**Trademe Packaging Solutions**

Assumptions:

1. Package will be calculated in cubic (Length X Breadth X Height)
2. Package should not exceed 25 kilograms
3. Package should not exceed the maximum dimension of the Largest Package
4. To easily identify the Parcel Type, I need to calculate the dimensions of the parcel in cubic

For example for Small Package (210X280X130):

The dimensions in cubic will be 7,644,000mm cubic

If the input Dimensions is (190X300X110)

The dimensions in cubic will be 5,700,000mm cubic; although it exceeded the Breadth it is still classified is Small Package because it does not exceeded the Dimensions in cubic of the small Package.

Inputs:

1. Dimension (Length, Breadth, Height) of the package
2. Weight of the Package
3. Quantity of Package

Output:

1. Package Type
2. Cost of the Package
3. Total Price

Solution:

Created a two table using Entity Framework.

1. tblParcelDimensionConfig – This table is used to store the dimensions for each package type. This will be useful for future configuration. Let’s say the company will now offer an Extra Large Package
2. tblWeightConfig – This table is used store the Maximum weight configuration. Although in the future, this can be directly related to the Parcel Dimension by adding a new table that each Package Type will have a configuration weight and base on the weight you can put an additional cost for it. For example for every excess weight in “Small Package” from the 25 kilos maximum. An additional $1 will be added.

**Future Enhancement:**

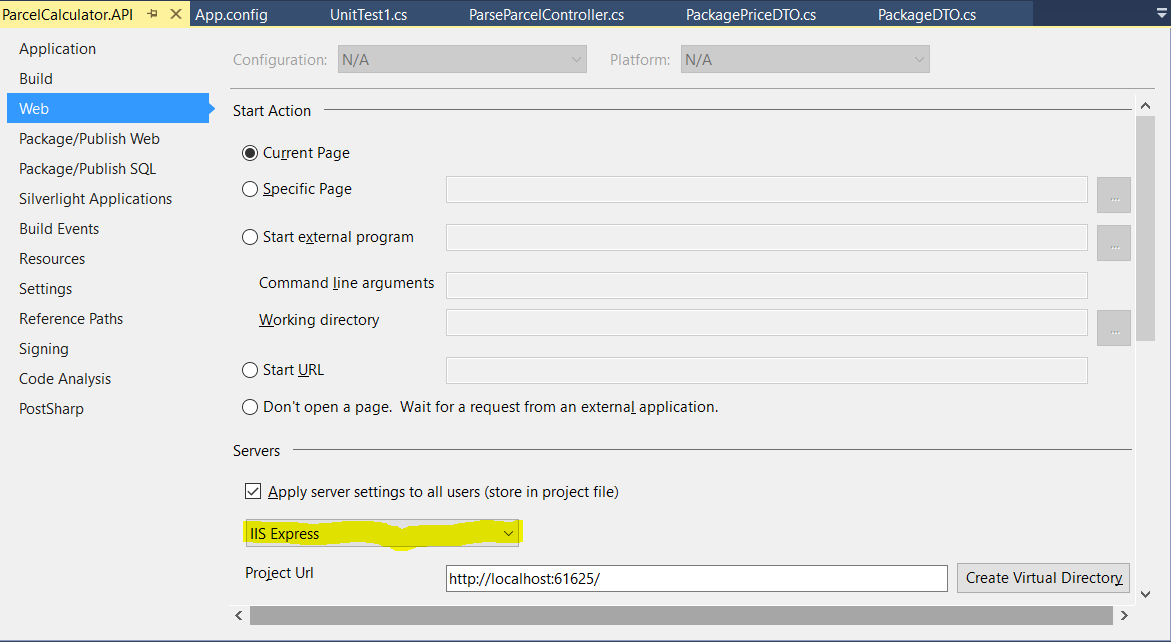
For future enhancement, you can add a table configuration that will define the Package and with unlimited weight the cost will be $200 per package for “**Small Package**”, $250 for “**Medium Package**” and $300 for “**Large Package**”

**The Package Parser consist of a three projects**

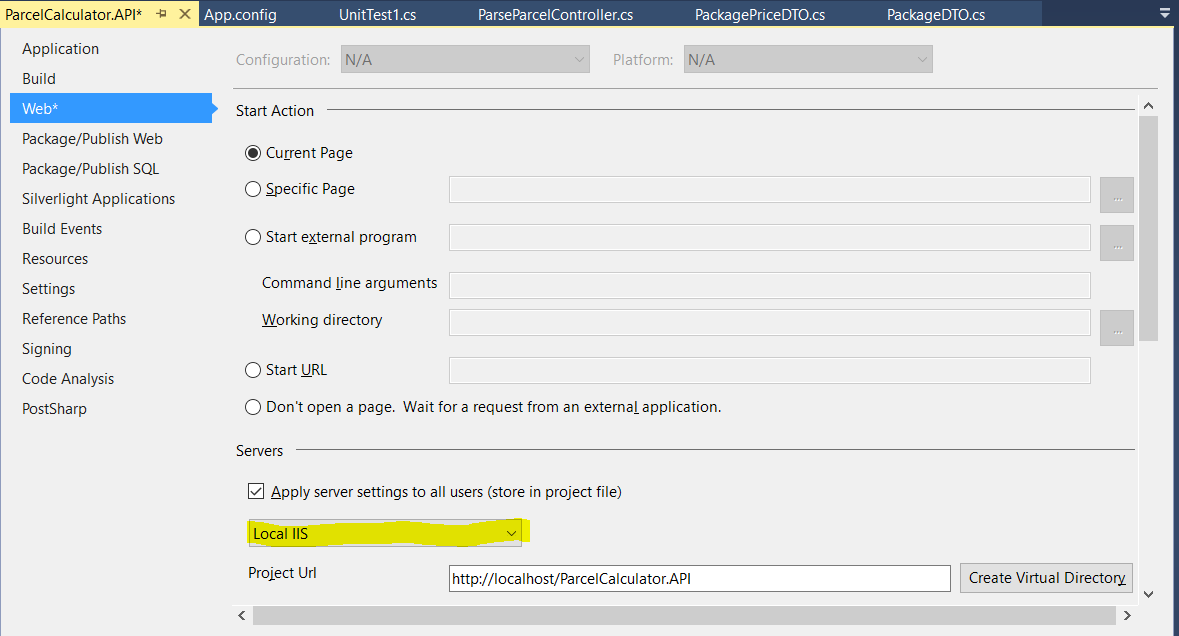
1. ParcelCalculator.API – This project is a Web API service that will connect to the database to return the Package Type and Cost of the Parcel. This project was called by the “ParcelCalculator” project which is the client interface so to be able for the dependency project to work. The following configuration needs to be done.

**Configuration**:

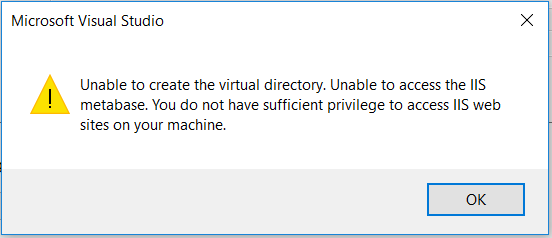
1. Right click on the project click “**Properties**”, Click the “**Web**” tab. See below:



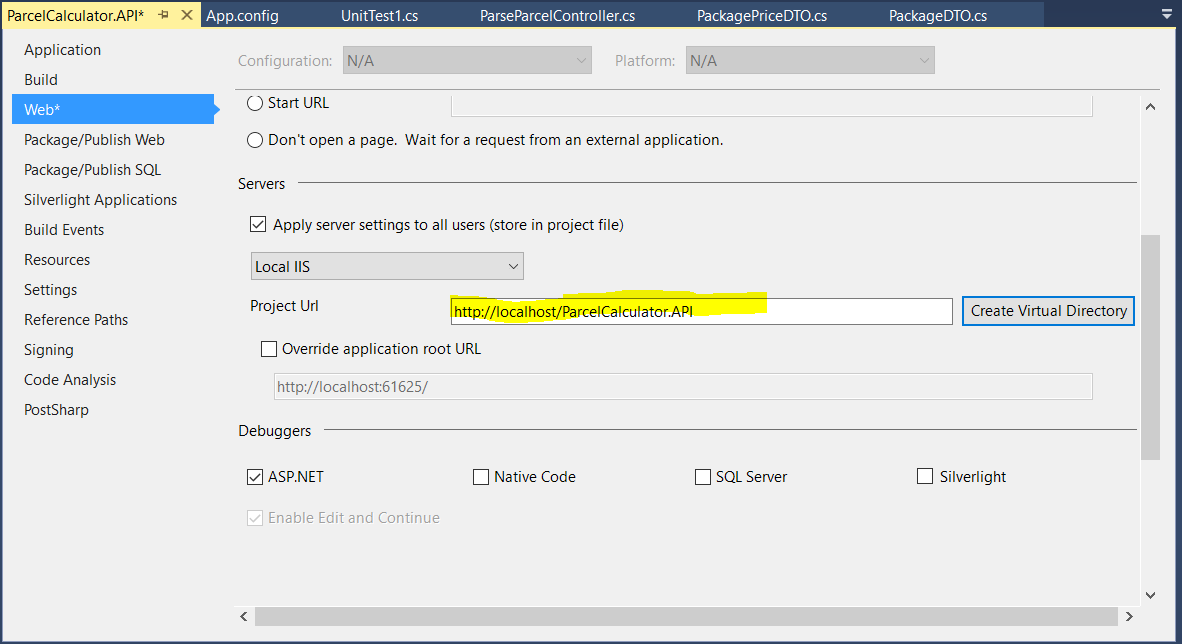
1. Change the “**IIS Express**” into “**Local IIS**” as shown below:



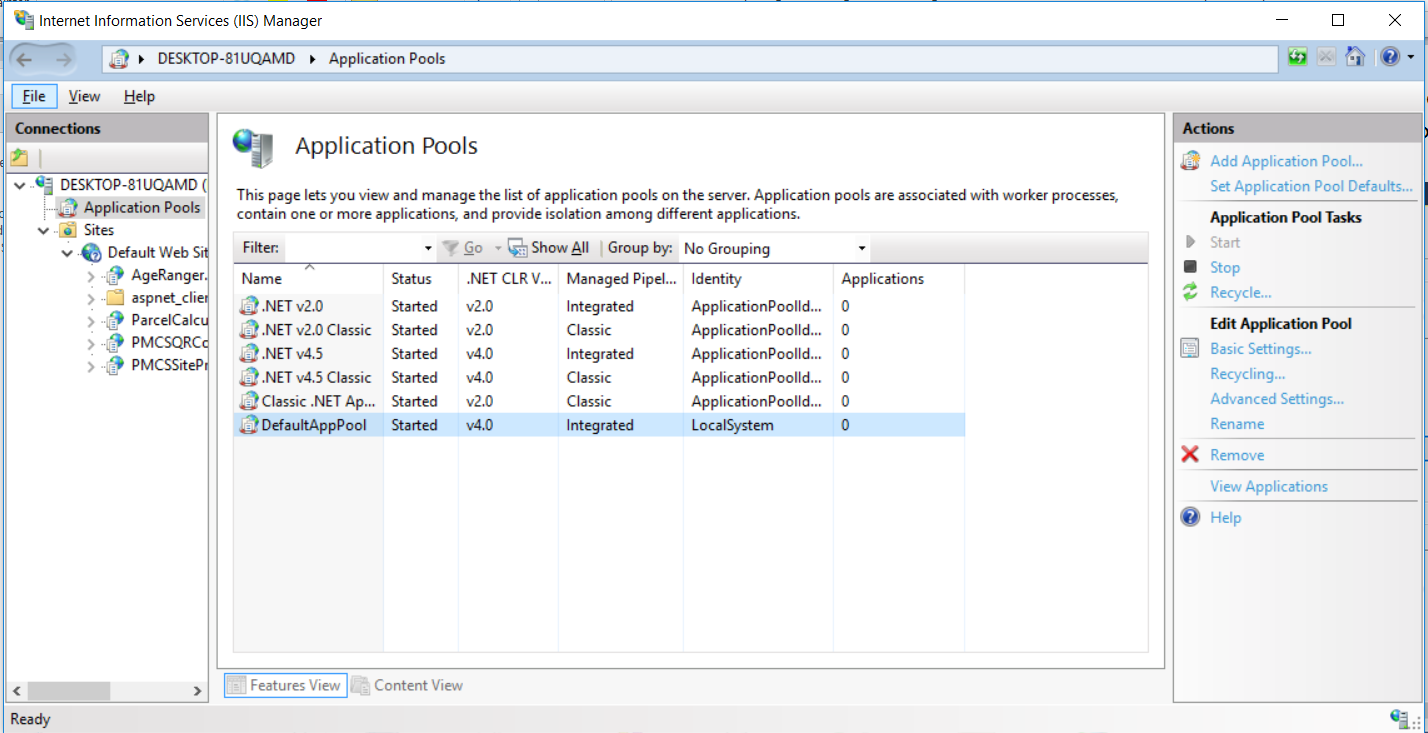
1. Then click “**Create Virtual Directory**” to create local directory in the **“Local IIS”.** Please take note that you should have an administrative privileges to Create the Virtual Directory otherwise it will display you the following message like the one below:



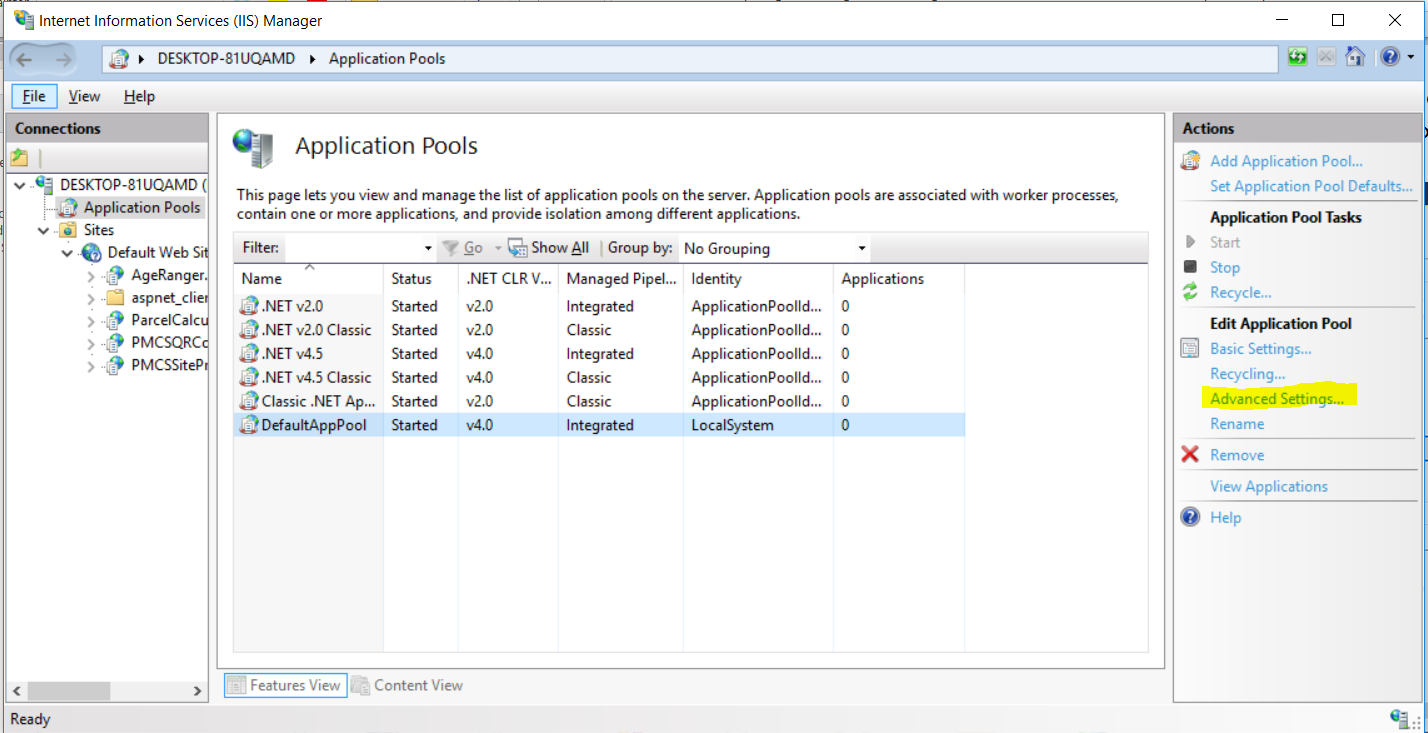
To do this, right click on the “**Visual Studio**” and Run as Administrator then open the Project Solution. Then, after that you should now have the permission to create the virtual directory like the one below:



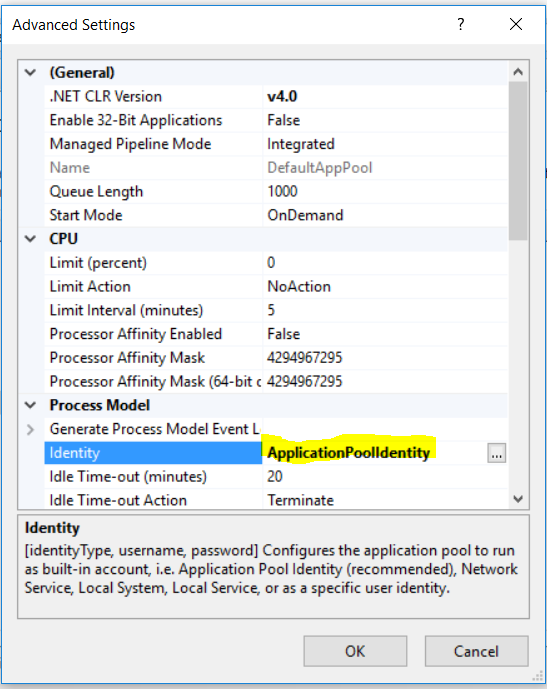
1. In the IIS Manager, Click “**Application Pools**”

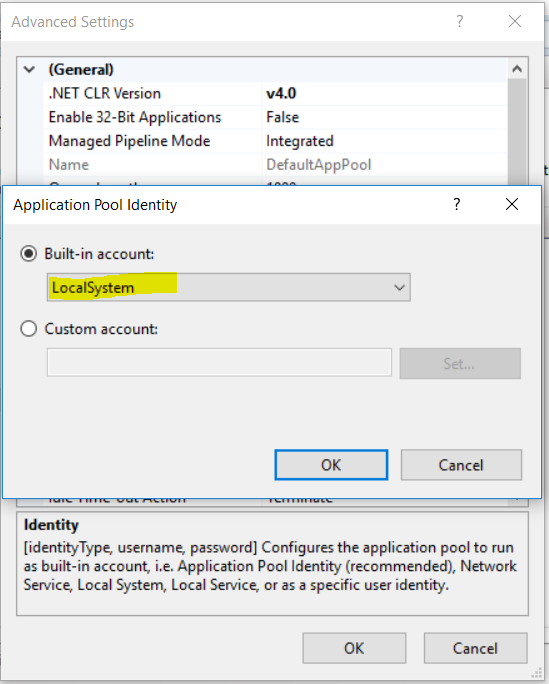


1. Select “**DefaultAppPool**” and click “**Advanced Settings**”. See below

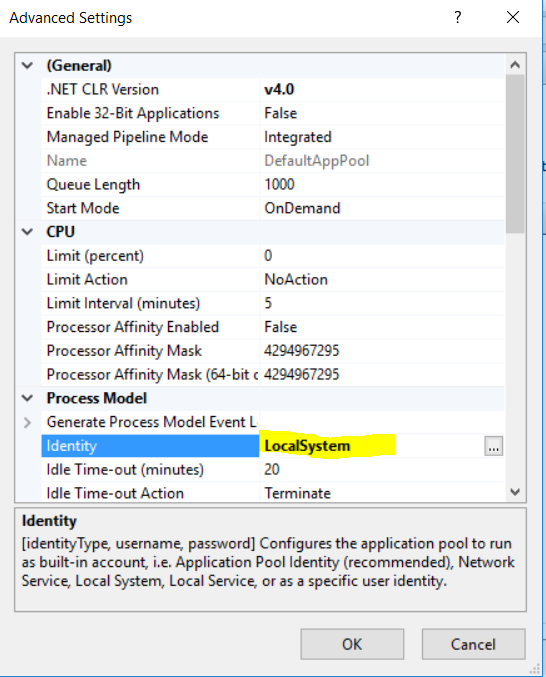


1. Under “**Identify**”, change “**ApplicationPoolIdentity**” to “**LocalSystem**”. See below:





1. Then click “**OK**” button. The new identity should look like the one below:

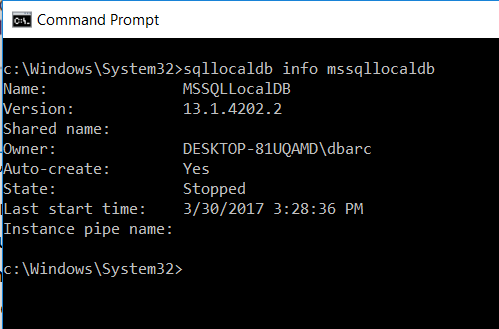


1. Then click “**OK**” button to close the dialog.
2. SQLLocalDB is required in this project. So if you have’nt installed the SQLLocalDB, Please install first. You can download it in the below link:

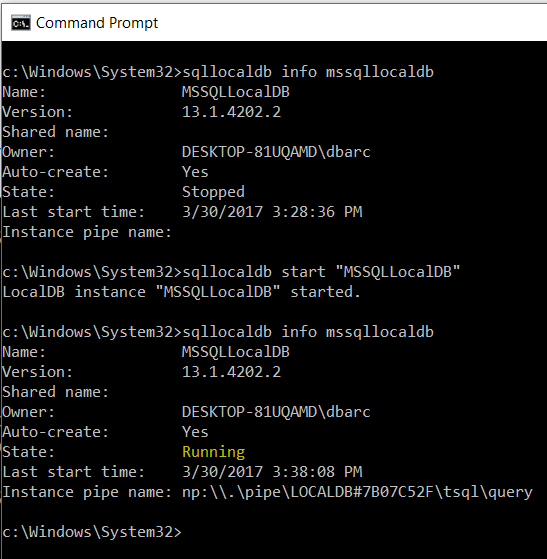
https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/sql-server-2016-express-localdb

1. To check the instance name of SQLLocalDB you can run the below command in cmd prompt

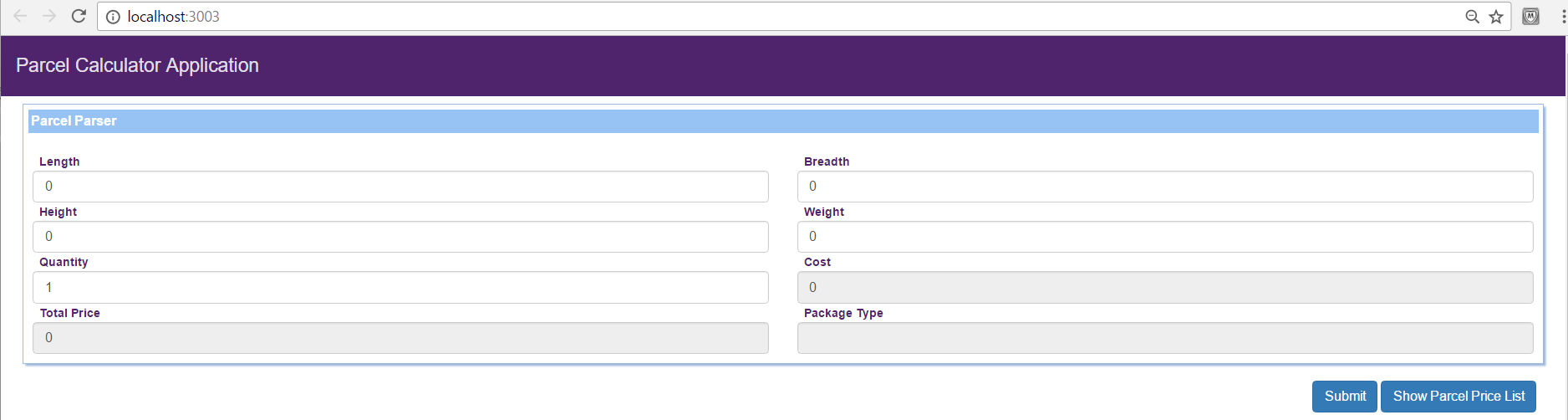
“**sqllocaldb info mssqllocaldb”** and you can see the below response.



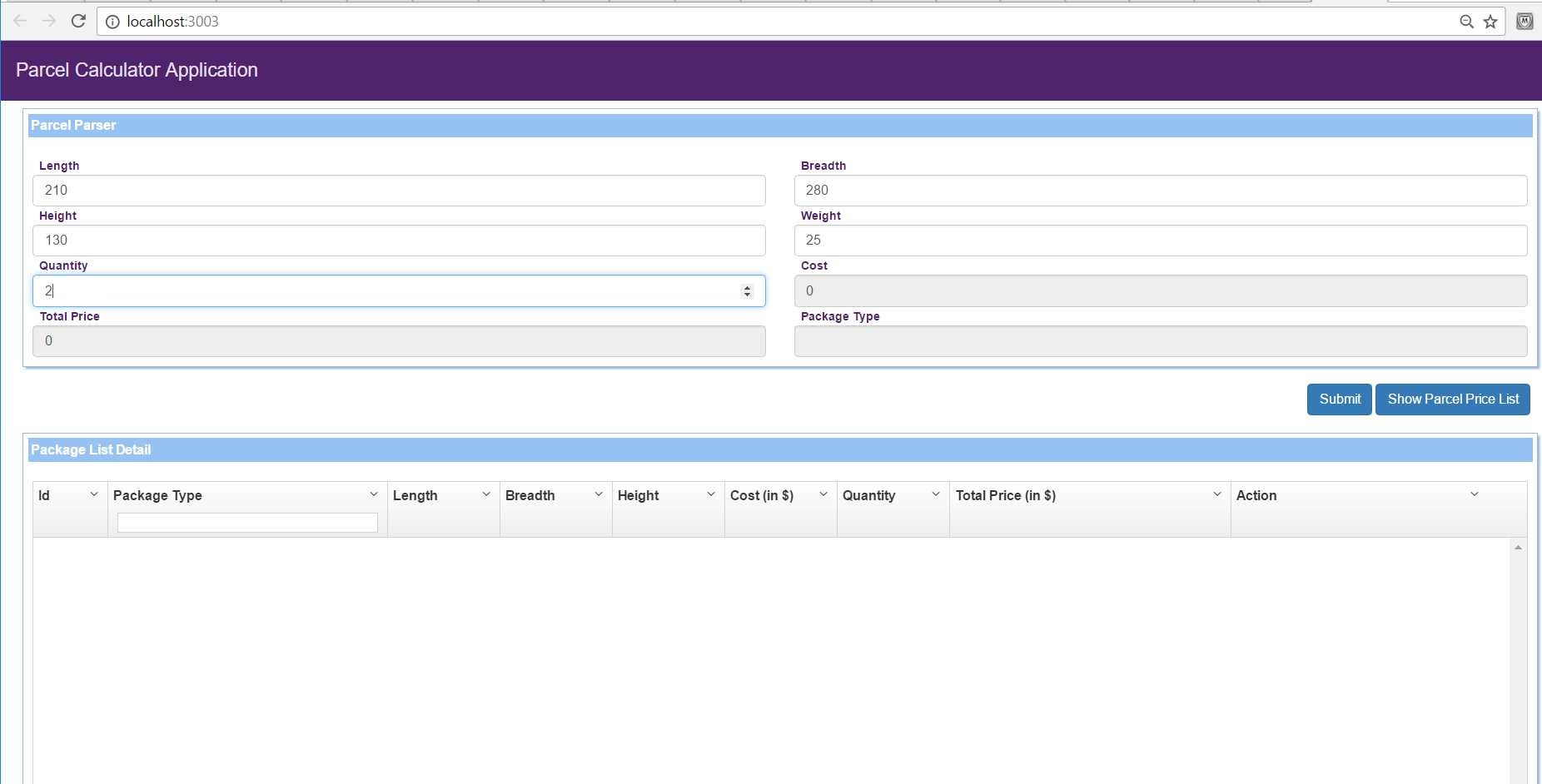
1. The current state of the sqllocaldb is “**Stopped**”. To start it you need to run the command “sqllocaldb start [InstanceName]”. In this case “**sqllocaldb start MSSQLLocalDB**”. Running again the previous step will show the information of sqllocaldb. See below:



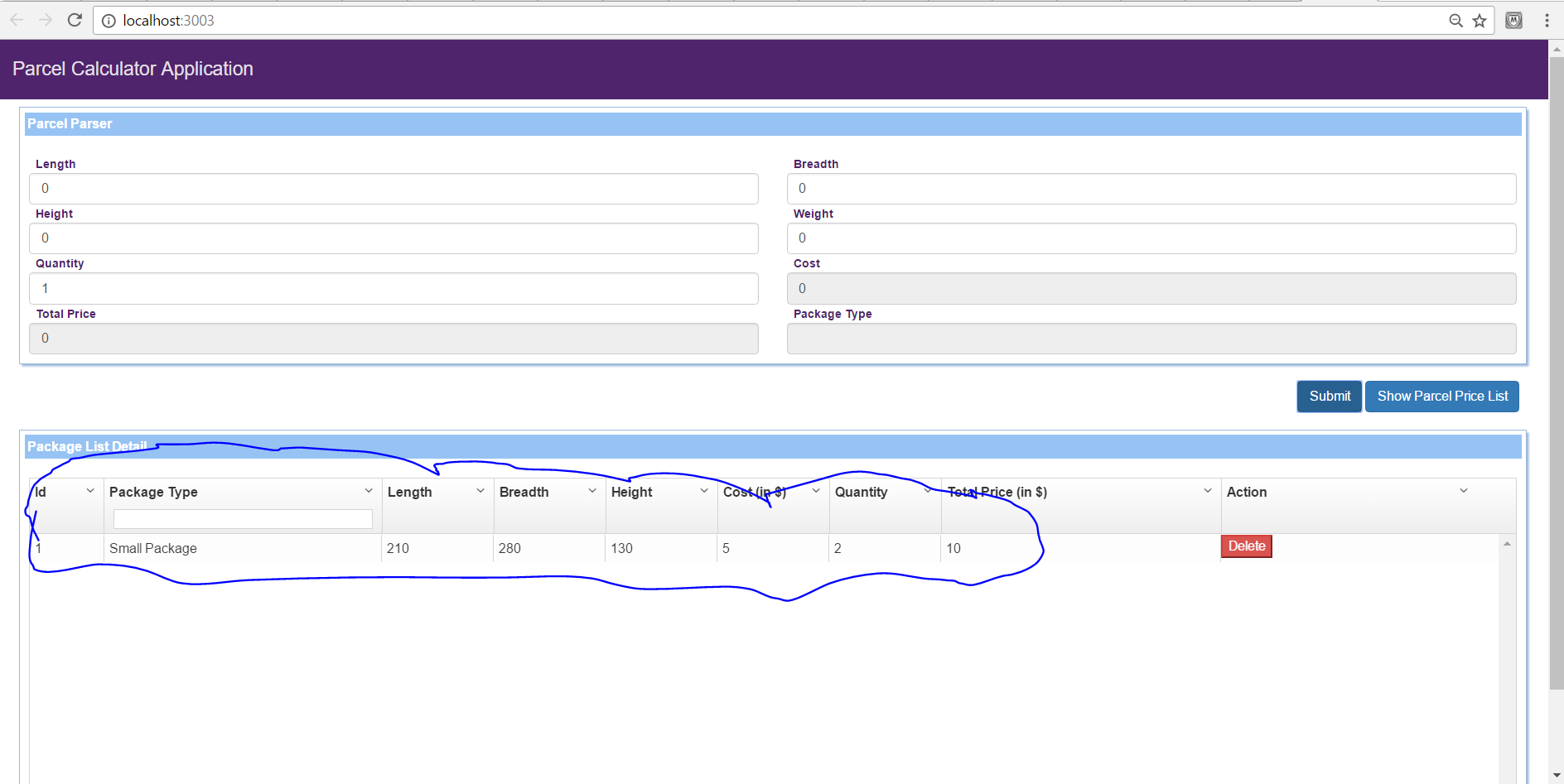
1. ParcelCalculator – This project is the client interface to enter the parameters to parse the parcel and it will return the equivalent Package Type. See below Screen shot:



The user needs to input the Dimensions of the Parcel (Length, Breadth, and Height), Weight and an additional input was added which quantity. Upon clicking “**Submit**”, it will return the **Package Type, Cost and Total Price**. See below:

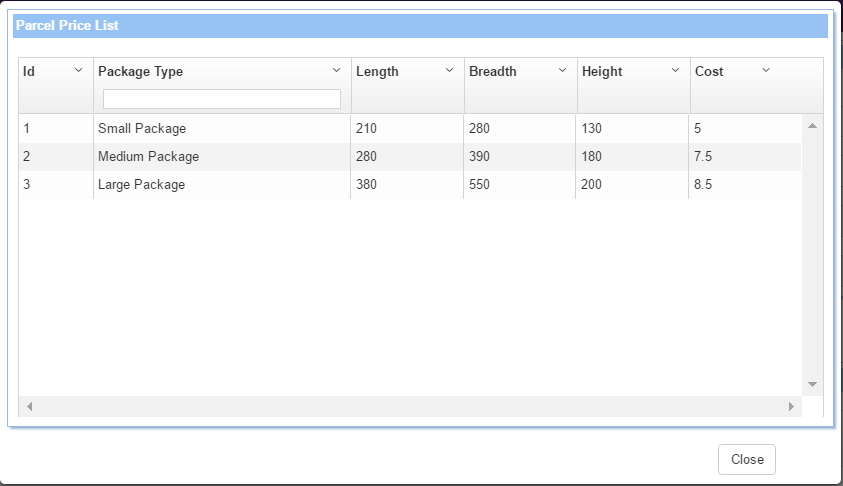


After clicking “**Submit**” button, it will add the data in the grid below the buttons. See snap shot:



It displayed the result in the below grid based on the entered parameter in the above screen.

Clicking in the button “Show Parcel Price List”, will open a modal window displaying the Price list. See below:



**Configuration:**

The system was developed using HTML5, CSS3, AngularJS, AngularMaterial, AngularUIGrid.

To able to run the project I used npm, node.js, bower, gulp

1. Please download first the npm and node.js, GIT shell
2. I used npm to install bower and gulp by entering the command below in git console

npm install –g bower

npm install –g gulp

npm install – install the gulp dependency in package.json

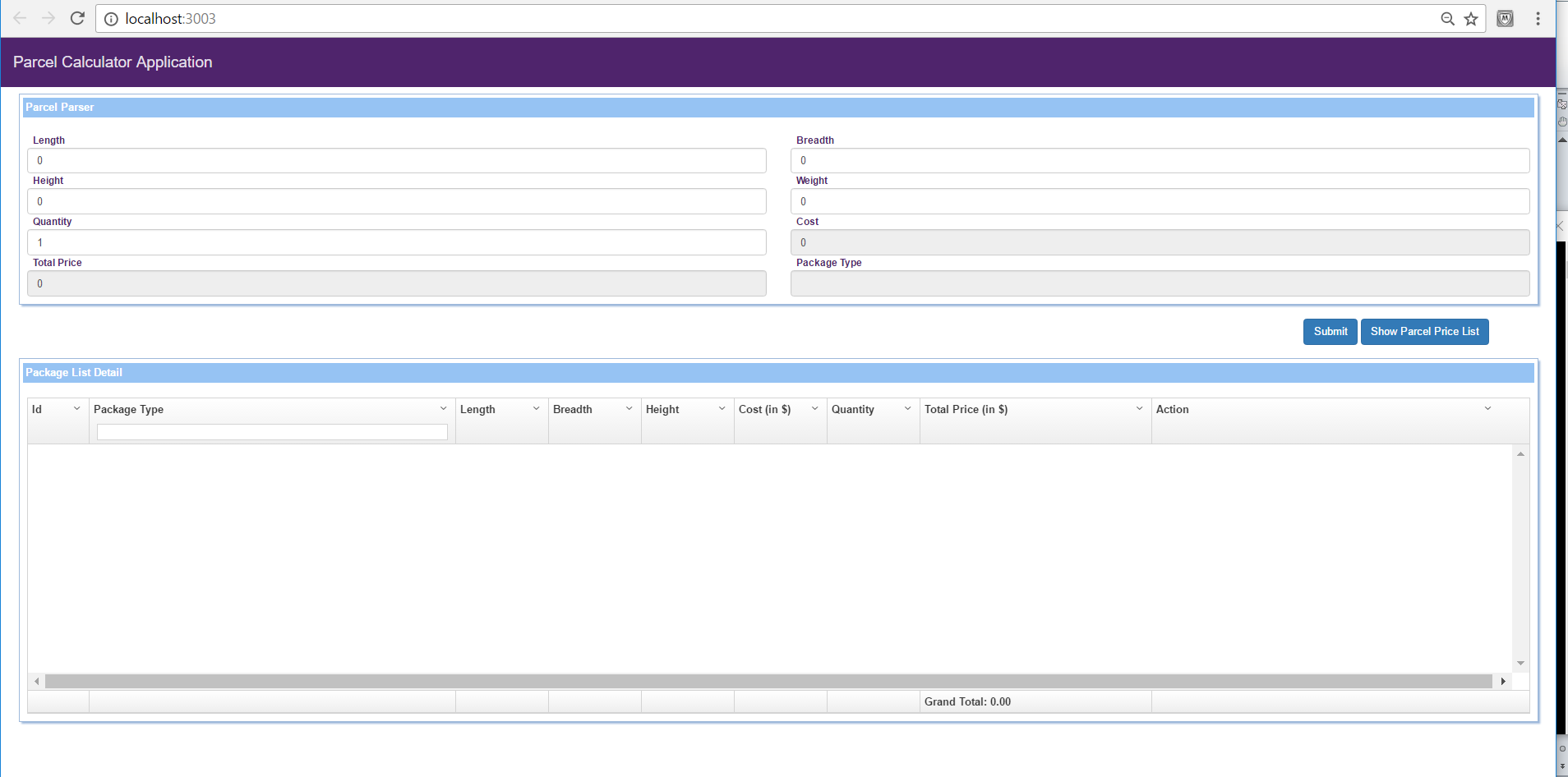
1. After downloading the said tools, I used “bower” to download the components that I define in “bower.json” that was needed in the project by entering the command below:

bower install

1. I used “gulp” to run the client interface in node.js server by entering the command below:

gulp serve

1. By entering “gulp serve”, a browser will be open like the one below:



1. ProjectCalculator.API.Tests – This project is just a unit testing for the ProjectCalculator.API