when a class consists of a large number of members (functions, properties and so on).  
  
In such cases functions can be developed in one file and properties in another.

Advantages:

1. It avoids programming confusion (in other words better readability).
2. Multiple programmers can work with the same class using different files.
3. Even though multiple files are used to develop the class all such files should have a common class name.

**Note:**

* When compilation is done all the files will be compiled into a single file.
* When the **CLR** executes the program it doesn't differentiate between a normal class or partial class.

*Foldername : Console* **Filename: partial1.cs**

1. using System;
2. partial class A
3. {
4. public void Add(int x,int y)
5. {
6. Console.WriteLine("sum is {0}",(x+y));
7. }
8. }

**Filename: partial2.cs**

1. using System;
2. partial class A
3. {
4. public void Substract(int x,int y)
5. {
6. Console.WriteLine("Difference is {0}", (x-y));
7. }
8. }
10. class Demo
11. {
12. public static void Main()
13. {
14. A obj=new A();
15. obj.Add(7,3);
16. obj.Substract(15,12);
17. }
18. }

**Note**

  Edit  Delete

# C# - Extension Method

Extension methods, as the name suggests, are additional methods. Extension methods allow you to inject additional methods without modifying, deriving or recompiling the original class, struct or interface. Extension methods can be added to your own custom class, .NET framework classes, or third party classes or interfaces.

In the following example, IsGreaterThan() is an extension method for int type, which returns true if the value of the int variable is greater than the supplied integer parameter.

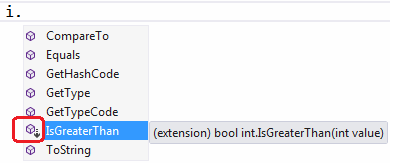
Example: Extension Method

int i = 10;

bool result = i.IsGreaterThan(100); //returns false

The IsGreaterThan() method is not a method of int data type (Int32 struct). It is an extension method written by the programmer for the int data type. The IsGreaterThan() extension method will be available throughout the application by including the namespace in which it has been defined.

The extension methods have a special symbol in intellisense of the visual studio, so that you can easily differentiate between class methods and extension methods.

[](https://www.tutorialsteacher.com/Content/images/csharp/extension-method.png)Extension Method Symbol in visual studio intellisense

Now let's see how to write an extension method.

https://www.tutorialsteacher.com/Content/images/tips.png LINQ is built upon extension methods that operate on IEnumerable and IQeryable type.

An extension method is actually a special kind of static method defined in a static class. To define an extension method, first of all, define a static class.

For example, we have created an IntExtensions class under the ExtensionMethods namespace in the following example. The IntExtensions class will contain all the extension methods applicable to int data type. (You may use any name for namespace and class.)

Example: Create a Class for Extension Methods

namespace ExtensionMethods

{

public static class IntExtensions

{

}

}

Now, define a static method as an extension method where the first parameter of the extension method specifies the type on which the extension method is applicable. We are going to use this extension method on int type. So the first parameter must be int preceded with the ***this*** modifier.

For example, the IsGreaterThan() method operates on int, so the first parameter would be, this int i.

Example: Define an Extension Method

namespace ExtensionMethods

{

public static class IntExtensions

{

public static bool IsGreaterThan(this int i, int value)

{

return i > value;

}

}

}

Now, you can include the ExtensionMethods namespace wherever you want to use this extension method.

Example: Extension method

using ExtensionMethods;

class Program

{

static void Main(string[] args)

{

int i = 10;

bool result = i.IsGreaterThan(100);

Console.WriteLine(result);

}

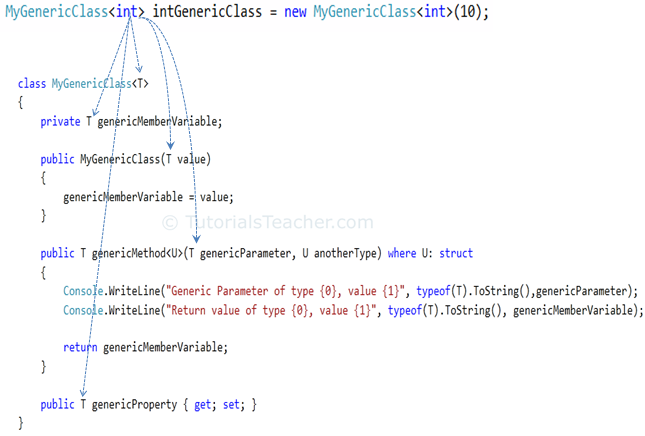
}

  Edit  Delete

# .what is delegates?

# what is Generic method?

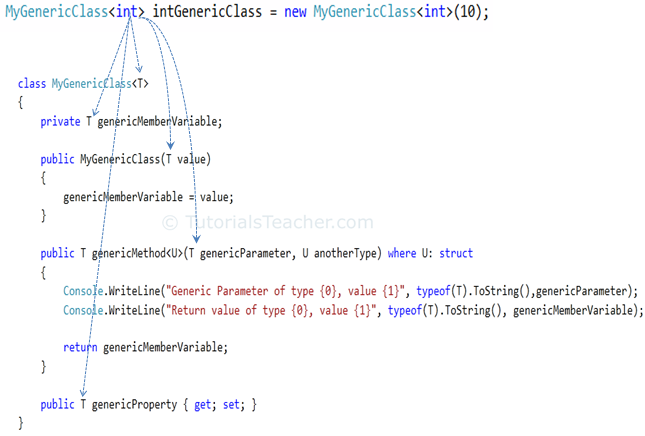
1. Increases the reusability of the code.
2. Generic are type safe. You get compile time errors if you try to use a different type of data than the one specified in the definition.
3. Generic has a performance advantage because it removes the possibilities of boxing and unboxing.

[](https://www.tutorialsteacher.com/Content/images/csharp/generic-class.png)C# Generic Class

  Edit  Delete

# what is Generic method?

1. Increases the reusability of the code.
2. Generic are type safe. You get compile time errors if you try to use a different type of data than the one specified in the definition.
3. Generic has a performance advantage because it removes the possibilities of boxing and unboxing.

[](https://www.tutorialsteacher.com/Content/images/csharp/generic-class.png)C# Generic Class

  Edit  Delete

# what is static method?

# Introduction

C# supports two types of class methods, static methods, and non static methods. Any normal method is a non static method.

A static method in C# is a method that keeps only one copy of the method at the Type level, not the object level. That means, all instances of the class share the same copy of the method and its data. The last updated value of the method is shared among all objects of that Type.

Static methods are called by using the class name, not the instance of the class.

The Console class and its Read and Write methods are an example of static methods. The following code example calls Console.WriteLine and Console.ReadKey methods without creating an instance of the Console class.

1. class Program
2. {
3. public static void withoutObj()
4. {
5. Console.WriteLine("Hello");
6. }
7. static void Main()
8. {
9. Program. withoutObj();
10. Console.ReadKey();
11. }
12. }

# Using Static Method

Usually we define a set of data members for a class and then every object of that class will have a separate copy of each of those data members. Let's have an example.

1. class Program
2. {
3. public int myVar;  //a non-static field
4. static void Main()
5. {
6. Program p1 = new Program();  //a object of class
7. p1.myVar = 10;
8. Console.WriteLine(p1.myVar);
9. Console.ReadKey();
10. }
11. }

In the above example, myVar is a non-static field so to use this field we first need to create the object of that class. On the other hand, static data is shared among all the objects of that class. That is, for all the objects of a class, there will be only one copy of static data. Let's have an example.

1. class Program
2. {
3. public static int myVar;  //a static field
4. static void Main()
5. {
6. //Program p1 = new Program();  //a object of class
7. myVar = 10;
8. Console.WriteLine(myVar);
9. Console.ReadKey();
10. }
11. }

In the above we don't have an object of the class to use that field since the field is static.

If you create your own class and you think only one copy of the data (method) is needed among all instances of the class, you can create your own static method.  Learn more here: [Static Class  and Static Class Members In C#](https://www.c-sharpcorner.com/blogs/what-is-static-class-and-static-class-members-in-c-sharp)

Notable Points here are:

1. A static method can be invoked directly from the class level
2. A static method does not require any class object
3. Any main() method is shared through the entire class scope so it always appears with static keyword.

  Edit  Delete

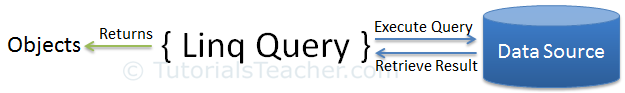
# What is LINQ?

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.

For example, SQL is a Structured Query Language used to save and retrieve data from a database. In the same way, LINQ is a structured query syntax built in C# and VB.NET to retrieve data from different types of data sources such as collections, ADO.Net DataSet, XML Docs, web service and MS SQL Server and other databases.

[](https://www.tutorialsteacher.com/Content/images/linq/linq-usage.PNG)LINQ Usage

LINQ queries return results as objects. It enables you to uses object-oriented approach on the result set and not to worry about transforming diffent formats of results into objects.

[](https://www.tutorialsteacher.com/Content/images/linq/linq-execution.PNG)

The following example demonstrates a simple LINQ query that gets all strings from an array which contains 'a'.

Example: LINQ Query to Array

// Data source

string[] names = {"Bill", "Steve", "James", "Mohan" };

// LINQ Query

var myLinqQuery = from name in names

where name.Contains('a')

select name;

// Query execution

foreach(var name in myLinqQuery)

Console.Write(name + " ");

In the above example, string array names is a data source. The following is a LINQ query which is assigned to a variable myLinqQuery.

from name in names

where name.Contains('a')

select name;

The above query uses query syntax of LINQ. You will learn more about it in the [Query Syntax](https://www.tutorialsteacher.com/linq/linq-query-syntax) chapter.

You will not get the result of a LINQ query until you execute it. LINQ query can be execute in multiple ways, here we used foreach loop to execute our query stored in myLinqQuery. The foreach loop executes the query on the data source and get the result and then iterates over the result set.

Thus, every LINQ query must query to some kind of data sources whether it can be array, collections, XML or other databases. After writing LINQ query, it must be executed to get the result.

  Edit  Delete

# what is enum?

# C# - enum

In C#, enum is a value type data type. The enum is used to declare a list of named integer constants. It can be defined using the *enum* keyword directly inside a namespace, class, or structure. The enum is used to give a name to each constant so that the constant integer can be referred using its name.

Example: enum

enum WeekDays

{

Monday = 0,

Tuesday =1,

Wednesday = 2,

Thursday = 3,

Friday = 4,

Saturday =5,

Sunday = 6

}

Console.WriteLine(WeekDays.Friday);

Console.WriteLine((int)WeekDays.Friday);

Output:

Friday   
4

By default, the first member of an enum has the value 0 and the value of each successive enum member is increased by 1. For example, in the following enumeration, Monday is 0, Tuesday is 1, Wednesday is 2 and so forth.

Example: enum

enum WeekDays

{

Monday,

Tuesday,

Wednesday,

Thursday,

Friday,

Saturday,

Sunday

}

Console.WriteLine((int)WeekDays.Monday);

Console.WriteLine((int)WeekDays.Friday);

Output:

0   
4

An explicit cast is necessary to convert from enum type to an integral type. For example, to get the int value from an enum:

Example: enum

int dayNum = (int)WeekDays.Friday;

Console.WriteLine(dayNum);

Output:

4

A change in the value of the first enum member will automatically assign incremental values to the other members sequentially. For example, changing the value of Monday to 10, will assign 11 to Tuesday, 12 to Wednesday, and so on:

Example: enum

enum WeekDays

{

Monday = 10,

Tuesday,

Wednesday,

Thursday,

Friday,

Saturday,

Sunday

}

Console.WriteLine((int)WeekDays.Monday);

Console.WriteLine((int)WeekDays.Friday);

Output:

10   
14

The enum can includes named constants of numeric data type e.g. byte, sbyte, short, ushort, int, uint, long, or ulong.

https://www.tutorialsteacher.com/Content/images/tips.png enum cannot be used with string type.

Enum is mainly used to make code more readable by giving related constants a meaningful name. It also improves maintainability.

# Enum methods:

Enum is an abstract class that includes static helper methods to work with enums.

| **Enum method** | **Description** |
| --- | --- |
| Format | Converts the specified value of enum type to the specified string format. |
| GetName | Returns the name of the constant of the specified value of specified enum. |
| GetNames | Returns an array of string name of all the constant of specified enum. |
| GetValues | Returns an array of the values of all the constants of specified enum. |
| object Parse(type, string) | Converts the string representation of the name or numeric value of one or more enumerated constants to an equivalent enumerated object. |
| bool TryParse(string, out TEnum) | Converts the string representation of the name or numeric value of one or more enumerated constants to an equivalent enumerated object. The return value indicates whether the conversion succeeded. |

Example: enum methods

enum WeekDays

{

Monday,

Tuesday,

Wednesday,

Thursday,

Friday,

Saturday,

Sunday

}

Console.WriteLine(Enum.GetName(typeof(WeekDays), 4));

Console.WriteLine("WeekDays constant names:");

foreach (string str in Enum.GetNames(typeof(WeekDays)))

Console.WriteLine(str);

Console.WriteLine("Enum.TryParse():");

WeekDays wdEnum;

Enum.TryParse<WeekDays>("1", out wdEnum);

Console.WriteLine(wdEnum);

Output:

Friday   
WeekDays constant names:  
Monday   
Tuesday   
Wednesday   
Thursday  
Friday  
Saturday  
Sunday   
Enum.TryParse():   
Tuesday

Visit MSDN to know more about [C# enum methods](https://msdn.microsoft.com/en-us/library/system.enum.aspx).

https://www.tutorialsteacher.com/Content/images/bulb-glow.png Points to Remember :

1. The enum is a set of named constant.
2. The value of enum constants starts from 0. Enum can have value of any valid numeric type.
3. String enum is not supported in C#.
4. Use of enum makes code more readable and manageable.

  Edit  Delete

# Collection and generics in c#

| **Non-generic Collections** | **Usage** |
| --- | --- |
| [ArrayList](https://www.tutorialsteacher.com/csharp/csharp-arraylist) | ArrayList stores objects of any type like an array. However, there is no need to specify the size of the ArrayList like with an array as it grows automatically. |
| [SortedList](https://www.tutorialsteacher.com/csharp/csharp-sortedlist) | SortedList stores key and value pairs. It automatically arranges elements in ascending order of key by default. C# includes both, generic and non-generic SortedList collection. |
| [Stack](https://www.tutorialsteacher.com/csharp/csharp-stack) | Stack stores the values in LIFO style (Last In First Out). It provides a Push() method to add a value and Pop() & Peek() methods to retrieve values. C# includes both, generic and non-generic Stack. |
| [Queue](https://www.tutorialsteacher.com/csharp/csharp-queue) | Queue stores the values in FIFO style (First In First Out). It keeps the order in which the values were added. It provides an Enqueue() method to add values and a Dequeue() method to retrieve values from the collection. C# includes generic and non-generic Queue. |
| [Hashtable](https://www.tutorialsteacher.com/csharp/csharp-hashtable) | Hashtable stores key and value pairs. It retrieves the values by comparing the hash value of the keys. |
| BitArray | BitArray manages a compact array of bit values, which are represented as Booleans, where true indicates that the bit is on (1) and false indicates the bit is off (0). |

| **Generic Collections** | **Description** |
| --- | --- |
| [List<T>](https://www.tutorialsteacher.com/csharp/csharp-list) | Generic List<T> contains elements of specified type. It grows automatically as you add elements in it. |
| [Dictionary<TKey,TValue>](https://www.tutorialsteacher.com/csharp/csharp-dictionary) | Dictionary<TKey,TValue> contains key-value pairs. |
| [SortedList<TKey,TValue>](https://www.tutorialsteacher.com/csharp/csharp-generic-sortedlist) | SortedList stores key and value pairs. It automatically adds the elements in ascending order of key by default. |
| Hashset<T> | Hashset<T> contains non-duplicate elements. It eliminates duplicate elements. |
| Queue<T> | Queue<T> stores the values in FIFO style (First In First Out). It keeps the order in which the values were added. It provides an Enqueue() method to add values and a Dequeue() method to retrieve values from the collection. |
| Stack<T> | Stack<T> stores the values as LIFO (Last In First Out). It provides a Push() method to add a value and Pop() & Peek() methods to retrieve values. |

  Edit  Delete

# what is Index and Types in SQL ?

An index is an on-disk structure associated with a table or view that speeds retrieval of rows from the table or view. An index contains keys built from one or more columns in the table or view. These keys are stored in a structure (B-tree) that enables SQL Server to find the row or rows associated with the key values quickly and efficiently.

A table or view can contain the following types of indexes:

# Clustered

* + Clustered indexes sort and store the data rows in the table or view based on their key values. These are the columns included in the index definition. There can be only one clustered index per table, because the data rows themselves can be stored in only one order.
  + The only time the data rows in a table are stored in sorted order is when the table contains a clustered index. When a table has a clustered index, the table is called a clustered table. If a table has no clustered index, its data rows are stored in an unordered structure called a heap.

# Nonclustered

* + Nonclustered indexes have a structure separate from the data rows. A nonclustered index contains the nonclustered index key values and each key value entry has a pointer to the data row that contains the key value.
  + The pointer from an index row in a nonclustered index to a data row is called a row locator. The structure of the row locator depends on whether the data pages are stored in a heap or a clustered table. For a heap, a row locator is a pointer to the row. For a clustered table, the row locator is the clustered index key.
  + You can add nonkey columns to the leaf level of the nonclustered index to by-pass existing index key limits, and execute fully covered, indexed, queries. For more information, see [Create Indexes with Included Columns](https://docs.microsoft.com/en-us/sql/relational-databases/indexes/create-indexes-with-included-columns?view=sql-server-2017). For details about index key limits see [Maximum Capacity Specifications for SQL Server](https://docs.microsoft.com/en-us/sql/sql-server/maximum-capacity-specifications-for-sql-server?view=sql-server-2017).

Both clustered and nonclustered indexes can be unique. This means no two rows can have the same value for the index key. Otherwise, the index is not unique and multiple rows can share the same key value. For more information, see [Create Unique Indexes](https://docs.microsoft.com/en-us/sql/relational-databases/indexes/create-unique-indexes?view=sql-server-2017).

Indexes are automatically maintained for a table or view whenever the table data is modified.

See [Indexes](https://docs.microsoft.com/en-us/sql/relational-databases/indexes/indexes?view=sql-server-2017) for additional types of special purpose indexes.

# Indexes and Constraints

Indexes are automatically created when PRIMARY KEY and UNIQUE constraints are defined on table columns. For example, when you create a table with a UNIQUE constraint, Database Engine automatically creates a non-clustered index. If you configure a PRIMARY KEY, Database Engine automatically creates a clustered index, unless a clustered index already exists. When you try to enforce a PRIMARY KEY constraint on an existing table and a clustered index already exists on that table, SQL Server enforces the primary key using a nonclustered index.

For more information, see [Create Primary Keys](https://docs.microsoft.com/en-us/sql/relational-databases/tables/create-primary-keys?view=sql-server-2017) and [Create Unique Constraints](https://docs.microsoft.com/en-us/sql/relational-databases/tables/create-unique-constraints?view=sql-server-2017).

# How Indexes are used by the Query Optimizer

Well-designed indexes can reduce disk I/O operations and consume fewer system resources therefore improving query performance. Indexes can be helpful for a variety of queries that contain SELECT, UPDATE, DELETE, or MERGE statements. Consider the query SELECT Title, HireDate FROM HumanResources.Employee WHERE EmployeeID = 250 in the **AdventureWorks2012** database. When this query is executed, the query optimizer evaluates each available method for retrieving the data and selects the most efficient method. The method may be a table scan, or may be scanning one or more indexes if they exist.

When performing a table scan, the query optimizer reads all the rows in the table, and extracts the rows that meet the criteria of the query. A table scan generates many disk I/O operations and can be resource intensive. However, a table scan could be the most efficient method if, for example, the result set of the query is a high percentage of rows from the table.

When the query optimizer uses an index, it searches the index key columns, finds the storage location of the rows needed by the query and extracts the matching rows from that location. Generally, searching the index is much faster than searching the table because unlike a table, an index frequently contains very few columns per row and the rows are in sorted order.

The query optimizer typically selects the most efficient method when executing queries. However, if no indexes are available, the query optimizer must use a table scan. Your task is to design and create indexes that are best suited to your environment so that the query optimizer has a selection of efficient indexes from which to select. SQL Server provides the [Database Engine Tuning Advisor](https://docs.microsoft.com/en-us/sql/relational-databases/performance/database-engine-tuning-advisor?view=sql-server-2017) to help with the analysis of your database environment and in the selection of appropriate indexes.

  Edit  Delete

# Session Management in MVC ?

State management is very important and useful concept in Web application and also its equally very important that resources can not be allocated unnecessarily. In ASP.NET MVC we can manage the session controller specific which helps to disable the session when you don't require IT for particular controller and due to this we can improve the performance of an application by freeing resources not needed to be allocated. Now you might be thinking that how it will be possible ?, So its possible with the help of controller attribute SessionState.  
  
Now let us learn about the SessionState attribute in brief so it will be useful to understand easily.

What is SessionStateSession State is the attribute of controller class which is used to control or manage the default session behavior.To use session state attribute we need to use System.Web.SessionState namespace.  
  
The following are the behavior of the Session State attribute

1. Default
2. Disabled
3. Required
4. ReadOnly

Let's learn about them in brief,

* DefaultThe default behavior of the session state checks for the session when user request comes to the controller with the help of IHttpHandler interface.  
    
  **Example**
  1. [SessionState(SessionStateBehavior.Default)]
  2. public class HomeController : Controller
  3. {
  4. // Action methods
  6. }
* DisabledWhen SessionState behavior set to disable then it will not check the session when user request comes to the controller and it disabled the session for entire controller and their views .  
    
  This property is very important when you want to disable the session for controller specific where there is no need to maintain the session.  
    
  **Example:**
  1. [SessionState( SessionStateBehavior.Disabled)]
  2. //controller
  3. public class HomeController : Controller
  4. {
  5. // Action methods
  7. }
* RequiredWhen the behavior is set to required then every request check for the session and it enabled the session for particular controller. This behavior is used when you want session for every user request.  
    
  **Example**
  1. [ SessionState( SessionStateBehavior.Required)]
  2. //controller
  3. public class HomeController : Controller
  4. {
  5. // Action methods
  7. }
* ReadOnlyWhen Session state behavior set to ReadOnly then session state can not be modified and updated. This behavior useful to make sure the session state will not be modified.  
    
  **Example**
  1. [ SessionState( SessionStateBehavior.ReadOnly)]
  2. //controller
  3. public class HomeController : Controller
  4. {
  5. // Action methods
  7. }

I hope you have understand about the SessionState from preceding brief summary. Now let's implement it practically, How to manage session at controller specific   
Step 1 : [Create an MVC application](http://www.compilemode.com/2015/09/how-to-create-aspnet-mvc-application.html).

1. "*Start*", then "All *Programs*" and select "Microsoft Visual Studio 2015".
2. "*File*", then "*New*" and click "*Project*" then select "*ASP.NET Web Application Template*", then provide the Project a name as you wish and click on *OK.*
3. Choose MVC empty application option and click on OK

Step 2: Add model class.  
  
Right click on Model folder in the created MVC application and add class named **EmpModel** and right the following lines of code as.

  Edit

# Base Class /Derived Class

\* We can create a base class object by initiating derived class also Eg: BClass s=new DClasss();

\* Only we can access base class oject when we define like above

\* When we create object for dervied class by initiating base base class it will show type cast err.

Eg: Dclass s=new Bclass();

# Static Class

\* Only static members /variables/methods can only access in static class.

\* Static should not allocate memory .

\* We cannot create an object for the static class.

\* Static classs cannot be used in inheritance concepts.

# Dynamic Polymorphism

\* Method polymorphism Virtual key word can be used when we implement the method so we can over ride the method .

\* When we call the

# Constructor

\* A special method of class that will be automatically invoked we an instance of the class created.

\* To Initialize private feild of class while creating instance of the class.

\* If you not initaing constructor for the class by default it will consider as default constructor

private feild should be string as null,int as 0.

\* A class can consist of any type of constructor

\* static constructor will not be a parameterized constructor

\* We cannot create more than one static constructor in a class.

\* Constructor doest not be a return type .we can just initailize the value for the feilds.

# Type Of Constructor

\* Default

\* parametrized

\* Copy constructor

\* Static

\* Private

# Default Constructor

\* A constructor which doesnt have any input parameter are called default constructor.

\* Default values can be assigned in default constructor class.

\* By defualt all int feild should be consider as 0 and string values consider as null

# Paramertized constructor

\* Constructor which consist of atleast single input parameter

\* We can assign the different value to variable when ever class initaited.

\* while initalizing the class we should pass the input parameter.

# Private Constructor

\* Once we denote the private constructor we should not able to create an object for the class.

\* Only static member and static variable can be decalre and access

\* Private constructor we should not able to pass any input parameter

# Static Constructor

\* Static constructors can‘t contain any access modifiers.

\* Static constructors can‘t be defined with arguments.

\* Static constructors can‘t access the non-static data members.

# Copy Constructor

The constructor which creates an object by copying variables from another object is called a copy constructor.

The purpose of a copy constructor is to initialize a new instance to the values of an existing instance.

# Dependency injection

\* Its software desgin pattern which is loosely coupled reusable code