RSS 피드 파서의 구현을 위한 패키지의 설계

이동규* · 김윤호**

A Package Design for an RSS Feed Parser Implementation

Dong-Kyu Lee* · Yun-Ho Kim**

요 약

RSS는 웹 사이트에서 제작한 문서를 배포하기 위한 문서 규격이다. 이 문서 규격에 맞게 제작되어 배포되는 문서를 RSS 피드 또는 피드라고 한다. 본 논문은 수집한 RSS 피드를 분석하여 정보를 저장하는 RSS 피드 파서를 위한 패키지의 설계를 제시한다. RSS 명세서에 의거하여 RSS 피드의 문법과 이를 인식하는 오토마타를 설정한다. 이에 의거하여 RSS 피드의 구성요소의 인식 기능을 하는 클래스들을 그룹화하는 방법으로 패키지를 설계하고, 이들 클래스의 예외처리를 위한 클래스를 설계한다.

ABSTRACT

The RSS is a document format for syndicate a document maded by a WebSite. A document that made and syndicated to fit for this format is called a RSS feed or a feed. This paper presents the design of package for an RSS feed parser that analyzes an RSS feed and saves its informations. according to the RSS specification. This paper establishes an RSS grammar and an automata recognizing that grammar. Based on the automata, we design a package for classes that recognize elements of an RSS feed.

키워드

Web syndication, RSS Feed, RSS parser, RSS reader

I. 서 론

서비스를 제공하기 위해 늘어 가는 웹 사이트에서 원하는 정보를 찾기란 모래사장에서 바늘 찾기와 같다. 모래사장을 웹 사이트라고 하면 바늘은 웹 사이트에서 제공하는 서비스 및 정보들인 콘텐츠라 할 수 있다. 서비스 제공 및 사용자들을 위하여 서로 간에 원하는 콘텐츠의 배포, 수집하기 위한 많은 노력이 있었는데, 이러한 노력의 일안으로 RSS(Really Simple Syndication)[1][2]가 대두되었다. 즉, RSS는 웹 사이트에서 콘텐츠를 규격화된 문서로 제작하여 배포하면, 사용자가 배포한 문서를 수집하여 원하는 콘텐츠만을 확인 할 수 있는 문서 규격이다[2]. 또한 RSS에 규격화된 문서를 RSS 피드(Feed) 또는 피드(이하 피드)라고 한다[2]. 웹 사이트에서 콘텐츠를 배포하기 위해 제작한 피드를 공개하면 사용자는 RSS 구독 애플리케이션을 이용하여 피드를 수집한다. 웹 기반과 데스크 탑 기반으로 분류되는 RSS 구독기[3][4][5][6][7][8]는 웹 사이트에서 제공하는 피드를 수집하고, 수집한 피드를 분석하여 콘텐츠 정보를 추출하며, 추출한 정보를 화면에 출력한다 [9]. RSS 구독기를

* (주) 넥스트리소프트 연구원
** 안동대학교 전자정보산업학부 부교수

접수일자 : 2006. 11. 9

이용하면 직접 웹 사이트에 접근하는 번거러움을 덜 수 있고, 급변하는 웹 사이트의 콘텐츠를 손쉽게 확인할 수 있다는 이점을 얻을 수 있다. 콘텐츠 제공자와 사용자가 피드를 배포, 구독하는 환경은 그림 1과 같다.

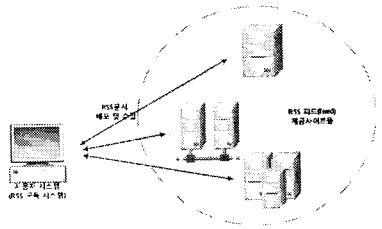

그림 1. RSS 구독기의 사용 환경
Fig. 1. Environment of using RSS Reader

RSS 구독기에 탑재되는 RSS 피드 파서는 수집한 피드를 분석하고, 분석한 결과를 저장하여 피드의 콘텐츠를 자유자재로 사용할 수 있도록 하는 핵심 컴포넌트이다 [10][11][12]. 본 논문은 위의 그림 1에서 보인 피드를 구독하기 위한 시스템의 피드 파서의 설계를 제시한다. 피드 파서의 설계를 위하여 RSS 2.0 명세서에 의거하여 설정한 피드의 문법을 EBNF(Extended Backus-Naur Form)[13] 와 구문 흐름도로 보이고, 피드를 인식하는 오토마타(automata)[13]의 설계 결과를 보인다. 이 결과에 의거한 피드 파서를 위한 설계는 객체지향 설계 방법론의 모델링 언어인 UML(Unified Modeling Language) [14][15]에서 클래스도 및 패키지 도를 이용하여 설계하도록 한다.

본 논문은 2장에서 피드의 문법을 설정하기 위하여 피드의 구조와 설정한 문법, 그리고 피드를 인식하는 오토마타의 설계 결과를 보인다. 3장은 설정한 문법 및 설계한 오토마타에 의거하여 피드의 구성요소의 인식 기능을 하는 클래스를 설정하고 패키징하여 설계한 패키지의 결과를 보이며, 클래스에서 발생할 수 있는 예외 사항 처리를 위한 클래스를 설계하였다. 그리고, 4장은 본 논문에서 설계한 RSS 피드 파서의 실험 결과를 RSS 구독기에 적용하여 보인다. 마지막으로 5장에서는 본 논문의 결론을 맺고, 향후 연구 방안에 대하여 논한다.

Ⅱ. RSS 피드

2.1 RSS 피드 구조

XML(Extensible Markup Language)[16]의 표현 형식을 따르는 피드는 여러 기관에서 공개한 여러 버전[2][17]이 있으나 본 논문에서는 현재 가장 많이 사용하고 있는 RSS 2.0 버전을 다루기로 한다. RSS 2.0 명세서(이하 RSS 명세서)에 의거하여 피드를 도식화 하면 그림 2와 같다. 그림 2에서 보인 RSS 2.0 명세서에서 정의한 피드는 피드의 본체인 XML 선언문과 rss 엘리먼트를 포함한다. rss 엘리먼트는 RSS 버전 정보를 나타내는 'version' 애트리뷰트를 포함하며, 웹 사이트의 메타 정보들을 포함하는 channel 엘리먼트를 필수적으로 포함한다. 그리고, channel 엘리먼트는 20 종류의 하부 엘리먼트를 포함하는데, titie, link, description 엘리먼트는 필수적으로 포함하며 기타 엘리먼트는 선택적으로 포함한다. 선택적으로 포함되는 엘리먼트 중 웹 사이트의 콘텐츠 정보를 포함하는 item 엘리먼트는 channel 엘리먼트에 여러 개 포함될 수 있다. item 엘리먼트는 title, link, description등 총 10 종류의 하부 엘리먼트를 선택적으로 포함할 수 있다.

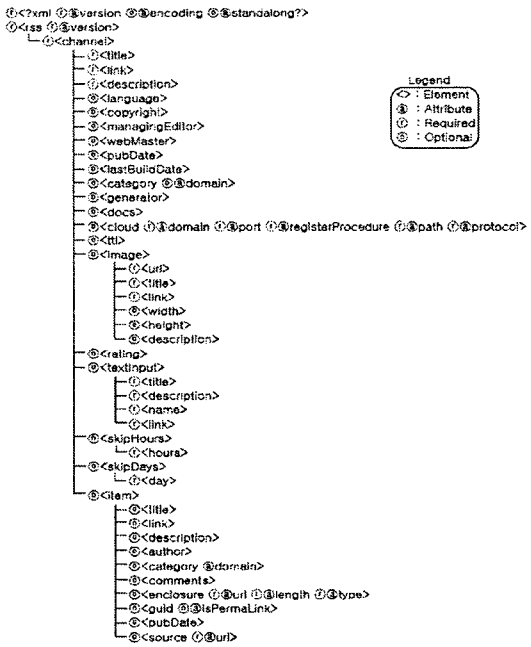

그림 2. RSS 피드의 구조 및 구성요소
Fig. 2. A structure and elements of the RSS Feed

2.2 RSS 피드 문법 및 오토마타

본 절에서는 피드의 분석을 위하여 RSS 문법을 설정한 결과를 기술한다. RSS 명세서에 규정된 정의로 부터 피드의 구성요소와 문법규칙을 추출하였으며, EBNF를 사용하여 표기하였다. 또한 피드의 표현 형식은 XML의 표현 형식을 따르므로, 이와 관련된 XML 선언문과 주석은 XML 명세서의 규칙에 따라 설정하였다[16]. 또한 피드의 본체인 rss 엘리먼트의 EBNF 설정 작업에서, 피드의 각 부분들의 넌터미널(nonterminal) 명은 각 부분들이 내포하는 특징이 이름에 드러나도록 하였다. RSS 명세서에 의거하여 피드를 EBNF로 설정한 결과를 보이면 그림 3과 같다.

```
<feed> ::= <xmldecl> {<misc>} <rsselement>

<xmldecl> ::= '<?' 'xml' <xmlverinfo> [<encodingdecl>] [<sddecl>] [<space>] '?>'
<xmlverinfo> ::= <space> 'version' <eq> ('" <xmlvernum> '" | '" <xmlvernum> '")
<xmlvernum> ::= '1.0' | '1.1'
<encodingedcl> ::= <space> 'encoding' <eq> ('" <encname> '" | '" <encname> '" )
<encname> ::= <latinchar> {<symbol> (<latinchar>| <number>)}
<sddecl> ::= <space> 'standalone' <eq> (('" ('yes' | 'no') '") | ('" ('yes' | 'no') '"))
<eq> ::= [<space>] '=' [<space>]

<comment> ::= '<!--' {(<char> - '--') | ('-' (<char> - '--'))} '-->'
<space> ::= <blank>{<blank>}
<blank> ::= '#x9' | '#x20' | '#xA' | '#xD'
<misc> ::= <comment> | <space>

<rsselement> ::= '<rss' <rssverinfo> [<space>] '>' {<misc>} <channelelement> {<misc>} '</rss>'
<rssverinfo> ::= <space> 'version' <eq> '" <rssvernum> '")
<rssvernum> ::= '0.90' | '0.91' | '0.92' | '1.0' | '2.0'
<title> ::= '<' 'title' > {misc} <contents> {misc} '</' 'title' '>'
<description> ::= '<' 'description' '>' {<misc>} <contents> {<misc>} '</' 'description' '>'
<category> ::= '<' 'category' {<categorydomain>} {<space>}'>' {<misc>} <contents>
{<misc>} '</' 'category' '>'
<categorydomain> ::= <space> 'domain' <eq> '" <contents> '"
<enclosure> ::= '<' 'enclosure' <enclosureurl> <enclosurelength> <enclosuretype> {<space>} '/>'
<enclosureurl> ::= <space> 'url' <eq> '"<contents>'"
<enclosurelength> ::= <space> 'length' <eq> '"<number>'"
<enclosuretype> ::= <space> 'type' <eq> '"</latinchar> '/' <latinchar>'"
<guid> ::= '<' 'guid' <guidsper> {<space>} '>' {<misc>} <contents> {<misc>} '</' 'guid' '>'
<guidsper> ::= <space> 'isPermalink' <eq> '"' ('true' | 'false') '"'
<source> ::= '<' 'source' <sourceurl> {<space>} '>' {<misc>} <contents> {<misc>} '</' 'source' '>'
<sourceurl> ::= <space> 'url' <eq> '" <contents> '"
<itemelelement> ::= '<' <itemelctag> '>' {<misc>} <contents> {<misc>} '</' <itemelctag> '>'
<itemelctag> ::= 'link' | 'author' | 'comments' | 'pubDate'
<itemsubelement> ::= <title> | <description> | <category> | <enclosure> | <guid> | <itemelelement> | <source>
<item> ::= '<' 'item' '>' {<misc>} (<title> | <description>) {<misc>} {<itemsubelement>}
{<misc>} '</' 'item' '>'
<chimage> ::= '<' 'image' '>' {<misc>} <imagereqelement> {<imagereqelement>} {<misc>}
{<imageoptelement>} {<misc>} '</' 'image' '>'
<imagereqelement> ::= '<' <imagereqtag> '>' {<misc>} <contents> {<misc>} '</' <imagereqtag>
'>'
<imagereqtag> ::= 'url' | 'title' | 'link'
<imageoptelement> ::= '<' <imageopttag> '>' {<misc>} <contents> {<misc>} '</' <imageopttag> '>'
<imageopttag> ::= 'width' | 'height' | 'description'
<chcloud> ::= '<' 'cloud' <cloudparameter> {<cloudparameter>} {<space>} '/>'
<cloudparameter> ::= <space> <cloudparaname> <eq> '" <contents> '"
<cloudparaname> ::= 'domain' | 'port' | 'registerProcedure' | 'path' | 'protocol'
<chtextinput> ::= '<' 'textInput' '>' {<misc>} {<textinputsubelement>} {<misc>} '</' 'textInput' '>'
<textinputsubelement> ::= '<' <textinputtag> '>' {<misc>} <contents> {<misc>} '</' <textinputtag> '>'
<textinputtag> ::= 'title' | 'description' | 'name' | 'link'
<chskipHours> ::= '<' 'skipHours' '>' {<misc>}
    <skipHourssubelement> {<skipHourssubelement>} {<misc>} '</' 'skipHours' '>'
<skipHourssubelement> ::= '<' 'hours' '>' {<misc>} <number> {<misc>} '</' 'hours' '>'
<chskipDays> ::= '<' 'skipDays' {<space>} '>' {<misc>}
    <skipDayssubelement> {<skipDayssubelement>} {<misc>} '</' 'skipDays' '>'
<skipDayssubelement> ::= '<' 'day' '>' {<misc>} <latinchar> {<misc>} '</' 'day' '>'
<etcoptelement> ::= '<' <etcopttag> '>' {<misc>} <contents> {<misc>} '</' <etcopttag> '>'
<etcopttag> ::= 'language' | 'copyright' | 'managingEditor' | 'webMaster' | 'pubDate' generator' |
'docs' | 'ttl' | 'rating' | 'lastBuildDate'

<optelement> ::= <chimage> | <chcloud> | <chtextInput> | <chcategory> | <chskipHours> |
<skipDays>) | <etcoptelement> | <item>
<reqelement> ::= '<' <reqtag> '>' {<misc>} <contents> {<misc>} '</' <reqtag> '>'
<reqtag> ::= 'title' | 'link' | 'description'
<channelelement> ::= '<' 'channel' '>' {<misc>} <reqelement> {<reqelement>} {<misc>}
  {<optelement>} {<misc>} '</' 'channel' '>'

<char> ::= '#x9' | '#xA' | '#xD' | ['#x20 - #xD7FF'] | ['#xE00 - #xFFFD'] | ['#x1000 - #x10FFFF']
<contents> ::= <char>{<char>}
<latin> ::= [A-Za-z]
<latinchar> ::= <latin> {<latin>}
<digit> ::= [0-9]
<number> ::= <digit> {<digit>}
<symbol> ::= . | _ | -
```

그림 3. 피드의 문법
Fig. 3. The grammer of the Feed

위의 그림 3의 모든 요소들 중 피드의 핵심적인 엘리먼트인 rss 엘리먼트의 문법의 설정방법만을 설명하도록 한다. RSS 명세서에서는 rss 엘리먼트를 '<rss version="2.0">... </rss>'로 정의 하고 있다. 이 정의에 의거하여 rss 엘리먼트를 EBNF로 설정하면, rss 엘리먼트의 넌터미널 이름을 '<rsselement>'로 명명하고, rss 엘리먼트의 시작 태그인'<rss'를 터미널로 설정하며, RSS 버전을 나타내는 'version="2.0"'부분은 넌터미널'<rssverinfo>'로 설정하며, 터미널 '>'로 rss 엘리먼트의 시작 태그를 종결 하도록 하였다. rss 엘리먼트에 포함되는 channel 엘리먼트는 넌터미널'<channelelement>'로 설정하였고, 시작 엘리먼트와 종결 엘리먼트의 사이에는 주석과 빈 공간이 포함할 수 있으므로 이를 설정한 넌터미널 '<misc>'을 설정하였다. 마지막으로 rss 엘리먼트는 '</rss>'로 종결하므로, '</rss>'를 터미널로 표기하여 rss 엘리먼트의 문법을 설정하였다. 그림 3에서 보인 피드의 구성요소들은 RSS 명세서에 의거하여 rss 엘리먼트의 문법을 설정한 방법으로 문법을 설정한 것이다. 그림 3의 점선은 XML 선언문, 주석 및 빈 공간, rss 엘리먼트 및 피드에서 사용하는 문자들을 구분하기 위하여 사용하였다. 그림 3에서 보인 문법에서 피드 전체를 나타내는 '<feed>'를 구문 흐름도로 나타내면 그림 4와 같다.

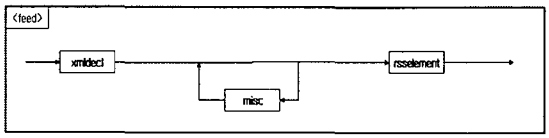

그림 4. 피드 문법의 구문 흐름도
Fig. 4. The grammer and syntax diagram of the Feed

그림 4에서 보인 바와 같이 XML 선언문(<xmldecl>)과 RSS엘리먼트(<rsselement>)는 필수적으로 포함하고 주석 및 빈 공간(<misc>)은 연속, 선택적으로 포함할 수 있음을 나타낸다. 그림 3에서 보인 피드 문법 설정 결과를 검증하기 위하여 피드를 인식하는 오토마타를 설계하면 그림 5(부록)와 같다.

그림 5(부록)에서 보인 피드를 인식하는 오토마타는 그림 3에서 보인 피드 문법에 의거하여 피드를 인식할 수 있는 범위 내에서 공통적인 부분을 묶어 간략화 하였다. 이렇게 설계함으로서 오토마타의 이해도를 높이고, 오토마타에 의거한 피드의 파서의 설계 및 구현 시 개발 시간 단축 및 소스 코드의 길이를 줄일 수 있다. 또한, 이러한 특징으로 소스코드의 가독성을 향상 등의 장점을 내재할 수 있다.

Ⅲ. 패키지 설계

본 장에서는 2장에서 설정한 피드의 문법에 의거하여 피드 파서를 구현하기 위한 패키지 설계를 보인다. 패키지 설계에 앞서 피드 인식 클래스 설계는 앞에서 설정하고 EBNF로 나타낸 피드 문법의 형식을 따라, 각각의 엘리먼트 및 애트리뷰트를 각각의 클래스로 설정하여 인식 기능을 독립적으로 이행하도록 하였다. 이렇게 하면 필수, 선택적으로 포함하는 엘리먼트 및 애트리뷰트의 특징을 그대로 반영할 수 있어, 이들의 인식 기능을 하는 모든 클래스의 인스턴스를 생성할 필요가 없으므로 피드 파서의 시스템의 자원 관리에 효율성을 높일 수 있다. 또한, 엘리먼트에 포함되는 주석 및 빈 공간에 관련된 클래스를 공유하여 불필요한 중복 클래스를 줄일 수 있도록 하였고, 클래스의 재사용성을 높였다. 피드의 인식 기능을 이행하는 클래스들의 패키지 구성은 먼저 XML 선언문, 주석 및 빈 공간, rss 엘리먼트등 피드의 최상단 구성요소의 인식 기능을 이행하는 클래스들을 패키징하고 rss 엘리먼트와 RSS 버전을 나타내는 구문을 인식하는 클래스들을 패키징하였다. 또한 rss 엘리먼트내에 포함되는 channel 엘리먼트를 인식하는 클래스들의 패키지를 설정하였고, channel 엘리먼트의 필수 엘리먼트들을 인식하는 클래스들의 패키지와 선택 엘리먼트를 인식하는 패키지로 분류하여 설정하였다. 그리고, 선택 엘리먼트에 포함되는 item 엘리먼트 패키지를 설정하였다. 이렇게 하면 2장에서 설정한 피드의 문법을 직관적으로 반영할 수 있어, 패키지의 설계에 대한 이해도를 높일 수 있다. 본 장은 피드 파서의 설계한 패키지들 중 피드의 본체인 rss 엘리먼트의 구성 요소들부터 기술하도록 한다.

3.1 RSS 피드의 item 엘리먼트처리 패키지

RSS 명세서는 피드의 item 엘리먼트에 title, link 등과 같은 10개의 하부 엘리먼트를 선택적으로 포함할 수 있도록 명시하고 있다[2]. 본 논문에서는 item 엘리먼트의 인식 기능을 하는 패키지를 앞에서 나타낸 피드 문법 설정 결과에 의거하여 문법이 같은 형식인 엘리먼트의 인식 기능을 하는 클래스를 하나로 할당하고, 나머지 엘리먼트와 애트리뷰트를 인식하는 클래스들을 할당하며 각 클래스에서 실제로 인식 기능을 이행 하는 메소드들을 정의하였다. 그리고 RSS 명세서에 의거하여 선택적으로 포함되는 엘리먼트의 인식 기능을 하는 클래스는 모두 '0..1' 관계로 나타내었다. item 엘리먼트의 인식 기능을 이행하는 패키지의 설계 결과를 보이면 그림 6과 같다.

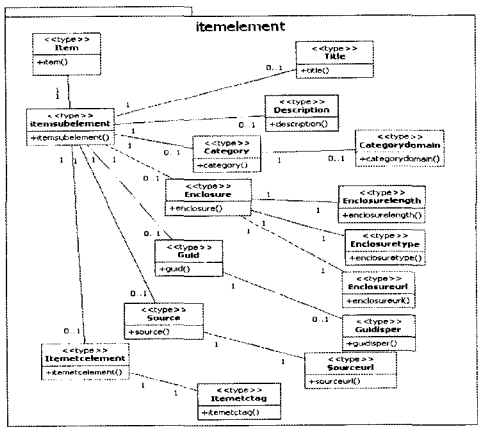

그림 6. item 엘리먼트의 인식 기능을 이행하는 패키지도

Fig. 6. The package diagram that function as recognizing a item element

그림 6의 패키지도에서 스테레오 타입[18]으로 기술한 '<<type>>'의 의미는 앞에서 EBNF로 보인 피드의 문법의 각 넌터미널들의 타입을 의미한다. 그림 6의 클래스 및 메소드에 대한 설명은 표 1과 같다.

표 1. item 엘리먼트의 인식 기능을 이행하는 클래스
Table. 1 The classes that function as recognizing item elements

클래스 명	메소드 명	내 용
Item	item()	item 엘리먼트의 인식기능을 이행한다.
Itemsubelement	itemsubelement()	item 엘리먼트의 하부 엘리먼트의 인식 기능을 이행한다. 본 클래스와 '0..1' 관계로 연관된 Title, Description등의 클래스는 선택적으로 연관됨을 의미한다.
Title	title()	title 엘리먼트의 인식 기능을 이행한다.
Description	description()	description 엘리먼트의 인식 기능을 이행한다.
Category	category()	category 엘리먼트의 인식 기능을 이행한다.
Cateogydomin	cateogydomain()	category의 domain 애트리뷰트의 인식 기능을 이행한다. category 엘리먼트에 선택적으로 포함되기 때문에 '0..1' 관계로 나타낸다.
Enclosure	enclosure()	enclosure 엘리먼트의 인식 기능을 이행한다.
Enclosurelength	enclosurelength()	enclosure의 length 애트리뷰트의 인식 기능을 이행한다.
Enclosurety pe	enclosurety pe()	enclosure의 type 애트리뷰트의 인식 기능을 이행한다.
Enclosureurl	enclosureurl()	enclosure의 url 애트리뷰트의 인식 기능을 이행한다.
Guid	guid()	guid 엘리먼트의 인식 기능을 이행한다.
Guidisper	guidisper()	guid 엘리먼트의 isPermaLink 애트리뷰트의 인식 기능을 이행한다. guid 엘리먼트에 선택적으로 포함되기 때문에 '0..1' 관계로 나타낸다.
Source	source()	source 엘리먼트의 인식 기능을 이행한다.
Sourceurl	sourceurl()	source 엘리먼트의 url 애트리뷰트의 인식 기능을 이행한다.
Itemetcelement	itemetcelement()	item 엘리먼트의 기타 엘리먼트의 인식 기능을 이행한다.
Itemetctag	itemetctag()	item 엘리먼트의 기타 엘리먼트이름의 인식 기능을 이행한다.

3.2 chanel 엘리먼트의 선택 엘리먼트처리 패키지

RSS 명세서에서 정의한 피드의 channel 엘리먼트의 선택 엘리먼트는 item 엘리먼트를 포함하여 language, copyright등과 같은 총 17개의 엘리먼트를 포함한다. 3.1절에서 item 엘리먼트 처리 패키지에서 보인 바와 같이 선택 엘리먼트 처리 패키지 또한 2장에서 보인 피드의 문법을 나타낸 넌터미널을 클래스로 설정하고, 이들의 관계를 '0..1' 관계로 설정하였으며, 이들을 패키징하는 방법으로 설계하였다. 앞 장에서 설정한 문법에 의거하여 channel 엘리먼트의 선택 엘리먼트의 인식 기능을 이행하는 패키지를 설계한 결과는 그림 7(부록)과 같다. 그림 7(부록)의 패키지도의 클래스 및 메소드에 대한 설명은 표 2와 같다. itemelement패키지의 설정 내용은 위의 3.1절을 참조한다.

표 2. 선택 엘리먼트의 인식 기능을 이행하는 클래스
Table. 2 The classes that function as recognizing option elements

클래스 명	메소드 명	내 용
Optelement	optelement()	channel 엘리먼트의 선택 엘리먼트의 인식 기능을 이행한다.
Chimage	chimage()	image 엘리먼트의 인식 기능을 이행한다.
Imagereqelement	imagereqelement()	image 엘리먼트의 필수 엘리먼트의 인식 기능을 이행한다.
Imagereqtag	imagereqtag()	image 엘리먼트의 필수 엘리먼트이름의 인식 기능을 이행한다.
Imageoptelement	imageoptelement()	image 엘리먼트의 선택 엘리먼트의 인식 기능을 이행한다.
Imageopttag	imageopttag()	image 엘리먼트의 선택 엘리먼트이름의 인식 기능을 이행한다.
Chcloud	chcloud()	cloud 엘리먼트의 인식 기능을 이행한다.
Chcloudparameter	chcloudparameter()	cloud 엘리먼트의 parameter 애트리뷰트의 인식 기능을 이행한다.
Chcloudparame	chcloudparame()	parameter 애트리뷰트의 이름의 인식 기능을 이행한다.
ChtextInput	chtextInput()	textInput 엘리먼트의 인식 기능을 이행한다.
TextInput-subelement	textInput-subelement()	textInput 엘리먼트의 하부 엘리먼트의 인식 기능을 이행한다.
TextInputtag	textInputtag()	textInput 엘리먼트의 하부 엘리먼트이름의 인식 기능을 이행한다.
ChskipHours	chskipHours()	skipHours 엘리먼트의 인식 기능을 이행한다.
Skiphours-subelement	skiphours-subelement()	skipHours 엘리먼트의 하부 엘리먼트의 인식 기능을 이행한다.
ChskipDays	chskipDays()	skipDays 엘리먼트의 인식 기능을 이행한다.
Skipdays-subelement	skipdays-subelement()	skipDays 엘리먼트의 하부 엘리먼트의 인식 기능을 이행한다.
Etcoptelement	etcoptelement()	channel 엘리먼트의 기타 엘리먼트의 인식 기능을 이행한다.
Etcopttag	etcopttag()	기타 엘리먼트의 이름의 인식 기능을 이행한다.

3.3 피드의 channel 엘리먼트의 필수 엘리먼트처리 패키지

RSS 명세서에서 피드의 channel 엘리먼트의 필수 엘리먼트는 3개의 엘리먼트를 포함한다고 정의하고 있다. 2장에서 설정한 피드 문법에 의거하면 channel 엘리먼트의 필수 엘리먼트의 인식 기능을 하는 클래스는 엘리먼트전체를 인식하는 클래스와 엘리먼트의 이름을 인식하는 클래스로 설정할 수 있다. 그리고 이들은 필수적인 관계이므로 1:1 관계를 표시할 수 있다. 이와 같은 방법으로 channel 엘리먼트의 필수 엘리먼트의 인식 기능을 이행하는 클래스들을 설계하여 패키징한 결과는 그림 8과 같고, 그림 8의 클래스와 메소드의 설명은 표 3과 같다.

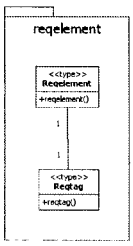

그림 8. 필수 엘리먼트의 인식 기능을 이행하는 패키지도
Fig. 8 The package diagram that function as recognizing require elements

표 3. 필수 엘리먼트의 인식 기능을 이행하는 클래스
Table. 3 The classes that function as recognizing require elements

클래스 명	메소드 명	내 용
Reqelement	reqelement ()	channel 엘리먼트의 필수 엘리먼트의 인식 기능을 이행한다.
Reqtag	reqtag()	필수 엘리먼트이름의 인식 기능을 이행한다.

3.4 피드의 channel 엘리먼트처리 패키지

RSS 명세서에 따르면 channel 엘리먼트는 선택 엘리먼트와 필수 엘리먼트를 포함한다. 그러므로, channel 엘리먼트를 인식하는 클래스들을 포함하는 패키지 또한 필수, 선택 엘리먼트들의 클래스들을 패키징한 패키지를 포함한다. 그리고 필수 엘리먼트를 인식하는 클래스는 총 3개 이므로, channel 엘리먼트와 필수 엘리먼트의 인식 기능을 하는 클래스들의 관계를 '1:1'으로 설정하였다. 또한 RSS 명세서에서 정의하기를 channel 엘리먼트는 item 엘리먼트를 0-n개 포함할 수 있으므로 channel 엘리먼트와 선택 엘리먼트의 인식 기능을 하는 클래스들의 관계는 '0:n'의 관계로 설정하였다. 앞 절에서 보인 channel 엘리먼트의 필수, 선택 엘리먼트의 인식 기능을 이행하는 클래스들을 종합하여 channel 엘리먼트의 인식 기능을 이행하는 클래스들을 패키징한 결과는 그림 9(부록)와 같다. 그림 9(부록)의 클래스 및 메소드에 대한 설명은 표 4와 같다.

표 4. channel 엘리먼트의 인식 기능을 이행하는 클래스
Table. 4 The classes that function as recognizing a channel element

클래스 명	메소드 명	내 용
Channelelement	channelelement()	channel 엘리먼트의 인식 기능을 이행한다. 필수 엘리먼트를 인식하는 클래스는 channel 엘리먼트를 인식하는 클래스에 3개 모두 필수적으로 포함해야 하고 선택 엘리먼트를 인식하는 클래스를 n개 포함할 수 있다.

필수 엘리먼트와 선택 엘리먼트에 관련된 클래스 및 메소드는 위의 3.2, 3.3절을 참조한다.

3.5 피드의 rss 엘리먼트처리 패키지

RSS 명세서에서 정의하는 rss 엘리먼트는 RSS 버전을 나타내는 구문과 channel 엘리먼트를 포함한다. 이에 의거하면 rss 엘리먼트를 인식하는 클래스들을 패키징한 패키지에는 RSS 버전을 나타내는 구문을 인식하는 클래스와 channel 엘리먼트의 그것을 포함하도록 설계할 수 있다. 이와 함께 앞 절에서 보인 패키징 방법으로 피드의 rss 엘리먼트의 인식 기능을 이행하는 패키지를 설계한 결과는 그림 10(부록)과 같다. 그림 10(부록)의 클래스 및 메소드에 대한 설명은 표 5와 같다. channelelement패키지는 위의 3.4절을 참조한다.

표 5. rss 엘리먼트의 인식 기능을 이행하는 클래스
Table. 5 The classes that function as recognizing a rss element

클래스 명	메소드 명	내 용
Rsselement	rsselement()	rss 엘리먼트의 인식 기능을 이행한다. RSS 버전을 나타내는 구문과 channel 엘리먼트를 인식하는 클래스를 필수적으로 1개씩 포함한다.
Rssverinfo	rssverinfo()	RSS 버전을 나타내는 구문의 인식 기능을 이행한다.
Rssvernum	rssvernum()	RSS 버전 정보의 인식 기능을 이행한다.

3.6 XML 선언문 처리 패키지

피드에 포함되고, XML 명세서에서 정의하는 XML 선언문은 XML 버전과 문서의 인코딩, 그리고 문서의 독립성을 나타낸다. 각 부분들을 인식할 수 있는 클래스들을 설정하고, 이들 전체를 인식하는 클래스를 할당하여 하나의 패키지로 패키징하는 방법으로 설계한 XML 선언문 패키지도는 그림 11과 같고, 그림 11의 클래스 및 메소드의 설명은 표 6과 같다.

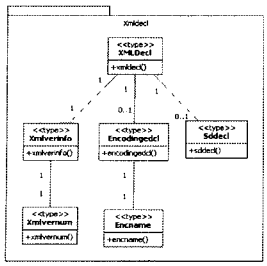

그림 11. XML 선언문의 인식 기능을 이행하는 패키지도

Fig. 11 The package diagram that function as recognizing a XML Declaration context

표 6. XML 선언문의 인식 기능을 이행하는 클래스

Table. 6 The classes that function as recognizing a XML Declaration context

클래스 명	메소드 명	내 용
XMLDecl	xmldecl()	XML 선언문의 인식 기능을 이행한다.
Xmlverinfo	xmlverinfo( )	XML 버전 정보를 나타내는 구문의 인식 기능을 이행한다.
XmlVernum	xmlvernum()	XML 버전 정보의 인식 기능을 이행한다.
Encodingedcl	encodingedcl()	문서의 인코딩 정보를 나타내는 구문의 인식 기능을 이행한다.
Encname	encname()	문서의 인코딩 정보의 인식 기능을 이행한다.
Sddecl	sddecl()	문서의 독립성을 나타내는 구문의 인식 기능을 이행한다.

3.7 주석 및 빈 공간 처리 패키지

XML 표현 형식에 따르는 피드의 주석 및 빈 공간중 일부를 XML 명세서에 의거하여 확장한 결과는 그림 2에 나타낸 피드의 문법에 보였으며 이에 의거하여 주석 및 빈 공간의 인식 기능을 이행하는 클래스들의 패키지를 설계한 결과는 그림 12와 같고, 그림 12의 클래스 및 메소드의 설명은 표 7과 같다.

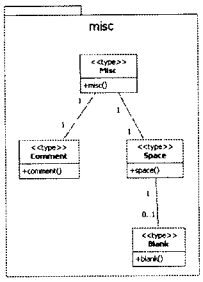

그림 12. 주석 및 빈 공간의 인식 기능을 이행하는 패키지도

Fig. 12 The package diagram that function as recognizing comments and space context

표 7. 주석 및 빈공간의 인식 기능을 이행하는 클래스

Table. 7 The classes that function as recognizing comments and space context

클래스 명	메소드 명	내 용
Misc	misc()	주석 및 빈 공간의 인식 기능을 이행한다.
Comment	comment()	주석문의 인식 기능을 이행한다.
Space	space()	빈 공간의 인식 기능을 이행한다.
Blank	blank()	빈 공간문자의 인식 기능을 이행한다.

3.8 문자 및 특수기호 처리 패키지

XML 표현 형식에 따르는 피드의 문자 및 특수기호들 중 일부를 XML 명세서에 의거하여 확장한 결과는 그림 2에 나타낸 피드의 문법에 보였으며 이에 의거하여 문자 및 특수기호의 인식 기능을 이행하는 클래스들을 하나의 패키지로 패키징한 결과는 그림 13과 같고, 그림 13의 클래스 및 메소드의 설명은 표 8과 같다.

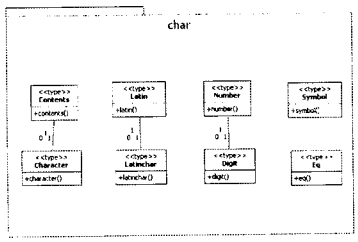

그림 13. 피드의 문자 및 특수기호의 인식 기능을 이행하는 패키지도

Fig. 13 The package diagram that function as recognizing characters and symbols

표 8. 피드의 문자 및 특수기호를 처리하는 클래스

Table. 8 The classes that function as recognizing characters and symbols

클래스 명	메소드 명	내 용
Contents	contents()	문자열의 인식 기능을 이행한다.
Character	character()	문자의 인식 기능을 이행한다.
Latin	latin()	알파벳 문자열의 인식 기능을 이행한다.
Latinchar	latinchar()	알파벳 문자의 인식 기능을 이행한다.
Number	number()	숫자열의 인식 기능을 이행한다.
Digit	digit()	10진수의 숫자의 인식 기능을 이행한다.
Symbol	symbol()	특수기호의 인식 기능을 이행한다.
Eq	eq()	equal 문자의 인식 기능을 이행한다. 피드의 엘리먼트의 애트리뷰트를 인식하는 과정에서 사용되므로 본 패키지에 포함하여 처리하도록 하였다.

3.9 피드 처리 패키지

위에서 나타낸 피드의 구성요소들을 인식하는 클래스들의 패키지를 종합하여 패키징한 결과는 그림 14(부록)와 같이 나타낼 수 있다. 그림 14(부록)에서 피드가 XML 선언문을 단 하나 포함하듯이 피드의 인식 기능을 이행하는 클래스는 XML 선언문의 인식 기능을 이행하는 클래스의 패키지와 1:1 관계를 맺는다. rss 엘리먼트 내에 주석 및 빈 공간이 많이 존재 할 수 있으므로 이들을 인식하는 클래스의 패키지들의 관계는 0..n:1 관계를 맺는다. 또한 피드에서 사용하는 문자를 인식하는 클래스들의 패키지는 타 패키지에 모두 사용되므로 n:1 관계들을 맺는다. 그림 14의 패키지도의 클래스 및 메소드에 대한 설명은 표 9와 같다. 피드 처리 패키지에 포함된 item, channel, rss 엘리먼트와 XML 선언문, 주석 및 빈 공간, 문자 및 특수기호 처리 패키지는 각각 3장의 1-8절을 참고한다.

표 9. 피드의 인식 기능을 이행하는 클래스
Table. 9 The classes that function as recognizing the feed

클래스 명	메소드 명	내 용
Feed	feed()	피드의 인식 기능을 이행한다.

3.10 공통 처리 패키지

피드의 파싱을 위하여 앞에서 나타낸 피드의 구성요소들을 인식하는 클래스들에서 공통적으로 자주 사용하는 모듈을 설계한 패키지는 그림 15와 같다.

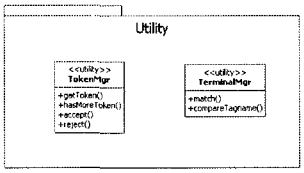

그림 15. 유틸리티 패키지도
Fig. 15 The package diagram function as utility classes

그림 15의 패키지 도의 클래스에서 스테레오 타입으로 나타낸 '<<utility>>'는 클래스가 피드의 데이터에는 영향을 미치지 않고 피드 파서에 유용한 유틸리티의 역할을 위해 존재한다는 의미이다. 위의 패키지도의 클래스 및 메소드에 대한 설명은 표 10과 같다.

표 10. 피드 파서의 유틸리티 클래스
Table. 10 The classes that function as utility classes

클래스 명	메소드 명	내 용
TokenMgr	getToken()	입력 데이터에서 토큰을 반환한다.
	hasMoreToken()	입력 데이터가 더 존재하는지 알아본다.
	accept()	옳게 처리한 데이터를 삭제한다.
	reject()	옳지않게 않은 처리를 하여 데이터를 처리하기 전으로 되돌린다
TerminalMgr	match()	토큰과 터미널을 비교한다.
	compareTagname()	현재 엘리먼트이름을 비교한다.

3.11. 피드 파서 패키지 설계

위에서 나타낸 피드 관련 패키지들을 종합하여 피드 파서를 설계한 결과는 다음과 같이 나타낼 수 있다. 그림 16(부록)에서 피드 인식 패키지와 관계하는 utility 패키지는 피드의 구성요소들을 인식하기 위한 모든 클래스에서 공통적으로 사용한다.

3.12 피드 파서의 예외 처리

이 절에서는 앞 절에서 보인 피드 파서의 설계 결과로 구현한 피드 파서를 이용하여 피드를 파싱하는 과정에서 나타날 수 있는 예외 처리를 위한 클래스들을 보인다. 피드 파서의 예외 처리를 위한 클래스는 피드 파서의 패키지 구조와 유사하게 설정하였다. 그리고 피드의 엘리먼트및 애트리뷰트 파싱 클래스 당 하나의 예외 처리 클래스가 연관되어 있다. 하지만 피드의 주석 및 빈 공간의 예외를 처리하는 클래스는 피드의 전반적인 부분에 관여하므로 피드 예외 처리 클래스의 구조와 분리하여 설계하였다. 설계한 피드 파서의 예외 처리 클래스 구조는 그림 17(부록)과 같다. 그림 17(부록)에서 피드의 예외처리를 위한 클래스는 XML 선언문(XMLDeclException)과 rss 엘리먼트(RssException)의 예외처리를 위한 클래스로 나눌 수 있다. 피드를 구성하는 주석 및 빈 공간의 예외처리 클래스(MiscException)는 rss 엘리먼트의 하부 엘리먼트들의 예외처리에도 포함할 수 있으므로 예외 처리 클래스의 구조에서 분리하여 설계하였다(그림17의 왼쪽 하단).

Ⅳ. 패키지 설계 결과 및 실험

본 장에서는 지금까지 보인 피드 파서의 패키지 설계 결과를 피드 파서를 적용할 수 있는 대표적인 RSS 클라이언트 애플리케이션인 RSS 구독기를 적용하여 피드 파서의 패키지 설계 결과를 실험한다. RSS 구독기는 피드 파서를 탑재하며 수집한 피드를 분석, 출력하는 애플리케이션이다. 실험을 위한 환경은 그림 18과 같다.

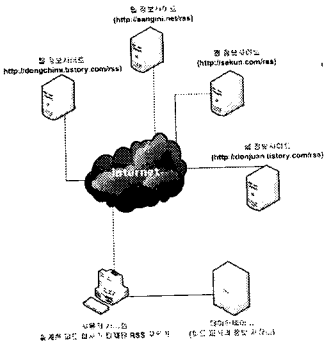

그림 18. 피드 파서의 실험 환경
Fig. 18 The environment of the feed parser

그림 18에 보인 바와 같이 피드 파서를 탑재한 RSS 구독기는 총 4개의 외부 웹 사이트에서 인터넷을 통하여 피드를 수집하도록 하였다. 그리고, 파싱 결과를 저장하는 데이터베이스는 사용자 시스템과 분리하여 구축하였다. 그림 18의 사용자 시스템에서 실행한 RSS 구독기의 화면은 그림 19와 같다.

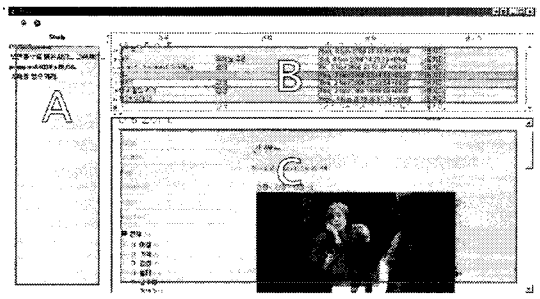

그림 19. 설계한 피드 파서를 내장한 RSS 구독기
Fig 19. The RSS reader load with the designed feed parser

그림 19의 RSS 구독기에서 총 4 개의 3.4절에서 보인 channel 엘리먼트 처리 패키지에서 인식한 정보가 A 부분에 출력되었고, 3.1절에서 보인 item 엘리먼트 처리 패키지에서 인식한 정보가 B와 C 부분에 출력되었다. 즉, A 부분에 출력한 웹 사이트의 제목은 channel 엘리먼트 처리 패키지의 Title 클래스를 사용하여 인식한 결과 정보이며, B 부분의 콘텐츠 제목, 카테고리, 날짜, 글쓴이 등의 정보는 item 엘리먼트 처리 패키지의 Title, Category, Itemetcelement 클래스를 사용하여 인식한 결과 정보이다. 그리고, C 부분은 item 엘리먼트 처리 패키지의 Itemetcelement 클래스에서 인식한 결과인 콘텐츠의 url 정보를 이용하여 접근할 수 있는 콘텐츠의 화면이 출력되었다.

Ⅴ. 결 론

본 논문은 피드를 분석하는 피드 파서를 위한 패키지의 설계를 제시하였다. 웹 상에서 많이 쓰이는 RSS 2.0 버전의 명세서에 의거하여 피드의 문법을 설정하였고 설정한 문법을 검증하기 위하여 피드를 인식하는 오토마타를 설계하였다. 그리고, 설정한 문법에 의거하여 피드를 인식 하는 클래스와 패키지를 설계하였다. 또한 피드 인식 중의 예외사항을 처리 할 수 있는 클래스를 설계하였다. 이렇게 설계한 클래스 및 패키지들은 설정한 피드의 문법 형식 및 EBNF의 넌터미널 이름 등을 그대로 따라 피드 파서 컴포넌트 설계의 이해도를 높였다. 또한 패키지 설계 서 객체지향방법론의 설계언어인 UML을 이용, 설계한 패키지에 의거하여 객체지향언어를 이용한 피드 파서의 구현이 용이 하도록 하였다.

향후에 본 논문에서 설계한 패키지에 의거하여 피드 파서를 구현할 수 있다. 이렇게 구현된 피드 파서가 적용된 RSS 클라이언트 애플리케이션은 컴포넌트 형식으로 설계된 본 논문의 파서의 이점을 그대로 얻어 재사용성의 향상 등의 장점을 얻을 수 있다. 또한 본 논문에서 제시한 파서는 XML 파서의 API(Application Programming Interface)인 DOM(Document Object Model)[19]과 SAX(Simple API for XML)[20]에 비 의존적인 이유로 불필요한 시스템 자원의 이용을 줄일 수 있고, 피드의 인식 기능을 이행하는 중의 예외사항에 대하여 유연하게 대처할 수 있다. 하지만, 제시한 피드 파서는 모든 RSS 버전을 지원하지 않아 RSS 1.0 버전의 피드를 제공하는 웹

사이트에서 사용할 수 없는 제약이 있다. 완벽한 피드 파서로서의 기능을 다 하기 위해서 이에 대한 연구가 필요하다.

참고문헌

[1] Ben Hammersley, *Developing Feeds with RSS and ATOM,* O'Reilly Media, 2005.

[2] The RSS 2.0 specification, http://blogs.law.harvard.edu/tech/rss.

[3] 연모, http://yeonmo.theple.com

[4] 샤프리더, http://www.sharpreader.net

[5] 엑스파이더, http://www.xpyder.co.kr

[6] 한RSS, http://hanrss.com

[7] 블로그라인스 http://www.bloglines.com

[8] rojo, http://rojo.com

[9] 이동규, 김윤호, "RSS 구독 시스템을 위한 패키지의 설계," *한국해양정보통신학회,* Vol.10 No.12, 2006.

[10] MagpieRSS : RSS for PHP, http://magpierss. sourceforge.net

[11] Universal Feed Parser, http://feedparser.org

[12] RSS Channel Presentation and Searching, http://rssxpress.ukoln.ac.uk/lite/include

[13] Sethi, Ullman Aho, Compilers: Principles, Techniques and Tools, Addison-Wesley, 1986.

[14] Martin Fowler, *UML Distilled : A Brief Guide to the Standard Object Modeling Language,* 3rd Edition, Addison-Wesley, 2003.

[15] Larman Craig, *Applying UML and Patterns,* 3rd Edition, Prentice Hall, 2005.

[16] Extensible Markup Language (XML) 1.0 (Fourth Edition), http://www.w3.org/TR/2006/REC-xml-20060816/

[17] The RSS 1.0 specification, http://web.resource.org/rss/1.0/spec

[18] James Rumbaugh, Ivar Jacobson, Grady Booch, *Unified Modeling Language Reference Manual,* 2nd Edition, Addison-Wesley, 2004.

[19] W3C Document Object Model, http://www.w3.org/DOM/

[20] SAX(Simple API for XML), http://www.saxproject.org/

저자소개

이 동 규(Dong-kyu Lee)

2005. 2 안동대학교 컴퓨터공학과 공학사
2007. 2 안동대학교 컴퓨터공학과 공학석사
2007.3-(주)넥스트리소프트 연구원
※관심분야 : 객체지향 분석/설계/프로그래밍

김 윤 호(Yun-Ho Kim)

1983. 2. 경북대학교 전자공학과 공학사
1993. 2. 경북대학교 대학원 컴퓨터공학과 공학석사
1997. 2. 경북대학교 대학원 컴퓨터공학과 공학박사
1997. 8. - 현재 안동대학교 전자정보산업학부 부교수
※관심분야 : 인터넷 컴퓨팅, 객체지향 분석/설계/프로그래밍, 분산객체시스템, 병렬처리

부록

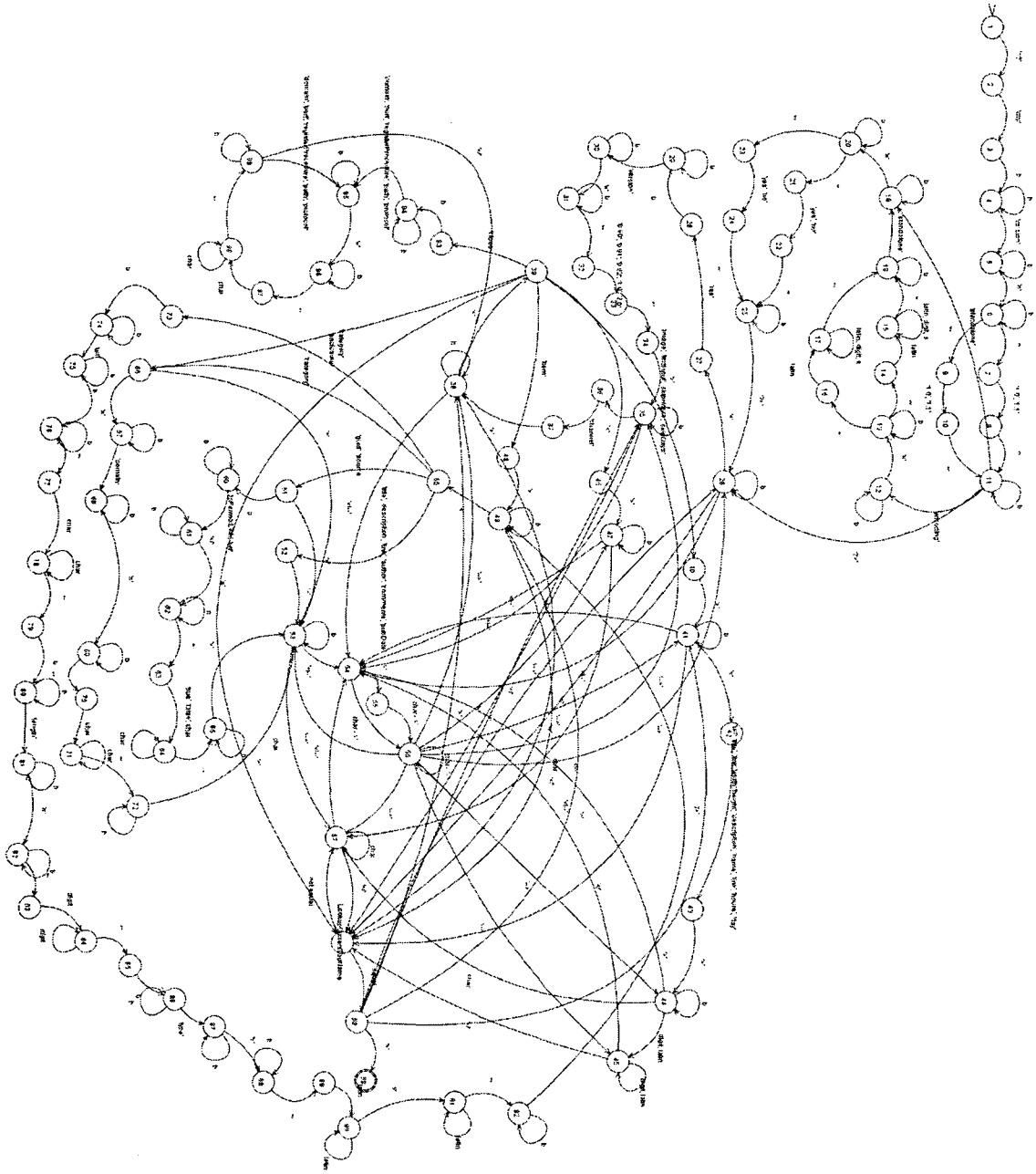

그림 5. 피드를 인식하는 오토마타
Fig. 5 The automata that recognized a Feed

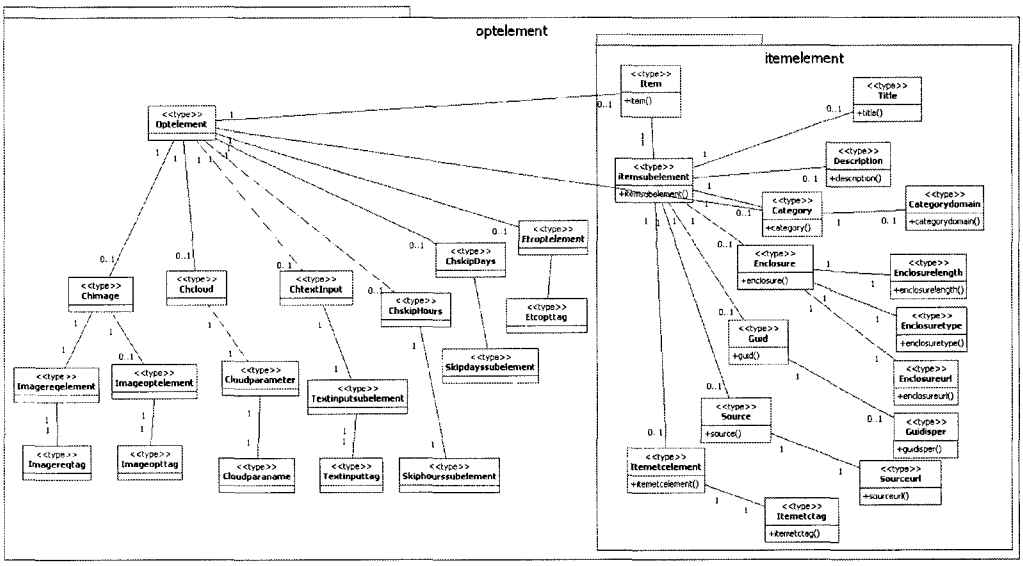

그림 7. 선택 엘리먼트의 인식 기능을 이행하는 패키지도
Fig. 7 The package diagram that function as recognizing option elements

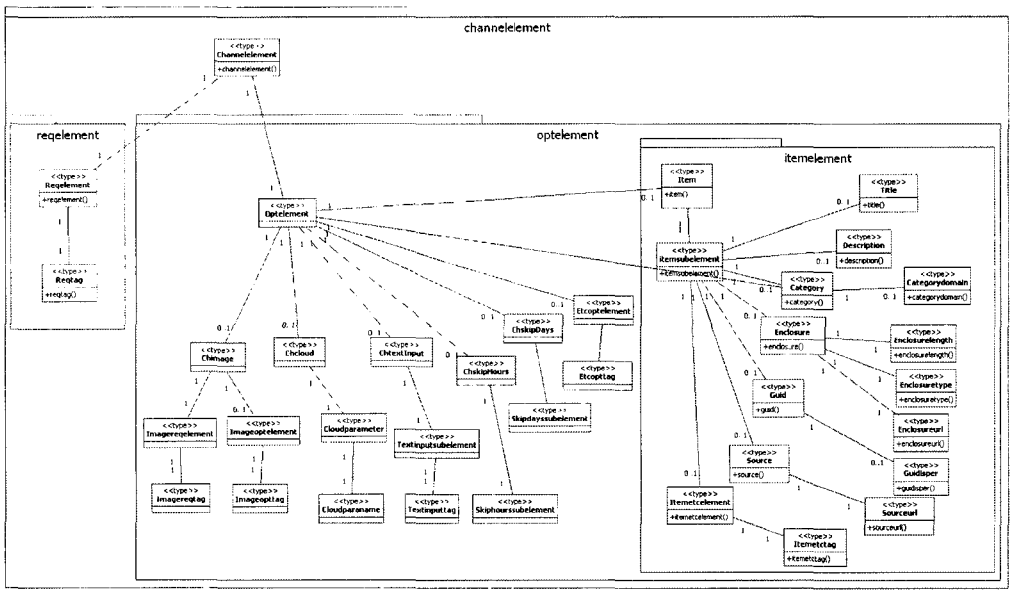

그림 9. channel 엘리먼트의 인식 기능을 이행하는 패키지도
Fig. 9 The package diagram that function as recognizing a channel element

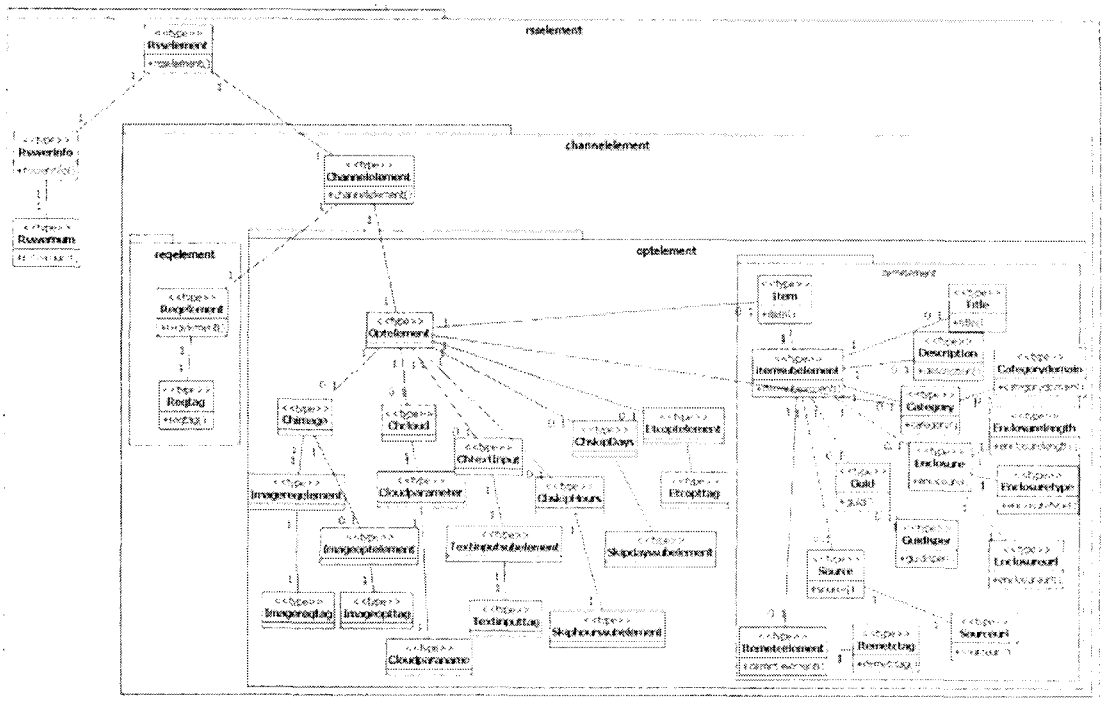

그림 10. rss 엘리먼트의 인식 기능을 이행하는 패키지도
Fig. 10 The package diagram that function as recognizing a rss element

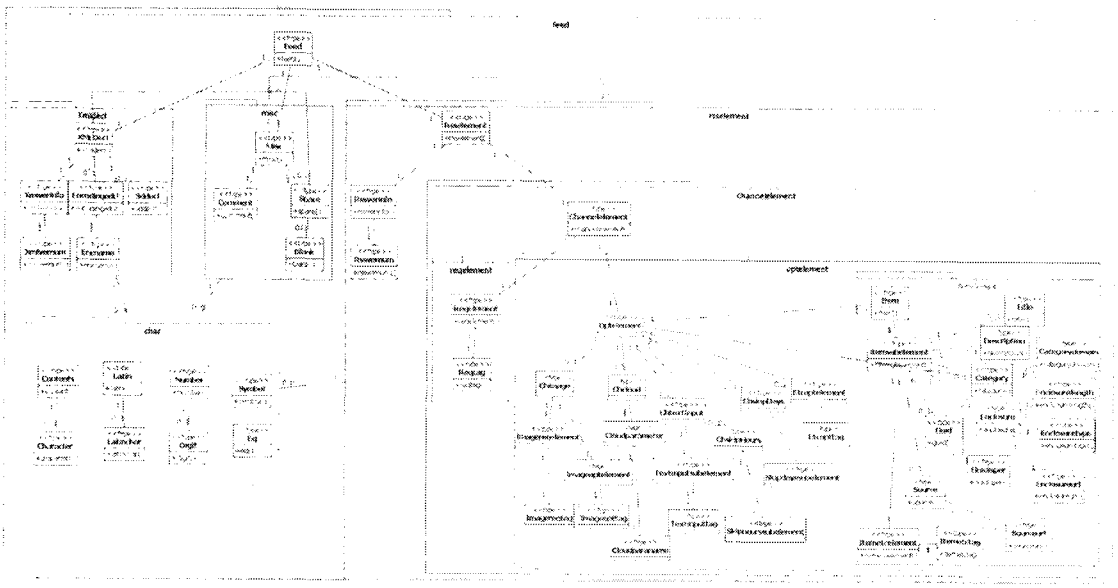

그림 14. 피드의 인식 기능을 이행하는 패키지도
Fig. 14 The package diagram that function as recognizing the feed

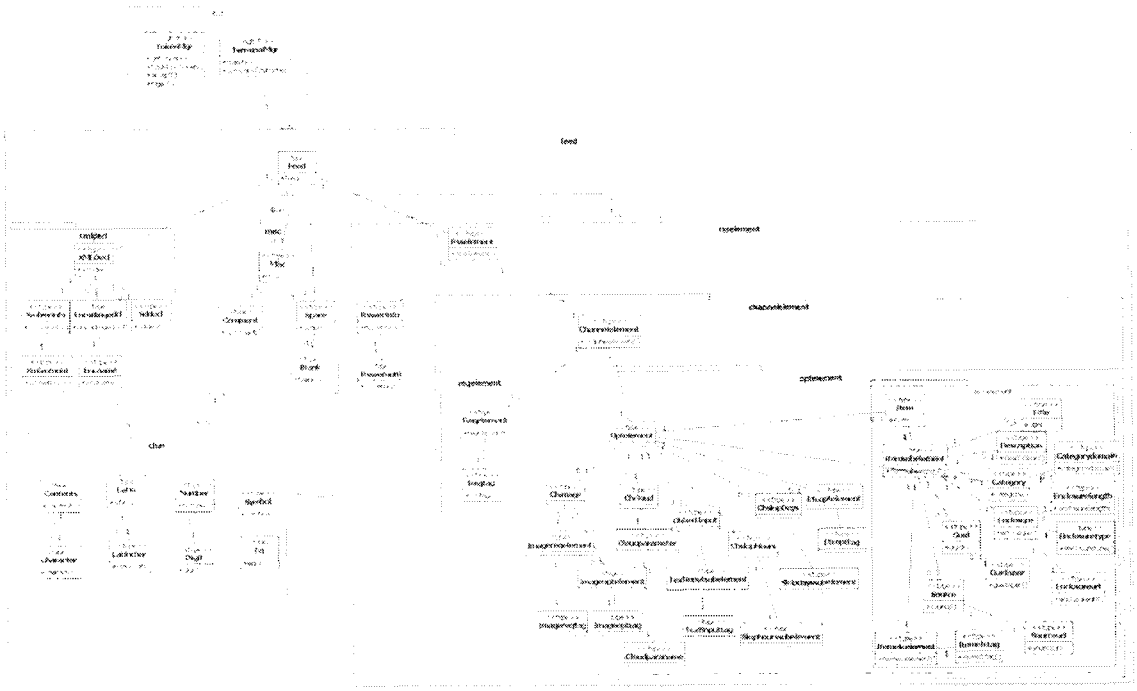

그림 16. 피드 파서 패키지도
Fig. 16 The package diagram of the feed parser

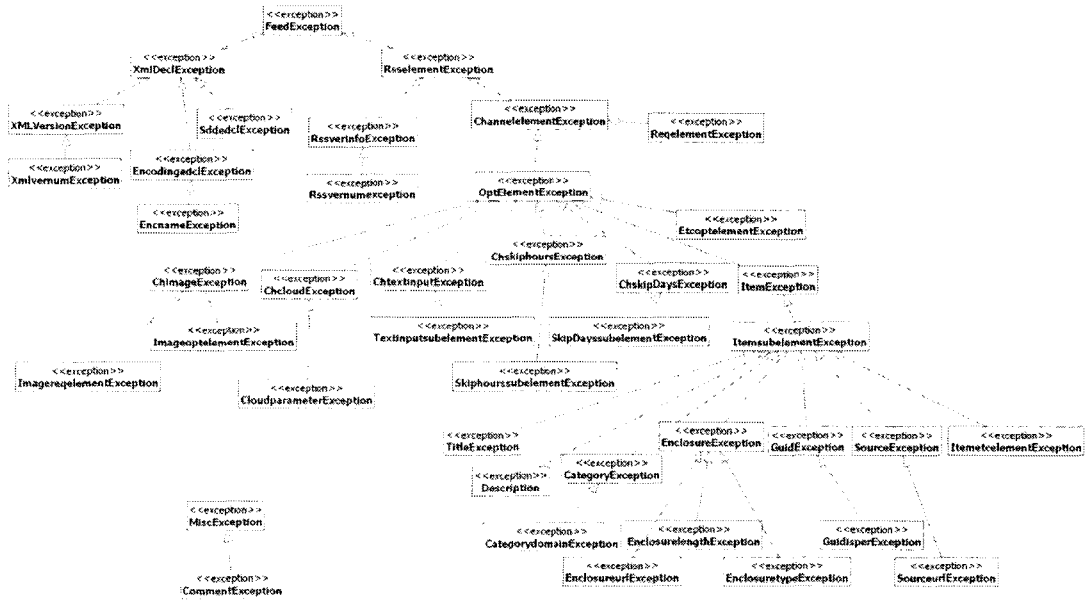

그림 17. RSS 파서의 예외처리 클래스 구조도
Fig. 17 The Structure diagram that exception classes of the RSS parser