02. 5주차. 8장 빌더사용하기

Table of Contents

경험에 패턴을 부여하는 것이 예술이다 그리고 그패턴을 알아차리는 것이 미적 기쁨이다
-알프레도 노스 화이트 헤드

무엇을 만든 building 이다 그것을 도와주는 것이 빌더에 역할이다.

B1마크업 빌더 예제

```
package _B1

import groovy.xml.*;
def builder = new MarkupBuilder();  //마크업빌더
builder.numbers{
    description 'sss';
    for(i in 10..15){
        number (value : i,square:i*i){
            for(j in 2..<i){
                if(i%j==0){
                    factor(value:j);
                }
            }
        }
    }
}

println builder;
/*결과
<numbers>
  <description>sss</description>
  <number value='10' square='100'>
    <factor value='2' />
    <factor value='5' />
  </number>
  <number value='11' square='121' />
  <number value='12' square='144'>
    <factor value='2' />
    <factor value='3' />
    <factor value='4' />
    <factor value='6' />
  </number>
  <number value='13' square='169' />
  <number value='14' square='196'>
    <factor value='2' />
    <factor value='7' />
  </number>
  <number value='15' square='225'>
    <factor value='3' />
    <factor value='5' />
```

```
  </number>
</numbers>groovy.xml.MarkupBuilder@11bed71
*/
```

B2노드빌더로 객체의 트리 구조만들기

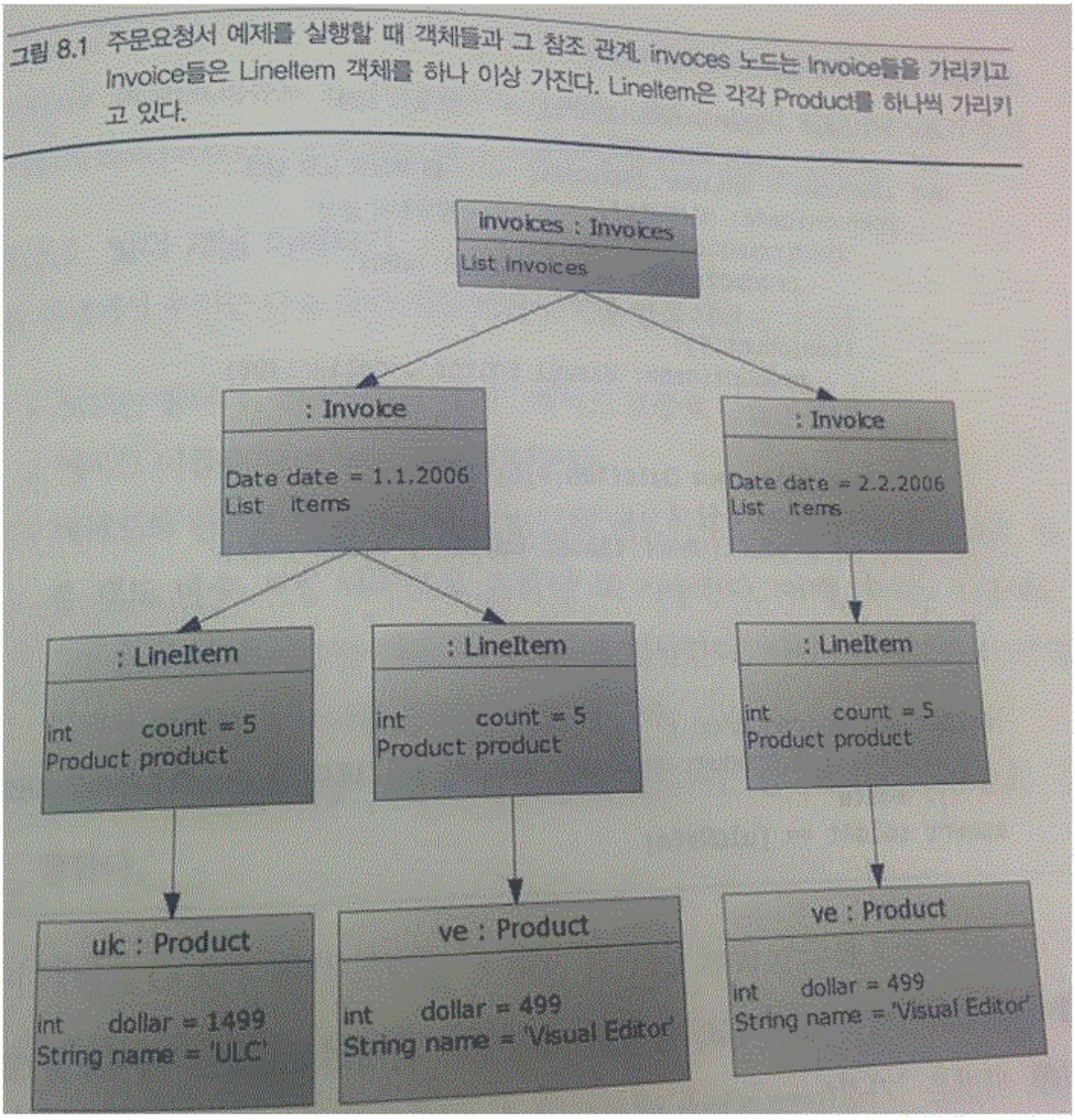


그림 Groovy in Action 발췌

```groovy
package _B2

def builder = new NodeBuilder()                         //#1
def ulcDate = new Date(107,0,1)
def invoices = builder.invoices{                        //#2
    invoice(date: ulcDate){                             //#3
        item(count:5){
            product(name:'ULC', dollar:1499)
        }
        item(count:1){
            product(name:'Visual Editor', dollar:499)
        }
    }
    invoice(date: new Date(106,1,2)){
        item(count:4) {
            product(name:'Visual Editor', dollar:499)
        }
    }
}

soldAt = invoices.grep {                                //|#4
      it.item.product.any{ it.'@name' == 'ULC' }        //|#4
  }.'@date'                                             //|#4
assert soldAt == [ulcDate]
def writer = new StringWriter()
invoices.print(new PrintWriter(writer))                 //#2
def result = writer.toString().replaceAll("r","")   //#3
print result;

/*결과
```

```
invoices() {
  invoice(date:Mon Jan 01 00:00:00 KST 2007) {
    item(count:5) {
      product(name:'ULC', dollar:1499)
    }
    item(count:1) {
      product(name:'Visual Editor', dollar:499)
    }
  }
  invoice(date:Thu Feb 02 00:00:00 KST 2006) {
    item(count:4) {
      product(name:'Visual Editor', dollar:499)
    }
  }
}
*/
```

```
package _B2


System.setProperty("user.timezone","CET")
def builder = new NodeBuilder()
def invoices = builder.invoices {
    for(day in 1..3) {                          //#1
        invoice(date: new Date(107,0,day)){
            item(count:day){
                product(name:'ULC', dollar:1499)
            }
        }
    }
}

def writer = new StringWriter()
invoices.print(new PrintWriter(writer))              //#2
def result = writer.toString().replaceAll("r","")   //#3
println result;

/*결과
invoices() {
  invoice(date:Mon Jan 01 00:00:00 CET 2007) {
    item(count:1) {
      product(name:'ULC', dollar:1499)
    }
  }
  invoice(date:Tue Jan 02 00:00:00 CET 2007) {
    item(count:2) {
      product(name:'ULC', dollar:1499)
    }
  }
  invoice(date:Wed Jan 03 00:00:00 CET 2007) {
    item(count:3) {
      product(name:'ULC', dollar:1499)
    }
  }
}

*/
```

product.name은 product['@name']이나 간단하게는 product.'@name'으로 써야된다.
골뱅이 기호로 어트리뷰트를 표시했다 이는 XPath의 표기법이다.

그루비에 빌더는 정말 대다나다~~

빌더개념 이해하기

1. 노드는 빌더의 가장(pretended) 메서드 호출로 만들어진다
2. 메서드 이름을 따라 노드 이름을 만든다
3. 메서드의 인자로 맵이 전달되면 노드의 어트리뷰트가 된다 맵의 키/값 쌍은 필드 변수 지정 메서드를 호출할때 사용한다 이때 메서드 이름에는 키를 주고 인자로 대응하는 값을 전달한다 이런 아이디어는 스윙 빌더에 이벤트 리스너를 등록할때도 쓰인다
4. 중첩 노드는 클로저를 이용해 표현한다 클로저는 빌더의 메서드 호출을 전달한다.

리턴타입	메서드이름	목적
object	name()	'invoce' 와 같은 노드의 이름
object	value()	노드자신
Map	attributes()	맵의 모든 어트리뷰트

Node	parent()	부모에 대한 참조
List	children()	모든 자식노드의리스트
Iterator	iterator()	모든자식노드에대한iterator
List	depthFirst()	깊이 우선 탐색을 이용한 모든 노드의 컬렉션
List	breadthFirst()	너비 우선 탐색을 이용한 모든 노드의 컬렉션
void	print(PrintWriter out)	중첩된 구조를 정리해서 출력

```
println invoices.depthFirst()*.name()
//[invoices, invoice, item, product, invoice, item, product, invoice, item, product]
```

B3. XML작성하기

```
package _B3

writer = new StringWriter()
builder = new groovy.xml.MarkupBuilder(writer)        //#1
invoices = builder.invoices {
    for(day in 1..3) {
        invoice(date: new Date(106,0,day)){
            item(count:day){
                product(name:'ULC', dollar:1499)
            }
        }
    }
}

result = writer.toString().replaceAll("r","")
println result;

/* 결과
<invoices>
  <invoice date='Sun Jan 01 00:00:00 KST 2006'>
    <item count='1'>
      <product name='ULC' dollar='1499' />
    </item>
  </invoice>
  <invoice date='Mon Jan 02 00:00:00 KST 2006'>
    <item count='2'>
      <product name='ULC' dollar='1499' />
    </item>
  </invoice>
  <invoice date='Tue Jan 03 00:00:00 KST 2006'>
    <item count='3'>
      <product name='ULC' dollar='1499' />
    </item>
  </invoice>
</invoices>
*/
```

B2_1 노드나 어트리뷰트에 특수문자 있을때 처리방법

```
package _B3
def writer  = new StringWriter()
def builder = new groovy.xml.MarkupBuilder(writer);

def web = builder.'web-app'{
    'display-name'('groopvyWeb1');
    builder.'display-name'('groopvyWeb2');
    builder.'display-name'(gogo:5,'-value1-');
    builder.'display-name'('go-go':'5-5','-value2-');
    displayname ('--------')
}

println writer.toString().replaceAll ("r","");
/*결과
<web-app>
  <display-name>groopvyWeb1</display-name>
  <display-name>groopvyWeb2</display-name>
  <display-name gogo='5'>-value1-</display-name>
  <display-name go-go='5-5'>-value2-</display-name>
  <displayname>--------</displayname>
</web-app>
*/
```

노드나 어트리뷰트 이름에 특수문자가 있을때는 주의해야한다. 마크업 빌더를 쓸때 이런경우가 더자주있는대
XML에서 여러 단어로 구성된 이름을 만들때 하이픈을 종종쓰기때문이다.
메서드이름에 마이너스 기호는 쓸수 없기때문에 다음처럼 따옴표로 감싸야한다

B4. html작성하기

```groovy
package _B4

//def writer = new FileWriter('markup.html')
def writer = new StringWriter();
def html   = new groovy.xml.MarkupBuilder(writer)
html.html {
  head {
    title 'Constructed by MarkupBuilder'
  }
  body {
    h1 'What can I do with MarkupBuilder?'
    form (action:'whatever') {
          for (line in ['Produce HTML','Produce XML','Have some fun']){
            input(type:'checkbox',checked:'checked', id:line, '')
            label(for:line, line)
            br('')
          }
      }
    }
  }

println writer.toString().replaceAll ("r","");
/*결과
<html>
  <head>
    <title>Constructed by MarkupBuilder</title>
  </head>
  <body>
    <h1>What can I do with MarkupBuilder?</h1>
    <form action='whatever'>
      <input type='checkbox' checked='checked' id='Produce HTML'></input>
      <label for='Produce HTML'>Produce HTML</label>
      <br></br>
      <input type='checkbox' checked='checked' id='Produce XML'></input>
      <label for='Produce XML'>Produce XML</label>
      <br></br>
      <input type='checkbox' checked='checked' id='Have some fun'></input>
      <label for='Have some fun'>Have some fun</label>
      <br></br>
    </form>
  </body>
</html>
*/
```

정말 간결하다. 코드 자체가 문서가 될정도로 간결하다.

B5 앤트 스크립트에서 그루비 스크립트로

ant

```xml
<project name="prepareBookDirs" default="copy">

  <property name="target.dir"    value="target"/>
  <property name="chapters.dir" value="chapters"/>

  <target name="copy">
    <delete dir="${target.dir}" />
    <copy todir="${target.dir}">
      <fileset dir="${chapters.dir}"
        includes="*.doc"
        excludes="~*"   />
    </copy>
  </target>
</project>
```

Groovy AntBuilder

```groovy
def TARGET_DIR ='target'
def CHAPTERS_DIR='chapters'
def ant = new AntBuilder();
ant.delete(dir:TARGET_DIR);
ant.copy(todir:TARGET_DIR){
    fileset(dir:CHAPTERS_DIR,includes:'*.doc',excludes:'~*')
};
```

안트문법과 그루비 안트빌더에 문법을 비교해보아라.. 엄청난 간결함을 자랑한다.

B6.스윙빌더로 GUI쉽게 만들어보자

자바는 GUI 짜기가 참-_- 그렇죠잉.~ 그루비 빌더를 사용하면 훨훨훨~ 수월해진다.
스윙빌더도 앞서 본 다른 빌더들과 같은 방식으로 동작한다 빌더 객체를 만들고 메서드를 호출하면 크롤저를 전달하면 이를 이용해 중첩 구조로 구성한다 그리고 인자로 전달되는 맵은 결과물의 프로포타거 된다

스윙빌더로 패스워드 읽기


```
1  package _B6
2
3  import groovy.swing.SwingBuilder
4
5  swing = new SwingBuilder()
6  frame = swing.frame(title:'Password') {
7      passwordField(columns:10, actionPerformed: { event ->
8          println event.source.text
9          // any further processing is called here
10         System.exit(0)
11         }
12     )
13 }
14 frame.pack()
15 frame.show()
```

패스워드필드에 ActionListener 를 붙일때 actionPreFormed 메서드를 호출해서 이벤트를 알리면 우리가 정의한 클로저가 호출된다.

스윙 위젯만들기

스윙 윗젯들은 스윙 빌더가 이미 알고 있는 것들이다 추가 설명이 없는경우 팩토리 메서드를 호출하면 객체를 얻을수 있고 맵인자로 전달해서 프로퍼티를 설정한다.

스윙빌더 메시지	결과물
button	JButton
buttonGroup	ButtonGroup
checkBox	JCheckBox
checkBoxMenuItem	JCheckBoxMenuItem
colorChooser	JColorChooser
comboBox	JComboBox
desktopPane	JDesktopPane
dialog	JDialog
editorPane	JEditorPane
fileChooser	jFileChooser
formattedTextField	JFormattedTextField
form	JForm
internalFrame	JInternalFrame
label	JLable
layeredPane	JLayeredPane
list	JList
menu	JMenu
menuBar	JMenuBar
menuItem	JMenuItem
optionPane	JOptionPane

panel	JPanel
passwordField	JPasswordField
popupMenu	JPopupMenu
progressBar	JProgressBar
radioBUtton	JRadioButton
radioBUttoneMenuItem	JRadioButtonMenuItem
scrollBar	JScrollBar
scrollPane	JScrollPane
separator	JSeparatr
slider	JSlider
spainner	JSpinner
splitPane	JSplitePane
tabbedPane	JTabbedPane
table	JTable
textArea	JTextArea
textField	JTextField
tetPane	JTextPane
toggleBUtton	jToggleButton
toolBar	JToolBar
tree	JTree
viewport	JViewport
Window	JWindow

JToolTip스윙빌더에없다, JApplet도 지원하지 않는다.

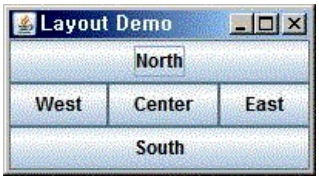
B7. 위젯만들기 예제


```
1  package _B7
2
3  import groovy.swing.SwingBuilder
4
5  swing = new SwingBuilder()
6  frame = swing.frame(title:'Demo',size:[600,20]) {
7      menuBar {
8          menu('File') {
9              menuItem 'New'
10             menuItem 'Open'
11         }
12     }
13     panel {
14         label 'Label 1'
15         slider()
16         comboBox(items:['one','two','three'])
17     }
18 }
19 frame.pack()
20 frame.show()
```

스윙빌더에서 인자로 어트리뷰트 맵을 전달할때 'text'키는 생략할수있다. 즉
menu(text:'File')은 menu('File') 가능하다

프레임의 size어트리뷰트를 설정하고 싶으면 자바에서 Dimension객체를 생성해야만 했다 그루비에서는 frame(size:[100,100]) 으로 충분하다.

B8.위젯 정돈하기




```
package _B8

import groovy.swing.SwingBuilder
import java.awt.BorderLayout as BL

swing = new SwingBuilder()
frame = swing.frame(title:'Layout Demo') {
    panel(layout: new BL()) {
        button(constraints: BL.NORTH,  'North' )
        button(constraints: BL.CENTER, 'Center')
        button(constraints: BL.SOUTH,  'South' )
        button(constraints: BL.EAST,   'East'  )
        button(constraints: BL.WEST,   'West'  )
    }
}
frame.pack()
frame.show()
```

위젯을 정돈하는 두번째 방식은 아래 나열된 메서드를 중첩 구조 내에서 호출하는것이다.
스윙의 표준 레이아웃 옵션들 외에 constraint, glue, strut 같은 객체들도 쉽게 사용할수 있다

스윙빌더 메서드	스윙 클래스/메서드	노트
borderLayout	BorderLayout	Layout Manager
boxLayout	BoxLayout	Layout manager-axis인자에 따라 다르게 펴시 디폴트값은 X_AXIS
cardLayout	CardLayout	Layout manager
flowLayout	FlowLayout	Layout manager
gridBagLayout	GridBagLayout	Layout manager
gridBagConstraints	GridBagCOnstraints	GridBagLayout과함께 사용되는 constraint
gbc	GridBagConstraints	gridBagConstraints의축약
gridLayout	GridLayout	Layout manager
overlayLayout	OverlayLayout	Layout manager
springLayout	SprintLayout	Layout manager
tableLayout	n/a	Container-내부 tr()/td()을 중첩시켜야한다
hbox	Box.createHorizontalBox	Container
hglue	Box.createHOrizontalGlue	Widget
hstrut	Box.createHorizontalStrut	Widget-height인자를 받는다 디폴트6
vbox	Box.createVerticalBox	Container
vglue	Box.createVerticalGlue	widget
vstrut	Box.createVerticalStrut	Widget-height인자를 받는다 디폴트6
glue	Box.createGlue	Widget
rigidArea	Box.createRigidArea	Widget-인자로 size나 (width,height)를받는다 디폴트6

사용자 정의 빌더 만들기

그루비의 빌더들은 모두 groovy.util.BuilderSupport상속한다 이클래스는 빌더의 일반적인 특징을 구현하다. 클로저 내의 메서드 호출을 빌더 쪽으로 전달하며 빌더에서 템플릿 메서드를 호출해준다.(중요)

빌더 만들기 (BuilderSupport 상속받기)

정보	리턴	이름	인자	호출되는상황
Abstract	Object	createNode	Object name	foo()

Abstract	Object	createNode	Object name, Object value	foo('x')
Abstract	Object	createNode	Object name,Map attributes	foo(a:1)
Abstract	Object	createNode	Object name,Map attributes,Object value	foo(a:1,'x')
Abstract void	void	setParent	Object parent, Object child	createNode완료시
Empty	void	nodeCompleted	Object parent,Object node	중첩 클로저 호출 완료시

BuilderSupport의 내부알고리즘은 다음과같다

1. 빌더의 메서드가 호출되면 적절한 createNode메서드를 호출한다
2. 방금 생성된 노드로 현재 저장된 부모의 setParent메서드를 호출한다(최상위 노드는 부모가 없으므로 예외)
3. 중첩된 클로저가 있다면 실행한다(여ㅑ기서 재귀호출)
4. 현재 부모와 생성된 노드에 nodeCompleted메서드를 호출한다 부모가 null이어도 호출)

h3 BuilderSupport에 추가메서드

리턴	이름	인자	사용
Object	getCurrent	.	현재 작성하고 있는 노드 즉 현재 클로저에서 처리중인 부모 노드
Object	getName	String methodName	빌더에서 사용할 이름 변환 규칙을 정의하려면 재정의한다 디폴트는 nameMappingClosure를 이용한다
void	setClosureDelegate	Closure closure,Object node	빌더 섞어서 사용하는것을 허용하는 경우 재정의

B9빌더 만들어보기

```groovy
package _B9
import groovy.util.BuilderSupport;

class MyBuilder extends BuilderSupport{
    protected def createNode( name) {
        println "createNode-> name : ${name}";
    }

    protected def createNode( name, Map attributes) {
        println "createNode-> name : ${name},  attributes : ${attributes}";
    }

    protected def createNode( name, Map attributes,  value) {
        println "createNode-> name : ${name},  attributes : ${attributes},  value : ${value}";
    }

    protected def createNode( name,  value) {
        println "createNode-> name : ${name},  value : ${value}";
    }
    protected void setParent( name,  child) {
        println "setParent-> name : ${name},  value : ${child}";
    }
    protected void nodeCompleted( parent,  node) {
        println "nodeCompleted-> name : ${parent},  value : ${node}";
    }
}

def builder = new MyBuilder();
builder.foo(){  //builder.createNode('foo') 호출  ,  최상위이므로 setParent()는 없다


    bar(a:1);
    //bar = bulder.createNode('bar',[a:1]);
    //builder.setParent(foo,bar);
    //bar는 처리할 클로저가 없다.
    //builder.nodeCompleted(foo,bar);
    //bulder.nodeCOmpleted(null,foo);
}

/* 결과
createNode-> name : foo
createNode-> name : bar,  attributes : [a:1]
nodeCompleted-> name : null,  value : null
nodeCompleted-> name : null,  value : null

*/
```

B10디버그 만들기

한번 추적해보시길 바랍니다

```
class DebugBuilder extends BuilderSupport {
    def result = ''<<''                         //#1
    def indent = ' ' * 4
    int indentCount = -1

    def createNode(name){                       //#2
        indentCount++
        return check(name)
    }
    def createNode(name, value){
        return check(createNode(name) << format(value))
    }
    def createNode(name, Map attributes){
        return check(createNode(name) << format(attributes))
    }
    def createNode(name, Map attributes, value){
        return check(createNode(name, attributes) << format(value))
    }
    void setParent(parent, child){
        result << "n" << indent*indentCount << child.toString()
    }
    void nodeCompleted(parent, node) {
        indentCount--
    }

    private check(descr){                       //#3
       if (!current) result << descr
       return descr
    }
    private format(value) {
        return '(' << value.toString() << ')'
    }
    private format(Map attributes) {
        StringBuffer formatted = '' << '['
        attributes.each { key, value ->
            formatted << key << ':' << value << ', '
        }
        formatted.length = formatted.size() - 2
        formatted << ']'
        return formatted
    }
}

def builder = new DebugBuilder()
builder.foo(){
    bar()
    baz('x') { map(a:1) }
}
assert "n" + builder.result == '''
foo
    bar
    baz(x)
        map[a:1]'''
```