<https://www.tutorialsteacher.com/webapi/consuming-web-api-in-dotnet-using-httpclient>

Install-Package Microsoft.AspNet.WebApi.Client

public class Student

{

public int Id { get; set; }

public string Name { get; set; }

}

**GET**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Net.Http;

using System.Net.Http.Headers;

namespace HttpClientDemo

{

class Program

{

static void Main(string[] args)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:60464/api/");

//HTTP GET

var responseTask = client.GetAsync("student");

responseTask.Wait();

var result = responseTask.Result;

if (result.IsSuccessStatusCode)

{

var readTask = result.Content.ReadAsAsync<Student[]>();

readTask.Wait();

var students = readTask.Result;

foreach (var student in students)

{

Console.WriteLine(student.Name);

}

}

}

Console.ReadLine();

}

}

}

**POST**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Net.Http;

using System.Net.Http.Headers;

namespace HttpClientDemo

{

class Program

{

static void Main(string[] args)

{

var student = new Student() { Name = "Steve" };

var postTask = client.PostAsJsonAsync<Student>("student", student);

postTask.Wait();

var result = postTask.Result;

if (result.IsSuccessStatusCode)

{

var readTask = result.Content.ReadAsAsync<Student>();

readTask.Wait();

var insertedStudent = readTask.Result;

Console.WriteLine("Student {0} inserted with id: {1}", insertedStudent.Name, insertedStudent.Id);

}

else

{

Console.WriteLine(result.StatusCode);

}

}

}

}

The following table lists all the methods of HttpClient to send different HTTP requests.

| Method Name | Description |
| --- | --- |
| GetAsync | Sends a GET request to the specified Uri as an asynchronous operation. |
| GetByteArrayAsync | Sends a GET request to the specified Uri and returns the response body as a byte array in an asynchronous operation. |
| GetStreamAsync | Sends a GET request to the specified Uri and returns the response body as a stream in an asynchronous operation. |
| GetStringAsync | Sends a GET request to the specified Uri and returns the response body as a string in an asynchronous operation. |
| PostAsync | Sends a POST request to the specified Uri as an asynchronous operation. |
| PostAsJsonAsync | Sends a POST request as an asynchronous operation to the specified Uri with the given value serialized as JSON. |
| PostAsXmlAsync | Sends a POST request as an asynchronous operation to the specified Uri with the given value serialized as XML. |
| PutAsync | Sends a PUT request to the specified Uri as an asynchronous operation. |
| PutAsJsonAsync | Sends a PUT request as an asynchronous operation to the specified Uri with the given value serialized as JSON. |
| PutAsXmlAsync | Sends a PUT request as an asynchronous operation to the specified Uri with the given value serialized as XML. |
| DeleteAsync | Sends a DELETE request to the specified Uri as an asynchronous operation. |

CONTROLLER

Example: Web API Controller

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace MyWebAPI.Controller

{

public class StudentController : ApiController

{

public IHttpActionResult GetAllStudents(bool includeAddress = false)

{

IList<StudentViewModel> students = null;

using (var ctx = new SchoolDBEntities())

{

students= ctx.Students.Include("StudentAddress").Select(s => new StudentViewModel()

{

Id = s.StudentID,

FirstName = s.FirstName,

LastName = s.LastName,

Address = s.StudentAddress == null ||

includeAddress == false ? null : new AddressViewModel()

{

StudentId = s.StudentAddress.StudentID,

Address1 = s.StudentAddress.Address1,

Address2 = s.StudentAddress.Address2,

City = s.StudentAddress.City,

State = s.StudentAddress.State

}

}).ToList<StudentViewModel>();

}

if (students.Count == 0)

{

return NotFound();

}

return Ok(students);

}

public IHttpActionResult PostNewStudent(StudentViewModel student)

{

if (!ModelState.IsValid)

return BadRequest("Not a valid data");

using (var ctx = new SchoolDBEntities())

{

ctx.Students.Add(new Student()

{

StudentID = student.Id,

FirstName = student.FirstName,

LastName = student.LastName

});

ctx.SaveChanges();

}

return Ok();

}

public IHttpActionResult Put(StudentViewModel student)

{

if (!ModelState.IsValid)

return BadRequest("Not a valid data");

using (var ctx = new SchoolDBEntities())

{

var existingStudent=

ctx.Students.Where(s=>s.StudentID== student.Id).FirstOrDefault<Student>();

if (existingStudent != null)

{

existingStudent.FirstName = student.FirstName;

existingStudent.LastName = student.LastName;

ctx.SaveChanges();

}

else

{

return NotFound();

}

}

return Ok();

}

public IHttpActionResult Delete(int id)

{

if (id <= 0)

return BadRequest("Not a valid studet id");

using (var ctx = new SchoolDBEntities())

{

var student = ctx.Students

.Where(s => s.StudentID == id)

.FirstOrDefault();

ctx.Entry(student).State = System.Data.Entity.EntityState.Deleted;

ctx.SaveChanges();

}

return Ok();

}

}

}