Author: Keith Wilson
Updated: June 19, 2014

Analysis of Coding Exercise: Morse Code Interpretation

The coding exercise requirements were to take a series of dots and dashes that represent Morse code and generate a list of the possible interpretations of those signals without having any indication of spaces between the letters of the received signal.

While I did not finish the coding exercise to a production level system, I spent about 12 hours on the exercise and built a working application that presents several significant coding techniques for the challenges of the application.

I chose to use tools from the stack in use at Acatar for the most part. My solution uses C#, MVC 5, SQLExpress with Entity Framework, and Bootstrap. I developed the web application with Visual Studio 2013 Express. I installed these tools on a machine at home, which was an interesting and production exercise in itself.

There were four specific challenges that I saw in the exercise:

1) Designing a simple and usable user interface for the application using Bootstrap
2) Making use of the database in an application that did not truly need a database to meet the requirements
3) Building the application with MVC 5
4) Designing a solid and usable "Morse code interpreter"

I started with the Morse code interpreter since that is the heart of the application. I built a console application with Visual Studio 2010 that made use of my MorseToAlpha class to normalize and interpret the Morse code input. This allowed me to play with the class easily and compare it to the reference implementation that was provided by Acatar. It took about 2 hours of work to build this application and get it to work correctly using a recursive approach to interpreting the string of dots and dashes.

Rather than worry about validation of input, I chose to simply pull the valid dots and dashes (and underscores) from whatever string was passed to the application. All other characters are discarded by the Normalization method before the interpretation begins. This simplified the use of the class. I also chose to limit the length of the input string to 24 dots and dashes to avoid memory issues. The requirements did not specify a limit, but in comparing my console app to the reference implementation, my app held up well in comparison with that length of input. In my tests I found that using a 24 dot/dash stream of Morse code results in more than 3,000,000 possible interpretations. For the sake of this exercise, I believed that to be sufficient.

Next, I began to build the shell of the web application after installing Visual Studio 2013 Express. I had not used Bootstrap for a while and doing a simple MVC application as well required a bit of research. The main on-line sources are listed below for the research that I did.

I wanted to keep the UI simple. I realize that the UI for a real application would need to be designed carefully to meet client needs. Since the exercise did not call for that, I chose not to spend a lot of time building the user interface beyond a simple, pleasing, and usable interface. It basically has a single page with the input on the left hand 1/3 of the screen and the results displayed on the right 2/3 of the screen. I chose to display some details about the interpretation along with the first 200 results of the interpretation.

I have included my console application to all a listing of the full result set. In the course of building the application, I ran into issues with the WebGrid that I used to display the results. I show the first 200 results, but the paging did not work correctly for me and after some time spent trying to correct it, I chose to leave it as a missing feature for the purpose of this exercise.

I built a two table data model and store the results of each interpretation in the database. The application will retrieve the interpretation of a string from the database if that string has been presented before. I also track and display the number of times that a string is interpreted.

I had intended to display a list of existing interpreted strings on the left side of the screen and allow the user to select one of those to see the list of possible values. However, this was also left as a future feature to be added.

I considered making more use of JavaScript. I do not store the presentation of how the Morse code characters are broken up into letters in the database. I chose to recalculate those in the WebGrid after the Morse code has been interpreted or retrieved from the database. I wrote a JavaScript function to build the representation of the string as Morse code, and I would like to have used the resources of the client machine to compute that. This might help a busy server to service more clients if that load was taken off, but I did not end up implementing that feature.

There are a large number of performance improvements that would be needed if this application was to be fully built out. The paging of results would be a critical issue once the length of the Morse code stream gets large. It takes several minutes to display the list of possible words for a long Morse code stream on the screen with the console application. Paging would allow us to reduce that time significantly for the display of possible interpretations.

I believe the interpretation of the stream itself is pretty well written. It is not the bottleneck even when the stream is 24 characters long. I implemented in the model the ability to record the amount of time it takes to interpret the stream and to retrieve the stream from the database. While not fully implemented, the thought was to only use the database when the retrieval of the data from the database was quicker than recomputing the possible words.

In general I am happy with the application given the amount of time I spent on it and the challenges and new tools, techniques, and technologies I used in the solution. As such, it was a very good exercise for me and I learned a lot about building .Net applications with these tools.

Resources Referenced

Pro ASP.NET MVC 3 Framework, 3rd Edition, Adam Freeman and Steven Sanderson, Apress, 2011.

MSDN website

http://www.itorian.com/2012/08/code-first-approach-in-entity-framework.html

http://msdn.microsoft.com/en-us/data/jj193542.aspx

http://www.dotnetcurry.com/showarticle.aspx?ID=618

http://www.asp.net/mvc/tutorials/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application

http://weblogs.asp.net/scottgu/asp-net-mvc-preview-5-and-form-posting-scenarios

http://msdn.microsoft.com/en-us/magazine/hh288075.aspx

http://ddmvc4.codeplex.com/

http://www.asp.net/mvc/tutorials/mvc-5/database-first-development/setting-up-database

http://www.mytecbits.com/microsoft/dot-net/bootstrap-3-with-asp-net-mvc-5