ARTIGO: 11401
Dica em Groovy - XML - Requisição a API e inspecionando a resposta
É normal nos depararmos com alguma API http que nos devolve XML.
Com Groovy, realizar a chamada e inspecionar a resposta é simples. Por exemplo, se precisamos realizar a chamada a alguma API, faça no Groovy Console assim:

Groovy Code
```
// biblioteca do groovy para http através do grape
@Grab(group='org.codehaus.groovy.modules.http-builder', module='http-builder', version='0.7')

import groovyx.net.http.*

def http = new HTTPBuilder()
def queryParams = [address: '1600 Amphitheatre Parkway, Mountain View, CA', sensor: 'false' ]
def xmlApiResponse

http.request( 'http://maps.googleapis.com', Method.GET, ContentType.XML ) { req ->
    uri.path = '/maps/api/geocode/xml'
    uri.query = queryParams

    response.success = { resp, xml ->
        xmlApiResponse = xml
    }

    response.failure = {
 println 'Oh no!'
    }
}
```

Observe como fica claro no código Groovy quais os parâmetros, URL e URI estão sendo manipulados.
O objeto em xmlApiResponse será do tipo NodeChild, pronto para ser iterado.
Antes de inspecionarmos a resposta, seria interessante mostrarmos de maneira formatada(pretty print) o conteúdo.
Pra isso podemos usar XmlUtil, dessa forma:

Groovy Code
```
import groovy.xml.* // adicionar esse import

println XmlUtil.serialize(xmlApiResponse)
```

Isso irá mostrar o texto com formatação, assim:

Groovy Code
```
<?xml version="1.0" encoding="UTF-8"?>
<GeocodeResponse>
  <status>OK</status>
  <result>
    <type>street_address</type>
    <formatted_address>1600 Amphitheatre Parkway, Mountain View, CA 94043, USA</formatted_address>
    <address_component>
      <long_name>1600</long_name>
      <short_name>1600</short_name>
      <type>street_number</type>
    </address_component>
    <address_component>
      <long_name>Amphitheatre Parkway</long_name>
      <short_name>Amphitheatre Pkwy</short_name>
      <type>route</type>
    </address_component>
    <address_component>
      <long_name>Mountain View</long_name>
      <short_name>Mountain View</short_name>
      <type>locality</type>
      <type>political</type>
    </address_component>
    <address_component>
      <long_name>Santa Clara County</long_name>
      <short_name>Santa Clara County</short_name>
      <type>administrative_area_level_2</type>
      <type>political</type>
    </address_component>
    <address_component>
      <long_name>California</long_name>
      <short_name>CA</short_name>
      <type>administrative_area_level_1</type>
      <type>political</type>
    </address_component>
    <address_component>
      <long_name>United States</long_name>
      <short_name>US</short_name>
      <type>country</type>
      <type>political</type>
    </address_component>
    <address_component>
      <long_name>94043</long_name>
      <short_name>94043</short_name>
      <type>postal_code</type>
```

```
    </address_component>
    <geometry>
      <location>
        <lat>37.4220033</lat>
        <lng>-122.0839778</lng>
      </location>
      <location_type>ROOFTOP</location_type>
      <viewport>
        <southwest>
          <lat>37.4206543</lat>
          <lng>-122.0853268</lng>
        </southwest>
        <northeast>
          <lat>37.4233523</lat>
          <lng>-122.0826288</lng>
        </northeast>
      </viewport>
    </geometry>
  </result>
</GeocodeResponse>
```

OK, temos a resposta em um objeto NodeChild. E agora? Como vamos atrás do que precisamos?
Esse artigo no website oficial do Groovy mostra vários exemplos de como vasculhar e procurar coisas na árvore xml.
Se eu precisar saber o street_number, atualmente contido no terceiro filho do elemento <result>, poderia fazer assim:

Groovy Code

```
// allNodes contém uma collection com todos os nós do documento
def allNodes = xmlApiResponse.depthFirst()

// achei o nó <type>street_number</type>
def streetNumberNode = allNodes.find{ it.name() == 'type' && it.text() == 'street_number' }

// vou até  pai desse elemento e inspeciono seus filhos
def streetNumber = streetNumberNode.parent().children().find{ it.name() == 'long_name' }.text()

println streetNumber // 1600
```

Só pra lembrar que esse exemplo pode ser rodado com oGroovy Console.
A classe NodeChild contém vários métodos para atravessar a árvore construída, como os métodostext(), parent() e children(), usados no exemplo.
Hoje a dica é essa pessoal.
Até a próxima