COMPUTING SUBJECT:	Restful ASP.Net Core-services
TYPE:	Assignment
IDENTIFICATION:	RestService#6
COPYRIGHT:	*Peter Levinsky & Michael Claudius*
LEVEL:	Medium
TIME CONSUMPTION:	1½-2 hours
EXTENT:	120 lines
OBJECTIVE:	Restful services using a Database
PRECONDITIONS:	Rest service theory. Http-concepts Computer Networks Ch. 2.2
COMMANDS:	

IDENTIFICATION: RestService#6 / PELE with kindly respect and inspiration from MICL

Overall Purpose

The overall purpose for the group of 'RestService' assignments is to be able to provide and consume restful ASP.Net Core web services, to prepare the 'RestService' to be published in Azure, including testing the service and finally to setup the 'RestService' to be consumed from a browser (e.g. using Typescript) i.e. support CORS.

The whole group of assignments consist of 7 steps:

1. A simple REST Service with CRUD.
2. More advanced and complex URI's.
3. Adding help-pages to the REST Service (Swagger)
4. Testing a REST Service and publish in Azure.
5. Consuming a REST service from a C# Console application.
6. Adding Support for CORS to the REST Service
7. A REST Service using a database (this assignment)

Background Material:

The HTTP protocol: See Computer Network chap 2 pp. 111-136

Note of REST (Peter Levinsky): See NetHttpNote.pdf

Oswago Universitet: RESTful Service Best Practices: Recommendations for Creating Web Services: See http://cs.oswego.edu/~alex/teaching/csc435/RESTful.pdf

Usefull tools (Postman & Fiddler): See Tools.htm (tool #3 & tool #4)

Helpful link from 2 semester:

SQL references: https://www.w3schools.com/sql/default.asp

C# SqlConnection reference: https://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection(v=vs.110).aspx

C# SqlCommand reference: https://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand(v=vs.110).aspx

This Assignment: RestService#7

Purpose
The purpose of this assignment is to refactor your REST Service so it can use a Database for persistence instead of a static list.

Mission
You are to refactor your implementation of the controllers to use Database. You will only work with one simple table to hold data i.e. no foreign key and no talk of 3th Normal Form.

You are NOT allowed to use Entity Framework, you must use SQLConnections and SQLCommands.

Assignment 1: Prepare Solution for Database persistency

a. Open your REST-service project (more correctly solution). You properly have a reference to your Model Library (see assignment 1) otherwise create a reference to your Model Library

b. Alternative create a new Solution and add the model library reference to the project.

c. In Azure create a Table 'Item' with the properties:

- int Id; // i.e. Id int not null primary key
- string Name; // i.e. Name nvarchar(35) not null
- string Quality; // i.e. Quality nvarchar(35) not null
- double Quantity; // i.e. Quantity float not null

Assignment 2: Create a Utility Class for Database connections

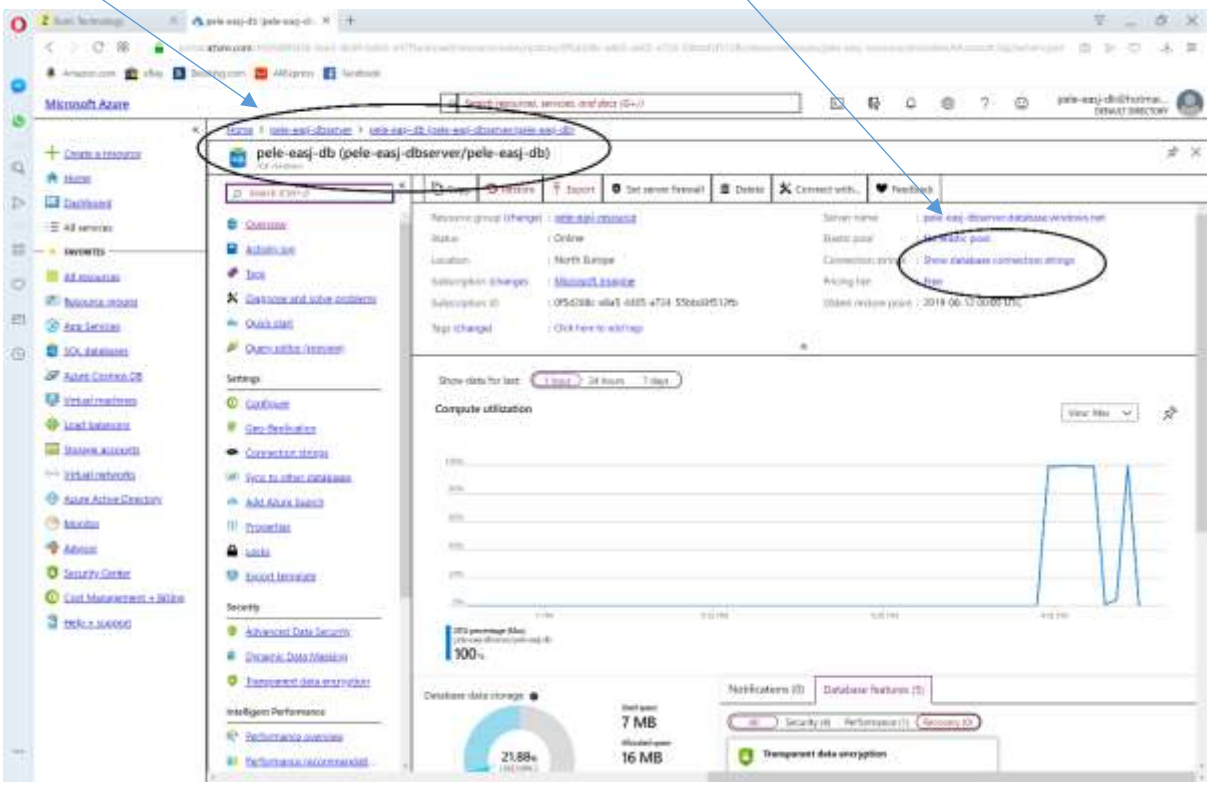
a. In the solution (Solution Exploire); Create a folder e.g. named 'DButil'.

b. In this folder create a class '**ManageItems**' with 5 methods:

```
public IEnumerable<Item> Get()
public Item Get(int id)
public void Post(Item value)
public void Put(int id, Item value)
public void Delete(int id)
```

If you have more methods in your REST service add them as well.

c. To implement these methods you need the connection string (get this string from database in Azure):



Show the connection strings and copy the one for ADO.Net, though you need to fill in your user name and password!!

Make a constant with the connection String.
Make a constant with the sql query e.g.
private const String GET_ALL = “select * from Items”;

```
List<Item> liste = new List<Item>();
using (SqlConnection conn = new SqlConnection(connectionString))
using (SqlCommand cmd = new SqlCommand(GET_ALL, conn))
{
    conn.Open();
    SqlDataReader reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        Pizza item = ReadNextElement(reader);
        liste.Add(item);
    }
    reader.Close();
}
return liste;
```

To make this working, you need to install a nuGet package ‘System.Data.SqlClient’

```
protected Item ReadNextElement(SqlDataReader reader)
{
	Item item = new Item();

	item.Id = reader.GetInt32(0);
	item.Name = reader.GetString(1);
	item.Quality = reader.GetString(2);
	item.Quantity = reader.GetDouble(3);

	return item;
}
```

d. Implement all the other methods

Hint: to insert values

```
GET_ONE = "select * from DemoBooking WHERE Booking_id = @ID";
```

On the sqlCommand 'cmd'

```
cmd.Parameters.AddWithValue("@ID", id);
```

Assignment 3: Refactor the Controller class

a. Refactor your ItemsController to use this ManageItems class, by calling the appropriated methods.

b. Run your component unit test and your integration test

c. If succeed publish the refactored REST service in Azure

By now you are 'full flying' REST service implementer
and can do other REST services