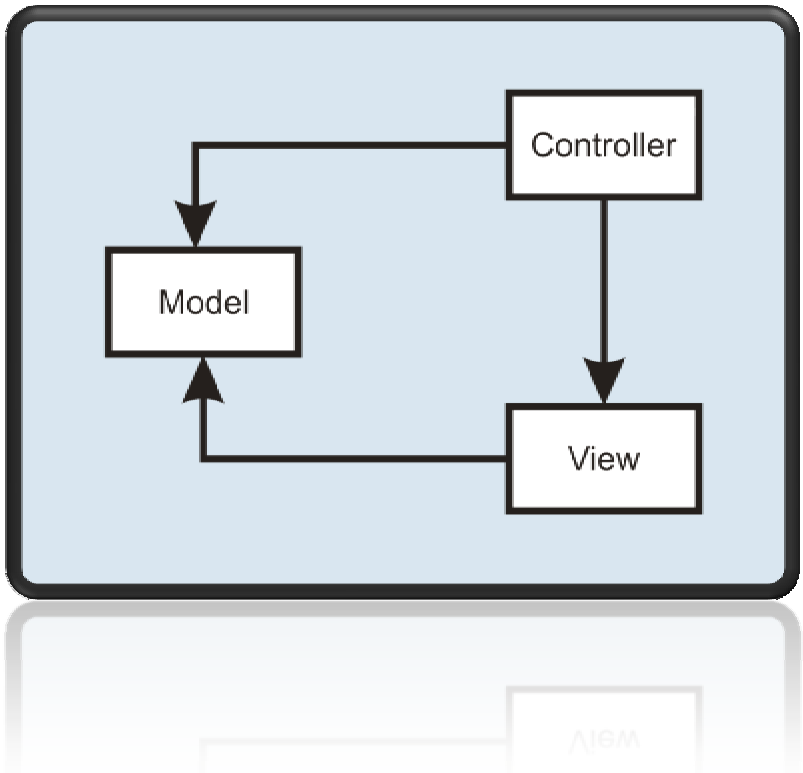
ASP.NET MVC

Michal Horák 2008

Model View Controller

Základem ASP.NET MVC je návrhový vzor (někdy se také říká architektonický vzor, protože jde spíše o architekturu aplikace) Model View Controller. Tento vzor poprvé popsal Trygve Reenskaug v roce 1979 a poprvé byl použit v jazyce Smalltalk. Model View Controller se, jak je z názvu vidět, skládá ze tří částí:

1. Model
    - Implementace business logiky
    - Persistence dat
2. View
    - Uživatelské rozhraní
3. Controller
    - Zpracování požadavků
    - Použití modelu
    - Navigace mezi View



Princip MVC

Uživatel provede nějakou akci na uživatelském rozhraní, Controller obdrží oznámení o této akci z uživatelského rozhraní (View) a následně Controller přistoupí k modelu a v případě potřeby ho zaktualizuje na základě provedené akce. Model je pouze jiný název pro doménovou vrstvu, doménová logika zpracuje změněná data a View získá data přímo z modelu a model o View nemá žádné informace. Uživateslké rozhraní pak čeká na další akci uživatele, které celý cyklus zahájí znovu.

Implementace MVC

V současné době se MVC používá především jako architektura webových aplikací, kde se hodí především pro složitější aplikace, kde zajišťuje flexibilitu a spolehlivost.

Příklady implementací

- JavaServer Faces
- Zend Framework
- Cake PHP
- Ruby on Rails
- ASP.NET MVC

ASP.NET MVC

ASP.NET MVC je implementace MVC pro ASP.NET, jde o alternativu k webforms, nikoliv však jeho náhradou. Spoustu věcí z klasického ASP.NET lze i nadále používat. Ke svému fungování využívá infrastruktury HTTP handlerů a HTTP modulů (Pro mapování url pravidel na akce Controlleru).

ASP.NET MVC vs. ASP.NET WebForms

V ASP.NET MVC máme tedy alternativu k ASP.NET Webforms, jaké jsou tedy mezi nimi rozdíly?

- Web forms
  - Využívá Page controller pattern
  - Postback
  - ViewState
  - Server controls
- MVC
  - Využívá Model View Controller pattern
  - Požadavky směřují na controllery
  - Controllery jsou separovány od view
  - ASPX stránky slouží jako šablony na data

Výhody a nevýhody ASP.NET MVC

Výhody

- Lepší oddělení logiky od prezentace
- Kontrola nad generovaným kódem prezentace
- Snazší testovatelnost
- Hezčí tzv. „SEO Friendly“ url

Nevýhody

- Složitější implementace
- Nemožnost používat komponenty využívající viewstate a postback

Oddělení logiky od prezentace

Z popisu MVC je oddělení logiky od prezentace jasně viditelné, ASP.NET MVC přímo nutí programátora, aby oddělil logiku od prezentace. Je to dáno tím, že Prezentace je Views, která získává controllerem aktualizovaná data z modelu. View tedy pouze zobrazuje již připravená data a je tedy pouhou šablonou, která nám říká, co se má kde zobrazit.

Projekt ASP.NET MVC

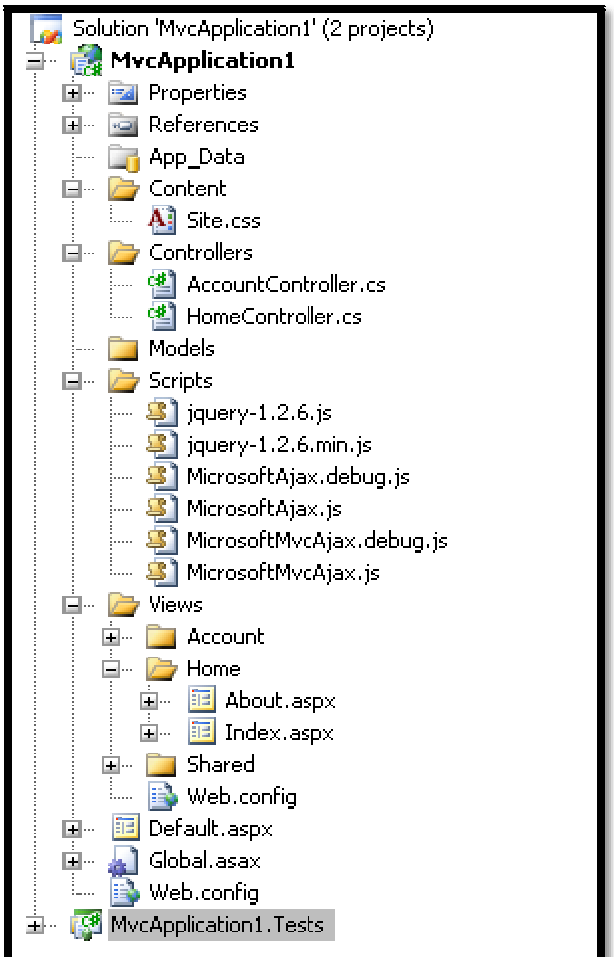
Vytvoření projektu ve Visual Studiu

Pro vytvoření projektu ve Visual Studiu potřebujeme mít buď nainstalované Visual Studio ASP.NET MVC Extension nebo mít minimálně Visual Studio 2008 SP1 a .NET Framework 3.5 SP1. I přesto, že jde o webový projekt, tak jej nevytváříme tak, že ve visual studiu vybereme File->New->Web Site, ale File->New->Project. Objeví se známé dialogové okno, kde si vybereme kategorii Web. V této kategorii máme položku ASP.NET MVC Web Application, to je přesně ta položka, kterou chceme.

Takže ji vybereme a potvrdíme, před vytvořením projektu budete dotázáni, jestli také vytvořit testovací projekt, to samozřejmě doporučuji. Takže máme vytvořený nový projekt, podívejme se na něj.

Máme v něm následující složky:

- Content – zde je uložen soubor s kaskádovými styly, můžeme sem také přidat všechny ostatní soubory, které mají něco společného s grafikou či jiným obsahem podobného typu. Různé fotky, obrázky atd.

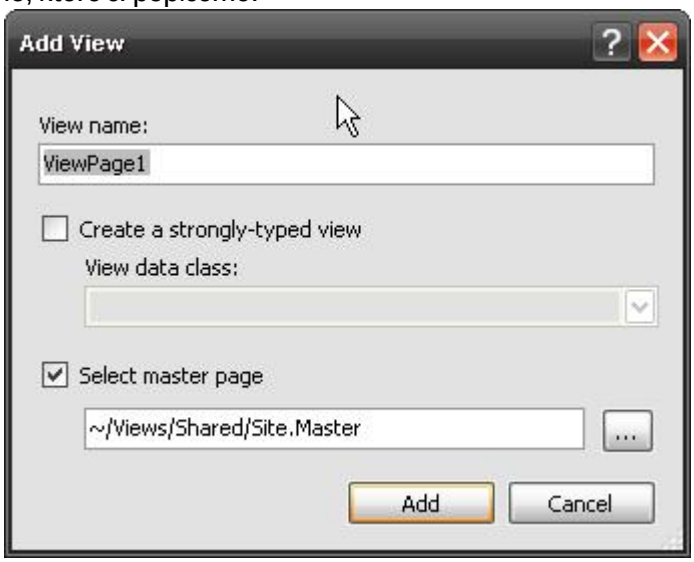


Obrázek 1: Nový projekt ASP.NET MVC

- Controllers – v této složce jsou uloženy Controllery
- Models – v novém projektu je tato složka prázdná, model si musíme nejdříve vytvořit, o tom později.
- Scripts – zde jsou uloženy soubory pro skriptování na straně klienta, převážne tedy JavaScripty, ve výchozím projektu zde máme javascriptové knihovny pro Ajax a také JQuery.
- Views – je složka určená pro Views, tedy stránky prezentační vrstvy.

Views

Ve složce Views má každý Controller svoji složku a v této složce jsou aspx stránky, které přidáváme jako nový View. Stačí na tuto složku kliknout pravým tlačítkem a vybrat Add -> View. Zobrazí se dialogové okno, které si popíšeme:



Jak je vidět, stejně jako u klasického ASP.NET MVC je možné používat Master Page. Dále je tu ještě jeden Cheb box, který z klasického ASP.NET neznáme a to „Create a strongly-typed view“. Jde o to, že můžeme mít View jako generickou třídu a tedy rovnou nastavit datový typ, který bude view akceptovat. Tento datový typ si můžeme rovnou vybrat z modelu, popřípadě lze vždycky upravit. Jestliže si tuto možnost neodškrtneme a budeme jej chtít použít, potom stačí když tento view zdědíme z generického typu ViewPage<T>. K datům se ve stránce dostaneme přes ViewData.Model.

Příklad View

```
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master"
AutoEventWireup="true" CodeBehind="Index.aspx.cs"
Inherits="MVC_pokus.Views.Home.Index" %>

<asp:Content ID="indexContent" ContentPlaceHolderID="MainContent"
runat="server">
    <h2><%= Html.Encode(ViewData["Message"]) %></h2>
    <p>
        To learn more about ASP.NET MVC visit <a
href="http://asp.net/mvc" title="ASP.NET MVC
Website">http://asp.net/mvc</a>.</p>
</asp:Content>
```

Controllers

Controller je třída zděděná ze System.Web.Mvc.Controller, která implementuje rozhraní IController. Obsahuje metody s návratovou hodnotou ActionResult a nejčastěji budou vracet:

- View() – vrátí View odpovídající názvu metody ve které se nachází
- RedirectToAction() – přesměruje na další akci
- ContentResult() – předává obsah bez vlastního View

Controller – příklad

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Mvc.Ajax;

namespace MVC_pokus.Controllers
{
    [HandleError]
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewData["Title"] = "Home Page";
            ViewData["Message"] = "Welcome to ASP.NET MVC!";

            return View();
        }

        public ActionResult About()
        {
            ViewData["Title"] = "About Page";

            return View();
        }
    }
}
```

Models

V modelu si můžeme napsat svoji Business logiku, nebo u některých projektů se vyplatí vygenerovat si třídy pomocí LINQ to SQL Clasess, nebo NHibernate.

Podpora skriptů na straně kilenta

Když se podíváme do složky Scripts, tak si všimneme, že jsou tam už nějaké scripty vloženy, konkrétně

- JQuery
- MS Ajax
- MS MVC AJAX

Tato podpora je spíše jen taková, že zde tyto skripty jsou, ovšem pomocí jQuery se dají dělat divy a určitě stojí za vyzkoušení. Stejně jako v ASP.NET MVC se dají použít i v klasickém ASP.NET nebo i jiných webových technologiích.

Unit testy

Při vytvoření projektu jste dotázání, zda vytvořit i projekt s UnitTesty. Testování probíhá tak, že máme pro každý controller test a ten spouští akce controlleru s daty, které by jinak přišli z View. Jak je vidět, toto testování je díky dobrému oddělení logiky od prezentace jednodušší než u klasického ASP.NET, kde se spousta akcí vykonává už na stránce.

Unit test - příklad

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Web.Mvc;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using MVC_pokus;
using MVC_pokus.Controllers;

namespace MVC_pokus.Tests.Controllers
{
    /// <summary>
    /// Summary description for HomeControllerTest
    /// </summary>
    [TestClass]
    public class HomeControllerTest
    {
        [TestMethod]
        public void Index()
        {
            // Setup
            HomeController controller = new HomeController();
            // Execute
            ViewResult result = controller.Index() as ViewResult;
            // Verify
            ViewDataDictionary viewData = result.ViewData;
            Assert.AreEqual("Home Page", viewData["Title"]);
            Assert.AreEqual("Welcome to ASP.NET MVC!",
viewData["Message"]);
        }
        [TestMethod]
        public void About()
        {
            // Setup
            HomeController controller = new HomeController();

            // Execute
            ViewResult result = controller.About() as ViewResult;

            // Verify
            ViewDataDictionary viewData = result.ViewData;
            Assert.AreEqual("About Page", viewData["Title"]);
        }
    }
}
```

Práce s url cestami (URL mapping)

V ASP.NET MVC mapujeme url cesty na konkrétní controller, url cesta je vlastně požadavek a podle požadavku pak daný controller tento požadavek obslouží. Toto mapování je vhodné pro tzv. SEO optimalizaci, protože díky ní snadno vytváříme „SEO friendly“ url. Výchozí nastavení v ASP.NET MVC je [controller]/[action]/[id], které si ovšem můžeme snadno změnit. Když se podíváme do souboru Global.asax.cs, tak uvidíme následující:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace MVC_pokus
{
    public class MvcApplication : System.Web.HttpApplication
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                "Default",                          // Route name
                "{controller}/{action}/{id}",  // URL with parameters
                new { controller = "Home", action = "Index", id = "" }
                // Parameter defaults
            );

        }

        protected void Application_Start()
        {
            RegisterRoutes(RouteTable.Routes);
        }
    }
   }
```

Jak je vidět, pomocí metody RegisterRoutes si můžeme toto mapování pro své potřeby upravit a není to nijak složité.

Závěr?

ASP.NET MVC je součástí Visual Studia 2008 SP1 s nainstalovaným .NET frameworkem 3.5 SP1. Více o ASP.NET MVC se dozvíte na stránkách http://www.asp.net/mvc/, kde je k nalezení i spousta tutoriálů a videí.

Příklad ASP.NET MVC aplikace – Jednoduchý blog

Jako příklad zde máme jednoduchý blog, který bude umět následující:

1. přihlášení a odhlášení uživatele,
2. zobrazení článků,
3. vkládání, mazání a editace článků

Nejdříve si vytvoříme nový projekt nazvaný například MVCweb. Přidáme do něj databázi, následující podoby:

Articles
• ID (int) Primary Key • Title (nvarchar(250)) • Content (ntext) • date(datetime) • IsVisible (bit)

Users
• ID (int) Primary Key • login (nvarchar(250)) • password (navrachar(250))

Do složky Models vložíme LINQ to SQL Clasess a vložíme do něj tyto dvě tabulky. Tím jsme si vytvořili jednoduchou logiku aplikace. Model máme tedy hotový. Nyní je vhodné si vytvořit základní controller, pro naše případy do něj vložíme pouze property IsLogin:

BaseController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Mvc.Ajax;

namespace MVCweb.Controllers
{
    public class BaseController : Controller
    {
        protected bool isLogin {
            get {
                bool login = false;
                if (Session["isLogin"] == null)
                {
                    return false;
                }
                else
                {
                    bool.TryParse(Session["isLogin"].ToString(), out
login);
                    return login;
                }
```

```
            }
            set {
                Session["isLogin"] = value;
            }
        }
    }
}
```

No a controllery si uděláme dva Home a Login, i když by se blog dal rozdělit klidně i na tři (Home, Admin, Login).

HomeController.cs

V tomto controlleru budeme reagovat na následující akce:

- Index – zobrazení posledních článků
- Article – zobrazení jednoho článku
- Create – zobrazení formuláře pro vytvoření článku
- CreateNew – uložení nového článku
- Edit – zobrazení formuláře pro editaci článku
- EditArticle – uložení editovaného článku
- DeleteArticle – odstranění článku
- VisibleArticle – zviditelnění článku pro Index
- Admin – zobrazení článků pro editaci

LoginController.cs

V Login controlleru máme pouze tři akce:

- Index – zobrazení formuláře pro přihlášení
- Login – přihlášení uživatele
- Logout – odhlášení uživatele

Zde si ukážeme i zdrojový kód:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Mvc.Ajax;
using MVCweb.Models;

namespace MVCweb.Controllers
{
    public class LoginController : BaseController
    {
        private myBlogDataContext db = new myBlogDataContext();

        public ActionResult Index()
        {
            return View();
```

```
        }

        public ActionResult Login(string login, string password)
        {
            var user = from u in db.Users where u.login == login
               && u.password == password select u;

            if (user.ToList().Count > 0)
            {
                isLogin = true;
                return RedirectToAction("Index", "Home");
            }
            else {
                return RedirectToAction("Index");
            }

        }

        public ActionResult Logout() {
            isLogin = false;

            return RedirectToAction("Index", "Home");
        }
    }
}
```

Jak vidíte tak při akci Login a Logout přesměrováváme na akci v jiném controlleru, toto jsme si zatím neukázali.

Controllery tedy máme hotové a již chybí pouze Views. Tak tedy, pro každý controller musíme mít udělanou složku ve složce Views. Máme tedy složky Home a Login. Také je dobré si vytvořit master page. Když se podíváme na akce controlleru Home, tak zjistíme, že pět z nich nám vrací pohled, pro tyto akce tedy musíme mít udělanou stránku:

- Admin
- Article
- Create
- Edit
- Index

Stejně tak pro Login, zde máme pouze jednu stránku s přihlašovacím formulářem Index. Kompletní zdrojové kódy naleznete v příloze, soubory, které zůstaly výchozí jsou vynechány. Stejně tak jako zdrojové kódy modelu, který je vygenerován pomocí LINQ to SQL Clasess, jsou vynechány.

Příloha: kompletní zdrojové kódy příkladu

BaseController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Mvc.Ajax;

namespace MVCweb.Controllers
{
    public class BaseController : Controller
    {
        protected bool isLogin {
            get {
                bool login = false;
                if (Session["isLogin"] == null)
                {
                    return false;
                }
                else
                {
                    bool.TryParse(Session["isLogin"].ToString(), out
login);
                    return login;
                }
            }
            set {
                Session["isLogin"] = value;
            }
        }
    }
}
```

HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Mvc.Ajax;
using MVCweb.Models;

namespace MVCweb.Controllers
{
    public class HomeController : BaseController
    {
        private myBlogDataContext db = new myBlogDataContext();

        // Display List of Last Articles
        public ActionResult Index()
        {
            var articles = from a in db.Articles where a.IsVisible orderby
a.date descending select a;
            return View(articles.ToList());
        }
```

```
        // Form for Create a new Article
        public ActionResult Create()
        {
            if (isLogin)
            {
                return View();
            }
            else {
                return RedirectToAction("Index", "Login");
            }
        }

        // Save Article to database
        public ActionResult CreateNew(string title, string content)
        {
            if (isLogin)
            {
                Article newArticle = new Article();
                newArticle.Title = title;
                newArticle.Content = content;
                newArticle.date = DateTime.Now;
                newArticle.IsVisible = false;

                db.Articles.InsertOnSubmit(newArticle);
                db.SubmitChanges();
                return RedirectToAction("Admin");
            }
            else
            {
                return RedirectToAction("Index", "Login");
            }
        }
        // Show Edit View
        public ActionResult Edit(int id)
        {
            if (isLogin)
            {
                var article = from a in db.Articles where a.ID == id select
a;

                return View(article.Single());
            }
            else
            {
                return RedirectToAction("Index", "Login");
            }
        }
        // Show Article View
        public ActionResult Article(int id)
        {

            var article = from a in db.Articles where a.ID == id select a;
            this.ViewData["title"] = article.Single().Title;
            return View(article.Single());
        }
        // Delete Article
        public ActionResult Delete(int id)
        {
            if (isLogin)
            {
```

```
                var article = from a in db.Articles where a.ID == id select
a;

                db.Articles.DeleteOnSubmit(article.Single());
                db.SubmitChanges();

                return RedirectToAction("Admin");
            }
            else
            {
                return RedirectToAction("Index", "Login");
            }
        }
        // Show articles to admin
        public ActionResult Admin()
        {
            if (isLogin)
            {
                var articles = from a in db.Articles orderby a.date
descending select a;
                return View(articles.ToList());
            }
            else
            {
                return RedirectToAction("Index", "Login");
            }

        }
        // Edit Article
        public ActionResult EditArticle(int articleId, string title, string
content)
        {
            if (isLogin)
            {
                var articles = from a in db.Articles where a.ID ==
articleId select a;
                foreach (Article a in articles)
                {
                    a.Content = content;
                    a.Title = title;
                    a.date = DateTime.Now;
                }

                db.SubmitChanges();

                return RedirectToAction("Admin");
            }
            else
            {
                return RedirectToAction("Index", "Login");
            }
        }
        // Set Article as Visible or Invisible
        public ActionResult VisibleArticle(int id)
        {
            if (isLogin)
            {
                var articles = from a in db.Articles where a.ID == id
select a;
                MVCweb.Models.Article article = articles.Single();
                    article.date = DateTime.Now;
```

```
                    if (article.IsVisible) {
                        article.IsVisible = false;
                    }
                    else
                    {
                        article.IsVisible = true;
                    }


                db.SubmitChanges();

                return RedirectToAction("Index");
            }
            else
            {
                return RedirectToAction("Index", "Login");
            }
        }

    }
}
```

LoginController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Mvc.Ajax;
using MVCweb.Models;

namespace MVCweb.Controllers
{
    public class LoginController : BaseController
    {
        private myBlogDataContext db = new myBlogDataContext();

        public ActionResult Index()
        {
            return View();

        }

        public ActionResult Login(string login, string password)
        {
            var user = from u in db.Users where u.login == login &&
u.password == password select u;
            if (user.ToList().Count > 0)
            {
                isLogin = true;
                return RedirectToAction("Index", "Home");
            }
            else {
                return RedirectToAction("Index");
            }

        }

        public ActionResult Logout() {
```

```
            isLogin = false;

            return RedirectToAction("Index", "Home");
        }
    }
}
```

Views/Home/Admin.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/myBlog.Master"
AutoEventWireup="true" CodeBehind="Admin.aspx.cs"
Inherits="MVCweb.Views.Home.Admin" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="contentPlace"
runat="server">
    <a href="/Home/Create">Create new Article</a>
    <ul>
    <% foreach(MVCweb.Models.Article article in ViewData.Model){%>
            <li><a href="/Home/Edit/<%= article.ID %>">Edit</a> <a
href="/Home/Delete/<%= article.ID %>">Delete</a> <%= article.date %> <%=
article.Title %>
            </li>

    <%  } %>
    </ul>
</asp:Content>
```

Views/Home/Admin.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MVCweb.Models;


namespace MVCweb.Views.Home
{
    public partial class Admin : ViewPage<List<MVCweb.Models.Article>>
    {
    }
}
```

Views/Home/Article.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/myBlog.Master"
AutoEventWireup="true" CodeBehind="Article.aspx.cs"
Inherits="MVCweb.Views.Home.Article" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="contentPlace"
runat="server">
<h2><%=  ViewData.Model.date.ToShortDateString() %> <%=
ViewData.Model.Title %></h2>
                <%= ViewData.Model.Content %>
</asp:Content>
```

Views/Home/Article.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MVCweb.Models;

namespace MVCweb.Views.Home
{
    public partial class Article : ViewPage<MVCweb.Models.Article>
    {
    }
}
```

Views/Home/Create.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/myBlog.Master"
AutoEventWireup="true" CodeBehind="Create.aspx.cs"
Inherits="MVCweb.Views.Home.Create" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="contentPlace"
runat="server">
    <h1>Create new Article</h1>
    <form action="/Home/CreateNew" method="post">
    <label for="title">Title</label>
    <input type="text" id="title" name="title" />
    <label for="content">Content</label>
    <textarea id="content" cols="20" rows="10" name="content"></textarea>
    <input type="submit" value="Add new Article" />
    </form>
</asp:Content>
```

Views/Home/Edit.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/myBlog.Master"
AutoEventWireup="true" CodeBehind="Edit.aspx.cs"
Inherits="MVCweb.Views.Home.Edit" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="contentPlace"
runat="server">
    <h2>Edit article</h2>
    <form action="/Home/EditArticle" method="post">
    <%= Html.Hidden("aticleID", ViewData.Model.ID) %>
    <label for="title">Title</label>
    <%= Html.TextBox("title", ViewData.Model.Title) %>
    <label for="content">Content</label>
    <%= Html.TextArea("content", ViewData.Model.Content, 10, 20, null)%>
    <input type="submit" value="Edit article" />
    </form>
    <%if (ViewData.Model.IsVisible) { }
      else
      { %>
    <a href="/Home/VisibleArticle/<%= ViewData.Model.ID %>">Visible
article</a>
    <%} %>
</asp:Content>
```

Views/Home/Edit.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MVCweb.Models;


namespace MVCweb.Views.Home
{
    public partial class Edit : ViewPage<MVCweb.Models.Article>
    {
    }
}
```

Views/Home/Index.aspx

```
<%@ Page Title="My Blog" Language="C#"
MasterPageFile="~/Views/myBlog.Master" AutoEventWireup="true"
CodeBehind="Index.aspx.cs" Inherits="MVCweb.Views.Home.Index" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="contentPlace"
runat="server">
    <ol>
    <% foreach(MVCweb.Models.Article article in ViewData.Model){%>
            <li><h2><a href="./Article/<%= article.ID.ToString() %>"><%=
article.date.ToShortDateString() %> <%= article.Title %></a></h2>
                <%= article.Content %>
            </li>

    <%  } %>
    </ol>
</asp:Content>
```

Views/Home/Index.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MVCweb.Models;


namespace MVCweb.Views.Home
{
    public partial class Index : ViewPage<List<MVCweb.Models.Article>>
    {

    }
}
```

Views/Login/index.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/myBlog.Master"
AutoEventWireup="true" CodeBehind="index.aspx.cs"
Inherits="MVCweb.Views.Login.index" %>
```

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="contentPlace"
runat="server">
<h2>Login page</h2>
<form action="/Login/Login" method="post">
    <label for="login">Login</label>
    <input type="text" id="login" name="login" />
    <label for="password">Password</label>
    <input type="password" id="password" name="password" />
    <input type="submit" value="Login" />
</form>
</asp:Content>
```

Views/myBlog.Master

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeBehind="myBlog.Master.cs" Inherits="MVCweb.Views.myBlog" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">

    <title>myBlog</title>
    <link href="~/Content/Site.css" rel="stylesheet" runat="server"
type="text/css" />
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <div id="page">
        <h1><a href="/Home/">My Blog</a></h1>
        <asp:ContentPlaceHolder ID="contentPlace" runat="server">

        </asp:ContentPlaceHolder>

    </div>
    <div id="footer">
        <a href="/Home/Admin">Administrate Blog</a>
    </div>
</body>
</html>
```