ASP.NET CORE

- Razor syntax

Server side code

@ ακολουθούμενο από το όνομα της μεταβλητής ή το function

- Html.Raw() function: output html been interpreted by the browser

π.χ. ① @{ var helloWorld = "Hello, world!"; }

ή

② <p>@helloWorld</p>

<p>@Html.Raw(helloWorld)</p>

Αποτέλεσμα

① Hello, world!

② Hello, world!

- Explicit expressions

@(. . . .)

- Multi-statement Razor blocks

@{ : <text> . . . . ή @plain text . . . ή δήλωση variable
   : </text> string nam = "kostas";
}

②

- Razor comments

```
@*
  .
  .
  .
*@
```

- Templated delegates

e.x. Func<dynamic, object> movieTemplate = @ @item.Title@<di

klion as Func movieTemplate

```
@foreach(var movie in movies)
 {
   @movieTemplate(movie)
 }
```

! methods of a controller is referred as actions

- Routing connects URL's to actions on Controllers.

- NonAction attribute to declare a public action not to be accessible by URL's

Exaple

```
[Http
[HttpGet]
public IActionResult Edit() { return View(); }
```

(3)

Methods for generating an Action result:

- Content() returns the specified string as plain text to the client
- View() returns a View to the client
- PartialView() returns a PartialView to the client
- File() returns the content of a specified file to the client
- Json() returns a JSON response to the client
- Redirect() & RedirectPermanent() returns a redirect response, redirecting the user to another URL
- StatusCode() returns a custom status code to the client

Views

Χρησιμοποιούμε την ειδική λογική για τις ονομασίες:

▾ Controllers
  ProductController.cs (Detail action inside the controller)

▾ Views
  ▾ Product
    Details.cshtml

- Αν δεν βρίσκει το View δηλώνουμε το όνομα του
  π.χ. return View("Details");

(4)

Passing data into Views

(1) Using a (View) Model

Exaple

- Model

```
public class Product
{
    public string Title {get; set;}
    public ~~string~~ double Price {get; set;}
}
```

- Controller

```
public IActionResult Details(int id)
{
    Product product = new Product()
    {
        Title = "Toilet Paper",
        Price = 1.99
    };
    return View(product);
}
```

(5)

- View

```
@model HelloMVCWorld.Models.Product

<h1>@Model.Title</h1>
Price: @Model.Price
```

Using the ViewData / View Bag containers

- Controller

```
public IAction Result Details(int id)
{
    ViewBag.ProductTitle = "Toilet Paper";
    ViewBag.ProductPrice = 1.99;
    return View();          or    ViewData["ProductTitle"] =
                                      "Toilet Paper";
}
```

- View

```
<h1>@ViewBag.ProductTitle</h1>      or  @ViewData
Price: @ViewBag.ProductPrice
```

⑥

Partial View

- View
  - Shared
    - _Greeting.cshtml

- Using _Greeting to other View

```
@await Html.PartialAsync("_Greeting")
```

⑦

- Layouts

- _Layout.cshtml

```
.
.
.
<body>
  @RenderBody()
</body>
```

Που θέλουμε να εμφανίζεται το Layout
π.χ. footer ή header

Σε ένα άλλο View

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

! _ViewStart is a default Layout placed inside View

For multiple ViewStart we can define it inside our sub folde

(3)

Sections

Προσθέτουμε στην RenderSection("Footer") στο Layout

Από αλλά, Views ορίζουμε το section

About.cshtml

```
@section Footer {
    Hello from About page:
}
```

Products.cshtml

```
@section Footer {
    .
    .
    .
}
```

! @RenderSection("Footer", required: false)
για να την χρειάζεται η αρχικοποίηση το Section από κάθε page

: @if (!: IsSectionDefined("Footer"))
{ Generic footer message :
}

Αν θέλω μια σελίδα δεν να κάνει define το section θα εμφανίζεται αυτό

- View Imports

- @using
- @ inject
- @ model
- @ inherits
- @ addTaghelper
- @ removeTagHelper
- @tagHelperPrefix

! Το _ViewImports έχει scope όλα τα Views
εκτός το _ViewStart μόνο στο file που είναι

Routing

Startup.cs

Configure()

app.UseMvc(routes => routes.MapRoute("ProductList",
"Products/List/", new{ controller = "Products", action = "List"}));

! define the default controller and method

routes.MapRoute("Default", "{controller=Home}/{action=Index}");

- Inside the controller

```
[Route("/products")]
public class ProductController : Controller
    .
    .
    .
```

(11)

```
[Route("/products")]
Controller

[Route("{id}")]
method  Details(int id)
```

Routing

π.χ.

```
[Route("blog/{entryID}/{slug}")]
```

calling it as

/blog/153/testing/

βάζουμε ? στο {slug} το κάνουμε optional

Routing constraints

- Data type

```
[Route("blog/{entryId:int}/{slug}")]
```

  - int
  - bool
  - datetime
  - double
  - float
  - decimal
  - guid
  - long

- Length / range

```
[Route("blog/{entryId:min(1)}/{slug}")]
```

  - min()
  - max()
  - range(,)

```
slug:minlength(3)
```

Regex

```
[Route(@"blog/{slug:regex(^[[0-9]]{{1,7}}\-[[a-z0-9\-]]
                                {{3,50}}$)}")]
```

View Model

access data of multiple Models from the same view

n.x.

```
public class EditItemsViewModel
{
    public Model1Type Model1 {get; set;}
    public Model2Type Model2 {get; set;}
}
```

Model Binding

```
public class WebUser
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
}
```

```
[HttpGet]
public IActionResult SimpleBinding()
{
    return View(new WebUser() { FirstName = "John", LastName = "Doe" });
}
```

```
@using (var form = Html.BeginForm())
{
    <div>
    @Html.LabelFor(m => m.FirstName)
    @Html.TextBoxFor(m => m.FirstName)
    </div>

    <div>
    @Html.LabelFor(m => m.LastName)
    @Html.TextBoxFor(m => m.LastName)
    </div>

    <input type="submit" value="Submit" />
}
```

```
[HttpPost]
public IActionResult SimpleBinding(WebUser webUser)
{
    // TODO: Update in DB
    return Content($"User {webUser.FirstName} updated!");
```

Data Annotations

```
public class WebUser
{
    [Display(Name="First Name")]
    public string FirstName {get; set;}
```

Model Validation

```
public class webUser
{
    [Required] ή [Required(ErrorMessage = "....")]
    [StringLength(25)]  ή [StringLength(50, MinimumLength = 3)] ή
    public string FirstName {get; set;}                    [EmailAddress]
```

@Html.ValidationSummary()

Εμφανίζει όλα τα μηνύματα σφαλμάτων

(16)

Types of Model Validation

- [Required]
  ↳ (AllowEmptyString = true)

- [StringLength]
  ↳ (50, MinimumLength = 3)

- [Range]
  ↳ (1, 100)

- [Compare]

  n.x. [Compare("MailAddressRepeated")]
  public string MailAddress { get; set; }
  public string MailAddressRepeated { get; set; }

- [CreditCard]
- [EmailAddress]
- [Phone]
- [Url]

- [RegularExpression]
  n.x. 2x-1234-5678
  ↳ ("[a-z]{2}-[0-9]{4}-[0-9]{4}")]

17

Custom Model Validation

Μέσα σε ν ValidationAttribute κάνουμε override τη IsValid()

π.χ. βλέπε Custom Model Validation

Client-side Model Validation

. Include some libraries

cloudflare.com/ajax/libs/jquery . . .

. . . .

. . . .

Manual Model Validation

βλέπε tutorial

Tag-Helpers

n.x. <input class="form-control" asp-for="Title" />

add TagHelper in _ViewImports.cshtml

@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers

Form Tag Helper

<form method="post" asp-controller="Blog" asp-action="UpdateEntry"></form>

- Route attribute

<form method="post" asp-route="UpdateBlEntry"></form>

Controller

[Route("Blog/Update", Nam = "UpdateBlEntry")]

... IAction ...

Label Tag Helper

old Html

<label for="FirstName">First name:</label>

tag helper

<label asp-for="FirstName"></label>

Χρησιμοποιώντας την Display δίνουμε στο label το όνομα που θέλουμε και όχι του property

[Display(Name = "First name:")]

Input Tag Helper

<input asp-for="Title" />

Textarea Tag Helper

<textarea asp-for="Description"></textarea>

Select Tag Helper

```
public class EditUserViewModel
{
  public WebUser User {get; set;}
  public List<string> Countries {get; set;}
}
```

```
public class WebUser
{ public string FirstName {get; set;}
        . LastName
        . Country
}
```

```
public IActionResult EditUser()
{ EditUserViewModel viewModel = new EditUserViewModel();
  viewModel.User = new WebUser()
  { FirstName = "John",
    LastName = "Doe",
    Country = "USA"
  };
  return View(viewModel);
}
```

! Περισσότερα στο tutorial

```
<form asp-action="UpdateUser" method="post">
   <input asp-for="User.FirstName" placeholder="First name" />
        . . User.LastName . .
   <select asp-items="@(new SelectList(Model.Countries))" asp-for="User.Countr
</form>  <input . . .
```

Anchor Tag Helper

```
<a asp-controller="Movie" asp-action="List">Movie list</a>
```

⇓ Θα δημιουργηθεί

```
<a href="/Movie/List">Movie list </a>
```

Script Tag Helper

Χρησιμοποιείται σε περίπτωση όπου ο server από που έχουμε πάρει τα css γίνει fail, να μπορέσουμε τοπικά αρχεία

```
<script
   src="https: ..."
   asp-fallback-src="~/Content/..."
</script>
```

Link Tag Helper

```
<link rel="stylesheet"
href="  ....  "
asp-fallback-href="..." />
```

Static Files

- Χρειάζονται να ενεργοποιηθούν

```
public void Configure(IApplicationBuilder...)
{ app.UseStaticFiles();
  ...
}
```

- Project folder → Add → New folder (όνομα = wwwroot)

- To give access to a ~~wwwroot folder~~ wwwroot folder

```
using Microsoft.Extensions.FileProviders;
using System.IO;

Configure()... {
  UseDirectoryBrowser(new DirectoryBrowserOptions
  { FileProvider = new PhysicalFileProvider
    ( Path.Combine(Directory.GetCurrentDirectory(), "wwwroot",
                                                    "Images")
    ),
    RequestPath = "/Images"
  });
  ...
```

Exception Handling

- App not in development mode

```
Configure(...) {
  if(env.IsDevelopment())
  {
    app.UseDeveloperExceptionPag();
  } else
  {
    app.UseExceptionHandler("/Error");
  }
```

```
public class ErrorController : Controller
{
  public IActionResult Index()
  {
    return Content("We are sorry....");
  }
```

```
var exceptionHandlerPathFeature = HttpContent.Features.Get<IExceptionHandlerPathFeature>()
if((exceptionHandlerPathFeature != null) && (exceptionHandlerPathFeature.Error != null))
```

Handling errors

Inside Configure() {

app.UseStatusCodePages(); ⟹ outputs Status code: 404; Not found

. . .

}

instead

app.UseStatusCodePages("text/html", "We're <b> really sorry . . . . ");

instead

```
app.UseStatusCodePages(async context =>
{
 context.HttpContent.Response.ContentType = "text/html";
 if(context.HttpContext.Response.StatusCode == 404)
 {
    // Log this erro . . .
 }
 await context.HttpContext.Response.WriteAsync(
    "we're really sorry . . ." $|| + context.HttpContext.Response.
                                    StatusCode);
```

UseStatusCodePagesWithRedirects() για καλύτερη error page

```
Use...Redirects($"/Error/Http?statusCode={0}");

ErrorController : Controller
{
   public IActionResult Http(int statusCode)
   {
     if(statusCode == 404)
     {
        ...
     }
     if (statusCode == 404)
        return View("Error404");
     return View();
```

Configuration settings

Μέσα στο appsettings.json

Accessing options from the controller

```
using Microsoft.Extensions.Configuration;

public class HomeController : Controller
{
  prirate readonly IConfiguration _configuration;
  public HomeController(IConfiguration configuration)
  {
    _configuration = configuration;
  }
```

```
public IActionResult Index()
{
  return Content("Welcom to" + _configuration.GetValue
                                <string>("WebsiteTitle"));
```

Accessing from startup class

```
...
{
 priv readonly IConfiguration _configuration;
 public Startup(IConfiguration configuration)
 {
  _configuration = configuration;
 }
```

Accessing from Views

```
@using Microsoft.Extensions.Configuration
@inject IConfiguration Configuration
<h1> Index </h1>
welcome to @(Configuration.GetValue<string>("WebsiteTitle")
```

HttpContext structure

- Request
- Response
- Session
- User

Mica ecov Controller

```
public IActionResult QueryTest()
{ string name = "John Doe";
  if(!String.IsNullOrEmpty(HttpContext.Request.Query["name"]))
     name = HttpContext.Request.Query["name"];
  return Content("Name from query string: " + name);
}
}
```

Accessing Form fields from controller

```
public IActionResult FormsTestPost()
{
  return Content("Hello, " + HttpContext.Request.Form["username"]
}
```

Cookies

Send cookie to browser

```
HttpContext.Response.Cookies.Append("user_id", "1");
```

Get cookie from browser

```
HttpContext.Request.Cookies["user_id"];
```

CookieOptions

```
using Microsoft.AspNetCore.Http;
CookieOptions cookieOptions = new CookieOptions();
HttpContext.Response.Cookies.Append("first_request", DateTime.Now.
                                    ToString(),
                                    cookieOptions);
```

- `cookieOptions.Expires = new DateTimeOffset(DateTime.Now.AddDays(7));`
- `cookieOptions.Domain = ".mywebsite.com";`
- `cookieOptions.Path = "/users/";` : cookie for specific pages

Session (short term data ~ 20 minutes)

Tools → NuGet Package Manager → Package Manager Console

Install-Package Microsoft.AspNetCore.Session

```
ConfigureServices(IServiceCollection services)
{   services.AddSession();
    services.AddMvc();
}

 services.AddSession(options =>
  {
   options.IdleTimeout = TimeSpan.FromMinutes(30);
  });

Configure(...)
{ if (env.IsDevelopment())
  {
   app.UseDeveloperExceptionPage();
  }
  app.useSession();
  app.useMvcWithDefaultRoute();
}
```

```
SessionsController : Controller
{
public IActionResult Index()
{
  string name = HttpContext.Session.GetString("Name");
  return View(model: name);
}
public IActionResult SaveName(string name)
{
  HttpContext.Session.SetString("Name", name);
  return RedirectToAction("Index");
}
```

Caching

```
[ResponseCache(Duration = 120)]
public IActionResult Index()
{ return View();
}
```

← That can be ignored by the browser (if the user refresh the page)

```
ConfigureServices(...)
{
  services.AddResponseCaching();
}

Configure(...)
{
  app.UseResponseCaching();
}
```

Ένα controller

```
[ResponseCache(Duration=120, VaryByQueryKeys = new[]{"id"})]
                              ή
                              VaryByHeader = "User-Agent"
```

Output Cache

- WebEssentials.AspNetCore.OutputCaching NuGet package

```
ConfigureServices(...)
{
  services.AddOutputCaching();
}
Configure(...)
{
app.UseOutputCaching();
}
```

```
[OutputCache(Duration = 120)]
public IActionResult Index()
```

webpage cached on the webserver

. . . .

Output Options

- Vary By Param
- Vary By Header
-

In memory caching with IMemoryCache

```
using Microsoft.Extensions.Caching
namespace MemoryCacheSample.Controllers
{
  public class HomeController : Controller
  {
    private IMemoryCache _memoryCache;
    public HomeController(IMemoryCache memoryCache)
    {
      this._memoryCache = memoryCache;
    }
```

more in tutorial

Databases

View → SQL Server... → Create the new Database

Data mapper

Tools → NuGet... → Install-Package Dapper (only GET support)

→ Install-Package Dapper.Contrib (everything)