01

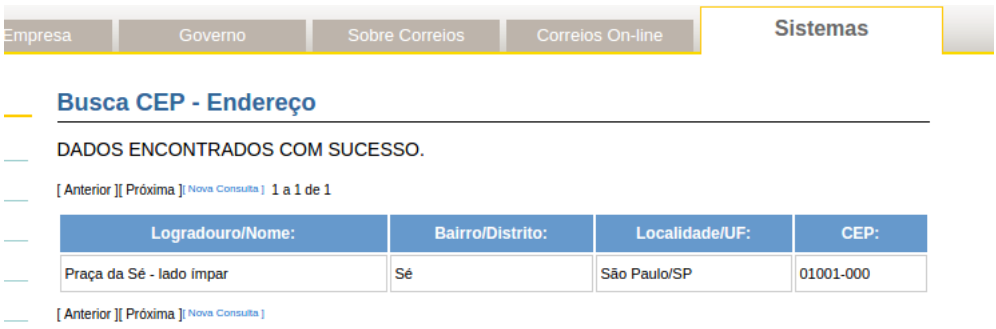
Consultando o Webservice do ViaCEP

Transcrição

Veremos agora como integrar consultas de endereços com correios em uma aplicação C#.

Vamos abrir o link dos Correios (http://www.buscacep.correios.com.br/sistemas/buscacep/) que faz a busca de endereços via CEP. Digitaremos o CEP **01001000** e teclar o **Enter**.

A página dos Correios nos trouxe as informações desse CEP:



Temos o **Logradouro**, **Bairro**, **Localidade/UF** e **CEP** formatado.

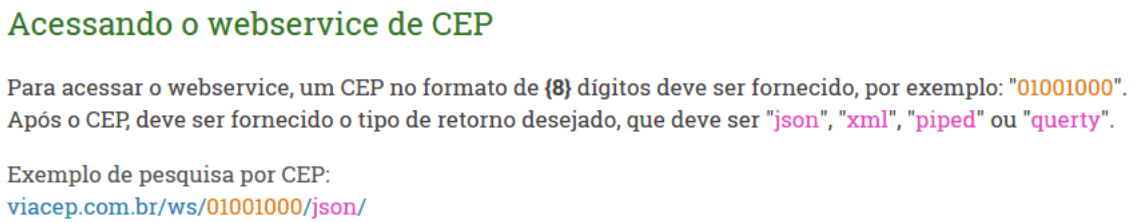
Como poderíamos integrar essas informações numa aplicação C#?

Poderíamos consultar esse endereço, abrir o HTML resultante, e depois, extrair somente as partes que nos interessam... Entretanto, essa tarefa daria um enorme trabalho e seria trabalhoso realizá-la.

Felizmente, existe um serviço que faz isso por nós: o ViaCEP (https://viacep.com.br/). OTrata-se de um serviço gratuito, que é utilizado pela comunidade de desenvolvedores brasileiros, em diversas linguagens como Java, C#, Python, PHP, e outras.

Como utilizamos o ViaCEP?

No próprio site do ViaCEP, temos um exemplo:



Quando abrimos o endereço que aparece na imagem acima, temos o resultado em formato **JSON**. Com esse resultado, podemos integrar essas informações em nossa aplicação.

No Visual Studio, adicionaremos um novo projeto chamado de `CEP`, sendo ele do tipo "Console App". Definiremos como um projeto inicial (*StartUp Project*).

O próximo passo é pegar a string do endereço do browser e copiar para a nossa aplicação. Essa URL será uma string.

```
static void Main(string[] args)
{
    string url = "https://viacep.com.br/ws/01001000/json/";
}
```

Iremos consumir a string para obter o resultado. Note que no meio da URL, temos o trecho que corresponde ao CEP procurado. É preciso extrair esse CEP para uma outra variável, assim podemos substituir no meio da URL, concatenando uma string com a outra:

```
static void Main(string[] args)
{
    string cep = 01001000;
    string url = "https://viacep.com.br/ws/" + cep + "/json/";
}
```

Para consultarmos esse endereço, precisamos acessar o objeto `clientHTTP` do C#. Criando uma nova instância do `HttpClient()`, vamos obter a string resultante que estava no navegador.

```
static void Main(string[] args)
{
    string cep = 01001000;
    string url = "https://viacep.com.br/ws/" + cep + "/json/";

    string result = new HttpClient().GetStringAsync(url). Result;

    Debug.WriteLine(result);
}
```

Como resultado, não temos uma *string*, e sim uma **Task**, já que esse método é *Assíncrono*. Para extrair uma string a partir dessa task, temos que consultar a propriedade `Result`. Essa propriedade transformará isso em um resultado **síncrono**, sendo possível consumir o valor dele a partir de uma variável string.

Vamos mostrar a variável `result` no console do Debug.

```
{
    "cep": "01001-000",
    "logradouro": "Praça da Sé",
    "complemento": "lado ímpar",
    "bairro": "Sé",
    "localidade": "São Paulo",
    "uf": "SP",
    "unidade": "",
    "ibge": "3550308",
    "gia": "1004"
}
```

Temos o resultado em formato **JSON**, que é uma string. O que aconteceria se quiséssemos consumir esses dados em outro formato diferente, por exemplo, em **XML**? Será preciso trocar a palavra `json` por `xml` na URL.

```
string url = "https://viacep.com.br/ws/" + cep + "/xml/";
```

E então, podemos rodar a aplicação. Como resultado, temos um código em **XML**.

```
<?xml version="1.0" encoding="UTF-8"?>
<xmlcep>
    <cep>01001-000</cep>
    <logradouro>Praça da Sé</logradouro>
    <complemento>lado ímpar</complemento>
    <bairro>Sé</bairro>
    <localidade>São Paulo</localidade>
    <uf>SP</uf>
    <unidade></unidade>
    <ibge>3550308</ibge>
    <gia>1004</gia>
</xmlcep>
```

Como reparamos, podemos trocar o tipo de resultado, alterando a URL. Voltemos a exibir o resultado em JSON para outros testes. Vamos modificar o CEP para que ele fique **inválido**.

Vamos modificar o CEP para que ele tenha um número a mais, e rodar a aplicação.

```
static void Main(string[] args)
{
    string cep = 010010007;
    string url = "https://viacep.com.br/ws/" + cep + "/json/";

    string result = new HttpClient().GetStringAsync(url). Result;

    Debug.WriteLine(result);
}
```

Como podemos ver, houve uma exceção! Os detalhes nos dizem que "o código de status de resposta não indica êxito: 400 (Bad Request)", isto é, o ViaCEP não aceitou o CEP informado com 9 dígitos.

Perceba que essa exceção **Bad Request**, é uma exceção genérica, ela não indica exatamente qual foi o erro, por isso, o erro pode ser qualquer coisa.

Com esse tipo de análise superficial, podemos ter problemas para identificar o retorno da chamada do web service. Um outro problema identificado é se quisermos trocar o formato do resultado, de `JSON` para `XML`, pois será preciso trocar manualmente.

Então, apesar da nossa busca estar funcionando e estar trazendo os dados direitinho, temos problemas na hora de tratar exceções, como também obter mudanças no tipo do retorno desse serviço. Por isso, existe uma maneira mais bem simples, que é utilizando a biblioteca `Caelum Stella CSharp`.

Essa biblioteca irá encapsular essa funcionalidade, dará um tratamento melhor de erros, e irá fornecer métodos adequados para que possamos obter os endereços do **ViaCEP**.