

SelfHosted AspNet WebAPI With Controller Classes In Different Project

Asked 10 years, 3 months ago Modified 10 years, 3 months ago Viewed 8k times



I have created a SelfHosted AspNet WebAPI with Visual Studio 2012 (.NET Framework 4.5). I enabled SSL for the WebAPI. It works fine when the controller is defined in the same project.

10



But when I add a reference of another project, containing controllers, it gives me the following error:



No HTTP resource was found that matches the request URI 'https://xxx.xxx.xxx.xxx:xxxxx/hellowebapi/tests/'.



I have created custom classes for HttpSelfHostConfiguration and MessageHandler.

Any help to resolve this problem would be a great time-savor for me.

Thanking in advance.

[asp.net-web-api](#) [multiple-projects](#) [self-hosting](#)

Share Improve this question Follow

asked Jun 21, 2013 at 2:07
user1624372

1 Answer

Sorted by: Highest score (default)



You can write a simple custom assemblies resolver which makes sure that your referenced assembly is loaded for the controller probing to work.

8



Following is a nice post from Filip regarding this:

<http://www.strathweb.com/2012/06/using-controllers-from-an-external-assembly-in-asp-net-web-api/>



Sample:



```
class Program
{
    static HttpSelfHostServer CreateHost(string address)
    {
        // Create normal config
        HttpSelfHostConfiguration config = new HttpSelfHostConfiguration(address);

        // Set our own assembly resolver where we add the assemblies we need
        CustomAssembliesResolver assemblyResolver = new CustomAssembliesResolver();
        config.Services.Replace(typeof(IAsembliesResolver), assemblyResolver);

        // Add a route
        config.Routes.MapHttpRoute(
            name: "default",
            routeTemplate: "api/{controller}/{id}",
            defaults: new { controller = "Home", id = RouteParameter.Optional });

        HttpSelfHostServer server = new HttpSelfHostServer(config);
        server.OpenAsync().Wait();

        Console.WriteLine("Listening on " + address);
        return server;
    }

    static void Main(string[] args)
    {
        // Create and open our host
        HttpSelfHostServer server = CreateHost("http://localhost:8080");

        Console.WriteLine("Hit ENTER to exit...");
        Console.ReadLine();
    }
}

public class CustomAssembliesResolver : DefaultAssembliesResolver
{
    public override ICollection<Assembly> GetAssemblies()
    {
    }
}
```

Share Improve this answer Follow

edited Jun 21, 2013 at 4:54

answered Jun 21, 2013 at 4:37




Kiran

57k

15

176

161

-
- 1 This shouldn't be necessary. The default resolver gets all assemblies from the current appdomain. I use controllers in other assemblies all the time and it works most of the time. I have to admit I have come across cases where it doesn't seem to work and I have never figured out why. – [Darrel Miller](#) Jun 21, 2013 at 11:00
-
- 5 @DarrelMiller: It would depend on whether you are referring a type in the other assembly which is causing the assembly to be loaded, in which case the probing would work. In the above sample, for example, without using the assemblies resolver I could just do something like `Type valuesControllerType = typeof(ControllersLibrary.ValuesController);` and this would cause the assembly to be loaded. – [Kiran](#) Jun 21, 2013 at 12:25 
-

Ahhh, thanks for that insight. I usually have a `MyApi.Configure(config)` method in the DLL with the controllers, so that would guarantee that the assembly is loaded. – [Darrel Miller](#) Jun 21, 2013 at 15:26

@DarrelMiller: I observed that if you are working with a MVC project, there is no problem in having controllers in another assembly. But, whenever you try to self-host your webapi, the self-host will not load other assemblies even if it has the project reference. You have to write your own assemblies resolver in that particular case. – [user1624372](#) Jun 21, 2013 at 16:42

@ImranNazir: You can look at my comment before. It not mandatory to have an assemblies resolver even in case of selfhost as long as you can cause the assembly to be loaded. – [Kiran](#) Jun 21, 2013 at 16:47
