**[HOW TO CREATE AN EXPANDED LIGHTSWITCH 2013 PROJECT!](https://blog.ofanitguy.com/2013/10/08/how-to-create-an-expanded-lightswitch-2013-project/)**

***This process requires the March 2014 Update***

*I’m happy to report that the March 2014 Update brought back SCAFFOLDING!! Yay!*

How to add Web API and MVC to a LightSwitch 2013 project

Visual Studio LightSwitch is a great product. But it’s not the end all and most professional development projects require a mix of technologies to be considered successful. By following these steps, you’ll end up with a LightSwitch project that has the HTML Client, Silverlight Client, MVC, Web API and Web Forms with the naming conventions of your choosing. Giving you a solution without restrictions!

First time thru the tutorial it may still take you 15 minutes. So give it a try and see what you think… Oh… btw you can clone/download the updated sample project from [github](https://github.com/dwm9100b/LsExtendedProject" \o "LsExtendedProject" \t "_blank)… you’ll need this, but really only for the WebApiConfig.cs.

1. Start Visual Studio 2013
2. Create a new project: File, New Project…
3. Select LightSwitch HTML Application
4. Name the project: myTest
5. Name the solution: myTest
6. Create new table
7. Name it TestItem, 1 property, Name
8. Add your browse/add/edit screens for the TestItems table
9. Double click on the properties file for the myTest project
10. Enable forms authentication, and allow Security Admin for debug
11. Save the solution, do a full build, run the app
12. Add some test data into the table
13. Back into Visual Studio
14. Right click on your server project
15. From the Add Scaffold window, click on MVC 5 Dependencies
16. Press the Add button
17. On the Add MVC Dependencies popup
18. Select the Full Dependencies option, press the Add button
19. Close the web.config that gets automatically loaded
20. In the App\_Start folder, open RouteConfig.cs
21. Add the following line under the other IgnoreRoutes

|  |  |
| --- | --- |
| 1 | routes.IgnoreRoute("{\*allsvc}", new { allsvc = @".\*\.svc(/.\*)?" }); |

1. Right click on the App\_Start folder, add a new Class file
2. Name it WebApiConfig.cs
3. Replace the contents with the contents from the WebApiConfig.cs file from our zip file
4. Add two folders under the Controllers folder
   1. api
   2. mvc
5. Right click on the api folder, select add, select New Scaffold Item
6. Select Web Api 2 Controller with Read/Write actions
7. Name it TestController
8. Follow the instructions of the readme.txt file that automatically opens in the editor
9. Make sure the GlobalConfiguration.Configure gets added to the beginning of Application\_Start
10. In the Controllers folder again, right click on the mvc folder, select add, select New Scaffold Item
11. Select MVC 5 Controller with read/write actions
12. Name it HomeController
13. In the HomeController.cs file that gets automatically loaded
14. Right click on the ActionResult Index()
15. Select add view
16. In the Add View dialog, leave all defaults, just click on add
17. Go edit your default.aspx page in the root of the solution
18. Comment out the first line… add some test text to the body
19. Do a full save, then rebuild, run your app

Your LightSwitch html app will run first with your custom URL path  
[http://localhost:{port}/App](http://localhost:%7bport%7d/App)

Moving on, let’s test access to the ApplicationData.svc file  
[http://localhost:{port}/ApplicationData.svc](http://localhost:%7bport%7d/ApplicationData.svc)

So far so good eh? How about our MVC?  
[http://localhost:{port}/Home](http://localhost:%7bport%7d/Home)

Yay! Looking good… now our default.aspx file in the root…  
[http://localhost:{port}/default.aspx](http://localhost:%7bport%7d/default.aspx)

Yep that works too… final test… the Web.Api side…  
[http://localhost:{port}/api/Test](http://localhost:%7bport%7d/api/Test)

You can now go in and clean up the project, delete the test screen, test table, etc.  
Save the solution again.

This would be the point when you would add your project into source control.

One of the great features out of this is we can use Visual Studio LightSwitch to do all of our data management without having to go down into the Entity Framework.

Happy coding!