Mastering ASP.NET MVC4 in Just One Day

Tiberiu Covaci
@tibor19

Who am I

Tiberiu ’Tibi’ Covaci

Software engineer, 20 years experience

MCT since 2004, teaching .NET

Senior Trainer & Mentor in Romania (Transylvania)

MVP for Windows Azure

Father & Geek

Twitter: @tibor19 / @gotober

Administration

We start 09:00

We end 16:00

Lunch between 12:00 and 13:00

We take 2 breaks, 15 minutes each, at 10:30 and 14:30

We end the day with Q&A

Ask your questions as they come

Artifacts at: http://bit.ly/18RgM9p

Agenda

Introduction to ASP.NET MVC 4

Models

Controllers

Views

Routing

Advanced Features

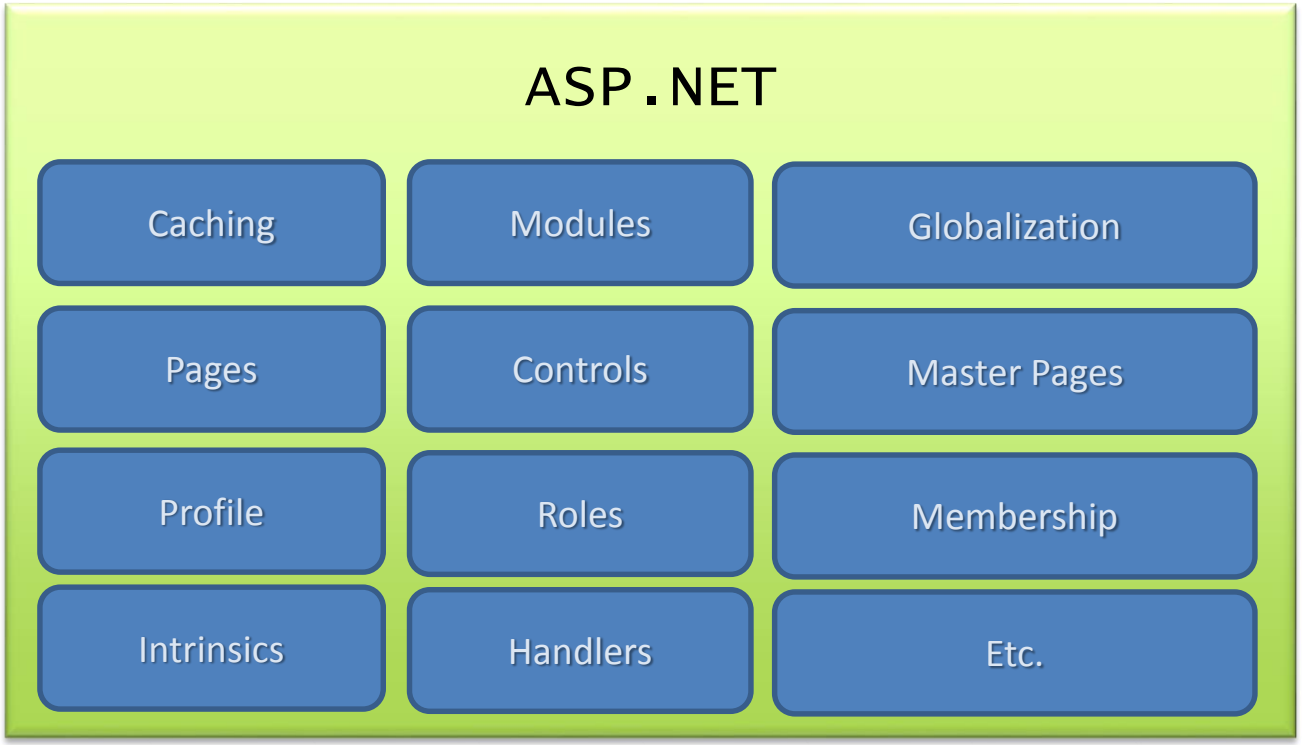
If Time Permits

- Mobile Web Applications
- Web API/SPA

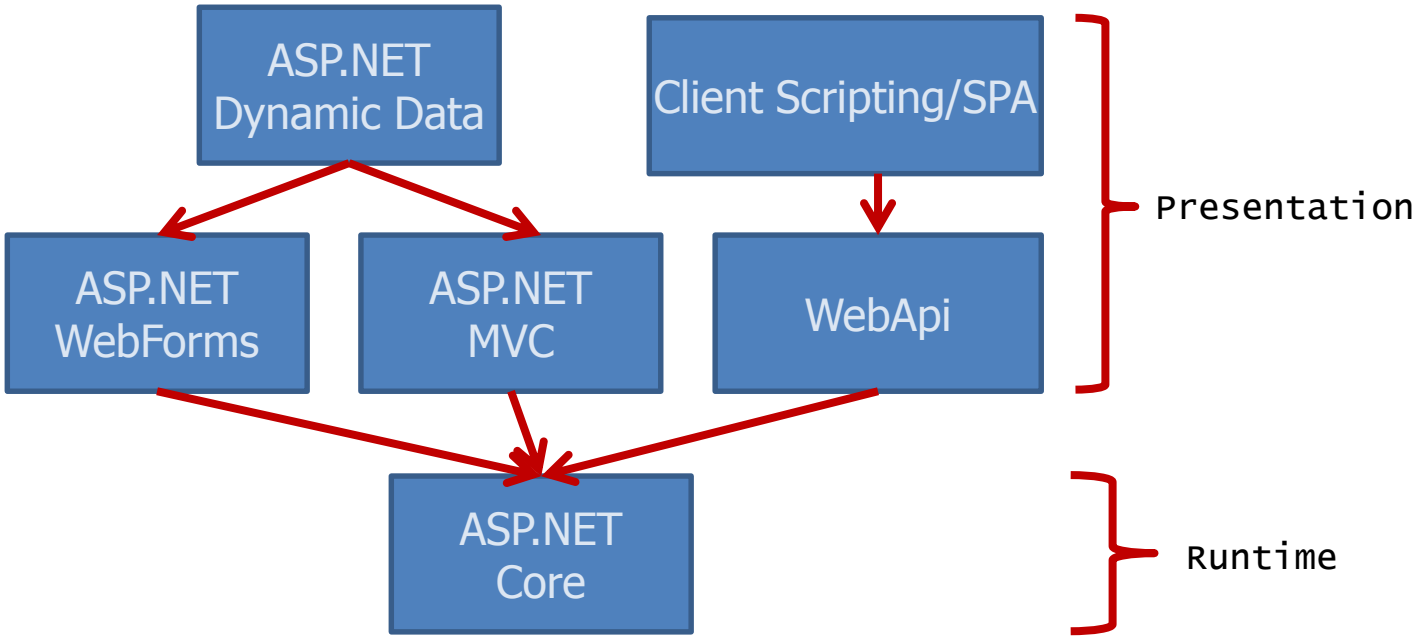
DEMO

Introduction to MVC

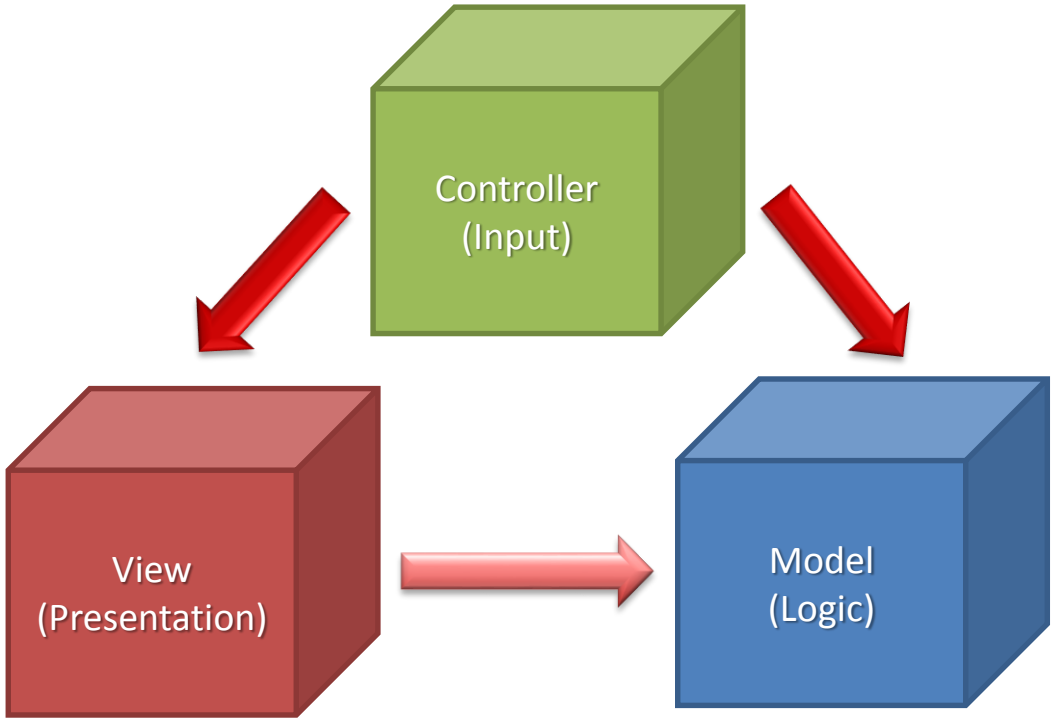
ASP.NET Before



ASP.NET Today



MVC=Model View Controller



Goals of ASP.NET MVC

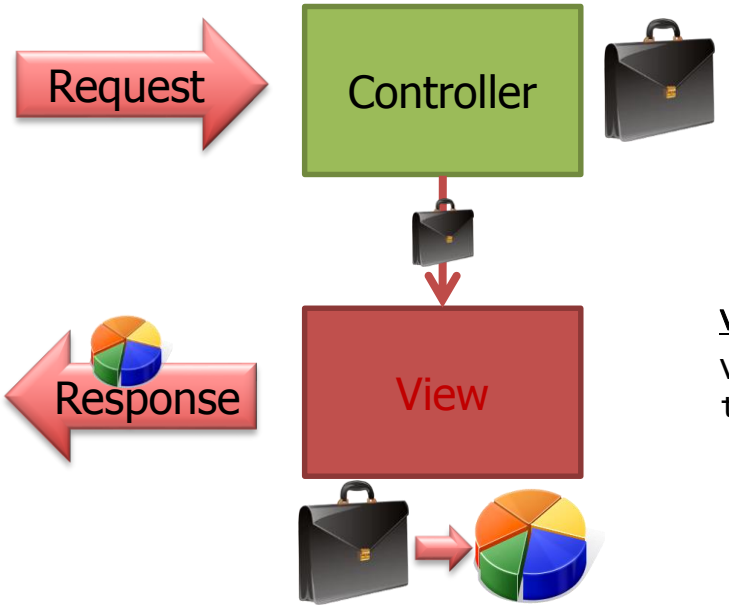
Frictionless Testability

Tight control over <markup>

Leverage the benefits of ASP.NET

Conventions and guidance

How does ASP.NET MVC works?



Controller

Handles input
(HTTP requests)

View

Visually represents
the model

How does ASP.NET MVC works?

Request -> UrlRoutingModule->Find a route

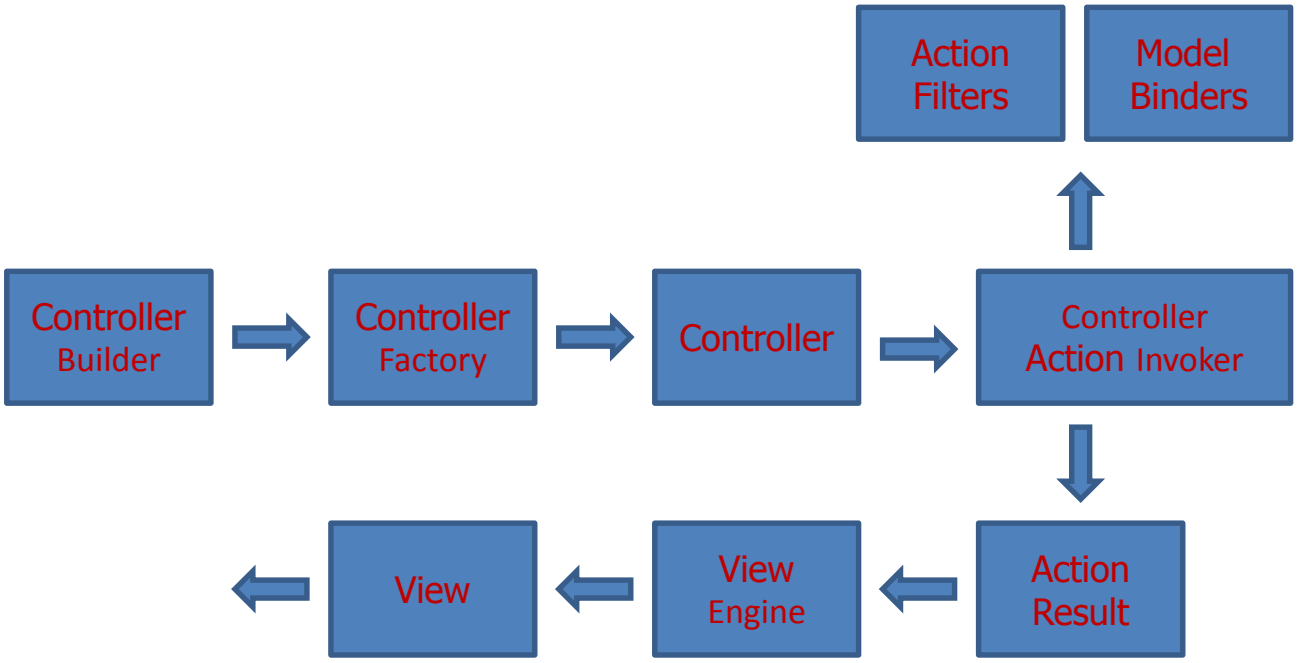
RouteBase(Route) -> IRouteHandler(MvcRouteHandler)

Handler->Controller->Execute()

Execute()->Action. Invoke()

```
Invoke(){
    var modelData = GetModelData();
    return View(modelData);
}
```

MVC Extensibility points



Demo

- File=>New =>MVC

Model

What is the model

Business data

Maps to a persistence storage like a database

Different styles of doing that

- CRUD
- DDD

CRUD

Create

Read

Update

Delete

DDD

Domain Driven Design

Capture business logic

Helps creating the Ubiquitous Language

Entities, Value objects, Aggregates

Bounded contexts

- Ubiquitous language
- Entities, Value objects, Aggregates

Patterns

- Repository pattern
  - Responsible for the interaction with the persistence layer
  - Wraps the basic CRUD methods
- Unit of Work
  - Groups several operations together to run atomic
- Service pattern
  - Encapsulates the complex logic of handling the domain objects

Demo

- Creating the Model

Controller

What are Controllers

Classes with the following properties:

- The class must be **public**
- The class must be **non-abstract**
- The class must be **non-nested**
- Implements **IController** or derive from a class that does

By default they **MUST** have the **Controller** suffix

By **convention** are placed in the **Controllers** folder

Spot the Controller

```
public class Category : Controller {…}

public abstract class CategoryContoller : Controller {…}

public class Foo
{
   public class CategoryController : Controller {…}
}

class BarController : Controller {…}

internal class BarController : Controller {…}

public class BazController : Controller {…}

public class BuzzController {…}
```

What are Actions

Each MVC request maps to an **Action** in a **Controller**

Action is a method that returns an **ActionResult**

The method must have the following properties:

- The method must be **public**.
- The method **cannot** be a **static** method.
- The method **cannot** be an **extension** method.
- The method **cannot** be a **constructor**, getter, or setter.
- The method **cannot** have **open generic** types.
- The method is **not** a method of the **base class**.
- The method **cannot** contain **ref or out** parameters.
- Is **not** decorated with the **NonActionAttribute**

Spot the Action

```
public void VoidAction(){…}
public ActionResult MyAction(){…}
protected ActionResult ProtectedAction(){…}
public ActionResult Execute(){…}
public static ActionResult StaticAction(){…}
public static ActionResult MyAction(this string data){…}
[NonAction]
public ActionResult NonAction(){…}
public string StringAction(){…}
public ActionResult TAction<T>(){…}
```

Action results

ActionResult – the abstract base class

ViewResult –HTML and markup

EmptyResult – No result

RedirectResult – Redirection to a new URL

JsonResult –JSON result for AJAX application

HttpStatusCodeResult – HTTP Result code

Action results continued

JavaScriptResult –JavaScript script

ContentResult – Text result

FileContentResult – Downloadable file (binary)

FilePathResult – Downloadable file (path)

FileStreamResult – Downloadable file (stream)

Results simplified

View – Returns a ViewResult

Redirect – Returns a RedirectResult

- RedirectToAction – Returns a RedirectToRouteResult
- RedirectToRoute – Returns a RedirectToRouteResult action result.

Json – Returns a JsonResult

JavaScript – Returns a JavaScriptResult

Content – Returns a ContentResult

File – Returns one of the FileResult types

HttpNotFound

Create Controller

Add->New class *Controller

Place it under Controllers folder

Add->New Controller

You can have CRUD action methods

Passing data to the view

We can use ViewBag or ViewData

We can use TempData

By setting the Model property of the ViewData

Demo

- Creating Controller

View

What is a view

As close to a page as it could get

Responsible for the user interface

Generated via ViewResult

Two kinds of view Out of the box

- Razor views (inherit from WebViewPage)
- ASPX WebForm views (inherit ViewPage)

Strongly typed views

WebForm pages Inherit from ViewPage<T>

Razor uses the @model <T> directive

- You get a new property called Model of type T so you can get strongly typed access to your data

Razor Syntax

Code Block	`@{`
	`  int x = 42;`
	`  string y = "Something";`
	`}`

Expression (Html Encoded)	`<span>@Model.Description</span>`

Expression (not encoded)	`<span>`
	`  @Html.Raw(@Model.Description)`
	`</span>`

Combining Text and markup	`@foreach(var item in items) {`
	`  <span>@item.Prop</span>`
	`}`

Razor Syntax

Mixing code and Plain text	`@if (@Model.Description != null) {`
	`  <text>Plain Text</text>`
	`}`

Mixing code and Plain text	`@if (@Model.Description != null)`
	`{`
	`  @:Description is @Model.Description`
	`}`

Comments	`@*`
	`This is a server side`
	`multiline comment`
	`*@`

HtmlHelper class

Encode()
ActionLink()
BeginForm()
CheckBox()
DropDownList()
EndForm()
DisplayFor()

Hidden()
ListBox()
Password()
RadioButton()
TextArea()
TextBox()
EditorFor()

Create your own

- Extension methods for the HtmlHelper class
- Use TagBuilder class to generate the html

Partial Views

Think UserControls in WebForms

Represent just a portion of your page

Very useful for reuse of markup

By convention in Razor partial view name starts with underscore.

Layout Views

Think Master page in WebForms

Set in the view via the Layout property of the view

By convention in Razor layout view name starts with underscore

Can be set for the whole application

- Create a view in the Views folder called _ViewStart.cshtml

Demo

- Adding Views

Posting Data

Attributes

[AcceptVerbs]

Specify HTTP Verb(s)

- Get || [HttpGet]
- Post || [HttpPost]
- Head || [HttpHead]
- Put || [HttpPut]
- Delete || [HttpDelete]

[ActionName]

Specify the name of the action

Getting the data

- UpdateModel/TryUpdateModel
  - Specify the receiver of data
  - You can specify a list of allowed properties
- Use Model binders
  - They use reflection to populate the parameters of the Action

Demo

- Posting Data

Routes

Why routes

/Recipes/Pizza is nicer than... /Recipe?ID=...

Part of ASP.NET (System.Web.Routing)

Introduced in .NET 3.5 (System.Web.Routing.dll)

Now is part of System.Web.dll (from .NET 4.0)

Default Route Table

Initialized in Application_Start()

Contains one route {controller}/{action}/{id}

Maps automatically to controller class

```
routes.MapRoute( "Default", // Route name
        "{controller}/{action}/{id}", // URL with
parameters
        new { controller = "Home", action = "Index", id =
"" }
        // Parameter defaults
);
```

Adding routes

Default routes are not always appropriate, so define your own.

```
routes.MapRoute( "Blog", // Route name
        "Archive/{entryDate}", // URL with parameters
        new { controller = "Archive", action = "Entry”}
        // Parameter defaults
);
```

Order is important!

Route constraint

What happens when you send wrong data format?

```
routes.MapRoute( "Product",
        "Product/{productId}",
        new {controller="Product", action="Details”}
        , new {productId = @"\d+" }
 );
...
public class ProductController : Controller {
    public ActionResult Details(int productId) {
        return View();
    }
}
```

Custom constraints

You need to create a class that implements IRouteConstraint

```
interface IRouteConstraint {
        bool Match( HttpContextBase httpContext,
            Route route,
            string parameterName,
            RouteValueDictionary values,
            RouteDirection routeDirection );
}
```

When applied to a route if Match returns false, only that route will not match.

Routes best practices

Make simple, clean URLs

Make hackable URLs

Allow URL Parameters to clash

Keep URLs short

Avoid exposing intern Ids

Consider adding unnecessary info ☺

Demo

- Creating Routes

Advanced Features

View Templates

Filters

Dependency Injection

Model Binders

Controller Factories

What are Filters?

Defined as attributes

A declarative way to add functionality to Actions

Three out of the box filters

- Authorize
- HandleError
- OutputCache

Which are the Filters?

- Classes that inherits from FilterAttribute
  - One property Order
- Implements one of the following interfaces
  - IAuthorizationFilter
  - IActionFilter
  - IResultFilter
  - IExceptionFilter

Mobile applications

Use CSS Media Queries

Create a New Mobile MVC 4 Application

The Mobile Templates

Working with Display Modes

- System.Web.WebPages.DisplayModeProvider

Techniques for Testing Mobile Applications

Web API

Intro to REST

Create a WebAPI Controller

Make the Controller RESTfull

Summary

Introduction to ASP.NET MVC 4

Models

Controllers

Views

Routing

Advanced Features

If Time Permits

- Mobile Web Applications
- Web API/SPA

Thank you!

Questions ?