Я написал небольшую сырую версию MVC framework для использования с классическим asp. На данный момент присутствуют следующие основные компоненты:

1. Маршрутизация
2. Контроллеры, представления, частичные представления
3. Привязка модели ModelBinders, фильтры методов дейсвий ActionFilters, ModelState
4. Модули (forms authentications, aspnet forms authentications для испоьзования с ASP.net)
5. HtmlHelpers

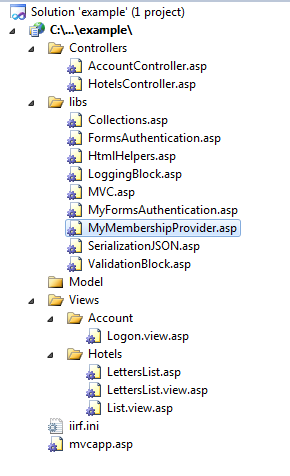
Плюс несколько вспомогательных блоков

1. Блок работы c коллекциями Collections
2. Блок работы с логированием LoggingBlock
3. Сериализация Json

Рассмотрим на простом примере использования, а затем рассмотрим подробнее реализацию фреймворка.

Шаблоны проектов для Visual Studio можно скачать **здесь**

Структура проекта выглядит следующим образом



Небольшое приложение, позволяющее администраторам некоторого туристического ресурса управлять списком гостиниц. Для входа на сайт Администратор использует свой логин и пароль. После успешной авторизации он может просматривать список отелей и редактировать его (добавлять новый отель, редактировать или удалить существующий). Скачать пример вы можете **здесь**

Проект содержит следующие файлы

1. AccountController.asp, HotelsController.asp - Соответственно в каталоге Controllers у нас два контроллера: Account и Hotels
2. Logon.view.asp, List.view.asp - Представления для каждого котроллера хранятся в каталоге Views, в подкаталоге с именем контроллера.

LettersList.view.asp - Контроллер Hotels в дополнении к представлению List, содержит частичное представления LettersList, представляющего из себя paginator разбивку на страницы по начальным буквам названий отелей

Каталог libs содержит основные библиотеки

1. Collections.asp – содержит классы для работы с коллекциями: ассоциативного массива (AssocArray) и словарь (Dictionary)
2. LoggingBlock.asp – блок логирования
3. SerializationJSON.asp – JSON сериазизатор
4. MVC.asp – сам mvc framework
5. HtmlHelpers.asp – содержит блоги для формирования html разметки
6. FormsAuthentication.asp – модули актентификации форм: базовый, и модуль для совместной работы с Form Authentication ASP.NET
7. MyFormsAuthentication.asp, MyMembershipProvider.asp - для примера, дополнительно некая специфичная для этого проекта нестандартнумая реализация модуля аутентификации MyFormsAuthentication, и провайдера для доступа к информации о пользователях из БД MyMembershipProvider.
8. iirf.ini – для маршрутизации использутся Ionics Isapi Rewrite Filter (IIS ISAPI модуль) http://iirf.codeplex.com/. Iirf.ini содержит базовые правила перезаписи url-адресов.
9. mvcapp.asp – файл самого MVC приложения

Реализация Account контроллера выглядит следующим образом

class cAccountController

    private membershipProvider

    private formsAuthenticationService

    Public Sub Class\_Initialize()

set baseController = (new cController).Init("Account")

        set membershipProvider = (new cMyMembershipProvider).Init(Application("TravelConnectionString"))

        set formsAuthenticationService = new cFormsAuthenticationService

End Sub

    public function LogOn()

        dim model: set model = new cLoginViewModel

        set LogOn = View("Logon", model)

    end function

    public function LogOn\_POST(model)

        if membershipProvider.ValidateUser(model.Username, model.Password) then

            formsAuthenticationService.SignIn model.Username, true

            set LogOn\_POST = Redirect("/hotels/")

        else

            baseController.ModelState.AddError("Access denied.")

            set LogOn\_POST = View("Logon", model)

        end if

    end function

    public function ModelBindForAction(actionName)

        if LCase(actionName) = "logon\_post" then

            set ModelBindForAction = new cLoginViewModel

            ModelBindForAction.Username = Request.Form("user")

            ModelBindForAction.Password = Request.Form("password")

        else

            ModelBindForAction = baseController.ModelBindForAction(actionName)

        end if

    end function

    public function GetFiltersForAction(actionName)

        GetFiltersForAction = baseController.GetFiltersForAction(actionName)

    end function

    public baseController

    private Property Get ModelState

        set ModelState = baseController.ModelState

    end Property

    private Property Get ControllerContext

        set ControllerContext = baseController.ControllerContext

    end Property

    private function View(viewName, model)

        set View = baseController.View(viewname, model)

    end function

    private function Redirect(url)

        set Redirect = baseController.Redirect(url)

    end function

    private function JSON(model)

        set JSON = baseController.JSON(model)

    end function

    private function EmptyContent()

        set EmptyContent = baseController.EmptyContent(model)

    end function

end class

У контроллера два action метода, думаю подробно их рассматривать не стоит, действия довольно очевидны. LogOn срабатывает по GET запросу /Account/LogOn и показывает страницу с формой авторизации. Представление Logon.view.asp выглядит следующим образом

<%

    dim model: set model = Session("model")

    dim ModelState: set ModelState = Session("ModelState")

%>

<html>

<head>

    <title>Log on</title>

</head>

<body>

    <% if not ModelState.IsValid then %>

        <ul>

       <%For Each error in ModelState.Errors   %>

        <li><%=error %></li>

       <%Next%>

        </ul>

    <%  end if %>

    <p>Please enter your username and password into the boxes below and click continue.</p>

    <form method="post" action="/account/logon.asp">

<label>Username</label><input type="text" name="user" size="20" value="<%=model.Username %>"/>

<label>Password</label><input type="password" name="password" size="20" value="<%=model.Password %>"/>

<input type="submit" value="Log In"/>

</form>

</body>

</html>

Данные с формы поступают в LogOn\_POST , который авторизует пользователя, используя некий свой сервис membership провайдер, и в случае успеха с помощью formsAuthenticationService как-то запоминает пользователя (через куки) и перенаправляет его на страницу со списком отелей. Здесь стоит обратить внимание, что метод LogOn\_POST в качестве параметра принимает модель cLoginViewModel, содержащую введенные пользователем данные с формы. Сам класс cLoginViewModel – view model для нашего контроллера, описан в том же файле что и контроллер

class cLoginViewModel

    public Username

    public Password

end class

Привязки модели cLoginViewModel к параметру метода LogOn\_POST происходит с помощью метода контроллера ModelBindForAction(actionName), который будет вызван автоматически во время привязки данных.

Сам formsAuthenticationService вызывает глобальный FormsAuthentication Module через «статическую» функцию FormsAuthentication\_SetAuthCookie для установки cookie.

class cFormsAuthenticationService

    public sub SignIn(username, isCreatePersistentCookie)

        FormsAuthentication\_SetAuthCookie username, isCreatePersistentCookie

    end sub

end class

HotelController интересен тем, что возвращает JSON данные и использует Action filter для авторизации, запрещающий вызывать все action-методы контроллера для неавторизованного пользователя.

'required #include file="libs/MVC.asp"

class cHotelsController

    private repository

    Public Sub Class\_Initialize()

set baseController = (new cController).Init("Hotels")

        set repository = new cHotelsRepository

End Sub

    public function ListByLetter(letter)

        dim model: set model = new cListByLetterViewModel

        model.FirstLetters = repository.GetAllFirstLettersOfHotels

        model.Hotels = repository.GetByCityFirstLetter(letter)

        set ListByLetter = View("List", model)

    end function

    public function Edit\_POST(hotel)

        repository.Update(hotel)

        set Edit\_POST = EmptyContent()

    end function

    public function Add\_POST(hotel)

        repository.Add(hotel)

        set Add\_POST = JSON(hotel)

    end function

    public function ModelBindForAction(actionName)

        dim hotel

        if LCase(actionName) = "listbyletter" then

            dim letter :letter = ClearString(Request.QueryString("letter"))

            if (letter="") then letter = "A"

            ModelBindForAction = letter

        elseif LCase(actionName) = "add\_post" then

            set hotel = new cHotel

            hotel.cityName = ClearString(Request.Form("cityName"))

            hotel.cityNameRus = ClearString(Request.Form("cityNameCyr"))

            hotel.hotelName = ClearString(Request.Form("hotelName"))

            hotel.hotelNameRus = ClearString(Request.Form("hotelNameCyr"))

            hotel.hotelType = ClearString(Request.Form("hotelType"))

            hotel.Address = ClearString(Request.Form("address"))

            hotel.Phone = ClearString(Request.Form("phone"))

            set ModelBindForAction = hotel

        elseif LCase(actionName) = "edit\_post" then

            set hotel = new cHotel

            hotel.id = CInt(ClearString(Request.Form("recordId")))

            hotel.cityName = ClearString(Request.Form("cityName"))

            hotel.cityNameRus = ClearString(Request.Form("cityNameCyr"))

            hotel.hotelName = ClearString(Request.Form("hotelName"))

            hotel.hotelNameRus = ClearString(Request.Form("hotelNameCyr"))

            hotel.hotelType = ClearString(Request.Form("hotelType"))

            hotel.Address = ClearString(Request.Form("address"))

            hotel.Phone = ClearString(Request.Form("phone"))

            set ModelBindForAction = hotel

        else

            ModelBindForAction = baseController.ModelBindForAction(actionName)

        end if

    end function

'.....base methods.........

end class