

# Localising Data Annotation Attributes in Razor Pages

17 December 2019 07:33

ASP.NET CORE   RAZOR PAGES   LOCALIZATION

This is the third article in a series that explores various aspects of localisation in ASP.NET Core Razor Pages applications. The [first article looked at how to work with cultures](#), and the [second covered the basics of using resource files for static content translations](#). In this article, I explain how to provide localised resources for form labels and validation error messages for PageModel properties that have Data Annotation attributes applied to them.

The application in this article is the same one that has featured in the previous articles. It's built using the standard Razor Pages 3.1 project template with no authentication. Many of the concepts in this article were originally introduced in the previous article, so you should read that first.

The first three steps that follow demonstrate the minimum configuration to enable localisation using resources. If you are continuing from the previous article, you will have covered those:

1. In `ConfigureServices`, localization is added to the DI container, specifying the location of resources in the application, and the cultures supported by the application are configured:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddLocalization(options => options.ResourcesPath = "resources");

    services.Configure<RequestLocalizationOptions>(options =>
    {
        var supportedCultures = new[]
        {
            new CultureInfo("en"),
            new CultureInfo("de"),
            new CultureInfo("fr"),
            new CultureInfo("es"),
            new CultureInfo("ru"),
            new CultureInfo("ja"),
            new CultureInfo("ar"),
            new CultureInfo("zh"),
            new CultureInfo("en-GB")
        };
        options.DefaultRequestCulture = new RequestCulture("en-GB");
        options.SupportedCultures = supportedCultures;
        options.SupportedUICultures = supportedCultures;
    });

    services.AddSingleton<CommonLocalizationService>();
}
```

The `CommonLocalizationService` is a wrapper around an `IStringLocalizer<T>` which is used to access resource files. It was introduced as a means of accessing page-agnostic resource files in the previous article.

2. Localisation middleware is added after routing, passing in the localisation options specified in the previous step, in the `Configure` method:

```
var localizationOptions = app.ApplicationServices.GetService<IOptions<RequestLocalizationOptions>>
().Value;
app.UseRequestLocalization(localizationOptions);
```

3. A folder named *Resources* is added to the root of the application, containing an empty class named `CommonResources`:

```
public class CommonResources
{
}
```

Resources are accessed using a localization provider (`IStringLocalizer<T>`) which needs to work with a specific type. If the resources are intended to be used in just one page, you can use the `PageModel` class as the type. Translations for data annotations should be made available for use in more than one page, so they need to be set up to be page-agnostic. The `CommonResources` class provides a page-agnostic type for the resources. It's empty (has no members) because it is just a placeholder.

4. In the previous article, the `AddViewLocalization` extension method is used to add the view localisation services to the application's service collection. In this article, the `AddDataAnnotationsLocalization` extension method is chained to enable configuration of the `IStringLocalizer` to be used for accessing resources that contain data annotation translations. A factory is used to create an `IStringLocalizer` which is typed to the empty `CommonResources` class created in the last article to support global or page-agnostic resource files.

```
services.AddMvc().AddViewLocalization().AddDataAnnotationsLocalization(options =>
{
    options.DataAnnotationLocalizerProvider = (type, factory) =>
    {
        var assemblyName = new AssemblyName(typeof(CommonResources).GetTypeInfo().Assembly.FullName);
        return factory.Create(nameof(CommonResources), assemblyName.Name);
    };
});
```

The example that follows demonstrates the use of data annotations on `PageModel` properties that represent values posted from a form. The form is a simple contact form, in which all the form fields are required.

- 1. Add a new Razor page to the application named `Contact.cshtml`
- 2. Add the following using directive to the top of the `PageModel` file:

```
using System.ComponentModel.DataAnnotations;
```

3. Add the following properties with data annotation attributes to the `ContactModel`:

```
[BindProperties]
public class ContactModel : PageModel
{
    [Display(Name = "Message"), Required(ErrorMessage = "Message Required")]
    public string Message { get; set; }
    [Display(Name = "First Name"), Required(ErrorMessage = "First Name Required")]
    public string FirstName { get; set; }
    [Display(Name = "Last Name"), Required(ErrorMessage = "Last Name Required")]
    public string LastName { get; set; }
    [Display(Name = "Email"), Required(ErrorMessage = "Email Required"), DataType(DataType.EmailAddress)]
    public string Email { get; set; }
}
```

I haven't included any handler methods in this example because the focus is not on processing posted form values.

4. This step builds on the shared resource files that were introduced in the previous article. Translations for the labels and the error messages are added to the English, French and German resources, along with entries for navigation to the Contact page and the submit button on the form. Only the additional entries for German resource file (`CommonResources.de.resx`) are shown here for brevity:

Contact	Kontakt
Contact Us	Kontaktieren Sie Uns
Email	E-Mail
Email Required	Eine gültige E-Mail ist erforderlich
First Name	Vorname
First Name Required	Ein Vorname ist erforderlich
Last Name	Nachname
Last Name Required	Ein Nachname ist erfordlich
Message	Nachricht
Message Required	Eine Nachricht ist erforderlich

Submit

Senden

The keys for each entry are the values passed to the **Name** property of the **Display** attribute, and the **ErrorMessage** property of the **Required** attribute. The French resource file (*CommonResources.fr.resx*) needs translations for most of the same keys as the German one, except for the words *Contact* and *Message*, which are the same in French as in English. The English resource file needs "translations" for the error messages, unless you are happy with the existing (fairly concise) values assigned within the attributes.

In addition to the data annotation entries, there are three further entries. These are for the navigation, the title on the contact page and the button that will be used to submit the contact form.

5. The navigation can be added to the layout page, using the **CommonLocalizerService** that was created and injected into the layout page in the last article:

```
<li class="nav-item">
  <a class="nav-link text-dark" asp-area="" asp-page="/Contact">@localizer.Get("Contact")</a>
</li>
```

6. Finally, the form can be created in *Contact.cshtml*:

```
@page
@inject CommonLocalizationService localizer
@model Localisation.Pages.ContactModel
@{
    ViewData["Title"] = localizer.Get("Contact Us");
}

<h1>@localizer.Get("Contact Us")</h1>
<form method="post">
    <div class="form-group">
        <label asp-for="FirstName"></label>
        <input class="form-control" asp-for="FirstName">
        <span asp-validation-for="FirstName"></span>
    </div>
    <div class="form-group">
        <label asp-for="LastName"></label>
        <input class="form-control" asp-for="LastName">
        <span asp-validation-for="LastName"></span>
    </div>
    <div class="form-group">
        <label asp-for="Email"></label>
        <input class="form-control" asp-for="Email">
        <span asp-validation-for="Email"></span>
    </div>
    <div class="form-group">
        <label asp-for="Message"></label>
        <textarea class="form-control" asp-for="Message"></textarea>
        <span asp-validation-for="Message"></span>
    </div>
    <button class="btn btn-secondary">@localizer.Get("Submit")</button>
</form>

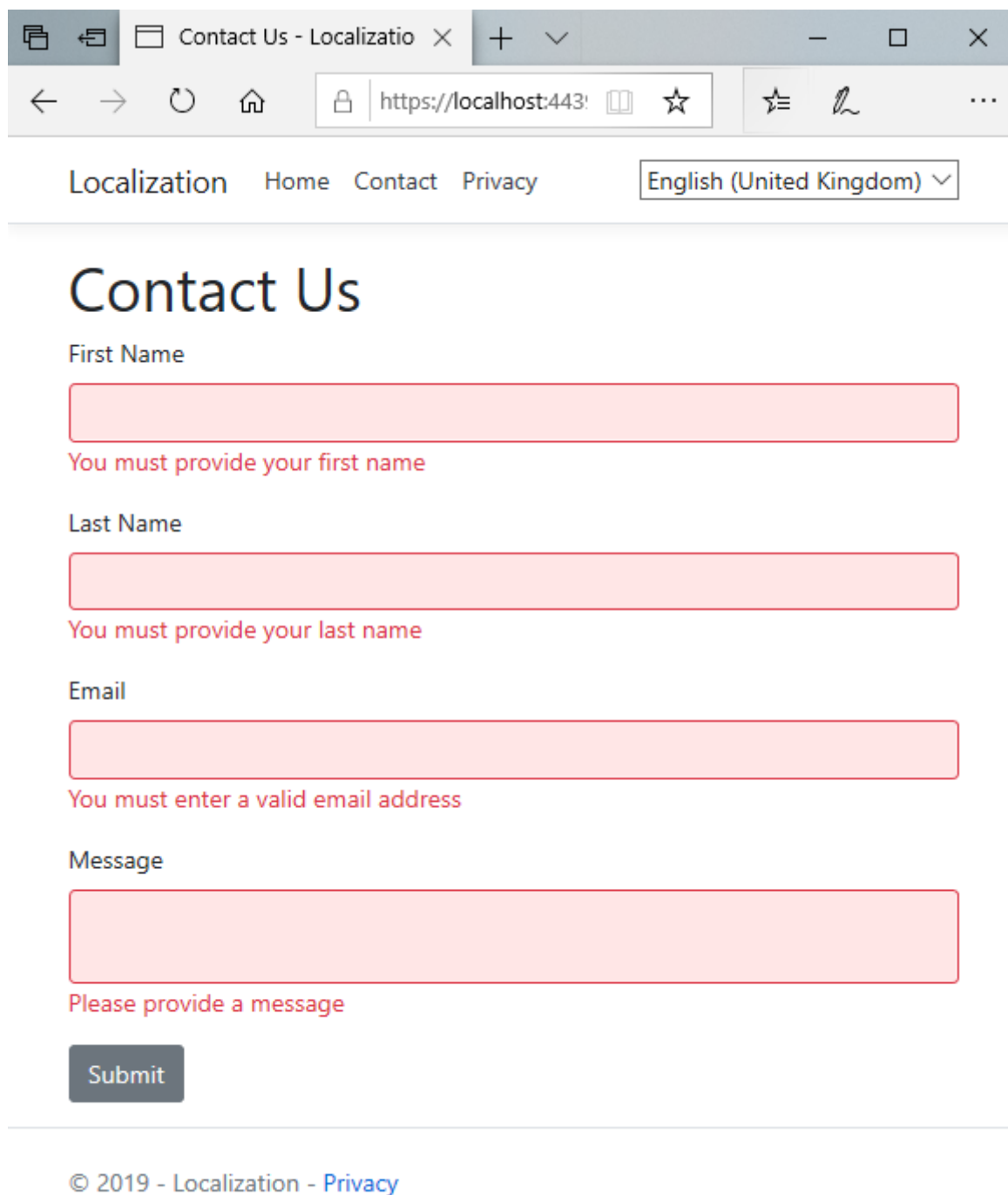
@section scripts{
    <partial name="_ValidationScriptsPartial"/>
}
```

The **CommonLocalizerService** is injected into the page, and is used to provide the translations for the page title, heading and the submit button. The rest of the form could be taken from any application. It uses standard tag helpers for labels, inputs and validation messages. Unobtrusive validation is enabled through the inclusion of the *ValidationScriptsPartial* file.

7. The final touch is to add some styles to the *site.css* file, in *wwwroot/css* to add some colour to inputs and messages in the event of validation failures:

```
.field-validation-error {
    color: #dc3545;
}
.input-validation-error {
    border-color: #dc3545;
    background-color: #ffe6e6;
}
```

If you run the application and navigate to the contact page, you can test the localisation simply by trying to submit the empty form. The client side validation should kick in since none of the required fields have values:



Localization Home Contact Privacy English (United Kingdom) ▾

## Contact Us

First Name

You must provide your first name

Last Name

You must provide your last name

Email

You must enter a valid email address

Message

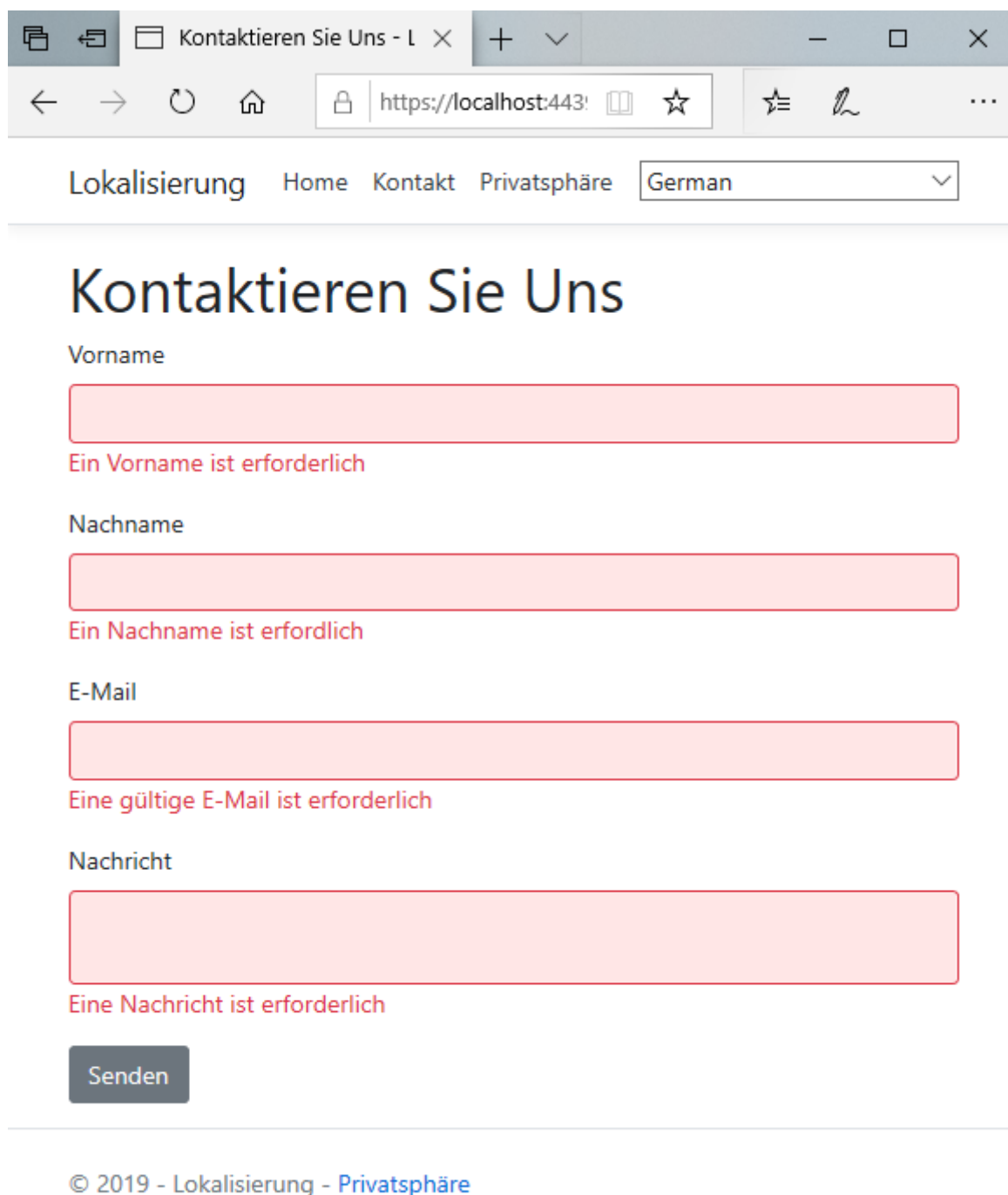
Please provide a message

Submit

---

© 2019 - Localization - [Privacy](#)

Then you can use the culture switcher to test translations:



Lokalisierung Home Kontakt Privatsphäre German ▾

## Kontaktieren Sie Uns

Vorname

Ein Vorname ist erforderlich

Nachname

Ein Nachname ist erforderlich

E-Mail

Eine gültige E-Mail ist erforderlich

Nachricht

Eine Nachricht ist erforderlich

Senden

---

© 2019 - Lokalisierung - [Privatsphäre](#)

## Summary

This article demonstrates how to localise data annotation attributes in a Razor Pages application. The process is based on the use of resources and requires its own configuration.

So far, the culture for a request has been set as a query string value via the Culture Switcher view component that was created in the first article in the series. This is [not a recommended approach](#). In the next article, I will look at how you can manage cultures via [Route Data instead](#).

◀◀ [Using Resource Files In Razor Pages Localisation](#)

[Razor Pages Localisation - SEO-friendly URLs](#) ▶▶

## Other Sites

[Learn Razor Pages](#)



## Categories

- [.NET 6 \(2\)](#)
- [ADO.NET \(24\)](#)
- [AJAX \(17\)](#)
- [ASP.NET 2.0 \(39\)](#)
- [ASP.NET 3.5 \(43\)](#)
- [ASP.NET 5 \(16\)](#)
- [ASP.NET Core \(61\)](#)
- [ASP.NET Identity \(3\)](#)
- [ASP.NET MVC \(89\)](#)
- [ASP.NET Web Forms \(31\)](#)
- [ASP.NET Web Pages \(89\)](#)
- [Blazor \(7\)](#)
- [Book Review \(7\)](#)
- [Bootstrap \(3\)](#)
- [C# \(40\)](#)
- [Classic ASP \(13\)](#)
- [CSS \(3\)](#)
- [Entity Framework \(34\)](#)
- [EPPlus \(4\)](#)
- [Extension Method \(6\)](#)
- [General \(12\)](#)
- [HTML \(8\)](#)
- [HTML5 \(2\)](#)
- [iTextSharp \(11\)](#)
- [Javascript \(22\)](#)
- [jQuery \(34\)](#)
- [LINQ \(5\)](#)
- [Localization \(4\)](#)
- [MS Access \(17\)](#)
- [Razor \(55\)](#)

- [Razor Pages \(44\)](#)
- [SEO \(3\)](#)
- [SQL \(6\)](#)
- [SQL Server Express \(2\)](#)
- [TypeScript \(1\)](#)
- [VB.Net \(29\)](#)
- [VBScript \(11\)](#)
- [Visual Studio \(5\)](#)
- [Web API \(1\)](#)
- [WebGrid \(16\)](#)
- [WebMatrix \(80\)](#)

# Archive

## 2021

- [July 2021 \(2\)](#)

## 2020

- [November 2020 \(3\)](#)
- [June 2020 \(2\)](#)
- [May 2020 \(1\)](#)
- [January 2020 \(1\)](#)

## 2019

- [December 2019 \(3\)](#)
- [November 2019 \(3\)](#)
- [October 2019 \(2\)](#)
- [August 2019 \(2\)](#)
- [July 2019 \(2\)](#)
- [May 2019 \(1\)](#)
- [April 2019 \(2\)](#)
- [March 2019 \(1\)](#)
- [February 2019 \(1\)](#)

## 2018

- [October 2018 \(3\)](#)
- [September 2018 \(4\)](#)
- [August 2018 \(2\)](#)
- [July 2018 \(2\)](#)
- [May 2018 \(2\)](#)
- [March 2018 \(2\)](#)
- [February 2018 \(1\)](#)
- [January 2018 \(1\)](#)

## 2017

- [September 2017 \(2\)](#)
- [July 2017 \(2\)](#)
- [May 2017 \(3\)](#)
- [February 2017 \(2\)](#)

## 2016

- [October 2016 \(2\)](#)
- [September 2016 \(1\)](#)
- [July 2016 \(2\)](#)
- [June 2016 \(1\)](#)
- [May 2016 \(1\)](#)
- [April 2016 \(1\)](#)
- [March 2016 \(2\)](#)
- [February 2016 \(1\)](#)
- [January 2016 \(2\)](#)

## 2015

- [December 2015 \(3\)](#)
- [October 2015 \(3\)](#)
- [September 2015 \(2\)](#)
- [August 2015 \(2\)](#)
- [July 2015 \(4\)](#)
- [June 2015 \(2\)](#)
- [May 2015 \(2\)](#)
- [April 2015 \(4\)](#)
- [March 2015 \(5\)](#)
- [February 2015 \(4\)](#)
- [January 2015 \(4\)](#)

2014

- [October 2014 \(2\)](#)
- [August 2014 \(1\)](#)
- [July 2014 \(1\)](#)
- [June 2014 \(12\)](#)
- [May 2014 \(11\)](#)
- [April 2014 \(1\)](#)
- [March 2014 \(2\)](#)
- [February 2014 \(2\)](#)
- [January 2014 \(1\)](#)

2013

- [December 2013 \(1\)](#)
- [November 2013 \(2\)](#)
- [October 2013 \(2\)](#)
- [August 2013 \(3\)](#)
- [July 2013 \(1\)](#)
- [June 2013 \(1\)](#)
- [May 2013 \(1\)](#)
- [February 2013 \(3\)](#)
- [January 2013 \(2\)](#)

2012

- [December 2012 \(4\)](#)
- [November 2012 \(1\)](#)
- [October 2012 \(1\)](#)
- [September 2012 \(5\)](#)
- [August 2012 \(2\)](#)
- [July 2012 \(2\)](#)
- [June 2012 \(3\)](#)
- [May 2012 \(1\)](#)
- [February 2012 \(1\)](#)
- [January 2012 \(1\)](#)

2011

- [December 2011 \(2\)](#)
- [October 2011 \(1\)](#)
- [September 2011 \(1\)](#)
- [August 2011 \(6\)](#)
- [May 2011 \(1\)](#)
- [April 2011 \(1\)](#)
- [March 2011 \(2\)](#)
- [January 2011 \(5\)](#)

2010

- [December 2010 \(3\)](#)
- [October 2010 \(4\)](#)
- [September 2010 \(2\)](#)
- [August 2010 \(2\)](#)
- [July 2010 \(9\)](#)
- [June 2010 \(2\)](#)
- [May 2010 \(7\)](#)
- [April 2010 \(1\)](#)
- [March 2010 \(1\)](#)
- [February 2010 \(4\)](#)
- [January 2010 \(2\)](#)

2009

- [December 2009 \(4\)](#)
- [November 2009 \(2\)](#)
- [October 2009 \(4\)](#)
- [September 2009 \(3\)](#)
- [August 2009 \(1\)](#)
- [July 2009 \(2\)](#)
- [June 2009 \(4\)](#)
- [May 2009 \(3\)](#)
- [April 2009 \(1\)](#)
- [March 2009 \(1\)](#)
- [February 2009 \(2\)](#)
- [January 2009 \(5\)](#)

2008

- [December 2008 \(4\)](#)

- [November 2008 \(5\)](#)
- [October 2008 \(6\)](#)
- [July 2008 \(1\)](#)
- [May 2008 \(3\)](#)
- [April 2008 \(2\)](#)

2007

- [November 2007 \(5\)](#)
- [September 2007 \(1\)](#)
- [August 2007 \(8\)](#)
- [July 2007 \(2\)](#)
- [June 2007 \(3\)](#)
- [May 2007 \(20\)](#)
- [April 2007 \(14\)](#)
- [March 2007 \(3\)](#)