APL64 Rest Web Service HttpClient

Table of Contents

Overview

In this example, an APL64 is used to access a web service. The web service in this example is a rest web service.

A rest web service is created in Visual Studio. For purposes of the example, the web service is run locally using Microsoft IIS web server. The web service exposes CRUD operations on a server-side data source. The simplified data source in the example is a list of Friend records. The server-side operations exposed to a client are:

- Create a new Friend record
- Read all or specified Friend record(s)
- Update a Friend record
- Delete a Friend record

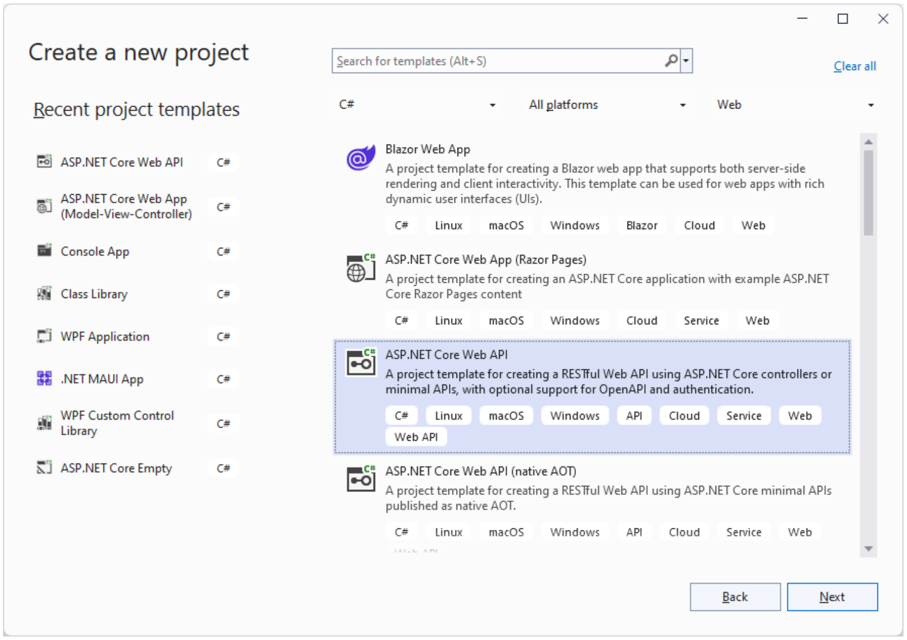
The APL64 C# script engine (⎕cse) is used to create a Microsoft .Net Http Client so APL64 can interact with the rest web service and invoke the CRUD operations on the server-side data source.

Create Rest Web Service

Create a New Visual Studio Project

Select the project template

This Visual Studio project template is a 'weather forecast' rest web service example.



Create a new project

Search for templates (Alt+S) | Clear all

C# | All platforms | Web

Recent project templates

ASP.NET Core Web API	C#
ASP.NET Core Web App (Model-View-Controller)	C#
Console App	C#
Class Library	C#
WPF Application	C#
.NET MAUI App	C#
WPF Custom Control Library	C#
ASP.NET Core Empty	C#

Blazor Web App
A project template for creating a Blazor web app that supports both server-side rendering and client interactivity. This template can be used for web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Blazor Cloud Web

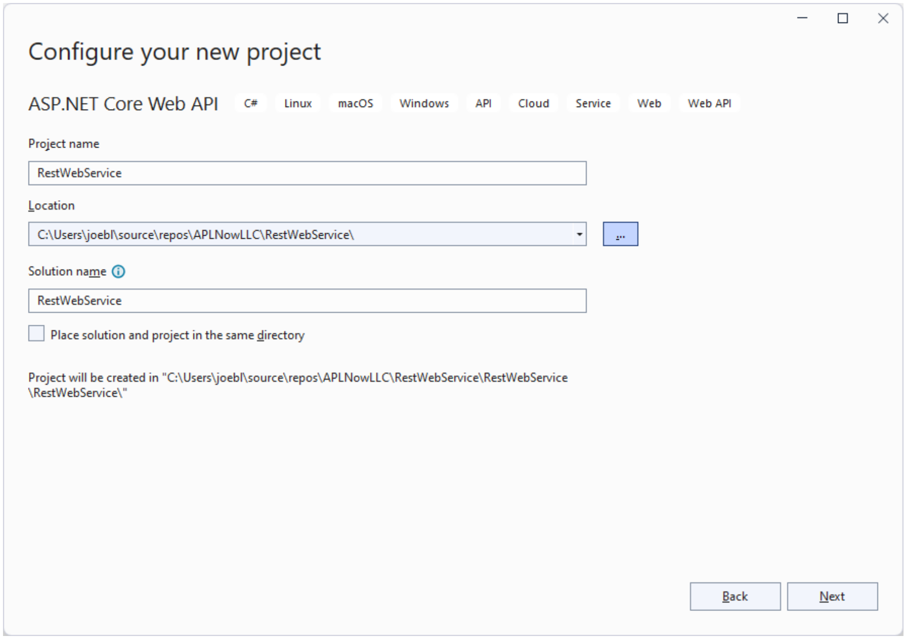
ASP.NET Core Web App (Razor Pages)
A project template for creating an ASP.NET Core application with example ASP.NET Core Razor Pages content
C# Linux macOS Windows Cloud Service Web

ASP.NET Core Web API
A project template for creating a RESTful Web API using ASP.NET Core controllers or minimal APIs, with optional support for OpenAPI and authentication.
C# Linux macOS Windows API Cloud Service Web Web API

ASP.NET Core Web API (native AOT)
A project template for creating a RESTful Web API using ASP.NET Core minimal APIs published as native AOT.
C# Linux macOS Windows API Cloud Service Web

Back | Next

Configure the RestWebService project


Configure your new project

ASP.NET Core Web API

Project name: RestWebService

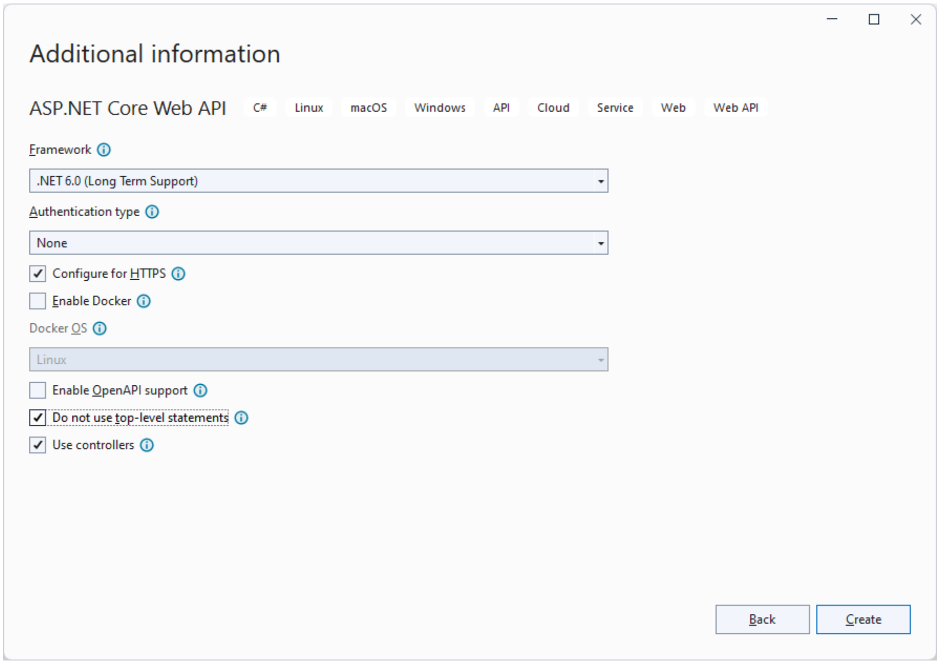
Location: C:\Users\joebl\source\repos\APLNowLLC\RestWebService\

Solution name: RestWebService

☐ Place solution and project in the same directory

Project will be created in "C:\Users\joebl\source\repos\APLNowLLC\RestWebService\RestWebService\RestWebService\"

Provide the additional information


Additional information

ASP.NET Core Web API

Framework: .NET 6.0 (Long Term Support)

Authentication type: None

☑ Configure for HTTPS

☐ Enable Docker

Docker OS: Linux

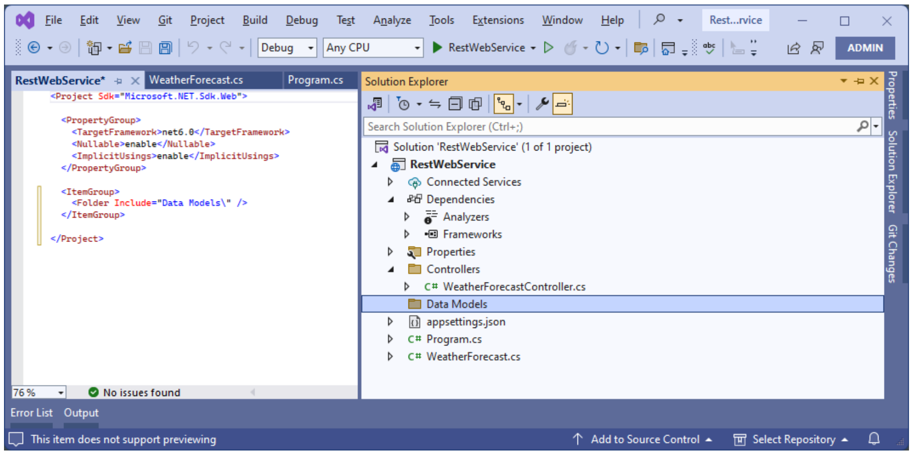
☐ Enable OpenAPI support

☑ Do not use top-level statements

☑ Use controllers

Modify the RestWebService project C# code

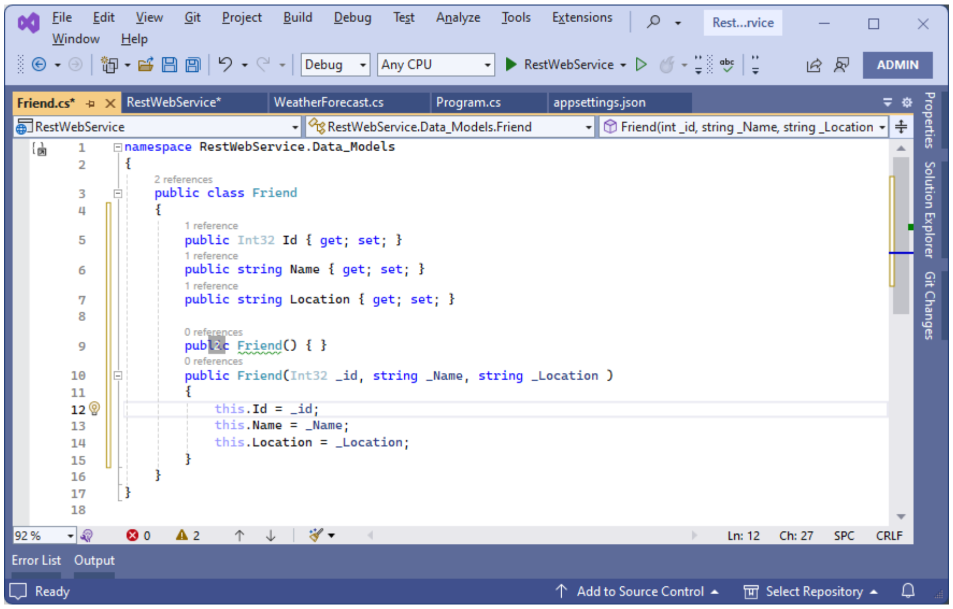
Delete the WeatherForecast.cs code file and add the ‘Data Models’ folder to the project:



```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

  <ItemGroup>
    <Folder Include="Data Models\" />
  </ItemGroup>

</Project>
```

Add the Friend.cs class file to the Data Models folder with the following content:

```
namespace RestWebService.Data_Models
{
    public class Friend
    {
        public Int32 Id { get; set; }
        public string Name { get; set; }
        public string Location { get; set; }

        public Friend() { }
        public Friend(Int32 _id, string _Name, string _Location )
        {
            this.Id = _id;
            this.Name = _Name;
            this.Location = _Location;
        }
    }
}
```

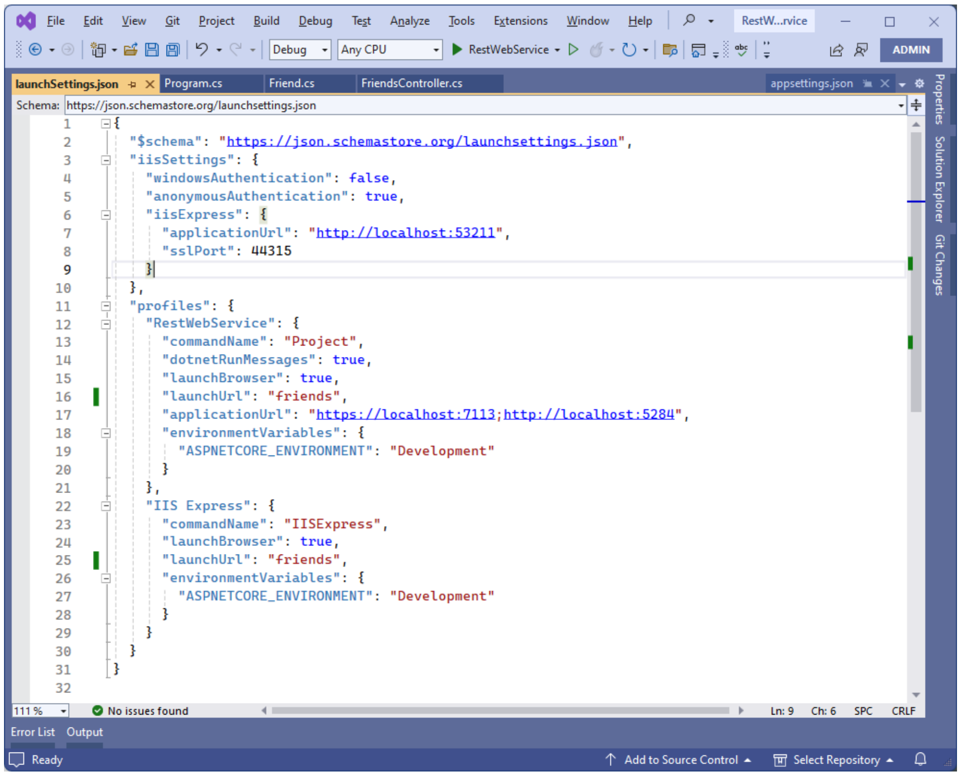


```
namespace RestWebService.Data_Models
{
    2 references
    public class Friend
    {
        1 reference
        public Int32 Id { get; set; }
        1 reference
        public string Name { get; set; }
        1 reference
        public string Location { get; set; }

        0 references
        public Friend() { }
        0 references
        public Friend(Int32 _id, string _Name, string _Location )
        {
            this.Id = _id;
            this.Name = _Name;
            this.Location = _Location;
        }
    }
}
```

The Friend class definition exposes the Id, Name and Location properties of a Friend record. In a production environment, the data models can include more properties and also expose methods.

Modify the LaunchSettings.json file with this content:

```
{
  "$schema": "https://json.schemastore.org/launchsettings.json",
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:53211",
      "sslPort": 44315
    }
  },
  "profiles": {
    "RestWebService": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "launchUrl": "friends",
      "applicationUrl": "https://localhost:7113;http://localhost:5284",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "launchUrl": "friends",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```



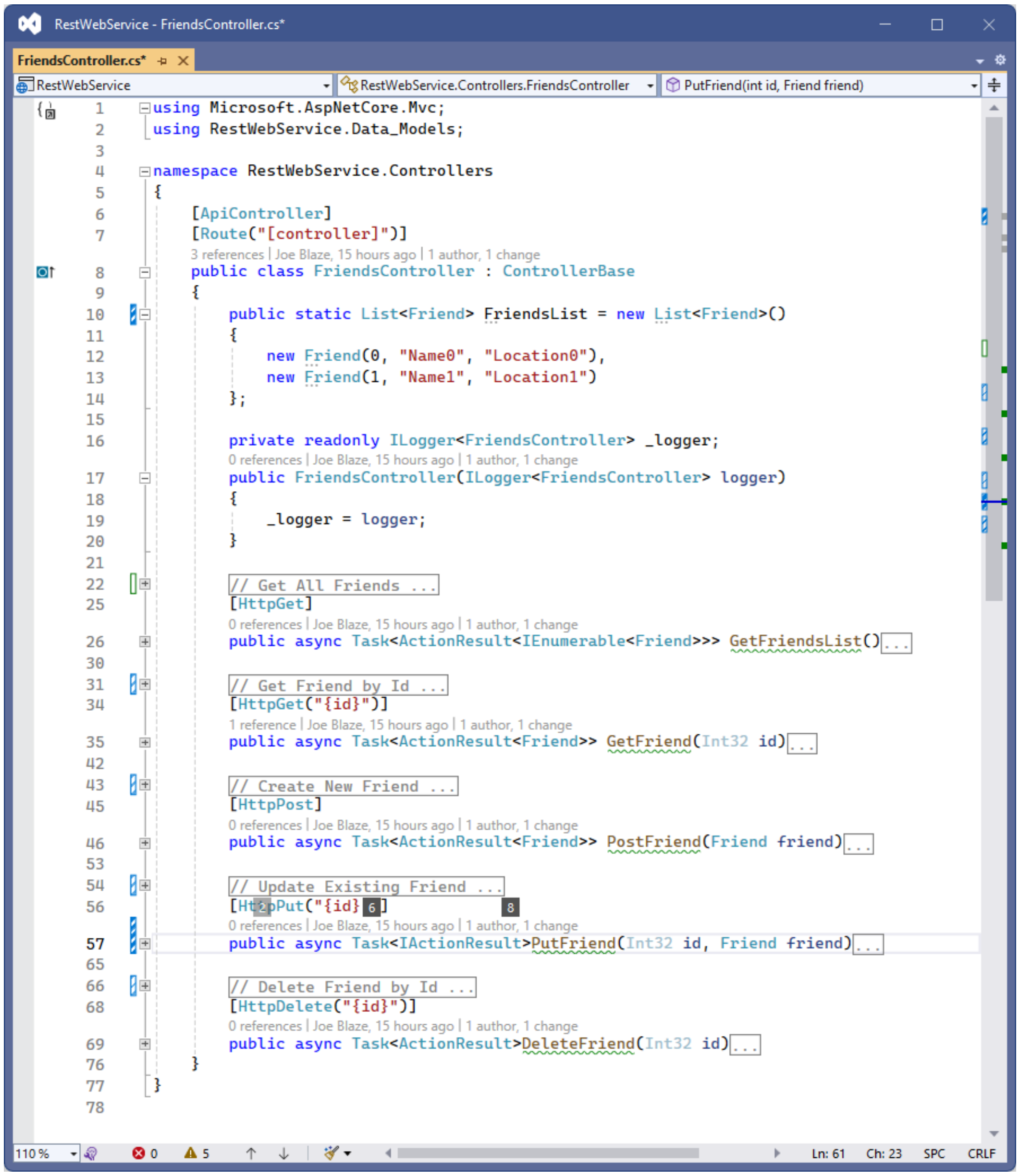
```
{
  "$schema": "https://json.schemastore.org/launchsettings.json",
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:53211",
      "sslPort": 44315
    }
  },
  "profiles": {
    "RestWebService": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "launchUrl": "friends",
      "applicationUrl": "https://localhost:7113;http://localhost:5284",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "launchUrl": "friends",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```

Rename the WeatherForecastController.cs code file to FriendsController.cs and replace the existing content with:

```
using Microsoft.AspNetCore.Mvc;
using RestWebService.Data_Models;

namespace RestWebService.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class FriendsController : ControllerBase
    {
        public static List<Friend> FriendsList = new List<Friend>()
        {
            new Friend(0, "Name0", "Location0"),
            new Friend(1, "Name1", "Location1")
        };

        private readonly ILogger<FriendsController> _logger;
        public FriendsController(ILogger<FriendsController> logger)
        {
            _logger = logger;
        }

        // Get All Friends
        // GET: api/Friends
        // https://localhost:{port#}/api/friends
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Friend>>> GetFriendsList()
        {
            return FriendsList;
        }

        // Get Friend by Id
        // GET: api/Friends/id
        // https://localhost:{port#}/api/friends
        [HttpGet("{id}")]
        public async Task<ActionResult<Friend>> GetFriend(Int32 id)
        {
            if (FriendsList.Count == 0) return NotFound();
            var friend = FriendsList.FirstOrDefault(friend => friend.Id == id);
            if (friend == null) return NotFound();
            return (Friend)friend;
        }

        // Create New Friend
        // POST: api/Friends
        [HttpPost]
        public async Task<ActionResult<Friend>> PostFriend(Friend friend)
        {
            if (FriendsList.Contains(friend))
                return BadRequest("Friend already exists!");
            FriendsList.Add(friend);
            return CreatedAtAction(nameof(GetFriend), new { id = friend.Id },
friend);
        }

        // Update Existing Friend
        // PUT: api/Friends
        [HttpPut("{id}")]
        public async Task<IActionResult>PutFriend(Int32 id, Friend friend)
        {
            if (id != friend.Id) return BadRequest($"friend.Id {friend.Id} does
not match id {id}");
            var index = FriendsList.Select(elt => elt.Id).ToList().IndexOf(id);
            if (index ==  -1) return NotFound();
            FriendsList[index] = friend;
            return NoContent();
        }

        // Delete Friend by Id
        // DELETE: api/Friends
        [HttpDelete("{id}")]
        public async Task<ActionResult>DeleteFriend(Int32 id)
        {
            var friend = FriendsList.FirstOrDefault(elt => elt.Id == id);
            if (friend == null) return NotFound();
            FriendsList.Remove(friend);
            return NoContent();
        }
    }
}
```

When started, the rest web service initializes the FriendsList, which is a .Net Generic List of Friend class instances (records). The running rest web service exposes methods which support the CRUD operations on the FriendsList. In a production environment, the simple FriendsList would be replaced by a robust and secure database.

CRUD Operation	RestWebServiceMethod
Read	GetFriends() and GetFriend()
Create	PostFriend()
Update	PutFriend()
Delete	DeleteFriend()

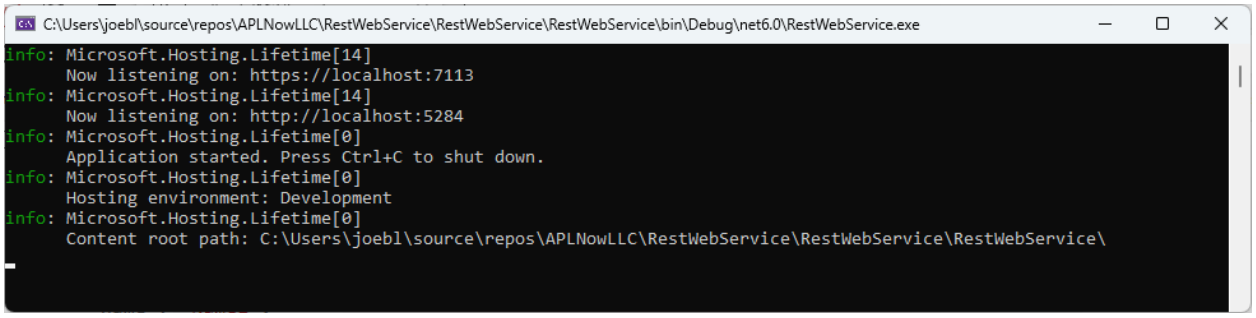


RestWebService - FriendsController.cs*
FriendsController.cs*
RestWebService
RestWebService.Controllers.FriendsController
PutFriend(int id, Friend friend)

```
using Microsoft.AspNetCore.Mvc;
using RestWebService.Data_Models;

namespace RestWebService.Controllers
{
    [ApiController]
    [Route("[controller]")]
    3 references | Joe Blaze, 15 hours ago | 1 author, 1 change
    public class FriendsController : ControllerBase
    {
        public static List<Friend> FriendsList = new List<Friend>()
        {
            new Friend(0, "Name0", "Location0"),
            new Friend(1, "Name1", "Location1")
        };

        private readonly ILogger<FriendsController> _logger;
        0 references | Joe Blaze, 15 hours ago | 1 author, 1 change
        public FriendsController(ILogger<FriendsController> logger)
        {
            _logger = logger;
        }

        // Get All Friends ...
        [HttpGet]
        0 references | Joe Blaze, 15 hours ago | 1 author, 1 change
        public async Task<ActionResult<IEnumerable<Friend>>> GetFriendsList()...

        // Get Friend by Id ...
        [HttpGet("{id}")]
        1 reference | Joe Blaze, 15 hours ago | 1 author, 1 change
        public async Task<ActionResult<Friend>> GetFriend(Int32 id)...

        // Create New Friend ...
        [HttpPost]
        0 references | Joe Blaze, 15 hours ago | 1 author, 1 change
        public async Task<ActionResult<Friend>> PostFriend(Friend friend)...

        // Update Existing Friend ...
        [HttpPut("{id}")]
        0 references | Joe Blaze, 15 hours ago | 1 author, 1 change
        public async Task<IActionResult>PutFriend(Int32 id, Friend friend)...

        // Delete Friend by Id ...
        [HttpDelete("{id}")]
        0 references | Joe Blaze, 15 hours ago | 1 author, 1 change
        public async Task<ActionResult>DeleteFriend(Int32 id)...
    }
}
```

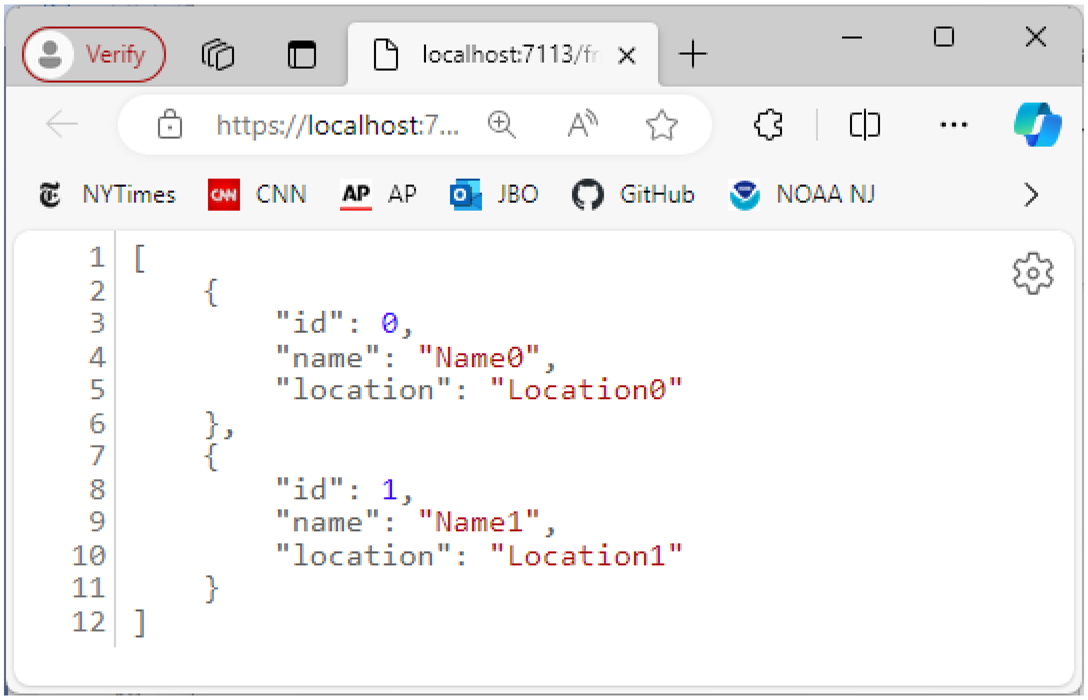
Run the RestWebService project in Visual Studio

The RestWebService http and https endpoints will be presented. The port numbers depend on the workstation environment:



```
C:\Users\joebl\source\repos\APLNowLLC\RestWebService\RestWebService\RestWebService\bin\Debug\net6.0\RestWebService.exe
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7113
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5284
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\joebl\source\repos\APLNowLLC\RestWebService\RestWebService\RestWebService\
```

The RestService web page will be presented in the workstation's default browser, illustrating the results of the ReadAllFriends action in json format:



```
[
    {
        "id": 0,
        "name": "Name0",
        "location": "Location0"
    },
    {
        "id": 1,
        "name": "Name1",
        "location": "Location1"
    }
]
```

Create an APL64 HttpClient for the RestWebService

The APL64 HttpClient and the RestWebService share knowledge of the Friend class definition.

Prepare the Client-side Components of the HttpClient

The data structures of the HttpClient and RestWebService are not directly consumable by APL64. The applicable, open source, .Net assemblies must be obtained to facilitate the exchange of data between APL64 and .Net.

Create the 'c:\APL64 Http Client' folder on the target workstation

Obtain the .Net json Serialization Tools

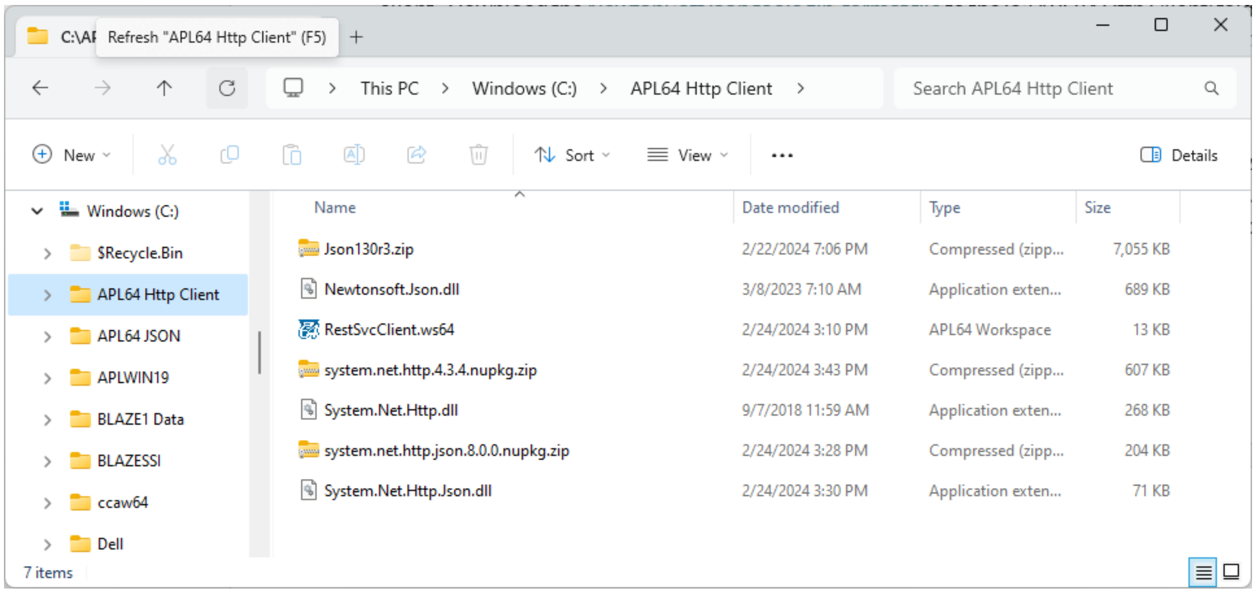
The rest web service uses json-format data to receive and transmit data between the server and a client. Download the NewtonSoft json tools zip-format file to the 'c:\APL64 Http Client' folder. Modify the properties of the zip-format file to unblock it. Copy the 'Newtonsoft.Json.dll' assembly file from the zip-format file to the 'c:\APL64 Http Client' folder.

Obtain the .Net HttpClient Tools

- Download the System.Net.Http Nuget package to the 'c:\APL64 Http Client' folder. Add a '.zip' file extension to the filename. Modify the properties of the zip-format file to unblock it. Copy the 'System.Net.Http.dll' assembly file from the '\runtimes\win\lib\netstandard1.3\' folder to the 'c:\APL64 Http Client' folder.
- Download the System.Net.Http.Json Nuget package to the 'c:\APL64 Http Client' folder. Add a '.zip' file extension to the filename. Modify the properties of the zip-format file to unblock it. Copy the 'System.Net.Http.Json.dll' assembly file from the '\lib\netstandard2.0' folder to the 'c:\APL64 Http Client' folder.

Resulting 'c:\APL64 Http Client' Folder

The 'c:\APL64 Http Client' folder should contain these files:

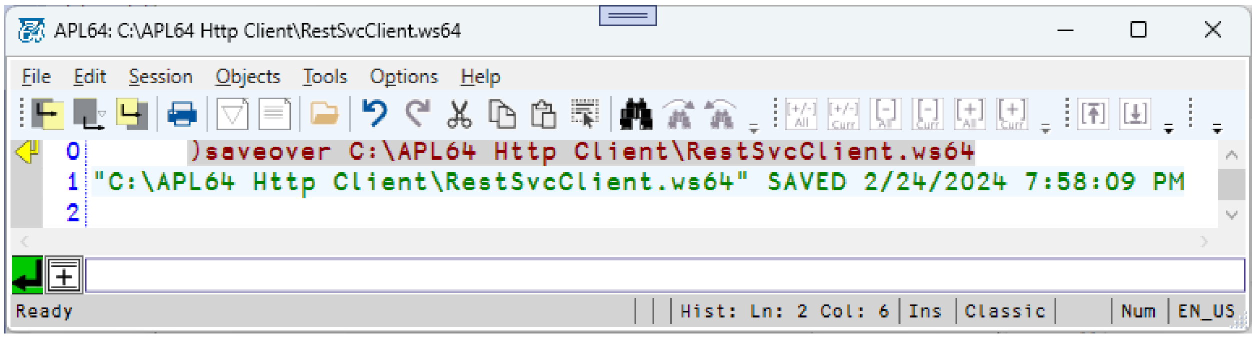


Name	Date modified	Type	Size
Json130r3.zip	2/22/2024 7:06 PM	Compressed (zipp...	7,055 KB
Newtonsoft.Json.dll	3/8/2023 7:10 AM	Application exten...	689 KB
RestSvcClient.ws64	2/24/2024 3:10 PM	APL64 Workspace	13 KB
system.net.http.4.3.4.nupkg.zip	2/24/2024 3:43 PM	Compressed (zipp...	607 KB
System.Net.Http.dll	9/7/2018 11:59 AM	Application exten...	268 KB
system.net.http.json.8.0.0.nupkg.zip	2/24/2024 3:28 PM	Compressed (zipp...	204 KB
System.Net.Http.Json.dll	2/24/2024 3:30 PM	Application exten...	71 KB

Use □cse to Create HttpClient in APL64

Start an instance of the APL64 Developer version

Name the workspace RestSvcClient and save it to the 'c:\APL64 Http Client' folder:



```
APL64: C:\APL64 Http Client\RestSvcClient.ws64
File  Edit  Session  Objects  Tools  Options  Help
0        )saveover C:\APL64 Http Client\RestSvcClient.ws64
1 "C:\APL64 Http Client\RestSvcClient.ws64" SAVED 2/24/2024 7:58:09 PM
2
Ready                    Hist: Ln: 2 Col: 6 | Ins | Classic |   Num | EN_US
```

Create a ⎕cse Script Variable

Open an APL64 variable editor with name ClientScript and this content:

```
public class Friend
{
    public Int32 Id { get; set; }
    public string Name { get; set; }
    public string Location { get; set; }

    public Friend() { }
    public Friend(Int32 _id, string _Name, string _Location)
    {
        this.Id = _id;
        this.Name = _Name;
        this.Location = _Location;
    }

    public static string SerializeToJson(Friend friend)
    {
        return JsonConvert.SerializeObject(friend);
    }
    public static Friend DeserializeFromJson(string json)
    {
        return JsonConvert.DeserializeObject<Friend>(json);
    }
}

public class HttpClientForRestSvc
{
    private HttpClient client;
    public string errMsg = "";
    public bool hasError;
    public string jsonResponse = "";
    public string returnUrl = "";

    public HttpClientForRestSvc() { }

    public void CreateHttpClient(TimeSpan httpClientTimeout, string baseAddress)
        {
            client = new HttpClient();
            client.Timeout = httpClientTimeout;
            client.BaseAddress = new Uri(baseAddress);
            client.DefaultRequestHeaders.Accept.Clear();
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));
```

```
        }

    public void DisposeHttpClient()
        {
            if (client != null) { client.Dispose(); }
        }

    public async Task GetAllFriends()
    {
        hasError = true;
        errMsg = string.Empty;
        jsonResponse = string.Empty;
        try
        {
            var response = await client.GetAsync("friends");
            response.EnsureSuccessStatusCode();
            if (!response.IsSuccessStatusCode)
                errMsg = $"Status Code: {response.StatusCode} Reason
Phrase:{response.ReasonPhrase}";
            else
            {
                hasError = false;
                jsonResponse = await response.Content.ReadAsStringAsync();
            }
        }
        catch (OperationCanceledException ex) when (ex.InnerException is
TimeoutException tex)
        {
            errMsg = $"Timed out: {ex.Message}, {tex.Message}";
            return;
        }
        catch (OperationCanceledException ex)
        {
            if (ex.InnerException != null)
                errMsg = $"{ex.Message}, {ex.InnerException.Message}";
            else
                errMsg = ex.Message;
            return;
        }
        catch (Exception ex)
        {
            errMsg = ex.Message;
            return;
        }
    }
    public async Task GetFriend(Int32 id)
        {
            hasError = true;
            errMsg = string.Empty;
            jsonResponse = string.Empty;
            try
            {
                var response = await client.GetAsync($"friends/{id}");
                response.EnsureSuccessStatusCode();
                if (!response.IsSuccessStatusCode)
                    errMsg = $"Status Code: {response.StatusCode} Reason
Phrase:{response.ReasonPhrase}";
                else
```

```
                 {
                     hasError = false;
                     jsonResponse = await response.Content.ReadAsStringAsync();
                 }
             }
             catch (OperationCanceledException ex) when (ex.InnerException is
TimeoutException tex)
             {
                 errMsg = $"Timed out: {ex.Message}, {tex.Message}";
                 return;
             }
             catch (OperationCanceledException ex)
             {
                 if (ex.InnerException != null)
                     errMsg = $"{ex.Message}, {ex.InnerException.Message}";
                 else
                     errMsg = ex.Message;
                 return;
             }
             catch (Exception ex)
             {
                 errMsg = ex.Message;
                 return;
             }
         }

     public async Task CreateFriend(Int32 id, string name, string location)
         {
             hasError = true;
             errMsg = string.Empty;
             jsonResponse = string.Empty;
             var friend = new Friend(id, name, location);
             try
             {
                 var response = await client.PostAsJsonAsync($"Friends", friend);
                 response.EnsureSuccessStatusCode();
                 if (!response.IsSuccessStatusCode)
                     errMsg = $"Status Code: {response.StatusCode} Reason
Phrase:{response.ReasonPhrase}";
                 else
                 {
                     hasError = false;
                     var todo = await response.Content.ReadFromJsonAsync<Friend>();
                     jsonResponse = await response.Content.ReadAsStringAsync();
                 }
             }
             catch (OperationCanceledException ex) when (ex.InnerException is
TimeoutException tex)
             {
                 errMsg = $"Timed out: {ex.Message}, {tex.Message}";
                 return;
             }
             catch (OperationCanceledException ex)
             {
                 if (ex.InnerException != null)
                     errMsg = $"{ex.Message}, {ex.InnerException.Message}";
                 else
                     errMsg = ex.Message;
```

```
                return;
            }
            catch (Exception ex)
            {
                errMsg = ex.Message;
                return;
            }
        }

    public async Task UpdateFriend(Int32 id, string name, string location)
        {
            hasError = true;
            errMsg = string.Empty;
            jsonResponse = string.Empty;
            var friend = new Friend(id, name, location);
            try
            {
                var response = await client.PutAsJsonAsync($"friends/{id}",
friend);
                response.EnsureSuccessStatusCode();
                if (response.IsSuccessStatusCode) hasError = false;
            }
            catch (OperationCanceledException ex) when (ex.InnerException is
TimeoutException tex)
            {
                errMsg = $"Timed out: {ex.Message}, {tex.Message}";
                return;
            }
            catch (OperationCanceledException ex)
            {
                if (ex.InnerException != null)
                    errMsg = $"{ex.Message}, {ex.InnerException.Message}";
                else
                    errMsg = ex.Message;
                return;
            }
            catch (Exception ex)
            {
                errMsg = ex.Message;
                return;
            }
        }

    public async Task DeleteFriend(Int32 id)
        {
            hasError = true;
            errMsg = string.Empty;
            jsonResponse = string.Empty;
            try
            {
                var response = await client.DeleteAsync($"friends/{id}");
                response.EnsureSuccessStatusCode();
                if (response.IsSuccessStatusCode) hasError = false;
            }
            catch (OperationCanceledException ex) when (ex.InnerException is
TimeoutException tex)
            {
                errMsg = $"Timed out: {ex.Message}, {tex.Message}";
                return;
            }
            catch (OperationCanceledException ex)
            {
                if (ex.InnerException != null)
                    errMsg = $"{ex.Message}, {ex.InnerException.Message}";
                else
                    errMsg = ex.Message;
                return;
            }
            catch (Exception ex)
            {
                errMsg = ex.Message;
                return;
            }
        }
}
```

When the ClientScript is executed by ⎕cse system function, the Friend and `HttpClientForRestSvc` classes will be defined in the ⎕cse instance.

Friend class definition

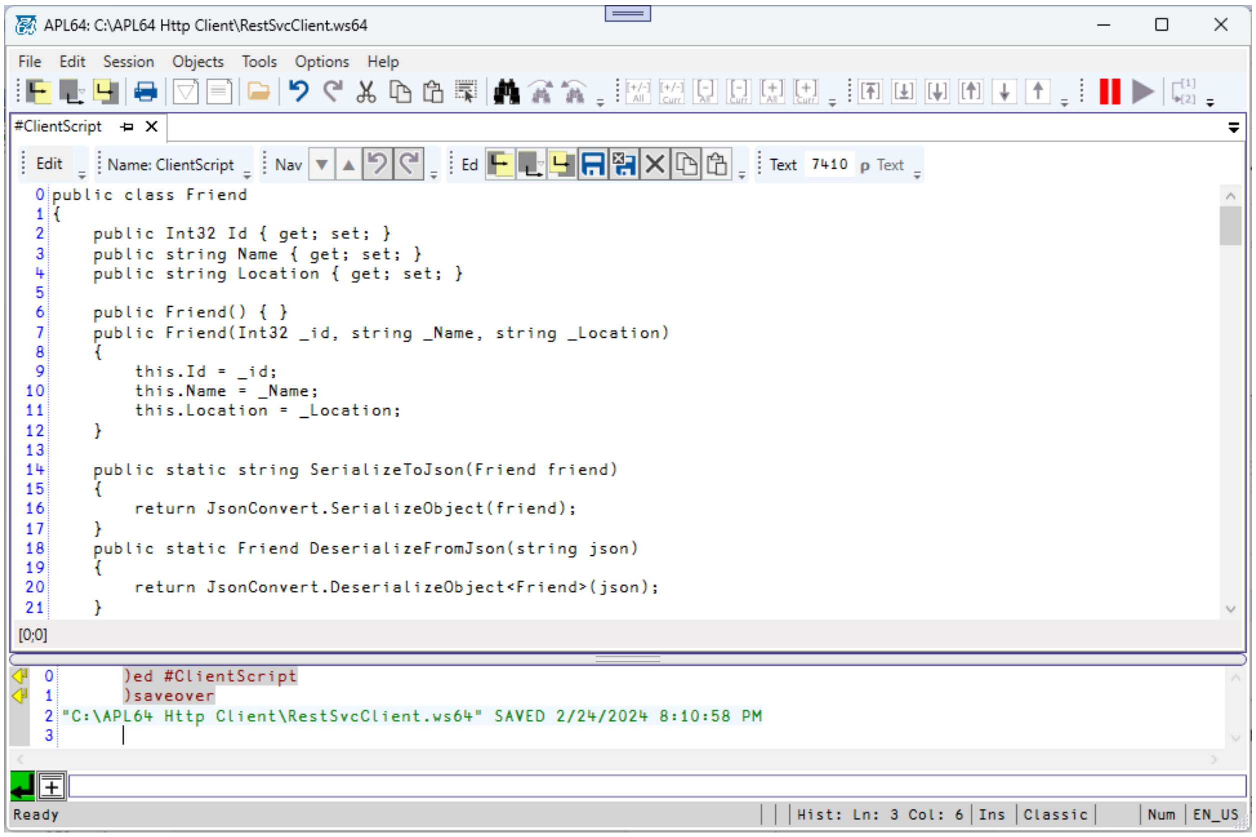
This is the same definition as the Friend class in the RestWebService, except that two 'helper' methods, SerializeToJson() and DeserializeFromJson(), have been added to the HttpClient version of the Friend class to facilitate the consumption of json- format data from the HttpClient and RestWebService to be conveniently consumed by APL64.

HttpClientForRestSvc class definition

- Properties used when accessing the RestWebService
    - `HttpClient client`
    - `string errMsg`
    - `bool hasError`
    - `jsonResponse`

- The .Net HttpClient is created using the `CreateHttpClient() method with arguments for httpClientTimeout and baseAddress.`
- HttpClientMethods to request the services exposed by the RestWebService
    - `GetAllFriends()`
    - GetFriend(Int32 id)
    - CreateFriend(Int32 id, string name, string location)
    - UpdateFriend(Int32 id, string name, string location)
    - DeleteFriend(Int32 id)
- One instance of the HttpClient may be used to submit any number of requests to the RestWebService. After all such requests have been made and resolved, the HttpClient instance should be 'disposed', using the `DisposeHttpClient() method`.

Save the ClientScript variable and save the workspace



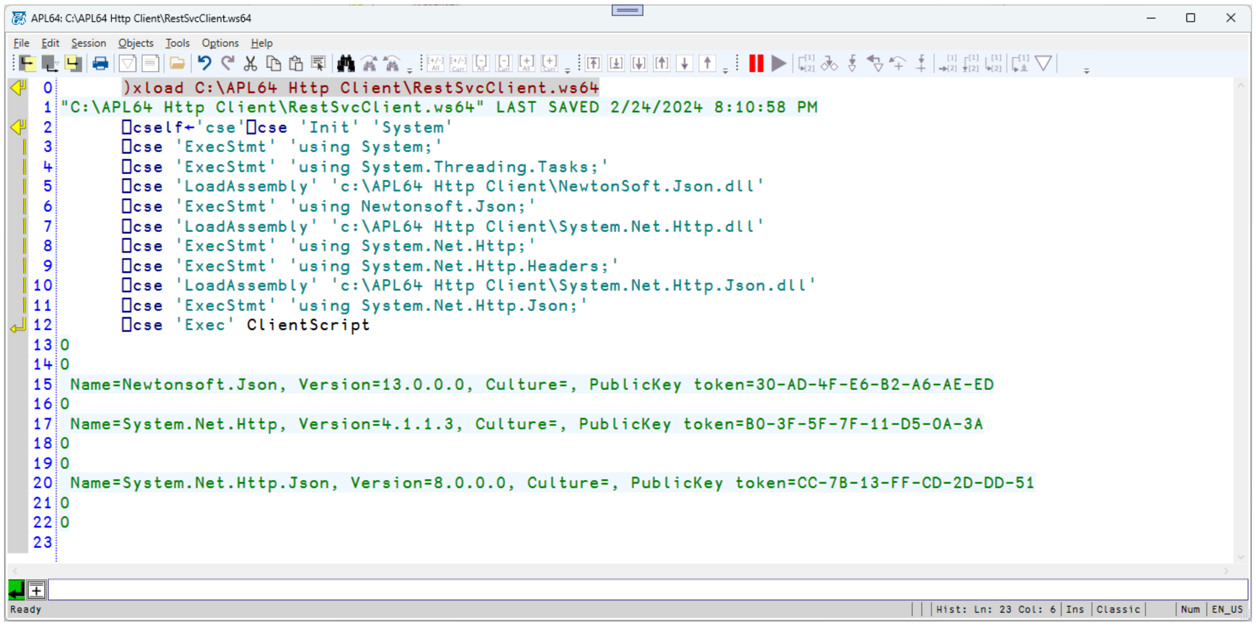
```
public class Friend
{
    public Int32 Id { get; set; }
    public string Name { get; set; }
    public string Location { get; set; }

    public Friend() { }
    public Friend(Int32 _id, string _Name, string _Location)
    {
        this.Id = _id;
        this.Name = _Name;
        this.Location = _Location;
    }

    public static string SerializeToJson(Friend friend)
    {
        return JsonConvert.SerializeObject(friend);
    }
    public static Friend DeserializeFromJson(string json)
    {
        return JsonConvert.DeserializeObject<Friend>(json);
    }
```

```
      )ed #ClientScript
      )saveover
"C:\APL64 Http Client\RestSvcClient.ws64" SAVED 2/24/2024 8:10:58 PM
```

Create a C# Script Engine Instance

Execute the following ⎕cse statements to load the required .Net assemblies:

```
⎕cself←'cse'⎕cse 'Init' 'System'
⎕cse 'ExecStmt' 'using System;'
⎕cse 'ExecStmt' 'using System.Threading.Tasks;'
⎕cse 'LoadAssembly' 'c:\APL64 Http Client\NewtonSoft.Json.dll'
⎕cse 'ExecStmt' 'using Newtonsoft.Json;'
⎕cse 'LoadAssembly' 'c:\APL64 Http Client\System.Net.Http.dll'
⎕cse 'ExecStmt' 'using System.Net.Http;'
⎕cse 'ExecStmt' 'using System.Net.Http.Headers;'
⎕cse 'LoadAssembly' 'c:\APL64 Http Client\System.Net.Http.Json.dll'
⎕cse 'ExecStmt' 'using System.Net.Http.Json;'
⎕cse 'Exec' ClientScript
```

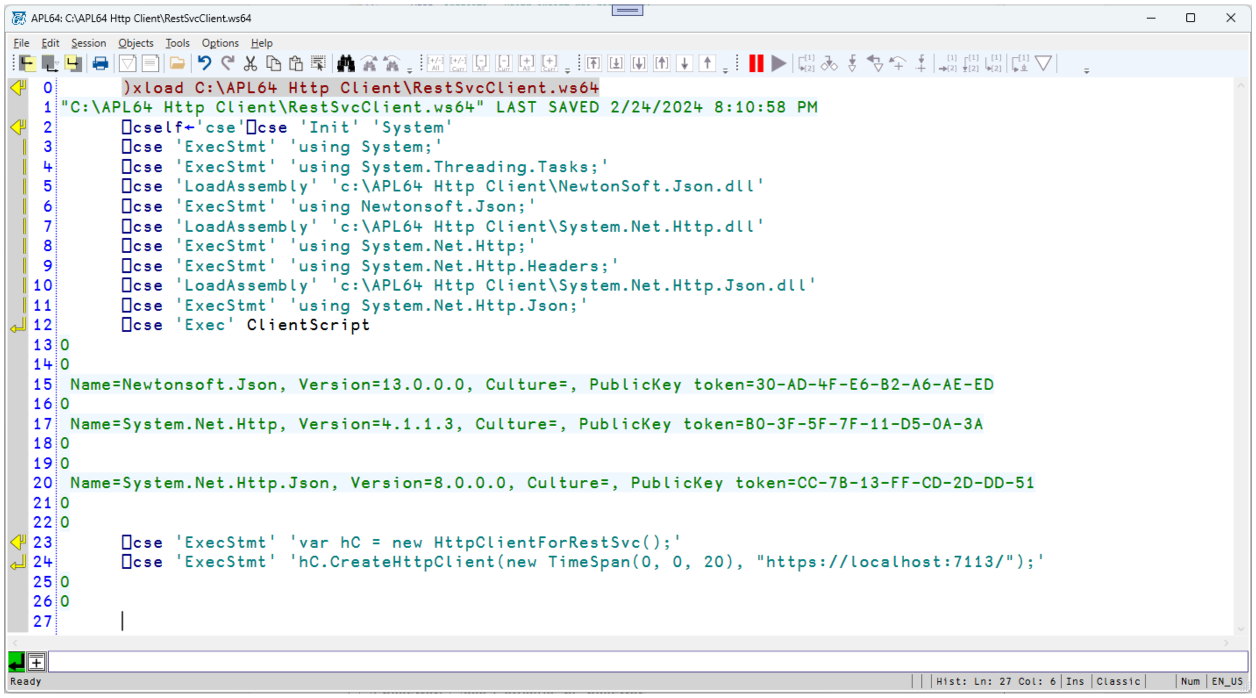

APL64: C:\APL64 Http Client\RestSvcClient.ws64
File Edit Session Objects Tools Options Help
Ready
Hist: Ln: 23 Col: 6 Ins Classic Num EN_US

```
      )xload C:\APL64 Http Client\RestSvcClient.ws64
"C:\APL64 Http Client\RestSvcClient.ws64" LAST SAVED 2/24/2024 8:10:58 PM
      ⎕cself←'cse'⎕cse 'Init' 'System'
      ⎕cse 'ExecStmt' 'using System;'
      ⎕cse 'ExecStmt' 'using System.Threading.Tasks;'
      ⎕cse 'LoadAssembly' 'c:\APL64 Http Client\NewtonSoft.Json.dll'
      ⎕cse 'ExecStmt' 'using Newtonsoft.Json;'
      ⎕cse 'LoadAssembly' 'c:\APL64 Http Client\System.Net.Http.dll'
      ⎕cse 'ExecStmt' 'using System.Net.Http;'
      ⎕cse 'ExecStmt' 'using System.Net.Http.Headers;'
      ⎕cse 'LoadAssembly' 'c:\APL64 Http Client\System.Net.Http.Json.dll'
      ⎕cse 'ExecStmt' 'using System.Net.Http.Json;'
      ⎕cse 'Exec' ClientScript
0
0
 Name=Newtonsoft.Json, Version=13.0.0.0, Culture=, PublicKey token=30-AD-4F-E6-B2-A6-AE-ED
0
 Name=System.Net.Http, Version=4.1.1.3, Culture=, PublicKey token=B0-3F-5F-7F-11-D5-0A-3A
0
0
 Name=System.Net.Http.Json, Version=8.0.0.0, Culture=, PublicKey token=CC-7B-13-FF-CD-2D-DD-51
0
0
```

Create an HttpClient in the ⎕cse instance

Execute the following ⎕cse statements to create the HttpClient:

```
⎕cse 'ExecStmt' 'var hC = new HttpClientForRestSvc();'
⎕cse 'ExecStmt' 'hC.CreateHttpClient(new TimeSpan(0, 0, 20), "https://localhost:7113/");'
```


APL64: C:\APL64 Http Client\RestSvcClient.ws64
File Edit Session Objects Tools Options Help
Ready
Hist: Ln: 27 Col: 6 Ins Classic Num EN_US

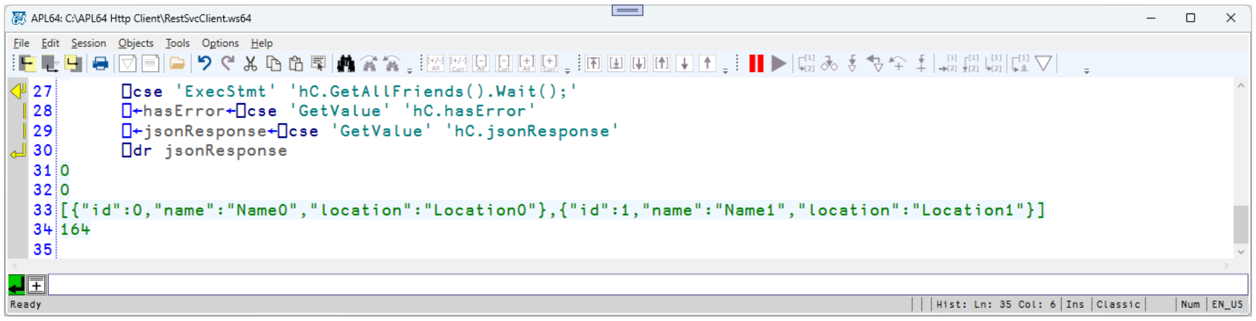
```
      )xload C:\APL64 Http Client\RestSvcClient.ws64
"C:\APL64 Http Client\RestSvcClient.ws64" LAST SAVED 2/24/2024 8:10:58 PM
      ⎕cself←'cse'⎕cse 'Init' 'System'
      ⎕cse 'ExecStmt' 'using System;'
      ⎕cse 'ExecStmt' 'using System.Threading.Tasks;'
      ⎕cse 'LoadAssembly' 'c:\APL64 Http Client\NewtonSoft.Json.dll'
      ⎕cse 'ExecStmt' 'using Newtonsoft.Json;'
      ⎕cse 'LoadAssembly' 'c:\APL64 Http Client\System.Net.Http.dll'
      ⎕cse 'ExecStmt' 'using System.Net.Http;'
      ⎕cse 'ExecStmt' 'using System.Net.Http.Headers;'
      ⎕cse 'LoadAssembly' 'c:\APL64 Http Client\System.Net.Http.Json.dll'
      ⎕cse 'ExecStmt' 'using System.Net.Http.Json;'
      ⎕cse 'Exec' ClientScript
0
0
 Name=Newtonsoft.Json, Version=13.0.0.0, Culture=, PublicKey token=30-AD-4F-E6-B2-A6-AE-ED
0
 Name=System.Net.Http, Version=4.1.1.3, Culture=, PublicKey token=B0-3F-5F-7F-11-D5-0A-3A
0
0
 Name=System.Net.Http.Json, Version=8.0.0.0, Culture=, PublicKey token=CC-7B-13-FF-CD-2D-DD-51
0
0
      ⎕cse 'ExecStmt' 'var hC = new HttpClientForRestSvc();'
      ⎕cse 'ExecStmt' 'hC.CreateHttpClient(new TimeSpan(0, 0, 20), "https://localhost:7113/");'
0
0
```

Use the APL64 HttpClient to Access the Rest Web Service

Obtain List of All Current Friend Records on the Server

Execute the following ⎕cse statements to obtain the current list of Friend records on the server. The jsonResponse is composed of two Friend records in a json-format string scalar. Each of them may be deserialized from json strings into separate Friend records.

```
⎕cse 'ExecStmt' 'hC.GetAllFriends().Wait();'
⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
⎕←jsonResponse←⎕cse 'GetValue' 'hC.jsonResponse'
⎕dr jsonResponse
```



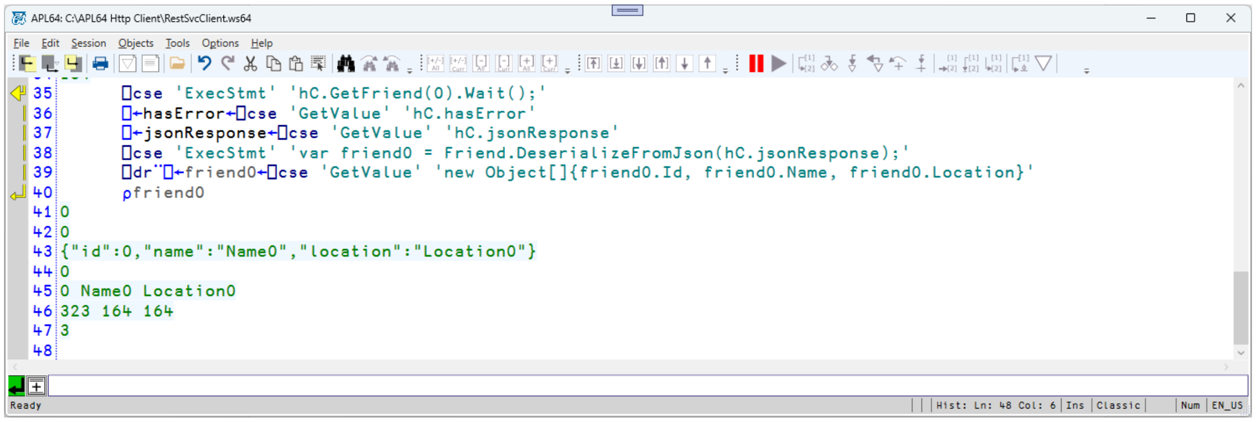
APL64: C:\APL64 Http Client\RestSvcClient.ws64
File Edit Session Objects Tools Options Help
Ready
Hist: Ln: 35 Col: 6 Ins Classic Num EN_US

```
      ⎕cse 'ExecStmt' 'hC.GetAllFriends().Wait();'
      ⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
      ⎕←jsonResponse←⎕cse 'GetValue' 'hC.jsonResponse'
      ⎕dr jsonResponse
0
0
[{"id":0,"name":"Name0","location":"Location0"},{"id":1,"name":"Name1","location":"Location1"}]
164
```

Obtain the Property Values of an Existing Friend Record on the Server

Execute the following ⎕cse statements to deserialize the jsonResponse to a Friend record and return the record values as an object vector which is conveniently consumed by APL64:

```
⎕cse 'ExecStmt' 'hC.GetFriend(0).Wait();'
⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
⎕←jsonResponse←⎕cse 'GetValue' 'hC.jsonResponse'
⎕cse 'ExecStmt' 'var friend0 = Friend.DeserializeFromJson(hC.jsonResponse);'
⎕dr¨⎕←friend0←⎕cse 'GetValue' 'new Object[]{friend0.Id, friend0.Name, friend0.Location}'
⍴friend0
```



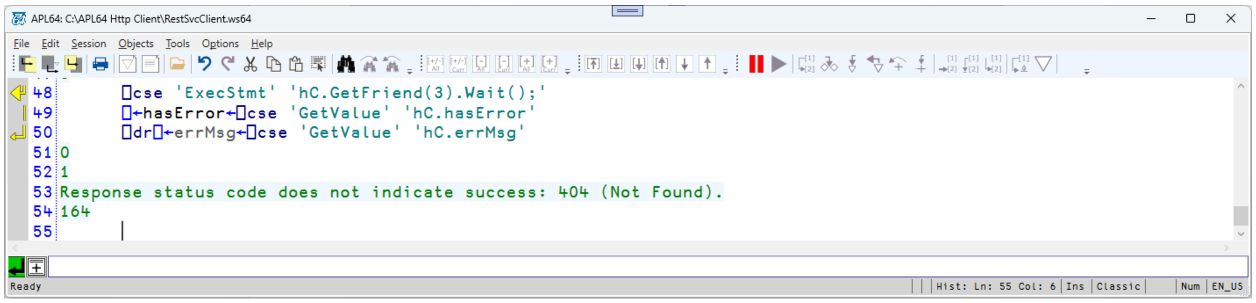
APL64: C:\APL64 Http Client\RestSvcClient.ws64
File Edit Session Objects Tools Options Help
Ready
Hist: Ln: 48 Col: 6 Ins Classic Num EN_US

```
      ⎕cse 'ExecStmt' 'hC.GetFriend(0).Wait();'
      ⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
      ⎕←jsonResponse←⎕cse 'GetValue' 'hC.jsonResponse'
      ⎕cse 'ExecStmt' 'var friend0 = Friend.DeserializeFromJson(hC.jsonResponse);'
      ⎕dr¨⎕←friend0←⎕cse 'GetValue' 'new Object[]{friend0.Id, friend0.Name, friend0.Location}'
      ⍴friend0
0
0
{"id":0,"name":"Name0","location":"Location0"}
0
0 Name0 Location0
323 164 164
3
```

Handling Exceptions Returned by the Server

If the requested Friend record does not exist in the server-side data:

```
⎕cse 'ExecStmt' 'hC.GetFriend(3).Wait();'
⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
⎕dr⎕←errMsg←⎕cse 'GetValue' 'hC.errMsg'
```

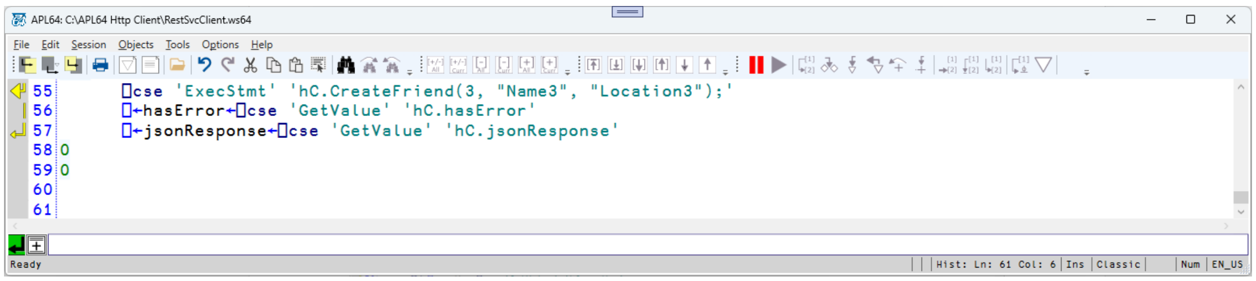

APL64: C:\APL64 Http Client\RestSvcClient.ws64
File Edit Session Objects Tools Options Help
Ready
Hist: Ln: 55 Col: 6
Ins
Classic
Num
EN_US

```
      ⎕cse 'ExecStmt' 'hC.GetFriend(3).Wait();'
      ⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
      ⎕dr⎕←errMsg←⎕cse 'GetValue' 'hC.errMsg'
0
1
Response status code does not indicate success: 404 (Not Found).
164
```

Create New Friend Record on the Server

Execute the following ⎕cse statements to create a new Friend record. The CreateFriend() method serializes the APL64 values to a new Friend record and submits it to the web server when the CreateFriend request is made.

```
⎕cse 'ExecStmt' 'hC.CreateFriend(3, "Name3", "Location3");'
⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
⎕←jsonResponse←⎕cse 'GetValue' 'hC.jsonResponse'
```

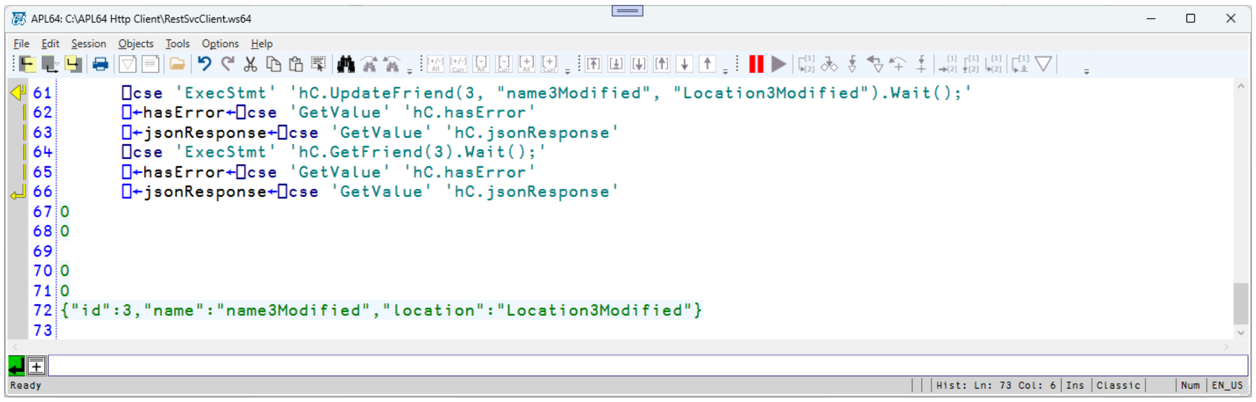

APL64: C:\APL64 Http Client\RestSvcClient.ws64
File Edit Session Objects Tools Options Help
Ready
Hist: Ln: 61 Col: 6
Ins
Classic
Num
EN_US

```
      ⎕cse 'ExecStmt' 'hC.CreateFriend(3, "Name3", "Location3");'
      ⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
      ⎕←jsonResponse←⎕cse 'GetValue' 'hC.jsonResponse'
0
0
```

Update an Existing Friend Record on the Server

Execute the following ⎕cse statements to update an existing Friend record. For this example, the UpdateFriend() method has an empty string scalar 'jsonReponse' result.

```
⎕cse 'ExecStmt' 'hC.UpdateFriend(3, "name3Modified", "Location3Modified").Wait();'
⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
⎕←jsonResponse←⎕cse 'GetValue' 'hC.jsonResponse'
⎕cse 'ExecStmt' 'hC.GetFriend(3).Wait();'
⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
⎕←jsonResponse←⎕cse 'GetValue' 'hC.jsonResponse'
```

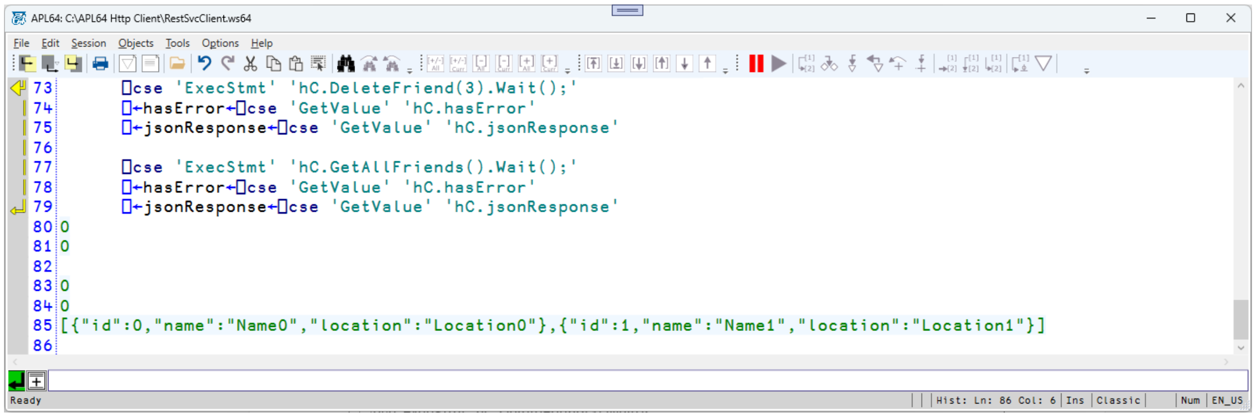


```
        ⎕cse 'ExecStmt' 'hC.UpdateFriend(3, "name3Modified", "Location3Modified").Wait();'
        ⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
        ⎕←jsonResponse←⎕cse 'GetValue' 'hC.jsonResponse'
        ⎕cse 'ExecStmt' 'hC.GetFriend(3).Wait();'
        ⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
        ⎕←jsonResponse←⎕cse 'GetValue' 'hC.jsonResponse'
0
0

0
0
{"id":3,"name":"name3Modified","location":"Location3Modified"}
```

Delete an Existing Friend Record on the Server

Execute the following ⎕cse statements to delete a Friend record:

```
⎕cse 'ExecStmt' 'hC.DeleteFriend(3).Wait();'
⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
⎕←jsonResponse←⎕cse 'GetValue' 'hC.jsonResponse'

⎕cse 'ExecStmt' 'hC.GetAllFriends().Wait();'
⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
⎕←jsonResponse←⎕cse 'GetValue' 'hC.jsonResponse'
```



```
        ⎕cse 'ExecStmt' 'hC.DeleteFriend(3).Wait();'
        ⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
        ⎕←jsonResponse←⎕cse 'GetValue' 'hC.jsonResponse'

        ⎕cse 'ExecStmt' 'hC.GetAllFriends().Wait();'
        ⎕←hasError←⎕cse 'GetValue' 'hC.hasError'
        ⎕←jsonResponse←⎕cse 'GetValue' 'hC.jsonResponse'
0
0

0
0
[{"id":0,"name":"Name0","location":"Location0"},{"id":1,"name":"Name1","location":"Location1"}]
```

Learn More

The rest web service example described in this document is very simplified so that basic concepts are emphasized. For production use, a rest web server and an HttpClient must incorporate client authentication, a responsive and resilient data base and other important features.

To obtain an APL64 subscription or for customized consulting, contact sales@apl2000.com

For APL64 subscribers, contact support@apl2000.com for technical assistance.