C# Razor View Engine Cheatsheet for ASP.NET MVC

|  |  |  |  |
| --- | --- | --- | --- |
| What is it called | Example | Rendered output | Explanation |
| Inline Expression (HTML Encoded) | The year is @DateTime.Now.Year | The year is 2011 | Razor looks at the characters after the @ character and determines if it is code. If so it executes it. And finally HTML encodes any characters that need encoding. |
| Inline Expression (Raw HTML) | The year is @Html.Raw(DateTime.Now.Year.ToString()); | The year is 2011 | Not that Html.Raw only accepts a string. Bye default all @code is HTML encoded to be safe from XSS. Html.Raw can be used to override this behavior. |
| Explicit expression | <p>@("Number is " + 1)</p> | <p>Number is 1</p> | When you want code statement that might be confusing for simple inline code block. E.g. useful for generic function calls |
| Explicit multiline expressions | @{  int number = 1;  string message = "Number is " + number;  }  <p>@message</p> | <p>Number is 1</p> | @{ *code* } |
| Escaping @ character | <p>This is not a variable name : @@foo</p> | <p>This is not a variable name : @foo</p> | Use double @@ characters which get rendered as one @ character. |
|  | <p>Email addresses : basarat@basarat.com do not need to be escaped</p> | <p>Email addresses : basarat@basarat.com do not need to be escaped</p> |  |
| Nested HTML in loops and conditions | @foreach(var item in items) {  <span>@item.Prop</span>  } | <span>asdf</span>  <span>asdf</span> | As long as the nested content in loops starts with an html tag its simple. Otherwise use text tag or @: shown next |
|  | @foreach(var item in items) {  <text>if I may @item.Prop</text>  } | if I may asdf  if I may asdf | <text> tag is ignored in output |
|  | @foreach(var item in items) {  @:if I may @item.Prop  } | if I may asdf  if I may asdf | Also you can use @: instead of <text> to start a text block inside of nested content. |
|  | @if (DateTime.IsLeapYear(DateTime.Now.Year))  {  <span>Hurray its as leap year;</span>  }  else  {  <span>sad face</span>  } | <span>sad face</span> | Sample showing nesting html in if block. Similar to foreach / for loops. |
| Server side comments | @\*  This is a server side  multiline comment  \*@ |  | Nothing is rendered on the html side for this … of course. |
| Templated razor delegates | @{  Func<dynamic, object> b = @<strong>@item</strong>;  }  @b("Bold this") | <strong>Bold this</strong> | <http://haacked.com/archive/2011/02/27/templated-razor-delegates.aspx> |

# Useful functions for ASP.NET Development:

## HttpUtility.HtmlEncode (string s)

Great for sanitizing any input removing any html and replacing with % encodings.

Note that such potentially harmful inputs are already filter by default e.g. http://localhost:1249/Store/Browse?genre=whate<er gives the error

“A potentially dangerous Request.QueryString value was detected from the client (genre="whate<er")”

Where genre is a parameter expected by a controller.

# Useful Html Helpers

Accessed from the Html property of the ViewBase class.

## Html.ActionLink

Talks to the Routing engine to figure out the actual link:

Html.ActionLink("Text","ActionName","ControllerName");

OR

Html.ActionLink("Text","ActionName",new {someParameter = "value"});

## Html.LabelFor Html.ValidationMessageFor Html.EditorFor

Html.EditorFor(model=>model.Property)

## Html.BeginForm

## Html.EndForm

# Useful functions for controllers

## TryUpdateModel

Updates the specified model object using the current data posted/got for this action.

# Creating Custom Html Helpers

Use an extension method of the form:

public static MvcHtmlString Whatever(this HtmlHelper helper,sometype someArgument)

Create the MvcHemlString from MvcHtmlString.Create(string). Reason for returning this class instead of a string is that otherwise @ would html encode the string that you return from your html helper.

# TagHelper

Useful for building properly formatted html tags. Use in your own Custom Html Helpers or Views.

var builder = new TagBuilder("img");

builder.MergeAttribute("src", src);

var string = builder.ToString(TagRenderMode.SelfClosing);

# Avoiding too many usings in your Views

You can add the namespaces to all view by adding it to the <namespaces> collection inside web.config.

# Conventions in ASP.NET MVC:

## Folders and files

/Scripts folder

/Content folder

/Controllers folder

/Views folder

/Views/Shared folder

/Views/\_ViewStart.cshtml file

### Note about \_ViewStart.cshtml

In the default project template simply configured to set the Layout to /Views/Shared/\_Layout.cshtml . So when you create a view the title is passed to \_Layout in a Viewbag.Title i.e.

@{

ViewBag.Title = "Index";

}

And when you request that you will not use a master page the \_ViewStart.cshtml setting is overridden as such (+ a complete html page markup is added to your generated view).

@{

Layout = null;

}

Also \_ViewStart.cshtml is hierarchical. You can have a different \_ViewStart for each controller by making the same in /View/ControllerName/ folder.

# Notes about Routing

You can access route data from your Views using RouteData property.

# ActionResults

Member functions on controller

* return File(Server.MapPath(“~/Content/Site.css”))
* return RedirectToRoute(“RouteName”,new {paramName = “value”})
* return RedirectToAction(“Controller”,”Action”,new {paramName = “value”})
* return Json(someObject,JsonRequestBehaviour.AllowGet); //any type of object even anonymous type. It will be converted to Json behind the scenes
* return View(….)

# ActionSelectors

Specified as attributes on Actions. Some common ones:

* [ActionName(“NewActionName”)]
* [AcceptVerbs(HttpVerbs.Post)] //same as [HttpPost]

# ActionFilters

Attributes on Actions OR Controllers for Cross cutting concerns. E.g

* Authorize

You can also have global filters. These are added to the GlobalFiltersCollection in RegisterGlobalFilters function which is called on Application Startup. This is how the global error handler is added. It uses the view present in /Views/Shared/Error.cshtml

# Creating Custom ActionFilters

Derive a class from ActionFilterAttribute

Override any or all of:

* OnActionExecuted
* OnActionExecuting
* OnResultExecuted
* OnResultExecuting

Add it to the GlobalFilter.Filters collection OR as an attribute to the controller class OR action method

# Custom Errors

The custom error handler is not run by default for localhost. To change that, add the following configuration to <system.web>

<customErrors mode=”On”>

The default value for mode is RemoteOnly.

# Creating sections in Views

Render them in the layout with the function:

@RenderSection("SectionName”,Boolean required)

And create them in the template view with :

@section SectionName {

}

# Useful Visual Studio Shortcuts

## Right click controllers folder and select add controller

## Right click anywhere on or inside the body of a controller function and select Add View