XML을 이용한 문제은행 시스템 설계

강 동 헌* 김 종 우
제주안덕초등학교* 제주교육대학교 컴퓨터교육과
ehd7gjs7@hanmir.com* woo@jejue.ac.kr

A Design of Item pool System using XML

Kang Dong-Heon* Jong-Woo Kim
Jeju Andeok Elementary School*
Dept. of Computer Education, Jeju National University of Education

요 약

본 논문에서는 XML을 이용하여 문제은행 시스템(Item pool System)을 설계하였다. XML은 단순하면서도 구조적인 내용을 표현하는데 용이하며, 확장성이 뛰어난 마크업 언어이다. 문제 은행은 문항의 형태, 내용, 난이도, 등을 포함한 문항의 특성과 관련된 정보들을 체계적으로 제시해야하므로 데이터베이스에 저장된 내용들을 체계적으로 표현할 수 있어야한다. 따라서, 본 논문은 내용과 관련된 태그를 직접 만들 수 있고, 여러 개의 XML 문서들을 하나의 큰 문서로 병합할 수 있으며, 어떠한 종류의 응용프로그램과도 통합될 수 있는 범용적 데이터베이스라고도 할 수 있는 XML의 특성을 이용하여 좀더 확장성이 좋아지게 하였다.

1. 서론

초기에 웹은 단순히 정보를 제공하기 위하여, 예를 들면 문자나 그림을 표현하기 위한 도구쯤으로 쓰였다. 인터넷상의 대부분의 정보는 HTML 문서로 구성되어있으며, HTML은 단지 문서의 재현을 위한 정보를 나타내는 하나의 정의된 DTD (Document Type Definition)를 사용하기 때문에 각 문서의 엘리먼트를 의미 있는 정보로 표현하는 기능이 부족하다.

이에 W3C(World Wide Web Consortium)에서는 차세대 웹 문서의 표준으로 XML(Extensible Markup Language)을 1998년에 지정하였다[1]. XML은 확장성 마크업 언어로서 이름그대로 HTML은 같은 고정된 형식이 아니라 확장이 가능한 언어이며 문서의 내용과 관련된 태그를 직접 정의 할 수 있으며 그 태그를 다른 사람들이 사용할 수 있다. XML은 본질적으로 다른 언어를 기술하기 위한 언어, 즉 메타언어이다[2]. 또한 구조적 데이터를 표현할 수 있으며 사용자가 정의한 DTD를 만족하는 트리 구조를 가지고 있어서 XML은 구조적 문서를 표현하는데 유용하다.

문제 은행은 문항의 형태, 내용, 난이도, 등을 포함한 문항의 특성과 관련된 정보들을 체계적으로 제시해야하므로 데이터베이스에 저장된 내용들을 체계적으로 표현할 수 있는 XML을 사용하여 확장성이 좋아지게 한다.

본 논문에서는 문제은행 시스템을 문제 출제, 문제 해결, Reporting의 세 부분으로 나누어 설계하고자 한다.

2. 관련기술

문제은행 시스템을 개발하기 위해서 **XML**은 문제를 정의하고 만들어 내고, 문제를 해결한 결과를 불러오며 Reporting 하고, **XSL**은 문제를 사용자 인터페이스로 변환하고, 문제를 해결한 결과를 리포팅 가능한 포맷으로 변환한다. 문제를 클라이언트의 웹 브라우저로 전달하고, 문제를 해결한 결과를 수집하고, 문제 및 결과를 저장하기 위해 데이터베이스에 접근하기 위해서 ASP를 사용했으며, 데이터 베이스 접근 및 레코드셋(RecordSet)에서 XML로의 변환을 캡슐화 하고, XML을 표시하기 위해 ASP 페이지로 전달하는 역할은 **서버측 Visual Basic ActiveX(DLL)**에서 처리하게 되고, 데이터 베이스는 **Access 데이터베이스**를 이용하여 데이터를 저장한다.

3. 문제은행 제작

문제은행 구축을 논의함에 있어서 두 가지 짚고 넘어가야 할 것이 있다. 하나는 문제출제를 위한 표준 XML 형식에 관한 것이고, 다른 하나는 XML 기반 문제를 출력하기 위한 웹 기반의 툴에 관한 것이다. 어떤 툴을 구축하기 전에 우선 툴을 써서 무엇을 생성해 낼 것인지 결정해야 한다.

3.1. 엔터티(Entity) 정의

3.1.1. 문제

문제는 두 가지 종류의 정보로 분류될 수 있는데, 하나는 과목, 주제와 같은 문제 자체에 관련된 사항들이고, 다른 하나는 텍스트, 타입, 가능한 값 같은 문제를 구성하는 질문에 관련된 사항들이다. 문제에는 여러 개의 질문 있기 때문에 질문이 별도의 고유한 엔터티(Entity)로 쪼개는 것이 자연스럽다.

다음 <표 1>과 <표 2>는 이 두 가지와 관련된 데이터를 보여준다.

문제 정보	
Id	문제에 대한 식별자, 중복되지 않아야 함
Subject	출제 과목
Unit	단원
Introduction	문제출제에 대한 대략적인 안내

<표 1> 문제 정보에 관한 데이터

문항	
Id	문항에 대한 유일한 식별자
problemid	문항에 속한 문제의 아이디
sequence	질문의 순서
caption	질문 대한 텍스트
type	문항의 형태
attributes	질문을 위한 데이터 수준의 속성(최대길이, 최소 길이, 데이터 타입 등등)
options	다중선택 문항을 위한 가능한 값들의 리스트
format	문항의 형식에 맞게 하기 위한 XSL 변환
creationdate	문항이 만들어진 날짜

<표 2> 문항에 관한 데이터

다음 단계는 데이터 연동을 하기 위해 정보에 대한 리스트를 XML 스키마로 변환하는 일이다. 스키마는 <그림-1>과 같다.

```
<Schema xmlns="urn:schemas-microsoft-com:xml-data">
<AttributeType name='problumid' required='yes' />
<ElementType name='subject' content='textOnly' />
<ElementType name='unit' content='textOnly' />
<ElementType name='intro' content='textOnly' />
<ElementType name='problum' content='eltOnly' order='many'>
  <attribute type='problumid' />
  <element type='subject' minOccurs='1' maxOccers='1' />
  <element type='unit' minOccurs='1' maxOccers='1' />
  <element type='intro' minOccurs='1' maxOccers='1' />
</ElementType>
</Schema>
```

<그림 1> 데이터 연동을 위한 XML 스키마

3.1.2. 문항 정보 구성

간단히 보면 문항은 캡션과 답변의 두 가지로 구성된다. 문항은 두 가지 종류로 나눌 수 있는데, 하나는 어떤 답변도 허용하는 자유형식(free-form)이고, 다른 하나는 기존에 정의된 답변 중에서 선택하고 제한하는 다중선택(multiple-choice)형이다[3]. 첫 번째 타입인 자유형식형은 사용자가 문자열을 얻어오기만 하면 되고 이것은 textbox와 textarea 같은 HTML 요소 중의 하나로 구성할 수 있다. 다중 선택형의 경우 드롭다운 콤보박스나 리스트 박스 혹은 라디오 버튼 묶음을 사용할 수 있다

문항을 HTML 표로 나타낼 것이기 때문에 각각의 질문은 <tr></tr> 한 쌍과 <td></td> 태그 두 쌍을 갖게 될 것이다. <표 3>에서 개략적으로 살펴볼 수 있다.

<tr></tr>에 대해 조정 가능한 속성들		
속성이름	가능한 값	설명
Align	left, center, right	셀에 대한 가로 정렬
Valign	bottom, middle, top	셀에 대한 세로 정렬
Bgcolor	미리 정한 컬러나 RGB 값	행에 대한 배경색
<td></td>에 대해 조정 가능한 속성들		
속성이름	가능한 값	설명
align	left, center, right	셀에 대한 가로 정렬
valign	bottom, middle, top	셀에 대한 세로 정렬
width	value or percent	셀이 차지하는 너비 혹은 비율
Bgcolor	미리 정한 컬러 값	셀에 대한 배경색

<표 3> HTML 표로 나타낼 때 조정 가능한 속성들

3.1.3. 자유형식형 문항

HTML로 나타낼 때 한 줄이냐 여러 줄이냐에 따라 두 가지 종류의 자유형식 문항이 있을 수 있다. 그러나 이것은 데이터와 상관없이 표현과 관련한 문항이므로 변환하는 과정에서 주의를 기울이면 된다. 단일 라인의 경우를 예를 들면,

```
<q id='1' type='1' seq='1'>
  <caption> 출제할 과목을 입력하시오.</caption>
</q>
```

변환을 통해 다음처럼 된다는 것을 알 수 있다.

```
<tr>
  <td> 출제할 과목을 입력하시오.</td>
  <td><input value='' type='text' name='_q1'/></td>
```

다음 XSL 코드는 단일 라인 입력을 나타낸다.

```
<tr>
  <td><xsl:value-df select='qcaption /></td>
  <td>
    <input value=''>
      <xsl:attribute name='type'>text</xsl:attribute>
      <xsl:attribute name='subject'>_q<xsl:value-of select='q/@id' />
      </xsl:attribute>
    </input>
  </td>
</tr>
```

복수라인에서는 XSL에서 텍스트 박스에 대해 조금만 변화를 주면 된다.

```
<tr>
  <td><xsl:value-df select='qcaption /></td>
  <td>
    <textarea>
      <xsl:attribute name='subject'>_q<xsl:value-of select='q/@id' />
      </xsl:attribute>
      <![CDATA[]]>
    </textarea>
  </td>
</tr>
```

3.1.4. 자유 형식형 문항을 위한 속성들

<atts>태그를 사용하여 문항마다 다른 데이터 속성을 정의한다. <표 4>는 자유 형식형 문항에서 최적화 할 수 있는 속성이다. 이런 것들을 다루려면 XSL 변환을 약간 수정할 필요가 있다.

이름	가능한 값	효과	단일라인/복수라인
maxsize	number	응답으로 받아들여 질 수 있는 문자의 최대 개수를 설정	다일라인
default	string	기본적으로 선택되어 있는 값으로 HTML에서는 이 값이 초기에 보여 진다.	복수라인

<표 4> 자유 형식 문항을 위한 속성

3.1.5. 다중 선택형 문항

다중 선택형 문항을 표시하는 데는 다양한 방법이 있기 때문에 자유 형식형 질문보다 가능한 변환 리스트가 훨씬 많을 수밖에 없다. 단일 선택과 복수 선택의 두 가지 스타일이 있는데 단일 선택 타입은 리스트박스, 드롭다운 콤보박스, 라디오 버튼 집합의 세 가지 표현 방법으로 표현 할 수 있고, 복수선택 타입의 경우 리스트 박스나 체크박스 집합 둘 중 하나로 표현할 수 있다.

3.1.6. 다중 선택형 문항을 위한 속성

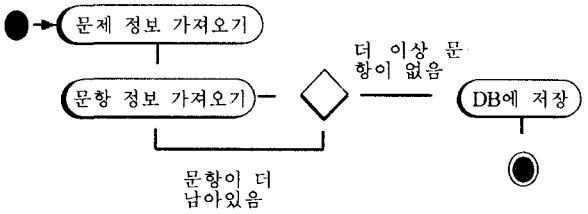
<atts> 태그로 질문별로 데이터에 대한 속성을 정의 할 수 있다. <표 5>는 다중 선택형 문항에서 쓸 수 있는 속성에 대한 리스트이다.

이름	가능한 값	효과	단일라인/복수라인
default	string	기본적으로 선택되어 있는 값. HTML에서는 이 값이 초기에 선택되어 있다.	모두 해당

<표 5> 다중 선택형 문항을 위한 속성

3.2. 문제 저작 도구 설계

다음에 있는 다이어 그램은 시스템 절차도이다. 우선 질문을 계속 투핑하면서 각각에 대한 정보를 수집함으로써 문항에 관계된 정보를 수집한다. 그런 다음 문항에 대한 정보를 수집한 후 문제를 데이터베이스에 저장한다.



<그림 2> 문제 저작도구 설계를 위한 다이어그램

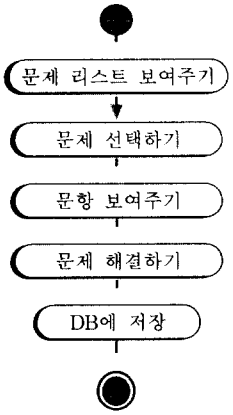
이 다이어그램을 보면 ASP페이지가 기본적으로 세 가지 상태에 따라 서로 다른 기능을 수행하는 간단한 상태 머신임을 알 수 있다. 세 가지 상태란 **1. 문제 정보를 얻는 단계, 2질문정보를**

얻는 단계, 3. 데이터 베이스에 저장하는 단계이다. 이를 구현하기 위해서는 현재 어떤 상태인지 추적하는 방법과 상태 사이를 이동할 수 있는 방법이 필요하다. 전자는 URL에 파라미터를 붙임으로써 해결할 수 있는데, 그 파라미터 이름을 step이라 하겠고, 세 가지 다른 값을 취할 수 있다. 2번 상태일 경우 어떤 질문에 대해 정보를 얻고 있는지 결정할 방법이 필요하다. 이를 위한 가장 쉬운 방법은 또 다른 파라미터 curq를 URL에 덧붙이는 것인데, 이 파라미터는 정수값을 취한다. 이 값을 갖게 되면(즉, 문항의 끝에 가게 되면) 3단계로 이동하면 된다. 후자는 상태별로 하나의 ASP에 어떻게 처리할 것인가 인데, 이것은 select case문을 통해 해결할 수 있다.

3.3. 문제 해결하기 설계

이번 시스템의 핵심은 문제를 디스플레이하고 응답을 처리하는 것이다. 기본적인 처리과정은 <그림 3>과 같이 요약될 수 있다. 문제은행 시스템은 보유하고 있는 문제 리스트를 보여 주고, 그 중 하나를 사용자가 선택하면 그 내용을 보여주고, 그에 대한 응답을 처리하는 세 가지 기본적인 함수로 나눌 수 있다. 앞의 문항 작성처럼 ASP페이지를 상태 머신으로 구현하고, URL에 1이나 2 혹은 3을 값으로 갖는 step이라는 파라미터를 URL에 덧붙여 각 상태를 구별한다.

데이터베이스에서 문제를 받아올 때는 XML로 리턴해주고 리턴된 값을 다음과 같은 XSL 변환하여 리스트 박스로 구현한다.



<그림 3> 문제 해결하기의 처리과정

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
  <select name='_problemid' size='10'>
  <xsl:for-each select="subject/unit">
    <option>
      <xsl:attribute name='value'><xsl:value-of select="@id" />
      </xsl:attribute>
      <xsl:value-of select="subject" />
    </option>
  </xsl:for-each>
  </select>
</xsl:template>
</xsl:stylesheet>
```

따라서 ASP페이지에서는 문제리스트를 얻기 위해 단순히 컴포넌트를 호출하면 되고, 호출 결과를 XSL 변환하여 결과물을 출력하면 된다.

3.4. 보고서 작성(Reporting)

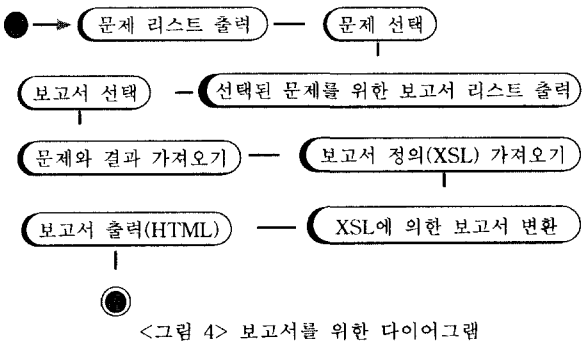
XML과 XSL의 장점을 최대한 이용한 부분이 바로 문제를 해결한 결과를 표시하는 부분이다. 해결한 문제 과목 및 문항수, 난이도, 정답률 등이 출력된다. <표 6>은 보고서를 위해 저장할 데이터를 나타낸 표이다.

Report		
Id	Number	보고서 식별자, 중복 안됨
problemid	Number	이보고서가 어떤 문제를 위한 것인가
Name	text	보고서 이름
description	Memo	문제를 해결한 결과에 대한 의견(매우 잘함, 잘함, 보통, 노력바람. 등)
transformation	memo	보고서를 위한 XSL 변환
Reactionresponse	text	사용자의 응답
corno	Number	정답 반응 횟수
creationdate	date/time	보고서가 만들어진 날짜

<표 6> 보고서 저장을 위한 데이터

다른 페이지들처럼 이 페이지도 step이라는 변수를 URL에 덧붙이는 방식으로 상태에 따라 다른 일을 하도록 작성된다. 문제 리스트를 보여 준 다음 선택된 질문에 대한 보고서 리스트를 보여주며, 최종적으로 선택된 보고서를 보여주면 된다. 이것 역시 ASP 구문 중 select case문으로 사용한다.

<그림 4>는 보고서를 위한 절차도 이다.



<그림 4> 보고서를 위한 다이어그램

이 보고서 시스템은 새로운 다양한 보고서 타입을 지원하기 위해서는 XSL 변환을 많이 추가하면 된다는 점에서 꽤 유연하다고 볼 수 있다.

4. 결론 및 향후 과제

XML이 등장하면서 웹 환경이 급속도로 변하고 있다. 일정한 형식을 갖춘 문서의 경우 표현하는 능력이 뛰어난 XML이 많이 사용되고 있으며 실제적인 표준이 되고 있다.

본 연구에서는 문제은행 시스템은 문제를 클라이언트의 웹브라우저로 전달하고, 문제를 해결한 결과를 수집하고, 문제 및 결과를 저장하기 위해 데이터베이스에 접근하기 위한 방법으로 ASP를 활용하도록 하고 있으며, 문제 출제, 제시, 보고서는 다양한 양식으로 확장이 가능한 XML의 형식을 활용하여 설계하였다. 향후 연구 방향으로는 본 설계를 이용한 문제 은행 시스템을 구현하고, 사용자 위주의 문제은행으로 더 나가기 위해 문제리스트 방식이 아니라 원하는 문제를 검색할 수 있는 검색 모듈을 추가하여 설계되고 구현하는 방법에 대한 연구가 필요하다.

5. 참고 문헌

[1] Extensible Markup Language(XML) 1.0, "http://www.w3.org/TR/REC-xml".
[2] Frank Boumphrey외 공저, 류광 역,"Professional XML APPLICATIONS", 정보문화사, 1999.
[3] 박광운, 웹을 이용한 학습자 중심 문제은행 저작시스템의 설계 및 구현, 한국교원대학교 석사학위졸업논문, 1999.
[4] 신문섭 저, "CONTACT ASP3", 도서출판 대림, 2000.
[5] James Britt 외 공저, 이태용 역, "Professional Visual Basic 6 XML", 정보문화사, 2000.
[6] Extensible Stylesheet Language(XSL)Version 1.0, "http://www.w3.org/TR/WD-xsl".