POLITÉCNICO DE LEIRIA
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

www.dei.estg.ipleiria.pt

Curso Técnico Superior Profissional

Programação de Sistemas de Informação

Serviços e Interoperabilidade de Sistemas

Clientes para Serviços REST JSON

Objetivos:

– Compreender como consumir um Serviço REST em JSON
– Usar Serviços públicos
– Utilizar diferentes tecnologias para consumir os serviços

Duração: 3 aulas

Porquê usar serviços REST?

Os Serviços REST podem ser consumidos por particamente qualquer cliente ou servidor já que são 100% serviços web. Exigem menos conhecimentos para a implementação em relação aos tradicionais serviços em Simple Object Access Protocol (SOAP) e exigem menos recursos computacionais e de largura de banda.

Dado que os Serviços REST são 100% suportados pelas tecnologias abertas da Web, estes são invocados usando os verbos definidos no protocolo HTTP, ou seja, GET, PUT, POST, DELETE, HEAD, etc. Quando transpomos as operações das bases de dados (CRUD) para os serviços REST, é comum assumir os seguintes verbos HTTP para as seguintes operações:

Ação HTTP	**Operação CRUD (BD)**	**Formato de URL usada** **<dominio>/recurso/{id}**	**Mais informação**
POST	Criar	<domínio>/clientes/	Erros HTTP a devolver: [404 (Not Found)] [409 (Conflict) Já existe] 201 (Created) e devolver novo cliente no HTTP Body
GET	Ler	<domínio>/clientes/	Erros HTTP a devolver: 200 (OK) e devolver registos no HTTP body
GET	Ler	<domínio>/clientes/5	Erros HTTP a devolver: 200 (OK) e registo no HTTP body 404 (Not Found)
PUT	Atualizar	<domínio>/clientes/6	Erros HTTP a devolver: 200 (OK) e devolver registo atualizado no HTTP body 204 (No Content) caso faltem campos 404 (Not Found) em caso de registo não encontrado
DELETE	Apagar	<domínio>/clientes/3	200 (OK) e devolver registo apagado 404 (Not Found) em caso de registo não encontrado

As operações requeridas são sempre acompanhadas por um código de erro HTTP que indica se a operação foi ou não foi bem-sucedida, tal como apresentado no quadro acima.

É possível consultar dados de um serviço a partir do browser.

Para além do browser, e uma vez que os serviços RESTful são 100% compatíveis com a Web, poderemos usar qualquer comando, programa ou ferramenta que exista para a Web para interagir com os mesmos. Nesta ficha prática iremos usar, como clientes, um utilitário da linha de comandos chamado cURL, o Fiddler/RESTClient/Postman/Advanced REST Client, Javascript/AJAX e clientes específicos desenvolvidos em C#.

Exercício 1:

Considere e consulte o seguinte Serviço RESTful JSON Público para a resolução deste exercício:

https://jsonplaceholder.typicode.com/

Este serviço permite efetuar um conjunto de operações sobre um conjunto de dados que simulam um blog com Users, Posts, Comments, etc…

Nota: As ações que produzem alterações no conjunto de dados são simuladas. Obtém-se resposta, mas os dados não são alterados.

a) Consultar o serviço usando o comando cURL (https://curl.haxx.se/download.html)
   i. Obtenha todos o *posts*
   ii. Mostre o *user* com o id 10
   iii. Obtenha todos os *posts* do *user* 10
   iv. Mostre o *post* com o id 5
   v. Mostre todos os *comments* do *post* 5
   vi. Adicione um *post* novo para *user* id 10
   vii. Altere o título do *post* com id 2
   viii. Apague o *post* com id 10

b) Use agora o utilitário gráfico Fiddler (ou RESTClient add-on do Firefox ou o Advanced REST client do Chrome ou o Postman) e repita as mesmas ações para o mesmo serviço.

O Javascript/AJAX como cliente REST

O Javascript é uma das três linguagens da Web (HTML, CSS e Javascript) que os *developers* devem aprender, segundo a W3Schools. É esta a linguagem que permite que as páginas Web se tornem interativas e, por isso, todos os browsers incluem suporte à execução desta.

Apesar de o Javascript oferecer suporte nativo a chamadas REST (`XMLHttpRequest`), nesta ficha será usada a biblioteca Javascript **jQuery** ("Write Less, Do More") a fim de tornar o processo mais simples. Assim, será necessário incluir a mesma no 'carregamento' da página, na qual pretendemos aceder ao serviço REST.

```
<html>

   <head>
      <title>Título</title>
      <!--<script type = "text/javascript"  src = "/jquery/jquery-2.1.3.min.js"></script> -->
      <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

      <script type = "text/javascript">
         //Código Javascript
      </script>

   </head>

   <body>
      <h1>Olá!</h1>
   </body>

</html>
```

Uma vez incluída a biblioteca JQuery, poderemos invocar um serviço RESTful usando a função **$.ajax()**, considerando os respetivos argumentos:

```
<script>
//var data = {"nome":var_x,"morada":var_y};
$.ajax(
{
   type: "post",                          //get, post, put, delete
   url: "http://abc.com/inprise/clientes",
   data: {"nome":"aaaa","morada":"xxxx"}, //JSON.stringify(data)
   dataType: "json",
   success: function(resp)
   {
      alert("O serviço respondeu: " + resp );
   },
   error: function(e)
   {
      alert("Erro: " + e);
   }
});
</script>
```

O código anterior irá ser executado de forma assíncrona (*default*), ou seja, enquanto o pedido está a ser executado o Javascript avança para o código seguinte. Contudo, se for imperativo bloquear a execução da função **$.ajax** então o argumento/chave **async** deverá ser passado a **false**.

```
<script>
    // ...
    $.ajax(... async: false ...); // Primeiro completa esta chamada, depois passa à próxima…

    $.ajax(...); // Executada após términus da chamada anterior
    // ...
</script>
```

Exercício 2:

a) Considere o serviço RESTful alojado no seguinte endereço: **http://ip.jsontest.com**
Usando Javascript, execute a ação HTTP GET e tente interpretar a resposta.

b) Execute a mesma ação agora para o endereço: **http://date.jsontest.com**

Exercício 3:

Considere agora o serviço RESTful instalado no computador do Professor, nesta sala de aula, que disponibiliza as ações GET, PUT, POST e DELETE para uma base de dados de clientes. Os detalhes do serviço serão colocados no quadro.

Usando Javascript, construa um formulário com os campos ID, NOME e MORADA. Por baixo deste formulário crie os botões Listar Cliente, Apagar Cliente e Alterar Cliente. Quando pressionados estes botões devem fazer o seguinte:

Listar Cliente – usa a informação do campo ID e obtém a ficha desse cliente

Apagar Cliente - usa a informação do campo ID e apaga a ficha desse cliente

Alterar Cliente – usa a informação completa do formulário e altera a informação do cliente em causa

Clientes RESTful em C#

A linguagem C#, tal como todas as outras, disponibiliza classes para invocar serviços RESTful remotos, tais como, DataContractJsonSerializer, MemoryStream, (…) para lidar com JSON e HttpWebRequest, WebResponse, (…) para invocar os serviços RESTful. Contudo, o uso destas classes não é tão simples como o esperado e por isso iremos recorrer à Biblioteca externa Json.NET para interagir com JSON e à Biblioteca RestSharp para interagir com os respetivos serviços. Estas bibliotecas são muito fáceis de instalar e muito intuitivas.

Instalação:

Tools| NuGetPackageManager| PackageManager Console

PM> Install-PackageNewtonsoft.Json

PM> Install-Package RestSharp

Utilização da Biblioteca Json.NET:

```
using Newtonsoft.Json;

public class Utilizador
{
      public bool EMembero{ get; set; }
      public string Nome { get; set; }
      public int Idade { get; set;}
}

String strTemp = "{ \"EMembro\" : true, \"Nome\" : \"Joao\", \"Idade\" : 18}";

//Passa de JSON String para object C#
Utilizador user = JsonConvert.DeserializeObject<Utilizador>(strTemp);
Console.WriteLine(user.Nome);

//Passa de Objeto C# para JSON String
Utilizador user = new Utilizador();
user.EMembero = true;
user.Nome= "João";
user.Idade= 20;
String strJson= JsonConvert.SerializeObject(user);
Console.WriteLine(strJson);
```

Quando não existe interesse em construir um objeto bem definido, a partir de uma string JSON (DeserializeObject), podemos recorrer a uma tabela de hash, que em C# se designa por dicionário, para obter a resposta à chamada a um serviço RESTful. Por exemplo:

```
String strTemp = "{ \"EMembro\" : true, \"Nome\" : \"Joao\", \"Idade\" : 18}";

Dictionary<string, string> resp = JsonConvert.DeserializeObject<Dictionary<string, string>>(strTemp);
MessageBox.Show(resp["Nome"]);
```

Utilização da Biblioteca RestSharp:

```
using RestSharp;

var client = new RestClient("http://www.example.com/1/2");
var request = new RestRequest();

request.Method = Method.POST;
request.AddHeader("Accept", "application/json");
request.Parameters.Clear();
request.AddJsonBody(
   new
   {
      nome = "UAT1206252627",
      idade = 18
   }
); // AddJsonBody serializes the object automatically

var response = client.Execute(request);
var content = response.Content;          //raw content as string
```

Exercício 4:

a) Considere o serviço RESTful alojado no seguinte endereço: **http://ip.jsontest.com**
Recorrendo a uma aplicação C# Form Based, execute a ação HTTP GET para o serviço e apresente a resposta numa caixa de texto.

b) Execute a mesma ação agora para o endereço: **http://date.jsontest.com**

Exercício 5:

Considere agora o serviço RESTful instalado no computador do professor, nesta sala de aula, que disponibiliza as ações GET, PUT, POST e DELETE para uma base de dados de clientes. Os detalhes do serviço serão colocados no quadro.

Usando uma aplicação C# Form Based, construa um formulário com os campos ID, NOME e MORADA. Por baixo deste formulário crie os botões Listar Cliente, Apagar Cliente e Alterar Cliente. Quando pressionados estes botões devem fazer o seguinte:

Listar Cliente – usa a informação do campo ID e obtém a ficha desse cliente

Apagar Cliente - usa a informação do campo ID e apaga a ficha desse cliente

Alterar Cliente – usa a informação completa do formulário e altera a informação do cliente em causa

Exercício 6:

Baseado nas ações do exercício 1 implemente uma aplicação Windows Form (C#) que:

a) Peça todos os *posts* e mostre os resultados numa *textbox* multilinha.

b) Tenha a opção para ir buscar um post mediante o id que permita:

   i. Ver os seus *comments* na *textbox* multilinha

   ii. Alterar os dados do *post*

   iii. Eliminar o *post*

c) Crie um botão que permita introduzir um novo *post* e mostre o seu novo id na *textbox* multilinha.