**Views**

ASP.NET MVC application, incoming browser requests are mapped to controller actions. A controller action might return a view. However, a controller action might perform some other type of action such as redirecting you to another controller action.

Action returns the View in return

public ActionResult Index()

{

return View();

}

For eg. The above code will return a view which will be searched in \Views\Home\Index.cshtml

If you prefer, you can be explicit about the view. The following line of code returns a view named Fred :

View( Fred );

When this line of code is executed, a view is returned from the following path:

\Views\Home\Fred.cshtml

**Using HTML Helpers to Generate View Content**

An HTML Helper, typically, is a method that generates a string. You can use HTML Helpers to generate standard HTML elements such as textboxes, links, dropdown lists, and list boxes.

For example, Following are the three HTML Helpers -- the BeginForm(), the TextBox() and Password() helpers -- to generate a Login form

<% using (Html.BeginForm())

{ %>

<label for="UserName">User Name:</label>

<br />

<%= Html.TextBox("UserName") %>

<br /><br />

<label for="Password">Password:</label>

<br />

<%= Html.Password("Password") %>

<br /><br />

<input type="submit" value="Log in" />

<% } %>

Following is the HTML in which your code will be converted

<form method="post" action="/Home/Login">

<label for="userName">User Name:</label>

<br />

<input name="userName" />

<br /><br />

<label for="password">Password:</label>

<br />

<input name="password" type="password" />

<br /><br />

<input type="submit" value="Log In" />

</form>

**Using View Data to Pass Data to a View**

All data passed from a controller to a view must be sent using this package.

public ActionResult Index()

{

ViewData["message"] = "Hello World!";

return View();

}

The controller ViewData property represents a collection of name and value pairs.

In Index.cshtml

<div>

<%= Html.Encode(ViewData["message"]) %>

</div>

The Html.Encode() HTML Helper encodes special characters such as < and > into characters that are safe to display in a web page

**Creating Custom HTML Helpers**

Creating HTML Helpers using Extension Methods

Extension methods enable you to add new methods to an existing class.

 public static class LabelExtensions

     {

          public static string Label(this HtmlHelper helper, string target, string text)

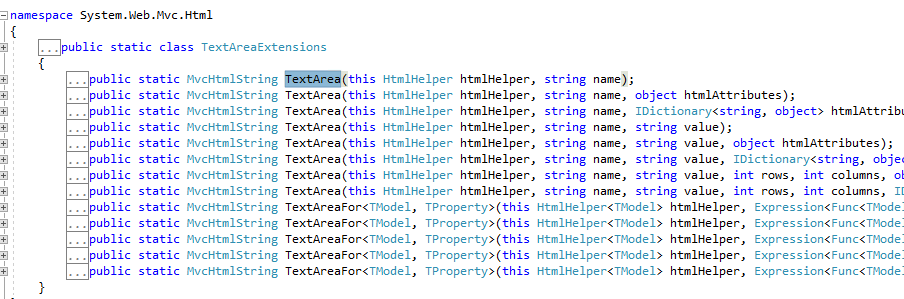
          {

               return String.Format("<label for='{0}'>{1}</label>", target, text);

          }

     }

Existing Helper classes are also created as extension methods as shown in the following figure



To use the Html Helper created above, Following code will be written in view

@Html.Raw(Html.Label("Welcome", "Welcome to the brand new helper", 1))

The ASP.NET MVC framework includes a useful utility class named the TagBuilder class that you can use when building HTML helpers. The TagBuilder class, as the name of the class suggests, enables you to easily build HTML tags.

## Creating an Image HTML Helper (Using Tag builder)

public static string Image(this HtmlHelper helper, string id, string url, string altText, object htmlAttr, int height, int width)

{

var builder = new TagBuilder("img");

builder.GenerateId(id);

builder.MergeAttribute("src", url);

builder.MergeAttribute("alt", altText);

builder.MergeAttribute("height", height.ToString());

builder.MergeAttribute("width", width.ToString());

builder.MergeAttributes(new RouteValueDictionary(htmlAttr));

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

return s;

## }

## In index.cshtml Following is the way to call helper class created via Tag Helper

## @Html.Raw(Html.Image("testHelperImg", "/Image/Test.png", "Test Image", 200, 200))

## Passing Data to View Master Pages

### **The Simple Solution**

## Let's start with the simplest solution to passing view data from a controller to a view master page. The simplest solution is to pass the view data for the master page in each and every controller action. As below example

  public ActionResult Index()

          {

               ViewData["categories"] = from c in \_dataContext.MovieCategories

                         select c;

               ViewData["movies"] = from m in \_dataContext.Movies

                         select m;

               return View();

}

## If this view data is not passed from every action, It will not be available for Master page therefore View Data should be passed from every action.

### **The Good Solution**

## Instead of adding the movie categories for the master page in each and every controller action, we add the movie categories to the view data only once. All view data used by the view master page is added in an Application controller.

public abstract class ApplicationController : Controller

{

MoviesEntities entities = new MoviesEntities();

// GET: Application

public ApplicationController()

{

ViewData["User"] = entities.Users.Where(c => c.FirstName.ToUpper() == "Megha").Select(p => p.Id).SingleOrDefault();

}

## }

## Now the controllers should be inherited from ApplicationController.