유비쿼터스의 상황인식 어플리케이션을 위한 XML 데이터 바인딩 기술에 대한 연구

문지숙, 윤회진, 최병주
이화여자대학교 컴퓨터학과
milkqu@hanmail.net, {hjyoon, bjchoi}@ewha.ac.kr

A Study on Data Binding of XML for Context Aware Application in Ubiquitous

Jisuk Moon, Hoijin Yoon, Byoungju Choi
Ewha Womans University, Computer Science

요 약

유비쿼터스 컴퓨팅의 상황정보들은, 서비스기반아키텍쳐와 같은 환경에서 XML기술을 기반으로 구성되며 공유된다. 따라서 상황인식 어플리케이션은 상황정보를 표현하는 XML문서를 대상으로 하는 행위들의 구현으로 볼 수 있다. 이때 상황인식 어플리케이션 구현의 시작을 XML 스키마 기반의 소스코드 템플리트를 이용한다면, 보다 정확하게 상황정보를 표현하는 어플리케이션을 구현할 수 있는 기반을 마련할 수 있다. 본 논문에서는 XML을 이용하는 방법 가운데, 기존의 파서를 통한 접근 방법이 아닌 바인딩 기술을 이용한 접근 방법을 통해, 상황정보를 가지고 있는 XML 파일의 스키마 구조를 표현하는 어플리케이션 템플리트 생성한다. 이를 위해 본 논문은, 첫째, 바인딩 기술이 기존의 파서를 통한 접근법보다 유비쿼터스 상황인식에 유리한 이유를 분석하였으며, 둘째, 여러 가지 바인딩 기술들을 항목별로 분석하여 상황인식 어플리케이션 템플리트를 만드는데 가장 적합한 한가지를 선정한다. 마지막으로 상황정보를 표현하는 XML을 이용하여 어플리케이션을 구현하는 방법의 이해를 돕기 위해 액티브 배지 시스템의 한 부분인 " Call Forwarding" 에 본 논문에서 제안하는 방법을 적용하는 예제를 보인다.

1. 서 론

상황정보(Context)란 한 개체의 물리적이고 사회적인 상황을 말하며 시간, 신원, 장소, 개체 등의 정보이다. 이러한 상황정보를 개체들 간에 주고 받을 수 있으며 상황에 따라 수행해야 할 서비스를 결정하게 된다[1]. 개체간의 통신을 가능하게 하는 것은 바로 인터넷 기술과 XML(Extensible Markup Language)이다. 충분한 인프라를 갖추고 있는 인터넷 환경 위에서 XML을 이용하여 표현한 데이터를 교환하고 활용할 수 있다. 또한 XML특유의 유연성을 바탕으로 모든 사물에 대한 상황정보를 효과적으로 표현할 수 있고, 형식에 맞추어 정보를 교환함으로써 개체간의 통신을 원활하게 할 수 있다.

XML을 이용하여 개체의 특성에 맞게 작성된 상황정보는 개체의 움직임이나 주변환경에 따라 변화한다. 이때, 이러한 기본 정보를 가공하여 서비스를 할 수 있는 능력을 갖는 어플리케이션을 제작하는 것이 중요한 과제이다. XML이 가지고 있는 이식성과 서비스의 재사용을 강조하면서, 차세대 인터넷에서 주목하고 있는 SOA(Service-Oriented Architecture)[2] 개념에 부합하는 어플리케이션을 개발할 수 있는 방법 또한 XML 형태의 정보들을 웹에서 교환하는 기술을 기본으로 한다.

. XML 문서를 처리하기 위해서 사용하는 API는 크게 두 가지로 나눌 수 있다. SAX(Simple API for XML)와 DOM(Document Object Model)과 같은 파서 방식의 전통적인 API와 데이터 바인딩에 입각하여 개발이 진행 되고 있는 Quick, Zeus, JAXB, JiBX, Castor 등과 같은 API 이다[3]. 여러 가지 API는 서로 다른 특성을 가지므로 그것을 유비쿼터스 컴퓨팅의 상황정보 인식을 위한 어플리케이션을 개발하는 것에 적용하기 위해서는 각각의 특징을 파악하여 알맞은 API를 선택하는 것이 중요하다.

본 논문에서는 XML을 기반으로 한 상황정보를 처리하기 위해 어플리케이션화 하는 과정에서 데이터 바인딩을 이용하여 접근하는 것이 1) 기존의 SAX나 DOM 파서를 이용한 방법 보다 어떤 면에서 유비쿼터스 컴퓨팅 환경을 구축하는데에 기여할 수 있는지에 대해서 논의할 것이며, 2) 여러 가지 데이터 바인딩 프래임웍들을 비교할 것이다. 3) 마지막으로 JAXB를 기반으로 XML 문서 형식의 상황정보를 받아들이고 간단한 서비스를 구현할 수 있는 어플리케이션을 개발하는 과정을 제안한다.

2. XML 바인딩 기술 분석

2.1 전통적 파서 방식과 데이터 바인딩의 비교

전통적인 파서 방식으로는 SAX와 DOM가 있다. SAX는 SAX 파서를 생성하고 콜백(call back)함수와 컨텐트 핸들러를 이용해야 한다. SAX가 가지고 있는 단점은 문서 속에 담겨 있는 데이터를 순차적으로 접근하는 방식이므로 메모리의 낭비가 심하며 또한 메모리 상에서의 데이터의 처리가 불가능하다는 것이다. 이는 SAX를 이용하여 XML 문서내의 데이터의 업데이트를 할 수 없음을 의미한다. DOM은 XML의 구조와 흡사한 다큐먼트 트리를 구성 함으로서 데이터에 접근할 수 있도록 한다. 다큐먼트 트리는 다큐먼트 오브젝트 아래 루트 노드와 자식 노드를 갖는 구조이며 트리 내에 XML의 구조와 데이터를 모두 가지고 있으므로 XML의 구조를 이해하고 있어야 파서를 통해 데이터를 찾아 갈 수 있다[4].

데이터 바인딩도 DOM의 다큐먼트 트리와 같은 트리 구조인 컨텍스트 오브젝트를 가지고 있다. 그러나 다큐먼트 트리와 달리 컨텍스트 오브젝트는 오로지 데이터만을 가지고 있다는 점이 다르다. 그러므로 다큐먼트 트리에서 유지해야 하는 오브젝트의 수보다 컨텍스트 오브젝트의 수가 작으므로 메모리 측면에서 이점이 생긴다. 또한 파서를 통해 구조를 따라가면서 데이터에 접근하는 것이 아니기 때문에 데이터에 비 순차적으로 직접 접근하고 조작할 수 있다[5]. 따라서, XML의 구조적인 면을 다루는 일을 중시하는 어플리케이션을 개발하는 경우를 제외하고 주로 데이터를 읽어서 조작할 경우에 데이터 바인딩을 이용할 경우 다음과 같은 이점이 있음을 알 수 있다.

1) XML의 구조를 알 필요가 없다.
2) 빠르고 직접적으로 데이터에 접근할 수 있다.
3) 메모리 측면에서 절약된다.

2.2 데이터 바인딩 관련 API 연구

데이터 바인딩에 관련한 여러 가지 프레임 워크가 존재하고, 각각의 특징도 다양하다. 본 논문에서는 다양한 데이터 바인딩 기술들[6,7,8,9]을 <표 1>과 같이 분석하여, 현재 상황인식 어플리케이션 개발에 활용하기 가장 적합한 기술을 선정하였다. <표 1>을 살펴보면 데이터 바인딩이 크게 매핑에 의한 바인딩과 코드 생성에 의한 바인딩으로 나누어지는 것을 알 수 있다. 매핑에 의한 방법은 클래스를 생성함에 있어서의 유연성에 가장 큰 장점이 있지만 그 유연성을 활용하기 위해서는 그만큼 많은 수작업이 요구되어 개발 시간이 오래 걸리며, XML 문서와 데이터가 확실히 연결되어있는지 확신을 할 수 없는 단점이 있다. 또한 강력한 기능에 주안점을 둔 Quick과 같은 경우 여러 스키마 파일을 관리해야 하는 단점을 가지고 있다.

표 1 데이터 바인딩 관련 API 연구

	Quick	Zeus	Castor	JAXB	JBind
제약	DTD	DTD	Schema	Schema	Schema
접근방법	mapped binding	mapped binding	mapped binding/ code generation	code generation	code generation
	유연성			XML 구조와 코드간의 tight coupling	
	다양한 입출력 포맷 제공하나 수작업이 요구됨			코드를 자동으로 생성 런타임 유효성 검사	
	바인딩 기술의 대세가 code generation 쪽으로 흐르는 추세이다.		인터페이스 없이 구현 클래스 생성	전체 구조를 모두 유지해야 하는 단점이 있다.	
특징	XMLdata + behavior	proto-typing	JavaBean-like structure		XMLdata+ behavior
	복잡하고 강력한 기능	간단하고 심플한 기능	안정적인 구현	인터페이스기반	완벽한 스키마를 제공
	여러 단계의 스키마 파일이 존재하여 복잡함	인터페이스와 팩토리 클래스를 이용하지 않고 프로토타입을 활용	인터페이스 없이 구현 클래스를 생성하므로 팩토리 클래스를 사용하지 않음	인터페이스와 구현 클래스로 이루어짐 팩토리 클래스 사용함	DOM 을 이용하여 바인딩 코드를 저장한 뒤 접근함
언마샬	약 60ms	약 65ms	약 115ms	약 55ms	약 650ms
마샬	약100ms	약 55ms	약 85ms	약 60ms	약 30ms
메모리	약450kb	약 550kb	약 300kb	약 400kb	약5300kb
		String형만 존재하여 많은 메모리 공간 필요	String을 int형으로 변환하여 메모리 절약	실제 데이터만 저장하지만 String형으로 저장하고 조작정보를 추가하므로 메모리 공간 필요	

코드 생성에 의한 방법은 자동적이고 빠르게 클래스들을 생성해 주고 그것이 XML 문서의 구조와 일치한다는 확신을 가질 수 있다. 그러나, 매핑에 의한 방법에 비해 구조와 데이터간에 결합도가 높아지게 된다. 그러므로 문서의 구조가 바뀐다면, 그 구조에 따라 다시 클래스를 생성해야 하는 수고가 따른다. 그럼에도 불구하고, 코드 생성에 의한 방법은 점차 신뢰성과 신속성을 바탕으로 매핑에 의한 방법보다 선호되고 있는 상황이다.

또한 SUN사에서 코드 생성 방식을 채택하고 있는 JAXB(Java Architecture for XML Binding)를 자바를 위한 바인딩 기술의 표준으로 함에 따라 앞으로 JAXB에 대한 연구와 활용이 활발히 진행될 것으로 보인다. JAXB는 플러그-인 호환성, 이식성의 장점을 바탕으로 자바 빈 스타일의 getX(), setX() 메소드를 통해 데이터에 보다 쉽게 접근하게 하는 인터페이스를 제공한다. 또한 JAXB를 이용하여 XML 문서를 컨텍스트 오브젝트화하는 언마샬링 과정과 다시 컨텍스트 오브젝트에서 XML 문서로 변환하는 마샬링 과정에 걸리는 시간을 2.2KB ~ 5.8KB의 비교적 작은 크기의 문서를 이용하여 측정해 보았을 때, 다른 API에 비해 오래 걸리지 않음을 알 수 있다. 메모리 사용량은 JAXB가 문자열을 정수형으로 변환하여 저장하는 방식의 Castor보다는 많음을 알 수 있다. 그러나 JAXB는 문자열 타입으로 저장하고 인터페이스를 통해 접근 할 수 있는 구현 클래스를 자동적으로 생성함에 따른 편리성을 갖고 여전히 기존의 DOM에 의해 사용되는 메모리보다는 작음을 알 수 있다.

3. JAXB 적용 예제

데이터 바인딩 관련 API중 JAXB를 이용하는 것은 표준이 갖는 이점과 코드 생성에 의한 방법이 갖는 이점을 동시에 갖고 있기 때문에 XML 문서의 구조적 변형 없이 데이터 만을 다루어 어플리케이션화 하기에 적당한 API이다. 비록 JAXB가 XML 구조와 코드 사이에 강력한 결합으로 인해 스키마가 바뀔 때마다 다시 스키마를 바인딩 해 주어야 하는 단점이 있으나 상황정보의 특성상 형식이 처음 설계한 것에서 크게 변화하지 않고 내용이 변하는 것이므로 영향을 주지 않는다.

JAXB를 이용하여 XML 문서를 처리하는 과정에 대해 사람의 위치 정보를 액티브 배지[10] 통해서 파악하여 그 장소에 전화를 연결해 주는 간단한 콜 포워딩(Call Forwarding)을 통해 알아보겠다.

먼저 JAXB는 Java Web Services Developer Pack 1.6을 다운로드 하면 얻을 수 있고, 상황정보를 가지고 있는 XML 문서와, 스키마가 준비 되어야 한다. <그림 1>은 콜 포워딩의 상황정보를 표현하는 XML의 스키마이다. XML 문서는 실제 데이터를 가지고 있고, XML 스키마는 각 요소들의 타입, 구조를 계층적으로 정의하고 있으므로 데이터 바인딩에서 자동적인 코드의 생성을 가능하게 한다.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="locationInformation" type="locationInformationType"/>
  <xsd:complexType name="locationInformationType">
    <xsd:sequence>
      <xsd:element name="lists" type="Lists"/>
      <xsd:element name="timeInfo" type="TimeInfo"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Lists">
    <xsd:sequence>
      <xsd:element name="list" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element name="id" type="xsd:long"/>
            <xsd:element name="location" type="xsd:string"/>
            <xsd:element name="probability" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="TimeInfo">
    <xsd:sequence>
      <xsd:element name="currentDate" type="xsd:string"/>
      <xsd:element name="currentTime" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

그림 1 콜 포워딩의 상황정보를 표현하는 XML의 스키마

JAXB는 스키마를 바인딩 하여 인터페이스와 그 인터페이스의 구현으로 이루어진 클래스들을 아래와 같은 명령어를 통해 자동으로 생성한다. - p는 생성될 클래스의 패키지를 정의해 준다.

```
%JWSDP_HOME%jaxb₩bin₩xjc -p mypackage.mylocation location.xsd
```

<그림 2>는 스키마 바인딩을 통해 생성된 인터페이스와 impl 폴더 내에 생성된 인터페이스를 구현한 클래스를 보여준다. 또한 클래스명을 살펴보면 스키마에서 정의한 루트 요소와 콤플렉스 타입(Complex Type) 요소의 이름과 같음을 알 수 있으므로 내부적으로 어떻게 클래스를 생성하는지 알 수 있다.

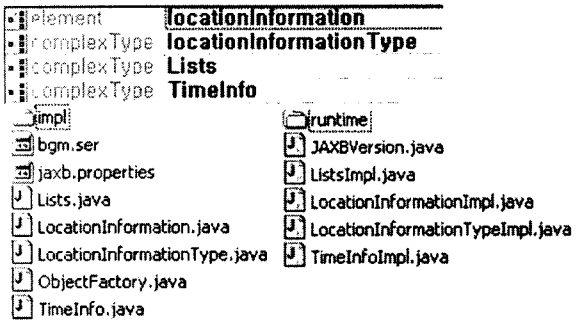

그림 2 생성된 클래스와 스키마의 구조

ListImpl.java 내부의 ListTypeImpl을 <그림 3>과 함께 살펴보면 XML 문서에 내용인 이름과 위치 정보 등을 직접 접근할 수 있는 getId(), setId()와 같은 형태의 메소드를 가지고 있는 것을 알 수 있다. 자동적으로 생성된 메소드를 이용할 수 있기 때문에 XML의 자세한 구조를 몰라도 쉽고 빠르게 데이터를 이용할 수 있게 되는 것이다.

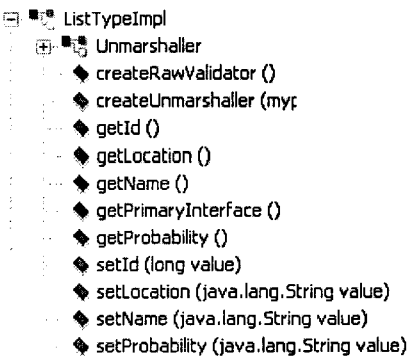

그림 3 ListImpl.java에 정의된 메소드

스키마 바인딩을 바탕으로 얻은 클래스들을 이용하여 실제 데이터를 가지고 있는 XML문서의 데이터를 이용하기 위해서 먼저 XML 문서를 컨텐트 오브젝트화 하여야 한다. 컨텐트 오브젝트를 생성한 뒤, 언마샬러(Unmarshaller)를 이용하여 XML 문서를 언마샬하면 되는 간단한 과정이다[11].

1) 먼저 JAXBContext를 사용자 패키지 내에 생성하고,

```
JAXBContext jc=JAXBContext.newInstance( "mypackage" );
```

2) 언마샬러를 생성하여 XML 문서를 언마샬 한다.

```
Unmarshaller u = jc.createUnmarshaller();
LocationInformation locinfo =(LocationInformation)u.
        unmarshal( new FileInputStream( "location.xml" ) );
```

3) Id, Name, Location으로 지정된 상황정보에 접근하기 위해서 다음과 같이 스키마 구조에 따라 list 객체를 생성한다.

```
Lists lists = locinfo.getLists();
List listTypeList = lists.getList();
```

4) list.getId(), list.getName(), list.getLocation()와 같은 접근 함수를 사용하여 데이터를 읽거나 업데이트를 할 수 있다.

4. 결론

유비쿼터스 컴퓨팅의 기본인 상황정보 인식에서 가장 필요한 것은 상황정보를 어떤 방식으로 표현하고 전송할 수 있느냐는 것이다. 인터넷 환경에서의 새로운 표준으로 자리잡고 있는 XML을 바탕으로 하는 상황정보가 어플리케이션의 하위에서 구조적으로 형식과 내용을 분리됨에 따라 데이터 바인딩 기술을 이용하여 효과적으로 XML 문서 내의 데이터에 접근할 수 있도록 해 주었다. 그러므로 기존에 널리 사용되었던 SAX나 DOM과 같은 파서를 이용한 방법을 이용하여 XML 문서를 다루고자 했을 때 요구되었던 XML 문서의 구조에 대한 이해와, 개발을 위해 투자되었던 시간과 노력이 무의미 하게 되었다. 또한 상황정보 인식을 위한 어플리케이션이 XML 문서의 형식과 구조를 변경시키는 데에 주안점을 두기 보다 XML 문서에서 데이터를 읽고 현재 수행해야 할 서비스가 무엇인지 결정하여 데이터를 이용한 서비스를 사용하는 성격을 가지기 때문에 무엇보다 데이터 바인딩을 이용함으로서 생기는 편의성과 효율성이 높을 것이다.

5. 참고 문헌

[1] Salber, Dey and Abowd,, Proceedings of CHI' 99, pp.434-441 The Context Toolkit, 1999

[2] Hao He, XML.COM, What is Service-Oriented Architecture? http://www.xml.com/pub/a/ws/2003/09/30/soa.html Sep, 2003

[3] McLaughlin,Brett, O' Reilly, Java and XML, 2/E, Jan, 2001

[4] Scott Fordin, Sun, Java Architecture for XML Binding http://java.sun.com/xml/jaxb/about.html Oct, 2004

[5] Ed Ort and Bhakti Mehta, Sun, Java Architecture for XMLBinding(JAXB) http://java.sun.com/developer/technicalArticles/WebServices/jaxb/index.html Mar, 2003

[6] Dennis M. Sosnoski, IBM developerWorks, Data binding Part 1: Code generation approaches—JAXB and more http://www-128.ibm.com/developerworks/library/x-databdopt/index.html Jan, 2003

[7] Dennis M. Sosnoski, IBM developerWorks, Data Binding with Castor http://www-128.ibm.com/developerworks/xml/library/x-bindcastor/index.html Apr, 2002

[8] Dennis M. Sosnoski, IBM developerWorks, Data binding Part 2: Performance http://www-128.ibm.com/developerworks/library/x-databdopt2/index.html Jan, 2003

[9] Mette Hedin, XML.COM, Comparing Java Data Binding Tools http://www.xml.com/pub/a/2003/09/ 03/binding.html Sep, 2003

[10] Roy Want, Andy Hopper, Ceronica Falcao and, Jonathan Gibbons, The Active Badge Location System, ACM Transactions on Information Systems, Vol. 10, No. 1, Pages 91-102, Jan, 1992

[11] Scott Fordin, Sun, Java Architecture for XML Binding http://java.sun.com/xml/jaxb/about.html Oct, 2004