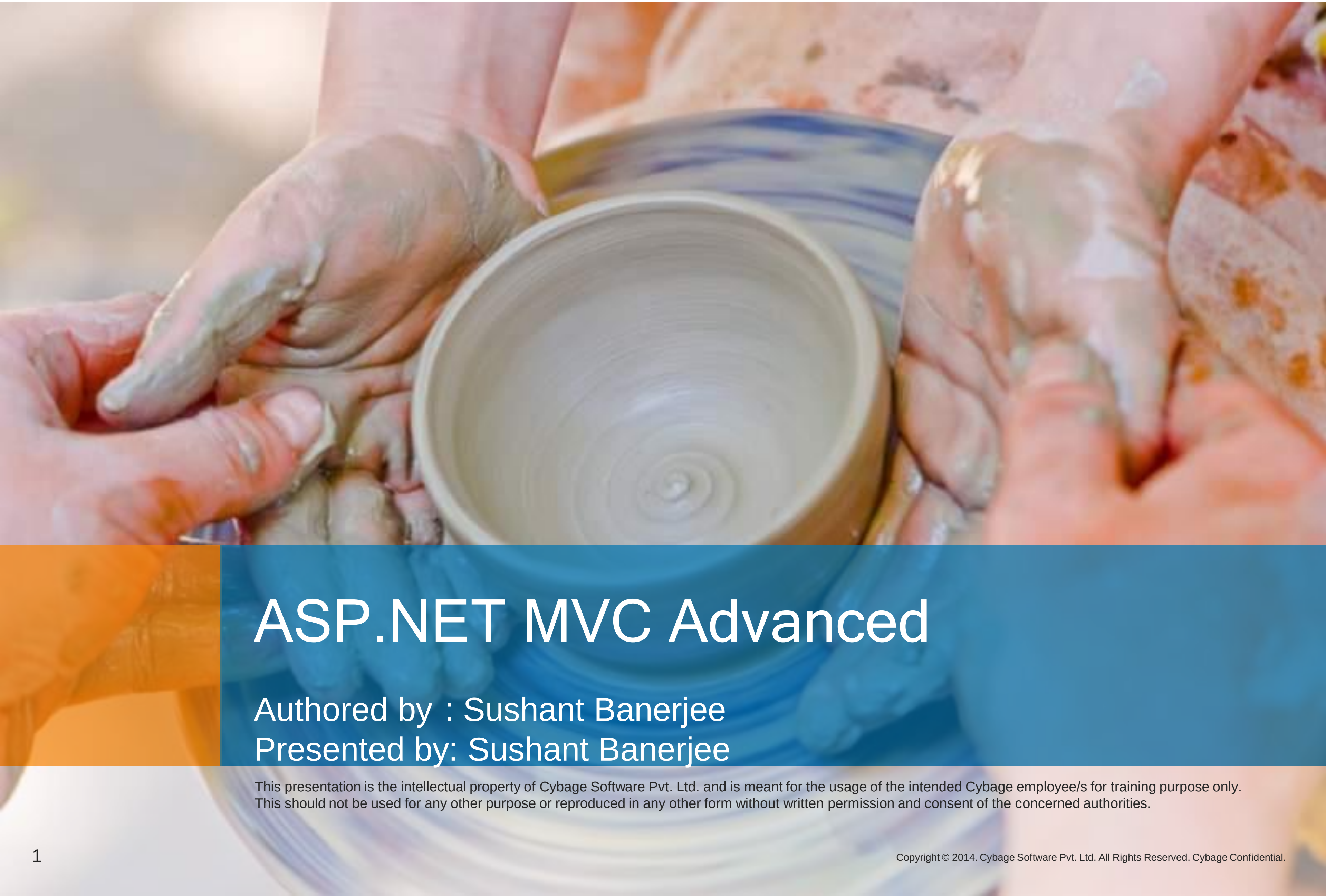


ASP.NET MVC Advanced

Authored by : Sushant Banerjee
Presented by: Sushant Banerjee

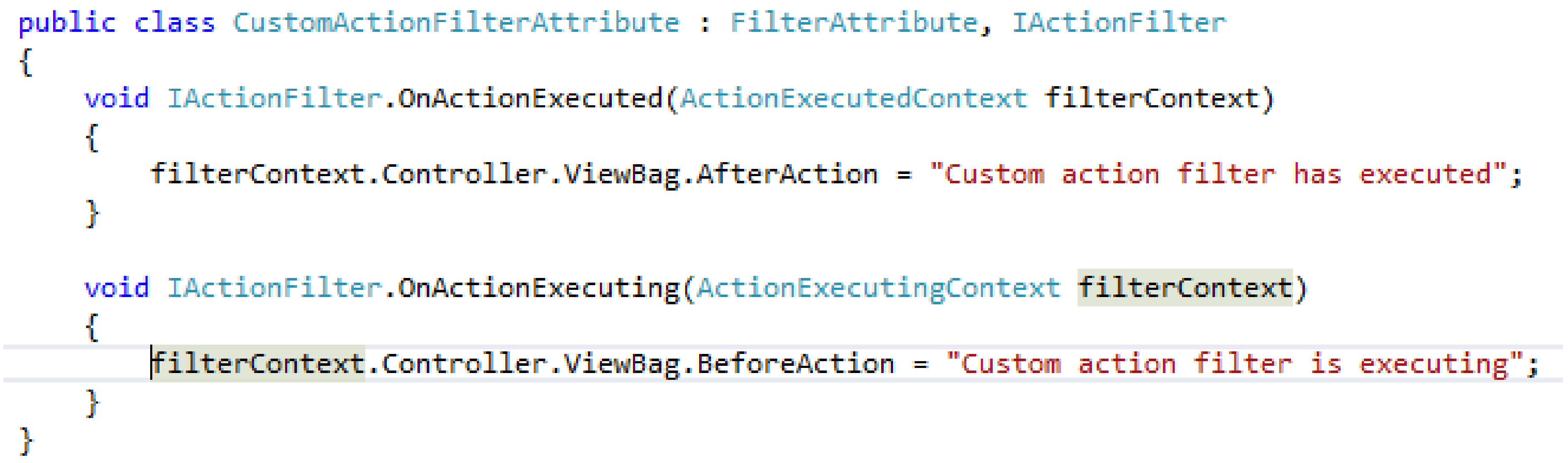
Agenda

- Custom Action Filters
- Custom HTML Helpers
- Custom Model Binders
- Dependency Resolution
- Caching
- Globalization
- Areas
- Asynchronous Controllers
- Diagnostics

MVC Filters

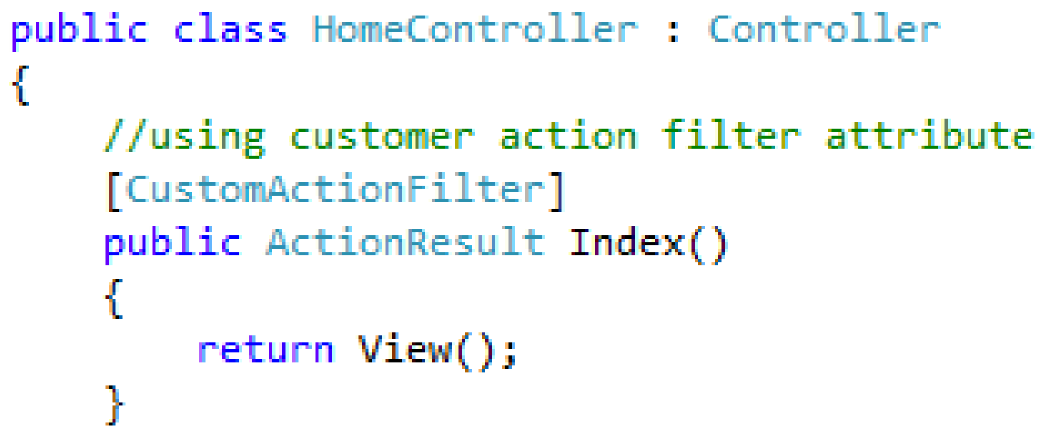
- Authorization Filter
- Action Filter
- Result Filter
- Exception Filter

Custom ActionFilters

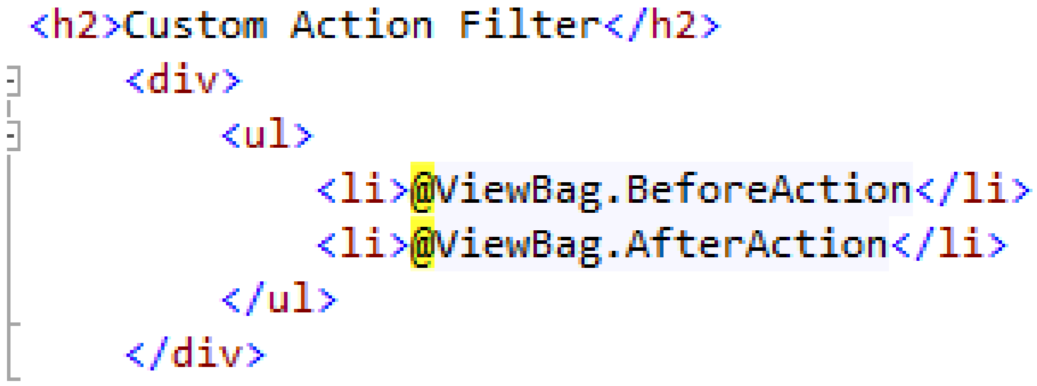


```
public class CustomActionFilterAttribute : FilterAttribute, IActionFilter
{
    void IActionFilter.OnActionExecuted(ActionExecutedContext filterContext)
    {
        filterContext.Controller.ViewBag.AfterAction = "Custom action filter has executed";
    }

    void IActionFilter.OnActionExecuting(ActionExecutingContext filterContext)
    {
        filterContext.Controller.ViewBag.BeforeAction = "Custom action filter is executing";
    }
}
```

Using Custom ActionFilter



```
public class HomeController : Controller
{
    //using customer action filter attribute
    [CustomActionFilter]
    public ActionResult Index()
    {
        return View();
    }
```

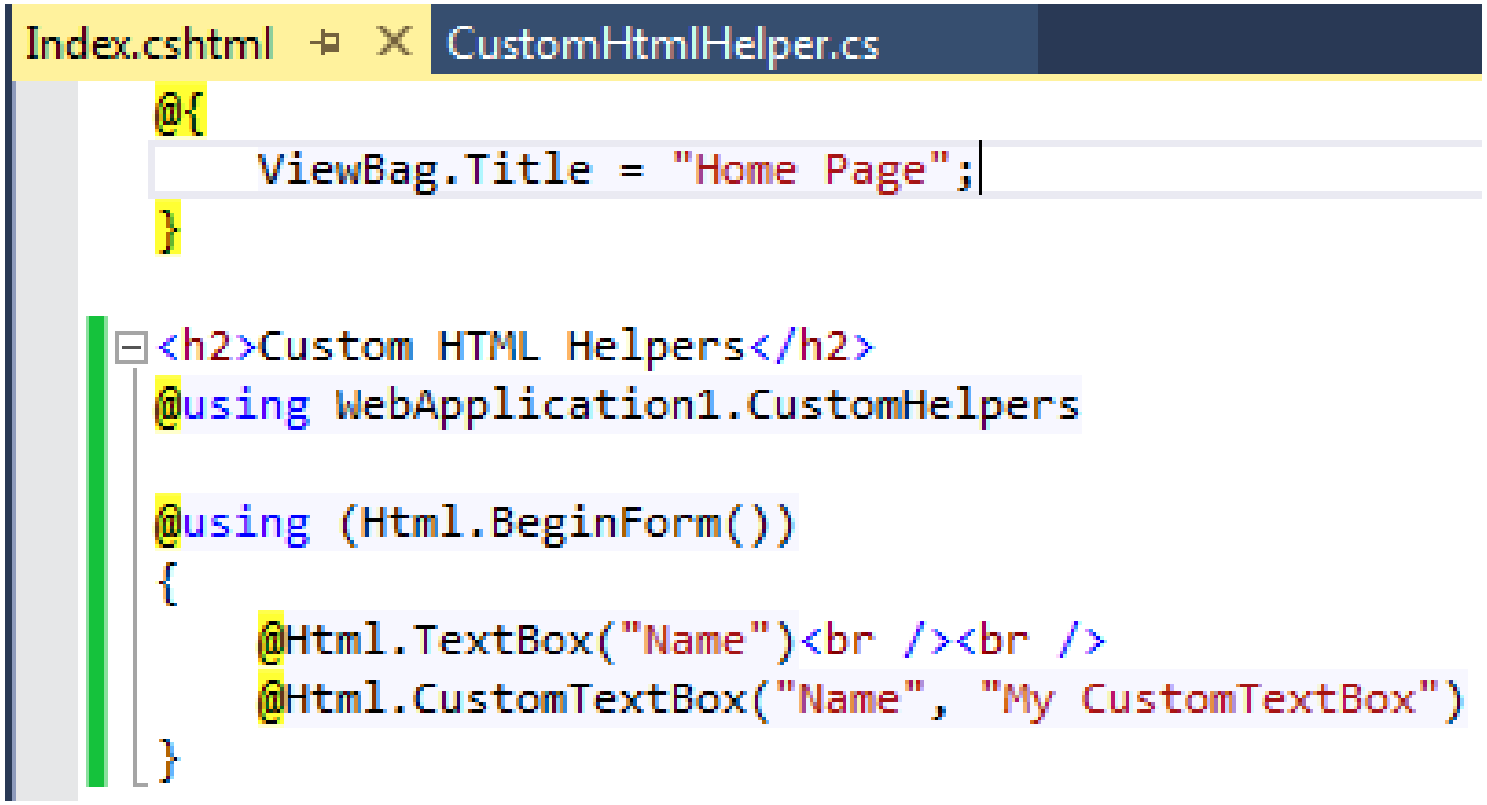


```
<h2>Custom Action Filter</h2>
    <div>
        <ul>
            <li>@ViewBag.BeforeAction</li>
            <li>@ViewBag.AfterAction</li>
        </ul>
    </div>
```

Custom HTML Helpers

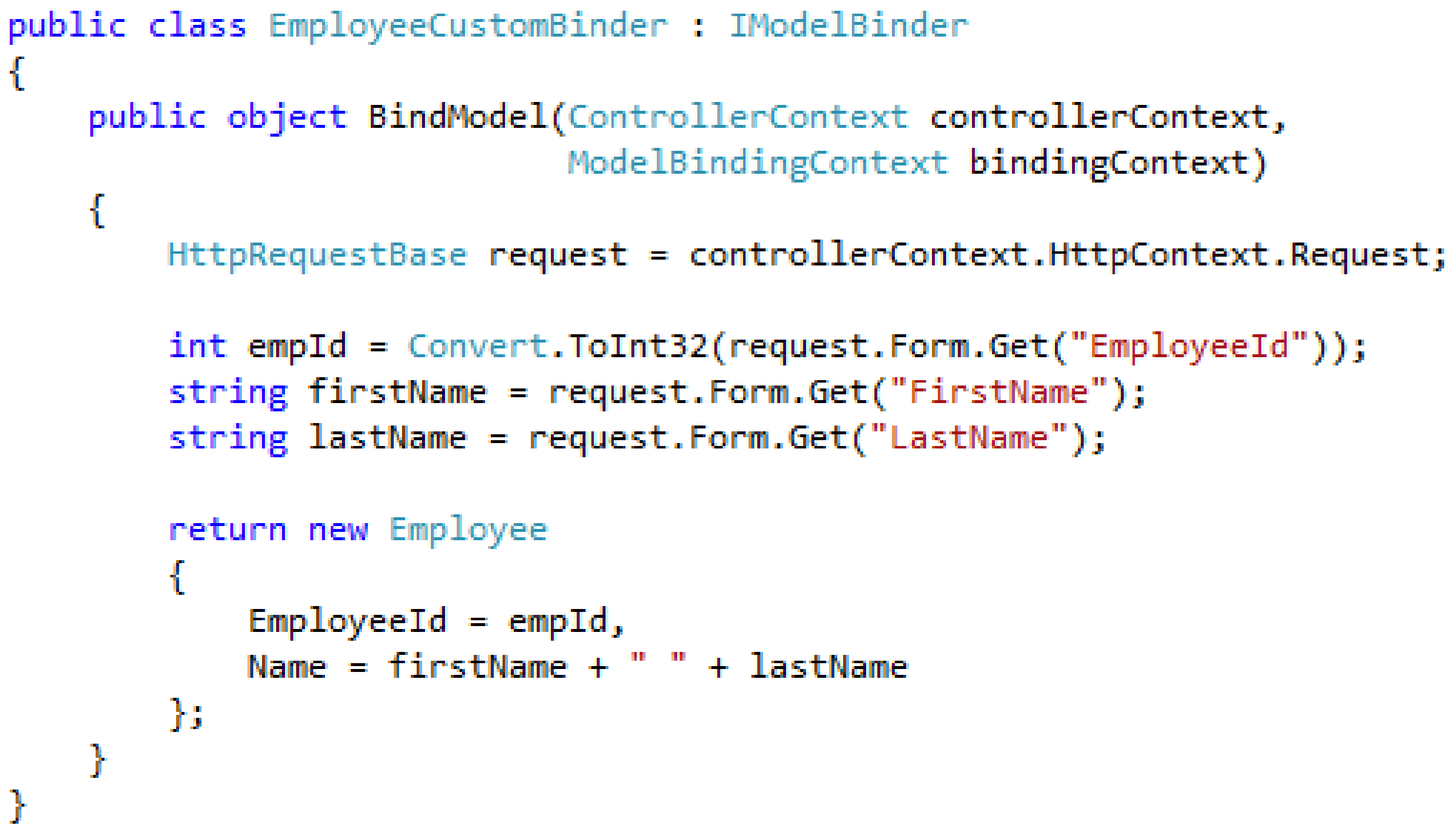
```
public static class CustomHtmlHelper
{
    //writing an extension method for html helper class
    public static MvcHtmlString CustomTextBox(this HtmlHelper htmlHelper, string name, string value)
    {
        var builder = new TagBuilder("input");
        builder.MergeAttribute("type", "text");
        builder.MergeAttribute("name", name);
        builder.MergeAttribute("value", value);
        return MvcHtmlString.Create(builder.ToString(TagRenderMode.SelfClosing));
    }
}
```

Using Custom Html Helper



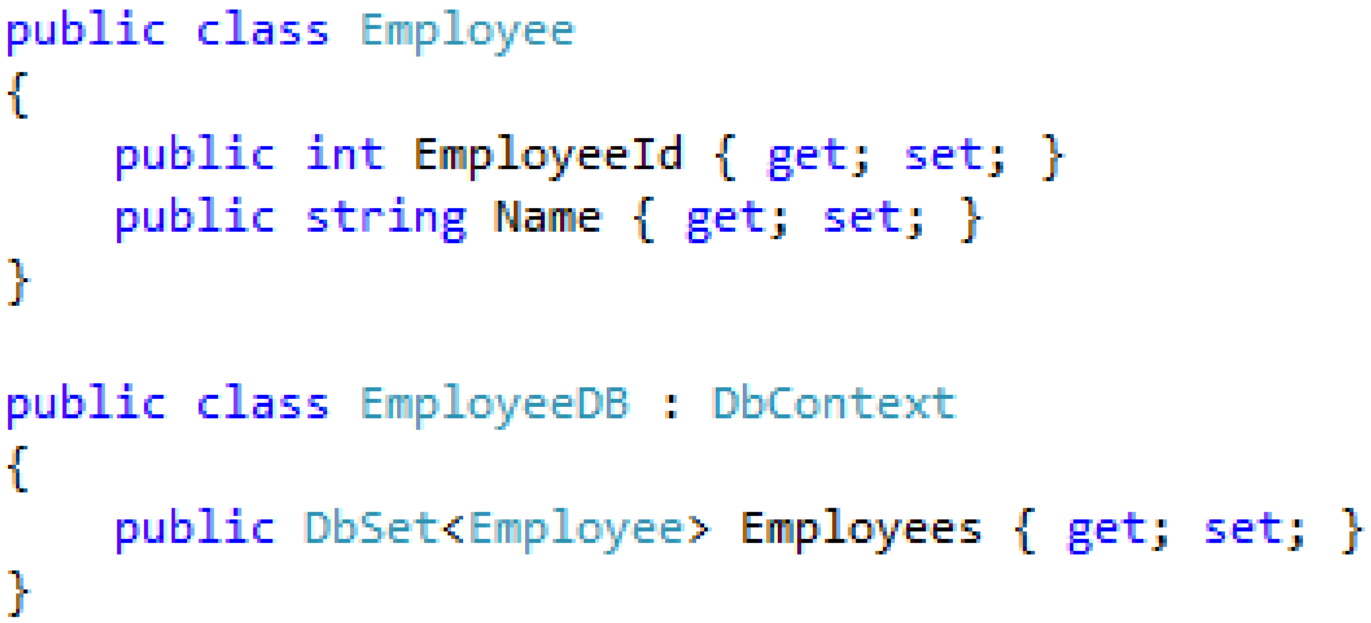
```
@{
    ViewBag.Title = "Home Page";
}

<h2>Custom HTML Helpers</h2>
@using WebApplication1.CustomHelpers

@using (Html.BeginForm())
{
    @Html.TextBox("Name")<br /><br />
    @Html.CustomTextBox("Name", "My CustomTextBox")
}
```

Custom Model Binder



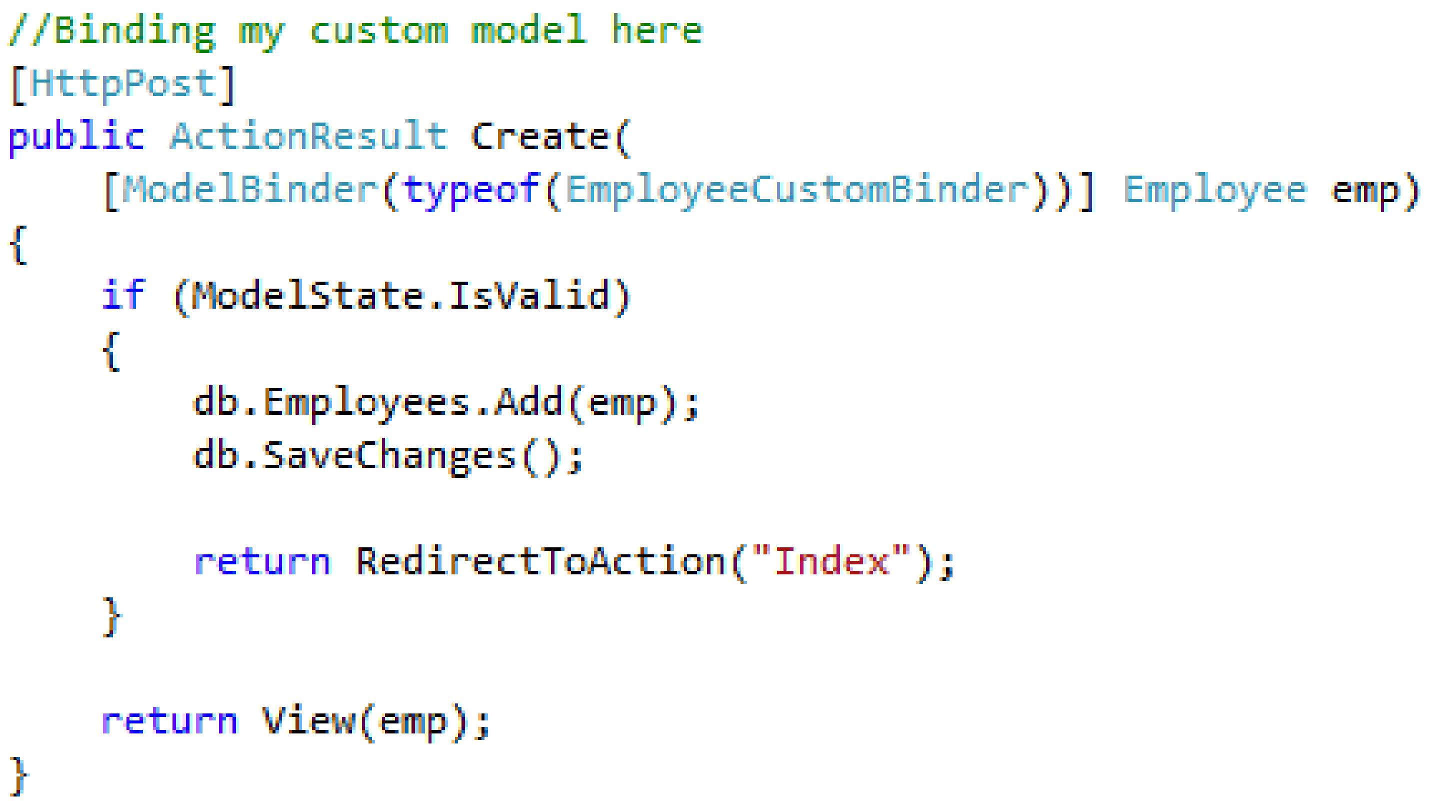
```
public class EmployeeCustomBinder : IModelBinder
{
    public object BindModel(ControllerContext controllerContext,
                            ModelBindingContext bindingContext)
    {
        HttpRequestBase request = controllerContext.HttpContext.Request;

        int empId = Convert.ToInt32(request.Form.Get("EmployeeId"));
        string firstName = request.Form.Get("FirstName");
        string lastName = request.Form.Get("LastName");

        return new Employee
        {
            EmployeeId = empId,
            Name = firstName + " " + lastName
        };
    }
}
```

Model Class



```
public class Employee
{
    public int EmployeeId { get; set; }
    public string Name { get; set; }
}

public class EmployeeDB : DbContext
{
    public DbSet<Employee> Employees { get; set; }
}
```

Binding Custom Model



```
//Binding my custom model here
[HttpPost]
public ActionResult Create(
    [ModelBinder(typeof(EmployeeCustomBinder))] Employee emp)
{
    if (ModelState.IsValid)
    {
        db.Employees.Add(emp);
        db.SaveChanges();

        return RedirectToAction("Index");
    }

    return View(emp);
}
```

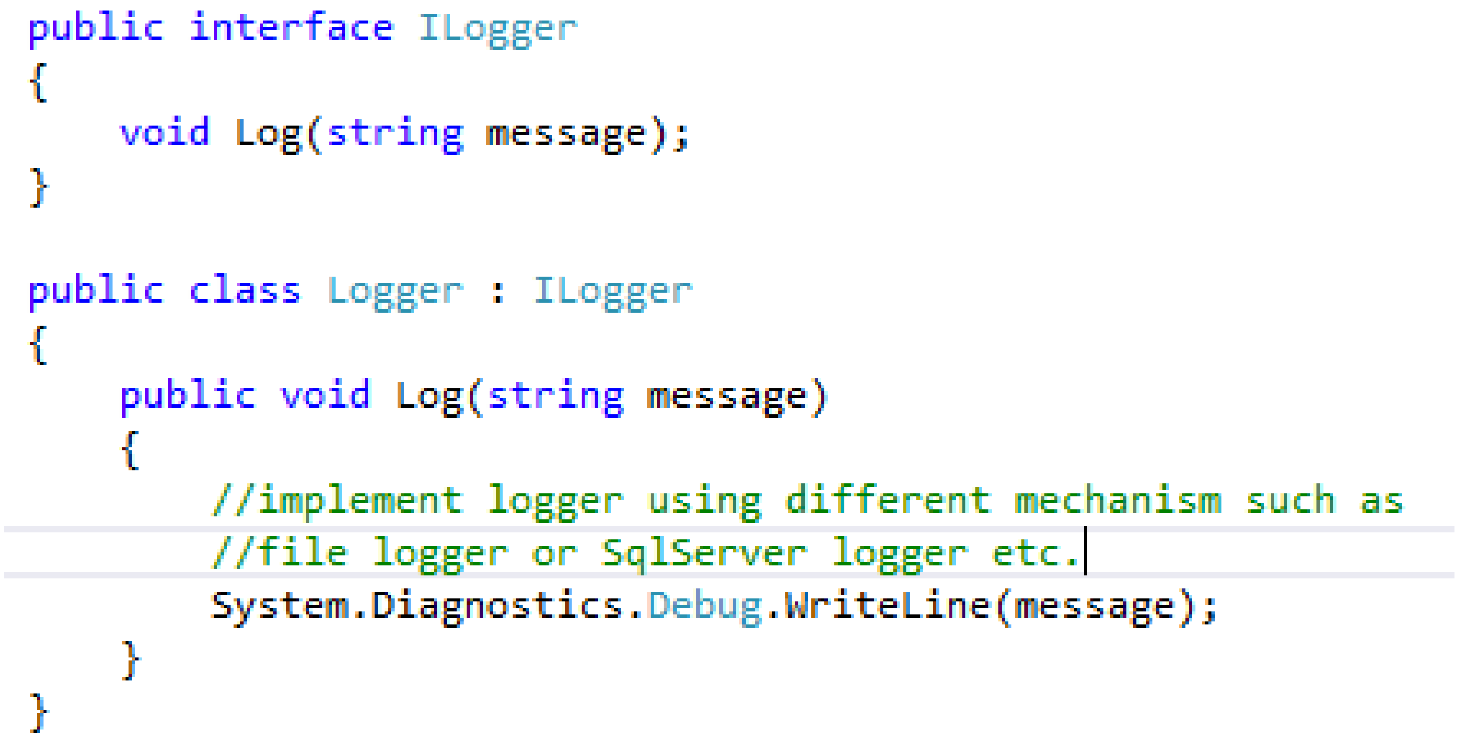
Dependency Injection

- The Dependency Injection is a design pattern
- Inversion of Control or IoC means
  - Objects do not create other objects on which they rely
  - Instead they get the objects from an outside source
- The DI is a implementation of IoC
- The Dependency Injection means
  - Objects are not created as that will be hardcoded
  - Instead framework components passes constructor parameters and set properties

Advantages of DI Pattern

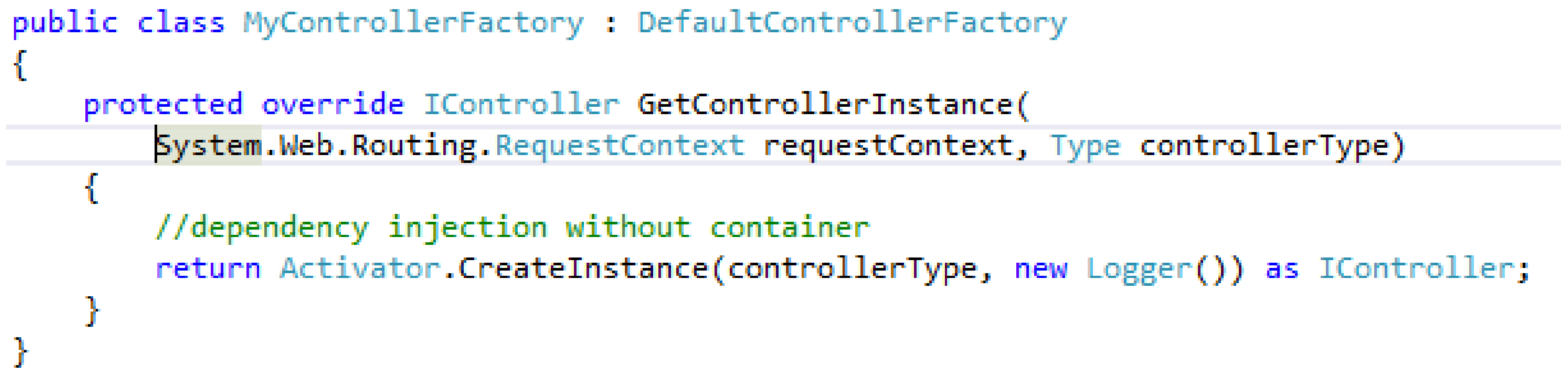
- Reduces class coupling
- Increases code reusing
- Improves code maintainability
- Improves application testing

Creating Interface



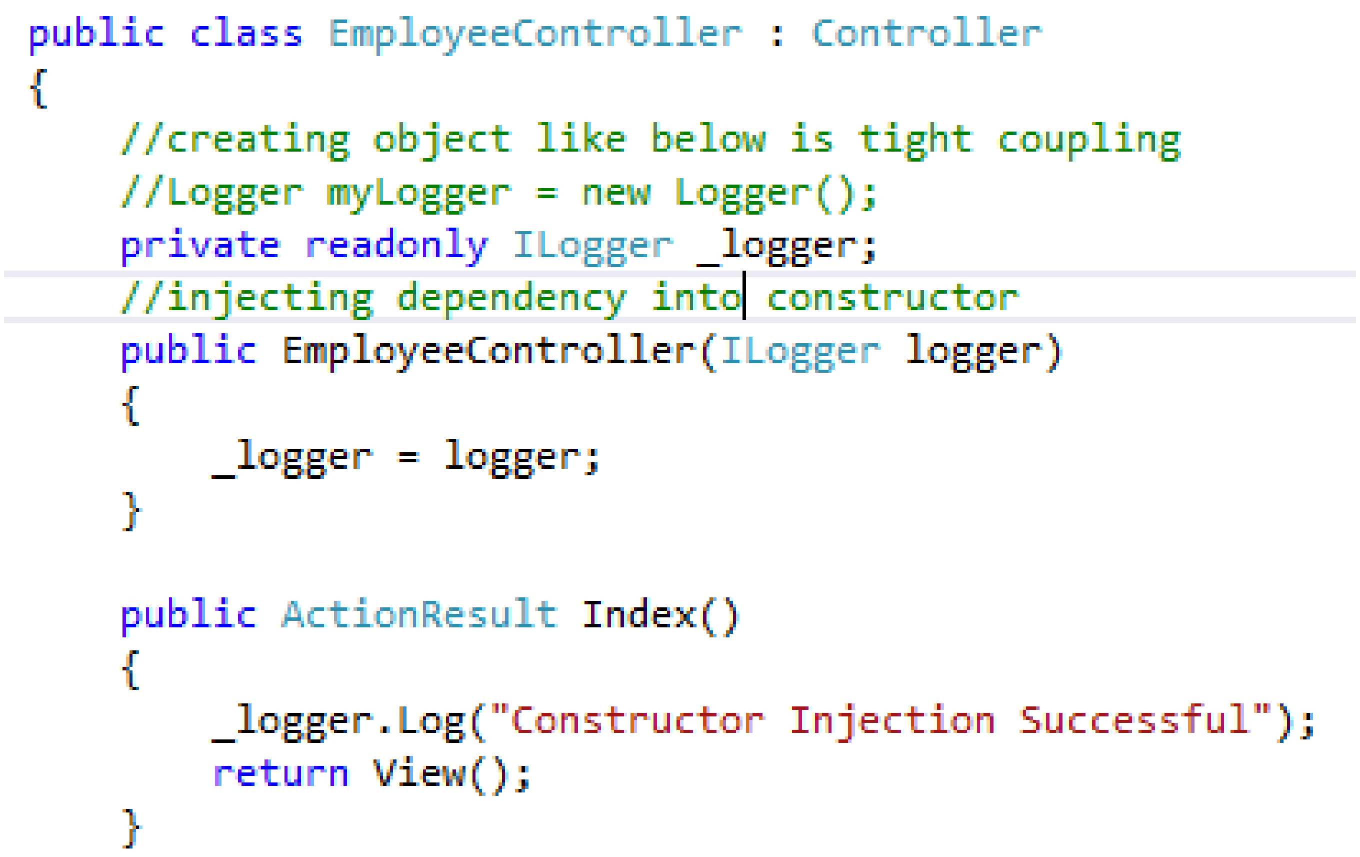
```
public interface ILogger
{
    void Log(string message);
}

public class Logger : ILogger
{
    public void Log(string message)
    {
        //implement logger using different mechanism such as
        //file logger or SqlServer logger etc.
        System.Diagnostics.Debug.WriteLine(message);
    }
}
```

Custom Controller Factory



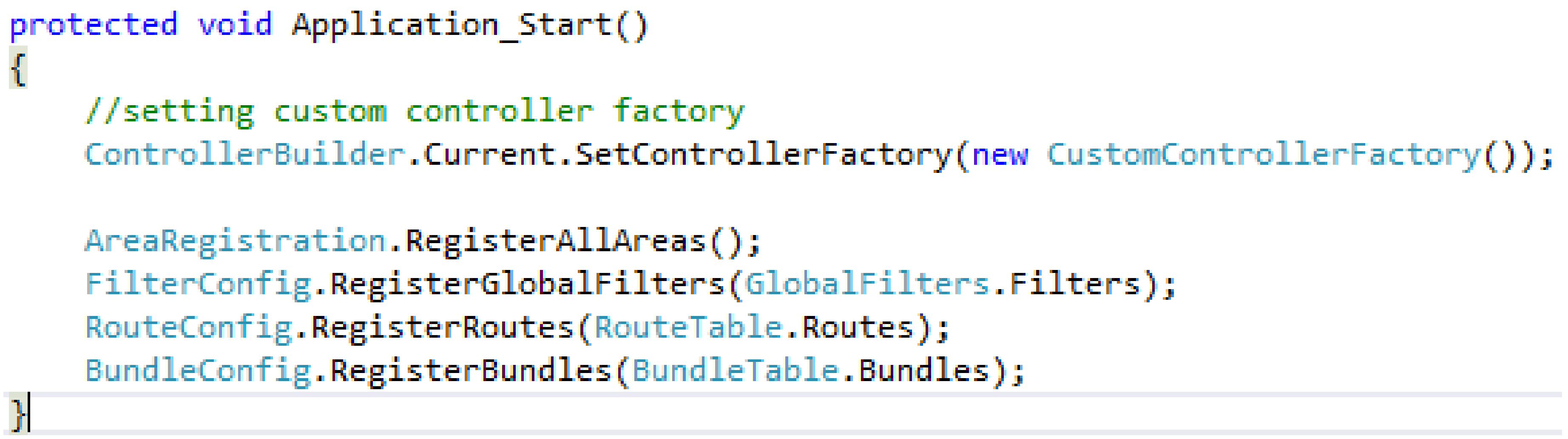
```
public class MyControllerFactory : DefaultControllerFactory
{
    protected override IController GetControllerInstance(
        System.Web.Routing.RequestContext requestContext, Type controllerType)
    {
        //dependency injection without container
        return Activator.CreateInstance(controllerType, new Logger()) as IController;
    }
}
```

Constructor Injection



```
public class EmployeeController : Controller
{
    //creating object like below is tight coupling
    //Logger myLogger = new Logger();
    private readonly ILogger _logger;
    //injecting dependency into constructor
    public EmployeeController(ILogger logger)
    {
        _logger = logger;
    }

    public ActionResult Index()
    {
        _logger.Log("Constructor Injection Successful");
        return View();
    }
```

Register Custom Factory



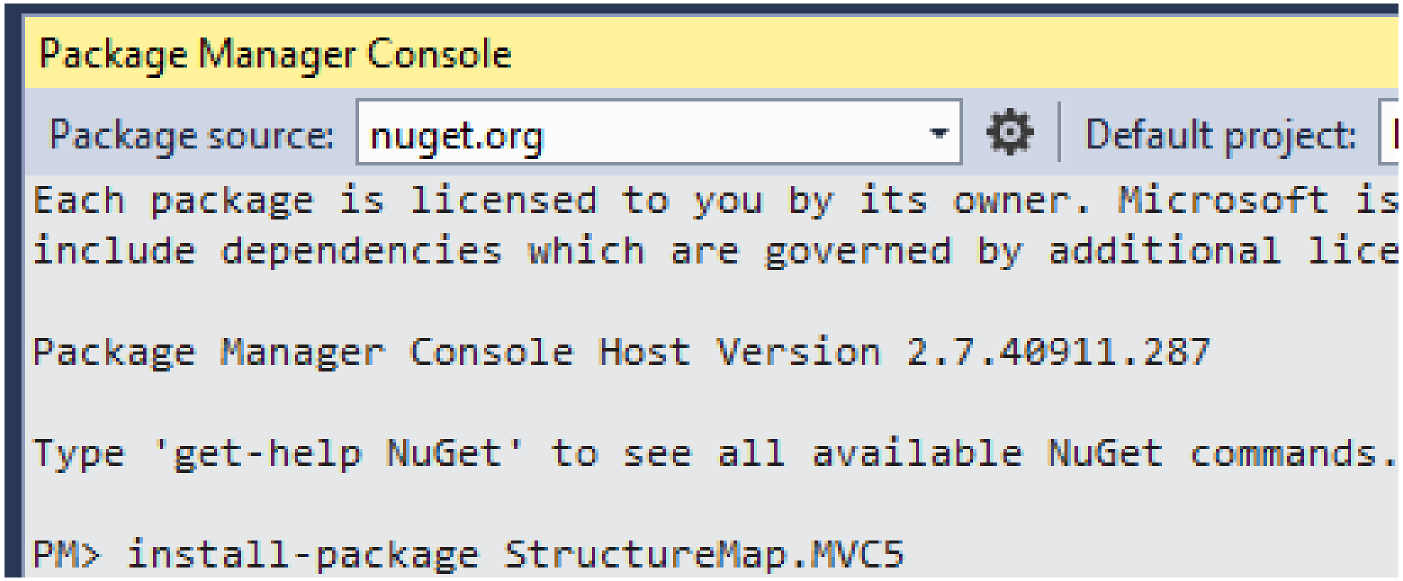
```
protected void Application_Start()
{
    //setting custom controller factory
    ControllerBuilder.Current.SetControllerFactory(new CustomControllerFactory());

    AreaRegistration.RegisterAllAreas();
    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);
    BundleConfig.RegisterBundles(BundleTable.Bundles);
}
```

Dependency Injection Containers

- StructureMap
- Ninject
- Unity
- Castle Windsor
- Spring.NET

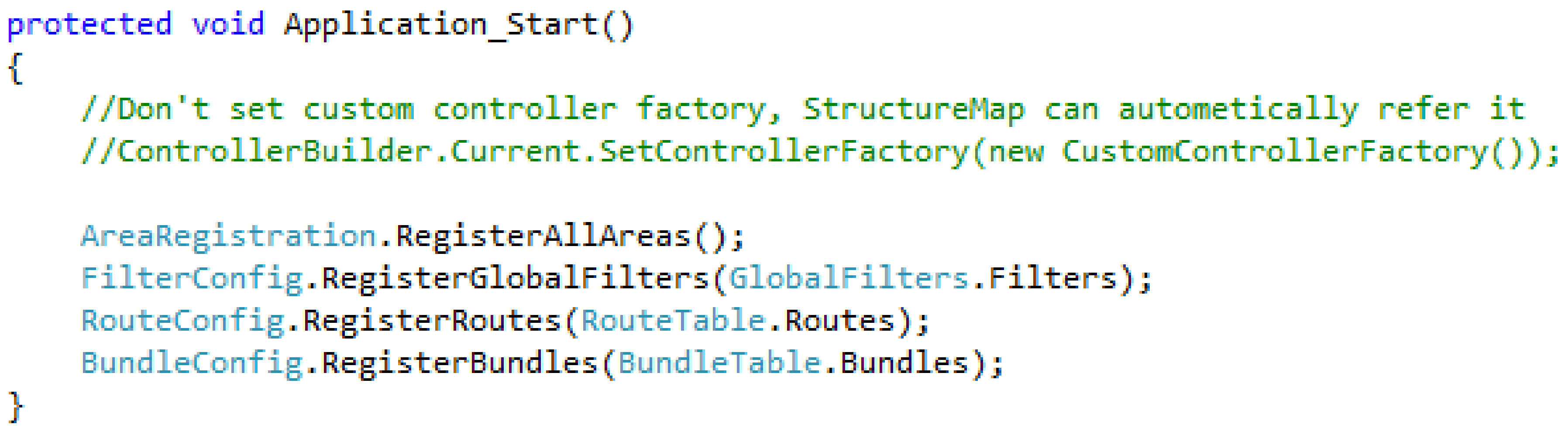
Using StructureMap Container

- Install StructureMap as Nuget package



```
Each package is licensed to you by its owner. Microsoft is
include dependencies which are governed by additional lice

Package Manager Console Host Version 2.7.40911.287

Type 'get-help NuGet' to see all available NuGet commands.

PM> install-package StructureMap.MVC5
```

StructureMap will do the rest



```
protected void Application_Start()
{
    //Don't set custom controller factory, StructureMap can autometically refer it
    //ControllerBuilder.Current.SetControllerFactory(new CustomControllerFactory());

    AreaRegistration.RegisterAllAreas();
    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);
    BundleConfig.RegisterBundles(BundleTable.Bundles);
}
```

Caching in MVC App

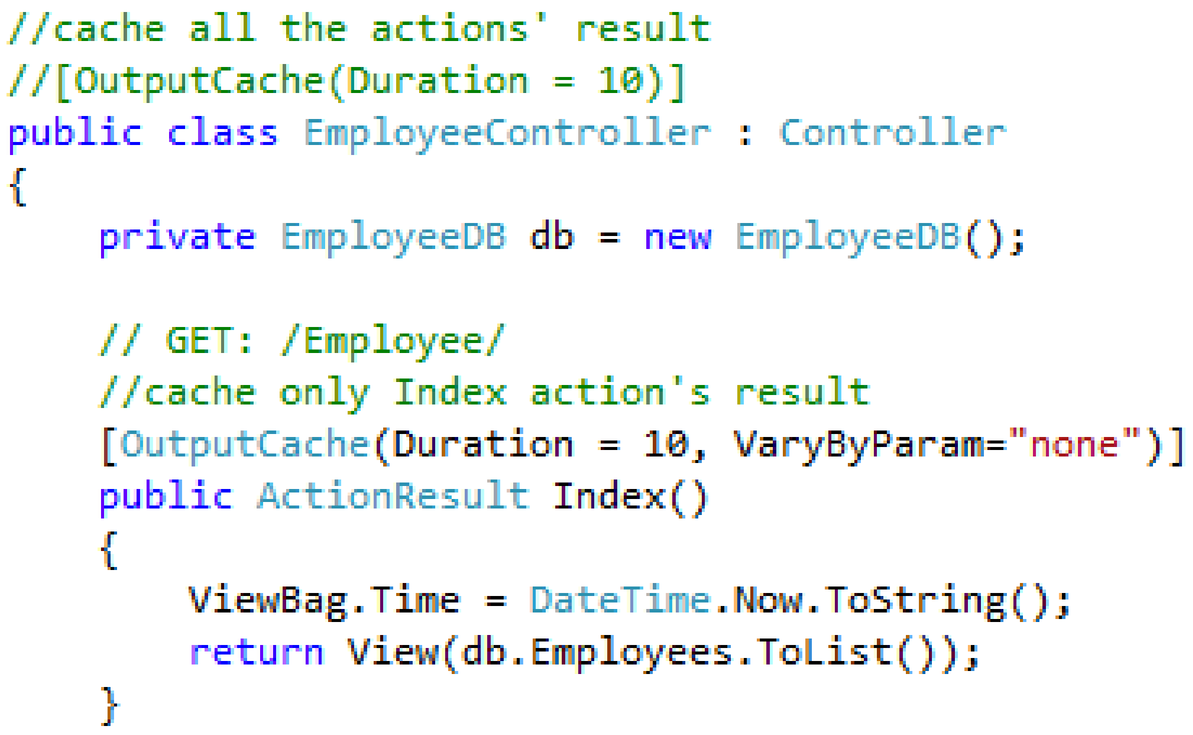
- Store frequently accessed data
- Reduce hosting server round-trips
- Reduce database server round-trips
- Reduce network traffic
- Maximize reusability
- Improve performance

Types of Caching

- Output Caching
  - Cache the content returned by a controller action
  - If same action invokes again, the server returns content from cache

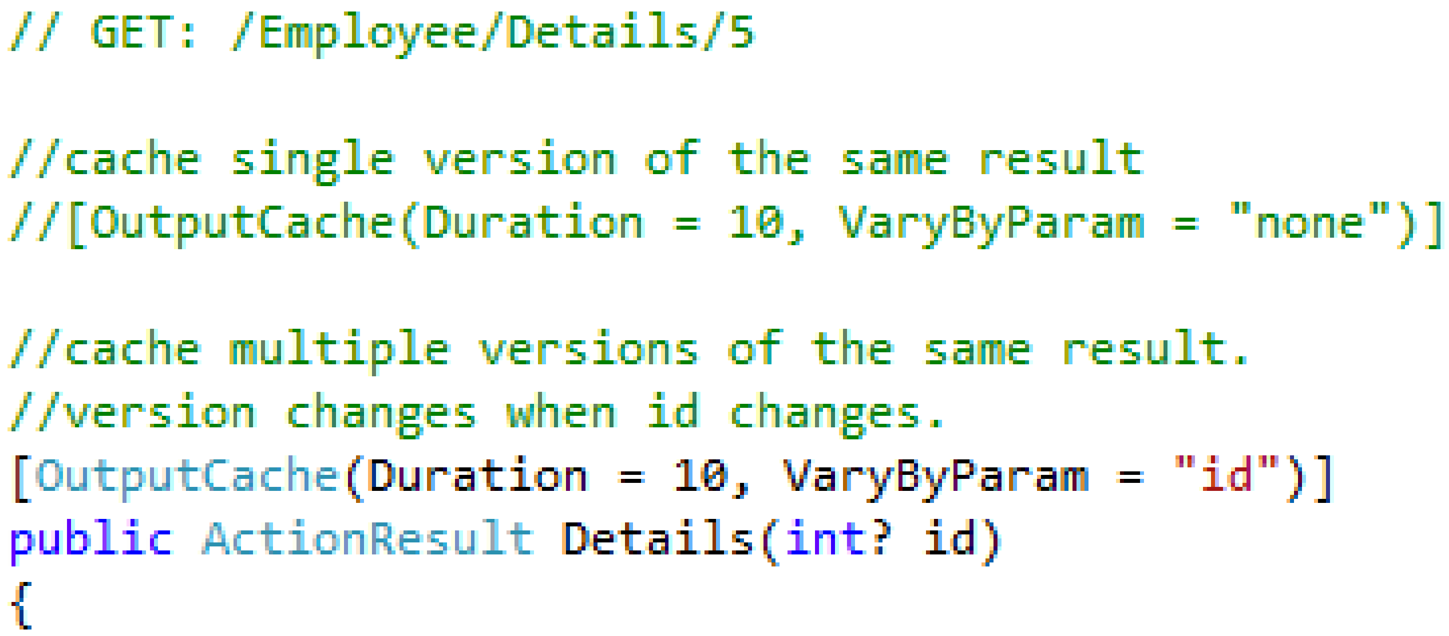
- Application Caching
  - Instead of storing entire page, it stores data objects
  - It stores the object in a key-value based cache

Output Cache



```
//cache all the actions' result
//[OutputCache(Duration = 10)]
public class EmployeeController : Controller
{
    private EmployeeDB db = new EmployeeDB();

    // GET: /Employee/
    //cache only Index action's result
    [OutputCache(Duration = 10, VaryByParam="none")]
    public ActionResult Index()
    {
        ViewBag.Time = DateTime.Now.ToString();
        return View(db.Employees.ToList());
    }
```

Vary By Parameter



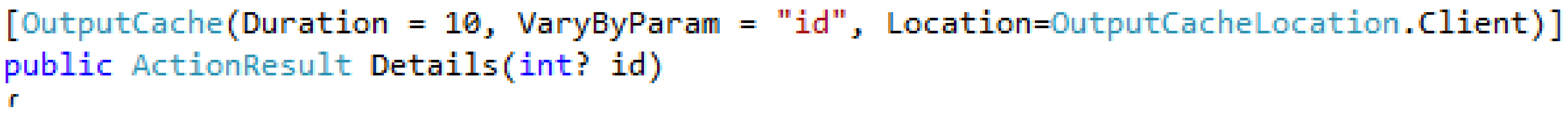
```
// GET: /Employee/Details/5

//cache single version of the same result
//[OutputCache(Duration = 10, VaryByParam = "none")]

//cache multiple versions of the same result.
//version changes when id changes.
[OutputCache(Duration = 10, VaryByParam = "id")]
public ActionResult Details(int? id)
{
```

Other Parameters

- VaryByParam = “*”
- VaryByCustom
- VaryByHeader
- SqlDependency

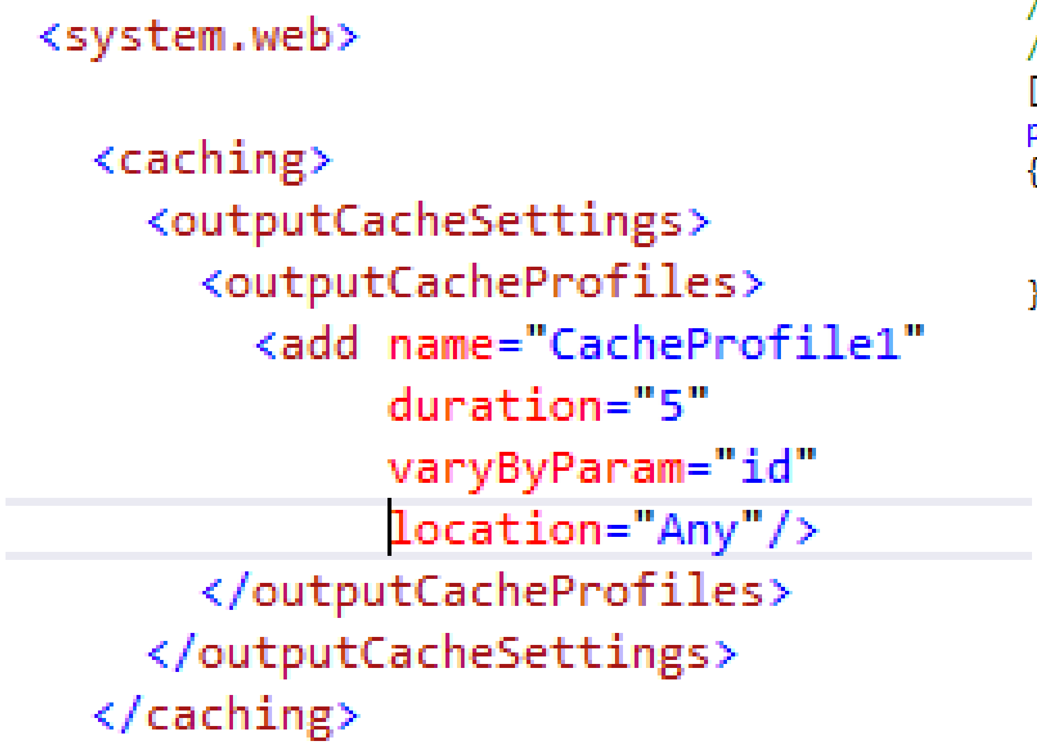
Cache Location

- Any (Default)
  - Cached in 3 locations, web server, proxy server and client browser
- Client
- Server
- ServerAndClient
- None

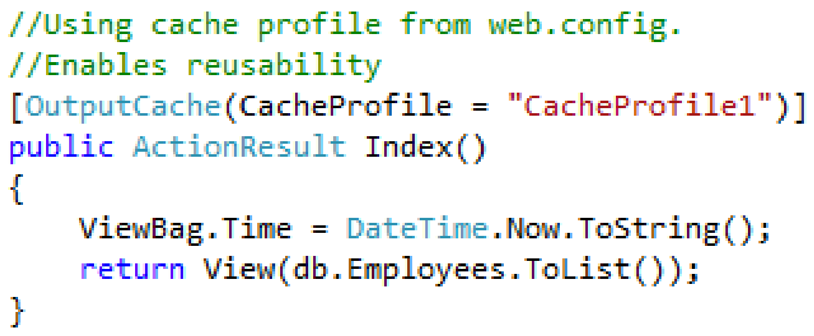


```
[OutputCache(Duration = 10, VaryByParam = "id", Location=OutputCacheLocation.Client)]
public ActionResult Details(int? id)
{
```

Cache Profile

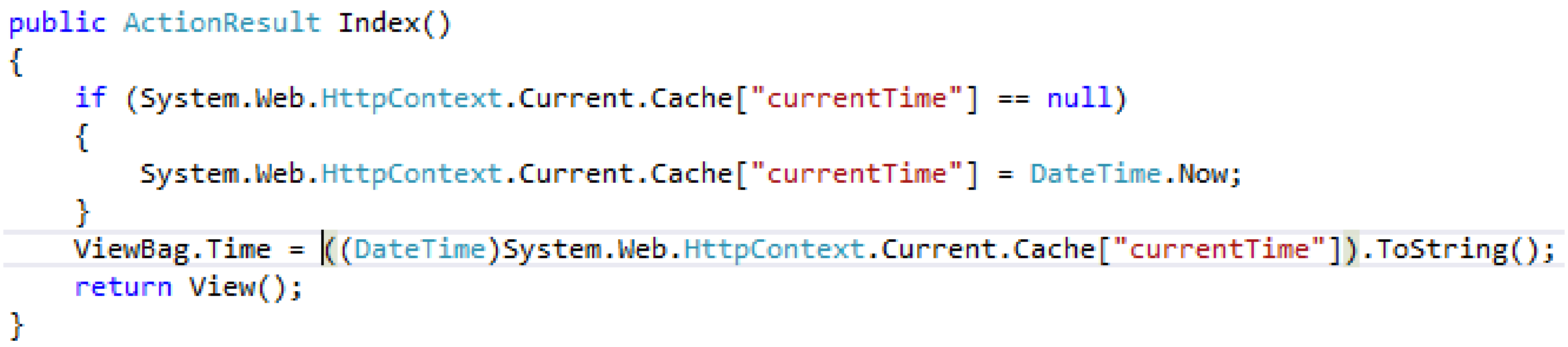


```
<system.web>

  <caching>
    <outputCacheSettings>
      <outputCacheProfiles>
        <add name="CacheProfile1"
             duration="5"
             varyByParam="id"
             location="Any"/>
      </outputCacheProfiles>
    </outputCacheSettings>
  </caching>
```

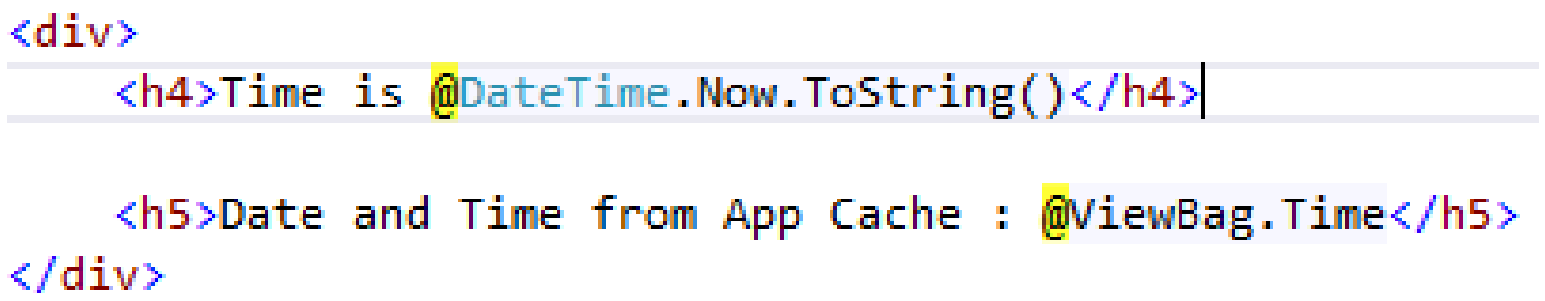


```
//Using cache profile from web.config.
//Enables reusability
[OutputCache(CacheProfile = "CacheProfile1")]
public ActionResult Index()
{
    ViewBag.Time = DateTime.Now.ToString();
    return View(db.Employees.ToList());
}
```

Application Caching

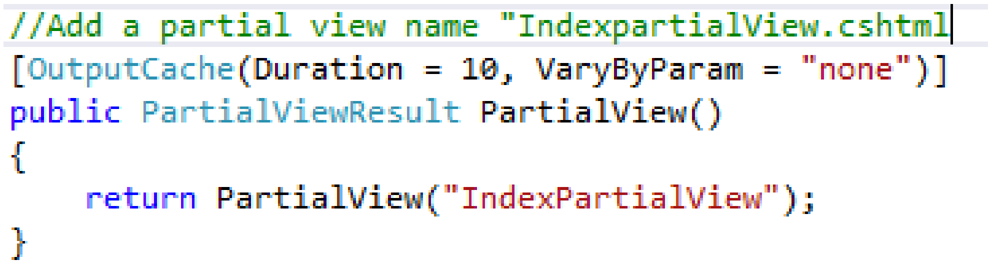


```
public ActionResult Index()
{
    if (System.Web.HttpContext.Current.Cache["currentTime"] == null)
    {
        System.Web.HttpContext.Current.Cache["currentTime"] = DateTime.Now;
    }
    ViewBag.Time = ((DateTime)System.Web.HttpContext.Current.Cache["currentTime"]).ToString();
    return View();
}
```

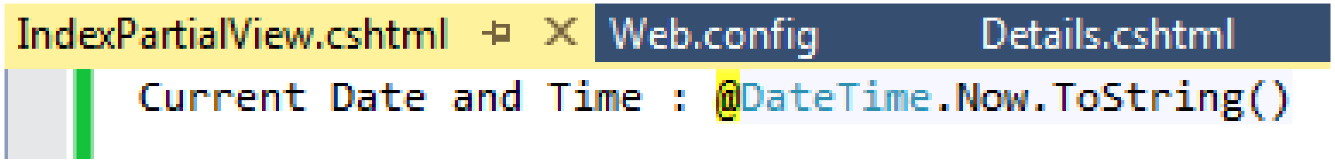


```
<div>
    <h4>Time is @DateTime.Now.ToString()</h4>

    <h5>Date and Time from App Cache : @ViewBag.Time</h5>
</div>
```

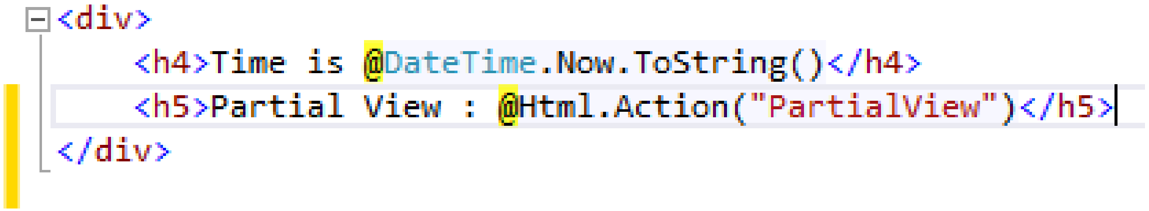
Partial Page Caching



```
//Add a partial view name "IndexpartialView.cshtml
[OutputCache(Duration = 10, VaryByParam = "none")]
public PartialViewResult PartialView()
{
    return PartialView("IndexPartialView");
}
```



```
Current Date and Time : @DateTime.Now.ToString()
```



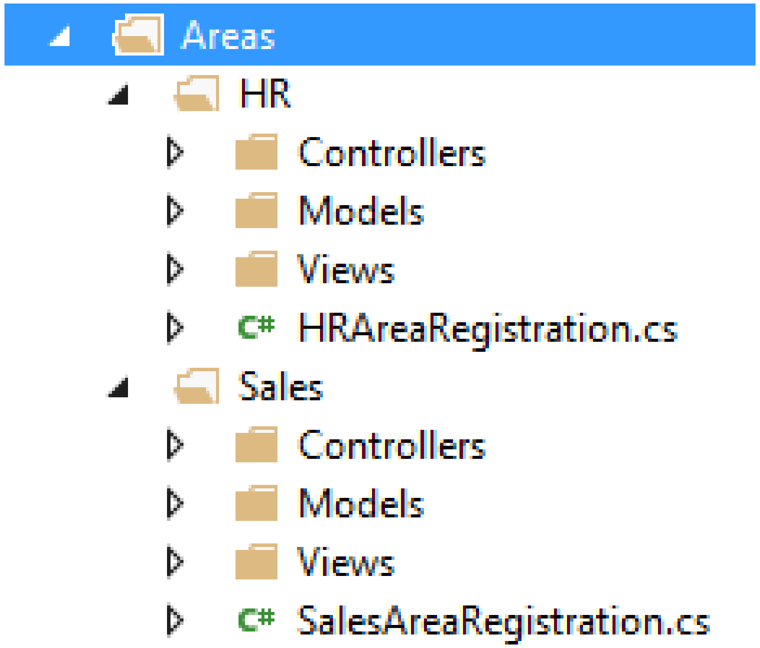
```
<div>
    <h4>Time is @DateTime.Now.ToString()</h4>
    <h5>Partial View : @Html.Action("PartialView")</h5>
</div>
```

Globalization

- Internationalization
  - Globalization – designing apps to support different culture
  - Localization – customizing apps for a specific culture
- Culture
  - Determines date, number and currency formatting
- UICulture
  - Determines which resources are to be loaded

Areas

- Dividing large application into logical areas



```
Areas
  HR
    Controllers
    Models
    Views
    HRAreaRegistration.cs
  Sales
    Controllers
    Models
    Views
    SalesAreaRegistration.cs
```

Asynchronous Controllers

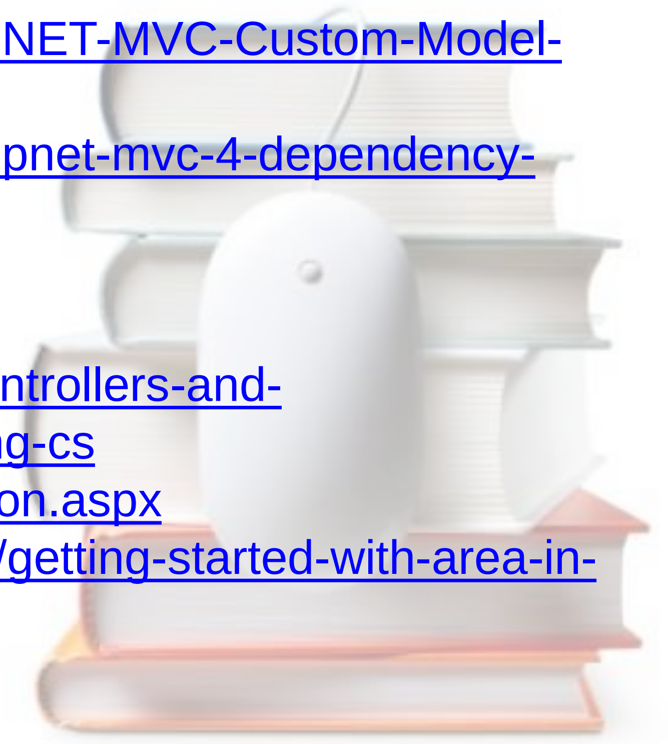
```
public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
{
    if (ModelState.IsValid)
    {
        var user = await UserManager.FindAsync(model.UserName, model.Password);
        if (user != null)
        {
            await SignInAsync(user, model.RememberMe);
            return RedirectToLocal(returnUrl);
        }
        else
        {
            ModelState.AddModelError("", "Invalid username or password.");
        }
    }

    // If we got this far, something failed, redisplay form
    return View(model);
}
```

Diagnostics

- Logging Tools
  - ASP.NET Health Monitoring
  - P&P Application Logging Block (Microsoft)
  - Log4net (Appache)
  - Elmah (Google)
  - Glimpse

Bibliography, Important Links

- http://www.asp.net/mvc/tutorials/older-versions/controllers-and-routing/understanding-action-filters-cs
- http://www.codeproject.com/Articles/605595/ASP-NET-MVC-Custom-Model-Binder
- http://www.asp.net/mvc/tutorials/hands-on-labs/aspnet-mvc-4-dependency-injection
- http://www.hanselman.com/blog/ListOf-NETDependencyInjectionContainersIOC.aspx
- http://www.asp.net/mvc/tutorials/older-versions/controllers-and-routing/improving-performance-with-output-caching-cs
- http://afana.me/post/aspnet-mvc-internationalization.aspx
- http://www.c-sharpcorner.com/UploadFile/4b0136/getting-started-with-area-in-mvc-5/



Cybage® Delivering Value. Scientifically.

Any Questions?

Delivering Value. Scientifically.

Thank you!