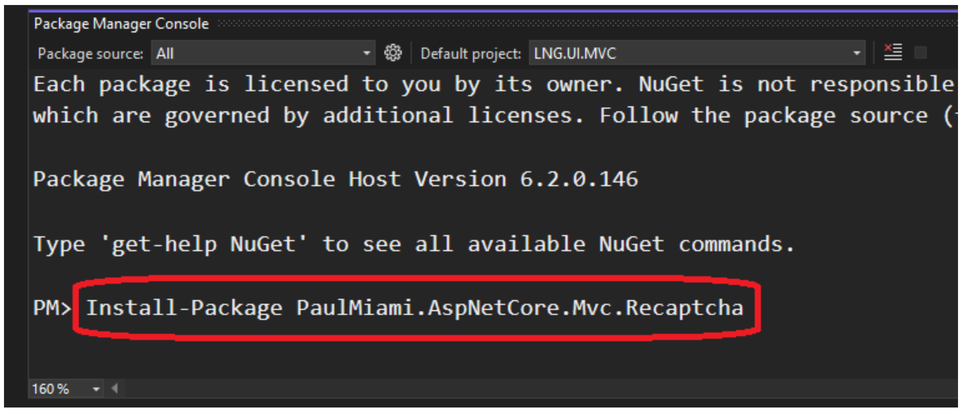
Adding Google reCAPTCHA to Your Contact Form

1. Open the NuGet Package Manager console and enter the following commandlet:

```
Install-Package PaulMiami.AspNetCore.Mvc.Recaptcha
```



Package Manager Console

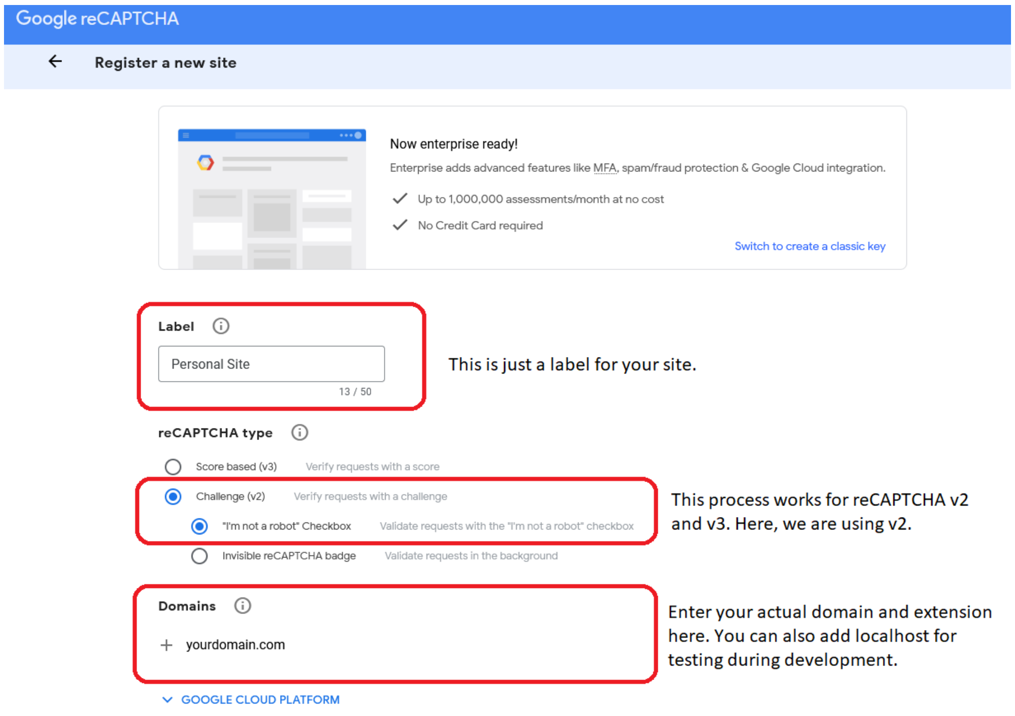
Package source: All | Default project: LNG.UI.MVC

```
Each package is licensed to you by its owner. NuGet is not responsible
which are governed by additional licenses. Follow the package source (

Package Manager Console Host Version 6.2.0.146

Type 'get-help NuGet' to see all available NuGet commands.

PM> Install-Package PaulMiami.AspNetCore.Mvc.Recaptcha
```

160 %

2. Open a web browser and go to: https://www.google.com/recaptcha/admin/create
   - You will need to register a Google account (or sign in to an existing one)

3. Enter a site label, select a reCAPTCHA type, and enter your site domain & extension.



Google reCAPTCHA

Register a new site

Now enterprise ready!

Enterprise adds advanced features like MFA, spam/fraud protection & Google Cloud integration.

✓ Up to 1,000,000 assessments/month at no cost

✓ No Credit Card required

Switch to create a classic key

Label: Personal Site (13 / 50)

This is just a label for your site.

reCAPTCHA type

○ Score based (v3) — Verify requests with a score

● Challenge (v2) — Verify requests with a challenge

● "I'm not a robot" Checkbox — Validate requests with the "I'm not a robot" checkbox

○ Invisible reCAPTCHA badge — Validate requests in the background

This process works for reCAPTCHA v2 and v3. Here, we are using v2.

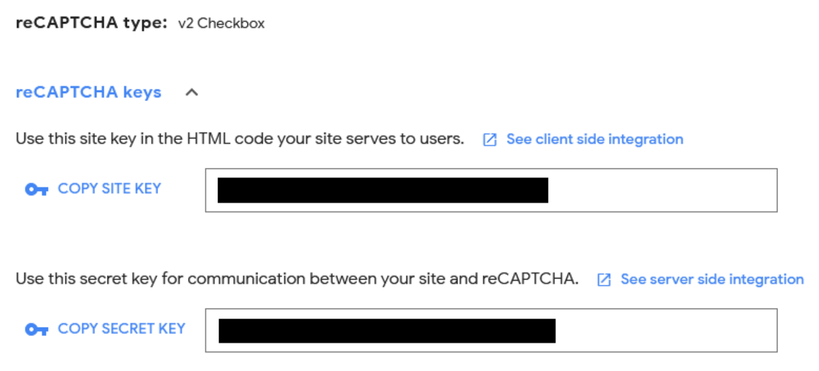
Domains

+ yourdomain.com

Enter your actual domain and extension here. You can also add localhost for testing during development.

GOOGLE CLOUD PLATFORM

4. Note the site key and secret key that are created for your site.



reCAPTCHA type: v2 Checkbox

reCAPTCHA keys

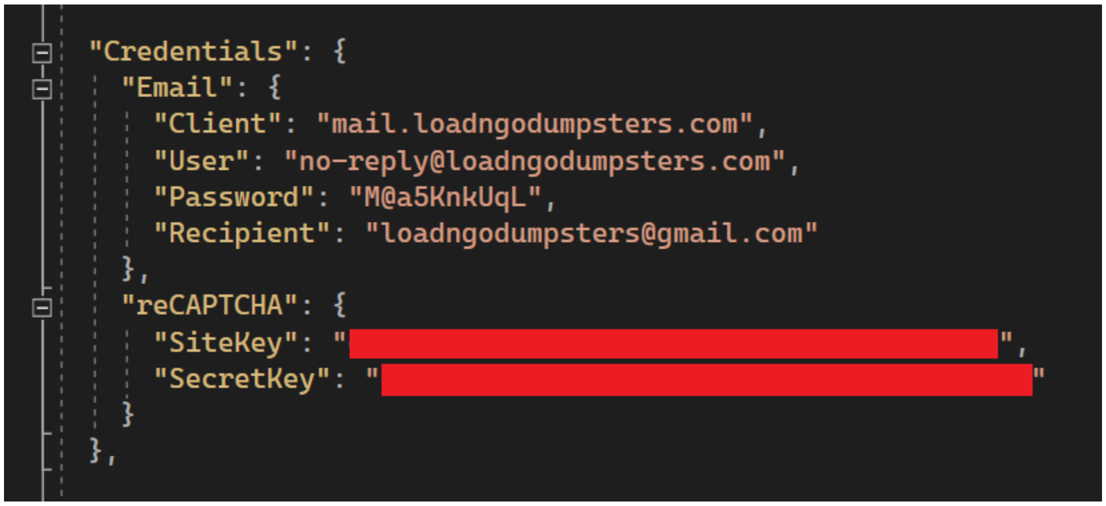
Use this site key in the HTML code your site serves to users. See client side integration

COPY SITE KEY [redacted]

Use this secret key for communication between your site and reCAPTCHA. See server side integration

COPY SECRET KEY [redacted]

5. Open appsettings.json and add a reCAPTCHA section with your keys.



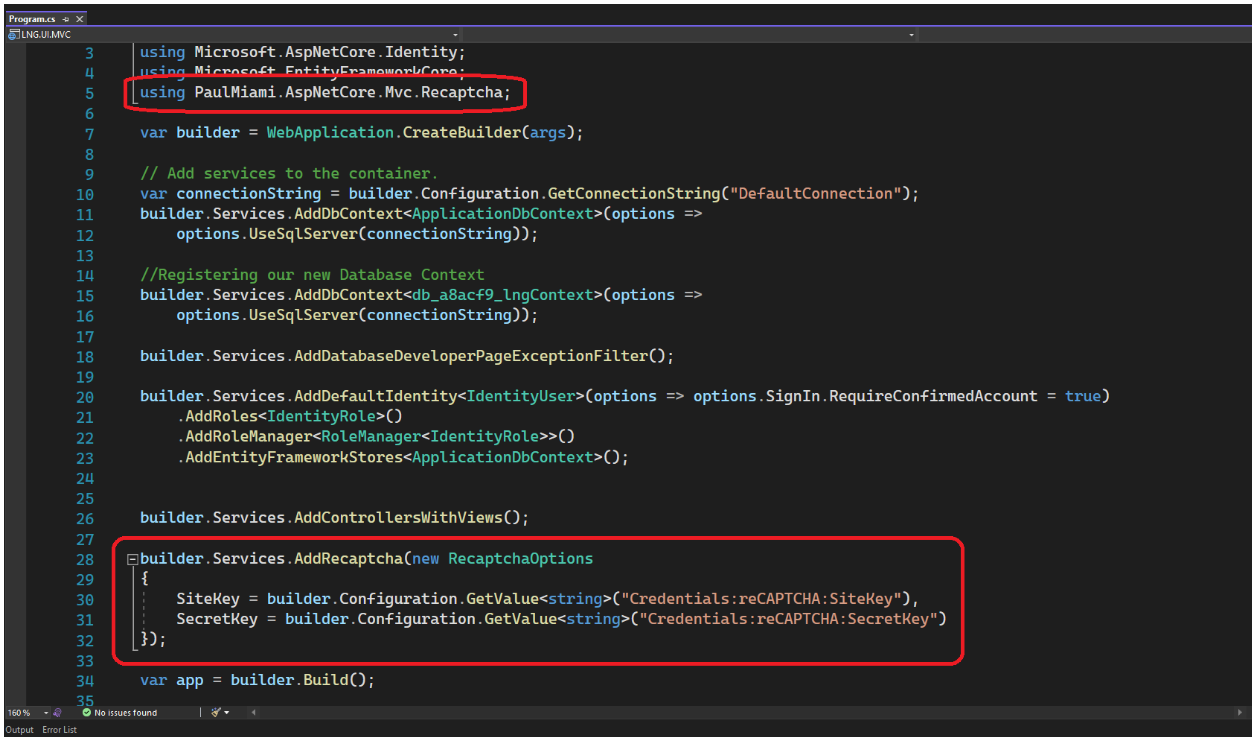
```
"Credentials": {
  "Email": {
    "Client": "mail.loadngodumpsters.com",
    "User": "no-reply@loadngodumpsters.com",
    "Password": "M@a5KnkUqL",
    "Recipient": "loadngodumpsters@gmail.com"
  },
  "reCAPTCHA": {
    "SiteKey": "[redacted]",
    "SecretKey": "[redacted]"
  }
},
```

6. Open Program.cs and add the following using statement:

```
using PaulMiami.AspNetCore.Mvc.Recaptcha;
```

Next, register and configure the reCAPTCHA service:

```
builder.Services.AddRecaptcha(new RecaptchaOptions {
 SiteKey = builder.Configuration.GetValue<string>("Credentials:reCAPTCHA:SiteKey"),
 SecretKey = builder.Configuration.GetValue<string>("Credentials:reCAPTCHA:SecretKey"),
});
```



Program.cs

LNG.UI.MVC

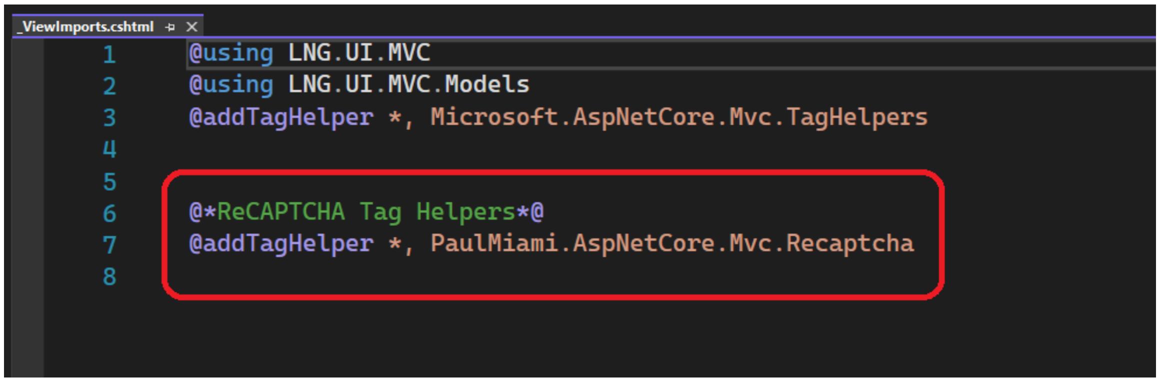
```
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using PaulMiami.AspNetCore.Mvc.Recaptcha;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(connectionString));

//Registering our new Database Context
builder.Services.AddDbContext<db_a8acf9_lngContext>(options =>
    options.UseSqlServer(connectionString));

builder.Services.AddDatabaseDeveloperPageExceptionFilter();

builder.Services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)
    .AddRoles<IdentityRole>()
    .AddRoleManager<RoleManager<IdentityRole>>()
    .AddEntityFrameworkStores<ApplicationDbContext>();


builder.Services.AddControllersWithViews();

builder.Services.AddRecaptcha(new RecaptchaOptions
{
    SiteKey = builder.Configuration.GetValue<string>("Credentials:reCAPTCHA:SiteKey"),
    SecretKey = builder.Configuration.GetValue<string>("Credentials:reCAPTCHA:SecretKey")
});

var app = builder.Build();
```

160 % No issues found

Output Error List

7. In your UI project, expand the Views folder, open _ViewImports, and add the following line to import the NuGet package's tag helpers:

   ```
   @addTagHelper *, PaulMiami.AspNetCore.Mvc.Recaptcha
   ```



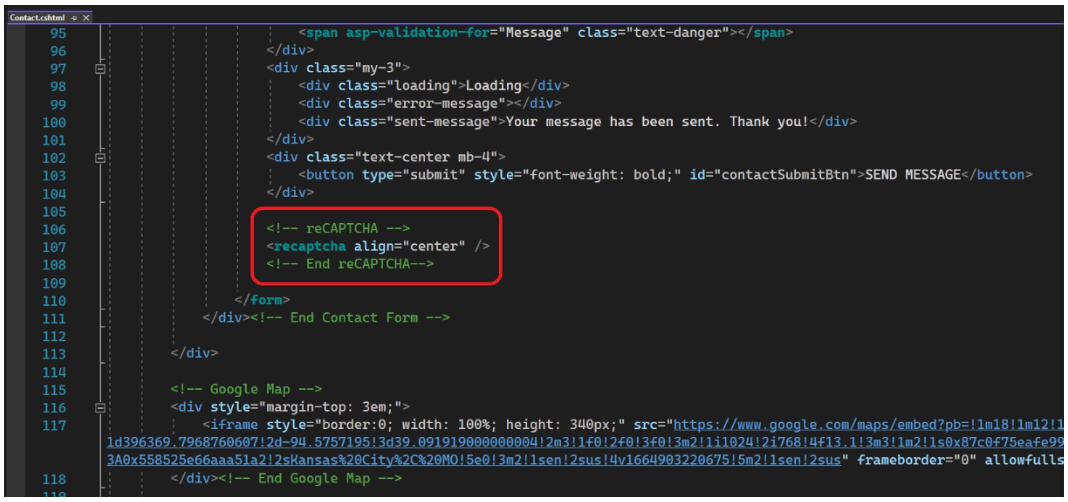
_ViewImports.cshtml

```
@using LNG.UI.MVC
@using LNG.UI.MVC.Models
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers

@*ReCAPTCHA Tag Helpers*@
@addTagHelper *, PaulMiami.AspNetCore.Mvc.Recaptcha

```

8. Open your Contact View and navigate to the <form>. Place the following tag helper where you would like the reCAPTCHA widget to appear:

   ```
   <recaptcha />
   ```

   *Note: The align="center" in the screenshot below is optional.

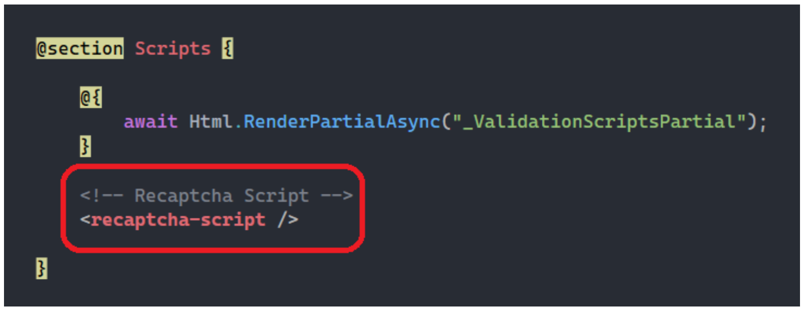


Contact.cshtml

```
                <span asp-validation-for="Message" class="text-danger"></span>
            </div>
            <div class="my-3">
                <div class="loading">Loading</div>
                <div class="error-message"></div>
                <div class="sent-message">Your message has been sent. Thank you!</div>
            </div>
            <div class="text-center mb-4">
                <button type="submit" style="font-weight: bold;" id="contactSubmitBtn">SEND MESSAGE</button>
            </div>

            <!-- reCAPTCHA -->
            <recaptcha align="center" />
            <!-- End reCAPTCHA-->

        </form>
    </div><!-- End Contact Form -->

</div>

<!-- Google Map -->
<div style="margin-top: 3em;">
    <iframe style="border:0; width: 100%; height: 340px;" src="https://www.google.com/maps/embed?pb=!1m18!1m12!1
1d396369.7968760607!2d-94.5757195!3d39.091919000000004!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x87c0f75eafe99
3A0x558525e66aaa51a2!2sKansas%20City%2C%20MO!5e0!3m2!1sen!2sus!4v1664903220675!5m2!1sen!2sus" frameborder="0" allowfulls
</div><!-- End Google Map -->
```

9. At the bottom of the Contact View, place the following tag helper in your Scripts section:

    ```
    <recaptcha-script />
    ```



```
@section Scripts {

    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial");
    }

    <!-- Recaptcha Script -->
    <recaptcha-script />

}
```

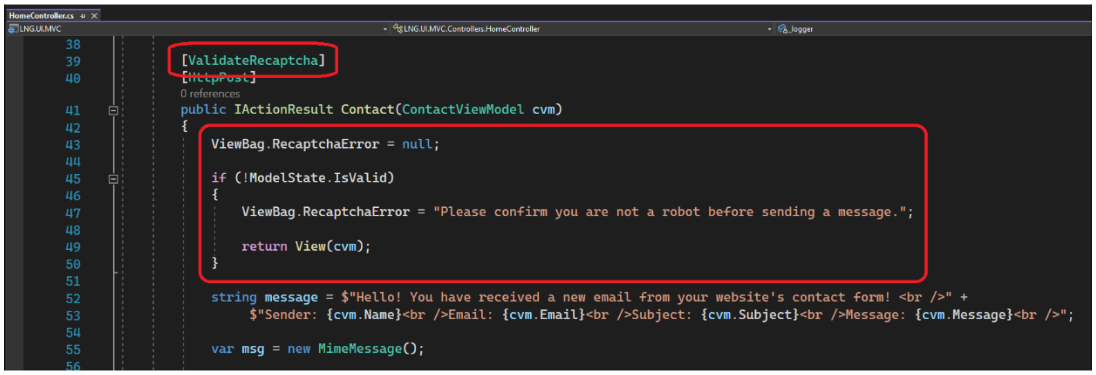
10. Navigate to the Controller for your Contact Actions and add the following using statement:

    ```
    using PaulMiami.AspNetCore.Mvc.Recaptcha;
    ```

11. Locate your Contact POST Action and add the following annotation:

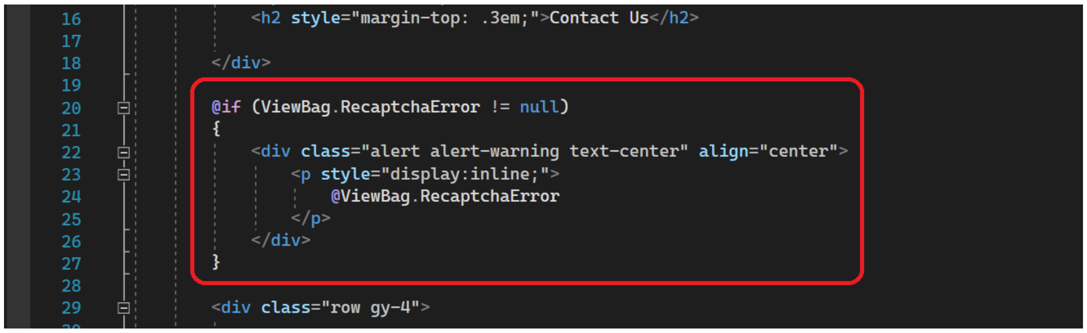
    ```
    [ValidateRecaptcha]
    ```

12. (OPTIONAL) In your Contact POST Action, you can store an error message in the ViewBag and pass it to the View in the event the user does not complete the reCAPTCHA.


HomeController.cs
LNG.UI.MVC
LNG.UI.MVC.Controllers.HomeController
_logger

```csharp
[ValidateRecaptcha]
[HttpPost]
0 references
public IActionResult Contact(ContactViewModel cvm)
{
    ViewBag.RecaptchaError = null;

    if (!ModelState.IsValid)
    {
        ViewBag.RecaptchaError = "Please confirm you are not a robot before sending a message.";

        return View(cvm);
    }

    string message = $"Hello! You have received a new email from your website's contact form! <br />" +
         $"Sender: {cvm.Name}<br />Email: {cvm.Email}<br />Subject: {cvm.Subject}<br />Message: {cvm.Message}<br />";

    var msg = new MimeMessage();
```

13. (OPTIONAL) If you completed the step above, add conditional logic to the Contact View to display the error message stored in the ViewBag.



```html
    <h2 style="margin-top: .3em;">Contact Us</h2>

</div>

@if (ViewBag.RecaptchaError != null)
{
    <div class="alert alert-warning text-center" align="center">
        <p style="display:inline;">
            @ViewBag.RecaptchaError
        </p>
    </div>
}

<div class="row gy-4">
```