# MVC の概要

Model, View, Control に分けた Web ページの開発手法。

Mode データベースのテーブルなどに該当。ロジックなどを処理。

View ユーザーが目にする出力の部分

Control クライアントからのリクエストに対応する。

html の中に、C# を直接書くような形となる。

※以前使われていたWeb Forms はもうサポートされていない。

# プロジェクト新規作成

プロジェクトの作成は C# → Web → ASP.NET WebApplication を選ぶ

（ない場合はインストーラーでASP.NET にチェックを入れ取得）

↓

次のダイアログ

テンプレート：

MVC

Empty : Web API を作成する時に、空のテンプレートから必要な機能を追加していく。

Web API : Http電文のWebサービス用。但しEmptyから作れるので選ぶ機会はなさそう

Single Page Application: JavaScript + HTML5向き

チェックボックス：

MVC ：対象プロジェクトでWebPage を作るならチェック。

Web API：文字通りWebAPI 関連のライブラリが読み込まれる。

認証の変更

## 名前空間について

アプリが実行され、Web ページを表示する際に Razor 用に動的にcsファイルが作成されるらしい。その再、Views/web.config のnamespace にあるライブラリが読み込まれる。

見つからない場合はApp\_Web\_index.cshtml のエラーとなる。

※特に製作アプリ自体の名前空間はShared の方にあるので注意！

# 空のテンプレートからの作成方法

## MVC

NuGet から Microsoft.AspNet.WebApi を追加。

※コア参照でMVC を選択すると、最初から含まれているが？18.08.07

情報もと

<https://blog.okazuki.jp/entry/2013/08/23/223558>

## WebAPI

# 各フォルダの概要

・Global.asax

C#でいうとこのProgram.cs のようなもので、ここからアプリが始まる。

・App\_Start

認証、ルーティングなどの情報

・App\_Data

アプリケーションで利用するデータ（.mdf / .xml）など

・Model フォルダ

実際に扱うクラスを格納する。必ずしもこのフォルダに入れる必要はない。

・Controllers フォルダ

クライアントからのリクエスト処理をするクラス （ApiController を継承）を格納

クラス名を間違える(例：×...Controllers　○...Controller) と動かないので、Controllersフォルダ → 右クリック → 追加 → コントローラーとするのが無難

Empty

with read/write actions

with views

・Content

スタイルシート(css) や画像ファイル

・Scripts

.JavaScriptライブラリ

# Global.asax（エントリポイント）

書けるのはルートだけではない。

AreaRegistration.RegisterAllAreas();

GlobalConfiguration.Configure(WebApiConfig.Register);

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

RouteConfig.RegisterRoutes(RouteTable.Routes);

BundleConfig.RegisterBundles(BundleTable.Bundles);

・AreaRegistration

エリア（区分）を登録。 エリアは、コントローラーをグループ化する機能。

※using System.Web.Mvc; が必要

•GlobalConfiguration.Configure(WebApiConfig.Register);

WebAPIに対する設定を行う

•FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

全コントローラーに適用するFilterを登録。

（WebAPIの設定は関係ない。WebApiConfig.Registerにて設定するため）

・BundleConfig.RegisterBundles

JavaScript、cssのミニファイ、結合を行うための設定

※using System.Web.Optimization; が必要

# App\_Start

## RouteConfig.cs

文字通り、ルーティング設定を行う。

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

);

}

}

### MapRouteメソッド

ルーティングの追加

name: 一意であれば特に決まりはない。

url: 他と重複しない、一意な url **パターン**

default: {} の値が指定されなかった場合のデフォルト値

controller コントローラー名

action メソッド名

id 引数 id が省略可能かどうか？

※処理の順序は上からなので、別のルートを追加する場合は**デフォルトより上に書く**必要がある。

実行例

http://localhost:[port]/ → Default ：HomeController のIndex メソッド

http://localhost:[port]/User/ → User のIndex　（無指定の部分はデフォルト？）

http://localhost:[port]/User/Edit/1 → Edit(id:1) となる。※ルートパラメータ

http://localhost:[port]/User/Edit?code=1 → Edit(code:1) となる。※ポストデータ

・モデルバインド（urlから引数をバインドする事）で指定された**名前の**引数がなかった場合はメソッドには引数としてnullが渡される。

・

constraints: 制約条件が書ける p374参考。

### IgnoreRouteメソッド

：無視する url パターン。※よく分からない 18.08.07

## WebApiConfig.cs

WebAPI のルートなどが指定されている。

## BundleConfig.cs

CSSやJavaScriptなど複数ファイルを、一つのファイルにまとめ（Bundle）、空白や不要な文字を削除してファイルサイズを小さくする（Minification）。

バンドリングの一例（テンプレートを参考）

|  |  |  |
| --- | --- | --- |
| 種類 | 紐付path | 対象ファイル |
| ScriptBundle | ~/bundles/jquery | ~/Scripts/jquery-{version}.js |
| ScriptBundle | ~/bundles/jqueryval | ~/Scripts/jquery.validate\* |
| ~/Scripts/blackword.validate.js |
| ScriptBundle | ~/bundles/jqueryui | ~/Scripts/jquery-ui-{version}.js |
| ScriptBundle | ~/bundles/bootstrap | ~/Scripts/bootstrap.js |
| StyleBundle | ~/Content/jqueryui | ~/Content/themes/base/jquery-ui.min.css |
| StyleBundle | ~/Content/css | ~/Content/bootstrap.css |
| ~/Content/site.css |

後は @Styles.Render([紐付path]) で最終的なhtmlに以下の様に出力され、

<link href=”/Content/bootstrap.css” rel=”stylesheet”>

@Scripts.Render([紐付path]) で最終的なhtmlに以下の様に出力される。

<script src=”/Scripts/jquery-3.3.1.js></script>

### 追加方法

Empty などから始めた場合はこのファイルは無いので、追加する。

※但しBundleConfig.cs の内容はglobal.asax に記述しても問題無い。

1. App\_Startで追加→新しい項目:Class [BundleConfig.cs]
   * 名前空間に注意！ App\_Start がついてしまうが不要。
2. using は以下の２つだけにする。

using System.Web;

using System.Web.Optimization;

※System.Web.Optimization の参照、またはNuGet によるDL が必要。

PMなら：Install-package **Microsoft.AspNet.Web.Optimization**

1. 以下のように記述

public static void RegisterBundles(BundleCollection bundles)

{

bundles.Add(new ScriptBundle("~/bundles/jquery").Include(

"~/Scripts/jquery-{version}.js"));

bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(

"~/Scripts/jquery.validate\*",

"~/Scripts/blackword.validate.js"));

}

1. Global.asax.cs に追加

MVC のテンプレートではRouteConfigの下、一番下に記載されていた

BundleConfig.RegisterBundles(BundleTable.Bundles);

1. Shared/\_Layout.cshtml にコード追記

Optimization.dll を追加した場合はリビルドが必要

1. 場合によってはWebGreaseの更新、Web.Optimizationの再インストールが必要

参考url（英語）

<https://www.techjunkieblog.com/2015/05/aspnet-mvc-empty-project-adding.html>

# 試験起動

## ポート番号の確認

ローカル開発では IIS Express がOS の起動中は常に起動しているので、これを用いればよい。(もともとはサービスを”受ける”為のアプリらしい；サーバーなのに？）

開発以外で、ローカルでサーバーを起動するには：

1.開発環境VS でアプリを起動。

2.起動したブラウザでurl を確認 or タスクバーの右下の雲のようなマークを右クリック

3.port 番号と物理パスをメモ。

4.cmd で

C:\Program Files\IIS Express/iisexpress.exe /path:"[物理パス]" /port:[ポート番号]

でIISが起動する。

## index.html

プロジェクト内に index.html のようなものがないと単体での起動はできない。

（HTTP Error 403.14）

## IIS の設定

ごくまれにIIS が無効になっている場合がある。

コントロールパネルの「Windows の機能の有効化または無効化」で設定。

# View

## \_ViewStart.cshtml

全てのView に対して適応される設定。

具体的には以下のように記載されている。

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

・この\_ViewStart.cshtml ファイルを各Viewフォルダに配置しておくと、各View（Controller）ごとにデザインを変更できる。

・またはShared/\_AdminLayout.cshtml を作成しておき

A: Controller のActionResult で指定する

return View("Index", "\_AdminLayout", model);

B: 各View の最初で指定

@{

Layout = "~/Views/Shared/\_AdminLayout.cshtml";

}

C: ifで使い分ける

@{

var controller = HttpContext.Current.Request.RequestContext.RouteData.Values["Controller"].ToString();

string layout = "";

if (controller == "Admin")

layout = "~/Views/Shared/\_AdminLayout.cshtml";

else

layout = "~/Views/Shared/\_Layout.cshtml";

Layout = layout;

}

※あまりいい方法とは思えない。

## \_Layout.cshtml

共通で使いまわすページデザインがhtml 形式で記述されている。

@RenderBody の部分に各View が配置される。

### @Styles.Render　と　@Scripts.Render

スクリプトとスタイルのバンドリング。

→バンドリング css/js の複数ファイル１ファイルにまとめる事。

（参考）[BundleConfig.cs](#_BundleConfig.cs)

## RenderSection メソッド

名前付きセクションを作成する。サイドバーなどが作成できる。

name: セクション名

required: セクションが必須であることを指定する場合は true

※Script の読み込みにも用いられている。

e.g.) @RenderSection("scripts", required: false)

## Razor

cshtml の中にC#を直接書き込む。

・コメント @\* \*@

・コードブロック @{ }

・@model Controller から渡されたデータにアクセスする。

e.g. @model TestClass

・@model dynamic インテリセンスが有効になる（らしい）18.08.07

### ＠ViewBag

Controller と View 間でデータの受け渡しをする変数。

・ViewBag.変数名 = 値 と使用。 e.g. ViewBag.Message = “hoge”;

・変数名は命名規則に則る限り自由。

・その都度に変数名を付けるので、インテリセンスが使えない？18.08.08 ←かなり不便

最初から定められているプロパティもある。

Title

### @ViewData

上のViewBag と同様。但しこちらはDictionary 的に働く

e.g. ViewData[“Message”] = “hoge”;

### @Html

メソッド 説明

EditorFor TextBoxを用いて編集。ラムダ式でプロパティを指定。

AntiForgeryToken セキュリティ対策？詳細不明 18.08.08

・POST で用いられる。

・usingForm内に書かないとpartial view ではエラー。

ValidationSummary エラーメッセージをSummary 表示する

ValidationMessageFor 個別でエラー・メッセージを表示する

@Html は入力フォーム（input に相当？)も実装可能。

（参考）[入力フォーム](#_入力フォーム)

## 入力フォーム

C# 風にコントロールの実装方法をまとめる。

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

}

BeginForm(“Method名”, “Controller名”) でsubmit ボタンを押した時の挙動を指定。

引数がない場合は呼び出し元のController名とAction(Method)名を維持する。

### DropDown

Enum

@Html.EnumDropDownListFor(model => model.EnumMember)

他の色々なやり方

@Html.DropDownListFor(model=>model.Project,(IEnumerable<SelectListItem>)ViewBag.SelectOptions)

@Html.DropDownList("NewsCategoriesSelection",(SelectList)ViewBag.NewsCategoriesID

@Html.DropDownListFor(model => model.Project, Model.GetProjectList())

@Html.DropDownList("name", list.Select(x => new SelectListItem() { Value = x.Field, Text = x. Field }))

⇒ 選択されたオブジェクトが変数名 ”name” としてController のメソッドに渡される

### TextBox

@Html.TextBox("value1", "", new { maxlength = 20 })

### Label

対象プロパティの値を表示するか、プロパティ名を表示するかで変わる。

値 @Html.DisplayFor(model => model.Field)

名前 @Html.LabelFor(model => model.Field)

### Input

html の入力フォーム。

### Button

@Html では実装されていない？？ 18.08.08

type="reset" リセットボタン。それまでに入力した情報をキャンセルして初期状態にtype="button "汎用ボタンをつくります。

type="image" 画像ボタン。画像ファイルはsrc属性で指定。またalt属性が必須

type="submit" 送信ボタン

## リンク

@Html.ActionLink(“表示テキスト”,”メソッド名”,”コントロール名”)

## View メソッド

以下のurl 的に実行される？ 18.08.08

/Views/コントローラ名/アクション名.cshtml

※つまりController 内の**１つのメソッドと１つのView が結びついている**。

・オーバーロード

View(object model) 引数を渡す。

View(string viewName) メソッド名指定。複数のメソッドでViewを共有する場合。

View(IView view) ？？？

## テンプレート

partial view で追加しないといけない

# Controls

## 概要・注意点

クラス名を間違えると動かない(例：×...Controllers　○...Controller)ので、

Controllersフォルダ → 右クリック → 追加 → コントローラーとするのが無難

・メソッド名を選択→右クリックでView を追加する事もできる。

## WebAPI 基本事項

ルーティングは以下のようにしておく

“api/ {controller} / {id}}”

⇒ こうではないのか？ ”http://localhost:[port]/api/ {controller} / {id}}”

public class ProductsController : System.Web.Http.ApiController

{

public IHttpActionResult GetCount()

{

return Ok(\_item.Count);

}

}

このクラス内に記述するメソッドは、メソッド名の Prefix, パラメータ名に注意が必要！

メソッドのパラメータ名は一致している必要がある。

メソッドの種類は大きく Get, Put, Post, Delete, Send があり、それぞれの単語を Prefix に用いる事が重要。逆にそれ以外はどうでもいい。

⇒ アトリビュートでの設定も可能だが、もろもろの手間が無駄。

Get : リソースの取得のみで、リソースに影響を与えない

Put : リソースを更新

Post: 新規作成

Delete: 削除

Send: ファイルなどのバイナリファイルを送る

## 戻り値

メソッドの戻り値は IHttpActionResult インターフェイスを継承したクラス

e.g.

View: View(), View(string) など

API: Ok, Ok<T>, NotFound

## Html 本文でパラメータを取得

参考資料

<https://msdn.microsoft.com/ja-jp/library/system.web.http.apicontroller_methods(v=vs.118).aspx>

# Model

## アトリビュート

### System.Runtime.Serialization.

DataContractAttribute

DataMember

IgnoreDataMember]

※System.Runtime.Serialization.dll を参照に追加する必要がある。

### System.ComponentModel

DisplayName Label などで表示される文字列を指定する

### System.ComponentModel.DataAnnotations.

DataType 型情報。

DisplayColumn 外部キーで関連付けられた参照先テーブルの列を指定

UIHint EditorTempletes の指定　※これは極力使わず、データ型で対応する。

DisplayFormat

DataFormatString書式文字列。string.Format 形式。

e.g. DataFormatString =　"{0:yyyy年MM月dd日}")

ApplyFormatInEditMode 編集モード時に書式文字列を適用するか

ConvertEmptyStringToNull 空文字列をNothing／nullに自動変換するか

NullDisplayText 値がNothing／nullのときの表示テキスト

Range

### System.Web.Http

FromBody

参考）

<http://www.atmarkit.co.jp/fdotnet/aspnetmvc3/aspnetmvc3_07/aspnetmvc3_07_01.html>

html 電文で用いるデータクラスのサンプル

// Json でシリアライズできるようにするアトリビュート

[System.Runtime.Serialization.DataContractAttribute]

public class TestClass

{

// Json でシリアライズできるようにするアトリビュート

[System.Runtime.Serialization.DataMember]

public int ID { get; set; }

[System.Runtime.Serialization.DataMember]

//[System.Runtime.Serialization.IgnoreDataMember]

public string Name { get; set; }

}

## テンプレートのカスタマイズ

NuGet でjQuery.UI.Widgets.Datepicker ダウンロードする必要がある。

（また、jQuery の対象バージョンが必要な事も　>=1.44 && < 1.6.0

# WebMethod

[FromBody] ルートで指定できない引数をhtml 本文から取得する。

※Json形式で送られるため、送信されるデータ型のプロパティはすべてJsonでシリアライズできるようにアトリビュートを付ける必要がある。

[FromRoute]　 その逆にルートからの引数である事を明示する。使わないような？

# JavaScript

## セレクタ

|  |  |
| --- | --- |
| .  (ドット) | クラスを表す  e.g. $(‘.cal’) ⇒ calクラス　$(‘span.red’) ⇒ spanタブ内のred クラス |
| # | ID を表す |
|  |  |
|  |  |
|  |  |

# jQuery

## 概要

軽量なJava Script ライブラリの事。

通常のhtmlでは以下のようにして読み込む

<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.0.3.min.js"></script>

ASP.NET プロジェクトでは標準で組み込まれている。（Scriptsフォルダ）

バージョンは　1.x ⇒ IE6以降 2.x ⇒ IE9以降

但し、1.x 系のまだまだ切り捨てられない。

たまにNuGet でjQuery.UI.Combined を更新した方が良い。

## 読み込み

・html で直接読み込む

<link href="@Url.Content("~/Content/themes/base/minified/jquery-ui.min.css")" rel="stylesheet" />

<script src="@Url.Content("~/Scripts/jquery-1.12.4.min.js")"></script>

・BundleConfig.cs で読み込む

（参考）[BundleConfig.cs](#_BundleConfig.cs)

## 具体例

<script>

$(function() {

$("#datepicker").datepicker();

});

</script>

html では以下のように実行。メソッド名は大文字小文字区別する。

<input type="button" value="get" onclick="get();" />

jQueryUI 公式： https://jqueryui.com/

jQueryUI 逆引き： <http://alphasis.info/jquery-ui/gyakubiki/>

# Ajax

Asynchoronous Javascript + Xml

# HTML

## 備忘録

<!-- --> コメント

formmethod タグ getとかput とかの指定の事

・input でViewBag使う方法

<input id="jquery-ui-datepicker" value="@(ViewBag.Date)">

## 注意点など

・form タグは入れ子にできない。

## HTML5

対応ブラウザ：

IE v8.0～ （※HTML5の新要素の認識など実質的な対応はv9.0から）

Firefox v3.1～

Opera v9.6～

Google chrome v3.0～

### 属性

formaction 例えば同じpostmethod でも、別のaction（メソッド名）を選択できる

**クライアント**

JSON のシリアライズ

・JSONに載せられるデータ型は実質的に、数値と文字列とそれらの配列だけ

・シリアライズするクラスには以下のようにアトリビュートを付ける必要がある

[DataContract]

internal class Person

{

[DataMember]

internal string name;

[DataMember]

internal int age;

}

* .NET Core では設定ファイルはJSON

# ブラウザ別の対応

## IE

・IE9互換モード

<meta http-equiv="x-ua-compatible" content="IE=9">

<meta http-equiv="x-ua-compatible" content="IE=EmulateIE9">

**疑問点**

・Form の共有（e.g. Edit とAdd ではほとんど同じフォーム）はできるか

・サーバーでApplication Directory を取得するには？

・データコンテキストクラスとは

・DatePicker の使い方

・DataGridView のようなものは表示可能か

・同時アクセスの場合のasync とかはどうやって実装？

・css のようなものはどこにあるのか。→ Content

# 参考url

ASP.NET MVC 開発を始める前に理解しておきたいこと

<https://qiita.com/kazuhisam3/items/f056819172d2b6d36a8c>

//WebAPI の作り方（英語）

https://docs.microsoft.com/en-us/aspnet/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api

// Web Client の作成方法

https://qiita.com/yz2cm/items/46d8e87a39f844904ea7

IISExpress\config\applicationHost.config

で設定をする。

<h2>日報入力</h2>

<div>

<p>@Html.EnumDropDownListFor(model => model.ReportType)</p>

@using (Html.BeginForm())

{

@Html.TextBox("value1", "", new { maxlength = 20 })

<input type="submit" value="送信" />

}

</div>