



# STRATHWEB.

A FREE FLOWING TECH MONOLOGUE.

---

## CONTACT

## USING CONTROLLERS FROM AN EXTERNAL ASSEMBLY IN ASP.NET WEB API

2012/06/20 · 536 words · 3 minutes to read

[net asp.net web api c-sharp](#)

In general, in my day-to-day job, I am working with large enterprise web applications, where clean projects, noise reduction and test driven development are holy. Therefore I have always been a big fan of placing MVC controllers outside the actual web application project, in a separate class library (or libraries).

With MVC it was very simple, but recently I faced this challenge in Web API, and turns out neither passing the assembly name through `MapHttpRequest` (not supported) or using something like `ControllerBuilder.Current.DefaultNamespaces` is not a viable option. So I turned to [Henrik Nielsen](#) for advice, and sure enough, [he helped immediately](#). Let's explore the solution.

## THE SOLUTION

The solution to our problem is to write a custom class implementing `IAssemblyResolver`, hook it up to the `GlobalConfiguration` and add a list of our additionally required assemblies there.

Let's assume we have our controllers in a library called `ControllersLibrary`, and it is a regular class library, located at `c:/libs/controllers/ControllersLibrary.dll`.

If you are not used to that concept, think about that for a second. Controllers don't really need `Global.asax` or anything else the web application provides; they might as well live elsewhere, and be loaded from there - think of it as plugging them into your application.

To achieve all this, following [Henrik's advice](#), you need a class implementing `IAssemblyResolver`, for example, your own class derived from

```
System.Web.Http.Dispatcher.DefaultAssembliesResolver :
```

```
public class AssembliesResolver : DefaultAssembliesRe
{
    public override ICollection<Assembly> GetAssemblies()
    {
        ICollection<Assembly> baseAssemblies = base.GetAssemb
        List<Assembly> assemblies = new List<Assembly>(baseAs
        var controllersAssembly = Assembly.LoadFrom(@"C:\libs\c
        baseAssemblies.Add(controllersAssembly);
        return assemblies;
    }
}
```

So effectively you are asking the framework to get its default assemblies (located under web app's `/bin/` folder), and merge your custom assembly in there as well.

There is a kicker though. It only works with the latest Web API source (nightly Nuget packages or build from Codeplex), not with RC.

## MAKING THIS WORK WITH ASP.NET WEB API RC

Why? The issue here is that the class from which we inherited from, `DefaultAssembliesResolver`, is internal in RC, and was only made public after the RC was released.

However, there is no reason why you wouldn't write your own class implementing `IAssembliesResolver`. Let's modify the above code to get there:

```
public class CustomAssemblyResolver : IAssembliesResc
{
    public ICollection<Assembly> GetAssemblies()
    {
        List<Assembly> baseAssemblies = AppDomain.CurrentDomai
        var controllersAssembly = Assembly.LoadFrom(@"C:\libs<
        baseAssemblies.Add(controllersAssembly);
        return baseAssemblies;
    }
}
```

So this looks very similar to the bit above. Instead of inheriting from `DefaultAssembliesResolver`, we implement `IAssembliesResolver` directly, and read the assemblies from `AppDomain` ourselves (that's what `DefaultAssembliesResolver` does), and merge in a List with our custom assembly.

Now we are ready to plug this in, and as expected this has to be done in `Global.asax`, or any other class you have to manipulate `GlobalConfiguration`.

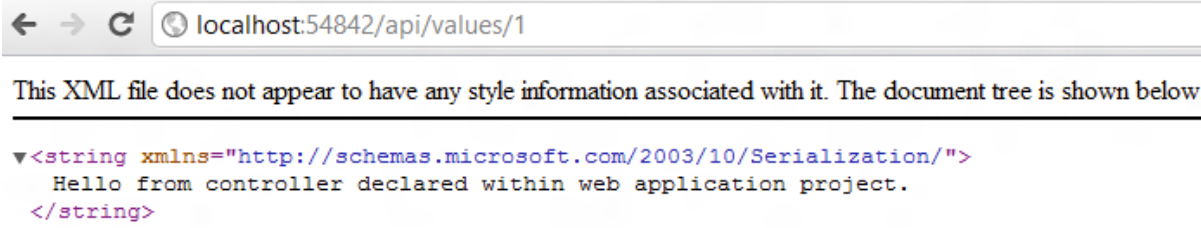
```
GlobalConfiguration.Configuration.Services.Replace(ty
```

## TRYING IT OUT

Now assuming that:

- `ValuesController` is located under main web application project
- `ExternalController` is located in `C:\libs\controllers\ControllersLibrary.dll`


You can see that it's easy to navigate to both of them:



← → ↻ 🌐 localhost:54842/api/values/1

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">
  Hello from controller declared within web application project.
</string>
```



← → ↻ 🌐 localhost:54842/api/external/1

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">
  Hello from external controller declared within ControllersLibrary.dll
</string>
```

## SUMMARY

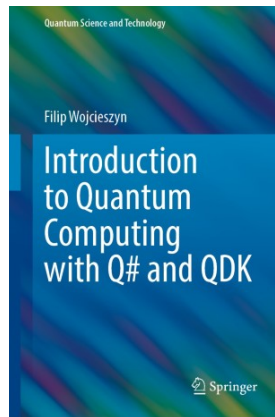
It is really easy to separate your controllers from your web application. While this might seem like an overkill in small projects, in large enterprise scale applications, it is an absolute lifesaver. I hope I don't have to convince anyone how much it helps maintainability, testability and mess/noise reduction.

ABOUT  
Enjoy!



Hi! I'm **Filip W.**, a cloud architect from Zürich 🇨🇭. I like **Toronto Maple Leafs** 🇨🇦, **Rancid** and **quantum computing**. Oh, and I love the **Lowlands** 🇳🇱.

You can find me on **Github** and on **Mastodon**.



## RECENT POSTS

2023/09/27, dotnet WASI applications in .NET 8.0

2023/09/13, Using embeddings model with Azure OpenAI Service

2023/08/22, Announcing Strathweb.Dilithium - a set of ASP.NET helper libraries for post quantum cryptography

2023/07/25, Calling Rust code from Swift on iOS and macOS

2023/06/22, Calling Rust code from C#

 [Full archive](#)

## CATEGORIES

ai (2)

apache-cordova (1)

asp.net-5 (17)

asp.net-core (45)

asp.net-mvc (35)

asp.net-mvc-6 (7)

asp.net-vnext (6)

asp.net-web-api (96)

astronomy (1)

azure (11)

azure-service-bus (1)

benchmark-dotnet (1)

bing-maps (1)

blazor (1)

c-plus (2)

c-sharp (153)

cryptography (3)

csharp (1)

csharp-10 (2)

dnx (3)

dotnet-cli (2)

dotnet-script (10)

duende (2)

editorconfig (1)

entity-framework (2)

espn-api (2)

events (1)

ffi (3)

fsharp (1)

git (1)

glimpse (1)

html5 (4)

identity-server (2)

iis (2)

il (1)

intro-to-qc (19)

ios (1)

javascript (9)

jquery (4)

jquery-mobile-metro (1)

katana (2)

kindle (1)

knockout.js (8)

last.fm-api (2)

linq (1)

mac (2)

macos (1)

mathematica (1)

msbuild (3)

mvc-core (3)

nancy (2)

native (1)

net (141)

net-5 (3)

net-6 (5)

net-7 (7)

net-8 (1)

net-core (49)

net-sdk (2)

ninject (2)

odata (4)

oidc (2)

omnisharp (13)

openai (2)

osx (2)

owin (5)

php (1)

q-sharp (30)

qir (2)

quantum-computing (33)

roslyn (30)

rust (2)

scriptcs (11)

scripting (9)

security (6)

servicestack (2)

signalr (7)

swift (3)

testing (5)



[twitter-bootstrap](#) (1)

[typescript](#) (1)

[visual-studio](#) (4)

[visual-studio-code](#) (11)

[wasi](#) (3)

[wasm](#) (3)

[windows-phone-7](#) (1)

[wordpress](#) (1)

[wpf](#) (2)

[RSS Feed](#) 

Copyright 2012-2023 strathweb.com / Filip W. All code samples are licensed under MIT license.

Adapted from [hugo-theme-flat](#)