HaloGo Converter Deployment Guide

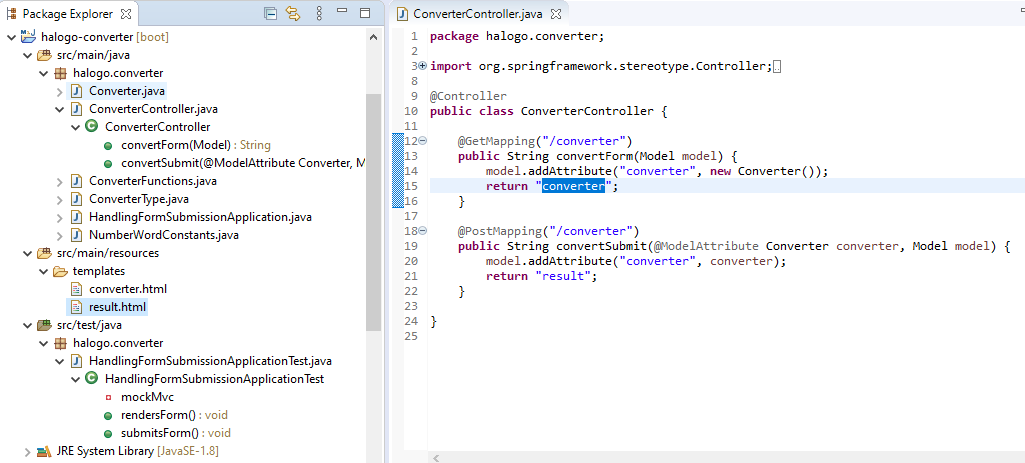
# Overview

This is a simple web application which captures a person’s name and a number in a form on one page and then displays the name as entered and the number converted to words on a second web page.

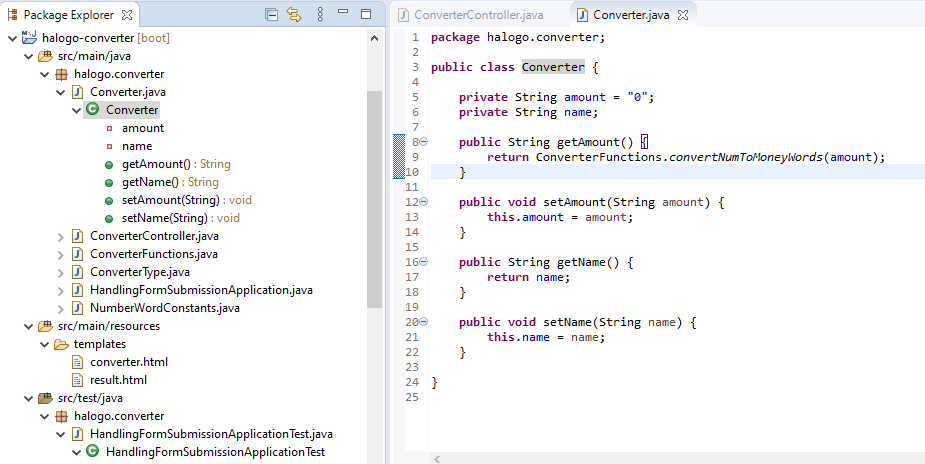
# Design

The application requires Java 1.8.0\_162 or similar to run, and was built using Spring Boot 2.2.2, and Spring MVC using Spring Tool Suite 4 as an IDE.

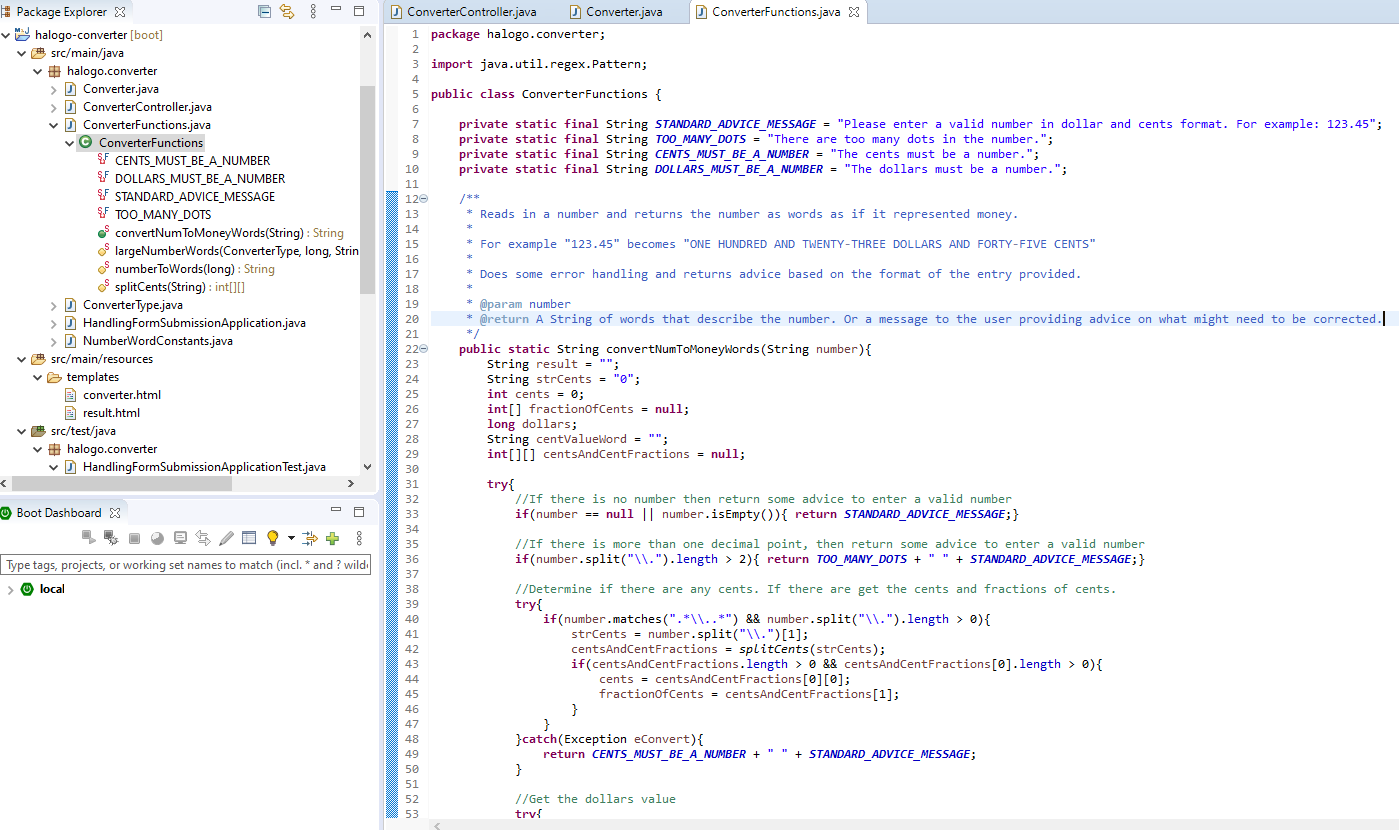
At the core, the ConverterController class uses the @GetMapping and @PostMapping annotations to map the HTTP GET and HTTP POST requests onto specific handler methods:



The Converter class holds the name and amount data, with getters and setters which are simple except for the getAmount() method which calls the a static function that performs the conversion from a number to words:



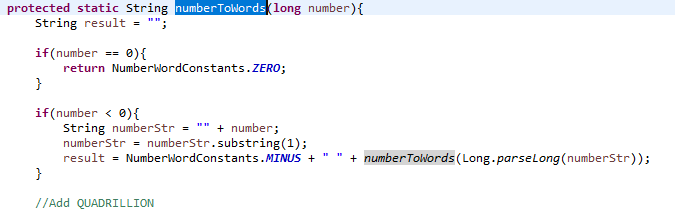
The ConverterFunctions class contains the static method convertNumToMoneyWords(String number). This calls the numberToWords(long number) method to get the number words, and then concatenates these with DOLLARS and CENTS to produce the final text for rendering to the web page:



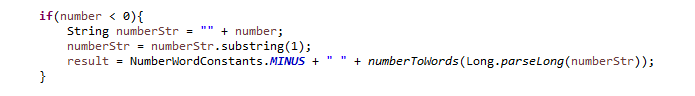
The convertNumbToMoneyWords(String number) method constructs the final result in the following format:

MINUS + *<dollar value>* + <DOLLAR or DOLLARS> + *<cent value>* + <CENT or CENTS> + POINT + *<fraction of cents as individual digits>*

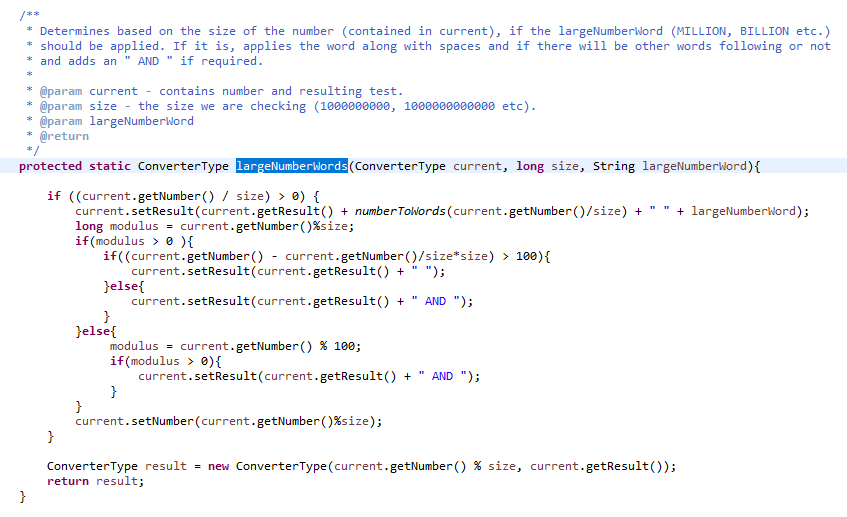
The convertNumbToMoneyWords(String number) method calls the numbToWords(long number) method, which converts the numbers to words for both the dollars and cents part:



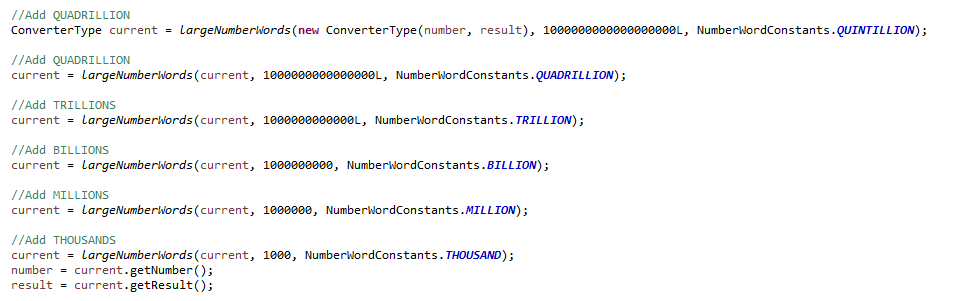
If the number is less than zero, the word MINUS starts off the final result:



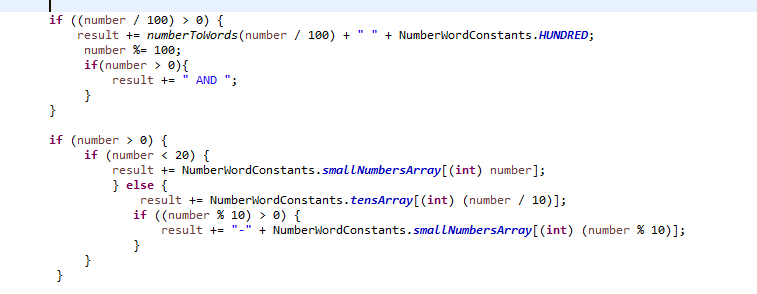
There is a specific method called largNumberWords() for handling large numbers like million, billion etc:



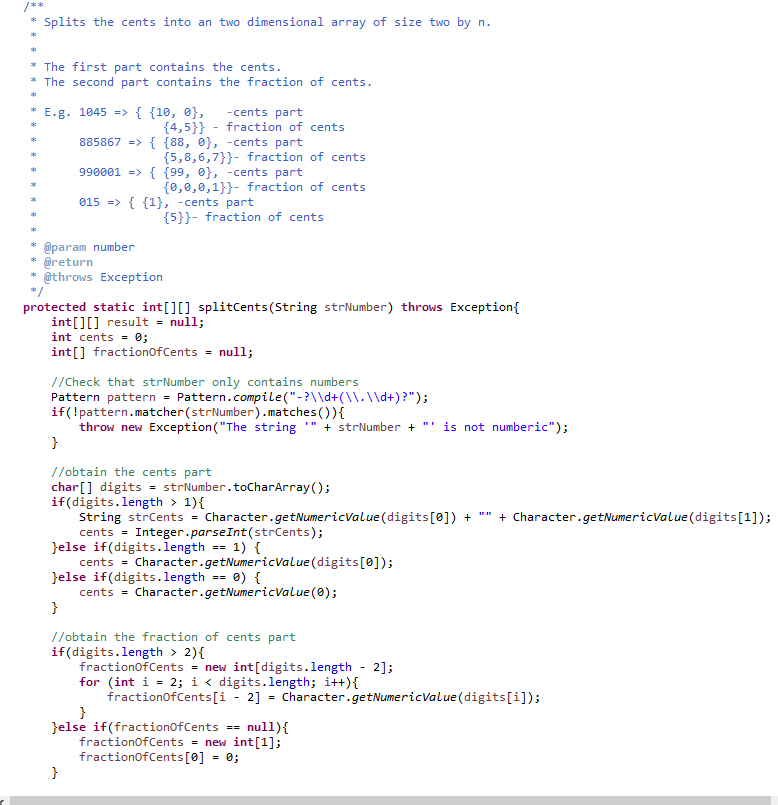
Each large number is handled by this function from THOUSAND up to QUINTILLION:



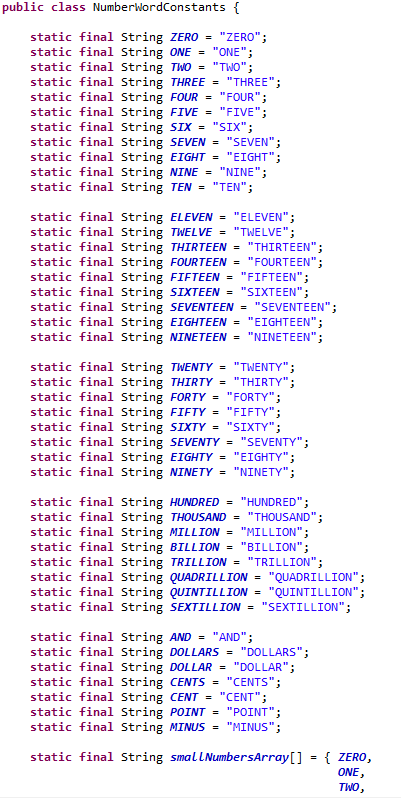
Smaller numbers are handled separately:



Cents are split into a two dimensional array with a cents part and a fraction of cents part:



There is a constants class which contains the actual text that is rendered to the result web page:



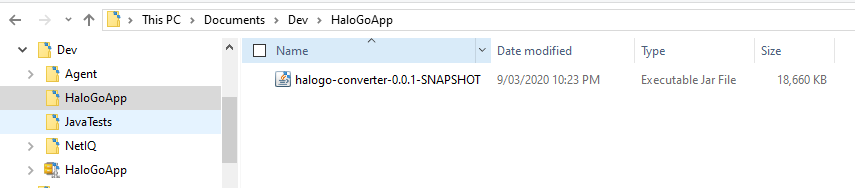
There are two simple HTML pages, one containing the form and the other for the result:



# Prepare the Environment

This solution comprises of an executable jar file which can be run using a JRE of version 1.8.0\_162 or similar, such as another JRE 1.8 release.

Firstly we need to extract the files from HaloGoApp.zip file into a directory:

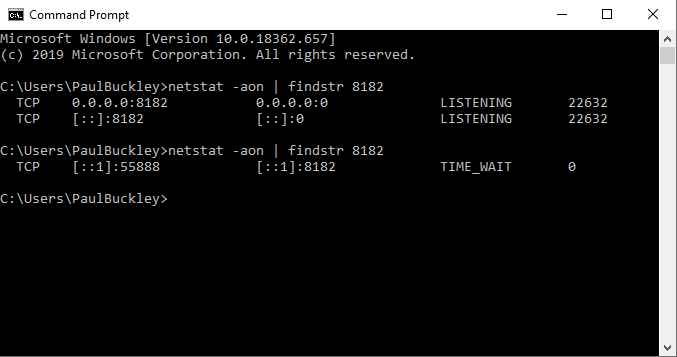


Open a DOS command window or a Linux terminal and then try to find a port that isn’t being used.

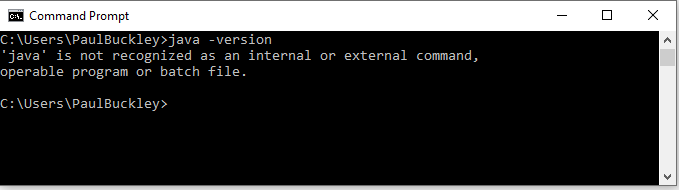
In DOS the command is:

**netstat -aon | findstr *<port\_number>***

If the port is being used, then either close down whatever is using the port or find another port:



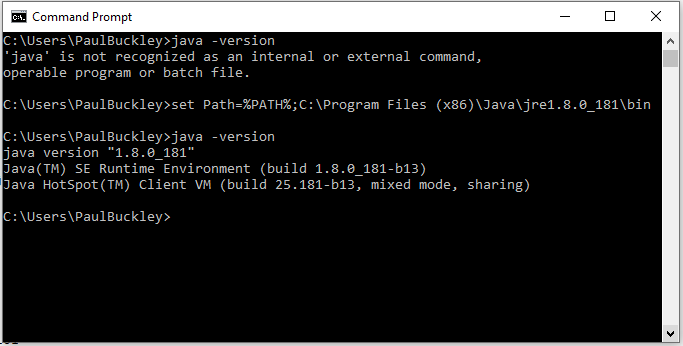
Next check if there if java is installed, configured and is the correct version:



If it is not, install a Java 8 JRE and put a reference to the bin directory in the DOS Path variable using a command like this:

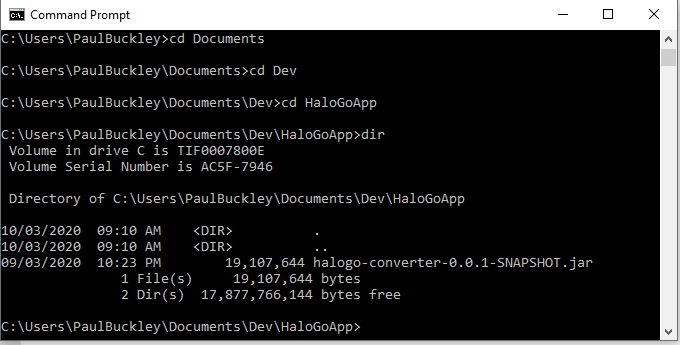
set Path=%PATH%;C:\Program Files (x86)\Java\jre1.8.0\_181\bin

Then check for the java version again to see that it is set up correctly:



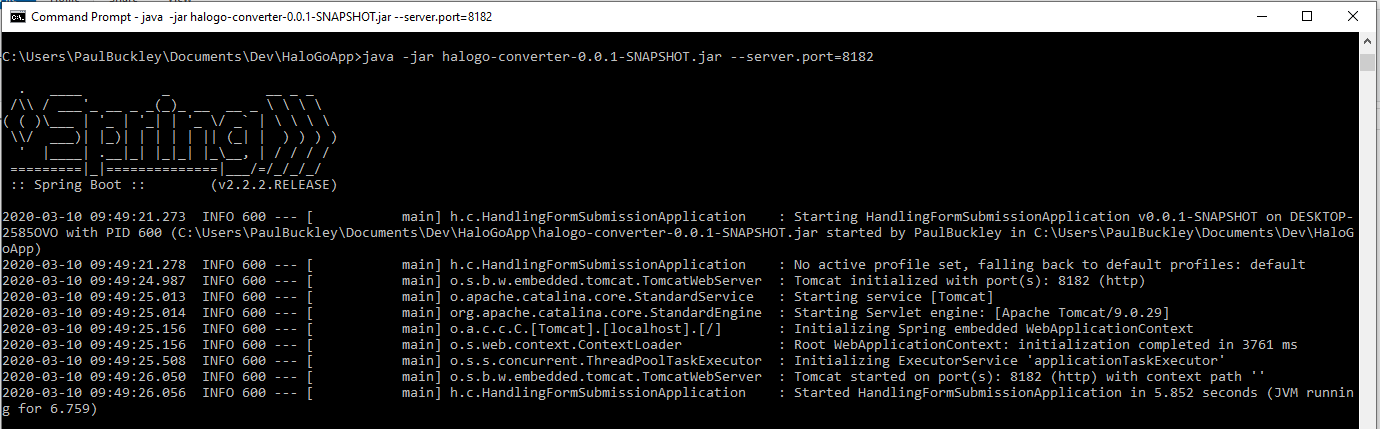
# Run the Application

Change directories to the directory containing the halogo-converter-0.0.1-SNAPSHOT.jar file:



Then execute the command:

java -jar halogo-converter-0.0.1-SNAPSHOT.jar --server.port=8182



If there are no errors in the startup and the final message is “Completed initialization in <number> ms” then open the application in a browser:

<http://localhost:8182/converter>

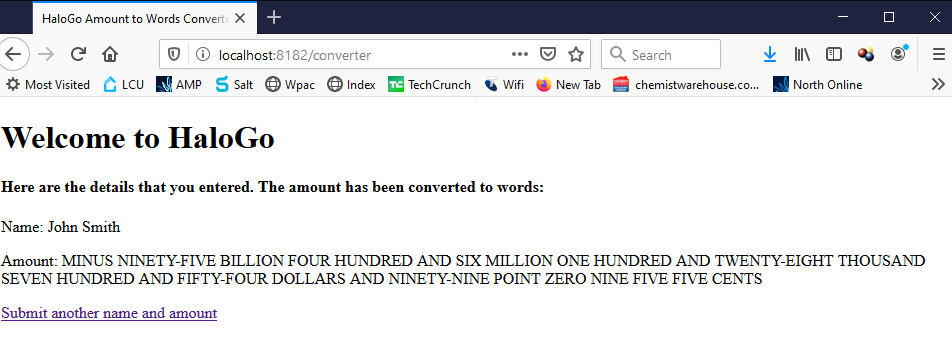
The result should be:



Enter a Name and an Amount and then press Submit:



Upon pressing the Submit button, the application will navigate to the next page where is displays the name as entered and the amount as words:



Clickink on the link at the bottom returns the user to the first page so that other values can be entered.