|  |
| --- |
| Reverend Jack’s |
| Website Architecture |
| Transfer Guide |

|  |
| --- |
| Ryan Sweitzer  11-14-2017 |

Contents

[Overview 2](#_Toc498541015)

[Microsoft Visual Studio Forms Website 3](#_Toc498541016)

[HTML, ASPX markups and .CS code behind files 4](#_Toc498541017)

[SQL Server Connection and LINQ to SQL files 6](#_Toc498541018)

[Microsoft Server with IIS Website Management settings 9](#_Toc498541019)

[Domain name/DNS provider 13](#_Toc498541020)

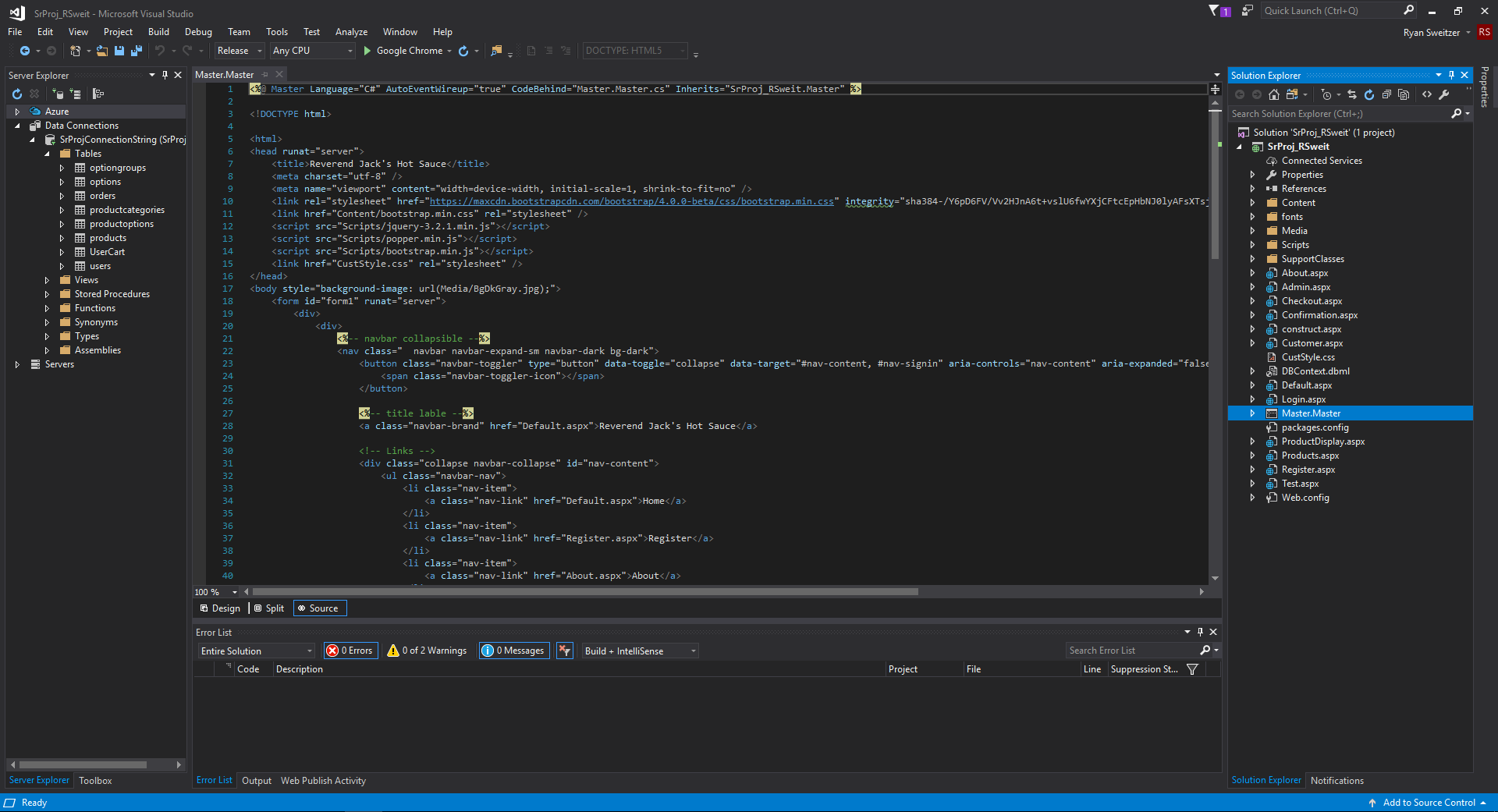
# Overview

This guide is to explain the overall architecture of the website and what is used to make it run. The purpose of making this guide is to ensure who ever is looking at the website knows how all the specific parts interact with one another and also serves as a guide for someone who comes in to take over duties as a website manager or joins the web development team at Reverend Jack’s. I think I would be useful for someone to have an overall guide of the whole website as well as have a document to reference if they run into an unfamiliar part of the site. My intent is to explain how generally all the parts of the website works:

1. Microsoft Visual Studio Forms Website and associated pages
2. HTML, ASPX markups and .CS code behind files
3. SQL Server Connection and LINQ to SQL files.
4. Microsoft Server with IIS Website Management settings.
5. Domain name/DNS provider.

All of the code in the files are commented to the best of my ability, they should fill in the details of how all the functions work and what their purpose is. As for how the website is hosted and accessible on the internet, how it gets and retrieves data from what data source and how the website is setup in general is what will be explained here. I will explain as far as I can go without giving specific credentials.

# Microsoft Visual Studio Forms Website



As you can see, this is mostly a typical view for a forms website. I have added a master page to handle having a persisting navigation bar and background across all pages and the code behind holds some methods that are used all over. All Images are stored in the media folder in their respective sub folders. The support classes folder contains all the C# code needed for the credit card transactions, if you need to modify the URL for the transactions or the associated email accounts, that will be located in the Configuration.cs file within that folder. I’ve incorporated Bootstrap from twitter into the website so the project will contain supporting CSS files as well as the scripts within the HTML, all located within the master page. As mentioned before, if you are looking for particulars of the coding itself then all the code is commented for what each piece does as well as all HTML sections defined in the comments.

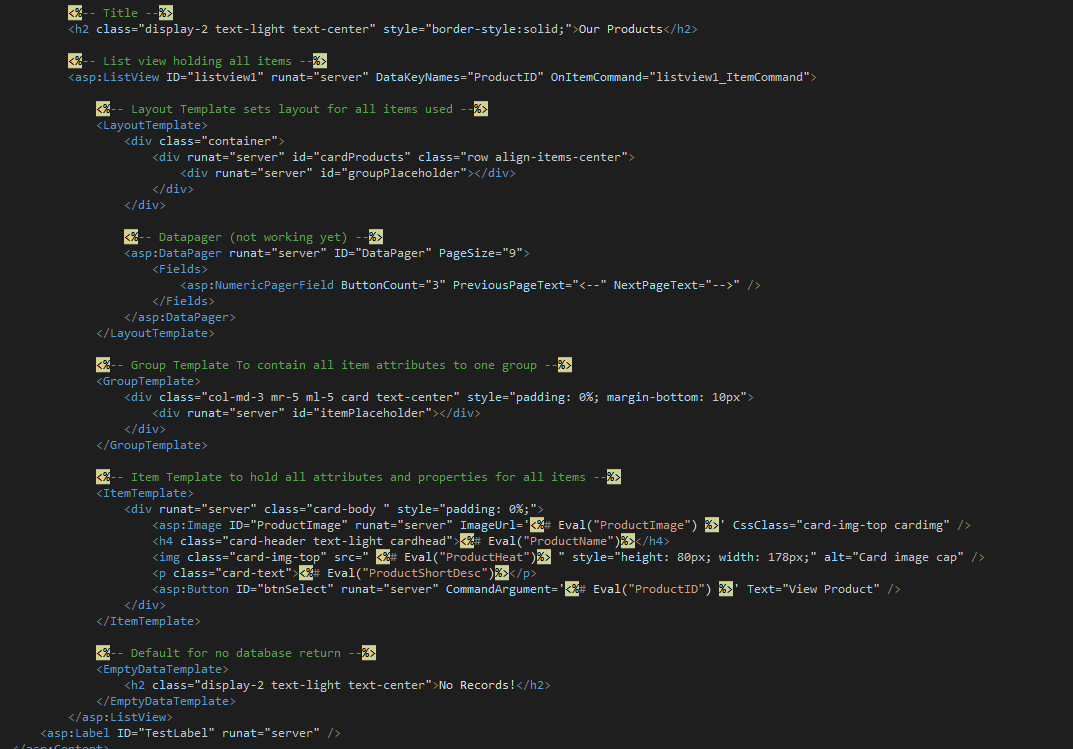
# HTML, ASPX markups and .CS code behind files



For the Admin, Customer and checkout pages, I have grid views set up to display database information for admins and users, this is a typical set up that I have in these sections along with SQL data source objects:

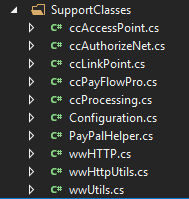


Its set up for one grid view to use on SQL data source. There are some grid views, mainly the user cart, that does not use a SQL data source but rather a LINQ to SQL call and binding the data right to the grid itself.

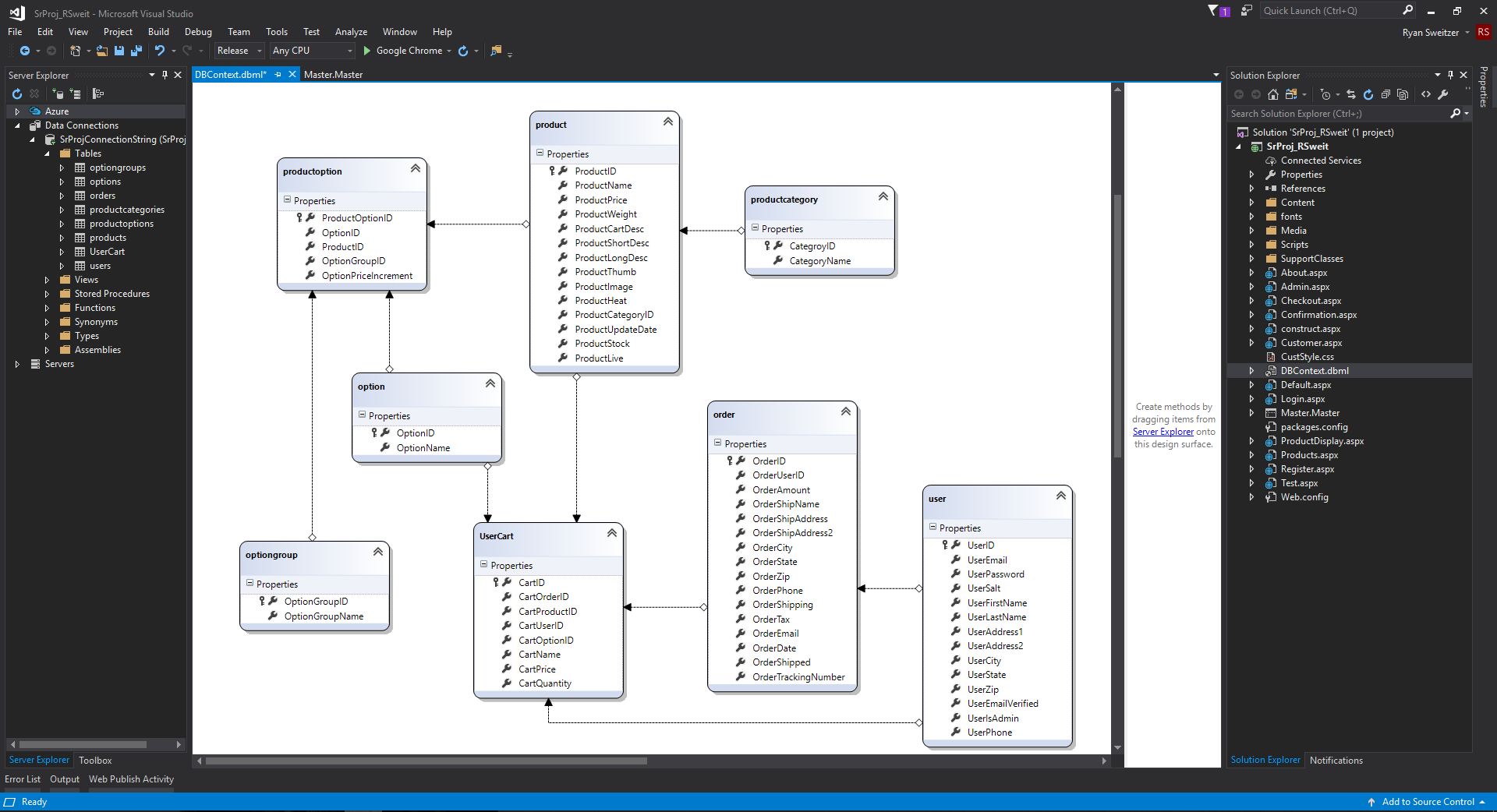


For the products page, I am using a custom list view to display the products using cards from Bootstrap and formatted with the list view all the while using a LINQ to SQL call to display everything in the database.

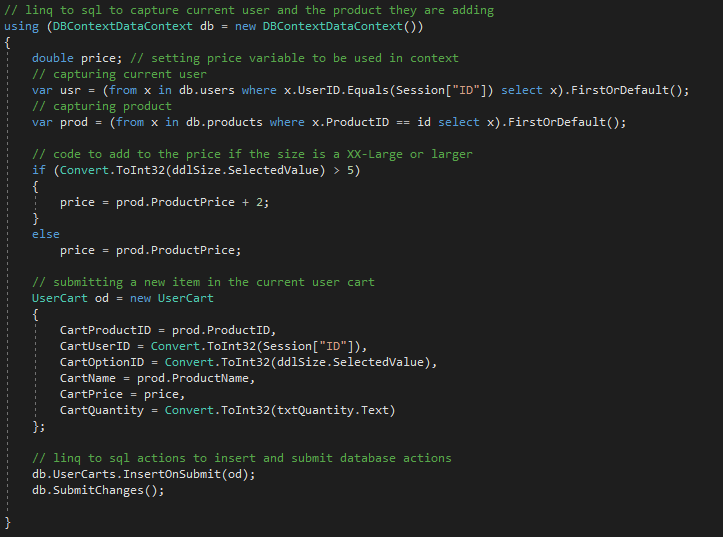
Again, if there is any changed that need to be made to the credit card processing, those C# files can be located in the supported classes folder:



# SQL Server Connection and LINQ to SQL files

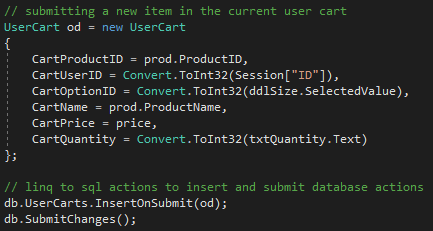


If you are unfamiliar with this file, this is a DBML file. This is what is used to generate the code, classes and methods associated with LINQ to SQL. The database was built using the SQL Server Management Studio using a AWS RDS instance of SQL server. The script that can build the database will be included with the project. Using Visual Studio’s Server explorer, I entered the required fields to import the database into the project, set up the DBML file then just dragged the tables into the view which automatically added the table associations all on its own. That allows for the project to use LINQ to SQL commands. When doing a LINQ to SQL command, you first want to create a context of the call. You can do this by simply calling the object of the database context file you created (DBML):

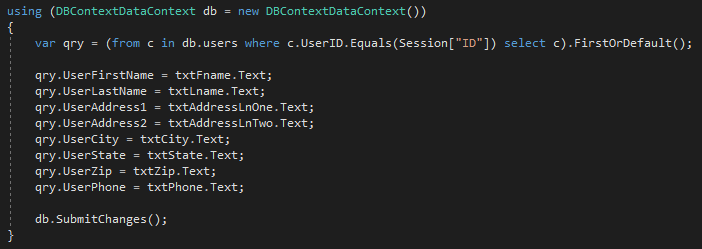


Next, you can make a database call by creating a variable, name it, then then use the LINQ to SQL syntax: 

This specific call is returning the user record that matches the Session variable’s ID, your typical SELECT statement. If you would like to do an insert, all you have to do is call the object for the table you want to insert into and use the following commands:

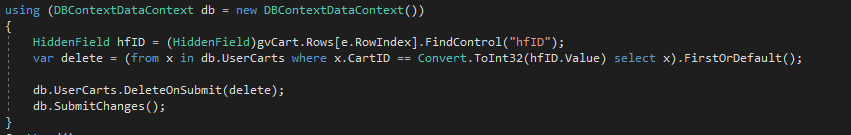


If you need to perform an update, it is as simple as same above, but only using these commands:



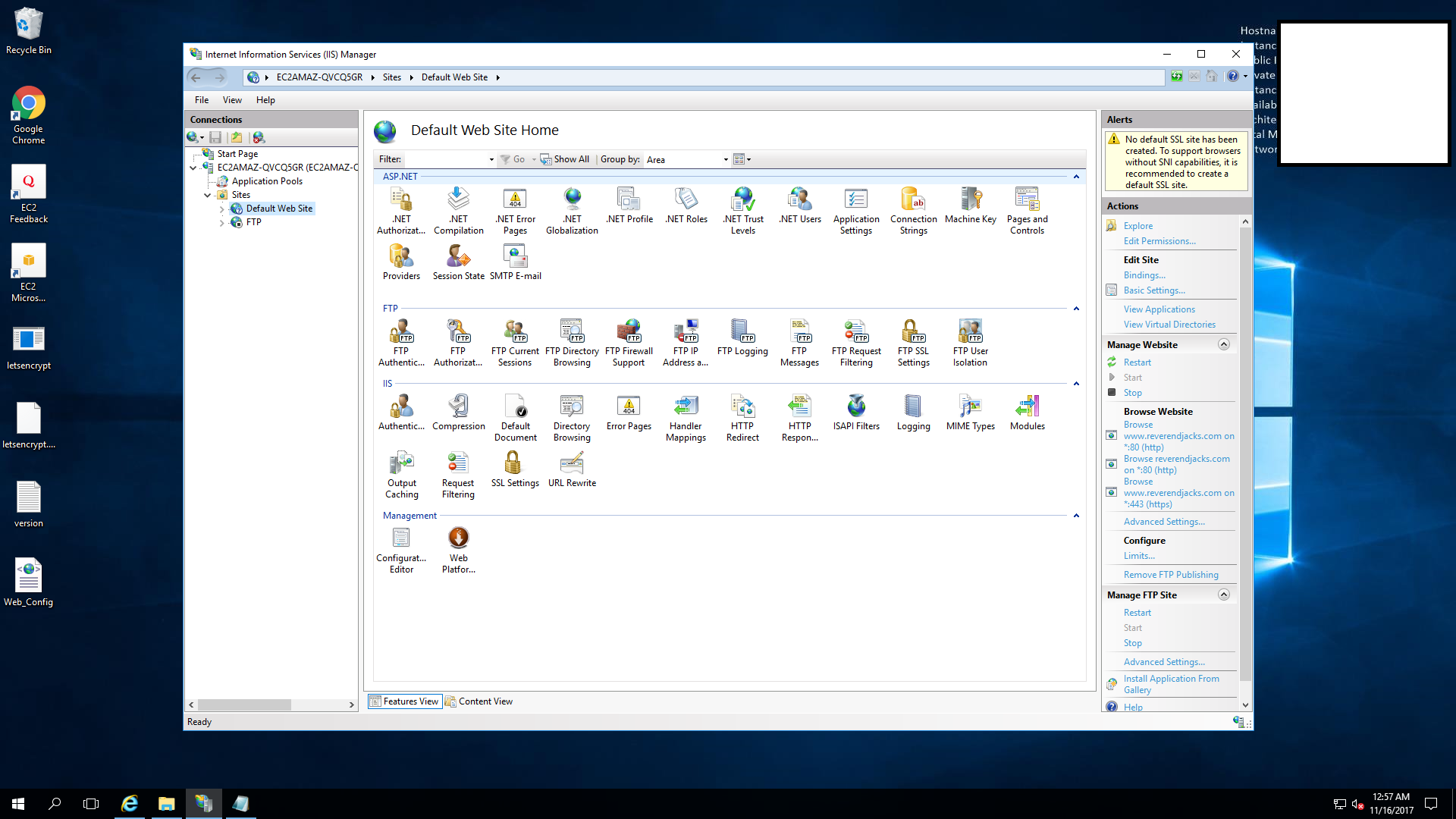
All you do is set the column attributes equal to the objects in your program such as textboxes in this example.

Lastly, an example for when you need to use LINQ to SQL to remove a record:

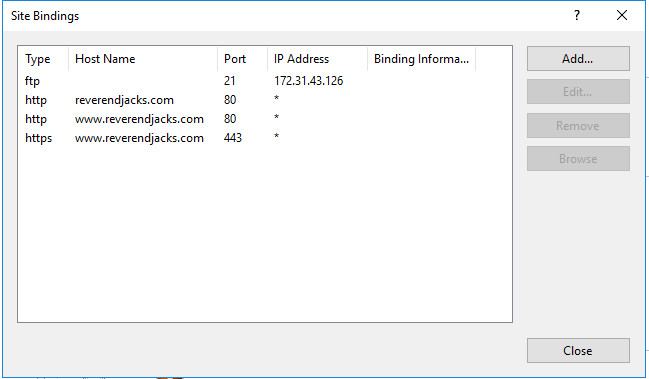


In this example, I’m finding the specific row to delete by the row ID then using LINQ to SQL to remove the row.

# Microsoft Server with IIS Website Management settings

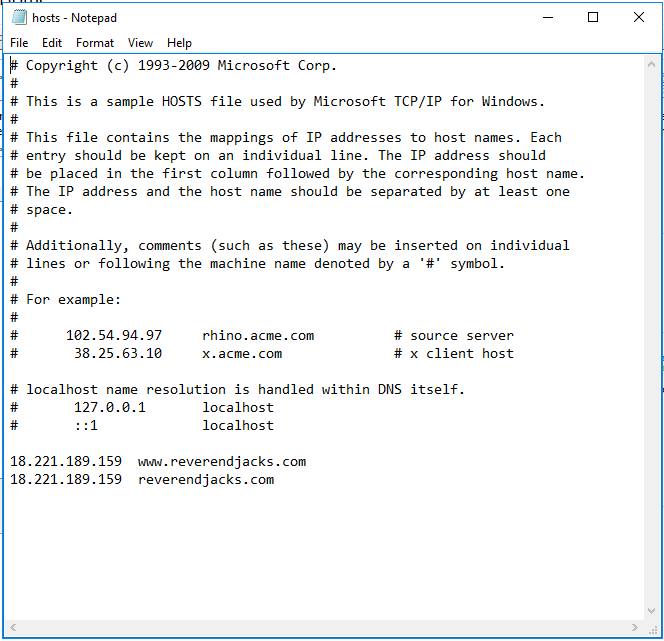


So, this is my EC2 instance in AWS. This is what I am using to host the website. Using IIS, it provides all the functionality you need to host a website. I’ll go over some configurations I went through to get this where it is now. Right clicking the website will bring up an option to view the bindings which will present this window:

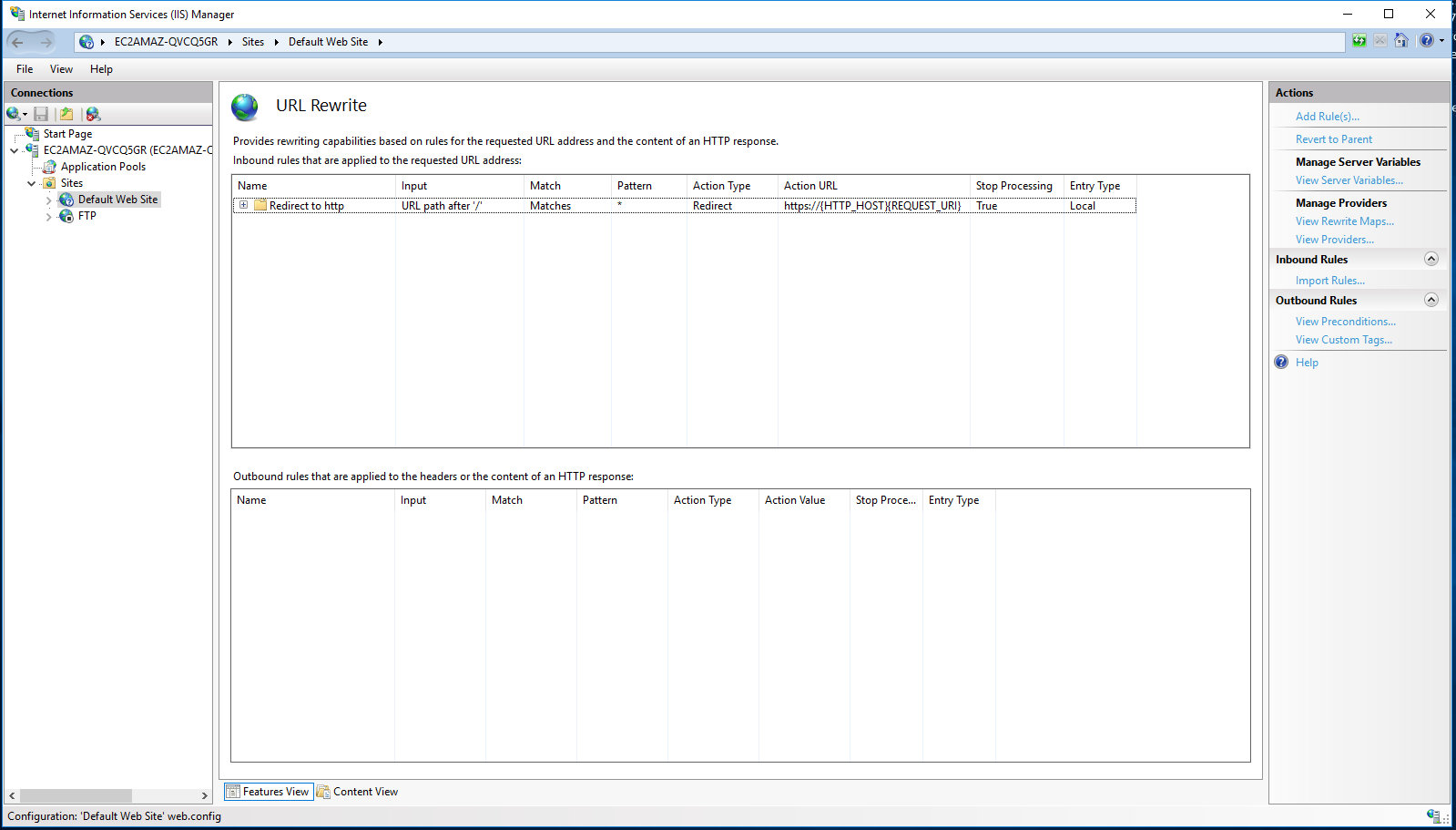


This is where you can change the address to the server to an actual domain name. Since you need a DNS server to actually find it without the IP address, that’s where you would go to a domain name provider like GoDaddy to point the DNS to this specific IP address, that will be shown in the next section.

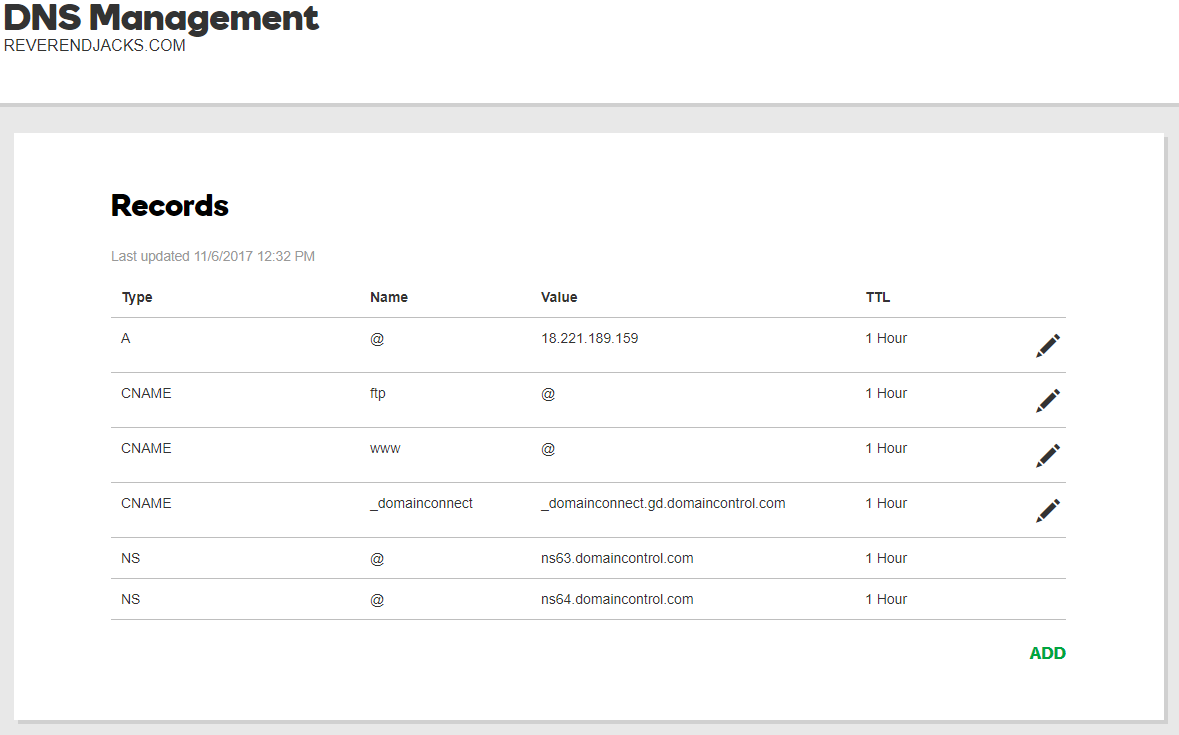
I’m not sure if its required, but I did change the “hosts” file to change the localhost of the Microsoft server to the domain name as well:



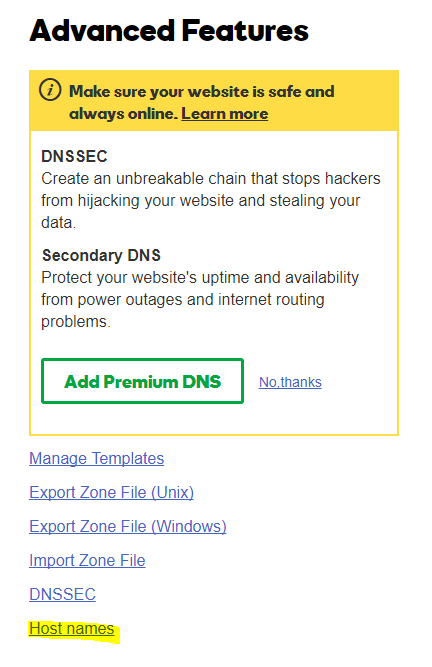
This file can be found in the directory C:\Windows\System32\drivers\etc.

To give the website SSL support, you need to have the domain name provider host the IP address of the website, once that happens, you can either get a certificate and add it in yourself to the server or use a program like Let’s Encrypt like I did. The instructions to use Let’s Encrypt are pretty straight forward. Once you have the SSL certificate, you need to include the URL Rewrite in the IIS installation so that the user is pointed to the HTTPS website instead of the non-encrypted one:

# Domain name/DNS provider



For this site, we are using GoDaddy for the domain name and DNS services for now. I have it configured to have the A record point to the current IP address of the website as well as using the host names option under advance features:





If there are any comments or questions, please direct them to the website owner or current website manager.