# Car Manufacturer Service

**Files for This Project**
CarMfg_jar.zip – The executable jar file
    Inside this zip file:
        CarMfg-1.0.jar
        carmfg_data.json
CarMfg_src.zip – All source files. It is a maven project, was developed with Eclipse IDE
CarMfg-Instructions.pdf – i.e. this file, containing a few simple instructions

**Start the Web Service by Running the Executable Jar File**

**A few Notes:**
1. JRE 8 is installed on a machine.
2. Port 8090 is free, it is the default port number to be used by CarMfg service.
   If it is used, then you need to decide what port number to use, e.g. 6789, and then
   pass '-Dserver.port=6789' to java, as shown below
3. The location of the initial static data file, carmfg_data.json, must be made available in order
   For the server to start up since it will try to load it on startup. If it is located in the same folder
   The jar file, then nothing special needs to be done, if not, then you need to pass
   '-Dcar.mfg.data.folder=<the folder containing the carmfg_data.json>

Follow the steps:
Unzip the attached CarMfg-1.0.jar.zip in a folder, say C:\CarMfg.
From command prompt,
cd to C:\ CarMfg
Java -jar CarMfg -1.0.jar

*Or a different port number is preferred, do this:*
Java -**Dserver.port**=6789 -jar CarMfg-1.0.jar

*Or if the carmfg_data.json file is not in the current folder your are running the jar file from, then*
Java -**Dserver.port**=6789 -D Dcar.mfg.data.folder= the folder containing the carmfg_data.json -jar CarMfg-1.0.jar

After it is started, you should see on the console, the message similar to *Started CarMfgApplication in 2.531 seconds.* This indicates that the web service is successfully started and is ready to accept REST calls.

**Running and Testing the Car Manufacturer Service**

Please note that for all methods:
content-type=application/json

The following examples are for using postman.

**Get all manufacturers in Json response Format. Optionally sort and order and search**

GET:

http://localhost:8090/api/car/mfg/v1/manufacturers?sort=true&orderByDesc=true&q=Country:"USA";Mfr_Name:"TESLA, INC.";VehicleTypeName:"Passenger Car"

NOTES:

1. If sort=true, will sort by Country, then Name. Default NO sorting
2. If orderByDesc=true, Default NO sorting
3. Searching pattern must be in the format equivalent to AND operations: Country:"USA";Mfr_Name:"TESLA, INC."VehicleTypeName:"Passenger Car". Default NO Search, display ALL. The names are not case sensitive though.

Improvement:

1. Pagination is desirable if a relative large number of manufacturers.
2. Support more advanced search capabilities, such as OR operations as well, and search other data types

**GET 1 manufacturer based on id**

GET:
http://localhost:8090/api/car/mfg/v1/manufacturers/{id}

**Change 1 manufacturer**

PATCH:
http://localhost:8090/api/car/mfg/v1/manufacturer

input or requestBody

```
{
   "country" : "United States (USA) testing",
   "mfr_ID" : 52,
   "mfr_Name" : "GENERAL MOTORS LLC testing",
   "vehicleTypes" : [ {
     "isPrimary" : false,
     "name" : "Passenger Car"
}
```

**Insert one or more manufacturers**

POST:
http://localhost:8090/api/car/mfg/v1/manufacturers

input or requestBody must be an array

```
[
{
   "country" : "United States (USA) added 1",
   "mfr_Name" : "GENERAL MOTORS LLC added 1",
   "vehicleTypes" : [ {
     "isPrimary" : false,
     "name" : "Passenger Car"
},
```

```
{
   "country" : "United States (USA) added 2",
   "mfr_Name" : "GENERAL MOTORS LLC added 2",
   "vehicleTypes" : [ {
     "isPrimary" : false,
     "name" : "Passenger Car"
}
]
```


**Delete 1 manufacturer based on id**

DELETE:
http://localhost:8090/api/car/mfg/v1/manufacturers/{id}

**Delete all manufacturers**

DELETE:
http://localhost:8090/api/car/mfg/v1/manufacturers

**Save the manufacturer data into a file on the server**

Please note that for delete/post/patch operations, the server saves the manufacturer data into a file, carmfg_new_data.json, in the same folder as carmfg_data.json.
and the duplicates in carmfg_data.json are removed.

**Some notes on the sources, build environment and testing**

```
#################################
# Java 8 EE is required to compile this project

#unzip the sources into a folder

# To compile only
cd /d <the source folder that contains pom.xml>
mvn clean compile

# To package without running tests
cd /d <the source folder that contains pom.xml>
mvn clean package -Dmaven.test.skip=true

# To package and publish with running tests
cd /d <the source folder that contains pom.xml>
mvn clean install

# To run Tests only from the command ine
cd /d <the source folder that contains pom.xml>
mvn clean test
```

# To  package, test, and start the CarMfg web service from the command line using the source:
# NOTE: 1. the default port number is 8090, if need to run on different
#         port number see below
#       2. carmfg_data.json must be in the source folder, or
#         use '-Dcar.mfg.data.folder=<???> pointing to the folder
#         containing the file, carmfg_data.json
cd /d <the source folder that contains pom.xml>
mvn clean package && java -jar target/CarMfg-0.1.0.jar

# or
mvn clean package && java -Dcar.mfg.data.folder=<the folder containing the json file> -jar target/CarMfg-0.1.0.jar

# or if carmfg_data.json file is in the source folder, run by doing:
mvn spring-boot:run

# Please note that the default is 8090 that is
# specified in application.properties files
# To start the CarMfg web service from the command line using
# a non-default port number, e.g. 6789, use '-Dserver.port=<???>', e.g.
cd /d <the source folder that contains pom.xml>
mvn clean package && java -Dserver.port=6789 -jar target/CarMfg-0.1.0.jar

# TESTING
# There are junit tests in src/test/java, but there are many incomplete ones

**# Save the manufacturer data into a file on the server**

# Please note that for delete/post/patch operations, the server saves the manufacturer
# data into a file, carmfg_new_data.json, in the same folder as carmfg_data.json.
# and the duplicates in carmfg_data.json are removed.