HaloGo Converter Deployment Guide

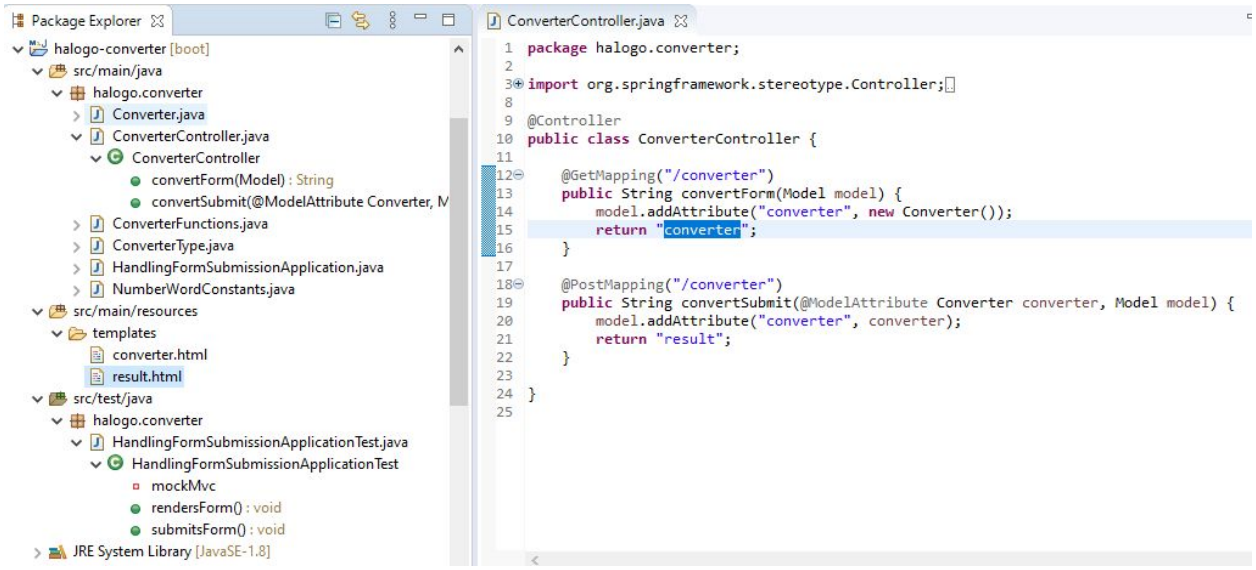
Overview

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.

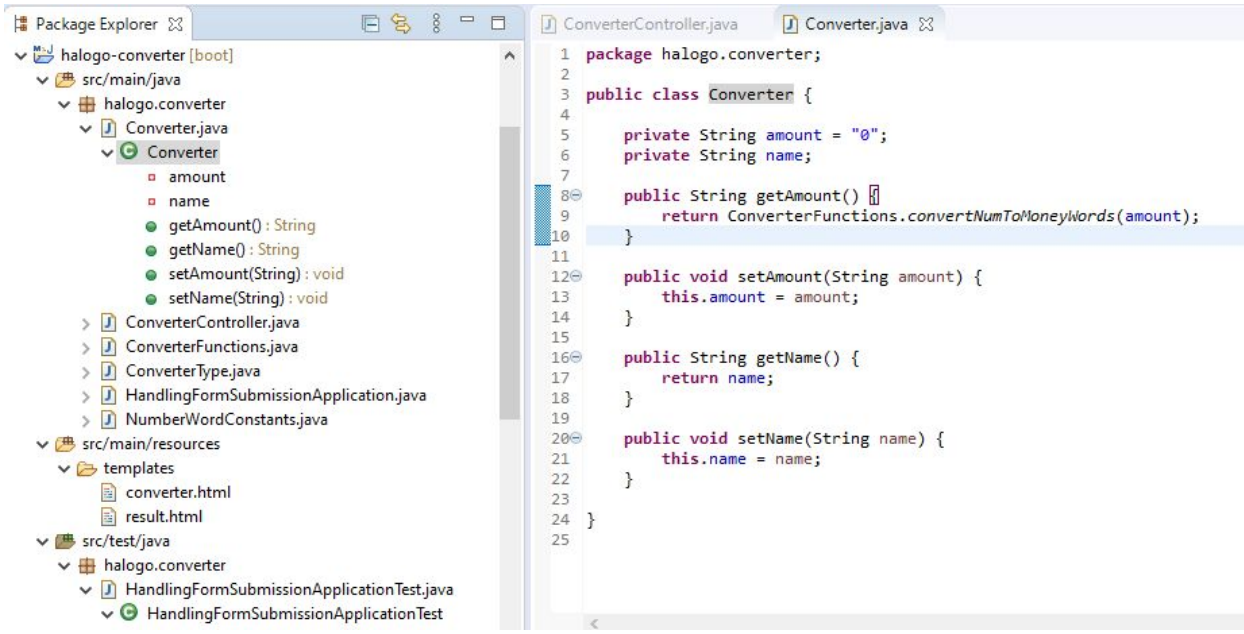
Design

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.

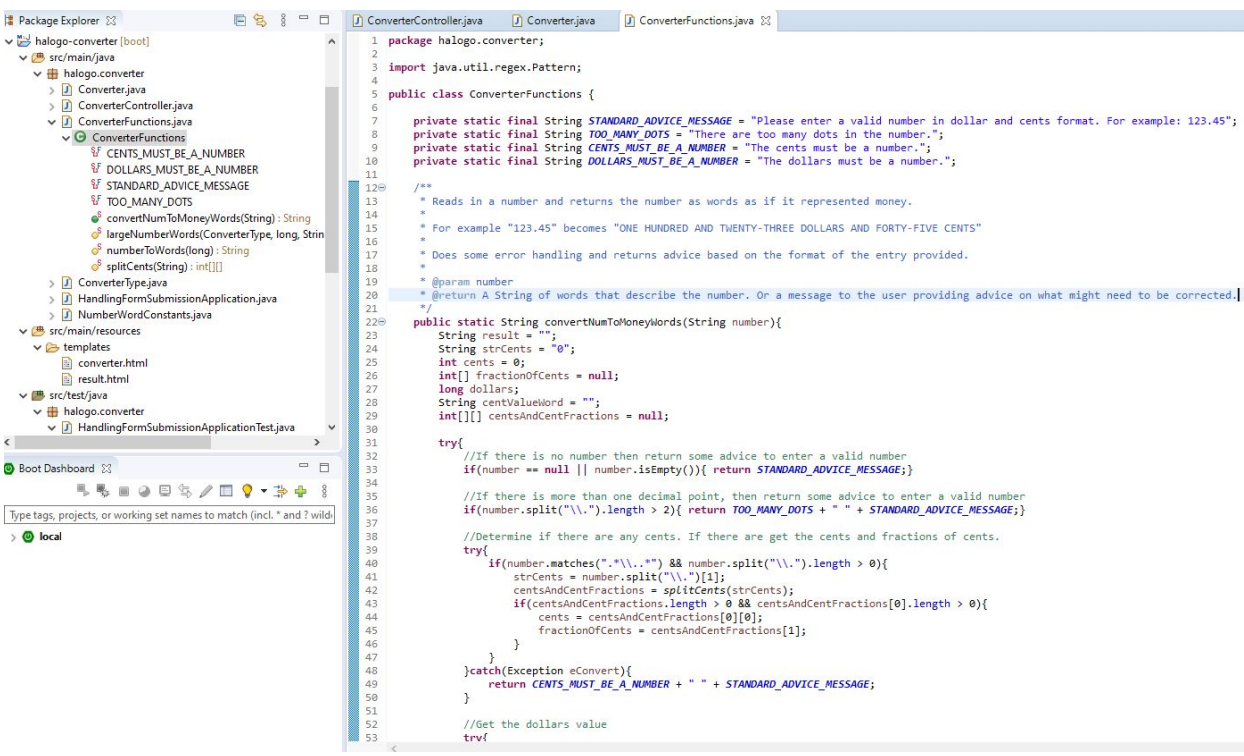
At the core, the ConverterController class uses the @GetMapping and @PostMapping annotations to map the HTTP GET and HTTP POST requests onto specific handler methods:



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:

```
protected static String numberToWords(long number){
    String result = "";

    if(number == 0){
        return NumberWordConstants.ZERO;
    }

    if(number < 0){
        String numberStr = "" + number;
        numberStr = numberStr.substring(1);
        result = NumberWordConstants.MINUS + " " + numberToWords(Long.parseLong(numberStr));
    }

    //Add QUADRILLION
```

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

```
if(number < 0){
    String numberStr = "" + number;
    numberStr = numberStr.substring(1);
    result = NumberWordConstants.MINUS + " " + numberToWords(Long.parseLong(numberStr));
}
```

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

```java
/**
 * Determines based on the size of the number (contained in current), if the largeNumberWord (MILLION, BILLION etc.)
 * should be applied. If it is, applies the word along with spaces and if there will be other words following or not
 * and adds an " AND " if required.
 *
 * @param current - contains number and resulting test.
 * @param size - the size we are checking (1000000000, 1000000000000 etc).
 * @param largeNumberWord
 * @return
 */
protected static ConverterType largeNumberWords(ConverterType current, long size, String largeNumberWord){

    if ((current.getNumber() / size) > 0) {
        current.setResult(current.getResult() + numberToWords(current.getNumber()/size) + " " + largeNumberWord);
        long modulus = current.getNumber()%size;
        if(modulus > 0 ){
            if((current.getNumber() - current.getNumber()/size*size) > 100){
                current.setResult(current.getResult() + " ");
            }else{
                current.setResult(current.getResult() + " AND ");
            }
        }else{
             modulus = current.getNumber() % 100;
             if(modulus > 0){
                 current.setResult(current.getResult() + " AND ");
             }
        }
        current.setNumber(current.getNumber()%size);
    }

    ConverterType result = new ConverterType(current.getNumber() % size, current.getResult());
    return result;
}
```

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

```java
//Add QUADRILLION
ConverterType current = largeNumberWords(new ConverterType(number, result), 1000000000000000000L, NumberWordConstants.QUINTILLION);

//Add QUADRILLION
current = largeNumberWords(current, 1000000000000000L, NumberWordConstants.QUADRILLION);

//Add TRILLIONS
current = largeNumberWords(current, 1000000000000L, NumberWordConstants.TRILLION);

//Add BILLIONS
current = largeNumberWords(current, 1000000000, NumberWordConstants.BILLION);

//Add MILLIONS
current = largeNumberWords(current, 1000000, NumberWordConstants.MILLION);

//Add THOUSANDS
current = largeNumberWords(current, 1000, NumberWordConstants.THOUSAND);
number = current.getNumber();
result = current.getResult();
```

Smaller numbers are handled separately:

```java
if ((number / 100) > 0) {
    result += numberToWords(number / 100) + " " + NumberWordConstants.HUNDRED;
     number %= 100;
     if(number > 0){
         result += " AND ";
     }
}

if (number > 0) {
     if (number < 20) {
         result += NumberWordConstants.smallNumbersArray[(int) number];
         } else {
             result += NumberWordConstants.tensArray[(int) (number / 10)];
            if ((number % 10) > 0) {
                result += "-" + NumberWordConstants.smallNumbersArray[(int) (number % 10)];
            }
     }
 }
```

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

```java
/**
 * Splits the cents into an two dimensional array of size two by n.
 *
 *
 * The first part contains the cents.
 * The second part contains the fraction of cents.
 *
 * E.g. 1045 => { {10, 0},   -cents part
 *                  {4,5}} - fraction of cents
 *      885867 => { {88, 0}, -cents part
 *                  {5,8,6,7}}- fraction of cents
 *      990001 => { {99, 0}, -cents part
 *                  {0,0,0,1}}- fraction of cents
 *      015 => { {1}, -cents part
 *                  {5}}- fraction of cents
 *
 * @param number
 * @return
 * @throws Exception
 */
protected static int[][] splitCents(String strNumber) throws Exception{
    int[][] result = null;
    int cents = 0;
    int[] fractionOfCents = null;

    //Check that strNumber only contains numbers
    Pattern pattern = Pattern.compile("-?\\d+(\\.\\d+)?");
    if(!pattern.matcher(strNumber).matches()){
        throw new Exception("The string '" + strNumber + "' is not numberic");
    }

    //obtain the cents part
    char[] digits = strNumber.toCharArray();
    if(digits.length > 1){
        String strCents = Character.getNumericValue(digits[0]) + "" + Character.getNumericValue(digits[1]);
        cents = Integer.parseInt(strCents);
    }else if(digits.length == 1) {
        cents = Character.getNumericValue(digits[0]);
    }else if(digits.length == 0) {
        cents = Character.getNumericValue(0);
    }

    //obtain the fraction of cents part
    if(digits.length > 2){
        fractionOfCents = new int[digits.length - 2];
        for (int i = 2; i < digits.length; i++){
            fractionOfCents[i - 2] = Character.getNumericValue(digits[i]);
        }
    }else if(fractionOfCents == null){
        fractionOfCents = new int[1];
        fractionOfCents[0] = 0;
    }
```

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

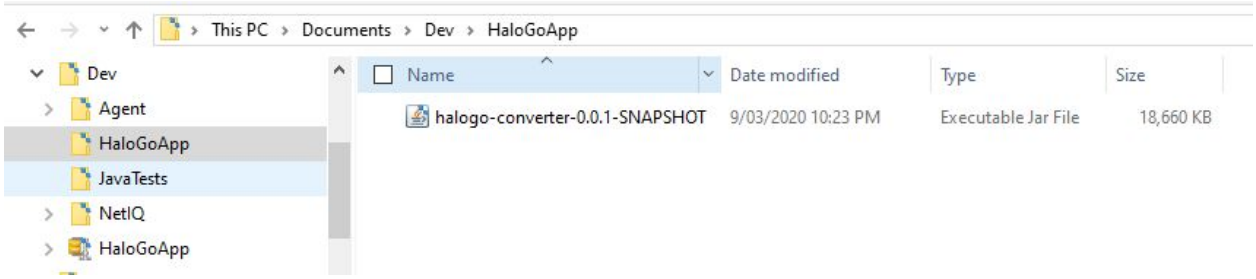
```
public class NumberWordConstants {

    static final String ZERO = "ZERO";
    static final String ONE = "ONE";
    static final String TWO = "TWO";
    static final String THREE = "THREE";
    static final String FOUR = "FOUR";
    static final String FIVE = "FIVE";
    static final String SIX = "SIX";
    static final String SEVEN = "SEVEN";
    static final String EIGHT = "EIGHT";
    static final String NINE = "NINE";
    static final String TEN = "TEN";

    static final String ELEVEN = "ELEVEN";
    static final String TWELVE = "TWELVE";
    static final String THIRTEEN = "THIRTEEN";
    static final String FOURTEEN = "FOURTEEN";
    static final String FIFTEEN = "FIFTEEN";
    static final String SIXTEEN = "SIXTEEN";
    static final String SEVENTEEN = "SEVENTEEN";
    static final String EIGHTEEN = "EIGHTEEN";
    static final String NINETEEN = "NINETEEN";

    static final String TWENTY = "TWENTY";
    static final String THIRTY = "THIRTY";
    static final String FORTY = "FORTY";
    static final String FIFTY = "FIFTY";
    static final String SIXTY = "SIXTY";
    static final String SEVENTY = "SEVENTY";
    static final String EIGHTY = "EIGHTY";
    static final String NINETY = "NINETY";

    static final String HUNDRED = "HUNDRED";
    static final String THOUSAND = "THOUSAND";
    static final String MILLION = "MILLION";
    static final String BILLION = "BILLION";
    static final String TRILLION = "TRILLION";
    static final String QUADRILLION = "QUADRILLION";
    static final String QUINTILLION = "QUINTILLION";
    static final String SEXTILLION = "SEXTILLION";

    static final String AND = "AND";
    static final String DOLLARS = "DOLLARS";
    static final String DOLLAR = "DOLLAR";
    static final String CENTS = "CENTS";
    static final String CENT = "CENT";
    static final String POINT = "POINT";
    static final String MINUS = "MINUS";

    static final String smallNumbersArray[] = { ZERO,
                                                ONE,
                                                TWO,
```

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:

```
Command Prompt
Microsoft Windows [Version 10.0.18362.657]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\PaulBuckley>netstat -aon | findstr 8182
  TCP    0.0.0.0:8182           0.0.0.0:0              LISTENING       22632
  TCP    [::]:8182              [::]:0                 LISTENING       22632

C:\Users\PaulBuckley>netstat -aon | findstr 8182
  TCP    [::1]:55888            [::1]:8182             TIME_WAIT       0

C:\Users\PaulBuckley>
```

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

```
Command Prompt
C:\Users\PaulBuckley>java -version
'java' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\PaulBuckley>
```

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:

```
Command Prompt
C:\Users\PaulBuckley>java -version
'java' is not recognized as an internal or external command,
operable program or batch file.

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

C:\Users\PaulBuckley>java -version
java version "1.8.0_181"
Java(TM) SE Runtime Environment (build 1.8.0_181-b13)
Java HotSpot(TM) Client VM (build 25.181-b13, mixed mode, sharing)

C:\Users\PaulBuckley>
```

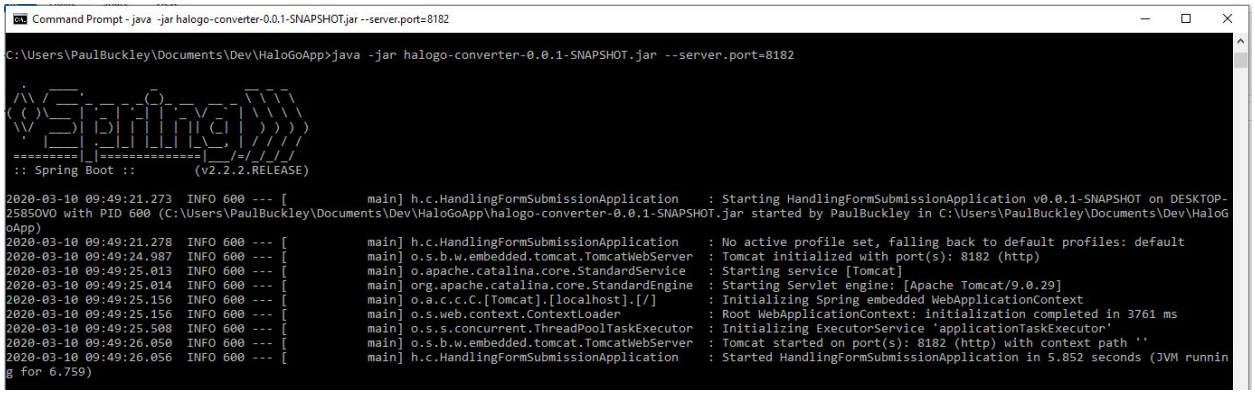
Run the Application

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

```
Command Prompt
C:\Users\PaulBuckley>cd Documents

C:\Users\PaulBuckley\Documents>cd Dev

C:\Users\PaulBuckley\Documents\Dev>cd HaloGoApp

C:\Users\PaulBuckley\Documents\Dev\HaloGoApp>dir
 Volume in drive C is TIF0007800E
 Volume Serial Number is AC5F-7946

 Directory of C:\Users\PaulBuckley\Documents\Dev\HaloGoApp

10/03/2020  09:10 AM    <DIR>          .
10/03/2020  09:10 AM    <DIR>          ..
09/03/2020  10:23 PM        19,107,644 halogo-converter-0.0.1-SNAPSHOT.jar
               1 File(s)     19,107,644 bytes
               2 Dir(s)  17,877,766,144 bytes free

C:\Users\PaulBuckley\Documents\Dev\HaloGoApp>
```

Then execute the command:

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

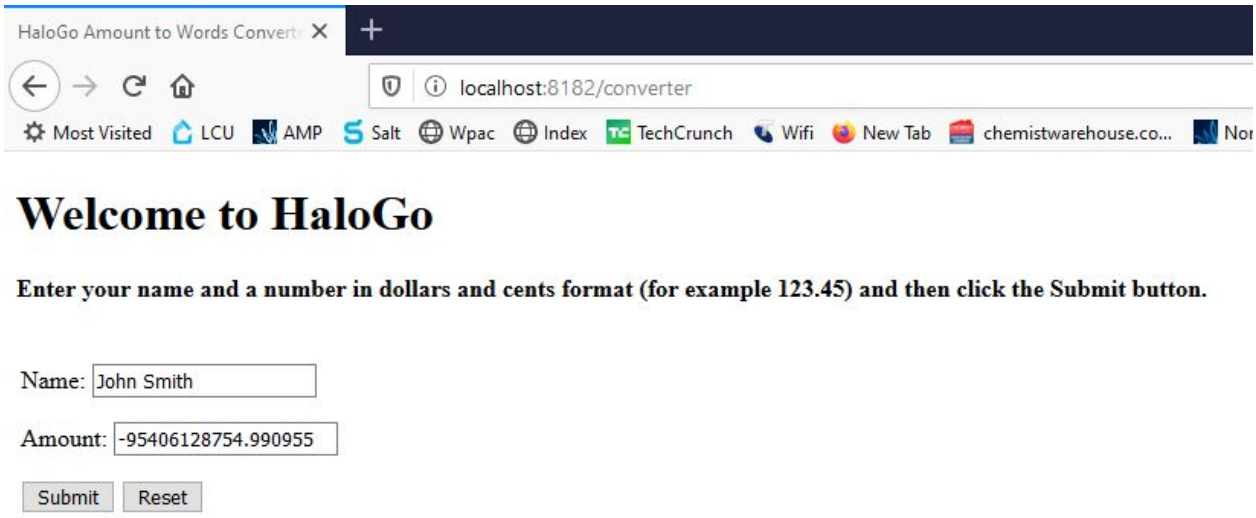


If there are no errors in the startup and the final message is “Completed initialization in <number> ms” then open the application in a browser:

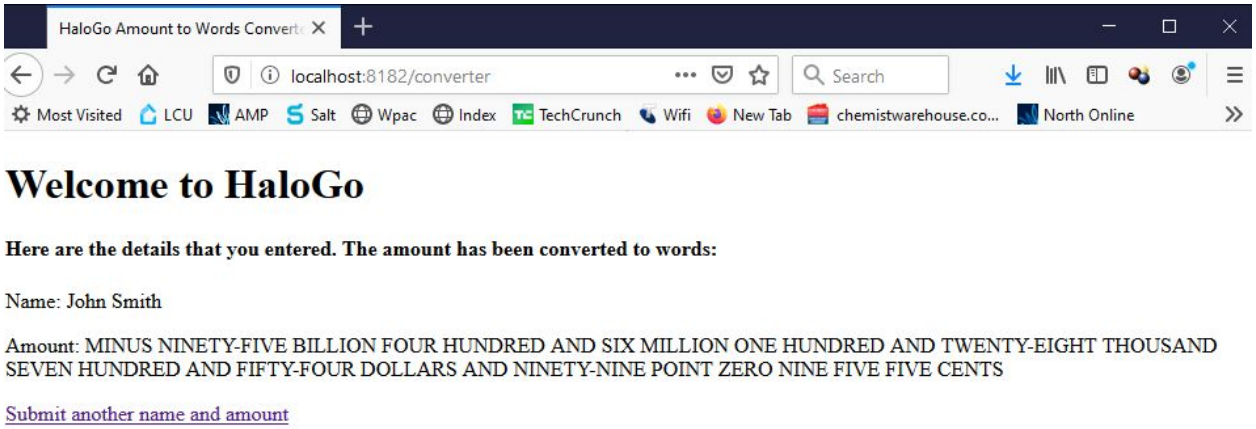
http://localhost:8182/converter

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.