인라인 XML바인딩 시스템의 구현

유가연, 이은정
경기대학교 정보과학부
e-mail: {despoin, ejlee}@kyonggi.ac.kr

Implementation of inline XML Binding System

Ga-Yeon Yoo, Eun-Jung Lee
Dept of Computer Science, Kyonggi University

요 약

XML데이터를 어플리케이션에서 직접 사용 할 수 있도록 XML데이터 바인딩이 널리 사용되고 있다. 그러나 기존의 바인딩 시스템에서는 자식을 가지는 요소마다 모두 클래스가 생성되어 시스템이 커진다는 문제점을 가지고 있다. 본 논문에서는 구조 표현에 필요한 클래스만 생성하기위해 인라이닝 기법을 바인딩에 적용하여 최소한의 클래스만 생성 할 수 있도록 시스템을 설계하고 구현한다.

1. 서 론

XML 데이터가 여러 시스템 간의 데이터 호환과 상호 운용성을 높이기 위해 많이 사용되고 있다. 일반적으로 XML 데이터를 이용하는 프로그램은 DOM API를 사용하여 데이터를 읽고 DOM트리를 방문하면서 필요한 데이터를 추출한다. 그러나 이러한 DOM은 기술의 어려움으로 사용이 어렵고 비효율적인 코드를 만들게 하는 문제점을 가지고 있다. 한편 XML데이터를 바로 프로그램에서 읽어 들여 사용하기 위한 방법으로 XML데이터 바인딩이 널리 사용되고 있다.[2,3]

XML 데이터 바인딩이란 XML 문서의 요소들에 대응하는 프로그램 객체를 생성하는 것으로 XML태그의 값을 객체의 필드 값에 대응하도록 하는 클래스를 생성하게 된다. XML 바인딩 툴을 이용하면 바인딩 클래스는 XML 문서의 타입 정보로부터 자동으로 생성된다. 그 결과 프로그래머는 XML 데이터 바인딩 객체들을 이용하여 DOM 트리나 파서를 이용한 XML 데이터의 입출력 처리를 수행할 수 있게 된다.[5,6]

가장 대표적인 바인딩 시스템으로는 JAXB[7]를 들 수 있다. 이 시스템은 스키마 컴파일러에서 XML 문서 구조에 따라 대응하는 자바 클래스의 소스를 생성해준다. XML 바인딩 툴로서 JAXB는 강력한 API와 유효성 검사 기능, 그리고 모든 XML 스키마의 사양을 지원하는 완성도를 가지고 있다. 그러나 강력한 기능만큼 JAXB는 일반 사용자들이 사용하기에 몇 가지 어려움이 있다. 우선 JAXB 시스템이 간단한 구조를 가지는 XML 문서라 하더라도 요소마다 따로 클래스가 생성되어 XML 스키마 컴파일러에 의해 생성되는 클래스들이 매우 많고 사용되는 API 구조도 복잡하다.

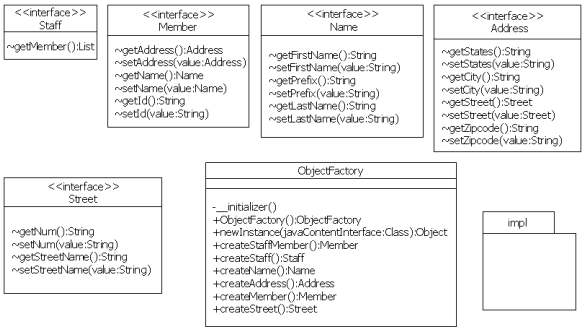
본 논문에서는 새로운 바인딩 방법을 제시하고 바인딩 클래스의 생성 및 마샬 및 언마샬을 수행할 수 있는 바인딩 시스템을 설계하고 구현하였다. 제안된 시스템은 인라이닝 기법을 적용하여 구조 표현에 필요한 클래스만 생성한다. 인라인은 DB에서 최소한의 테이블을 생성하기 위해서 사용되는 기법으로서 바인딩에 적용하여 클래스의 수를 줄이는데 효율적으로 사용될 수 있을 것으로 예상된다.[1]

2. 인라인에 의한 바인딩 방법

2.1 동기

```
<!ELEMENT staff (member)*>
<!ELEMENT member (id, name, address)>
<!ELEMENT name (prefix, lastname, firstname)>
<!ELEMENT address (street, city, states, zipcode)>
<!ELEMENT street (num, streetname)>
```

(그림 1) 바인딩 클래스를 작성할 DTD



(그림 2) JAXB 생성 클래스 구조

위의 그림 2는 JAXB에서 생성되는 클래스의 구조의 예를 보여준다. name이나 address와 같은 요소와 대응하는 클래스가 생성되어 member클래스의 필드의 타입으로 포함된 것을 알 수 있다. 이와 같이 우선 JAXB의 스키마 컴파일러는 문서 정의에서 나타나는 모든 요소에 대해 기본적으로 클래스를 생성하게 된다. 그러나 문서의 구조를 표현하는데 생략 가능한 요소들까지 모두 클래스로 표현되기 때문에 실제 생성되는 클래스의 개수가 무척 많아지게 된다. 예를 들어 그림 1의 DTD에서 구조를 나타내기 위해 필요한 staff, member이외에도 name, address, street와 같이 단순히 #PCDATA타입의 자식요소만을 가지는 모든 요소에 대해 클래스가 생성된다. 이런 것들은 DTD를 설계할 당시에 바인딩 방식을 고려하거나 사용자 정의 바인딩 규칙(Binding Declaration)에 의해 어느 정도 해결 할 수는 있으나 기본적인 JAXB의 바인딩 클래스 생성 방식에 의해 사용자가 구조를 변경할 수 없기 때문에 불필요한 경우까지 모두 클래스로 생성되는 경우가 많다.

2.2 인라이닝 클래스 추출방법

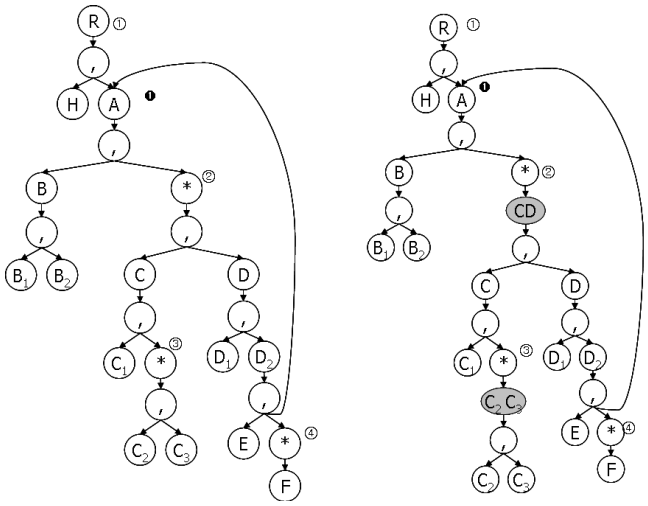
DTD로 주어진 XML 문서의 정의로부터 인라이닝에 의해 생성될 클래스를 추출해내는 방법을 살펴본다. 여기서 사용하는 방법은 Jayavel 등이 제안한 관계형 데이터 추출 방법[4]에서 사용된 인라이닝 방법과 유사하다. 클래스로 생성할 요소를 구분하기 위해 DTD(방향) 그래프를 사용한다.

DTD 그래프에서 노드의 종류는 다음과 같이 나누어 볼 수 있다.

(ⅰ) 내부 노드 : 나가는 에지를 가짐
  A. 루트 요소 노드
  B. 내부 기호 노드 - (*), (+), (?), (,), (|)
  C. 내부 요소 노드

(ⅱ) 터미널 노드 : 값의 타입을 가짐
  A. 터미널 요소 노드
  B. 속성 노드

내부 노드 중에서 자기 자신으로 다시 돌아오는 경로가 있는 경우를 재귀 노드라고 한다. 재귀 노드는 내부 노드 중에서 루트나 기호 노드가 아닌 내부 요소 노드만이 가능하다. 또한 내부 기호 노드 중에서 (*), (+), (?) 노드를 반복부 노드라고 한다. 아래 그림에서 반복부 노드는 ②③④번의 세 개이고 루트 노드는 ①번, 재귀노드는 ❶이다. 이렇게 정의되는 DTD그래프로부터 클래스로 생성할 대상을 추출하기 위해 반복 노드 그래프를 다음과 같이 생성한다. 먼저 반복부 노드에 대해 자식 노드가 요소 노드가 아닌 경우 가상으로 요소 노드를 자식을 추가한다. 아래 그림의 예에서 ①과 ④의 노드는 자식이 요소노드이고, ②와 ③의 노드는 자식이 기호 노드이다. 그러므로 어두운 색으로 표시된 (CD)노드와 (C2C3)노드를 가상 자식 요소 노드로 생성하게 된다.



(그림 3) DTD 그래프

주어진 DTD그래프에 G에 대해 RT(G) ={v|v는 루트 노드 및 반복부 노드의 자식 요소 노드 또는 재귀 노드} 라고 하고 RT(G)를 이용하여 반복부 그래프를 생성한다. 반복부 그래프에서 DTD그래프 G의 정보 중 바인딩 클래스 코드 생성을 위해 필요한 정보는 모두 유지되어야 함을 보장하기 위해서 각 반복부 그래프 노드 v에 대해 직접자손터미널노드 집합 term_nodes(v)가 정의된다. 또한 반복부 그래프 노드 v에 속하는 각 직접 자손 터미널 노드 w는 언마샬링/마샬링을 위해서 인라이닝된 클래스로부터 원래 트리의 경로와 순서가 그대로 회복될 수 있도록 하기 위해 자신이 속하는 RT(G)로부터의 경로를 rel_path$_v$(w)로 표시한다. 또한 각 직접 자손 터미널 노드에 대해 타입정보 type(w)와 자식 순서 정보가 index$_v$(w)에 저장된다.

다음은 반복부 그래프 GR을 생성하는 재귀 알고리즘과 위의 그림 3의 DTD그래프 G로부터 생성된 반복부 그래프를 나타낸다.

[반복부 그래프 생성 재귀 알고리즘]

현재 반복부 노드를 V_R, 현재 상대경로를 rel_path라 하고 재귀적으로 전달한다.

DTD 그래프 G의 모든 노드 g에 대해서

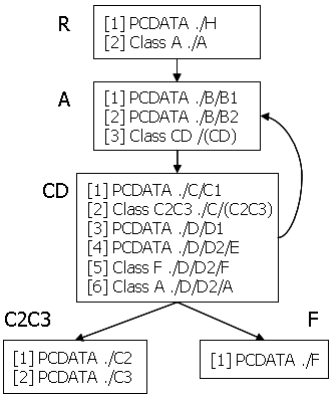
ⅰ. V_R = 루트노드

ⅱ. g가 RT(G)에 속하는 경우

  A. g가 재귀 노드인 경우

    1. g에 대해서 GR에 노드 W_R이 존재 한다면, V_R = parent(W_R)

2. 없다면 B와 같이 반복부 그래프를 생성
B. 재귀 노드가 아닌 경우
1. 반복부 그래프 노드 W_R을 생성
2. rel_path를 초기화
3. V_R = parent(W_R)
4. V_R = W_R
5. g = g의 자식 노드
ii. g가 터미널 노드인 경우
A. rel_path에 g의 레이블을 추가
B. term_nodes(V_R)에 g를 추가
iv. g가 기호 노드이거나 요소노드이면서 반복부 노드의 자식이 아닐 경우
A. rel_path에 g의 레이블을 추가
B. g = g의 자식노드



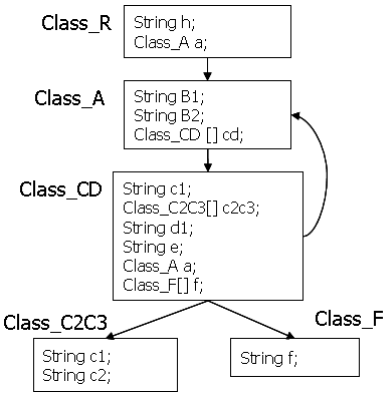
(그림 4) 반복부 그래프

자식 순서 정보는 직접 자손 터미널 노드 이외에 RT(G)에 속한 노드로 나가는 에지도 순서에 포함시켜 인덱스를 계산한다. 예를 들어 위 그림 4에서 relective_path$A$(B2) = ./B/B2이고 index$A$(B2) = 2가 되고 index$A$(CD) = 3이 된다.

2.3 클래스 및 필드 생성 방법

생성된 반복부 그래프를 가지고 다음과 같이 클래스 및 필드를 생성하게 된다.

[클래스 및 필드 생성 알고리즘]
반복노드 그래프의 모든 노드 v에 대해
i. 클래스 헤더를 생성
ii. 직접 자손 터미널 노드에 대해
A. 대응하는 멤버 변수를 추가.
B. set/get 함수를 추가.
iii. v의 자식 노드 w에 대해
A. 재귀 노드이면 해당 클래스 객체를 멤버 변수로 추가.
B. 반복부 노드이면 클래스 객체 리스트를 멤버 변수로 추가하고, add/remove 함수를 추가
C. 생략 노드이면 클래스 객체를 멤버 변수로 추가하고, insert/delete 함수를 추가



(그림 5) 클래스 구조 및 필드

2.4 마샬링 메소드 생성 방법

바인딩 클래스의 주요 메소드로는 XML 데이터를 읽어 객체를 생성하는 언마샬과 메모리상의 객체를 XML 파일로 내보내는 마샬이 있다. 언마샬은 정해진 순서에 따라 XML트리를 순회 하면서 필요한 데이터를 해당 필드에 저장하는 기능이므로 비교적 간단하지만 마샬은 인라이닝에 의해 변경된 부분을 회복하기 위하여 반복부 그래프에 저장된 순서 정보 및 경로 정보로부터 유효한 XML문서를 생성해 내야한다.

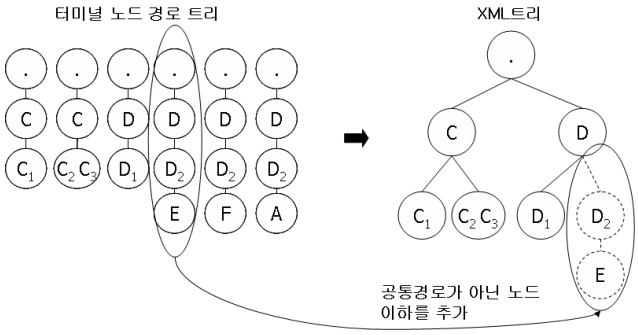


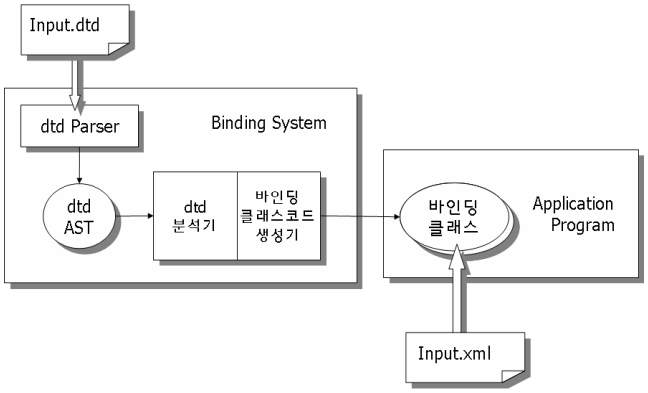
그림 6 마샬 메소드 알고리즘

위의 그림은 그림 4의 반복부 그래프 중 CD노드에 대하여 생성된 마샬 메소드를 나타낸 것이다. 반복부 그래프의 각 노드에 대하여 직접자손터미널노드가 가지는 경로를 각각의 트리로 만들고, 먼저 나온 요소와의 공통 경로가 아닌 노드 이하를 일치하는 위치에 추가해 준다.

3. 구 현

3.1 전체 시스템의 구현

본 시스템은 Java로 개발하였으며 플랫폼 뿐 아니라 응용프로그램에 독립적이다. DTD로부터 문서 정보를 얻어 내기 위해 Antlr라는 파서 생성기를 이용하여 DTD파서를 생성하였다. 또한 바인딩 시스템으로부터 생성된 바인딩 클래스는 DOM을 사용하여 xml문서를 언마샬하여 객체를 생성한다.

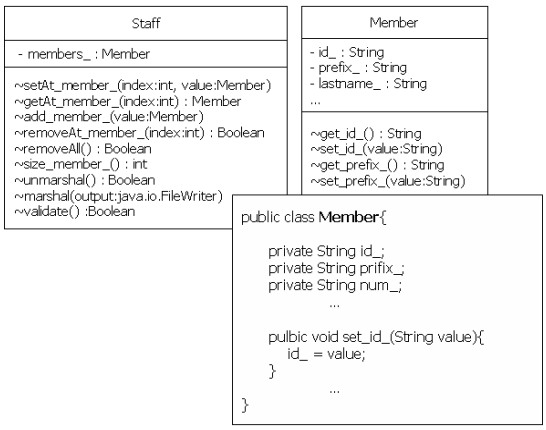


(그림 7) 전체 바인딩 시스템의 구조

3.2 생성된 바인딩 클래스의 예

아래의 그림 8은 그림 1에 정의된 DTD를 본 논문에서 구현한 시스템에 의해 생성된 바인딩 클래스를 나타낸다. Member클래스는 하위 요소들의 터미널 요소들을 모두 인라이닝 하여 필드로 가진다. 이렇게 생성된 클래스들은 위의 그림 2에서 보았던 JAXB에서 생성된 클래스들보다 그 수가 줄었다는 것을 알 수 있다.

비록 생성된 클래스들의 수는 줄었지만, 아래 그림 9에서의 어플리케이션 코드의 사용과 같이 기존의 바인딩 시스템과 동일한 수준의 바인딩 기능을 수행하는데 별 무리가 없을 것으로 보인다.



(그림 8) 생성된 바인딩 클래스

```
public class StaffApp {

public static void main(String args[]) throws Exception{

      Staff staff = new Staff();
      staff.unmarshal("staff.xml");

      Member member = staff.getAt_member_(2);

      System.out.println ("name : " + member.get_prefix_()
        + member.get_lastName_() + member.get_firstName_());

      member.set_id_ (12345);

      staff.marshal("staff-2.xml");
   }
}
```

(그림 9) 바인딩 클래스 사용 예

4. 결 론

본 논문에서는 XML 바인딩 시스템의 구현을 목표로 최소한의 클래스만 생성할 수 있도록 인라이닝 기법을 사용하여 시스템을 설계하고 구현하였다. 지금까지의 바인딩 시스템은 XML구조에 해당하는 모든 요소에 대해 클래스를 생성해 주는 것과는 달리, 본 논문의 시스템은 구조 표현에 필요한 클래스만을 생성해 주기 때문에 어플리케이션의 향상된 성능을 기대할 수 있을 것이다.

참고문헌

[1] 조정길, 인라이닝에 기반한 XML 스키마의 관계형 스키마 변환 기법, 한국정보처리학회 논문지, 2004 . 10.

[2] Fabio Simeoni, et. al, "Language Bindings to XML", IEEE Internet Computing 7(1), Jan., Feb. 2003, p. 19-27.

[3] Robert van Engelen, et. al, "Developing Web Services for C and C++", IEEE Internet Computing 7(2), Mar., Apr. 2003, p. 53-61.

[4] Jayavel Shanmugasundaram, et. al, "Relational Databases for Querying XML Documents: Limitations and Opportunities", VLDB 1999.

[5] XML Data Binding Resources. http://www.rpbourret.com/xml/XMLDataBinding.htm.

[6] XML and Java technologies: Data binding. http://www-106.ibm.com/developerworks/xml/library/x-databdopt/.

[7] Java Architecture for XML Binding (JAXB). http://developer.java.sun.com/developer/technicalArticles/WebServices/jaxb/.