MVC Notes

Opening slide

1. Orig drop in Dec, Preview 2 came out in march.
2. New preview code drop came out on Thursday (4/17)
3. Preview 3 out on May 27

About Me

1. Live in Hilliard
2. Family, Disney addicts
3. Hobbies, brewing, some gaming, and…

Hockey

1. Love hockey
2. Play, coach, watch, etc, etc

Work

1. 10 years in IT in Central OH
2. 4 years with Quick
3. All in BSG at Q, group stated by BHP

MVC Diagram

1. Smalltalk ‘79
2. Proven pattern
3. Separation of concerns

Controller

1. Runs the show
2. Handles request
3. Determines view
4. If model is needed, gets info from model
5. Returns request

View

1. Presentation to the user
2. No Business Logic, only presentation logic
3. Asp.net version can be strongly typed

Model

1. Returns info
2. Unaware of requestor
3. Consists of everything that’s not in the controller or view

[MVC Framework Slides] – Peer Pressure

1. Python
2. PHP – 35 implementations on wikipedia
3. Java
4. Ruby

Letters

1. So, the requests poured in to MS to have an MVC implementation
2. Monorail points

What’s the Problem

1. “What are we trying to solve? What’s wrong with WebForms?”
2. WebForms tried to move WinForms to web development
3. Introduced page event model
4. Some developers think this is the way the whole web works

ASP.Net Lifecycle

1. Who doesn’t love this?
2. Was that OnPreInit or OnPreLoad?
3. Page Lifecycle brings state to the web, not totally a bad thing, but we all know the web is stateless
4. Runat=server
5. Naming Container in javascript <% %> all over the place

Page Load

1. The spaghetti of the page\_load. (A steaming pile of Page\_Load)
2. Neal Ford: “Bad developer will move heaven and Earth to do the wrong thing.”
3. Mix of presentation and logic in the code behind, no Separation of Concerns
4. Extremely difficult to test code behinds
5. Dependent on web server to run
6. Concrete classes on the web server run WebForms
7. No way to fake or mock those classes

NEW!

1. New offering from MS
2. Shared source project (Similar to AjaxControlToolkit)
3. Not replacing web forms, alternative
4. Not WebForms 4.0

Increased Testability

1. Encourage use of OOP principles
   1. Separation of Concerns
   2. Single Responsibilty
   3. Dependencies are abstracted out to HttpContextBase, HttpRequestBase, etc
   4. All Http classes now easily mocked or faked (MVC Mock Helpers)

More control over your HTML

1. Bye bye naming container
2. No more control.ClientId
3. Separation of concerns, keep business logic out of your presentation code
4. Use of CSS easier

Routing

1. Maps incoming URLs to the correct Controllers
2. Maps outgoing URLs so the can be called back
3. Originally part of mvc namespace, has been moved “up” to web namespace
4. RESTFUL – uses get and post out of the box (Adam Tybor’s blog post)

<code />

1. Out of the box – (Default routing, home controller, home/about views)
2. Test Project
3. Webforms View Engine
4. [See Demo 1 notes]

Request/Response

1. Back to Requset and Response
   1. Request comes in
   2. Controller interprets the request and decides what to do
   3. Application does something cool
   4. Response is sent back to the Internets
2. No Postback
3. Page lifecycle still happens (look at trace)
4. Still based on asp.net

You have options

1. Quick and dirty, plain old html. Id replaced by classic asp-ish code
   1. Will work, but magic strings everywhere
   2. If your controller changes, refactoring here will be uncovered at runtime
2. Better way, as controller is implied. (Unless needed to move to a different controller)
   1. Controller changes will again be uncovered at runtime.
3. Best way, the view is strongly typed to the controller.
   1. Controller change here will break the build…woo-hoo! Fail early!

MVC contrib

1. Open source project on CodePlex
2. Adding code for MVC to work with Castle, Spring.Net, Structure Map, etc.
3. Original HtmlHelpers folded into release 2, MvcContrib picking up where they left off
4. Add on code for VS and for R#

MVC Circles – Out of the box?

1. What do you get out of the box?
   1. Working Sample App
      1. Default Routing
      2. Home Controller
      3. Home and About Views
   2. WebFormsViewEngine
   3. Test Project
2. What’s Going on?
   1. Browser makes request (use Products)
   2. Route is determined
   3. Controller Activated
      1. Gets data from Model
      2. Determines View
   4. Renders View

Ted Nugent slide

1. Still going to fit larger, data driven sites better. (At least for now.)
2. Much more mature framework to build ASP.Net apps on
3. Quicker deployment, more drag and drop
4. More Better UI controls available.
5. Lots of 3rd party controls available…like our Friends at Telerik
6. Better Documentation
7. More examples and code samples to draw from
8. And, of course, the update panel!