COMPUTING SUBJECT:	Restful ASP.Net Core-services
TYPE:	Assignment
IDENTIFICATION:	RestService#3
COPYRIGHT:	*Peter Levinsky & Michael Claudius*
LEVEL:	Medium
TIME CONSUMPTION:	1½-2½ hours
EXTENT:	80 lines
OBJECTIVE:	Testing the Restful services
PRECONDITIONS:	Rest service theory. Http-concepts Computer Networks Ch. 2.2
COMMANDS:	

IDENTIFICATION: RestService#3 / 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. **Testing a REST Service and publish in Azure. (this assignment)**
4. Adding Support for CORS to the REST Service
5. Consuming a REST service from a C# Console application.
6. A REST Service using a database

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)

This Assignment: RestService#3

Purpose
The purpose of this assignment is to test your REST Service and published in the Azure-cloud.

Mission
You are to control, your Restful web services based on the ASP.Net Core services is working correctly. You will be able set up at test-project (unit test) for the REST-Service, and to publish it in the cloud in this case Azure:

1. Component test (**unit test**) of your Controller

2. Integration Test (**postman**) of your REST-Service

3. Publish in Azure

Additional reading:

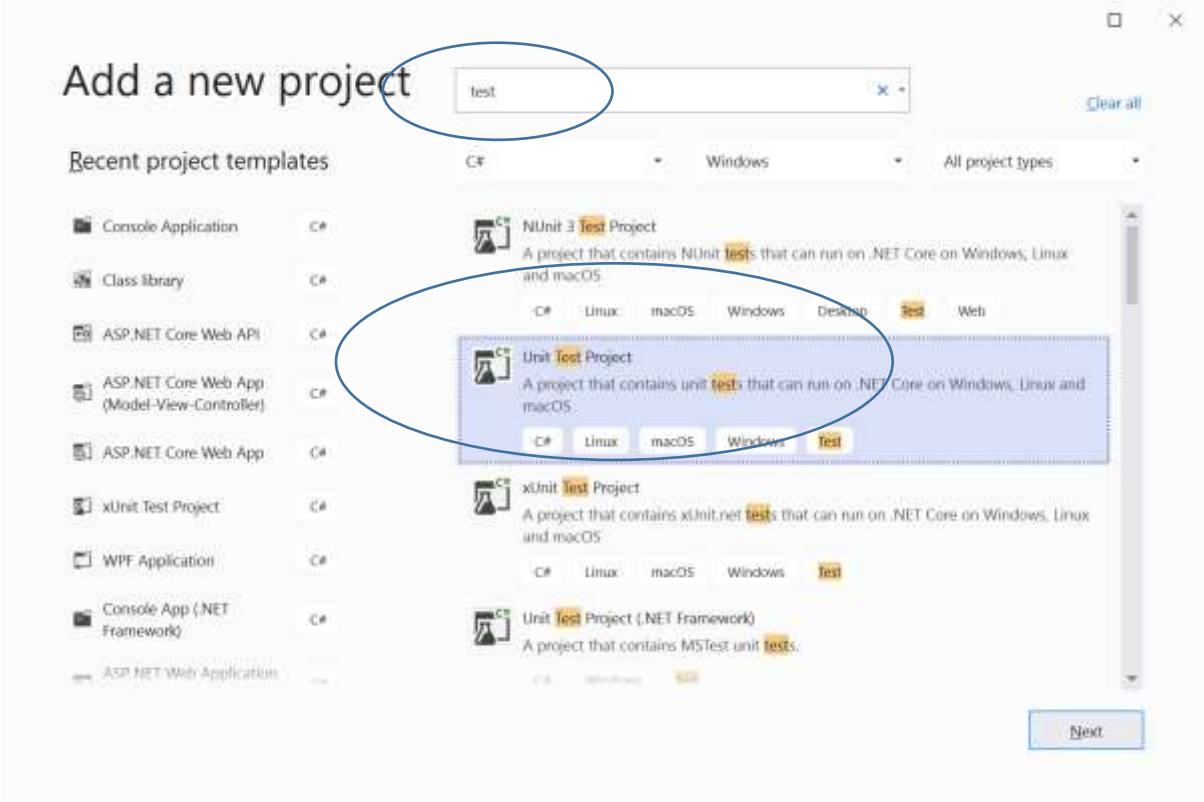
- **Unittest in C# / Visual Studio : https://docs.microsoft.com/en-us/visualstudio/test/unit-test-basics?view=vs-2019**
- **Walkthrough Test : https://docs.microsoft.com/en-us/visualstudio/test/walkthrough-creating-and-running-unit-tests-for-managed-code?view=vs-2019**
- **Azure at Zealand : https://helpdesk.zealand.dk/hc/en-us/articles/205679095-Access-to-Azure-Dev-Tools-Microsoft-Imagine-**

Assignment 1: Component test (unit test) of your Controller

Before use of your REST-Service you should test the manager(s) in this case the '**ManageItems**', by creating a unit test.

You are to create your test-project your-self, so follow these steps:

a. Add a new project to your solution. Right-click on the solution -> add new project – pick the Unit Test (remember to choose among .Net Core applications):



b. In the Test-project in 'dependencies' add a reference to your REST-Service project (i.e. to the controller to be tested), And add second reference to your '**ModelLib**'.

c. Now test all your methods (2 get (could be 4 with the two from exercise RESTService#2), 1 post, 1 put, 1 delete). Make use of the Arrange, Act and Assert principle.
Remember to test for different parameters and return values.

d. Check your coverage of your test by at the 'Test-tab -> analyse code coverage -> all test'

What would be a good coverage?
If you have 100% coverage is your controller you are testing correct?

Assignment 2: Integration Test of your REST-Service

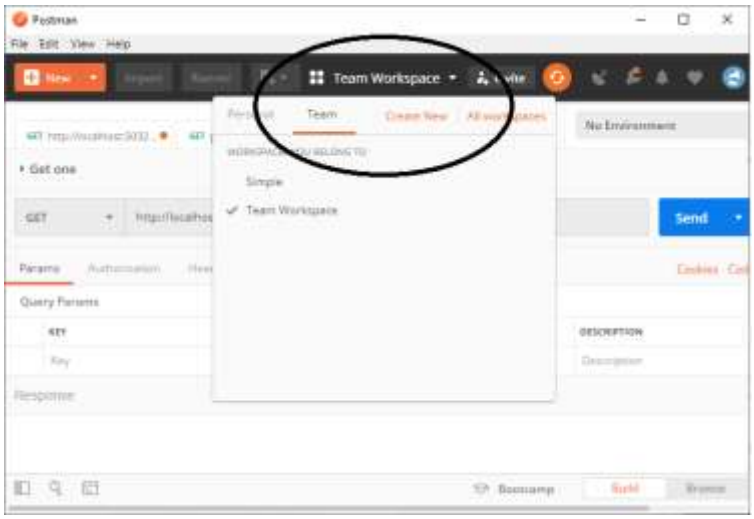
After testing that your controller functional are working correct, -- at least is tested ☺, you are to test your URI are well designed and working i.e. you are going to make an integration test. With this in mind you needed as minimum testing five entries (URI's):

1. Read All (Get)
2. Read One (Get)
3. Create one (Post)
4. Update one (Put)
5. Delete one (Delete)

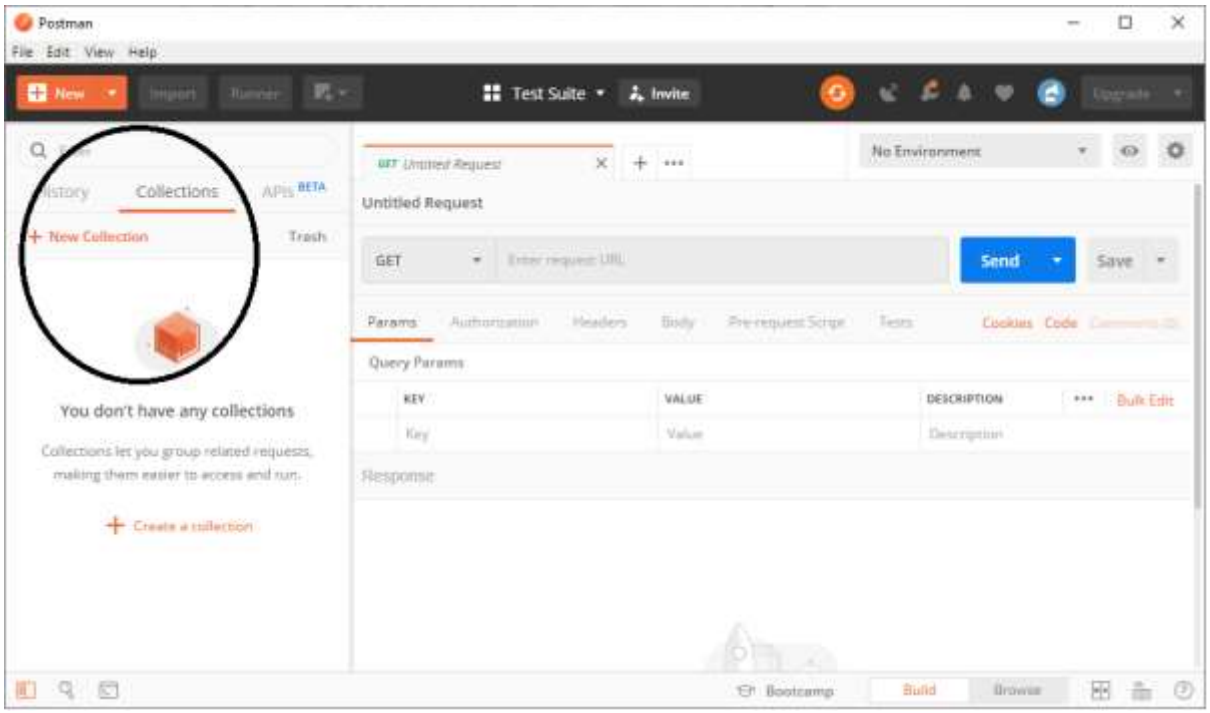
For this purpose you could have two approaches either make a new UnitTest-project, where you making HTTP-connection (see RESTService#5) OR you make a test-suite using Postman.

In this assignment you will go through steps to make a test suite in Postman.

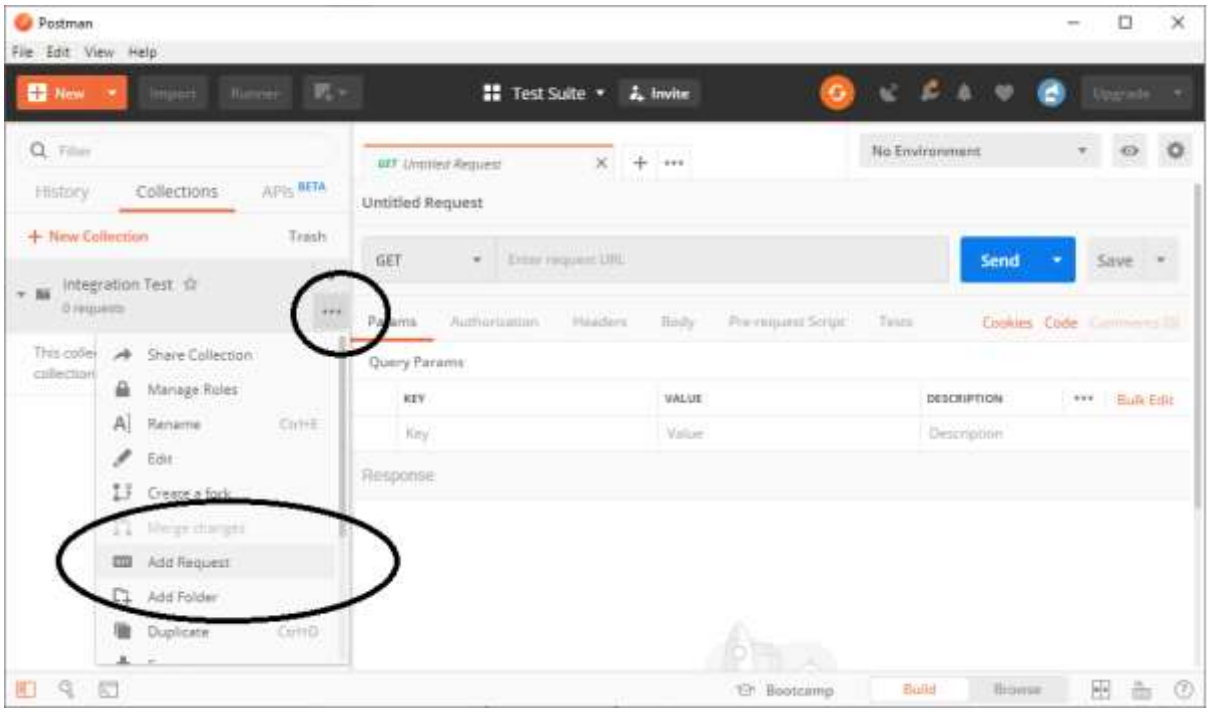
a. Open Postman and create a new workspace e.g. '**Test Suite**'



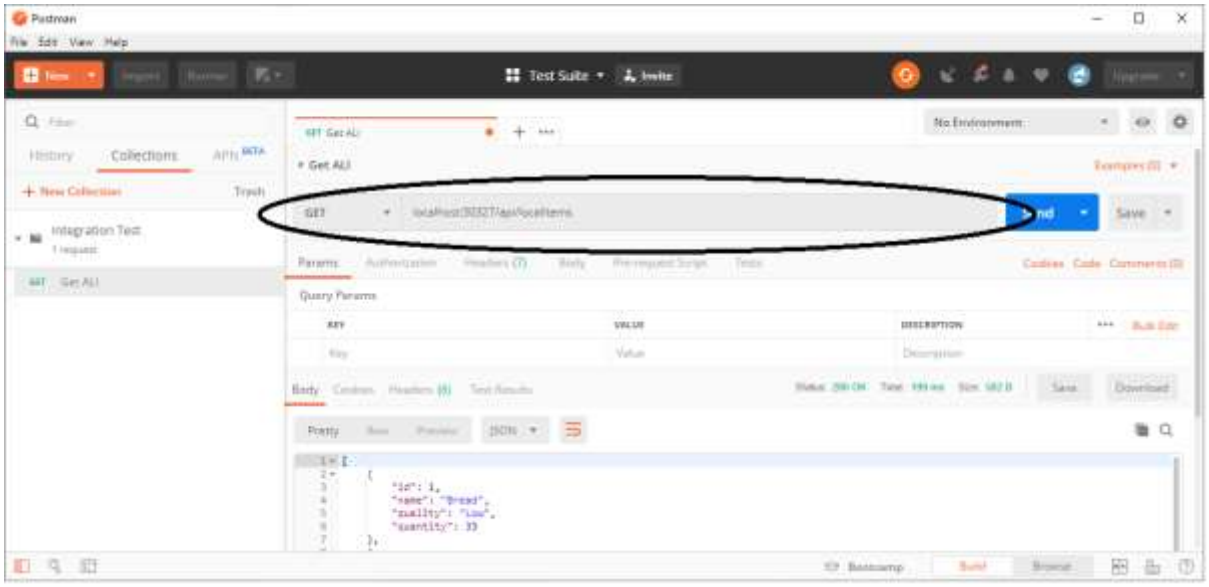
b. Create a new Collection e.g. '**Integration Test**'



c. Next Step is to add the different request to this collection



d. Now you fill out this request as normal i.e. choose method, write URI, and for Post and Put set the body as a json-string e.g. first request ‘Get ALL’ like:



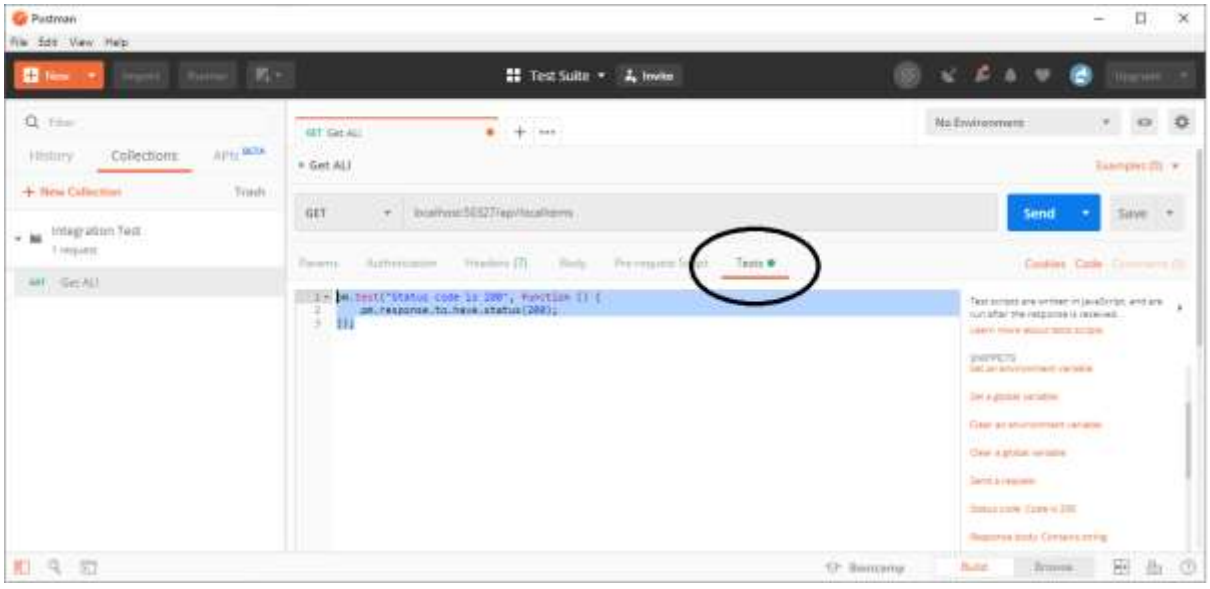
e. Next step is set up the test conditions. This is done in the test-tab using a postman-scripting language.

References to the scripting language:

* https://learning.getpostman.com/docs/postman/scripts/postman_sandbox_api_reference/
* https://learning.getpostman.com/docs/postman/scripts/intro_to_scripts/

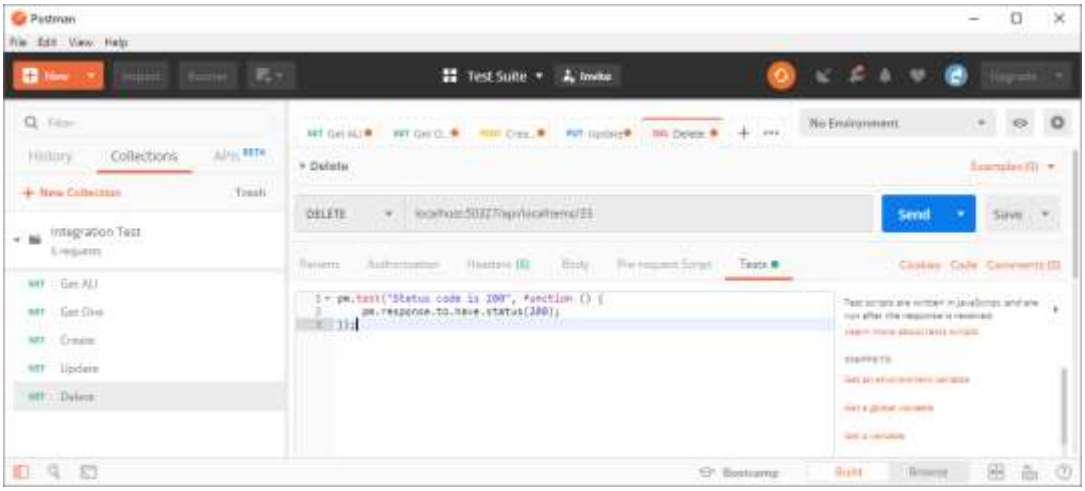
You typical need to test the status code:

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

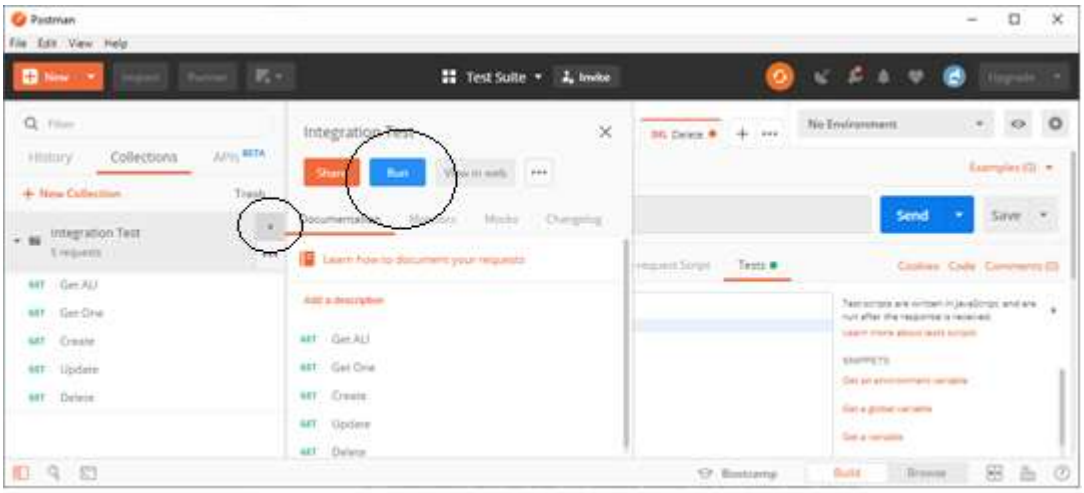


f. You can try this individual request by clicking ‘send’ if it is satisfied, then click ‘**save**’.

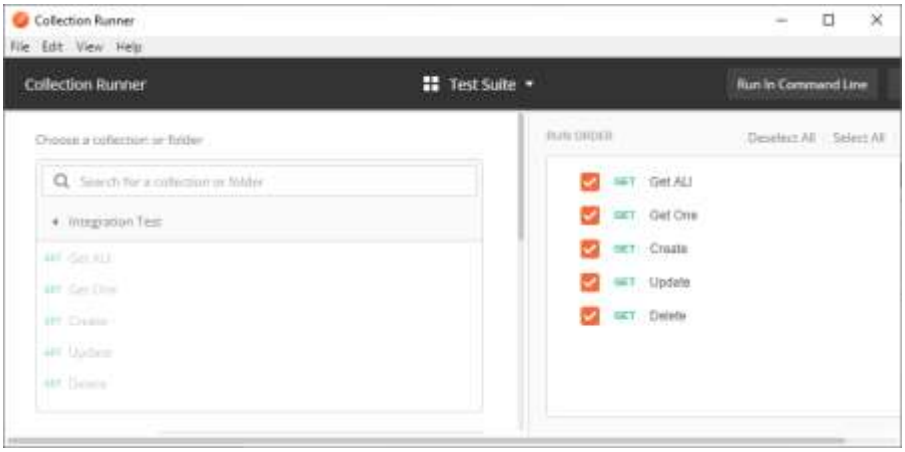
g. After all request have been added to the collection:



h. You can run the whole collection i.e. the test suite by:



And your result should look like:



i. *Extra: Add more in deep test of the result – using the scripting language i.e. check the content of the result body e.g. 'pm.response.to.have.json;'*