MVC 기반 웹 애플리케이션 빌더 설계

김귀정*
*건양대학교 의공학과
e-mail:gjkim@konyang.ac.kr

Design of MVC-based Web Application Builder

Gui-Jug Kim*
*Dept. of Bio Medical Engineering, KonYang University

요 약

웹 애플리케이션에서 요구되고 있는 필요 기술이 증가함에 따라 이들 기술 사이에 상호작용이 많아지고 복잡도가 증가하여 웹 애플리케이션을 개발하고 효율적으로 사용하기가 점차로 더 어려워지고 있다. 이에 본 연구는 웹 애플리케이션의 기본적인 요구사항을 재검증하여 복잡도를 감소시키고, 폼 기반 웹 애플리케이션 모델을 생성할 수 있는 MVC 기반 웹 애플리케이션 빌더를 정의하고자 한다. 이 툴은 클라이언트와 서버 사이의 입·출력 데이터 변환에 필요한 코드를 자동으로 생성해 주고, 클라이언트와 서버 간의 데이터 교환을 감소시켜 복잡도를 줄일 수 있도록 해준다.

1. 서론

웹 애플리케이션은 클라이언트/서버 소프트웨어로 설명될 수 있으며, 클라이언트는 웹 브라우저와 컴퓨팅 장치에서 실행될 수 있는 다양한 표준화들을 의미한다. 특히, HTML이 웹 표준화의 기본이 된다. HTML 1.0은 기본적으로 문서 포맷과 하이퍼링크를 제공했으며, HTML 2.0은 사용자 입력을 받아서 서버에 전달하는 submit 포맷을 정의함으로써 한층 더 역동적이고 상호작용성이 강한 웹을 만들 수 있었다[1].

다이나믹한 웹을 만들기 위한 이러한 노력은 웹의 효용성과 웹 사용자의 경험을 향상시켰다. 하지만, 이는 웹 애플리케이션을 프로그래밍 하는데 있어서 복잡도를 증가시키는 결과를 초래하게 되었다. 복잡도는 다음 2 가지 주요 원인에 의해서 발생된다. 첫째, 다이나믹 웹 페이지가 비록 하나의 소스 코드로 이루어져 있다 하더라도, 이들은 마크업 언어, 클라이언트의 스크립트 언어, 그리고 서버의 함수 호출 등과 섞여서 사용되기 때문에 이들을 모두 이해하기란 거의 불가능하다. 또한, 이러한 소스 코드를 이해하는데 필요한 기술들은 계속적으로 늘어나기 때문에 코드를 체계적으로 관리하기가 매우 어렵게 된다. 둘째, 일부 웹 애플리케이션에 사용되고 있는 소프트웨어 기술의 종류가 너무 다양하기 때문에 이해하기도 힘들 뿐 아니라 개발하고 실행시키는데 많은 어려움이 따른다. 이러한 기술들에는 JavaScript[2], servlet[3], SOAP[4], Enterprise JavaBeans[5] 등이 있다.

본 연구는 웹 애플리케이션의 복잡도를 줄이고 사용자의 의도에 반응하기 쉬운 웹 애플리케이션 툴을 제안한다. 특히, 본 연구에서는 브라우저와 서버 소프트웨어 간의 상호작용 방법에 초점을 맞추고자 한다. 이를 위해 폼 인터페이스를 통해 사용자 입력을 받아들이고 사용자에게 응답을 해줄 수 있는 폼 기반 웹 애플리케이션을 이용하였다. 클라이언트 측의 XML 마크업과 서버의 프로그램 언어를 분리함으로써 폼 기반 웹 애플리케이션의 설계를 단순화시키는 모델을 제안한다. 이를 위해 컨트롤러 에디터와 뷰 에디터, 그리고 코드 생성기로 구성된 MVC (Model-View-Controller) 기반 웹 애플리케이션 빌

더를 제안하였다. MVC 기반 애플리케이션 빌더를 사용하여 모델을 설계하는 방법과 XML과 Java 사이의 자동 데이터 변화에 대해서 설명한다.

2. 연구배경

폼 기반 애플리케이션에서 사용자 입력을 받아들이고 이에 대한 적절한 응답이 이루어지는 것은 매우 중요한 작업이다. 다이나믹한 웹 컨텐츠에 대한 이러한 요구사항은 CGI 스크립트와 같은 ad-hoc 방법을 이용함으로써 가능하다. 또한, Java servlet은 HTML 응답 페이지를 생성하기 위해 프로그램 제어 하에 HTML 마크업 언어를 결합한다[3]. 이 방법은 웹 페이지가 변하면 자바 코드도 수정이 이루어져야 하고, 이는 결과적으로 페이지 설계자는 자바 언어를 다룰 수 있어야 함을 의미한다. JavaServer Pages(JSPs)는 호스트와 임베디드 언어 사이의 관계를 역으로 한다. JSPs에서 자바 코드는 HTML 마크업 언어 안에 위치한다. 이는 HTML 문서의 기본 구조를 유지시키면서 자바 호출이 필요한 어느 곳이든지 자유롭게 위치시킬 수 있게 해 준다. JSPs는 servlet으로 변형되어 처리된다. 또한, JSPs는 같은 프로그램 내에 마크업 소스와 자바 소스를 함께 사용할 수 있는데, 같은 페이지 내에 서버 실행 자바 코드와 클라이언트 실행 스크립트 코드가 함께 사용될 경우 매우 이해하기 어려운 프로그램이 될 수 있다.

XForms[6]는 다이나믹한 폼에 대한 일관적이고 선언적인 구조를 정의함으로써 폼 기반 웹 애플리케이션의 이해력을 높였다. XForms는 임베디드 될 수 있는 XML 언어이다. 즉 다른 XML 언어 안에 포함되어 사용될 수 있다. 본 연구에서는 XHTML을 확장하기 위하여 좀더 강력한 능력을 가진 XForms를 이용하였다. XForms는 또한 XPath, XML Schema 그리고 Cascading Style Sheets와 같은 기존의 마크업 언어를 통합할 수도 있다[4]. XForms는 폼 데이터와 UI 컨트롤을 정의하고, 컨트롤과 데이터를 바인딩하며, 이벤트에 동적으로 반응하고, 입력 데이터의 유효성을 검사할 수 있으며, 입력 데이터를 전송하고, 그리고 응답해오는 결과를 출력한다. 가장 중요한 점은, 이벤트 핸들링과 유효성 함수가 내장되어 있기 때문에 많은 XForms 페이지에서 스크립트를 사용하지 않아도 된다는 점이다. 페이지 설계자에게 XForms는 구조적이고 선언적인 설계를 가능하게 해 준다.

브라우저에서 폼 데이터가 입력되면 서버로 전송되어져야 한다. XForms 전송은 HTTP POST request를 이용하여 XML 데이터 전달에 의해 이루어진다. 어떤 데이터가 보내지고, 어디로 보내지며, 어떻게 전송되는지는 XForms 마크업에 기술한다.

서버 측에서 볼 때, 데이터에 대한 궁극적인 표현은 서버 프로그램 언어 구조 안에 있다. 본 연구에서 제안한 애플리케이션 빌더는 Java 언어를 이용한다. 서버 프로그래머가 클라이언트에 사용된 XML을 직접적으로 처리하지 않도록 하기 위해 애플리케이션 개발 동안 XML 스키마로부터 JavaBeans를 생성하도록 하였다. 실행 시 JavaBeans는 들어오는 XML 데이터와 함께 자동적으로 생성되고, 그 후 서버 프로그래머에 의해 정의된 비즈니스 로직 코드에 이 JavaBeans가 전달된다. 비즈니스 로직이 실행된 후 XForms 페이지에 결과가 나타난다. 출력 JavaBeans로부터 XForms 응답 페이지의 XML으로의 데이터 변환은 자동으로 이루어지게 된다.

본 연구에서 제안한 웹 애플리케이션 빌더 설계의 목적은 서버와 클라이언트의 관점을 분리시키는 것이며, 컴포넌트를 개발하는데 필요한 지나치게 다양한 기술들을 줄이고자 하는데 있다. 클라이언트 측에서는 페이지 설계자가 XML 언어를 이용하여 선언적으로 폼을 생성한다. 서버 측에서는 프로그래머가 XML 기술을 사용하지 않고 Java만을 이용하여 비즈니스 로직을 구현하다. 애플리케이션은 애플리케이션 페이지와는 별도로 컨트롤 흐름을 정의하여 개발한다.

3. MVC 기반 애플리케이션

본 연구에서 제안한 웹 애플리케이션 빌더는 비쥬얼 에디터와 코드 생성기로 구성된다. 비쥬얼 에디터는 MVC 애플리케이션의 컨트롤과 뷰를 정의하는데 사용된다. 에디터와 코드 생성기는 애플리케이션 모델을 정의하기 위하여 XML 스키마를 이용한다. 빌더는 J2EE 환경을 기반으로 한다.

3.1 웹 애플리케이션 빌더의 구조

웹 애플리케이션 빌더는 툴 컴포넌트를 통합할 수 있는 프레임웍을 제공한다. 그림 1은 빌더의 주요 컴포넌트가 애플리케이션의 MVC 구조에 어떻게 반영되는가를 보여주고 있다. 컨트롤러 에디터는 애플리케이션의 컨트롤 흐름을 정의하는데 사용된다. 뷰 에디터는 애플리케이션의 XHTML+XForms 페이지

를 정의하는데 사용된다. 이러한 페이지는 또한 데이터 모델과 스키마를 정의하는데 사용된다. 스키마는 XML 스키마 에디터와 같은 툴을 이용하여 생성할 수 있다.

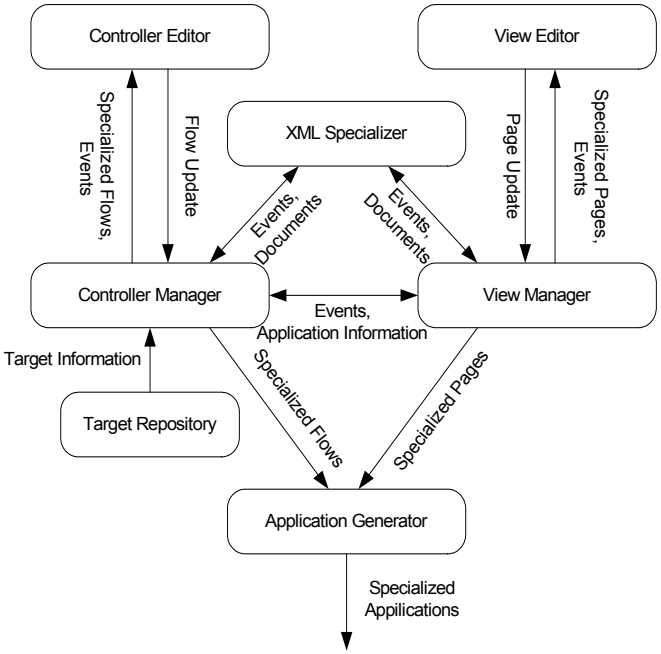

그림 1. 웹 애플리케이션 빌더의 구조

컨트롤러 에디터와 뷰 에디터는 각각 매니저 컴포넌트를 가지고 있으며, 매니저는 에디터와 시스템의 나머지 부분 사이의 중재를 담당한다. 에디터는 그들의 매니저와 데이터를 주고받는다.

3.2 컨트롤러 정의

웹 애플리케이션 빌더의 컨트롤러는 방향성 그래프를 이용하여 클라이어트와 서버 사이에 일어나는 상호작용 흐름을 모델링한다. 이 그래프는 2가지 타입의 노드와 3가지 타입의 간선(전이)으로 표현된다.

・노드

- 페이지 노드 : 클라이언트에서 실행되는 XForms 페이지를 나타낸다.
- 분기 노드 : 서버에서 실행되는 액션 클래스에 해당하는 결정점을 나타낸다.

・간선

- 페이지-분기 전이 : XForms의 <submission> 요소에 해당하는 페이지로부터의 request를 모델링한다.
- 분기-페이지 전이 : 서버에서의 실행결과를 응답 페이지에 보내준다.
- 분기-분기 전이 : 하나의 액션에서 새로운 액션으로의 이동을 모델링한다.

그림 2는 날짜를 요일로 계산해주는 간단한 애플리케이션을 컨트롤러 에디터를 이용하여 모델링한 것이다. “inputPage”와 “outputPage”는 페이지 노드에 해당하고, “computeDay”와 “nextDate”는 분기 노드에 해당한다. 또한, “sub”는 페이지-분기 전이이며, “success”와 “failure”는 분기-페이지 전이에 해당한다. 이와 같이 전이는 간선의 방향성을 이용하여 컨트롤의 흐름을 모델링 한다.

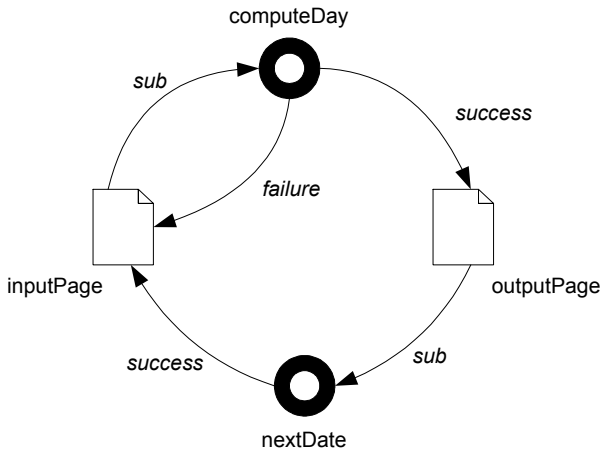

그림 2. 컨트롤러 에디터

3.3 뷰 정의

웹 애플리케이션 빌더의 뷰 에디터는 XForms 페이지를 생성하기위해 사용한다. XForms 페이지는 애플리케이션 뷰 컴포넌트를 정의해 놓은 것이다. 뷰 에디터는 XHTML에 임베디드되어 있는 XForms를 지원한다. XForms는 형식과 장치에 독립적인 XML 언어이다. 동일한 XForms 문서는 음성이나 스타일러스 입력과 같은 서로 다른 기술을 사용한 장치에서도 운영이 가능하다.

뷰 에디터는 XML 기반 웹 페이지를 계층적으로 표현하기 위하여 트리 형식을 이용한다. 이러한 트리 형태를 잘 조작하기 위하여 에디터는 위저드, 문맥지향 메뉴, 드래그 앤 드롭 등의 기능을 포함하도록 하였다. 또한 생성하고 있는 웹 페이지를 미리 볼 수 있는 프리뷰어 기능을 제공함으로써 개발자가 추상적인 페이지를 구체적으로 형상화 할 수 있도록 도와준다. 문맥지향 메뉴는 XHTML+XForms의 각 요소를 계층적으로 보여줌으로써 개발자가 적절한 요소를 선택할 수 있도록 해준다. 또한 위저드는 새롭게 생성된 요소의 속성을 자동으로 할당해 준다.

3.4 모델 정의

XForms 페이지는 XML 스키마를 사용하여 데이타 모델을 정의한다. 브라우저에서 실행되는 XForms 프로세서는 사용자 입력 데이타의 유효성을 자동적으로 체크할 수 있다. 또한 서버에서 실행되는 JavaBeans도 빈즈의 값이 할당될 때 타입 검사에 의해 유효성을 체크한다. 이처럼 클라이언트와 서버 양 측에서 이루어지는 데이터 모델의 유효성 검사는 개발자에게 개발노력을 감소시켜주지만, 대신 클라이언트 인터페이스의 효율적인 상호작용과 정확한 서버 처리가 요구된다. 따라서 실행 시에 클라이언트 상의 XML 데이터가 어떻게 서버의 JavaBeans로 처리되는지가 매우 중요하다. 그림 3은 클라이언트와 서버 사이의 데이터 흐름을 모델링한 것이다. 데이터 흐름의 단계는 다음과 같다.

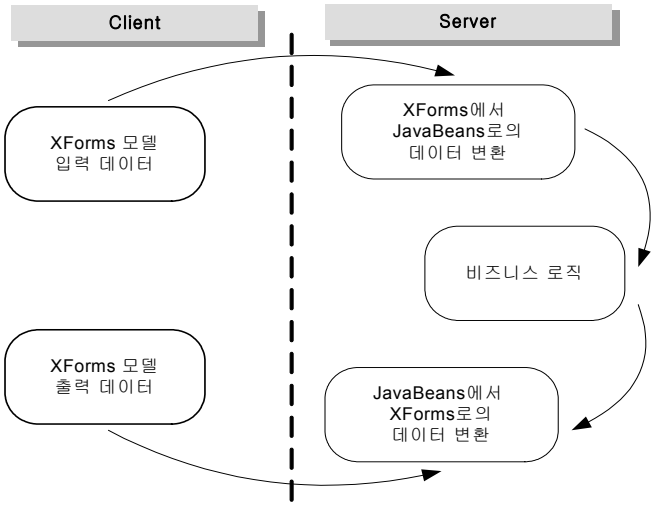

그림 3. 데이터 흐름

1 단계 : 클라이언트 상에서 사용자 데이터는 XForms에 의해 XML 데이터로 인식된다.

2 단계 : 사용자가 <submission> 버튼을 누르면, XML 데이터를 포함한 HTTP POST request가 서버로 전송된다.

3 단계 : 서버상의 실행 코드가 request를 번역하고, XML 데이터를 적당한 JavaBeans 인스턴스로 변환한다.

4 단계 : 액션 클래스가 실행되는 동안, 이 빈즈는 비즈니스 로직으로 전달된다.

5 단계 : 비즈니스 로직이 실행되는 동안, 출력 JavaBeans에 결과 데이터가 저장된다.

6 단계 : 비즈니스 로직의 실행이 완료되면, 클라이언트로 적절한 응답 페이지를 보낸다.

7 단계 : 실행 코드가 응답 페이지를 번역하고, XForms의 "model" 요소에 맞도록 출력 JavaBean의 내용을 변환한다.

8 단계 : 마지막으로, HTTP 응답이 클라이언트로 전달되고, 브라우저의 XForms 프로세서는 내용을 화면에 출력한다.

4. 결론

본 연구는 브라우저와 서버 간의 상호작용에 따른 복잡도를 줄이고 폼 인터페이스를 통해 사용자 입력을 받아들이고 사용자에게 응답을 해줄 수 있는 폼 기반 웹 애플리케이션 툴을 제안하였다. 클라이언트 측의 XML 마크업과 서버의 프로그램 언어를 분리시킴으로써 폼 기반 웹 애플리케이션의 설계를 단순화시키는 모델을 제안하였다. 이를 위해 컨트롤러 에디터와 뷰 에디터, 그리고 코드 생성기로 구성된 MVC 기반 웹 애플리케이션 빌더를 제안하였다. 빌더는 서버 프로그래머가 클라이언트에 사용된 XML을 직접적으로 처리하지 않고 XML 스키마로부터 JavaBeans를 생성한다. 웹 애플리케이션 빌더 설계의 목적은 서버와 클라이언트의 관점을 분리시키는 것이며, 컴포넌트를 개발하는데 필요한 지나치게 다양한 기술들을 줄이고자 하는데 있다.

참고문헌

[1] Berners-Lee, T. and Connolly, D. Hypertext Markup Language-2.0, RFC 1866, Internet Engineering Task Force, Nov. 1995.

[2] ECMA International, Standard ECMA-262, ECMAScript Language Specification, 3rd edition.

[3] Java 2 Platform, Enterprise Edition (J2EE).

[4] World-Wide Web Consortium standards including XForms, XML Schema, XPath and Cascading Style Sheets.

[5] Java Community Process site for Java standards such as Enterprise JavaBeans, Service Data Objects, etc.

[6] Dubinko, M., Klotz, L., Merrick, R. and Raman, T. XForms 1.0, World-Wide Web Consortium, (Recommendation) Oct. 2003.