In this video, we will discuss the use of **NonAction**attribute in asp.net mvc  
  
**The following questions could be asked in an interview**  
What is the use of NonAction attribute in MVC?  
OR  
How do you restrict access to public methods in a controller?   
  
   
  
**An action method is a public method in a controller that can be invoked using a URL.** So, by default, if you have any public method in a controller then it can be invoked using a URL request. To restrict access to public methods in a controller, NonAction attribute can be used. Let's understand this with an example.  
  
   
  
**We have 2 public methods in HomeController, Method1() and Method2().**  
**Method1** can be invoked using URL **/Home/Method1**  
**Method2** can be invoked using URL **/Home/Method2**  
public class HomeController : Controller  
{  
    public string Method1()  
    {  
        return "<h1>Method 1 Invoked</h1>";  
    }  
  
    public string Method2()  
    {  
        return "<h1>Method 2 Invoked</h1>";  
    }  
}  
  
Let's say **Method2**() is for doing some internal work, and we don't want it to be invoked using a URL request. To achieve this, decorate **Method2**() with **NonAction** attribute.  
[NonAction]  
public string Method2()  
{  
    return "<h1>Method 2 Invoked</h1>";  
}  
  
Now, if you naviage to URL **/Home/Method2**, you will get an error - The resource cannot be found.  
  
**Another way to restrict access to methods in a controller, is by making them private.**  
private string Method2()  
{  
    return "<h1>Method 2 Invoked</h1>";  
}  
  
In general, it's a bad design to have a public method in a controller that is not an action method. If you have any such method for performing business calculations, it should be somewhere in the model and not in the controller.  
  
However, if for some reason, if you want to have public methods in a controller and you don't want to treat them as actions, then use NonAction attribute.