ASP.NET MVC を もっと使ってみたよ！

しばやん

注意！

- 前回のLTを聞いてる人には繰り返しになるかも知れない
  - 聞いてなかったふりのご協力を
- 前で喋ってる人間はまだ勉強中です
  - 間違いもあると思いますが、見つけたときはやさしく教えてあげてください

ASP.NET MVC とは

- ASP.NET 4.0 で入る予定のフレームワーク
  - AJAX と同じく単体で先行リリースされました
- Web Form とは違い、HTML (+ CSS と JS) を書かないとだめ
  - ぽとぺたで出来るわけ無い
  - より Web の知識が要求されます
- 構成要素が分離されていてわかりやすい
  - テストが容易
  - TDDについては今回説明しません

MVC ってなによ？

- Model View Controller の略
  - モデル: DB からデータ引っ張ってきたりする
  - ビュー: 最終的に表示するもの
  - コントローラ: うまいことモデルとビューをつなぎ合わせる
- ASP.NET MVC が提供するのは V と C
  - Model は LINQ to SQL など

ASP.NET MVC の構造は？

- ルーティング
  - 定義に従いコントローラとアクションを決定する
- コントローラ
  - アクションを持つクラス
  - モデルからデータ持ってきたり、ビューを呼んだり
- ビュー
  - 最終的に出力される HTML
  - ASP.NET MVC では aspx と ascx になります

ASP.NET MVC の仕組みは？

- ルーティングエンジン が URI からコントローラとアクションを決定する
  - デフォルトでは /{controller}/{action}/{id}
- 決定されたコントローラクラスをインスタンス化し、アクションメソッドを呼び出す
  - アクションの引数はルーティングで指定したパラメータと URI のクエリパラメータと対応する
- アクション内で DB からデータ引っ張ってきたりして、ビューに引き渡したりリダイレクトしたり
- ビュー (aspx や ascx) 内でデータを展開する
- レスポンスとして返す

ルーティング

- デフォルトは /{controller}/{action}/{id}
  - /users/details/shiba-yan は
    - controller = users
    - action = details
    - id = shiba-yan になる
  - この場合だと UsersController.Details(string id) が呼ばれる
- デフォルト値や条件なども指定できる
  - デフォルトの定義では / の時は HomeController.Index が呼ばれる
  - 「id は数値のみ」といった条件を正規表現で書けます
- ちなみに case-insensitive です

ルーティングを定義する

- MapRoute を使います
  - MVC FW で定義された拡張メソッド
  - 結構面倒なルーティングを楽に追加できます
- 基本は MapRoute(名前,URLパターン)
  - デフォルト値や条件を指定できるメソッドも用意されてます

コントローラ（とアクション）

- 規約があります
  - Controllers ディレクトリ内の コントローラ名 + Controller が対象になる
  - つまり Users + Controller = UsersController
- 当然ながらコントローラクラスとアクションメソッドは public で
  - リフレクション使って呼び出している。はず……
- VS に作ってもらうのが基本です

ActionResult

- アクションは ActionResult を継承したクラスを返すようにしないとだめです
  - return View(); は ViewResult を作成して返しています
- ビュー以外にリダイレクトやファイルなど返したいときはどうするん？
  - View メソッド以外にも File/Redirect/Json/JavaScript/Content メソッドなどが用意されてます
    - 意味は名前から読み取ってください

ASP.NET MVC と属性

- コントローラやアクションには属性をつけることが出来ます
  - <% OutputCache %> などは使いません
- で、どんな属性があるん？
  - いろいろあるし、自分でも作れます
    - 自分で作る時は ActionFilterAttribute を継承
    - これ以上は説明しません
- では、基本的な属性の紹介始まり

属性(1)

- AcceptVerbs
  - 受け入れる HTTP メソッドを制限する
- ActionName
  - アクション名を指定する
- Authorize
  - 認証が必要にする
- HandleError
  - 例外をハンドリングする
  - 捕まえる例外や表示するビューを指定できる

属性(2)

- OutputCache
  - 出力キャッシュを設定する
- ValidateAntiForgeryToken
  - CSRF 対策で埋め込んだトークンを検証する
  - Html.AntiForgeryToken と併用する必要があります
- ValidateInput
  - 危険な文字列の混入を検証するか指定する
- などなどあります

モデル

- LINQ to SQL や Entity Framework を使うのが定石
  - 海外だと Entity Framework 使ってる例が多いですね
  - 今のバージョンは超絶使いにくいです
  - MS 先生の次のバージョンに期待しましょう
- テストのことを考えると Repository で
  - コントローラに DataContext や Entities を書くのはやばい

モデルバインダ

- controller と action 以外のパラメータの型はアクションの引数の型に自動的に変換される
  - Nullable<T> 使っても平気。便利すぎます
- 独自のクラスでも引数として使える
  - デフォルトでは対応する名前のプロパティにマッピングされる
  - 独自のモデルバインダを登録することで特殊な変換も出来ちゃう
- (Try)UpdateModel メソッドも用意されてます
  - エンティティクラスに対して Update とか

ビュー

- これも規約あります
  - Views/(コントローラ名|Shared)/アクション名.aspx など
  - Shared 下に置くと、全コントローラから使うことが出来る
    - ビューの検索順序は コントローラ名 以下 -> Shared の順
- <% %> タグを使ってじゃんじゃん埋め込む
  - HtmlHelper<TModel> という便利アイテムもあるので、もちろん使います
    - AjaxHelper もありますが残念ながら(ry
- VS に作ってもらうのが基本です
  - Ctrl+M,V で作れます
  - scaffolding にも対応している
    - (専用のテンプレート作れば)楽です

モデルとビューデータ

- ビューに値を渡す方法は 2 種類あります
  - ViewData コレクションに値を突っ込む
  - View メソッドの引数として渡す
- ビューから参照する方法
  - ViewData からキーを指定して参照する
  - Model プロパティを直接参照する
- ViewData は object なのに対し、Model は型を指定出来る

ビューモデル

- こんな時、どうします？
  - Model には IEnumerable<T> を入れているが、ページャを付けるためにページ情報を追加したい。
  - 普通に考えると Model に IEnumerable<T> を、ViewData にページ情報を入れればいい
    - でも ViewData って object 返すので、使うときにキャストしないとダメ
  - じゃあ、IEnumerable<T> とページ情報を持つクラスを新しく作ろう！
  - ビューとモデル間でやり取りするのでビューモデル
- MVC には含まれてないです

ヘルパークラス

- ViewPage には 3 種類のヘルパークラスを扱うためのプロパティが用意されています
  - Ajax/Html/Url の 3 プロパティ
  - 実体は AjaxHelper/HtmlHelper/UrlHelper
- よく使うのは HtmlHelper
  - リンクを作ったり、フォームを定義したりするときに使いまくります
  - <%= Html.ActionLink("ホーム", "index", "home") %> のように埋め込むだけ

Ajax

- MicrosoftAjax を簡単に使うためのヘルパーメソッドが用意されてます
  - AjaxHelper を ViewPage.Ajax 経由で使うこと出来ます
    - Ajax.ActionLink/RouteLink/BeginForm/BeginRouteForm を使って非同期処理を行います
- Ajax と言っても、一番単純な XHR で HTML 断片を取得する機能だけです
  - HTML 断片はパーシャルビューで定義して、PartialView 返してやるだけ

パフォーマンス

- Html.ActionLink と Url.Action は遅い
  - そりゃ、コントローラ・アクション・パラメータ名まで探索してたら遅い
- Html.RouteLink と Url.RouteUrl は早い
  - よく使うヘルパーメソッドなので注意しましょう
- やっぱりキャッシュしよう
  - <% OutputCache %> は使いません
  - OutputCache 属性を使います

結局、誰うま？

- HTML + CSS で普通にデザインやってた人
  - Web Form の時よりは HTML の知識が必要です
  - デザイナとの連携がしやすいんじゃないかな
- 俺の書いた完璧な HTML にイミフなコード勝手に埋め込むんじゃねぇよという人
  - 隠しフィールドとか埋め込まれますよね、大量に
- 大量のビューステートが生成されるのがいやな人
  - 巨大なビューステートは携帯にとっては悪
- かっこいい URI を使いたい人（俺
  - /User.aspx?id=shiba-yan -> /shiba-yan/