Entity Framework / Validation Model

```
namespace MyApp.Model {
    public class Employee
    {
        [Key]
        [DatabaseGeneratedAttribute(
            DatabaseGeneratedOption.Identity)]
        public int EmployeeId { get; set; }

        [Required]
        public string FirstName { get; set; }

        [Required]
        public string FamilyName { get; set; }

        [EmailAddress]
        public string Email { get; set; }

        [RegularExpression("\\d+\\.\\d+\\.\\d+\\.\\d+",
            ErrorMessage = "Invalid IP address format")]
        public string IpAddress { get; set; }

        public virtual Department Department { get; set; }

        [InverseProperty("Author")]
        public virtual ICollection<Message> Messages { get; set; }

        public bool IsFullTime { get; set; }

        [Range(1, 12)]
        public int BirthMonth { get; set; }

        [Timestamp]
        public byte[] RowVersion { get; set; }

        [NotMapped]
        public string FullName
        {
            get { return FirstName + " " + FamilyName; }
        }
    }
}
```

Database primary key

Key is automatically generated by incrementing a counter

Property value cannot be null / empty

Property value must be formatted as an email address

Property value must match the supplied regular expression

Declare navigation properties to be virtual if you want lazy loading

This list contains all Messages with an *Author* property that is this Employee

Property value must be between 1 and 12 (inclusive)

This property is a timestamp used for optimistic concurrency control

This property is to be ignored by Entity Framework

Razor View

```
@using MyApp.Models
@model Employee

@{
    Layout = "~/Views/_Base.cshtml";
    ViewBag.Title = "Employee Information";
}

<h2>Employee Information</h2>
<p>These are the details of @Model.FullName:</p>
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary()
    <div>
        First Name: @Html.TextBoxFor(x => x.FirstName)
    </div>
    <div>
        Family Name: @Html.TextBoxFor(x => x.FamilyNa
    </div>
    <div>
        Birth Month: @Html.TextBoxFor(x => x.BirthMon
    </div>
    <div>
        IP Address: @Html.TextBoxFor(x => x.IpAddress)
    </div>
    <div>
        Full Time:
        @Html.DropDownListFor(
            x => x.IsFullTime,
            new [] {
                new SelectListItem {
                    Text = "Full Time",
                    Value = "true",
                    Selected = Model.IsFullTime
                },
                new SelectListItem {
                    Text = "Part Time",
                    Value = "false",
                    Selected = !Model.IsFullTime
                }
            })
    </div>
    <input type="submit" value="Save Changes" />
```

Import namespace into Razor view

Declare a typed View

Use a master layout / template

Set a property of the ViewBag dynamic object

Write out a value

Create a HTML <form> tag

Generate a token to prevent Cross-Site-Forgery-Requests (CSFRs)

Show any validation errors

Create Text-boxes for Properties of the Model

Create a Drop-down list for IsFullTime property of the model

```
}

@* Don't show the messages section if there are no messages *@

@if (Model.Messages != null && Model.Messages.Count() > 0)
{
    <p>Messages for @Model.FullName:</p>
    <ul>
    @foreach (var message in Model.Messages)
    {
        <li>@message.MessageId</li>
    }
    </ul>
}

<a href="@Url.Action("Index")">List All Employees</a>
@Html.ActionLink("List All Employees", "Index")
@Html.ActionLink("Go to Home", "Index", "Home")
@Html.ActionLink("View Employee 1", "Details", new { id = 1 }, null)
@Html.Raw(" ")

@section Author {
    This page created by Ben.
}
```

Razor comments

If Statement

Loop

Manually create a URL to use in a link

Create a link referring to the Index action of the same controller

Create a link referring to the Index action of the HomeController

Create a link referring to the Details action with route/model parameters

Output a string directly as raw HTML, without any escaping

Define a section (to use in a template)

Template

```
<html>
    <title>@ViewBag.Title</title>
    <body>
        <h1>My App</h1>
        <div>
            @RenderBody()
        </div>
        <div>
            @if (IsSectionDefined("Author"))
            {
                @: Author Information:
                @RenderSection("Author")
            }
        </div>
    </body>
</html>
```

Render the main content of the file that is using this template

Switch back to plain HTML text when in a section of C# code

Render a section that was defined using @section

Controller

```
...
using System.Data.Entity;

namespace MyApp.Controllers
{
    [Authorize]
    public class EmployeeController : Controller
    {

        [AllowAnonymous]
        public ActionResult Index()
        {
            using (var context = new MyAppContext())
            {
                var emps = context.Employees
                    .Include(x => x.Messages).Include(x => x.Department)
                    .ToList();
                return View(emps);
            }
        }

        [AcceptVerbs(HttpVerbs.Get | HttpVerbs.Post)]
        [Authorize(Roles = "Administrator")]
        public ActionResult Delete(int id)
        {
            using (var context = new MyAppContext())
            {
                var employee = new Employee { EmployeeId = id };
                context.Employees.Attach(employee);
                context.Employees.Remove(employee);
                context.SaveChanges();
                return RedirectToAction("Index");
            }
        }
```

Contains the .Includes(…) extension method used for Eager loading

A user must be logged in to use the Actions on this controller

An exception to the need for authorization (no need to login)

Eager loading for view rendering (when there is no DbContext)

Show the strongly typed View matching this Action (Index.cshtml)

Accept HTTP verbs GET and POST

Require the user to belong to the Administrator Role

Redirect the user to the Index action

```
        [HttpGet]
        public ActionResult Edit(int id)
        {
            using (var context = new MyAppContext())
            {
                var emp = context.Employees
                    .Include(x => x.Messages).Include(x => x.Department)
                    .Where(x => x.EmployeeId == id).Single();
                return View(emp);
            }
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(int id, Employee employee)
        {
            if (ModelState.IsValid)
            {
                if (employee.IpAddress == "127.0.0.1")
                {
                    ModelState.AddModelError("", "Do not use 127.0.0.1");
                    return View(employee);
                }
                else
                {
                    using (var context = new MyAppContext())
                    {
                        var oldEmployee = Context.Employees.Find(id);
                        UpdateModel(oldEmployee);
                        context.SaveChanges();
                        return View("Details", employee);
                    }
                }
            }
            else
            {
                return View(employee);
            }
        }
    }
}
```

Use this action for GET requests (i.e., to retrieve an empty/current form)

Use this action when the user POSTs back a form (i.e., saves changes)

Check for the token created by @Html. AntiForgeryToken()

Use HTML Get/Post parameters to construct an Employee object

Were there any validation errors when creating the Employee object?

Add a custom validation error for @Html.ValidationSummary()

Use HTML Get/Post parameters to update the Employee object

Return the Details strongly typed view (Details.cshtml)