ASP .NET MVC

Introduction

garth@ggilmour.com

Introducing ASP .NET MVC

- ASP .NET Web Controls have many benefits
  - They allow web applications to be built rapidly
  - They provide a very rich user experience
  - They can easily be bound to data sources
- But there are many disadvantages
  - As functionality increases so does the number of post-backs
  - The amount of View State data can grow alarmingly
  - The underlying HTTP infrastructure is not hidden
- The most serious problems are architectural
  - ASP .NET actively discourages MVC based best practises

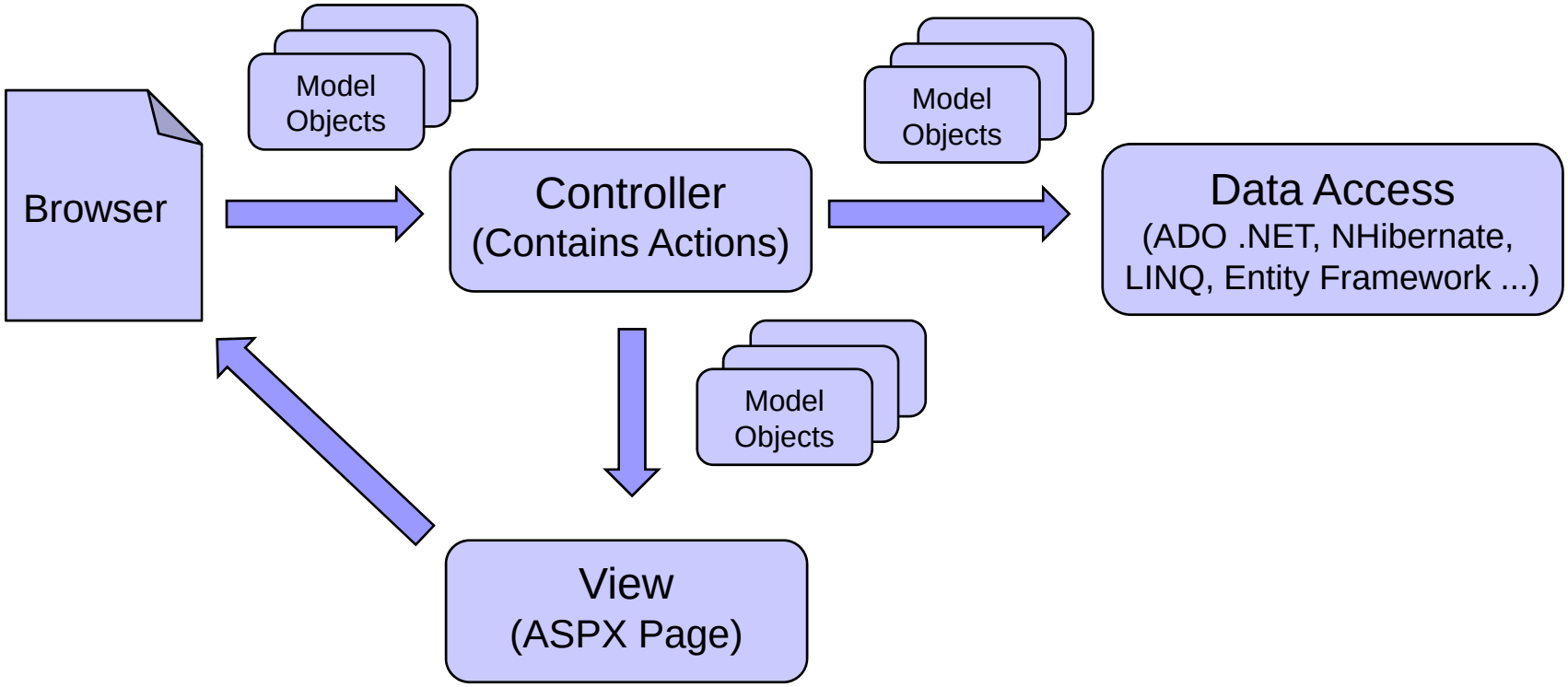
The MVP and MVC Architectures

- The current best practise for ASP .NET is MVP
  - In the Model/View/Presenter pattern a separate presenter object holds the business logic and is called from the code-behind
  - Because of the event-driven, post-back based nature of web controls it is hard to keep the presenter ‘pure’
- Modern web frameworks outside .NET use MVC strictly
  - In Struts 2, Spring @MVC, Grails, Ruby on Rails etc… business logic lives in controller classes which are ‘plain old objects’
  - Because these have no dependencies on the Web Framework they can be developed and tested independently of the GUI

The Architecture of ASP .NET MVC

- ASP .NET MVC is a new approach to .NET web apps
  - It enables the creation of MVC-centric applications
- The framework was developed within Microsoft
  - In response to the popularity of similar tools on other platforms
- Using MVC does not rule out using web controls
  - Both can exist in the same web application
  - Both are based on the same infrastructure
- How both approaches develop will depend on usage
  - Customer demand will determine which becomes dominant

The Architecture of ASP .NET MVC



Goals of ASP .NET MVC

- Enable proper unit and integration tests
  - Controllers can be tested without the browser
  - Data access layer can be ‘mocked out’
- Transparent naming conventions
  - URL’s are simple and can be bookmarked
- Take control over markup
  - No hidden tags are sent to the client
- Build on top of ASP .NET infrastructure
  - But make the post-back style controls optional
- Provide clear conventions and guidance
  - So new projects can be started rapidly

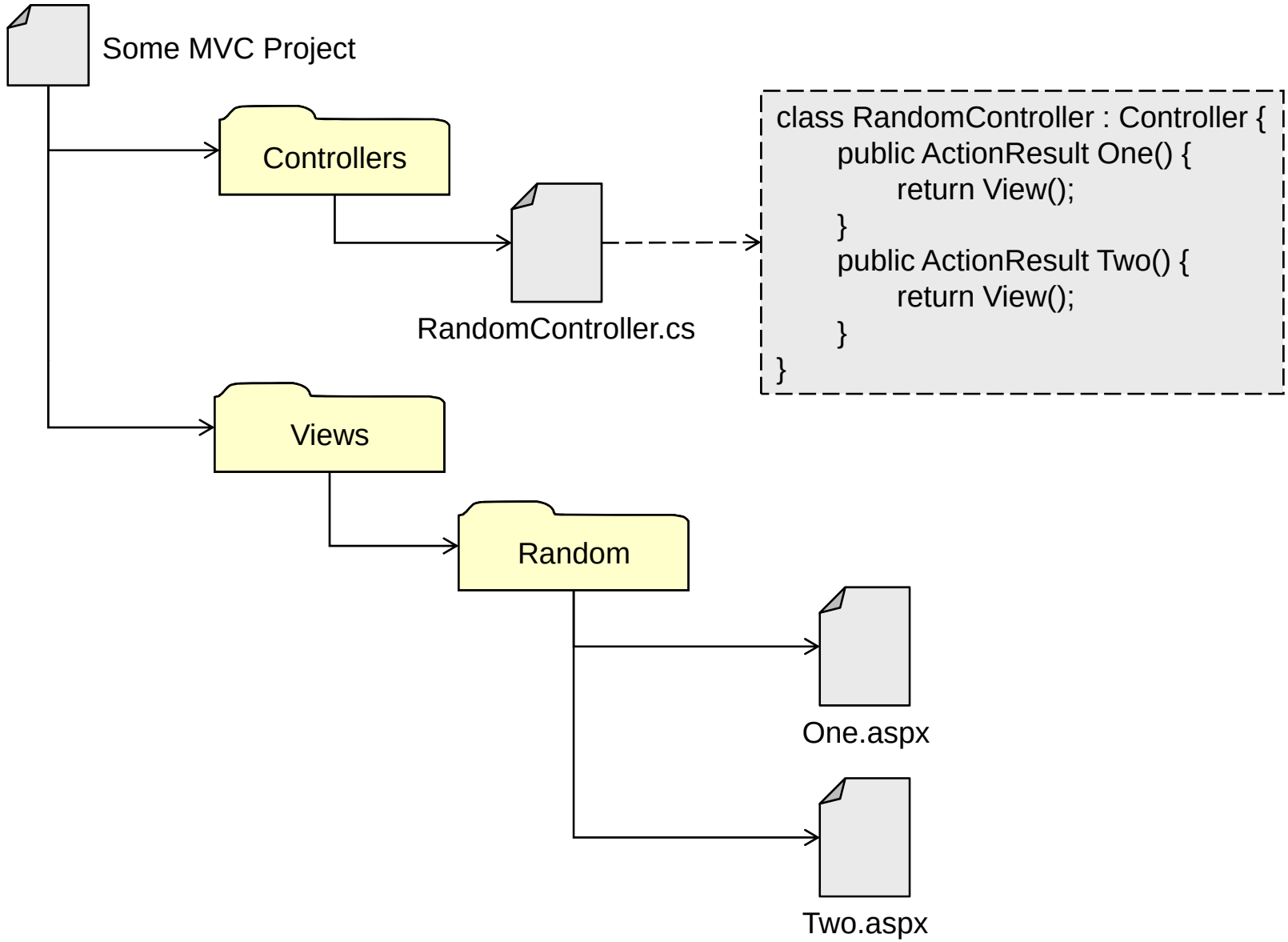
The Architecture of ASP .NET MVC

- When an MVC app starts up a routing table is built
  - This associates URL patterns with controller names
  - E.g. a request to the URL ‘/random/one’ could be associated with method ‘One’ in class ‘RandomController’
- Each request is passed to the associated controller
  - The controller method performs business logic and returns a name which can be mapped to a path to a server page
  - This then determines the next view the client sees
- The framework is highly customizable
  - The strategy design pattern is used to let you customize or replace individual parts of the framework as required

The Architecture of ASP .NET MVC

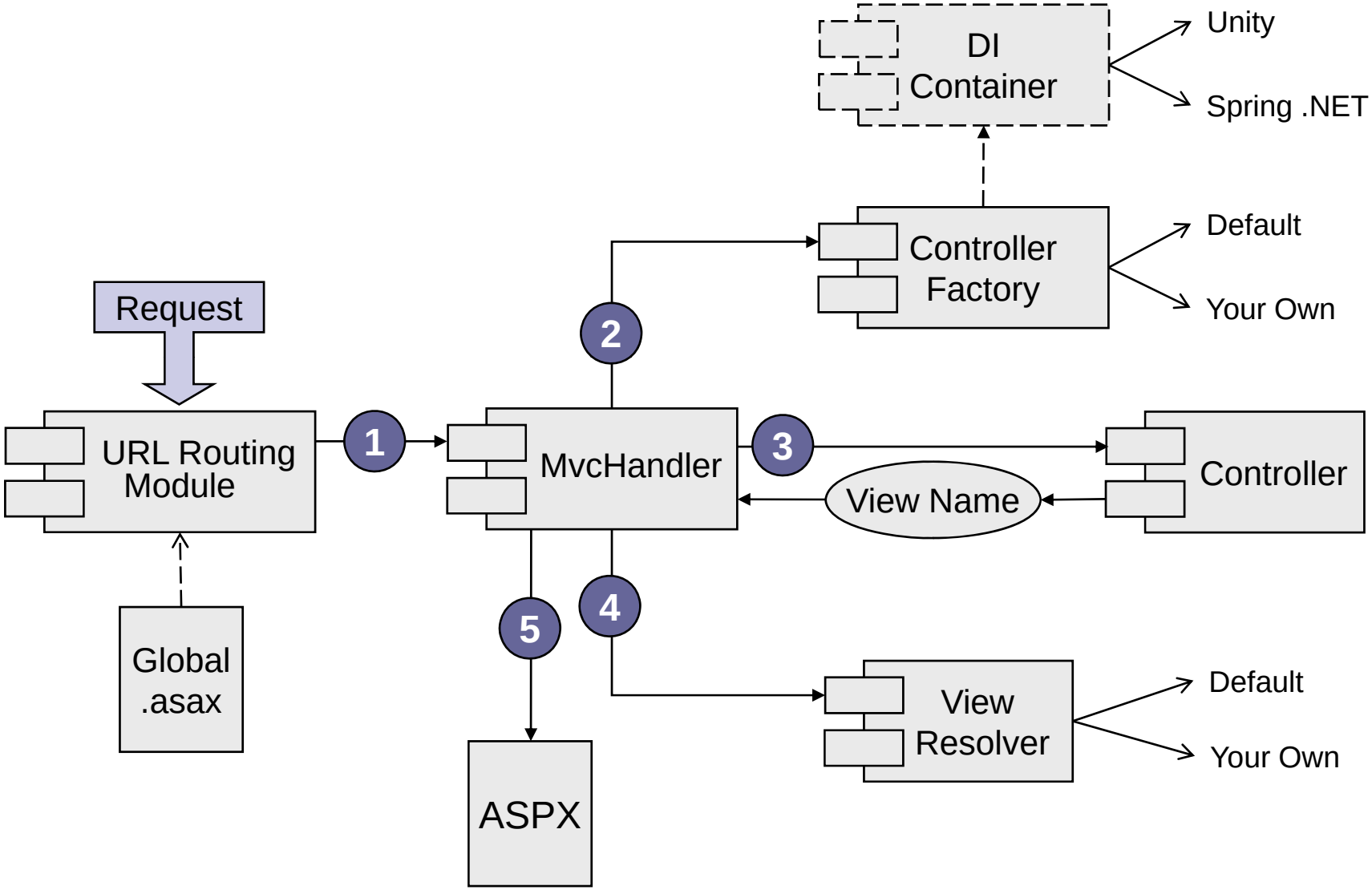
```
public class MvcApplication : System.Web.HttpApplication {
    public static void RegisterRoutes(RouteCollection routes) {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
        routes.MapRoute(
            "Default",                                          // Route name
            "{controller}/{action}/{id}",                      // URL with parameters
            new { controller = "Home", action = "Index", id = "" }  // Parameter defaults
        );
    }
    protected void Application_Start() {
        RegisterRoutes(RouteTable.Routes);
    }
}
```

```
<add name="UrlRoutingModule" type="System.Web.Routing.UrlRoutingModule,
        System.Web.Routing, Version=3.5.0.0, Culture=neutral,
        PublicKeyToken=31BF3856AD364E35" />
```



The Lifecycle of an MVC Request

1. The ‘UrlRoutingModule’ receives the request
   - It creates and runs the correct ‘MvcHandler’
2. The handler creates and runs a controller
   - Creation is via the current ‘ControllerFactory’
   - The entry point to the controller is ‘Execute’
     - A ‘ControllerContext’ is passed as a parameter
3. The controller runs the appropriate method
   - This is found using standard reflection
   - A list of parameters can be passed in
4. Control is passed to the current ‘ViewEngine’
   - Which (typically) builds and calls a server page



Using MVC with Other Frameworks

- The constructor factory extension point enables the use of a dependency injection framework
  - This builds trees of components at runtime
    - Based on declarations in an XML configuration file
  - Components have their dependencies ‘injected’
    - Making the code very flexible
- Thee are many options for database access
  - Standard ADO .NET using readers and data sets
  - LINQ to SQL to create anonymous types based on queries
  - Object relational mapping frameworks
    - Such as NHibernate or the ADO .NET Entity Framework

jQuery Support in ASP .NET MVC

- ASP .NET MVC bundles jQuery
  - A very popular framework for cross-browser JavaScript
  - It uses a feature called selectors to simplify finding parts of the DOM tree and changing their CSS
- Microsoft's stated plans are to:
  - Contribute to jQuery rather than forking it
  - Introduce intelli-sense support in Visual Studio
- jQuery can be used in many ways:
  - To decorate ‘pure’ XHTML with presentation
  - Through one of many plug-in libraries for rich clients
  - As an addition or replacement for ASP .NET AJAX

Examples of jQuery Selectors

- $(“#name”) or $(“.name”)
  - Selects elements by ID or class
- $(“p:even”) or $(“p:odd”)
  - Selects even or odd numbered paragraphs
- $(“table tr:nth-child(1)”)
  - Selects the first row in each table
- $(“table tr:nth-child(even)”)
  - Selects alternative rows of each table
- $(“li:last-child”)
  - Selects the last item in each list

MVC and Test Driven Development

- A key selling point of MVC is that it enables TDD
  - In TDD tests are written before the code in order to drive the design and keep it simple and user-centric
  - MVC controllers are very easy to unit test
- Any unit testing tool can be used
  - NUnit is the leading open source testing tool
  - Visual Studio ships with a built in testing tool
    - In VS 2005 this was only available in the team edition
    - In VS 2008 it is available in the professional edition as well
  - Typically a mock objects generator is also required
    - NMock is the most common mocking framework in .NET

Processing Data from the Request

- Action methods of the controller can take parameters
  - These are matched against parts of the URL first, then parameters in the form and finally the query string
    - This means that both “/controller/action/value” and “/controller/action?param=value” can be used at once
  - The first match is always the one used
- Methods can also take Value Objects
  - The framework matches parameters to properties
  - You can also supply your own binding objects
- A strength of MVC is comprehensible URL’s
  - E.g. ‘/products/electronic/tvs/abc123’

Passing Data Into The Server Page

- The simplest mechanism for passing data to the server page is via the ‘ViewData’ object
  - This is a dictionary that holds arbitrary objects
  - The dictionary is populated in the controller and automatically passed to the server page by the ‘MvcHandler’
- Information can be retrieved via a key or via generics
  - E.g. ‘ViewData[“order”]’ or ViewData.Get<Order>()
- A more strongly typed solution is possible
  - An arbitrary object is passed back from the controller
  - The server page inherits from ‘ViewPage<T>’
    - Where ‘T’ is set to the type returned from the controller

Mixing MVC with Normal ASP .NET

- There is no problem combining MVC and Web Forms
  - As everything is based on the same infrastructure
- An ASP .NET Web Application can contain:
  - MVC based controllers, actions and views
  - ASP .NET pages containing HTML and Web Controls
  - Windows Communication Foundation based Web Services
- Some changes may be required to routing tables
  - MVC can be configured to ignore requests to existing files
  - But you may wish to add explicit exceptions to prevent unnecessary calls which access the file system
    - You can do this via ‘routes.IgnoreRoute(“...”)’

Using AJAX in Web Applications

- MVC uses jQuery to implement AJAX
  - Using ‘Ajax.BeginForm()’ you can generate markup and scripts that support partial post-backs
    - It takes an ‘AjaxOptions’ parameter that holds the ids of elements to be updated and where updates should be placed
  - This can be detected via ‘Request.IsMvcAjaxRequest’
    - You then send changes via ‘return PartialView(...)’
- ASP .NET AJAX is not replaced
  - It remains a better option for:
    - Calling .NET based Web Services
    - Creating client-side only controls
  - Plus there is all the code in the control toolkit