GroovyMarkup 확장을 이용한 SWT Builder 설계

이동주°*, 지정훈*, 장한일*, 우균*, 김원영**, 최완**
*부산대학교 컴퓨터공학과
**한국전자통신연구원 디지털 홈연구단
{mrlee°, jhji, daystar, woogyun}@pusan.ac.kr, {wykim, wchoi}@etri.re.kr

The Design of SWT Builder Using Groovy Markup Extention

Dongju Lee°*, JungHoon Ji*, Hanil Jang*, Gyun Woo*, Won-Young Kim**, Wan Choi**
*Dept. of Computer Engineering, Pusan National University
**Digital Home Research Division, Electronics and Telecommunications research Institute

요 약

Java 플랫폼 기반 스크립트 언어인 Groovy는 Java와 같은 객체지향 언어지만 Java보다 훨씬 고급 수준의 언어로서 간결한 코드와 쉬운 프로그래밍 환경을 제공한다. Groovy가 제공하고 있는 기능 중에서 GroovyMarkup은 XML 문서와 같이 각각의 객체가 중첩된 트리 구조를 다루는 응용프로그램을 쉽게 생성할 수 있도록 해준다. GUI 프로그램은 Component 및 Container 객체가 중첩된 구조로 이루어져 있으므로 GroovyMarkup을 이용하면 GUI 프로그램을 간결하고 쉽게 작성할 수 있다. 본 논문에서는 Java 플랫폼 GUI 중 최근 각광 받고 있는 SWT(Standard Widget Toolkit)를 Groovy에서 지원할 수 있게 GroovyMarkup을 확장하여 SWT Builder를 설계한다. SWT Builder는 마크업 형태로 기술한 SWT widget 이름과 속성을 SWT widget 객체와 일대일로 대응하는 구조로 설계된다. 따라서 GUI 프로그램의 골격을 구성하는 SWT Builder를 마크업 형식의 코드로 작성하고 GUI 컴포넌트 내의 이벤트 처리는 클로저(closure)를 이용함으로써 좋은 성능을 내는 GUI 프로그램을 비교적 쉽게 구성할 수 있다.

1. 서 론

Java 플랫폼 기반의 객체 지향 스크립트 언어인 Groovy는 2003년 Jame Strachan과 Bob McWhirter에 의해 만들어졌다. Groovy는 정적 타입과 동적 타입을 지원하며, 리스트, 맵, 배열, JavaBean에 대한 네이티브 구문(native syntax), 함수형 언어에서 주로 사용되는 클로저(closure), GroovyMarkup과 같은 고급 기능들이 포함되어 있다. 이와 같이 Groovy는 Java에는 없는 고급기능으로 인하여 Java보다 간결하고 쉬운 프로그래밍 환경을 제공해준다[2].

이 중 GroovyMarkup은 XML문서와 같이 각각의 객체가 중첩된 트리구조를 다루는 응용프로그램을 쉽게 생성하거나 제어할 수 있다. Container와 Component의 중첩된 구조를 가지는 GUI프로그램은 GroovyMarkup의 적합한 응용대상이 될 수 있다. 따라서 현재 Groovy에서는 GroovyMarkup 기반의 Swing Builder를 제공하는데 이를 이용할 경우 간결한 코드로 Swing 프로그램을 구성할 수 있다. 하지만 Swing의 수행속도는 그렇게 빠르지 않으며 따라서 Swing Builder를 이용하여 Groovy 응용프로그램을 개발할 경우 전체적으로 수행 속도가 매우 느려지는 단점이 있다.

본 논문은 Groovy에서 간결한 코드로 비교적 좋은 성능을 내는 GUI프로그램을 구성하는 방법을 제시하고자 한다. Swing 라이브러리 대신 최근 빠른 성능으로 각광받고 있는 SWT(Standard Widget Toolkit)를 지원할 수 있도록 Groovy를 확장한다. 구체적으로 말해서 SWT에 적합하게 GroovyMarkup을 확장함으로써 SWT Builder를 설계한다.

본 논문의 구성은 다음과 같다. 2장에서는 Groovy의 핵심 기능인 GroovyMarkup의 특징과 원리를 살펴본다. 3장에서는 Swing에 비해 SWT가 가지는 장점에 대해서 분석한다. 4장에서는 SWT를 위한 Builder를 설계한다. 또한 SWT Builder를 이용하는 SWT 프로그램을 구성해 본다. 끝으로 예제 프로그램을 중심으로 SWT Builder를 평가하고 향후 발전 방향에 대해서 논한다.

2. GroovyMarkup

GroovyMarkup은 Groovy가 가지는 가장 핵심적인 기능으로 XML, HTML, SAX, W3C, DOM, GUI과 같은 다양한 응용분야의 프로그램을 마크업 형식의 표현으로 구성할 수 있도록 해주는 기능이다. GroovyMarkup의 기본적인 아이디어는 XML 문서와 같이 각각 객체가 중첩된 트리 구조를 다루는 응용프로그램을 간결한 코드로 쉽게 구성할 수 있도록 하는 것이다. 다음과 같은 구조를 가지는 XML 객체가 있다고 가정하자.

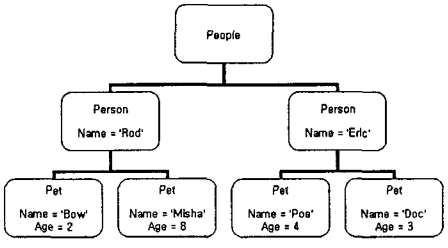

그림 1. People XML 문서의 객체 구조

최상위 태그는 People이며, Name이라는 속성을 가진 Person 태그로 구성되어있다. 또한 각 Person은 Name과 Age 속성을 가진 Pet 태그로 구성되어있다.

```
data = ['Poe':4 ,'Doc':3]
def xml = new MarkupBuilder()
px = xml.People() {
        Person(Name : 'Rod') {
          Pet(Name : 'Bow', Age : 2)
          Pet(Name : 'Misha',Age : 8)
        }
        Person(Name : 'Eric') {
          for(entry in data) {
           Pet(Name : entry.key, Age : entry.value)
          }
        }
}
```

그림 2. GroovyMarkup을 이용한 People XML

일반적으로 GroovyMarkup 도구를 Builder라고 부르는데, 응용분야에 따라 다양한 Builder가 존재할 수 있다. 그림 2는 마크업 언어를 다루는 MarkupBuilder를 사용하여 그림 1과 같은 XML 문서를 구성한 코드를 보여준다. 위 코드의 구조는 마크업 언어 형식과 매우 유사하지만 Groovy 문법에 맞게 작성된 것이다.

그림 2에서 각 노드의 객체는 MarkupBuilder의 멤버 메소드를 호출함으로써 구성된다. MarkupBuilder에서는 각 멤버 메소드의 이름을 XML 태그 이름으로 이용하며, 첫 번째 파라미터인 Map 타입의 객체는 태그가 가진 속성으로 사용된다. 두 번째 파라미터는 각 태그의 하위 태그를 나타내는데 이는 Groovy의 클로저를 이용하여 정의되어 있다. 위 예제에서 두 번째 Person의 하위 태그 Pet을 구성하는 클로저 내부에 for문을 이용하였다. Groovy에서는 클로저 내부에 함수 몸체에 해당하는 임의의 구문을 사용할 수 있다. 이처럼 GroovyMarkup은 두 개의 매개변수를 통해 Builder의 멤버 메소드를 호출함으로써 XML 문서를 쉽게 기술할 수 있다.

현재 버전의 Groovy에서는 XML과 같은 마크업 언어를 위한 MarkupBuilder, DOM객체를 위한 DOMBuilder, Swing과 같은 GUI 프로그램을 위한 Swing Builder등을 제공하고 있다.

3. SWT

SWT는 Java공개 소프트웨어 기반의 IDE 툴인 Eclipse에서 사용된 그래픽 라이브러리이다. SWT의 주요 목표는 Java 플랫폼에서 보다 적은 비용으로 좋은 성능을 내는 GUI 프로그램을 쉽게 구현 하는 것이다

SWT는 GUI 컴포넌트에 대해 운영체제가 제공하는 라이브러리(native interface)가 있으면 그 라이브러리를 사용하고, 없다면 SWT 라이브러리가 직접 운영체제 그래픽 장치를 이용해 그린다. 기존의 AWT와 Swing에서 사용하는 방법을 적절히 결합한 형태로 볼 수 있다.

SWT 라이브러리는 Swing 이나 AWT에 비해 수행속도가 빠르다[5]. 그 이유는 SWT 라이브러리가 독특한 구조를 가지고 있기 때문이다. 그림 3은 SWT 라이브러리 구조를 나타내고 있다. 그림 3에서 볼 수 있듯이 SWT 라이브러리는 동일한 인터페이스를 제공하지만 각 운영체제 마다 전혀 다르게 구현되어 있다. 엄밀히 말해 SWT는 각 운영체제에 아주 의존적이며, 각 운영체제 별로 제공되는 라이브러리로 인하여 운영체제에 독립적인 모습을 나타낸다. SWT 라이브러리의 내부 구현은 JNI(Java Native Interface)를 이용하여 호스트 운영체제의 라이브러리를 이용하는 구조로 구성되어 있다.

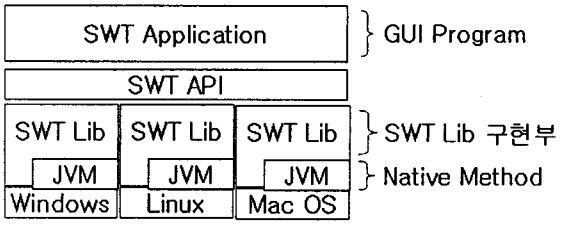

그림 3. SWT Framework

운영체제의 GUI 라이브러리를 사용하는 부분은 AWT 라이브러리 구조와 비슷하게 보이지만 실제로는 큰 차이점이 있다. AWT는 플랫폼에 동일한 구현부를 가지며 각기 다른 운영체제의 라이브러리를 사용하기 위해 동일한 인터페이스를 한 단계 더 거치기 때문에 비효율적이다. 운영체제에 아주 밀접하게 동작하는 SWT는 각 운영체제에 최적화되어 있다. 따라서 기존의 AWT나 Swing에 비해 빠른 수행 성능과 메모리 사용량이 적다는 장점이 있다.

반면 SWT가 가지는 가장 큰 걸림돌은 Java 표준 GUI 라이브러리가 아니라는 것이다. 이는 Java 버전에 따라 그 버전에 SWT도 수정해야 하며, SWT를 사용하는 개발자는 수정된 SWT 라이브러리가 나오길 기다려야 된다는 것을 의미한다. 또한 SWT는 Java 표준 패키지가 아니기 때문에 SWT로 만든 프로그램을 배포할 때 이식성 문제가 발생할 수 있다.

4. SWT Builder 설계 및 구현

새로운 Builder는 groovy.util.BuilderSupport 클래스를 상속받아 확장함으로써 구현 가능하다. BuilderSupport 클래스는 아래와 같은 추상 메소드를 포함하고 있으며 이를 SWT 객체에 맞게 재정의 함으로써 쉽게 SWT Builder를 구현할 수 있다.

```
void setParent(Object p, Object c);
Object createNode(Object name);
Object createNode(Object name, Map attr);
```

위 메소드들은 GroovyMarkup에서 사용하는 문법 구조와 아주 밀접하게 연관되어 있다. 2장에서 소개하였듯이 Builder의 멤버 메소드 호출로 객체가 생성된다고 하였다. 따라서 GroovyMarkup의 멤버 메소드 호출은 createNode 메소드와 일대일로 대응되며, 호출시 사용되는 매개변수에 따라 각각 다른 createNode 메소드가 사용된다. createNode 메소드의 첫 번째 매개변수는 호출한 멤버 메소드의 이름이며, Map 타입의 두 번째 매개변수는 생성할 객체가 가지는 속성에 대한 정보를 포함하고 있다. 클로저를 이용하여 하위노드를 구성하는 마크업 코드는 setParent 메

소드에서 그 역할을 담당하게 된다.

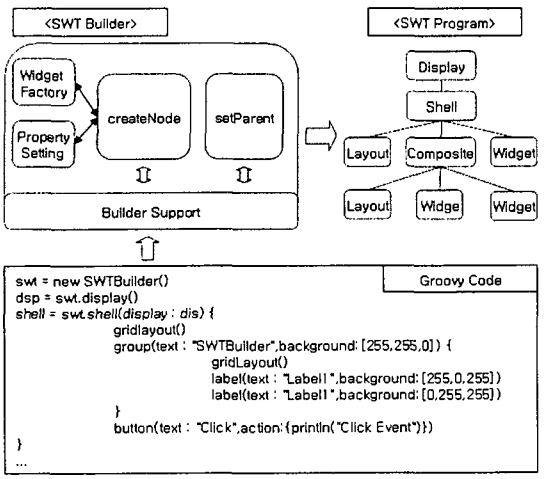

그림 4. SWT를 위한 GroovyMarkup 구조도

그림 4는 GroovyMarkup에서 SWT를 이용하는 구조도이다. GroovyMarkup형식의 Groovy 코드는 SWT Builder를 통하여 SWT 프로그램으로 변환된다. SWT Builder는 앞에서도 언급하였듯이 BuilderSupport 클래스를 상속받아 구현하며, GroovyMarkup을 구현하는데 공통적으로 사용되는 기본적 모듈은 부모 클래스에서 제공하고 있다.

그림 4에서 볼 수 있듯이 SWT 프로그램 구조는 최상위 Display, Shell, Composite, Widget과 같은 순서의 트리구조이다. createNode에서는 각각의 SWT 객체를 Widget Factory를 이용하여 생성한다. Widget Factory에는 (마크업이름 : Widget클래스)와 같은 데이터 형태로 모든 SWT Widget이 등록되어 있다. Factory를 이용할 경우 createNode의 첫 번째 매개변수가 전달하는 마크업 이름을 이용하여 객체를 생성하게 된다. 따라서 Builder의 멤버 메소드 이름과 SWT Widget 객체가 일대일로 대응되는 구조로 설계된다.

Groovy에서는 기본적으로 유니코드를 지원하므로 변수나 메소드 이름을 한글로 구성할 수 있다. Factory에 등록된 마크업 이름을 한글로 수정할 경우 GroovyMarkup을 한글화하는 것이 가능하다. 예를 들어 ("버튼": Button)와 같이 마크업 이름을 한글로 구성할 경우, 실제 Groovy 코드에서는 "버튼"이라는 키워드로 Button 객체를 생성할 수 있게 된다.

SWT Widget 객체의 속성 값이나 Action에 대한 설정은 Property Setting 모듈에서 담당하게 된다. createNode는 첫 번째 매개변수로 들어온 이름을 통하여 Factory에서 객체를 생성한 뒤 두 번째 매개변수와 Property Setting을 이용하여 생성된 객체를 설정한다. 이러한 방식을 이용함으로써 비교적 간단한 구조로 설계할 수 있었다.

그림 4에 나타낸 것과 같이 SWT 프로그램은 GUI를 구성하는 각각의 객체가 트리 구조를 가지게 된다. 이는 createNode 메소드에서 생성된 각각의 객체가 setParent 메소드를 통해 각 객체간의 관계를 형성한다. setParent 메소드는 부모노드와 자식노드에 해당하는 두 객체를 매개변수로 받아 자식 노드에 부모 노드객체의 참조를 등록한다. 예를 들어 부모노드 객체가 Composite 이고 자식노드가 Widget 일 경우 Widget의 setParent(Composite parent) 메소드를 이용하여 부모노드 객체의 참조를 설정한다. SWT Builder는 createNode 메소드와 setParent 메소드를 재정의함으로써 비교적 쉽게 구성할 수 있다.

그림 4의 Groovy Code는 SWT Builder를 이용하여 GUI 프로그램을 구성한 예를 나타낸 것이다. 비록 프로그램 자체가 매우 간단하지만 마크업 형태로 코드를 작성함으로써 간결하게 프로그램을 구성할 수 있음을 알 수 있다. 또한 이벤트 처리 코드 부분은 익명 내부 클래스를 이용하는 Java에 비해 클로저를 이용하여 깔끔하게 구성할 수 있음을 볼 수 있다.

5. 결론

Java 플랫폼 기반의 스크립트 언어인 Groovy는 기존에 Java에 비해 빠른 개발, 간결한 코드, 쉬운 프로그래밍 환경을 위하여 다양한 기능을 제공한다. 이중 대표적인 기능인 GroovyMarkup은 매우 단순한 문법적 구조를 가지고 있었으며, 이를 이용하면 간결한 형태로 응용프로그램을 구성할 수 있었다. 특히 마크업 언어가 트리구조를 간결히 표현할 수 있으므로 사용자로 하여금 쉽게 GUI프로그램을 구성할 수 있도록 한다.

본 논문에서는 GroovyMarkup이 가지는 장점과 Swing에 비해 빠른 성능을 가지는 SWT를 이용하여 Groovy에서 GUI 프로그램을 구성하는 방법을 제시하였다. 이를 위해 SWT 프로그램 구조에 적합하게 GroovyMarkup을 확장하였으며, SWT Builder를 설계하였다. GroovyMarkup에서는 다양한 응용 분야의 Builder를 쉽게 구현할 수 있도록 쉬운 인터페이스를 제공하고 있으며, SWT Builder 또한 이를 이용하여 쉽게 설계할 수 있다.

이 논문에서 제시한 SWT Builder에서는 GUI 프로그램의 골격은 마크업 형식의 코드로 작성하고 Widget 내부의 이벤트 처리는 클로저를 이용함으로써 깔끔한 구조로 GUI 프로그램을 쉽게 구성할 수 있도록 하고 있다.

참고문헌

[1] Groovy Homepage, http://groovy.codehaus.org
[2] Rod Cope and James Strachan, The Groovy Programming Language: Let's Get Groovy, Presentation at 2004 JavaOne, Conference, 2004. http://www.codehaus.org/~jstrachan/GroovyJavaOne-2004.pdf
[3] Marc Hedlund, G$D : Basic SwingBuilder, Oct. 2004. http://www.oreillynet.com/pub/wlg/5790
[4] SWT: The Standard Widget Toolkit, http://www.eclipse.org/swt
[5] Mauro Marinilli, Swing and SWT: A Tale of Two Java GUI Libraries, 2004. http://www.developer.com/java/other/article.php/10936_2179061_1
[6] M.Scarpino, S. Ng, and L. Mihalkovic, SWT/JFace in Action: GUI Design with Eclipse 3.0 (In Action series). Manning Publications Co, 2004