



REALDOLMEN

Introducing the ASP.NET MVC framework

Maarten Balliauw – RealDolmen
Website: www.realdolmen.com
E-mail: maarten.balliauw@realdolmen.com
Blog: http://blog.maartenballiauw.be

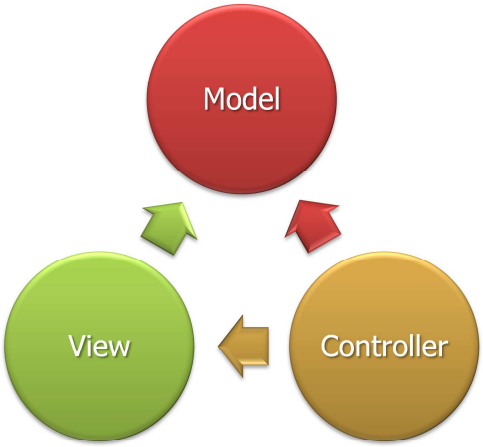
AGENDA

- MVC
- ASP.NET MVC
- Hello world!
- Routing
- Extensibility
- Testing
- …

REALDOLMEN

MVC – Model-View-Controller

- Design pattern
- Isolating business logic from UI
- Maintainability
- Layering for UI layer



October 15, 2008 | Slide 4

REALDOLMEN

Model-view-controller or MVC in short, is a pattern used in software engineering. The main purpose of this pattern is isolating business logic from the user interface, focusing on better maintainability and cleaner structure of the application. The MVC pattern consists of 3 key parts: the model, the controller and the view.
The model consists of some encapsulated data with its processing and is isolated from the manipulation logic, which is encapsulated in the controller. Presentation logic is located in the view component.

The model object knows all the data that needs to be displayed. It can also define some operations that manipulate this encapsulated data. It knows absolutely nothing about the graphical user interface: it has no clue on how to display data or respond to actions that occur in the GUI. An example of a model would be a calculator: it contains data (a numeric value), some methods to manipulate this data (add, subtract, multiply, divide) and a method to retrieve the current value from this model.

The view object refers to the model: it uses read-only methods of the model to query and retrieve data from it. It can look like anything: a HTML page, a Windows GUI or even the LED display of a physical calculator.

The controller object is the interaction glue of the model and the view. It knows that the model expects actions like add, subtract, multiply, divide and that the GUI will send some events that may require these operations on the model to be called. In the calculator example, clicking the “+” button on the view will trigger the controller to call the add method on the model and re-render the view with updated date if needed.

Applications are commonly split into separate layers: presentation, domain logic and data access. The MVC pattern fits into the presentation layer, where it handles a user’s interaction (controller) with a specific model trough a view. It can often be found in a web application where the view is rendered as HTML, the controller sits on the web server and communicates with the business layer and domain model.

ASP.NET MVC

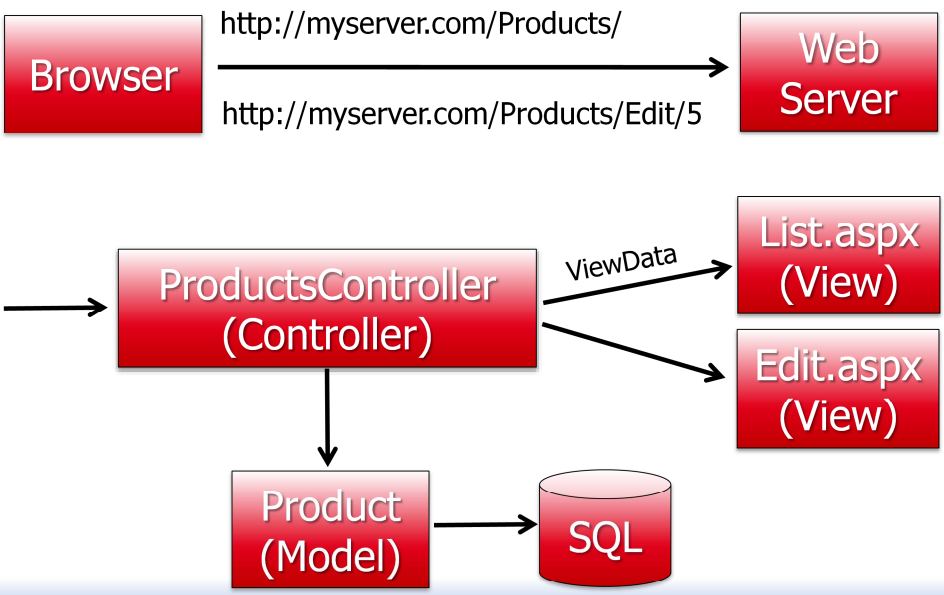
- A new option for ASP.NET
- Simpler way to program ASP.NET
- Easily testable and TDD friendly
- More control over your <html/>
- More control over your URLs
- Not for everyone! (car vs. motorcycle)
- Not a replacement for webforms!
- Supports existing ASP.NET features

WHAT YOU WILL LOSE…

- Viewstate
- Postbacks
- Tricky interview questions about nuances of the page lifecycle

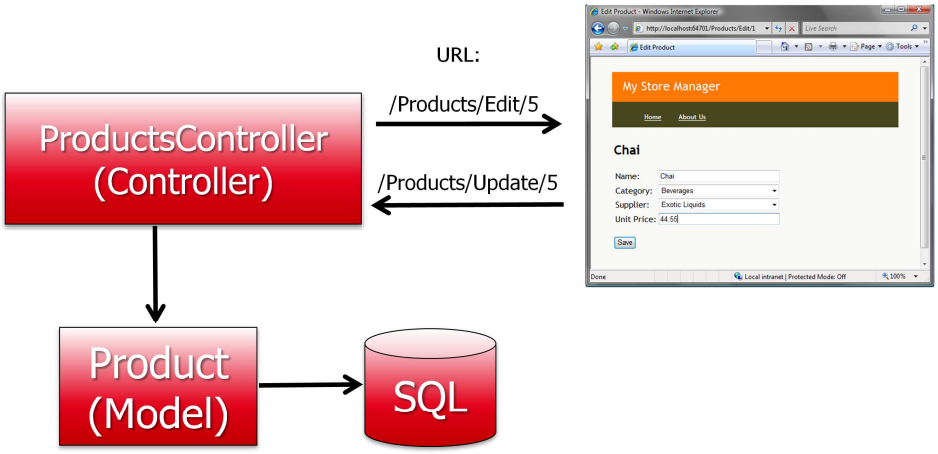
REALDOLMEN

How it works…



REALDOLMEN

INTERACTIONS



REALDOLMEN



-Demo on creating a new MVC project (project template)
-Test project picker in regular release

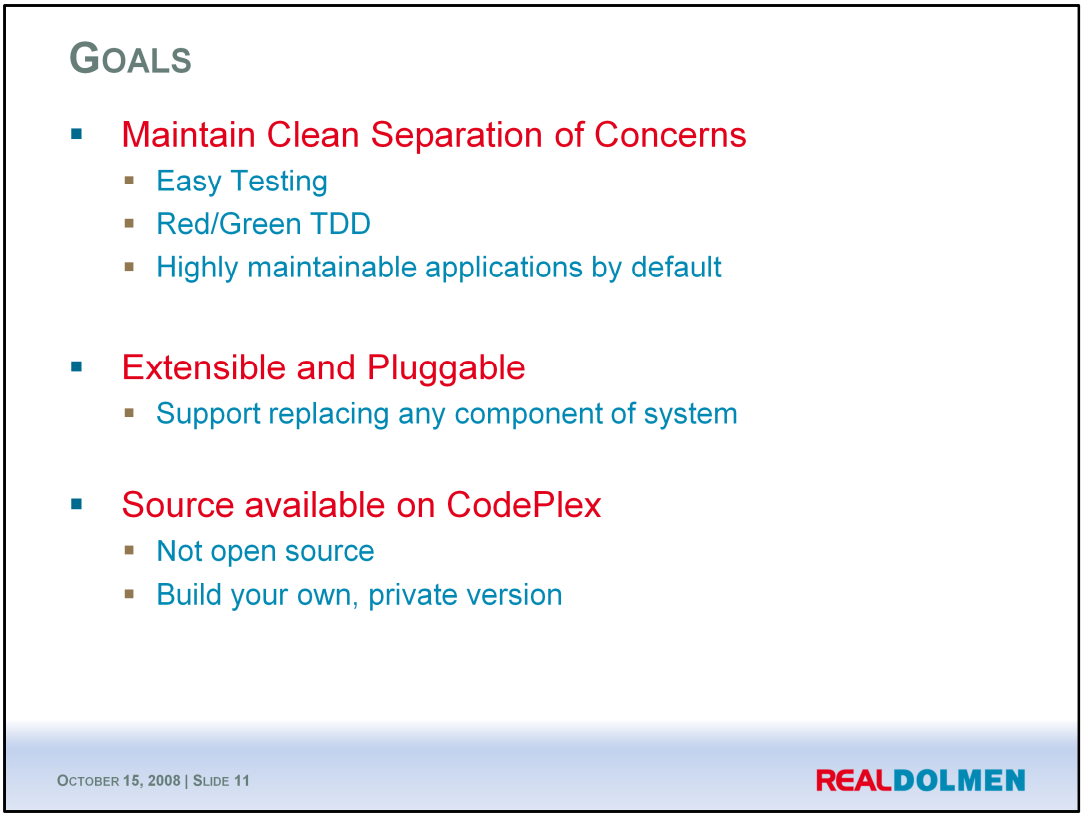
-References of the project: System.Web.Abstractions, System.Web.Routing, System.Web.Mvc

-Explain folders → Models, Views, Controllers
-Run → /Home/Index (HomeController, Index action)
-Something with a strong-typed view

WHAT'S THE POINT?

- It's not WebForms 4.0!
  - It's an **alternative**
- Flexible
  - Extendible in each part of the process
- Fundamental
  - Only System.Web! (Not WPF or Winforms)
- Plays well with others
  - NHibernate for models, NVelocity for views, Unity for dependency injection, …
- Keep things simple!

REALDOLMEN



-Model has one purpose
-Controller has one purpose
-View really handles the view, no business logic in there!

-Each part is replacable
-Source code available on CodePlex. Not open source, but enables you to build your own, private flavour

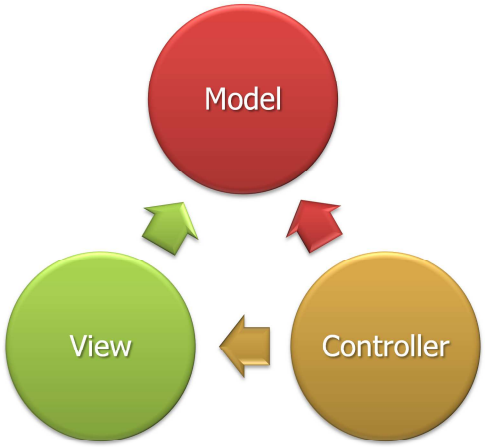
DRIVING GOALS

- Enable clean URLs and SEO
  - SEO friendly URL structures
- Great ASP.NET integration
  - All providers are supported
  - Membership, Caching, Session, …
  - ASP.NET designer in VS2008
  - Check my blog!
    - http://tinyurl.com/maarten-outputcache
    - http://tinyurl.com/maarten-mvcsitemap

REALDOLMEN

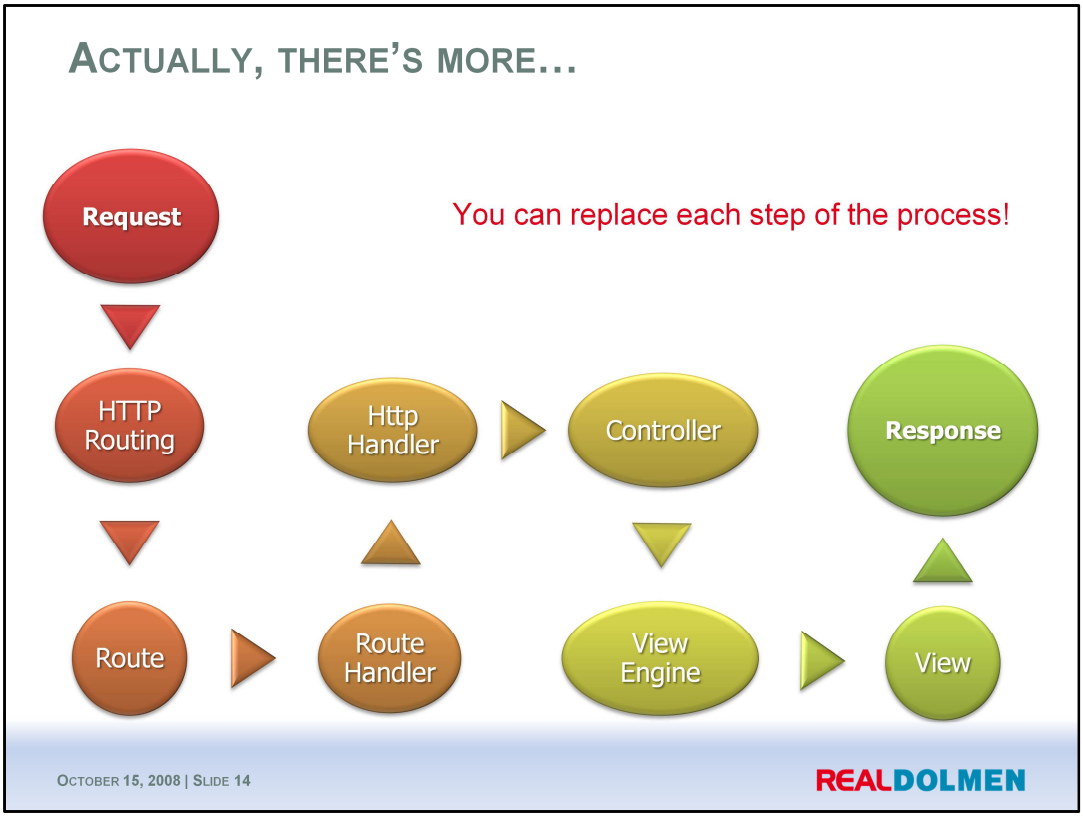
-Users expect clean URLs nowadays
-Not replacing ASP.NET → built on top of it

REQUEST LIFECYCLE

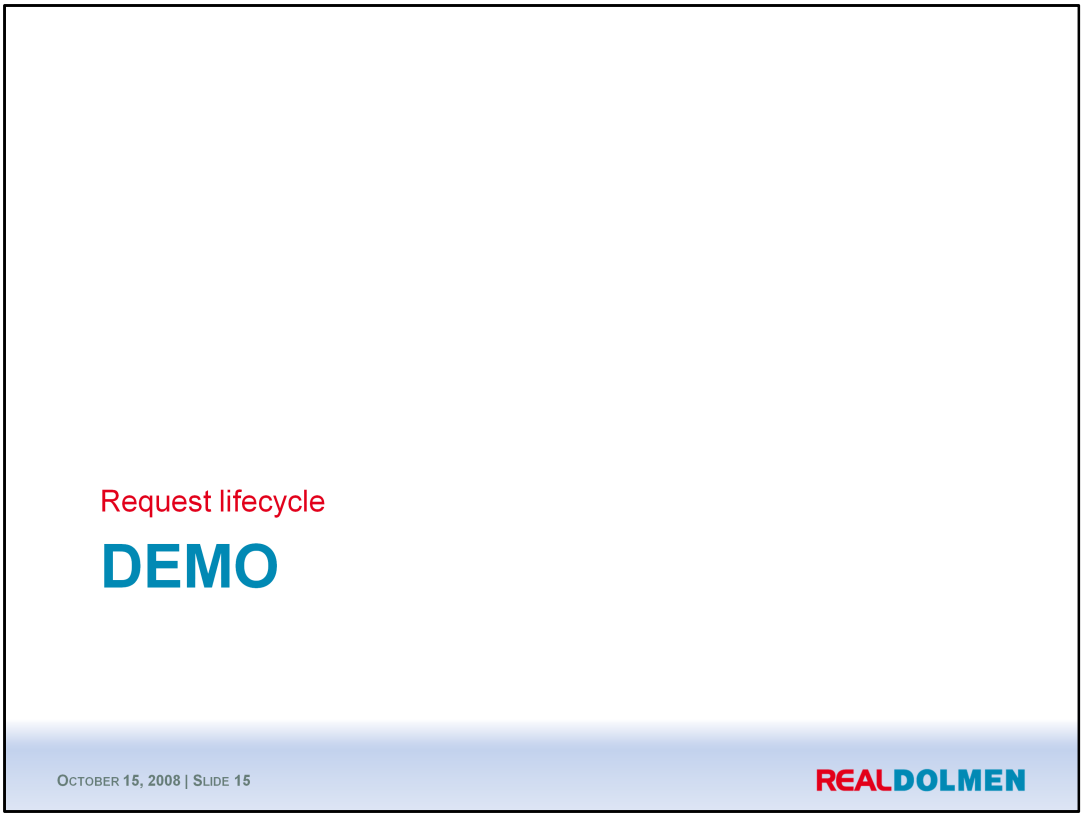


- Browser requests /Products/
- Route is determined
- Controller is activated
- Method on Controller is invoked
- Controller does some stuff
- Renders View, passing in ViewData

REALDOLMEN



-Defaults
-Every step can be replaced
-Actually, there's even more… Action Filters!



-Breakpoint on return View()
-Call stack → show external code
-System.Web.Mvc HttpHandler
-Mixing System.Web and System.Web.Mvc is possible

-View() → folders /Views/Controller/Action.aspx or /Views/Shared/Action.aspx by default
-Not found? You get a clear error message!

-Show contents of the view
-Run it, show View Source → clean!

ROUTING – PRETTY URLS

- Developer adds Routes to a global RouteTable
- Mapping creates RouteData – a bag of key/values

```
public static void RegisterRoutes(RouteCollection routes) {
  routes.MapRoute(
    "Default
    "{controller}/{action}/{id}",
    new { controller = "Home", action = "Index", id = "" },
    new { controller = @"[^\.]*" }
  );
}
```

- This is **NOT** URL rewriting! It's more!

REALDOLMEN

URL routing in ASP.NET differs from other URL rewriting schemes. URL rewriting typically does not have an API for creating URLs based on your patterns. URL rewriting processes incoming requests by actually changing the URL before it sends the request to the Web page. For example, an application that uses URL rewriting might change a URL from /Products/Widgets/ to /Products.aspx?id=4. In URL rewriting, if you change a URL pattern, you must manually update all hyperlinks that contain the original URL.

With URL routing, the URL is not changed when an incoming request is handled, because URL routing can extract values from the URL. When you have to create a URL, you pass parameter values into a method that generates the URL for you. To change the URL pattern, you change it in one location, and all the links that you create in the application that are based on that pattern will automatically use the new pattern.

→ ActionLink

CUSTOM URL ROUTING

- Can setup custom URL routing rules within the RegisterRoutes() method in Global.asax
- Enables very flexible mapping and route rules:
  - Default parameters
  - Constraints
  - Named routes
  - Wildcard “catch-all” routes

REALDOLMEN



-Show route tables
-Create fuzzy new route (i.e. Show/Me/The/Index)
-First route wins!
-Create new route for /Home/About → /About
-What if it is not defined? Error? 404? Perhaps catch it all! {*catchall}

```
routes.MapRoute(
            "Whatever",                                 // Route name
            "{*fullpath}",                           // URL with parameters
            new { controller = "Home", action = "About", id = "" },  // Parameter
defaults
            new { fullpath = @"(.*)\.htm$" }
        );
```

-Show creation of link in master page

Extensibility

- These are all pluggable!
  - Views
  - Controllers
  - Models
  - Routes
- Check my blog!
  - http://tinyurl.com/maarten-mvccustomviewengine
- MvcContrib
  - http://www.codeplex.com/MvcContrib

REALDOLMEN

VIEWENGINE, VIEW

- View engines instantiate views
- Views render output
- You get WebForms by default
- Can implement your own
  - MvcContrib has them for Brail, NHaml, NVelocity, …
- Customizing these can be used to
  - Offer new DSLs to make HTML easier
  - Generate totally different mime/types
    - Images, RSS, JSON, XML, OFX, VCards, DOCX, whatever

```
public abstract class VirtualPathProviderViewEngine : IViewEngine
{
    protected abstract IView CreatePartialView(...);
    protected abstract IView CreateView(…);
}

public interface IView
{
    void Render(ViewContext viewContext, TextWriter writer);
}
```

NHAML – FROM…

```
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master"
AutoEventWireup="true"
    CodeBehind="List.aspx"
Inherits="MvcApplication5.Views.Products.List" Title="Products" %>
<asp:Content ContentPlaceHolderID="MainContentPlaceHolder"
runat="server">
  <h2><%= ViewData.CategoryName %></h2>
  <ul>
    <% foreach (var product in ViewData.Products) { %>
      <li>
        <%= product.ProductName %>
        <div class="editlink">
          (<%= Html.ActionLink("Edit", new { Action="Edit",
ID=product.ProductID })%>)
        </div>
      </li>
    <% } %>
  </ul>
  <%= Html.ActionLink("Add New Product", new { Action="New" }) %>
</asp:Content>
```


REALDOLMEN

NHAML – TO…

- http://code.google.com/p/nhaml/

```
%h2= ViewData.CategoryName
%ul
    - foreach (var product in ViewData.Products)
  %li = product.ProductName
    .editlink
       = Html.ActionLink("Edit",
           new { Action="Edit",
           ID=product.ProductID })
       = Html.ActionLink("Add New Product",
           new { Action="New" })
```


REALDOLMEN

SPARK – FROM…

```
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master"
AutoEventWireup="true"
    CodeBehind="List.aspx"
Inherits="MvcApplication5.Views.Products.List" Title="Products" %>
<asp:Content ContentPlaceHolderID="MainContentPlaceHolder"
runat="server">
  <h2><%= ViewData.CategoryName %></h2>
  <ul>
    <% foreach (var product in ViewData.Products) { %>
      <li>
        <%= product.ProductName %>
        <div class="editlink">
          (<%= Html.ActionLink("Edit", new { Action="Edit",
ID=product.ProductID })%>)
        </div>
      </li>
    <% } %>
  </ul>
  <%= Html.ActionLink("Add New Product", new { Action="New" }) %>
</asp:Content>
```


REALDOLMEN

SPARK – TO…

- http://dev.dejardin.org/

```
<viewdata model="MvcApplication5.Models.Product"/>

<h2>${ViewData.CategoryName}</h2>
<ul>
  <for each="var product in ViewData.Products">
    <li>
      ${product.ProductName}

      <div class="editlink">
        (<a href="/Products/Edit/${product.ProductID}">Edit</a>)
      </div>
    </li>
  </for>
</ul>

<a href="/Products/New">Add New Product</a>
```

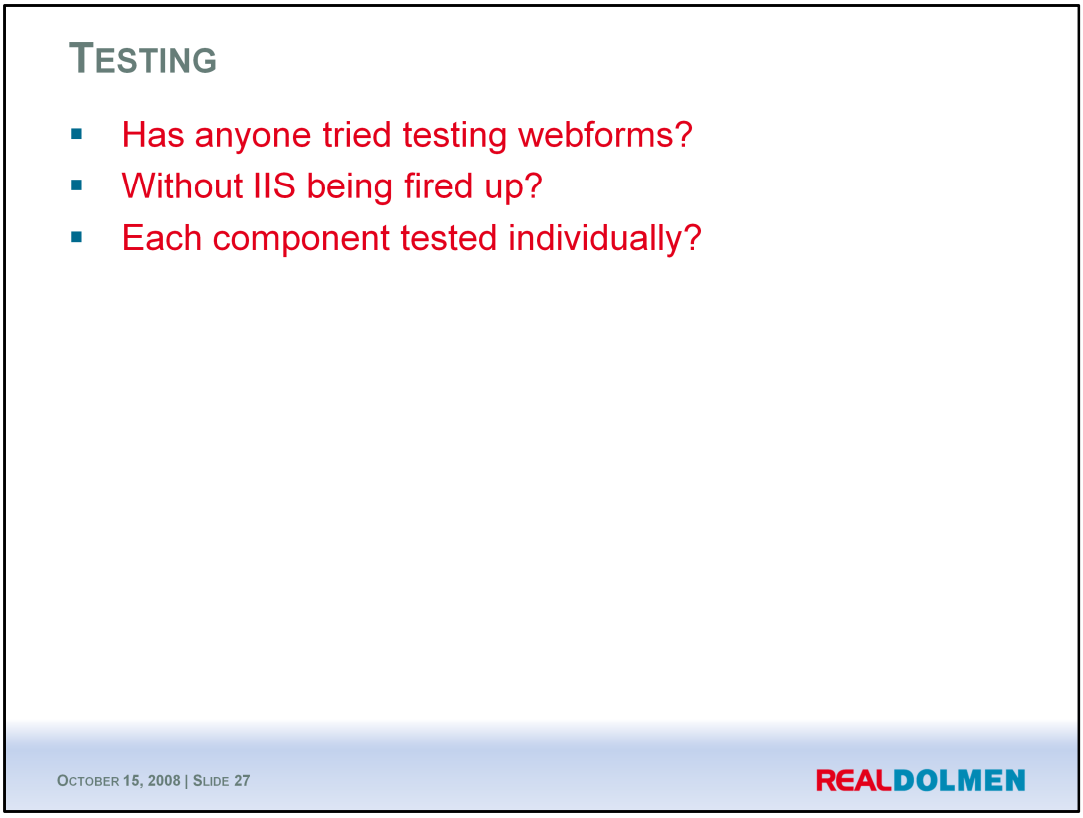

REALDOLMEN

FILTER ATTRIBUTES

- Enable custom behavior to be added to Controllers and Controller actions
- 4 types
  - Authentication filter
  - Action filter
  - Result filter
  - Exception filter
- Examples:
  - [Authorize], [LogAction], [OutputCache], …
  - http://tinyurl.com/maarten-outputcache




```
public class TitleFilter : ActionFilterAttribute
  {
     public override void OnActionExecuted(ActionExecutedContext filterContext)
     {
        ViewResult viewResult = filterContext.Result as ViewResult;
        if (viewResult != null) {
           viewResult.ViewData["Title"] += " - Demo of MVC";
        }
     }
  }
```



Let me trick you…

INTERFACES AND TESTING

- These are all easily mockable!
  - HttpContextBase, HttpResponseBase, HttpRequestBase
- Extensibility
  - IController
  - IControllerFactory
  - IRouteHandler
  - IViewEngine

Testing controller actions

- No requirement to test within ASP.NET runtime!
  - Can mock parts of runtime you want to fake
  - Using Moq, Rhino, TypeMock, …
    - http://code.google.com/p/moq/

```
[TestMethod]
public void ShowPostsDisplayPostView()
{
   BlogController controller = new BlogController(…);

   var result = controller.ShowPost(2) as ViewResult;

   Assert.IsNotNull(result);
   Assert.AreEqual(result.ViewData["Message"], "Hello");
}
```

Testing controller actions using Moq

```
[TestMethod]
public void TestInvalidCredentials()
{
    LoginController controller = new LoginController();

    var mock = new Mock<System.Web.Security.MembershipProvider>();
    mock.Expect(m => m.ValidateUser("", "")).Returns(false);
    controller.MembershipProviderInstance = mock.Object;

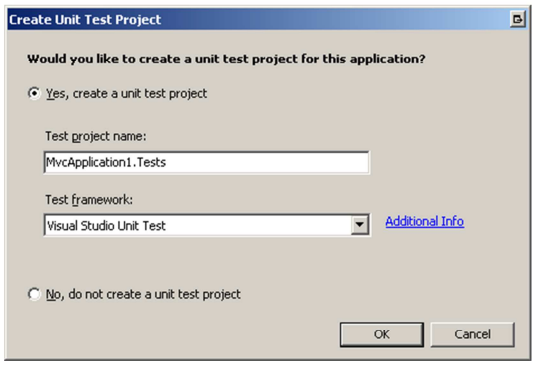
    var result = controller.Authenticate("", "") as ViewResult;

    Assert.IsNotNull(result);
    Assert.AreEqual(result.ViewName, "Index");
    Assert.AreEqual(controller.ViewData["ErrorMessage"], "Invalid
credentials! Please verify your username and password.");
}
```


REALDOLMEN

TESTING FRAMEWORKS

- Any framework is supported!
  - Visual Studio Test
  - NUnit
  - XUnit
  - …



REALDOLMEN



-Demo helpers → webforms can not be rewritten, so special interfaces for ASP.NET MVC have been created
-Demo Moq → based on expectations
-Demo FormsAuthentication default stuff

```
        // Setup
        var formsAuthenticationMock = new Mock<IFormsAuthentication>();
        var membershipProviderMock = new Mock<MembershipProvider>();

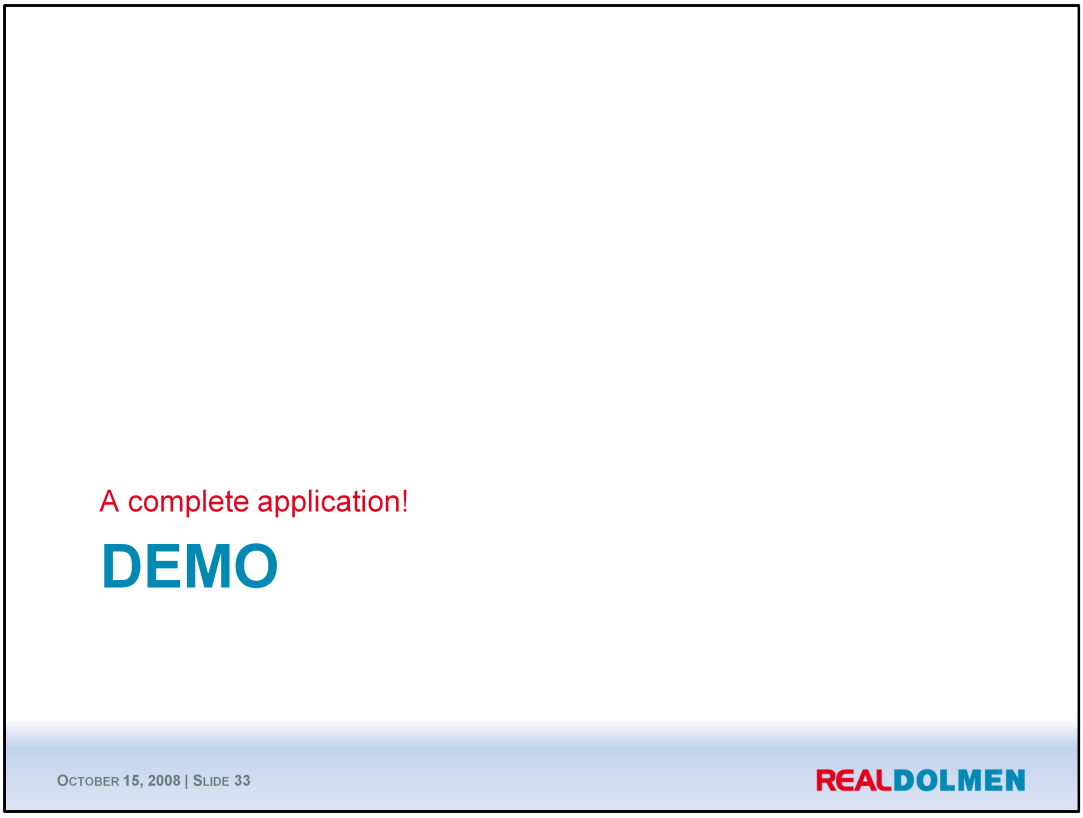
        AccountController target = new
AccountController(formsAuthenticationMock.Object,
membershipProviderMock.Object);
        target.SetFakeControllerContext();
        target.HttpContext.Request.SetHttpMethodResult("POST");

        // Execute
        string username = "";
        string email = "test@example.com";
        string password = "testing123";
        string confirmPassword = "testing123";

        ViewResult result = target.Register(username, email, password, confirmPassword)
as ViewResult;

        // Verify
        ViewDataDictionary viewData = result.ViewData;
        Assert.AreEqual("Register", viewData["Title"]);
        Assert.AreEqual(1, ((List<string>)viewData["errors"]).Count);
```

-Unfortunately not on classes from mscorlib.dll



CarTrackr is an online software application designed to help you understand and track your fuel usage and kilometers driven. You will have a record on when you filled up on fuel, how many kilometers you got in a given tank, how much you spent and how much liters of fuel you are using per 100 kilometer.

CarTrackr will enable you to improve your fuel economy and save money as well as conserve fuel. Fuel economy and conservation is becoming an important way to control your finances with the current high price.

Data layer
-LINQ to SQL → DataClasses

Domain/business layer
-Repositories → "Mediates between the domain and data mapping layers using a collection-like interface for accessing domain objects." → http://martinfowler.com/eaaCatalog/repository.html
-IDataBoundRepository → LINQ to SQL based
-I…Repository → Domain
-Testing

Controllers
-CarController → Requires repositories
-Filled via custom controller factory → UnityControllerFactory
-Global.asax → Unity registrations

Views
-Regular views
-Ajax (/Views/Car/List.aspx)
-Partial views (/Views/Car/GeneralCarDetails.aspx)

Form validation
-RuleViolation
-RuleViolationException
-IRuleEntity → /Domain/Car.cs
-CarController -> Edit
-Extension methods

SUMMARY

- A new option for ASP.NET.
- More control over your <html/> and URLs
- More easily testable approach
- Not for everyone – only use it if you want to
- Shipping later this year

REALDOLMEN

Common questions

- Should I use WebForms or MVC?
- Are there any controls?
- Do I have to write <%= %> code in my view?
- What about AJAX?
- How fast/scalable is it?
- Is it safe?
- When will it ship?

REALDOLMEN

-Should I use WebForms or MVC?
If you are using business applications with datagrids, master-detail stuff, …, maybe ASP.NET MVC is not for you.
If you are XHTML and CSS fan, want to easily test web applications, …, maybe ASP.NET MVC is for you.
Play with it and find out!

-Difference between MVC and MVP?
Model-View-Controller vs. Model-View-Presenter (patterns & practises)
Similar, but other responsibilities in presentation. Presenter also supports a sort of postbacks driven by events on the view.

-Are there any controls?
Still working on that… There is partial rendering, and classic ASP.NET controls as long as they do not need viewstate.
Use JS libraries? Ext-JS

-Do I have to write <%= %> code in my view?
No! All sort of view engines available, or write your own.

-What about AJAX?
AJAX: Yes! Client JS libraries (DoJo, jQuery, MS AJAX, Ext-JS, …)
AjaxHelper available for postbacks and links into DIV.

-How fast/scalable is it?
Same as ASP.NET.

-Is it safe?
Same story as ASP.NET: clean your input, prevent SQL injection, …
It's built on top of ASP.NET.

-When will it ship?
No official dates yet.
Last preview version 5, next one will be beta and feature complete.

QUESTIONS?



REALDOLMEN

Make sure to check
http://blog.maartenballiauw.be/category/MVC.aspx

THANK YOU!

REALDOLMEN