In this video we will discuss, **including and excluding properties from model binding using interfaces.** Please watch [Part 21](http://csharp-video-tutorials.blogspot.com/2013/05/part-21-including-and-excluding.html), before proceeding.   
  
In [part 20](http://csharp-video-tutorials.blogspot.com/2013/05/part-20-preventing-unintended-updates.html), we have seen how to include and exclude properties from model binding, by passing a string array to **UpdateModel**() method, and in [part 21](http://csharp-video-tutorials.blogspot.com/2013/05/part-21-including-and-excluding.html) we have seen achieving the same using **"BIND"** attribute.

**To include and exclude properties from model binding using interfaces**  
**Step 1:** Create an interface **"IEmployee"** as shown below. Notice that this interface, has got only the properties that we want to include in model binding. **"Name"** property is not present. This means, **"Name"** property will be excluded from model binding. Copy and paste this code in **"Employee.cs"** class file in **"BusinessLayer"** project  
public interface IEmployee  
{  
    int ID { get; set; }  
    string Gender { get; set; }  
    string City { get; set; }  
    DateTime? DateOfBirth { get; set; }  
}  
  
**Step 2:** Make **"Employee"** class inherit from **IEmployee** interface  
public class Employee : IEmployee  
{  
    public int ID { get; set; }  
    public string Name { get; set; }  
    [Required]  
    public string Gender { get; set; }  
    [Required]  
    public string City { get; set; }  
    [Required]  
    public DateTime? DateOfBirth { get; set; }  
}  
  
**Step 3:** Modify **"Edit\_Post()"** controller action method that is present in **"EmployeeController.cs"** file, as shown below.  
[HttpPost]  
[ActionName("Edit")]  
public ActionResult Edit\_Post(int id)  
{  
    EmployeeBusinessLayer employeeBusinessLayer = new EmployeeBusinessLayer();  
    Employee employee = employeeBusinessLayer.Employees.Single(x => x.ID == id);  
    UpdateModel<IEmployee>(employee);  
  
    if (ModelState.IsValid)  
    {  
        employeeBusinessLayer.SaveEmmployee(employee);  
        return RedirectToAction("Index");  
    }  
  
    return View(employee);  
}  
  
Notice that we are explicitly calling the model binder, by calling **UpdateModel**() function passing in our interface **IEmployee**. The model binder will update only the properties that are present in the interface.  
  
So, if we were to generate a post request using fiddler as we did in the previous session, **"Name"** property of the **"Employee"** object will not be updated.  
  
So, in short, there are several ways to **include** and **exclude** properties from **Model Binding**. Depending on the architecture and requirements of your project, you may choose the approach that best fit your needs.