|  |  |
| --- | --- |
| **ESG Test Java Application** | **Jonathan Campbell Jones**  **18th August 2024** |

**Introduction**

This Project demonstrates an API updating a database using the data posted to it.

It is a bare bones implementation to demonstrate functionality, and though some validation and error management is performed, this is not comprehensive,

**API**

Project located: <https://github.com/jonathan-hcj/ESGAPI>

The SQL Customer table that holds the customer records takes for following format:

|  |  |  |
| --- | --- | --- |
| Table Customer schema | | |
| CustomerRef | nvarchar(50) | NOT NULL – Primary Key |
| CustomerName | nvarchar(50) | NULL |
| AddressLine1 | nvarchar(50) | NULL |
| AddressLine2 | nvarchar(50) | NULL |
| Town | nvarchar(50) | NULL |
| County | nvarchar(50) | NULL |
| Country | nvarchar(50) | NULL |
| Postcode | nvarchar(20) | NULL |

**Post method**

*https://localhost:[port]/customer*

When adding records, a test is made against the CustomerRef if the reference already exists it will error with the message ‘Customer account already exists’, if no CustomerRef is suppled at all, the error ‘Customer has no reference specified’ is returned, and if the Insert fails for any other reason, the post method returns the error ‘Customer record could not be inserted’.

**Get method**

*https://localhost:[port]/customer?reference=[CustomerReference]*

Searches the database Customer table for a record with the CustomerRef matching the parameter supplied in the call to the method. If the Customer ref does not exist in the table a NotFound response is issued, if it is, then a customer object is populated and returned.

**Setup**

From the project folder ‘Files’ retore ‘ESG.bak’ using MSSQL Management studio, this will create the database in which the customer records are store. The customer table is empty in this backup. Create a user API1 and set the password to ‘Blat’. Ensure that the user has both read and write access to the database.

Open the ESGAPI solution (*ESGAPI.sln*) in visual studio 2022

Set the connection details in the connection string builder as appropriate for your database instance (file CustomerController.cs):

private readonly SqlConnectionStringBuilder builder = new()

{

DataSource = "JONATHANS\_LT\\SQLEXPRESS",

UserID = "API1",

Password = "Blat",

InitialCatalog = "ESG",

TrustServerCertificate = true

};

Run the project. When it starts, a swagger screen will open, this has standard swagger functionality, you may add customers via the post method and recover customers information via the get method.

A screenshot of a computer

Description automatically generated

**Java Console App**

Projects located <https://github.com/jonathan-hcj/ESG-Gradle>

For us to know that the updates have been performed I have added a call to get call to the API to recover a single record. If this works you will see the line ‘Customer 012254 recovered’

Setup

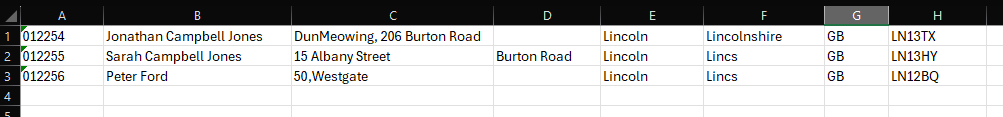
Notes

The app uses two libraries:

**gson** provides Json support for serialising and deserializing objects

**opencsv** provides support for opening the customer CSV from file, this library is a reliable implementation that allows you to wrap field that contains a field separator with quotes.

**Expected results**



Terminal output

c:\Projects\ESG Gradle> c: && cd "c:\Projects\ESG Gradle" && cmd /C "C:\Users\jonat\.gradle\jdks\eclipse\_adoptium-21-amd64-windows\jdk-21.0.4+7\bin\java.exe @C:\Users\jonat\AppData\Local\Temp\cp\_4bz0ivw9t490ai37miyos50qa.argfile esg.gradle.App "

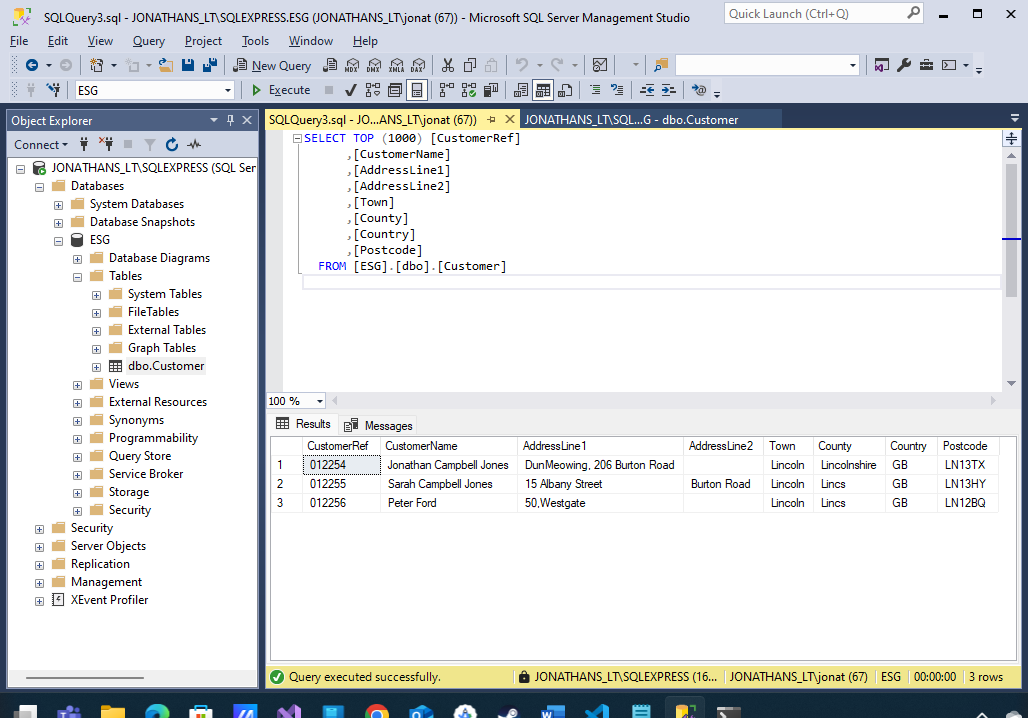
Customer 012254 created

Customer 012255 created

Customer 012256 created

Customer 012254 recovered

Looking back at management studio you will see the records have been inserted, the final line of the output shows that the app was able to recover a customer record from the API.



**Error handling**

A file has been included in the project to demonstrate basic error handling: Customers.error.csv. Clear the datatable in MSSQL Manager studio and replace the records in the customer.csv file with this data in this file.

A screenshot of a computer

Description automatically generated

The system will skip blank lines, however if a record is found, then validation occurs to ensure that the record is both 8 fields long and a CustomeRef is present. If any single record is malformed as described, the the app will not post any records to the API.

Terminal output

um-21-amd64-windows\jdk-21.0.4+7\bin\java.exe -agentlib:jdwp=transport=dt\_socket,server=n,suspend=y,address=localhost:57670 @C:\Users\jonat\AppData\Local\Temp\cp\_4bz0ivw9t490ai37miyos50qa.argfile esg.gradle.App "

Some customers were not valid in the csv:

Line 5: Customer reference missing

Line 6: 8 fields are required per customer

Unable to recover customer: response 404

**Authentication**

I have added code to manage authentication on the API to demonstrate how we might implement basic authentication, however I have disabled it in the development environment to simplify this demonstration.

See BasicAuthenticationHandler.cs.