

Consuming Web Services

Using Different C# APIs

Data Structures and Algorithms

Telerik Software Academy

<http://academy.telerik.com>



Table of Contents

- ◆ Consuming Web Services
- ◆ Performing GET requests
 - ◆ The WebClient class
- ◆ Performing ANY requests
 - ◆ Using System.Net.WebRequest and HttpWebRequest
- ◆ Working with JSON and XML POST data
 - ◆ JSON.NET and XDocument

JSON.NET

- ◆ JSON.NET is a popular open source .NET framework for working with JSON data
- ◆ JSON.NET supports:
 - ◆ Serializing .NET objects into JSON objects
 - ◆ Deserializing JSON objects into .NET objects
 - ◆ LINQ to JSON
 - ◆ Converting JSON data to and from XML
- ◆ JSON.NET is included in many projects, like:
 - ◆ ASP.NET Web API for serialization
 - ◆ ASP.NET SignalR

Working with JSON.NET

Serializing and Deserializing .NET Objects

- ◆ **Serialization and deserialization of objects is done using methods of the JsonConvert class**

```
var person = new Person()
{ FirstName = "Doncho", LastName = "Minkov", Age = 24 };
var personJSON = JsonConvert.SerializeObject(person);
Console.WriteLine(personJSON);
/* outputs:
{"FirstName":"Doncho","LastName":"Minkov","Age":24} */
```

```
Person personDeserialized =
JsonConvert.DeserializeObject<Person>(personJSON);
Console.WriteLine(personDeserialized.FullName);
//outputs: Doncho Minkov
```

Consuming Web Services with C#

Consuming Web Services

- ◆ Using WebClient
 - ◆ One-line-of-code GET and POST HTTP requests
 - ◆ Kind of strange for the other HTTP requests
- ◆ Using HttpWebRequest
 - ◆ More configurable than WebClient
 - ◆ Nice way of performing ANY HTTP requests
- ◆ Using HttpClient with HttpRequestMessage
 - ◆ Look like a native HTTP request
 - ◆ Obsoletes HttpWebRequest in .NET 4.5
 - ◆ Better async operations

WebClient

- ◆ WebClient is a C# class used for communication with web services/resources
 - ◆ Works for ANY HTTP requests methods
 - ◆ Yet works best for GET and POST
 - ◆ One-line-of-code requests

```
var webClient = new WebClient();

//perform GET HTTP request on serviceUrl
webClient.DownloadString(serviceUrl);

//perform POST HTTP request with data on serviceUrl
//data should be serialized to string
webClient.UploadString(serviceUrl, data);
```

Simple WebClient Requests

Live Demo

Configuration of WebClient

- ◆ WebClient can be configured to work with the full power of REST services
 - ◆ ANY HTTP requests
 - ◆ GET, POST, PUT, DELETE, etc...
 - ◆ Adding HTTP Headers
 - ◆ ContentType, Accept, Cache, etc...
 - ◆ Sync and async calls
 - ◆ Authentication credentials

Configuration of WebClient (2)

- ◆ WebClient supports ANY HTTP request types
 - ◆ DownloadString() for GET
 - ◆ UploadString() for others

```
var webClient = new WebClient();

//perform GET HTTP request on serviceUrl
webClient.DownloadString(serviceUrl);

//perform POST HTTP request with data on serviceUrl
//data should be serialized to string
webClient.UploadString(serviceUrl, data);

//perform DELETE HTTP request with data on serviceUrl
//data should be empty
webClient.UploadString(serviceUrl, "DELETE", "");
```

Making Requests with WebClient

Live Demo

The `HttpWebRequest` Class

HttpWebRequest

- ◆ WebClient is good, but kind of hard to play with REST
- ◆ HttpWebRequest is a class that can access the full power of REST in an easy-to-use way
 - ◆ Much more easily configurable than WebClient

```
//create the HTTP request
var req = WebRequest.Create(resourceUrl) as HttpWebRequest;

//configure the HTTP request
req.ContentType = "application/json";
req.Method = "GET";

//send the request
var response = req.GetResponse();

//read the response body
```

Needs a cast to
HttpWebRequest

Performing a Request with HttpWebRequest

Live Demo

Working with `HttpWebRequest`

- ◆ How does `HttpWebRequest` work?
 - The client (the C# app) builds a HTTP request object
 - Wrapped in a `HttpWebRequest` object
 - The client sends the HTTP request to the server
 - Through the `GetResponse()` method
 - Then the server returns a response
 - Wrapped in a `WebResponse` object
 - Accessing the request/response body happens through a Stream

Request/Response body access

- ◆ **HttpWebRequest & WebResponse have bodies**
 - ◆ Data sent to/received from the server (e.g. in a POST request)
 - ◆ **GetRequestStream()/GetResponseStream()**
- ◆ A Stream can be read/written with a **StreamReader/StreamWriter**

```
...
var writer = new StreamWriter(request.GetRequestStream());
writer.Write(dataString);
writer.Close(); //or put the writer in a using directive
var response = request.GetResponse();
var reader = new StreamReader(response.GetResponseStream());
Console.WriteLine(reader.ReadToEnd());
reader.Close(); //or put the reader in a using directive
```

GET with `HttpWebRequest`

- ◆ Making GET requests

- ◆ Set the requested content type (e.g. "application/json")
- ◆ Set the request method to GET
- ◆ Call `GetResponse()` and process the data
 - ◆ E.g. use JSON.NET to deserialize to an object

```
public static void Get(string resourceUrl)
{
    var request = WebRequest.Create(resourceUrl) as HttpWebRequest;
    request.ContentType = "application/json";
    request.Method = "GET";
    request.GetResponse();
    ...
}
```

POST with HttpWebRequest

- ◆ POST request – similar to GET request, except
 - ◆ Different method ("POST")
 - ◆ Set request body (write to the request stream)

```
public static void Post(string resourceUrl, object data)
{
    var request = WebRequest.Create(resourceUrl) as HttpWebRequest;
    request.ContentType = "application/json";
    request.Method = "POST";
    var jsonData = JsonConvert.SerializeObject(data);
    using (StreamWriter writer =
        new StreamWriter(request.GetRequestStream()))
    {
        writer.Write(jsonData);
    }
    request.GetResponse();
    ...
}
```

Class for GET and POST HTTP Requests

Live Demo

HttpClient and HttpRequestMessage

The new APIs in .NET

- ◆ Modern HTTP client for .NET
- ◆ Flexible API for accessing HTTP resources
- ◆ Has ONLY async methods
 - Using the new async APIs
- ◆ Sends and receives HTTP requests and responses
 - `HttpRequestMessage`, `HttpResponseMessage`
 - Responses/requests are accessed ONLY async
- ◆ Can have defaults configured for requests

- ◆ Methods for directly sending GET, POST, PUT and DELETE requests
 - ◆ For commonly used requests
 - ◆ No need to construct the request from scratch

```
static async void PrintStudents(HttpClient httpClient)
{
    var response = await httpClient.GetAsync("students");
    Console.WriteLine(await response.Content.ReadAsStringAsync());
}

static void Main(string[] args)
{
    var httpClient = new HttpClient();
    httpClient.BaseAddress = new Uri("http://localhost:7232/api/");
    PrintStudents(httpClient);
    Console.WriteLine("Press Enter to exit");
    Console.ReadLine();
}
```

- ◆ **HttpContent defines the request/response**
 - ◆ Contains the body
 - ◆ Contains the ContentType header
 - ◆ Can be set with several content classes
 - ◆ StringContent, StreamContent, BlockContent...
 - ◆ Essential for POST and similar requests

```
HttpContent postContent = new  
StringContent(JsonConvert.SerializeObject(theStudent));  
  
postContent.Headers.ContentType = new  
System.Net.Http.Headers.MediaTypeHeaderValue("application/json");
```

Simple HttpClient Requests

Live Demo

HttpRequestMessage

- ◆ Full flexibility of defining an HTTP request
- ◆ Basically access to low-level request options
- ◆ Sent by an HttpClient
 - ◆ SendAsync() method

Complex Requests with HttpClient & HttpRequestMessage

Live Demo

Consuming Web Services

Questions?

1. Write a console application, which searches for news articles by given a query string and a count of articles to retrieve.
The application should ask the user for input and print the Titles and URLs of the articles.
For news articles search use the [Feedzilla API](#) and use one of WebClient, HttpWebRequest or HttpClient.