

API Testing with C#

RestSharp



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

Have a Question?

sli.do

#QA-Auto-BackEnd

1. Understanding APIs:

- Concepts

2. Serialization and Deserialization:

- JSON files

3. API Testing with C#:

- GitHub API Requests
- NUnit + RestSharp





Integration and API Testing

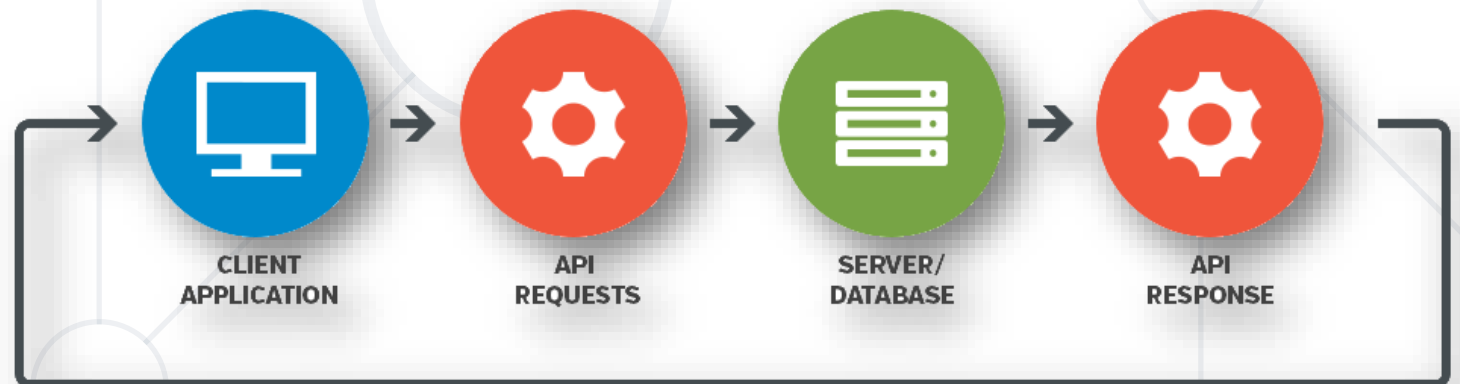
Concepts

Brief Recap on Integration Testing

- Validates combined functionality of software modules
- Detects interface defects, verifies data flow and communication
- Follows unit testing; precedes system testing
- Applies to database integration, **APIs (REST, SOAP, GraphQL)**, service interactions
- **Objective:** Ensure system integrity and seamless functionality

- **API** == **A**pplication **P**rogramming **I**nterface
 - APIs serve as the intermediaries that allow two applications to talk to each other
 - Set of **functions** and **specifications** that software programs and components follow to talk to each other
- **API examples:**
 - **JDBC** – Java API for apps to talk with database servers
 - **Windows API** – Windows apps talk with Windows OS
 - **Web Audio API** – play audio in the Web browser with JS

- **Web services** expose **back-end APIs** over the **network**
 - May use different **protocols** and **data formats**: HTTP, REST, GraphQL, gRPC, SOAP, JSON-RPC, JSON, BSON, XML, YML, ...
- **Web services** are hosted on a Web server (HTTP server)
 - Provide a set of functions, invokable from the Web (Web API)
- **RESTful APIs** is the most popular Web service standard



- **SOAP (Simple Object Access Protocol)** APIs are protocol-based, ensuring high levels of security and standardized communication via XML
- Preferred for enterprise-level web services where security and transactional reliability are paramount
- **Testing Focus:** Requires thorough validation of the SOAP envelope, headers, and body, along with adherence to WS-* standards

- **National Weather Service (NWS)** SOAP-based API for weather data
- <http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl>
- Includes WSDL (Web Services Description Language) document for describing the services offered
 - **Operation:** Get weather forecast
 - **SOAP Action:** LatLonListCityNames
 - **Request:** XML formatted SOAP envelope specifying the desired operation and parameters(of city names or geographic coordinates)
 - **Response:** XML-formatted response (weather forecast data for the requested locations)

- Defined by **GraphQL** (query language) that allows clients to request exactly what they need, making it highly efficient for data retrieval
- Enables clients to define the structure of the data required, reducing over-fetching and under-fetching issues common in REST APIs
- **Testing Considerations:** Emphasizes validating query responses, handling dynamic data retrieval, and ensuring efficient performance

- GitHub offers a GraphQL API, enabling clients to request exactly the data they need
- GraphQL queries are tailored by the requester
- GraphQL API has a single endpoint:
 - <https://api.github.com/graphql>
- More on GitHub GraphQL API:
 - <https://docs.github.com/en/graphql>

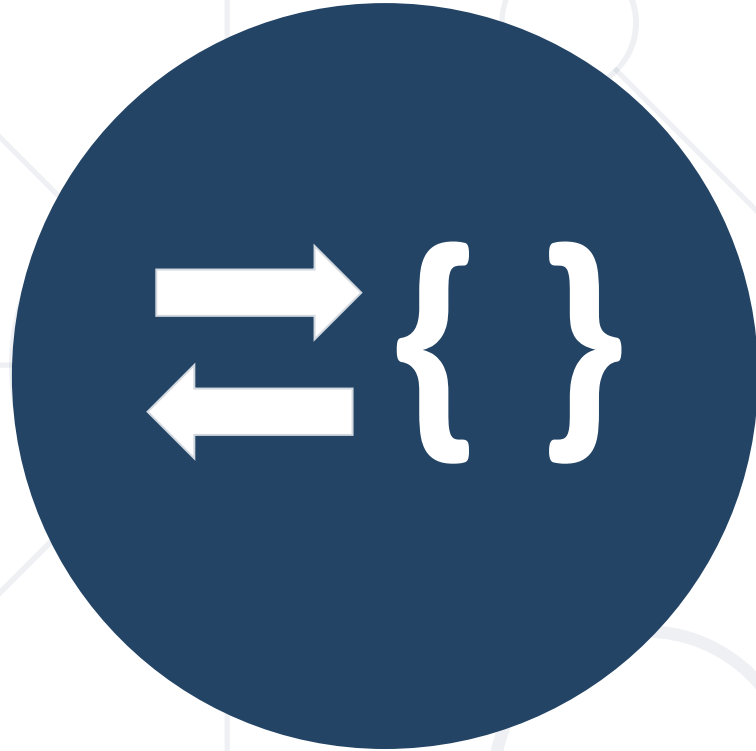
- Fetching a user's name and the last three issues from a repository

```
query {  
  repository(owner:"octocat", name:"Hello-World") {  
    issues(last:3) {  
      edges {  
        node {  
          title  
          url  
        }  
      }  
    }  
  }  
}
```

- Based on the Representational State Transfer architectural style, emphasizing simplicity, statelessness, and a uniform interface
- Utilizes standard **HTTP methods** (**GET, POST, PUT, DELETE**) for operations, making it widely adopted for web services
- Offers scalability, performance, and ease of use, with data typically exchanged in **JSON** or XML format

- The GitHub REST API - widely used for actions like creating repositories, fetching user data, or automating workflow processes
 - Base URL for GitHub API v3: <https://api.github.com>
 - Example Operation: Fetching a user's public repositories
 - Request Type: GET
 - Endpoint: /users/{username}/repos
 - Sample Request: GET <https://api.github.com/users/octocat/repos>
- Retrieves data in **JSON format**, listing **public repositories** for the specified GitHub username

- The process includes sending various HTTP requests to the API endpoints and **assessing the responses against expected outcomes**
 - **Endpoints:** Each URL that allows access to the API's resources
 - **HTTP Methods:** GET, POST, PUT, DELETE, etc., which define the action on the resources
 - **Response Status:** HTTP response status codes like 200 OK, 404 Not Found, etc.
 - **Data Exchange:** Validating the request payload and response body



Serialization and Deserialization

Data Handling in RESTful APIs

Data Exchange

- During RESTful API interactions, data is frequently exchanged between a client and a server
- The data must be in a format that both can understand
- **Serialization**: Converts complex data structures or object states into a **flat format** suitable for HTTP communication, storage, or file-based persistence
- **Deserialization**: Reconstructs the flat data back into usable objects or data structures after it's received



Built-in JSON Support

- .NET has built-in JSON support through the **System.Text.Json** NuGet Package



System.Text.Json ✓ by Microsoft, **1.78B** downloads

8.0.1

Provides high-performance and low-allocating types that serialize objects to JavaScript Object Notation (JSON) text and deserialize JSON text to objects, with UTF-8 support built-in. Also provi...

- It supports **serializing** objects and **deserializing** (parsing) strings
- Include **the following namespaces** into your project

```
using System.Text.Json;  
using System.Text.Json.Serialization;
```



- The **System.Text.Json** serializer can read and write JSON

```
class WeatherForecast
{
    public DateTime Date { get; set; } = DateTime.Now;
    public int TemperatureC { get; set; } = 30;
    public string Summary { get; set; } = "Hot summer day";
}

static void Main()
{
    WeatherForecast forecast = new WeatherForecast();
    string weatherInfo = JsonSerializer.Serialize(forecast);
    Console.WriteLine(weatherInfo);
}
```

- To deserialize from a file, we read the file into a string and then use the **Deserialize** method

```
static void Main()
{
    string jsonString = File.ReadAllText(file);
    WeatherForecast forecast =
        JsonSerializer.Deserialize<WeatherForecast>(jsonString);
}
```



Name	Value	Type
forecast	{JsonDemo.Program.WeatherForecast}	JsonDemo.Program.WeatherForecast
Date	{5.2.2023 r. 20:16:32}	System.DateTime
Summary	"Hot summer day"	string
TemperatureC	30	int

What is JSON.NET?

- **JSON.NET** is a JSON **framework** for .NET
 - **More functionality** than built-in functionality
 - Supports **LINQ-to-JSON**
 - Out-of-the-box support for parsing between **JSON** and **XML**
 - Open-source project: <http://www.newtonsoft.com>
 - To install JSON.NET use the **NuGet Package Manager**



Newtonsoft.Json ✓ by James Newton-King, **4.2B** downloads
Json.NET is a popular high-performance JSON framework for .NET

13.0.3

- JSON.NET exposes a static service **JsonConvert**
- Used for parsing and configuration to
 - **Serialize** an object

```
var jsonProduct = JsonConvert.SerializeObject(product);
```

- **Deserialize** an object

```
var objProduct =  
    JsonConvert.DeserializeObject<Product>(jsonProduct);
```

- By default, the result is a **single line of text**
- To indent the output string use **Formatting.Indented**

```
JsonConvert.SerializeObject(products, Formatting.Indented);
```

```
{  
  "pump": {  
    "Id": 0,  
    "Name": "Oil Pump",  
    "Description": null,  
    "Cost": 25.0  
  },  
  "filter": {  
    "Id": 0,  
    "Name": "Oil Filter",  
    "Description": null,  
    "Cost": 15.0  
  }  
}
```

- Deserializing to **anonymous** types

Incoming JSON

```
var json = @"{ 'firstName': 'Svetlin',  
               'lastName': 'Nakov',  
               'jobTitle': 'Technical Trainer' }";
```

```
var template = new  
{  
    FirstName = string.Empty,  
    LastName = string.Empty,  
    JobTitle = string.Empty  
};
```

Template
objects

```
var person = JsonConvert.DeserializeAnonymousType(json,  
template);
```


- By default JSON.NET takes each property / field from the class and parses it
 - This can be controlled using **attributes**

```
public class User
{
    [JsonProperty("user")]
    public string Username { get; set; }

    [JsonIgnore]
    public string Password { get; set; }
}
```

Parse **Username**
to **user**

Skip the property

- By default JSON.NET takes each property / field from the class and parses it
 - This can be controlled using **ContractResolver**

```
DefaultContractResolver contractResolver =  
    new DefaultContractResolver()  
    {  
        NamingStrategy = new SnakeCaseNamingStrategy()  
    };  
var serialized = JsonConvert.SerializeObject(person,  
    new JsonSerializerSettings()  
    {  
        ContractResolver = contractResolver,  
        Formatting = Formatting.Indented  
    });
```

- LINQ-to-JSON works with **JObjects**

- Create from JSON string

```
JObject obj = JObject.Parse(jsonProduct);
```

- Reading from file

```
var people = JObject.Parse(File.ReadAllText(@"c:\people.json"))
```

- Using **JObject**

```
foreach (JToken person in people)
{
    Console.WriteLine(person["FirstName"]); // Ivan
    Console.WriteLine(person["LastName"]); // Petrov
}
```

- **JObjects** can be queried with LINQ

```
var json = JObject.Parse(@"{'products': [  
  {'name': 'Fruits', 'products': ['apple', 'banana']},  
  {'name': 'Vegetables', 'products': ['cucumber']}]}");
```

```
var products = json["products"].Select(t =>  
  string.Format("{0} ({1})",  
    t["name"],  
    string.Join(", ", c["products"]  
  ));
```

```
// Fruits (apple, banana)  
// Vegetables (cucumber)
```



RestSharp

Simplifying REST API Calls in C#

- **RestSharp** is popular REST API client library for .NET
 - Very simple, quite powerful
 - Official site: <https://restsharp.dev>
 - Execute HTTP **requests** (sync & async)
 - Submit HTTP **parameters**, forms, query string, URL segment, etc.
 - Send / receive / serialize / parse **JSON** and **XML** payloads
 - Multiple **authentication** schemes: Basic, JWT, OAuth
 - **Community** of millions developers



- Installing RestSharp through NuGet:



RestSharp by .NET Foundation and Contributors
Simple REST and HTTP API Client

110.2.0

- Executing simple **HTTP GET** request:

```
using RestSharp;
```

```
var client = new RestClient("https://api.github.com");
```

```
var request = new RestRequest("/users/softuni/repos", Method.Get);
```

```
var response = client.Execute(request);
```

```
Console.WriteLine(response.Content);
```

Using URL Segment Parameters

```
var client = new RestClient("https://api.github.com");
```

```
var request = new RestRequest(  
    "/repos/{user}/{repo}/issues/{id}", Method.Get);
```

```
request.AddUrlSegment("user", "testnakov");  
request.AddUrlSegment("repo", "test-nakov-repo");  
request.AddUrlSegment("id", 1);
```

```
var response = client.Execute(request);
```

```
Console.WriteLine(response.StatusCode);  
Console.WriteLine(response.Content);
```


Deserializing JSON Responses

```
var client = new RestClient("https://api.github.com");
```

```
var request = new RestRequest(  
    "/users/softuni/repos", Method.Get);
```

```
var resp = client.Execute(request);
```

```
public class Repo {  
    public int id { get; set; }  
    public string full_name { get; set; }  
    public string html_url { get; set; }  
}
```

- Reference the [GitHub REST API documentation](#), which outlines the properties of a repository object

```
var repos = JsonSerializer.Deserialize<List<Repo>>(resp.Content);
```

- **Reading** from a public GitHub project is open to everyone
- **Modifying** data in a GitHub project requires **authentication**
 - Get an **API access token** from your GitHub profile:
<https://github.com/settings/tokens/new>
 - Use **HTTP basic authentication**: username + token

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

- Executing **HTTP POST** request with RestSharp:

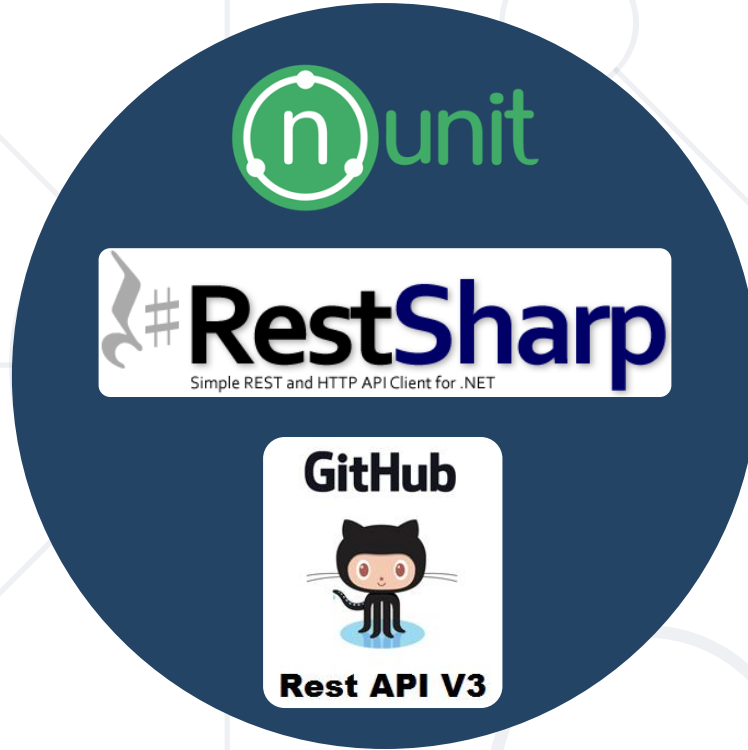
```
var client = new RestClient(new RestClientOptions("https://api.github.com")  
{  
    Authenticator = new HttpBasicAuthenticator("username", "api-token")  
});
```

```
var request = new RestRequest  
    ("/repos/testnakov/test-nakov-repo/issues", Method.Post);
```

```
request.AddHeader("Content-Type", "application/json");  
request.AddJsonBody(new { title = "Title", body = "Body" });
```

```
var response = client.Execute(request);
```

```
Console.WriteLine(response.StatusCode);
```



API Testing for GitHub Issues

Testing RESTful Services

- Creating **API tests** in C# with **NUnit + RestSharp**:
 1. Create **new NUnit Test Project** in Visual Studio
 2. Install the **RestSharp** package from NuGet
 3. Write the test methods

```
[Test]
✓ | 0 references
public void Test_GitHubAPIRequest()
{
    var client = new RestClient("https://api.github.com");
    var request = new RestRequest("/repos/testnakov/test-nakov-repo/issues", Method.Get);
    var response = client.Get(request);
    Assert.That(response.StatusCode, Is.EqualTo(HttpStatusCode.OK));
}
```

- Use MaxTimeout on the RestClient to establish a maximum timeout that applies to all requests made by that client instance

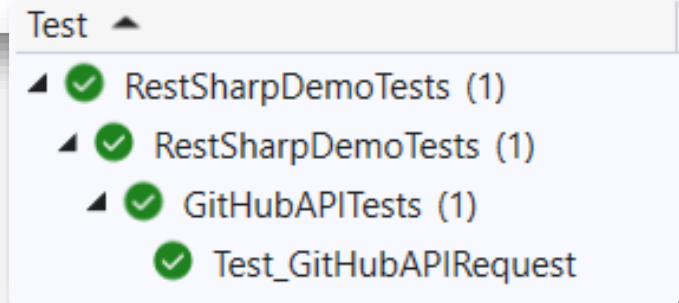
```
public class GitHubAPITests
{
    private RestClient client;

    [SetUp]
    0 references
    public void Setup()
    {
        var options = new RestClientOptions("https://api.github.com")
        {
            MaxTimeout = 3000
        };

        client = new RestClient(options);
    }
}
```

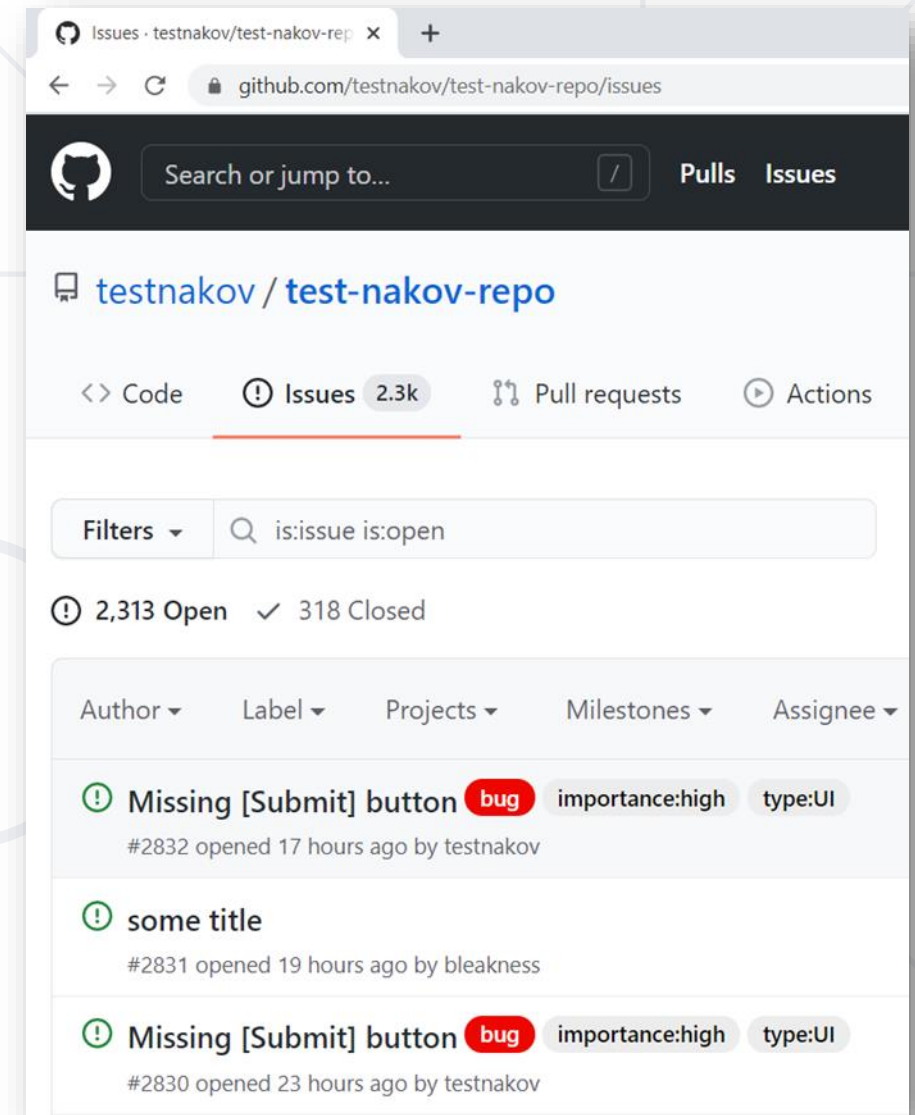
- Use Timeout on the RestRequest when you want to set a timeout for individual requests

```
[Test]
| 0 references
public void Test_GitHubAPIRequest()
{
    var client = new RestClient("https://api.github.com");
    var request = new RestRequest("/repos/testnakov/test-nakov-repo/issues", Method.Get);
    //request.Timeout = 1000;
    var response = client.Get(request);
    Assert.That(response.StatusCode, Is.EqualTo(HttpStatusCode.OK));
}
```



Problem: Testing the GitHub API

- Using the **GitHub official REST API** create the following requests and **test them**:
 - List all issues from the repo "testnakov/test-nakov-repo"
 - Create a **new issue** in the same repo
 - Edit created issue
- You can check the results in the project's issue tracker



Solution: Setup the Test Class

- Setup the **test class**, create the **REST client** and configure HTTP Basic Authentication

0 references

```
public class GitHubAPITests
```

```
{
```

```
    private RestClient client;
```

```
    [SetUp]
```

0 references

```
    public void Setup()
```

```
{
```

```
        var options = new RestClientOptions("https://api.github.com")
```

```
{
```

```
    Authenticator = new HttpBasicAuthenticator("username", "token")
```

```
};
```

```
    this.client = new RestClient(options);
```

```
}
```

Solution: Get All Issues (HTTP GET)

[Test]

✓ | 0 references

```
public void Test_GetAllIssuesFromARepo()
{
    var request = new RestRequest("repos/testnakov/test-nakov-repo/issues");
    var response = client.Execute(request);
    var issues = JsonSerializer.Deserialize<List<Issue>>(response.Content);

    Assert.That(issues.Count > 1);
```

```
    foreach (var issue in issues)
    {
        Assert.That(issue.id, Is.GreaterThan(0));
        Assert.That(issue.number, Is.GreaterThan(0));
        Assert.That(issue.title, Is.Not.Empty);
    }
}
```

```
public class Issue
{
    2 references | ✓ 2/2 passing
    public int id { get; set; }
    2 references | ✓ 2/2 passing
    public int number { get; set; }
    2 references | ✓ 2/2 passing
    public string title { get; set; }
    0 references
    public string body { get; set; }
}
```

Solution: Create New Issue (HTTP POST)

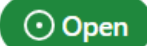
```
private Issue CreateIssue(string title, string body)
{
    var request = new RestRequest("repos/testnakov/test-nakov-repo/issues");
    request.AddBody(new { body, title });
    var response = client.Execute(request, Method.Post);
    var issue = JsonSerializer.Deserialize<Issue>(response.Content);
    return issue;
}
```

[Test]

✓ | 0 references

```
public void Test_CreateGitHubIssue()
{
    string title = "This is a Demo Issue";
    string body = "QA Back-End Automation Course February 2024";
    var issue = CreateIssue(title, body);
    Assert.That(issue.id, Is.GreaterThan(0));
    Assert.That(issue.number, Is.GreaterThan(0));
    Assert.That(issue.title, Is.Not.Empty);
}
```

This is a Demo Issue #4946

 Open QA-Automation-Testing-Demo opened this issue now



QA-Automation-Testing-Demo commented now

QA Back-End Automation Course February 2024

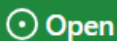
Solution: Edit Created Issue (HTTP PATCH)

```
[Test]
✓ | 0 references
public void Test_EditIssue()
{
    var request = new RestRequest("repos/testnakov/test-nakov-repo/issues/4946");
    request.AddJsonBody(new
    {
        title = "Changing the name of the issue that I created"
    });

    var response = client.Execute(request, Method.Patch);
    var issue = JsonSerializer.Deserialize<Issue>(response.Content);

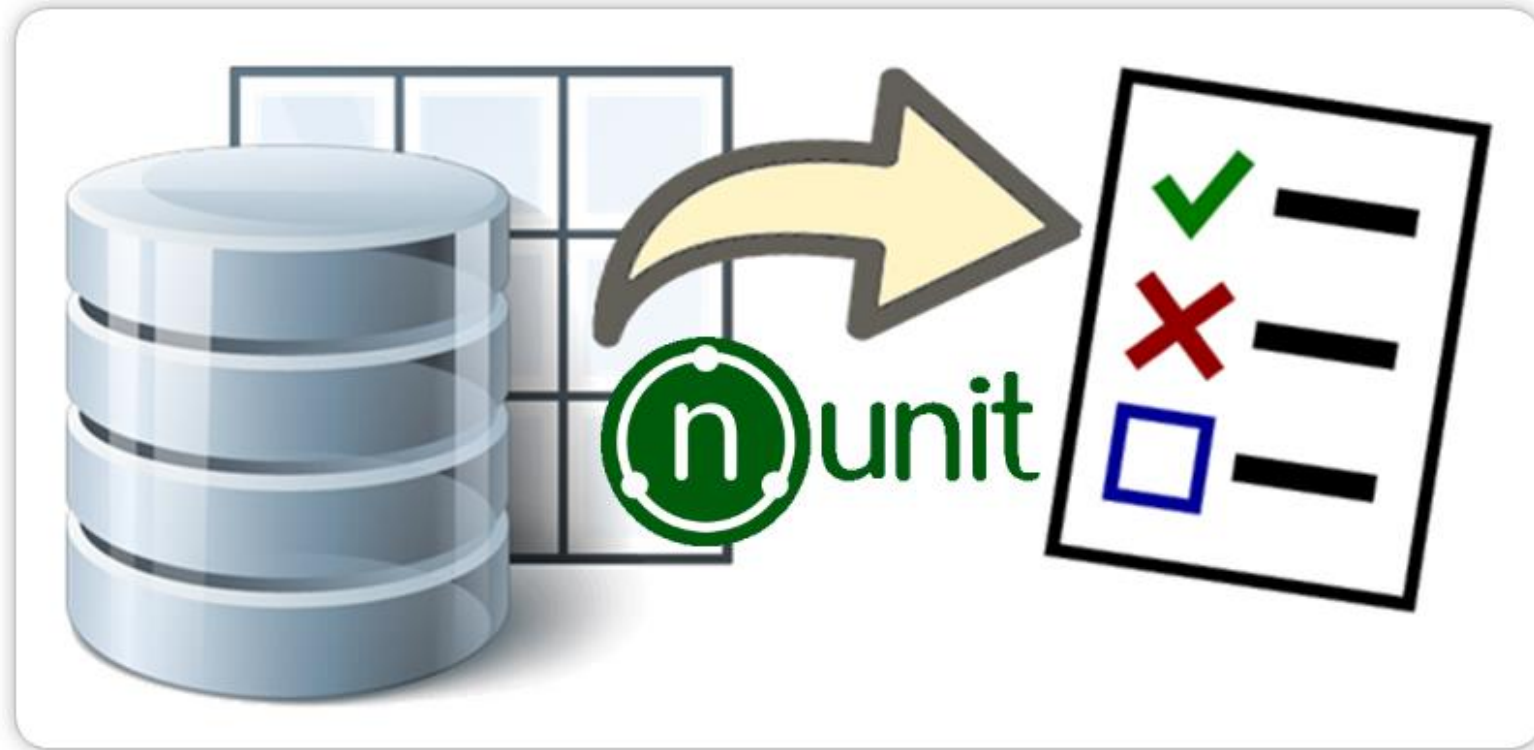
    Assert.That(response.StatusCode, Is.EqualTo(HttpStatusCode.OK));
    Assert.That(issue.id, Is.GreaterThan(0), "Issue ID should be greater than 0.");
    Assert.That(response.Content, Is.Not.Empty, "The response content should not be empty.");
    Assert.That(issue.number, Is.GreaterThan(0), "Issue number should be greater than 0.");
    Assert.That(issue.title, Is.EqualTo("Changing the name of the issue that I created"));
}
```

Changing the name of the issue that I created #4946



Open

QA-Automation-Testing-Demo opened this issue 7 minutes ago · 0 comments



Data-Driven API Tests

Using **[TestCase]** to Assign Data to Tests

- **Data-driven testing** == running the same test case with multiple data (e. g. datasets in the C# code / Excel spreadsheet)
 - Each `[TestCase(... data ...)]` creates a separate unit test

Data Set

```
[TestCase("BG", "1000", "Sofija")]
[TestCase("BG", "5000", "Veliko Turnovo")]
[TestCase("CA", "M5S", "Toronto")]
[TestCase("GB", "B1", "Birmingham")]
[TestCase("DE", "01067", "Dresden")]
public void TestZippopotamus(
    string countryCode, string zipCode,
    string expectedPlace)
```



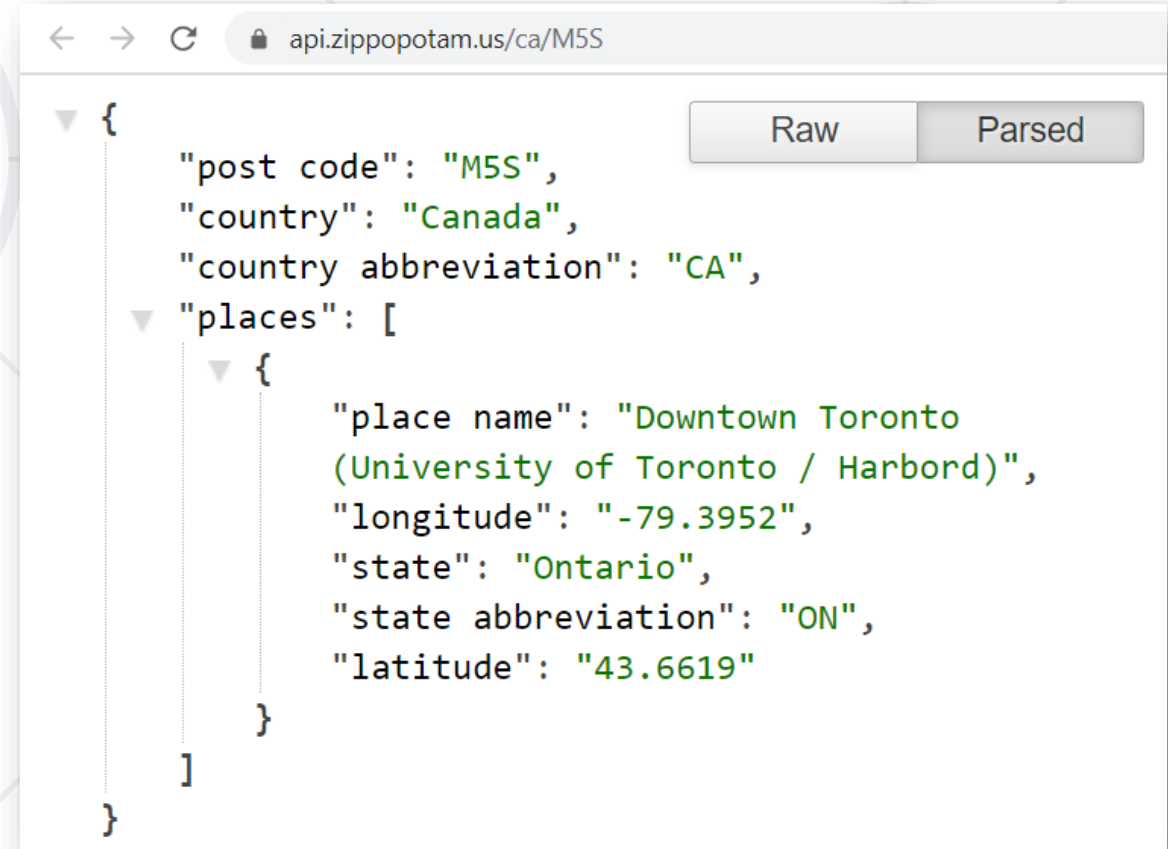
Testing Script

```
public void TestZippopotamus(
    string countryCode, string zipCode,
    string expectedPlace)
// Arrange
var restClient = new RestClient("https://api.zippopotam.us");
var httpRequest = new RestRequest(countryCode + "/" + zipCode);

// Act
var httpResponse = restClient.Execute(httpRequest);
var location = new JsonSerializer().Deserialize<Location>(httpResponse);

// Assert
StringAssert.Contains(expectedPlace, location.Places[0].PlaceName);
}
```


- Zippopotam.us is free API
 - Provides location data by **country code + zip code**
- Example HTTP GET request:
<https://api.zippopotam.us/ca/M5S>
- Use the "**JSON Formatter**" plugin for Chrome to view the JSON response



```
{
  "post code": "M5S",
  "country": "Canada",
  "country abbreviation": "CA",
  "places": [
    {
      "place name": "Downtown Toronto (University of Toronto / Harbord)",
      "longitude": "-79.3952",
      "state": "Ontario",
      "state abbreviation": "ON",
      "latitude": "43.6619"
    }
  ]
}
```

Data-Driven NUnit Tests with [Main TestCase]

```
[TestCase("BG", "1000", "Sofija")]
[TestCase("BG", "5000", "Veliko Turnovo")]
[TestCase("CA", "M5S", "Toronto")]
[TestCase("GB", "B1", "Birmingham")]
[TestCase("DE", "01067", "Dresden")]
public void TestZipopotamus(
    string countryCode, string zipCode, string expectedPlace)
{
    // Arrange
    var restClient = new RestClient("https://api.zipopotam.us");
    var httpRequest = new RestRequest(countryCode + "/" + zipCode);

    // Act
    var httpResponse = restClient.Execute(httpRequest);
    var location = new JsonSerializer().Deserialize<Location>(httpResponse);

    // Assert
    StringAssert.Contains(expectedPlace, location.Places[0].PlaceName);
}
```

▲ ✓ GitHubTests.ZipopotamusApiTests.TestZipopotamus	1,8 sec
✓ TestZipopotamus("BG","1000","Sofija")	597 ms
✓ TestZipopotamus("BG","8600","Jambol")	322 ms
✓ TestZipopotamus("CA","M5S","Toronto")	305 ms
✓ TestZipopotamus("DE","01067","Dresden")	309 ms
✓ TestZipopotamus("GB","B1","Birmingham")	308 ms

Data-Driven NUnit Tests with Classes

```
public class Location
{
    [JsonPropertyName("post code")]
    0 references
    public string postCode { get; set; }

    [JsonPropertyName("country")]
    0 references
    public string Country { get; set; }

    [JsonPropertyName("country abbreviation")]
    0 references
    public string CountryAbbreviation { get; set; }

    [JsonPropertyName("places")]
    1 reference | 🟢 5/5 passing
    public List<Place> Places { get; set; }
}
```

```
public class Place
{
    [JsonPropertyName("place name")]
    1 reference | 🟢 5/5 passing
    public string PlaceName { get; set; }

    0 references
    public string State { get; set; }

    0 references
    public string StateAbbreviation { get; set; }

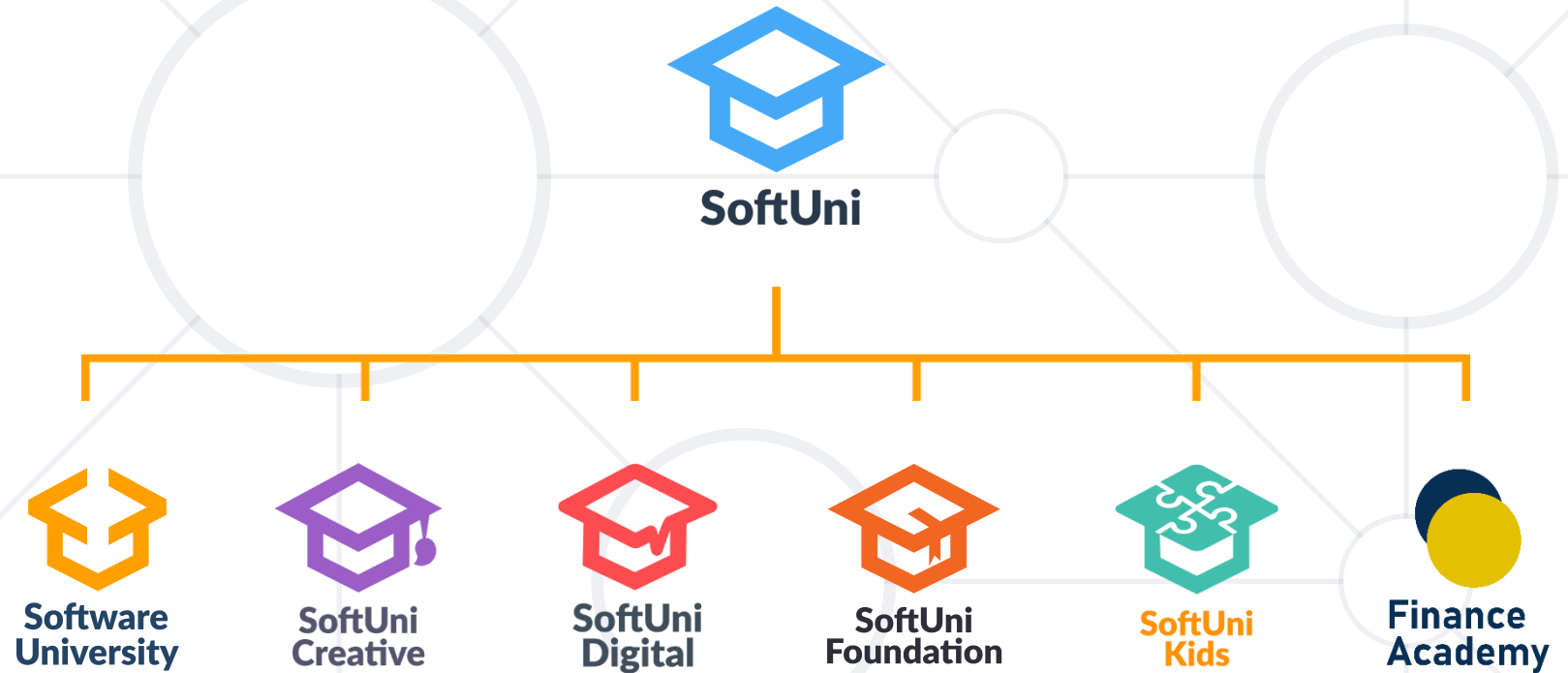
    0 references
    public string Latitude { get; set; }

    0 references
    public string Longitude { get; set; }
}
```

- Understanding APIs: **Interface for services**
- **Serialization** and **Deserialization**: Data to format **conversion**
- **JSON**: Data interchange and object mapping
- How to **use NUnit** to structure the tests and run them, and **RestSharp** to make the **HTTP requests** to the API and **verify the responses**
- Data-Driven **API Tests**



Questions?



SoftUni Diamond Partners



- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity



Software University



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>

