In this video we will discuss **Attribute Routing** introduced in ASP.NET Web API 2   
  
Let us understand attribute routing with an example   
  
  
  
**Step 1 :** Create a new ASP.NET Web API project. Name it WebAPI  
  
**Step 2 :**Right click on the **"Models"**folder and add a class file. Name it **Student.cs**. Copy and paste the following code.

namespace WebAPI.Models

{

    public class Student

    {

        public int Id { get; set; }

        public string Name { get; set; }

    }

}

**Step 3 :** Let's now add **Students Controller**. Right click on the Controllers folder and add a new Web API 2 Empty controller. Name it **StudentsController.cs**. Copy and paste the following code.

using System.Collections.Generic;

using System.Linq;

using System.Web.Http;

using WebAPI.Models;

namespace WebAPI.Controllers

{

    public class StudentsController : ApiController

    {

        static List<Student> students = new List<Student>()

        {

            new Student() { Id = 1, Name = "Tom" },

            new Student() { Id = 2, Name = "Sam" },

            new Student() { Id = 3, Name = "John" }

        };

        public IEnumerable<Student> Get()

        {

            return students;

        }

        public Student Get(int id)

        {

            return students.FirstOrDefault(s => s.Id == id);

        }

        public IEnumerable<string> GetStudentCourses(int id)

        {

            if (id == 1)

                return new List<string>() { "C#", "ASP.NET", "SQL Server" };

            else if (id == 2)

                return new List<string>() { "ASP.NET Web API", "C#", "SQL Server" };

            else

                return new List<string>() { "Bootstrap", "jQuery", "AngularJs" };

        }

    }

}

In Web API 1, we had **convention-based routing** that define routes using route templates. When we create a new Web API project using Visual Studio, a default route is created in **WebApiConfig.cs** file. The default route is shown below

config.Routes.MapHttpRoute(

    name: "DefaultApi",

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

    defaults: new { id = RouteParameter.Optional }

);

So with the above default route and **StudentsController** in place   
/api/students is mapped to Get() method in StudentsController  
  
When we navigate to /api/students/1 we get the following exception message  
Multiple actions were found that match the request:   
   Get on type WebAPI.Controllers.StudentsController   
   GetStudentCourses on type WebAPI.Controllers.StudentsController  
  
This is because the framework does not know which of the 2 following action methods to map to the URI /api/students/1  
Get(int id)  
GetStudentCourses(int id)  
  
This can be very easily resolved by using **Attribute Routing**. Here is what we want the framework to do  
1. URI /api/students/1 should be mapped to **Get(int id)**. This method returns the student by id.  
2. URI /api/students/1/courses should be mapped to **GetStudentCourses(int id)**. This method returns the student courses by student id.  
  
To achieve the above, simply decorate GetStudentCourses() action method with the below [Route] attribute

[Route("api/students/{id}/courses")]

At this point build the solution and navigate to /api/students/1. Notice you now get student details whose id=1. When you navigate to /api/students/1/courses you get all the courses into which student with id=1 is enrolled.  
  
**What is Attribute Routing**  
Using the [Route] attribute to define routes is called Attribute Routing  
  
**What are the advantages of using Attribute Routing**  
Attribute routing gives us more control over the URIs than convention-based routing. Creating URI patterns like hierarchies of resources (For example, students have courses, Departments have employees) is very difficult with convention-based routing. With attribute routing all you have to do is use the [Route] attribute as shown below.  
  
[Route("api/students/{id}/courses")]  
  
**How to enable Attribute Routing**  
In ASP.NET Web API 2, Attribute Routing is enabled by default. The following line of code in WebApiConfig.cs file enables Attribute Routing.  
  
config.MapHttpAttributeRoutes();  
  
**Can we use both Attribute Routing and Convention-based routing in a single Web API project**  
Yes, both the routing mechanisms can be combined in a single Web API project. The controller action methods that have the [Route] attribute uses Attribute Routing, and the others without [Route] attribute uses Convention-based routing.