This page describes how to use ASP.NET Web API within Kentico projects.

> **Note**: Kentico uses ASP.NET Web API requests for internal APIs. These internal routes are registered at the */cmsapi/* endpoint and are not meant for public use.

Start by adding a custom project that will contain your Web API logic:

1. Open your Kentico solution in Visual Studio.
2. Create a new *Class Library* project in the Kentico solution (or reuse an existing custom project). The project will contain your Api Controllers.
3. Add references to the required Kentico libraries (DLLs) for the new project:
    a. Right-click the project and select **Add -> Reference**.
    b. Select the **Browse** tab of the **Reference manager** dialog, click **Browse** and navigate to the ***Lib*** folder of your Kentico web project.
    c. Add references to the following libraries (and any others that you may need in your custom code):
        - **CMS.Base.dll**
        - **CMS.Core.dll**
        - **CMS.DataEngine.dll**
4. Reference the custom project from the Kentico web project *(CMSApp* or *CMS)*.
5. Edit the custom project's **AssemblyInfo.cs** file (in the *Properties* folder).
6. Add the **AssemblyDiscoverable** assembly attribute:

```
using CMS;

[assembly:AssemblyDiscoverable]
```

7. Right-click your custom project in the **Solution Explorer** and select **Manage NuGet Packages**.
8. Install the **Newtonsoft.Json** package into the project. The version must be **7.0.1** or newer.

> If you install a newer version than **7.0.1** of the *Newtonsoft.Json* package, you also need to synchronize the version in the Kentico web project:
>
> a. Right-click the Kentico web project *(CMSApp* or *CMS)* in the **Solution Explorer** and select **Manage NuGet Packages**.
> b. Update the **Newtonsoft.Json** package to the same version that you installed into your Web API project.

9. Install the **Microsoft.AspNet.WebApi** package.
10. Make sure the project contains a framework reference to the **System.Web** assembly.

Now you can define Web API controllers and register routes:

1. Create Web API controllers in your custom class library project. Make the controller classes inherit from the **System.Web.Http.ApiController** class.

```
using System.Net;
using System.Net.Http;
using System.Web.Http;

namespace MyCompany.MySpace
{
    public class CustomWebAPIController : ApiController
    {
        public HttpResponseMessage Get(int id = 0)
        {
            // You can return a variety of things in Web API controller actions.
For more details, see http://www.asp.net/web-api/overview/getting-started-with-
aspnet-web-api/action-results
            return Request.CreateResponse(HttpStatusCode.OK, new { Data = "test
data", ID = id });
        }
    }
}
```

2. Create a custom module class in your project.
3. Override the module's **OnInit** method and register custom Web API routes.

> **Note**: When adding routes, make sure there are no conflicts with the default Kentico **cmsapi** routes or any other page paths.

```
using System.Web.Http;

using CMS;
using CMS.DataEngine;

// Registers the custom module into the system
[assembly: RegisterModule(typeof(MyCompany.MySpace.CustomWebAPIModule))]

namespace MyCompany.MySpace
{
    public class CustomWebAPIModule : Module
    {
        // Module class constructor, the system registers the module under the name
"CustomWebAPI"
        public CustomWebAPIModule()
        : base("CustomWebAPI")
        {
        }

        // Contains initialization code that is executed when the application starts
        protected override void OnInit()
        {
            base.OnInit();

            // Registers a "customapi" route
            GlobalConfiguration.Configuration.Routes.MapHttpRoute("customapi",
"customapi/{controller}/{id}", new { id = RouteParameter.Optional });
        }
    }
}
```

4. Rebuild the solution.

You can now make Web API calls on the custom route that you registered.

Example

```
$http.get("http://myserver/customapi/CustomWebAPI", { id: 1})
    .success(function (data) {
        alert(data);
    })
    .error(function(error) {
        // Handle the error
    });
```