# Engineering Notes

## MVC vs Razor Pages (MVVC)

I am using Razor Pages for this solution, since it removes much of the boiler plate code typical of an MVC implementation (I have a lot more experience with MVC however).

## DecimalTranslatorServiceImpl.cs

This is my solution to the “Number to Words” problem presented here. I have defined the INumberTranslatorService interface, to decouple any single implementation for the operation so to allow for additional implementations to be added other than the standard decimal numbering system if required. I am using dependency injection to add the implementation into the application’s context and should allow for simple context switching to be used for any locales or regions that may not be compatible with the same decimal numbering system.

The standard decimal numbering system is convenient since it exhibits a reoccurring pattern. This specific implementation simply tokenizes the number to be translated into parts (i.e. every three digits), translates each individually, and then concatenating each translated part accordingly with their relevant “grouping” name (see DecimalTranslatorServiceImpl.LARGE\_NUMBERS list).

For example: “… [X0] **Billion** [X1] **Million** [X2] **Thousand** [X3­­] …”, where Xi is then each translated to their corresponding “[Y] **Hundred And** [Z]” parts.

I will point out that the pattern does display an element of recursion that could be taken advantage of here, but I have opted for a tokenising approach since I believe it is simpler and would be a lot more easily understood, and, in turn, improve maintainability.

To support very large numbers, I had considered using a basic string tokenizer (since strings won’t have the same size limitations as decimals), as well as a divide and modulo approach presented here using BigIntegers which takes advantage of the fact that each digit uses base 10 positioning. Basic benchmarking (via measuring CPU time) showed that the divide and modulo approach to be marginally more performant and so I chose to put this solution forward (NB: benchmarks were taken from Java implementations though). I have also defined a new BigCurrency type to represent the number value being translated for increased type safety.

Another approach I briefly considered was to generate all the possible word values in an “eager” fashion and storing them into a map for quick look up using the number to be translated as the key (thus maximising performance). However, this means adding additional memory pressure since it will involve caching a large amount of data in-memory. An alternative to populating the map eagerly, could be to do so lazily as requests are made by the user. This approach probably won’t lend itself very well with respect to localisation either.

Yet another approach could be to use a simple series of “if” constructs that checks whether the value falls within a group and then allocating the words accordingly. I chose not to use this approach as I envisioned that this would not be very readily extensible for larger numbers and the resulting code would, whilst probably more readable to an extent and perhaps more performant even, grow unchecked over time and ultimately become unmanageable.

## Input Validation

Basic validation has been implemented on both the client and server side as per standard best practice. Defensive programming practices have been observed where I think it was necessary.

## Internationalisation (i18n) and Localisation (l10n)

I have added some very basic support for this concern. I believe any application, especially a web-based application, should at least consider this so to reach a broader user base and generally enhance user experience. Just for demonstration purposes I have provided a translation resource bundle for Spanish (courtesy of Google Translate).

## Testing

Standard automated unit tests have been added as per normal practice. I would have added integration tests as well but felt it unnecessary for this exercise. I am familiar with selenium and we use it extensively at my current employer to drive such tests on different web browsers under different configurations.