REST APIs

Szolgáltatásorientált rendszerintegráció

Service-Oriented System Integration

Dr. Balázs Simon

BME, IIT

DataContract

WCF DataContract

- Strongly typed mapping between:
  - .NET classes and XSD
  - .NET objects+values and XML
- Fast, easy to use
- Can be used for JSON

WCF DataContract Attributes

- [DataContract]
  - used on a class or enum
  - maps to a JSON object
- [DataMember]
  - used on a field or property
  - maps to a JSON attribute
- [EnumMember]
  - used on an enum value
  - maps to an integer value
- [CollectionDataContract]
  - used on a class inherited from a collection
  - maps to a JSON array

[DataContract], [DataMember]

```
[DataContract]
public class Complex
{
  [DataMember]
  public double Re { get; set; }
  [DataMember]
  public double Im { get; set; }
}
```

JSON:

```
{"Im":4.7,"Re":3.1}
```

[CollectionDataContract]

```
[CollectionDataContract]
public class ComplexValues : List<Complex>
{
}
```

JSON:

```
[{"Im":4.7,"Re":3.1},
 {"Im":5.6,"Re":3.9},
 {"Im":1.33,"Re":5.34}]
```

[EnumMember]

```
[DataContract]
public enum Colors
{
    [EnumMember]
    Red,
    [EnumMember]
    Green,
    [EnumMember(Value = "Blueish")]
    Blue
}

[DataContract]
public class Lamp
{
    [DataMember]
    public Colors Color { get; set; }
}
```

JSON (for Blue):

```
{"Color":2}
```

Serialization

```
// Creating the object to serialize:
var complex = new Complex();
complex.Re = 3.1;
complex.Im = 4.7;
// Creating the serializer:
var dcjs = new DataContractJsonSerializer(typeof(Complex));
// Creating the file:
using (FileStream file =
          new FileStream("complex.json", FileMode.Create))
{
    // Writing the Complex object:
    dcjs.WriteObject(file, complex);
}
```

Deserialization

```
// Creating the serializer:
var dcjs = new DataContractJsonSerializer(typeof(Complex));
// Opening the file:
using (FileStream file =
          new FileStream("complex.json", FileMode.Open))
{
    // Reading the Complex object:
    Complex complex = (Complex)dcjs.ReadObject(file);
}
```

ASP.NET Core Web API

ASP.NET Core Web API

- Goal:
  - Mapping between RESTful web services and .NET classes
  - Uses .NET attributes

General Web API Attributes

- [ApiController]
  - used on a class
  - defines a Controller
- [Route]
  - used on a class or a method
  - specifies a relative path from the parent (application, controller)
- [HttpGet], [HttpPost], [HttpPut], [HttpDelete], [HttpHead], [HttpPatch]
  - used on a method
  - specifies the HTTP verb and a relative path within the controller
  - may specify a route template for parameters in the URL

Attributes for Parameters

- [FromQuery]
  - gets value from the query string
- [FromRoute]
  - gets value from route data
- [FromForm]
  - gets value from posted form fields
- [FromBody]
  - gets value from the request body
- [FromHeader]
  - gets value from HTTP headers

Parameter binding

- By default, model binding gets data in the form of key-value pairs from the following sources in an HTTP request:
  - Form fields
  - Request body
  - Route data
  - Query string parameters
  - Uploaded files
- Route data and query string values are used only for simple types
- Uploaded files are bound only to target types that implement `IFormFile` or `IEnumerable<IFormFile>`

Registering the controller in Program.cs

```
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllers();

var app = builder.Build();
app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();

app.Run();
```

Register API controllers

Query parameters

```
http://localhost/api/calculator/add?left=5&right=8
```

```
[ApiController]
[Route("api/[controller]")]
public class CalculatorController : Controller
{
    [HttpGet("add")]
    public double Add([FromQuery] double left,
               [FromQuery] double right) { ... }

    [HttpGet("multiply")]
    public double Multiply([FromQuery] double left,
                    [FromQuery] double right) { ... }
}
```

Form parameters

```
http://localhost/api/calculator/add
```

```
[ApiController]
[Route("api/[controller]")]
public class CalculatorController : Controller
{
    [HttpPost("add")]
    public double Add([FromForm] double left,
                      [FromForm] double right) { ... }

    [HttpPost("multiply")]
    public double Multiply([FromForm] double left,
                      [FromForm] double right) { ... }
}
```

Path parameters

```
http://localhost/api/calculator/add/5/8
```

```
[ApiController]
[Route("api/[controller]")]
public class CalculatorController : Controller
{
    [HttpGet("add/{left}/{right}")]
    public double Add([FromRoute] double left,
                    [FromRoute] double right) { ... }

    [HttpGet("multiply/{left}/{right}")]
    public double Multiply([FromRoute] double left,
                    [FromRoute] double right) { ... }
}
```

Returning an error

```
http://localhost/api/calculator/divide?left=5&right=0
```

```
[ApiController]
[Route("api/[controller]")]
public class CalculatorController : Controller
{
  [HttpGet("divide")]
  public double Divide(double left, double right)
  {
    if (right == 0)
    {
      throw new HttpResponseException(HttpStatusCode.BadRequest);
    }
    return left/right;
  }
}
```

JSON and XML formatter configuration

Both are supported by default

Uses Json.NET by default

```
var json = GlobalConfiguration.Configuration.Formatters
             .JsonFormatter;
json.UseDataContractJsonSerializer = true;
```

Switch to DataContract serialization

```
var xml = GlobalConfiguration.Configuration.Formatters
            .XmlFormatter;
```

Uses DataContract serialization by default

Body parameters: XML and JSON format at the same time

```
[ApiController]
[Route("api/[controller]")]
public class CalculatorController : Controller
{
  [HttpGet("divide")]
  public Complex Add(Operands operands)
  { ... }
}
```

DataContract attribute is optional.
All public properties are serialized:

```
public class Operands
{
  public Complex Left
  { get; set; }

  public Complex Right
  { get; set; }
}
```

Only DataMembers are serialized:

```
[DataContract]
public class Operands
{
  [DataMember]
  public Complex Left
  { get; set; }

  [DataMember]
  public Complex Right
  { get; set; }
}
```

Client: only untyped

```
// Creating a client:
var client = new HttpClient();
client.BaseAddress =
    new Uri("http://localhost/SoiApp/api/calculator");

// Calling the service:
var operands = new Operands() { ... }
var response = await client.PostAsJsonAsync("add", operands);
var result = await response.Content.ReadFromJsonAsync<Complex>();
```

HTML Clients for REST

Plain HTML: query parameters

```
http://localhost/SoiApp/Calculator.svc/add?left=5&right=8
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
</head>
<body>
    <form method="get"
     action="http://localhost/SoiApp/Calculator.svc/add">
        Left: <input type="text" name="left"/><br/>
        Right: <input type="text" name="right"/><br/>
        <input type="submit" value="Add"/>
    </form>
</body>
</html>
```

Plain HTML: form parameters

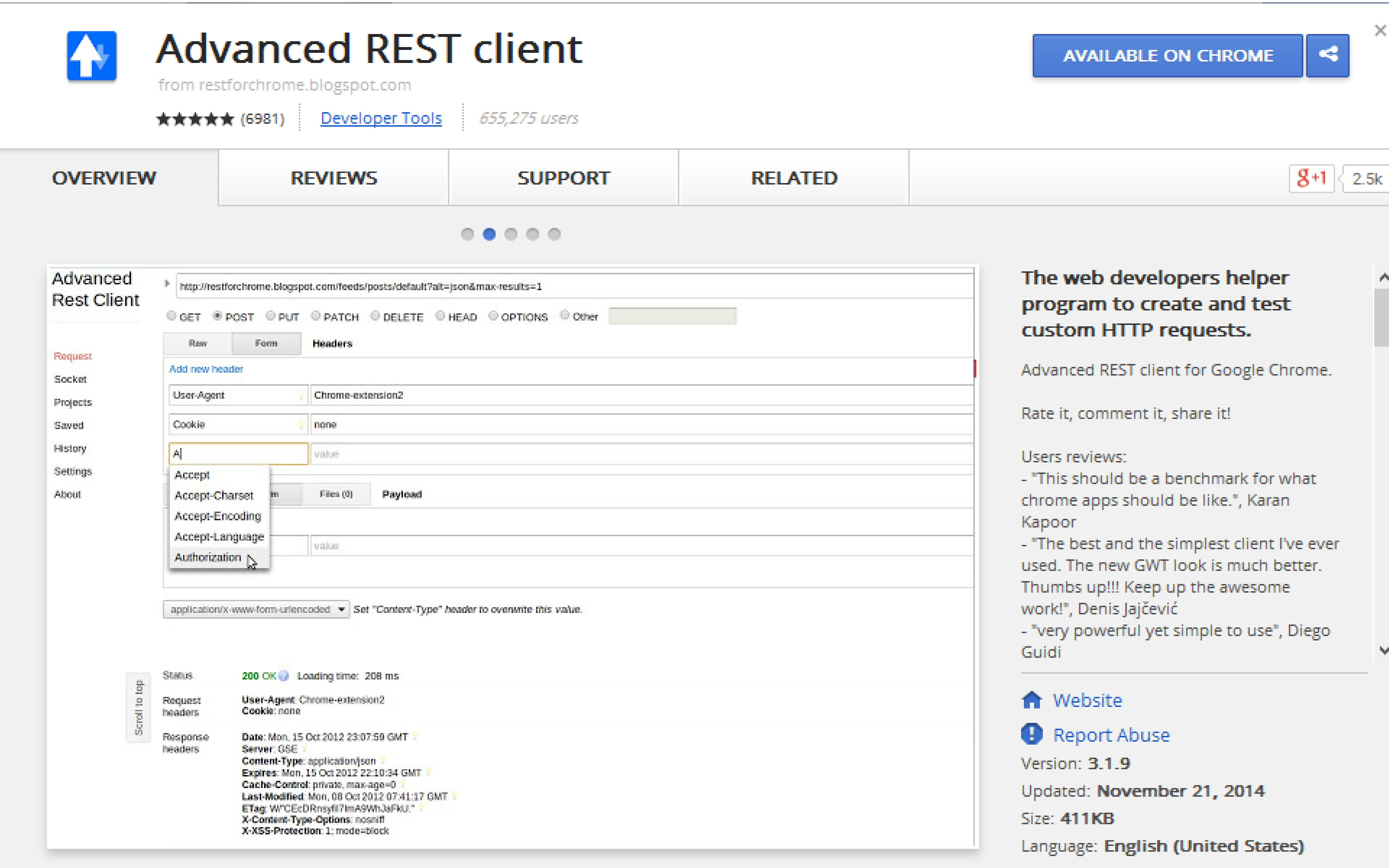
```
http://localhost/SoiApp/Calculator.svc/add
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
</head>
<body>
    <form method="post"
     action="http://localhost/SoiApp/Calculator.svc/add">
        Left: <input type="text" name="left"/><br/>
        Right: <input type="text" name="right"/><br/>
        <input type="submit" value="Add"/>
    </form>
</body>
</html>
```

jQuery AJAX

```
$.ajax({
    url: 'http://localhost/SoiApp/Calculator.svc/add',
    type: 'POST',
    data: $('#form').serializeArray(),
    success: function () { alert('POST completed'); }
});
```

Advanced REST Client



http://restforchrome.blogspot.com/feeds/posts/default?alt=json&max-results=1

User-Agent: Chrome-extension2
Cookie: none
A: value

application/x-www-form-urlencoded ▾ Set "Content-Type" header to overwrite this value.

```
Status            200 OK   Loading time:  208 ms

Request           User-Agent: Chrome-extension2
headers           Cookie: none

Response          Date: Mon, 15 Oct 2012 23:07:59 GMT
headers           Server: GSE
                  Content-Type: application/json
                  Expires: Mon, 15 Oct 2012 22:10:34 GMT
                  Cache-Control: private, max-age=0
                  Last-Modified: Mon, 08 Oct 2012 07:41:17 GMT
                  ETag: W/"CEcDRnsyfl7ImA9WhJaFkU."
                  X-Content-Type-Options: nosniff
                  X-XSS-Protection: 1; mode=block
```

The web developers helper program to create and test custom HTTP requests.

Advanced REST client for Google Chrome.

Rate it, comment it, share it!

Users reviews:
- "This should be a benchmark for what chrome apps should be like.", Karan Kapoor
- "The best and the simplest client I've ever used. The new GWT look is much better. Thumbs up!!! Keep up the awesome work!", Denis Jajčević
- "very powerful yet simple to use", Diego Guidi

Website

Report Abuse

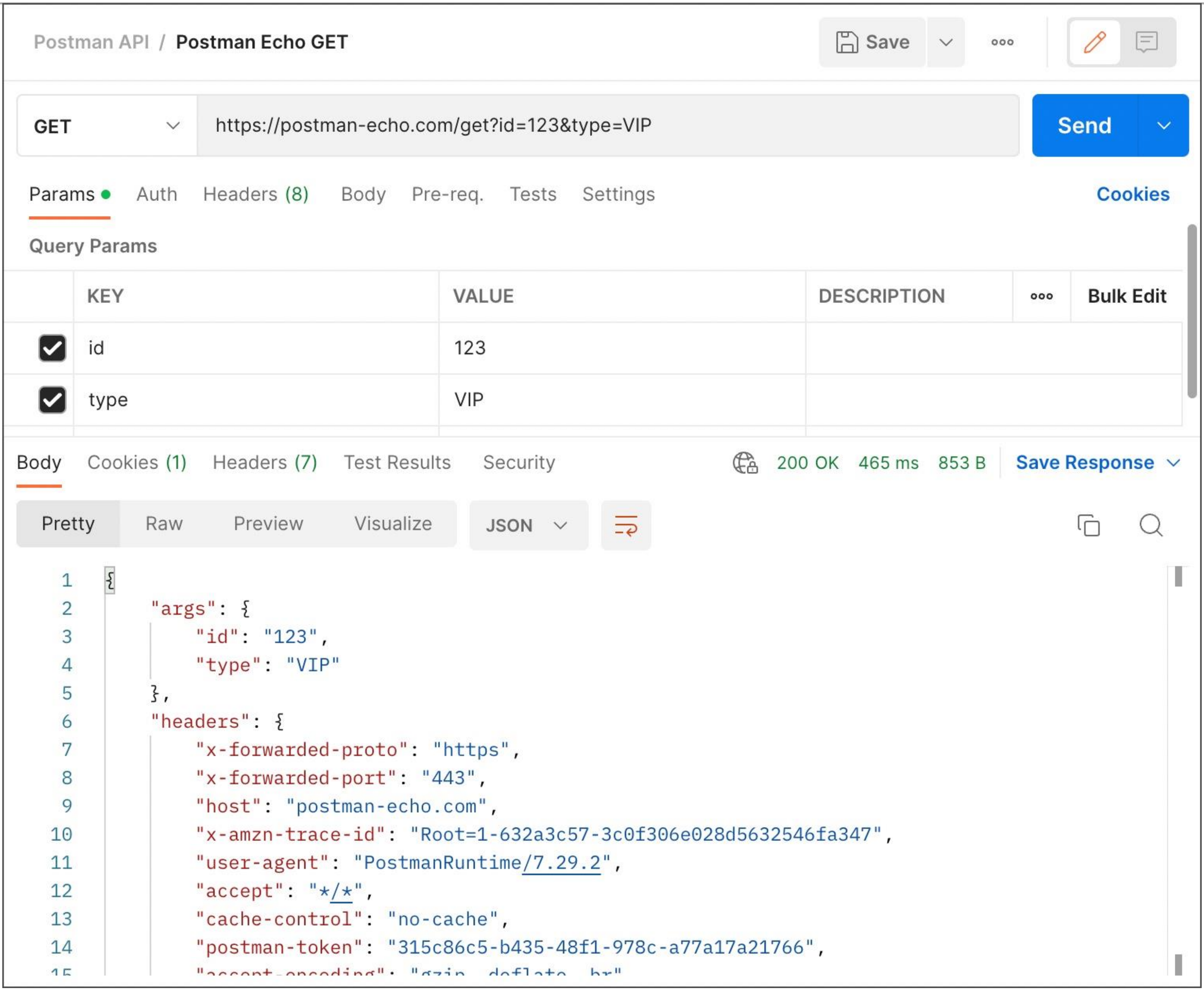
Version: **3.1.9**

Updated: **November 21, 2014**

Size: **411KB**

Language: **English (United States)**

Postman



GET https://postman-echo.com/get?id=123&type=VIP

Query Params

	KEY	VALUE	DESCRIPTION
☑	id	123	
☑	type	VIP	

200 OK 465 ms 853 B

```
{
    "args": {
        "id": "123",
        "type": "VIP"
    },
    "headers": {
        "x-forwarded-proto": "https",
        "x-forwarded-port": "443",
        "host": "postman-echo.com",
        "x-amzn-trace-id": "Root=1-632a3c57-3c0f306e028d5632546fa347",
        "user-agent": "PostmanRuntime/7.29.2",
        "accept": "*/*",
        "cache-control": "no-cache",
        "postman-token": "315c86c5-b435-48f1-978c-a77a17a21766",
        "accept-encoding": "gzip, deflate, br"
```

Summary

Summary

- Java
  - JAXB
  - JAX-RS
- .NET
  - DataContract
  - ASP.NET Core Web API