



Xml 다루기

Table of Contents

- Xml 다루기
 - XML 문서 읽기
 - DOM 파서 사용하기
 - 그루비 파서 사용하기
 - SAX
 - StAX
- XML 처리
 - 메모리에서 처리
 - 스트림 방식
 - XPath
- XML과 분산처리
 - RSS ATOM 읽기
 - REST
 - XML-RPC
 - SOAP

XML 문서 읽기

DOM 파서 사용하기

XPath 적용시 DOM을 사용해야 함.

DOM의 NodeList를 지원하는 기능이 그루비에 있으며 DOM을 사용시 필요한 기능을 제공하는 헬퍼 클래스도 그루비에 추가되었기 때문에 XML을 읽는 구식기술을 알아야 그루비의 좋은점을 더 잘알 수 있기 때문

```

1 // 샘플 plan.xml
2
3 <plan>
4   <week capacity="8">week
5     <task done="2" total="2" title="read XML chapter" />
6     <task done="3" total="3" title="try some reporting" />
7     <task done="1" total="2" title="use in current project" />
8   </week>
9   <week capacity="8">
10    <task done="0" total="1" title="re-read DB chapter" />
11    <task done="0" total="3" title="use DB/XML combination" />
12    <task done="1" total="2" title="use todo">todo check</task>
13  </week>
14 </plan>
    
```

java Sample

```

1 // java에서 DOM 파서 사용 예
2
3 package com.coma.java.xml.dom;
4
5 import java.io.File;
6
7 import javax.xml.parsers.DocumentBuilder;
8 import javax.xml.parsers.DocumentBuilderFactory;
9
10 import org.w3c.dom.Document;
11 import org.w3c.dom.Element;
12 import org.w3c.dom.Node;
13 import org.w3c.dom.NodeList;
14
15 public class JXmlDomTest {
16
17     public static void main(String[] args) {
18         DocumentBuilderFactory docBuildFact = DocumentBuilderFactory.newInstance();
19         String strPath = "C:\\reakosys\\workspace\\xml\\plan.xml";
20
21         try {
22             DocumentBuilder docBuild = docBuildFact.newDocumentBuilder();
23             File xmlFilePath = new File(strPath);
24             Document doc = docBuild.parse(xmlFilePath);
25         }
26     }
27 }
    
```

```

25 Document doc = docBuilderFactory.newInstance().newDocumentBuilder().parse(xmlFile);
26
27 NodeList weekNodeList = doc.getElementsByTagName("week");
28
29 for(int i = 0; i < weekNodeList.getLength(); i++){
30     Node weekNode = weekNodeList.item(i);
31     Element weekElement = (Element)weekNode;
32     System.out.println("week capacity " + i + "번째 :: " + weekElement.getAttribute("capacity"));
33
34     NodeList taskNodeList = weekElement.getElementsByTagName("task");
35
36     for(int j=0; j < taskNodeList.getLength(); j++){
37         Element taskElmnt = (Element) taskNodeList.item(j);
38
39         System.out.println("week >> task >> done ==> " + taskElmnt.getAttribute("done"));
40         Node taskNode = taskElmnt.getFirstChild();
41
42         if(taskNode != null){
43             System.out.println(taskNode.getNodeValue());
44         }
45     }
46 }
47 } catch (Exception e) {
48     e.printStackTrace();
49 }
50
51 }
52
53 }

```

groovy sample 1 Dom 파서 이용하기

```

1 //groovy DOM 파서 예제1
2
3 package com.coma.groovy.xml.dom
4
5 import javax.xml.parsers.DocumentBuilderFactory
6
7 String strPath = "C:\\reakosys\\workspace\\xml\\plan.xml"
8
9 String weekNodeinfo(node){
10     if(node.nodeName != 'week') return
11
12     def weekCapacity = node.attributes.getNamedItem('capacity')
13
14     def taskNode = node.getElementsByTagName('task')
15
16     (0..<taskNode.length).each{
17         println taskNode.item(it).attributes.getNamedItem('done').nodeValue
18         println taskNode.item(it).attributes.getNamedItem('total').nodeValue
19         println taskNode.item(it).attributes.getNamedItem('title').nodeValue
20
21         if(taskNode.item(it).firstChild != null) println taskNode.item(it).firstChild.nodeValue
22     }
23 }
24
25
26 def docBuildFact = DocumentBuilderFactory.newInstance();
27 def docBuild = docBuildFact.newDocumentBuilder();
28 def doc = docBuild.parse(new File(strPath))
29 def root = doc.documentElement
30
31 def plan = root.childNodes
32
33 (0..<plan.length).each{
34     def week = plan.item(it)
35     println weekNodeinfo(week)
36 }

```

groovy sample 2 DOMCategory 이용하기

```

1 //DOMCategory 이용한 Sample
2
3 package com.coma.groovy.xml.dom
4
5 import groovy.xml.dom.DOMCategory
6
7 import javax.xml.parsers.DocumentBuilderFactory
8
9 String strPath = "C:\\reakosys\\workspace\\xml\\plan.xml"
10
11 def docBuildFact = DocumentBuilderFactory.newInstance();
12 def docBuild = docBuildFact.newDocumentBuilder();
13 def doc = docBuild.parse(new File(strPath))
14 def root = doc.documentElement
15
16 def plan = root.childNodes
17
18 use (DOMCategory) {
19     def week = root.week
20
21     (0..<week.size()).each{
22         println week[it].@capacity
23     }
24
25     def task = week.task
26
27     (0..<task.size()).each{

```

```

28     println 'task title ==> ' + task[it].title
29     println 'task text ==> ' + task[it].text()
30 }
31 }
32 }

```

그루비 파서 사용하기

<http://groovy.codehaus.org/api/groovy/util/XmlParser.html>

<http://groovy.codehaus.org/api/groovy/util/XmlSlurper.html>

XmlParser 클래스는 groovy.util 패키지에 포함되기 때문에 따로 import가 필요없음.

XmlSlurper 클래스도 XmlParser와 동일한 기능을 하는 클래스

차이점은 XmlParser 의 파싱 메서드는 groovy.util.Node를 리턴하고 XmlSlurper는 GPathResult를 리턴함.

groovy sample 3 그루비 파서 이용하기

```

1 package com.coma.groovy.xml.dom
2
3
4 String strPath = "C:\\reakosys\\workspace\\xmlplan.xml"
5
6 def plan = new XmlParser().parse(new File(strPath))
7 //def plan = new XmlSlurper().parse(new File(strPath))
8
9 (0..<plan.week.size()).each {
10     println plan.week[it].capacity
11 }

```

SAX

푸시 방식 이벤트 기반의 파서.

파싱 결과를 이벤트로 만들어 코드 쪽으로 전달.

파싱 결과를 저장할 메모리가 필요 없음.

```

1 PlanHandler.groovy
2
3 import org.xml.sax.*
4 import org.xml.sax.helpers.DefaultHandler
5
6 class PlanHandler extends DefaultHandler {
7     def messages = []
8     def currentMessage
9     def countryFlag = false
10    void startElement(String ns, String localName, String qName, Attributes atts) {
11        switch (qName) {
12            case 'week':
13                currentMessage = 'week of capacity ' + atts.getValue('capacity'); break
14            case 'task':
15                currentMessage += ' task of done ' + atts.getValue('done'); break
16        }
17    }
18
19    void endElement(String ns, String localName, String qName) {
20        switch (qName) {
21            case 'week':
22                messages << currentMessage; break
23            case 'task':
24                currentMessage += ' has a '; break
25        }
26    }
27 }
28
29 import javax.xml.parsers.SAXParserFactory
30
31 import org.xml.sax.InputSource
32
33 String strPath = "C:\\reakosys\\workspace\\xmlplan.xml"
34
35 def handler = new PlanHandler()
36 def reader = SAXParserFactory.newInstance().newSAXParser().XMLReader
37 reader.setContentHandler(handler)
38 def inputStream = new FileInputStream(strPath)
39 reader.parse(new InputSource(inputStream))
40
41 println handler.messages
42
43 //결과
44 // I week of capacity 8 task of done 2 has a task of done 3 has a task of done 1 has a week of capacity 10 task of done 0 has a ta

```

pull 방식의 이벤트 기반 파서.

데이터를 처리하는 측에서 파서에 이벤트를 요청해야 함.

이벤트가 발생하여 메시지가 호출되기를 기다리는 방식이 아님.

메인 루프를 파서가 아니라 작성자가 직접 실행하는 방식.

```

1  StaxCategory.groovy
2
3  import javax.xml.stream.XMLStreamReader
4
5  class StaxCategory {
6      static Object get(XMLStreamReader self, String key) {
7          return self.getAttributeValue(null, key)
8      }
9      static String name(XMLStreamReader self) {
10         return self.name.toString()
11     }
12     static String text(XMLStreamReader self) {
13         return self.elementText
14     }
15 }
16
17 def input = "file:reakosysworkspacexmlplan.xml".toURL()
18 def underway = []
19 def upcoming = []
20
21 def eachStartElement(inputStream, Closure yield){
22     def token = XMLInputFactory.newInstance().createXMLStreamReader(inputStream)
23
24     try{
25         while (token.hasNext()) {
26             if (token.startElement) yield token
27             token.next()
28         }
29     }finally{
30         token?.close()
31         inputStream?.close()
32     }
33 }
34
35 use (StaxCategory) {
36     eachStartElement(input.openStream()){
37         element ->
38         if (element.name.toString() != 'task') return
39
40         switch (element.done){
41             case '0' :
42                 upcoming << element.title
43                 break
44             case {it != element.total} :
45                 underway << element.title
46         }
47     }
48 }
49
50
51 println 'upcoming : ' + upcoming
52 println 'underway : ' + underway
53
54 // 결과
55 // upcoming : [re-read DB chapter, use DB/XML combination]
56 // underway : [use in current project, use todo]

```

XML 처리

xml에서 정보를 얻고 무언가 변화를 주고 출력.

메모리에서 처리

일반적인 xml 처리 방식.

정보를 모두 메모리에 담고 메모리에서 처리하는 방식.

```

1  void numberfy(node){
2      def atts = node.attributes()
3      atts.keySet().grep(['capacity','total','done']).each {
4          atts[it] = atts[it].toInteger()
5      }
6
7      node.each { numberfy(it) }
8  }
9
10 void taskStatus(task){
11     def atts = task.attributes()
12     switch (atts.done) {
13         case 0 : atts.status = 'scheduled' : break

```

```

14     case 1..<atts.total : atts.status = 'in progress' ; break
15     default : atts.status = 'finished' ;
16   }
17 }
18
19 void weekStatus(week){
20   week.task.each {
21     taskStatus(it)
22   }
23
24   def atts = week.attributes()
25   atts.status = 'scheduled'
26   if (week.task.every{it.'@status' == 'finished'}) atts.status = 'finished'
27   if (week.task.any{it.'@status' == 'in progress'}) atts.status = 'in progress'
28 }
29
30 void htmlReport(builder, plan){
31   builder.html{
32     head {
33       title('Current Groovy progress')
34       link(rel:'stylesheet',type:'text/css',href:'style.css')
35     }
36     body {
37       plan.week.eachWithIndex { week, i ->
38         h1("Week No. $i: ${week.'@status'}")
39         d1{
40           week.task.each { task ->
41             dt(class:task.'@status',task.'@title')
42             dd(
43               "(${task.'@done'})/${task.'@total'}: ${task.'@status'}"
44             )
45           }
46         }
47       }
48     }
49   }
50 }
51
52 String strPath = "C:\\reakosysworkspace\\xml\\plan2.xml"
53 String strHtmlPath = "C:\\reakosysworkspace\\html"
54 def node = new XmlParser().parse(new File(strPath))
55
56 numberfy(node)
57
58 node.week.each {weekStatus(it)}
59
60 new File(strHtmlPath + 'Groovy.html').withWriter { writer ->
61   def builder = new groovy.xml.MarkupBuilder(writer)
62   htmlReport(builder, node)
63 }
64
65 /*
66 결과
67
68 <html>
69 <head>
70   <title>Current Groovy progress</title>
71   <link rel='stylesheet' type='text/css' href='style.css' />
72 </head>
73 <body>
74   <h1>Week No. 0: in progress</h1>
75   <d1>
76     <dt class='finished'>read XML chapter</dt>
77     <dd>(2/2): finished</dd>
78     <dt class='finished'>try some reporting</dt>
79     <dd>(3/3): finished</dd>
80     <dt class='in progress'>use in current project</dt>
81     <dd>(1/2): in progress</dd>
82   </d1>
83   <h1>Week No. 1: in progress</h1>
84   <d1>
85     <dt class='in progress'>re-read DB chapter</dt>
86     <dd>(0/1): in progress</dd>
87     <dt class='in progress'>use DB/XML combination</dt>
88     <dd>(0/3): in progress</dd>
89     <dt class='in progress'>use todo</dt>
90     <dd>(1/2): in progress</dd>
91   </d1>
92 </body>
93 </html>
94
95 */

```

스트림 방식

```

1   def taskStatus(task){
2     switch (task.'@done'.toInteger()) {
3       case 0 : return 'scheduled'
4       case 1..<task.'@total'.toInteger() : return 'in progress'
5       default : return 'finished'
6     }
7   }
8
9   def weekStatus(week){
10    if (week.task.every{taskStatus(it) == 'finished'}) return 'finished'

```

```

11     if (week.task.any{taskStatus(it) == 'in progress'}) return 'in progress'
12
13     return 'scheduled'
14 }
15
16 String strPath = "C:reakosysworkspacexmlplan2.xml"
17 String strHtmlPath = "C:reakosysworkspacehtml"
18
19 def plan = new XmlSlurper().parse(new File(strPath))
20
21 Closure markup = {
22     html{
23         head {
24             title('Current Groovy progress')
25             link(rel:'stylesheet',type:'text/css',href:'style.css')
26         }
27         body {
28             plan.week.eachWithIndex { week, i ->
29                 h1("Week No. $i: ${week.'@status'}")
30                 d1{
31                     week.task.each { task ->
32                         dt(class:task.'@status',task.'@title')
33                         dd{
34                             "${task.'@done'}/${task.'@total'}: ${task.'@status'}"
35                         }
36                     }
37                 }
38             }
39         }
40     }
41 }
42
43 def heater = new groovy.xml.StreamingMarkupBuilder().bind(markup)
44
45 new File(strHtmlPath + 'Groovy2.html').withWriter { it << heater}
46
47 /*
48 결과
49
50 <html>
51 <head>
52 <title>Current Groovy progress</title>
53 <link rel='stylesheet' type='text/css' href='style.css' />
54 </head>
55 <body>
56 <h1>Week No. 0: in progress</h1>
57 <d1>
58 <dt class='finished'>read XML chapter</dt>
59 <dd>(2/2): finished</dd>
60 <dt class='finished'>try some reporting</dt>
61 <dd>(3/3): finished</dd>
62 <dt class='in progress'>use in current project</dt>
63 <dd>(1/2): in progress</dd>
64 </d1>
65 <h1>Week No. 1: in progress</h1>
66 <d1>
67 <dt class='in progress'>re-read DB chapter</dt>
68 <dd>(0/1): in progress</dd>
69 <dt class='in progress'>use DB/XML combination</dt>
70 <dd>(0/3): in progress</dd>
71 <dt class='in progress'>use todo</dt>
72 <dd>(1/2): in progress</dd>
73 </d1>
74 </body>
75 </html>
76
77 */

```

XPath

자바나 그루비 코드 내에 문자열로 표시되는 표현식 (정규표현식이나 SQL).
GPath는 도트 연산자를 사용하고 XPath는 슬래시 연산자를 사용.

/plan/week/task

plan아래 있는 모든 week의 모든 task를 선택.
맨 앞의 슬래시는 최상위 요소에서 선택하라는 의미.

표 12.7 XPath 선택 기준 지정자

지정자	선택되는 노드들	단축 문법
child	바로 하위 노드	아무표시 안하거나 * 표시
parent	바로 상위 노드	
self	노드 자신 (나중에 참조하기 위해 사용)	
ancestor	모든 상위 노드	
ancestor-or-self	자신을 포함한 모든 상위 노드	//
descendant	모든 하위 노드	
descendant-or-self	자신을 포함한 모든 하위 노드	
following	이후에 XML 문서에 존재하는 모든 자신과 동일한 레벨의 노드	
following-sibling	이후에 XML 문서에 존재하는 부모가 같은 모든 동일한 레벨의 노드	@
preceding	앞쪽으로 XML 문서에 존재하는 모든 자신과 동일한 레벨의 노드	
preceding-sibling	앞쪽으로 XML 문서에 존재하는 부모가 같은 모든 동일한 레벨의 노드	
attribute	어트리뷰트 노드	
namespace	네임스페이스 노드	

표 12.8 XPath 표현식 예제들

카테고리	표시	노트
경로 연산	/, //, @, [], *, .., .	본문에서 설명했다.
조합 연산		노드 그룹 두 개를 조합한다.
불린 연산	and, or, not()	not()는 함수가 아니다.
수치 연산	+, -, *, div, mod	
비교 연산	=, !=, <, >, <=, >=	
문자열 함수	concat(), substring(), contains(), substring-before(), substringafter(), translate(), normalize-space(), string-length()	정확한 의미와 인자는 문서를 참고하자.
숫자 함수	sum(), round(), floor(), ceiling()	
노드 함수	name(), local-name(), namespace-uri()	
컨텍스트 함수	position(), last()	[n]은 [position()=n]의 단축 문법이다.
변환 함수	string(), number(), boolean()	

표 12.9 XPath 사용 예

XPath	의미와 내용	노트
/plan/week[1]	첫째 ^a week 노드	번호는 1부터 시작된다.
//task[@done<@total]	완료하지 못한 모든 task들	숫자로 자동 변환된다.
//task[@done<@total][@done>0]	진행 중인 모든 task들	각식 사이에는 암시적 인 and가 있다.
sum(//week[1]/task/@total)	첫 주의 모든 task의 시간을 합함.	숫자를 리턴한다.

a. 정확히 말하면 plan 노드 하위의 1번 위치에 있는 week 노드

XML과 분산처리

RSS ATOM 읽기

RSS(Really Simple Syndication)는 뉴스나 블로그 사이트에서 주로 사용하는 콘텐츠 표현 방식.

Atom은 웹로그나 최신 소식과 같은 웹 콘텐츠의 신디케이션을 위한 XML 기반의 문서 포맷이자, 웹로그 편집을 위한 HTTP 기반의 프로토콜.

```

1 // http://news.bbc.co.uk/rss/newsonline_uk_edition/front_page/rss091.xml
2 def base = 'http://news.bbc.co.uk/rss/newsonline_uk_edition/'
3 def url = base + 'front_page/rss091.xml'
4
5 println 'The Top three news items today:'
6 def items = new XmlParser().parse(url).channel[0].item
7
8 for(item in items[0..2]){
9     println item.title.text()
10    println item.link.text()
11    println item.description.text()
12    println '-----'
13 }
14
15 /*
16 결과
17
18 The Top three news items today:
19 Labour stalwart Tony Benn dies at 88
20 http://www.bbc.co.uk/news/uk-politics-26573929#sa=ns_mchannel=rss&ns_source=PublicRSS20-sa
21 Veteran Labour politician Tony Benn has died at home at the age of 88, his family says.
22 -----
23 US-Russia in Ukraine crisis talks
24 http://www.bbc.co.uk/news/world-europe-26572530#sa=ns_mchannel=rss&ns_source=PublicRSS20-sa
25 US and Russian envoys hold key talks in London on the Ukraine crisis, as Sunday's disputed referendum in Crimea looms.
26 -----
27 Crash peer 'concern' over helicopter
28 http://www.bbc.co.uk/news/uk-northern-ireland-26573899#sa=ns_mchannel=rss&ns_source=PublicRSS20-sa
29 A Northern Ireland peer killed in an air crash in Norfolk along with three others, had raised concerns about a helicopter he owned, the
30 -----
31
32 */

```



```

1 def atom = new Namespace("http://www.w3.org/2005/Atom")
2 def titles = new XmlParser().parse(url)[atom.entry][atom.title]
3
4
5 println titles*.text().join("\n")

```

REST

```

1  /*
2  <?xml version="1.0" encoding="UTF-8"?>
3  <user>
4    <id type="integer">995791683</id>
5    <login>Lane</login>
6    <email>highoh@gmail.com</email>
7  </user>
8  */
9
10 def url = new URL("http://localhost/users/18278273.xml")
11 def connection = url.openConnection()
12 connection.setRequestMethod("GET")
13 connection.connect()
14 def returnMessage = ""
15
16 if (connection.responseCode == 200 || connection.responseCode == 201){
17   returnMessage = connection.content.text
18
19   //print out the full response
20   println returnMessage
21
22   //parse the xml response
23   def records = new XmlParser().parseText(returnMessage)
24   println "id " + records.id.text()
25   println "login " + records.login.text()
26   println "email " + records.email.text()
27 } else {
28   println "Error Connecting to " + url
29 }

```

XML-RPC

XML-RPC란, RPC 프로토콜의 일종으로서, 인코딩 형식에서는 XML을 채택하고, 전송 방식에서는 HTTP 프로토콜을 사용.

```

1 def remote = new Proxy("http://localhost:8080/")
2 println remote.echo("Hello")

```

SOAP

SOAP(Simple Object Access Protocol)은 일반적으로 널리 알려진 HTTP, HTTPS, SMTP 등을 사용하여 XML 기반의 메시지를 컴퓨터 네트워크 상에서 교환하는 형태의 프로토콜.

```

1 public class MathService {
2   double add(double arg0, double arg1) {
3     return (arg0 + arg1)
4   }
5   double square(double arg0) {
6     return (arg0 * arg0)
7   }
8 }
9
10 def server = new SoapServer("localhost", 6980)
11
12 server.setNode("MathService")
13
14 server.start()
15
16
17 def proxy = new SoapClient("http://localhost:6980/MathServiceInterface?wsdl")
18
19 def result = proxy.add(1.0, 2.0)
20 assert (result == 3.0)
21
22 result = proxy.square(3.0)
23 assert (result == 9.0)

```

- <http://groovy.codehaus.org/api/groovy/util/XmlParser.html> - <http://groovy.codehaus.org/api/groovy/util/XmlParser.html>
 - <http://groovy.codehaus.org/api/groovy/util/XmlSlurper.html> - <http://groovy.codehaus.org/api/groovy/util/XmlSlurper.html>
-