

***Entity Framework***

**Lab Guides**

|  |  |
| --- | --- |
| Document Code | 25e-BM/HR/HDCV/FSOFT |
| Version | 1.1 |
| Effective Date | 20/11/2012 |

**Hanoi, 07/2019**

RECORD OF CHANGES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Effective Date | Change Description | Reason | Reviewer | Approver |
|  | 01/Oct/2018 | TuTB create new | Draft |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Contents

[Lab: Create custom helper to display friendly date and time value 4](#_Toc13575470)

[Objectives: 4](#_Toc13575471)

[Prerequisites: 4](#_Toc13575472)

[Problem Description: 4](#_Toc13575473)

[Guidelines: 4](#_Toc13575474)

[Step 1: Open project that created in NWEB.S.L001 4](#_Toc13575475)

[Step 2: Add new folder named Extensions then add new class named DateTimeExtension 4](#_Toc13575476)

[Step 3: Add method named ToFiendlyString as an extension method 4](#_Toc13575477)

[Step 4: Create method to get friendly string 4](#_Toc13575478)

[Step 5: Update method to return HtmlString 7](#_Toc13575479)

[Step 6: Retrieve message in view Index.cshtml 7](#_Toc13575480)

[Step 7: Check result 7](#_Toc13575481)

|  |  |
| --- | --- |
|  | **CODE: NWEB.L.L008**  **TYPE: MEDIUM**  **LOC: 300**  **DURATION: 60 MINUTES** |

# Lab: Create custom helper to display friendly date and time value

Objectives:

* Learn to create custom helper to display friendly date and time value

Prerequisites:

* Complete lab NWEB.S.L001.

Problem Description:

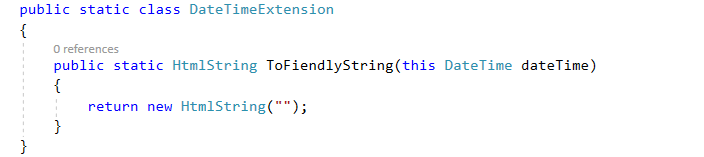
* How to use ViewBag
* How to use ViewData
* How to use TempData
* How to use strongly typed

Guidelines:

### Step 1: Open project that created in NWEB.S.L001

### Step 2: Add new folder named Extensions then add new class named DateTimeExtension

### Step 3: Add method named ToFiendlyString as an extension method



### Step 4: Create method to get friendly string

private static string GenerateRelativeString(DateTime dateTime)

{

TimeSpan span = (DateTime.Now - dateTime);

// Normalize time span

bool future = false;

if (span.TotalSeconds < 0)

{

// In the future

span = -span;

future = true;

}

// Test for Now

double totalSeconds = span.TotalSeconds;

if (totalSeconds < 0.9)

{

return "Now";

}

// Date/time near current date/time

string format = (future) ? "in {0} {1}" : "{0} {1} ago";

if (totalSeconds < 55)

{

// Seconds

int seconds = Math.Max(1, span.Seconds);

return String.Format(format, seconds,

(seconds == 1) ? "second" : "seconds");

}

if (totalSeconds < (55 \* 60))

{

// Minutes

int minutes = Math.Max(1, span.Minutes);

return String.Format(format, minutes,

(minutes == 1) ? "minute" : "minutes");

}

if (totalSeconds < (24 \* 60 \* 60))

{

// Hours

int hours = Math.Max(1, span.Hours);

return String.Format(format, hours,

(hours == 1) ? "hour" : "hours");

}

// Format both date and time

if (totalSeconds < (48 \* 60 \* 60))

{

// 1 Day

format = (future) ? "tomorrow" : "yesterday";

}

else if (totalSeconds < (3 \* 24 \* 60 \* 60))

{

// 2 Days

format = String.Format(format, 2, "days");

}

else

{

// Absolute date

if (dateTime.Year == DateTime.Now.Year)

format = dateTime.ToString(@"MMM d");

else

format = dateTime.ToString(@"MMM d, yyyy");

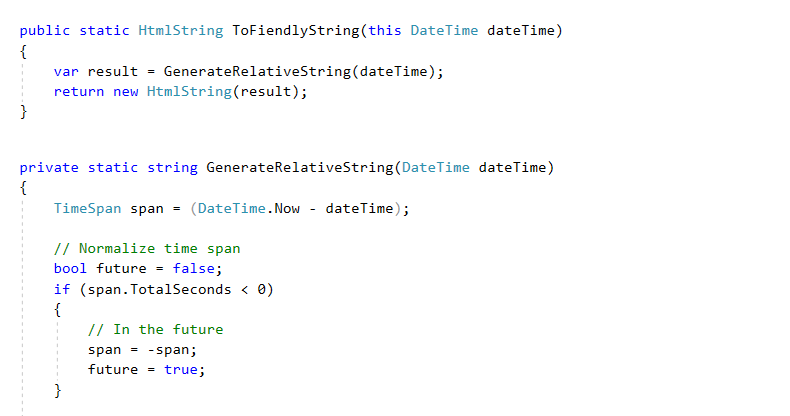
}

// Add time

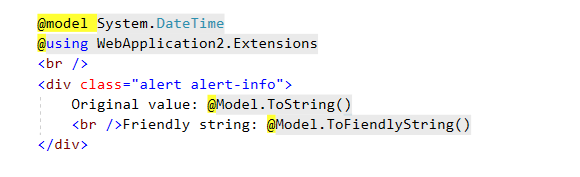
return String.Format("{0} at {1:h:mm tt}", format, dateTime);

}

### Step 5: Update method to return HtmlString



### Step 6: Retrieve message in view Index.cshtml



### Step 7: Check result

