In this video we will discuss, customizing the auto-generated edit view. Please [watch Part 27](http://csharp-video-tutorials.blogspot.com/2013/05/part-27-customizing-autogenerated.html), before proceeding.   
  
If you want **"Select Department"** as the first item in the **"Department"** dropdownlist on **"Edit"** view, then,   
**REPLACE THE FOLLOWING LINE**  
@Html.DropDownList("DepartmentId", String.Empty)  
  
**WITH**  
@Html.DropDownList("DepartmentId", "Select Department")

Notice that, a textbox is used for gender. It is ideal to have a dropdownlist for gender rather than a textbox. To achieve this, make the following changes to **"Edit.cshtml"**view.  
  
**REPLACE THE FOLLOWING CODE**  
@Html.EditorFor(model => model.Gender)  
  
**WITH**  
@Html.DropDownList("Gender", new List<SelectListItem>  
    {  
    new SelectListItem { Text = "Male", Value="Male" },  
    new SelectListItem { Text = "Female", Value="Female" }  
    }, "Select Gender")  
  
Let's make **"Name"** non-editable. To achieve this  
**CHANGE THE FOLLOWING CODE IN EDIT.CSHTML**  
@Html.EditorFor(model => model.Name)  
  
**TO**  
@Html.DisplayFor(model => model.Name)  
@Html.HiddenFor(model => model.Name)  
  
At this point, we will still be able to change **"Name"** property of the employee, using tools like fiddler. We discussed this in [Part 19](http://csharp-video-tutorials.blogspot.com/2013/05/part-19-unintended-updates-in-mvc.html) of this [video series](http://www.youtube.com/playlist?list=PL6n9fhu94yhVm6S8I2xd6nYz2ZORd7X2v). There are several ways to prevent **"Name"** property from being updated.  
**1.** Use **UpdateModel**() function and pass include and exclude list as a parameter - Discussed in [Part 20](http://csharp-video-tutorials.blogspot.com/2013/05/part-20-preventing-unintended-updates.html)  
**2.** Use Bind attribute - Discussed in [Part 21](http://csharp-video-tutorials.blogspot.com/2013/05/part-21-including-and-excluding.html)  
**3.** Use interfaces - Discussed in [Part 22](http://csharp-video-tutorials.blogspot.com/2013/05/part-22-including-and-excluding.html)  
  
Now, let's discuss using **BIND**attribute to prevent updating **"Name"** property using tools like fiddler. Along the way, I will demonstrate adding model validation errors dynamically.  
  
Change the implementation of **"Edit"** controller action method, that responds to [HttpPost] request as shown below  
[HttpPost]  
public ActionResult Edit([Bind(Exclude="Name")] Employee employee)  
{  
    Employee employeeFromDB = db.Employees.Single(x => x.EmployeeId == employee.EmployeeId);  
  
    employeeFromDB.EmployeeId = employee.EmployeeId;  
    employeeFromDB.Gender = employee.Gender;  
    employeeFromDB.City = employee.City;  
    employeeFromDB.DepartmentId = employee.DepartmentId;  
    employee.Name = employeeFromDB.Name;  
  
    if (ModelState.IsValid)  
    {  
        db.ObjectStateManager.ChangeObjectState(employeeFromDB, EntityState.Modified);  
        db.SaveChanges();  
        return RedirectToAction("Index");  
    }  
    ViewBag.DepartmentId = new SelectList(db.Departments, "Id", "Name", employee.DepartmentId);  
    return View(employee);  
}  
  
**Please note** that, we have excluded **"Name"** property from model binding using **"Bind"**attribute. Even without **BIND** attribute, users will not be able to change the **"NAME"** of the employee, as we are copying only the required properties(Excluding NAME property) from **"employee"** object to **"employeeFromDB"** which in turn is persisted to the database. Since, I want to demonstrate adding model validation errors dynamically, let the attribute be there.  
  
At this point if we run the application and click on "Save" button on "Edit" view, we get a validation error stating - The Name field is required. This is because "Name" property is decorated with [Required] attribute in "Employee.cs" file. To prevent the validation error, remove the [Required] attribute.

The problem with this change is that, **"Name"** field on **"Create"** view is no longer mandatory. This means we will now be able to create a new employee, without **NAME**. To fix the **"Create"** view, let's add model validation errors dynamically. Change the implementation of **"Create"** controller action method that responds to [HttpPost] request as shown below.  
[HttpPost]  
public ActionResult Create(Employee employee)  
{  
    if (string.IsNullOrEmpty(employee.Name))  
    {  
        ModelState.AddModelError("Name", "The Name field is required.");  
    }  
              
    if (ModelState.IsValid)  
    {  
        db.Employees.AddObject(employee);  
        db.SaveChanges();  
        return RedirectToAction("Index");  
    }  
  
    ViewBag.DepartmentId = new SelectList(db.Departments, "Id", "Name", employee.DepartmentId);  
    return View(employee);  
}