제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

Model 2 프레임웍 기반 웹 애플리케이션 자동 생성 빌더 설계 및 구현

○
권기현*, 천상호**, 최형진**
*삼척대학교 정보통신공학과
**강원대학교 전자계산학과
e-mail: kweon@samcheok.ac.kr

A Design and Implementation of Automated Builder of Web Application on Model 2 Framework

Ki-Hyeon Kweon*, Sang-Ho Cheon**, Hyung-Jin Choi**
* Dept of Info. & Communication Engineering, Samcheok Natl. Univ.
** Dept of Computer Science, Kangwon Natl. Univ.

요 약

분산 인터넷 환경에서 웹 애플리케이션의 상호운영성, 유연성, 확장성, 유지보수성, 재사용성을 높이기 위해서 프레임웍 및 패턴을 시스템 개발에 적용하는 연구가 진행되고 있다. 프레임웍 및 패턴을 사용하여 시스템을 개발하기 위해서는 여러 가지 설비(facility)를 요구하며 개발자는 시스템에 대해 보다 많은 부분을 고려해야 한다. 본 논문에서는 특정 도메인에서 MVC Model2 프레임웍을 기반으로 하는 시스템에 대해 공통으로 사용 가능한 코드를 재사용할 수 있도록 시스템의 골격 코드 및 프로토타입 형태의 시스템을 자동 생성하는 빌더를 설계하고 구현한다. 생성되는 프로토타입 시스템에서는 커스텀 태그 및 재사용 가능한 빈을 정의하고 XML, DOM, 국제화, 지역화 등을 지원하도록 작성되어 커스텀 태그 및 빈의 추가 및 변경에 의해 웹 애플리케이션의 기능을 변경시키고 성능을 향상시키는 것이 용이하게 된다. 본 연구에서 설계하고 구현한 Model 2 기반 프로토타입 자동생성 도구는 웹 애플리케이션 개발시에 활용되어 생산성 향상에 기여할 수 있다.

1. 서론

프레임웍 및 패턴을 시스템 개발에 적용하는 것은 웹 애플리케이션의 상호운영성, 유연성, 확장성, 유지보수성, 재사용성을 극대화하기 위한 것이다. 또한, 한번 작성된 시스템은 여러 가지 형태(DAO의 분리 등)로 분리가 가능하여 시스템의 개선 및 확장 그리고 다른 시스템을 개발할 때 생산성을 높여 줄 수 있다[1][2]. 그러나, 프레임웍 및 패턴을 사용하여 시스템을 개발하기 위해서는 여러 가지 설비(facility)를 요구하며 개발자는 시스템에 대한 보다 많은 부분을 고려해야 한다. 그리고, 프레임웍 및 패턴을 사용하는 궁극적인 목표가 상호운영성 등의 특징을 만족하면서 재사용성을 극대화하기 위한 것이라면 특정 도메인하의 프레임웍 및 패턴은 일부의 수정만으로 재사용될 수 있어야 있다.

본 논문에서는 특정 도메인에서 MVC Model2 프레임웍을 기반으로 하는 시스템에 대해 공통으로 사용 가능한 코드를 재사용하여 시스템의 골격 코드 및 프로토타입 형태의 시스템을 자동 생성하는 빌더를 설계하고 구현한다.

논문의 내용은 2장에서는 분산 프레임웍 및 패턴 그리고 Model 2 프레임웍에 관한 이론적인 내용을 설명한다. 3장에서는 Model 2 프레임웍 기반 시스템에서 공통적으로 사용될 수 있는 부분을 정의하고 본 논문의 결과로서 자동생성 되는 Model 2 기반 프로토타입의 내용을 설명한다. 4장에서는 Model 2 프레임웍 기반 자동생성 빌더를 설계 및 구현 사항에 대해 설명한다. 마지막으로 결론 및 향후 연구 방향을 제시한다.

2. 관련연구

인터넷 환경에서 시스템을 구현할 때 컴포넌트를 이용하여 사용되는 디자인 패턴을 정리하면 다음과 같다.

2.1 웹서버와 CGI 및 서블릿 이용 패턴

Tier 1은 브라우저, Tier 2는 서블릿이나 CGI로 연

동하는 애플리케이션 서버 그리고 Tier 3은 데이터베이스 스토어로서 동작한다. 이 경우는 모든 입출력 처리가 브라우저에 의한 인터페이스로 가능한 경우이다[3].

2.2 웹서버와 CORBA 이용 분산 패턴

Tier 1은 브라우저와 CORBA 클라이언트, Tier 2는 서블릿과 CORBA 구현 객체 그리고 Tier 3은 데이터베이스 스토어로서 동작한다. 이 경우는 이질적인 환경이나 기존의 이질적인(legacy) 시스템과의 통합이 요구된다[4].

2.3 웹서버와 RMI 이용 분산 패턴

Tier 1은 브라우저와 RMI(Remote Method Invocation) 클라이언트, Tier 2는 서블릿과 RMI 구현 객체 그리고 Tier 3은 데이터베이스 스토어로서 동작한다. 이 경우는 사용자 인터페이스 환경이 동적인 경우에 사용할 수 있으며 자바로 구성된 시스템에 사용될 수 있다[5].

2.4 Model 2 프레임웍

Model 2 프레임웍은 모델, 뷰, 컨트롤러를 이용하여 MVC(Model-View-Controller)의 구조로 작성하는 방법으로 기능을 캡슐화하여 변경에 대해 영향을 최소로 하는 구조이기 때문에 확장성, 유지보수성 면에서 좋은 방법으로 인식되고 있다. 모델 2 구조는 View와 비즈니스 객체를 분리하여 웹 개발 프로젝트를 작성하는 것을 기본적인 내용으로 한다. 개발 중에 비즈니스 객체의 상태가 변경될 요소가 많은 경우에는 비즈니스 객체의 개발이 뷰와 연관되지 않도록 하기 위해서 모델 2 구조를 사용하는 것이 바람직하게 된다.

3. Model 2 기반 자동생성 내용 정의

컴포넌트 자동 생성 시스템을 구현하기 환경을 다음과 같이 설정하고 설계한다.

3.1 구현 환경에 따른 Model 2 프레임웍 구조

컴포넌트 자동 생성 시스템의 구현 환경은 Java 기반의 웹 애플리케이션으로 설정하며 Model 2 프레임웍에 대한 MVC 부분은 다음과 같다.

- Model : Java Bean
- View : JSP(Java Server Page)
- Controller : Java Servlet

위와 같이 JSP, 서블릿 및 Java 빈 환경에서 자동 생성되어지는 시스템을 위한 Model 2 프레임웍은 다음과 같이 정의한다.

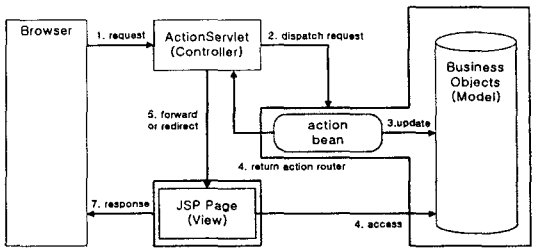

그림 1 구현환경의 Model 2 구조

이 프레임웍에서는 하나의 서블릿을 컨트롤러로서 사용한다. 이 서블릿을 액션 서블릿이라고 하며 모든 HTTP 요청은 액션 서블릿에 의해 처리되어 요청(request)을 자바 빈으로 전달하게 된다.

액션 빈은 비즈니스 객체를 갱신하고 제어를 액션 서블릿으로 리턴한다. 액션 서블릿에서는 처리 결과를 표시하기 위해 포워드(forward)하거나 재지향(redirect)를 통해 JSP 페이지를 호출하게 된다. 이 JSP 페이지에서는 비즈니스 객체를 참조하거나 비즈니스 객체를 참조하는 커스텀 태그를 정의하여 페이지를 구성한 후에 브라우저에 응답하는 구조이다.

3.2 자동 생성될 컴포넌트 및 요소 정의

그림 1의 Model 2 프레임웍 구조에 입각하여 컴포넌트 자동 생성 시스템에서 생성될 컴포넌트 및 요소는 다음과 같다.

3.2.1 뷰(view) 관련 페이지

(1) 기본 JSP 페이지

웹 애플리케이션의 칫 페이지 및 web.xml 문서

(2) 뷰 관련 템플릿(template) 페이지

템플릿은 웹 애플리케이션에서 액션이나 이벤트에 따라 뷰의 레이아웃을 다르게 표시하기 위해 사용한다.

(3) 뷰 영역(region) 별 페이지

템플릿을 호출하는 메인 페이지(page.jsp)와 각 영역별 페이지가 해당된다. page.jsp가 호출되면 regionDefinition.jsp에 정의된 각 영역은 hscf.jsp의 템플릿 구조에 의해 레이아웃이 결정되게 된다.

3.2.2 태그 라이브러리 정의 문서

태그 라이브러리 정의 문서는 커스텀(custom) 태그

(데이터베이스, 국제화, 뷰, 컨트롤러, 폼 등) 작성에 사용되는 태그 정의 문서이다.

3.2.3 DDL 정의 문서

입력된 XML에 입력에 의해 자동 생성된 데이터베이스 DDL(Database Definition Language) 문서가 생성되는 것이 필요하다.

3.2.4 컨트롤러 서블릿과 프로퍼티 파일

(1) 컨트롤러(Controller) 서블릿

- SetupServlet.java : web.xml에 의해 초기화를 수행하며 데이터베이스 커넥션을 설정한다.
- ActionServlet.java : 폼 전송 등의 서버로의 입력이 있을 때 자동으로 호출되어 actions.properties에 지정된 액션으로 이동된다. ActionServlet의 처리 흐름은 그림 2와 같다.

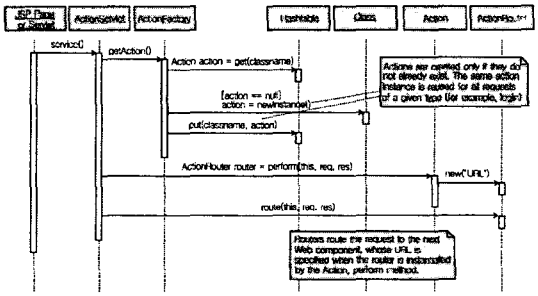

그림 2 ActionServlet sequence 다이어그램

(2) 프로퍼티 파일

ActionServlet에 의해 전달될 액션 목록을 가지는 actions.properties와 국제화에 사용되는 키(key) 이름에 대한 프로퍼티 파일이 요구된다.

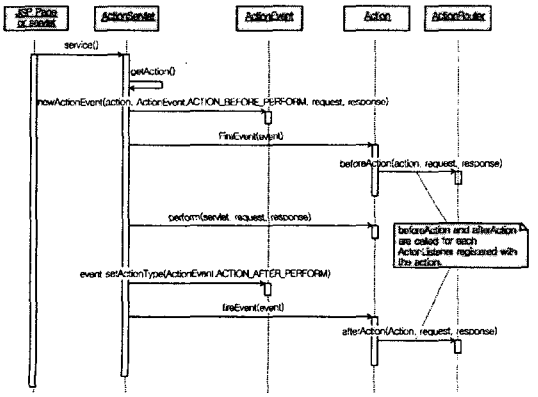

그림 3 ActionEvent sequence 다이어그램

3.2.5 자바 빈(Model) 파일

(1) Action Bean

Action 빈은 웹 애플리케이션에서 사용자 액션이 발생하면 ActionServlet에 의해 개시되어 액션을 처리하게 된다. 다음은 액션을 처리하는데 요구되는 Bean 들이며 액션의 처리 흐름은 그림 3과 같다.

(2) Actions Bean

Actions 빈은 웹 애플리케이션에서의 실제 액션을 정의한 Bean으로 각 액션은 하나의 Bean으로 작성된다. 본 프로토타입에서는 2개의 기본 액션을 제공하고 개발자가 추가 할 수 있도록 구성한다.

(3) Beans Bean

웹 애플리케이션 전체에 걸쳐 사용되는 Bean으로 regions, jdbc, i18n, html, app 관련 Beans 등이 필요하다.

(4) Tags Bean

커스텀(custom) 태그 지원하는 라이브러리가 필요하다. 커스텀(custom) 태그 라이브러리는 필요에 따라 추가 및 제거 될 수 있으며 독립적인 관리가 가능하다.

4. Model 2 기반 빌더 구현

4.1 프로토타입 빌더 제한 사항

- DB에 접속하는 데이터소스는 한 개만 존재한다.
- jsp의 파일위치, DDL의 파일위치, java source의 파일위치는 하나씩만 설정한다.
- 여러 개의 테이블에 대해 설정을 할 수 있다.
- 한 테이블에 대한 정보에는 RDBMS 상의 테이블 이름과, 한 개짜리 Entity class에 대한 class 명, 해당 class의 java package, 화면상에 출력될 명칭이 하나씩 존재한다.
- 필드는 여러 개가 올 수 있는데, 각 필드에는 java class에서 쓰일 name 속성, RDBMS 의 컬럼 이름으로 매칭될 colname 속성, 화면상에 출력될 display 속성, java type으로 변환할 type 속성이 지정되어야 한다.
- 필드의 크기를 나타내는 size 속성은 숫자와 문자 필드에 필수이며, scale 속성은 숫자의 경우 소수점이하의 속성이 필요할 때 쓰이는 속성이다.
- 프라이머리 키는 여러 개가 올 수 있는데 필드에 첨가된 요소여야만 한다.

4.2 DTD 및 입력 명세(XML) 작성

4.1 절의 규약을 정리한 DTD는 그림 4와 같다.

```
<!ELEMENT auto-coding ( deploy, tables ) >
<!ATTLIST auto-coding name NMTOKEN #REQUIRED >
<!ELEMENT deploy ( doc-root, src-root, sql-root, datasource-name ) >
<!ELEMENT doc-root ( #PCDATA ) >
<!ELEMENT src-root ( #PCDATA ) >
<!ELEMENT sql-root ( #PCDATA ) >
<!ELEMENT datasource-name ( #PCDATA ) >
<!ELEMENT tables ( table+ ) >
<!ELEMENT table ( table-name,class-name,javapackage,display,primary-keys, fields ) >
<!ELEMENT table-name ( #PCDATA ) >
<!ELEMENT class-name ( #PCDATA ) >
<!ELEMENT javapackage ( #PCDATA ) >
<!ELEMENT display ( #PCDATA ) >
<!ELEMENT primary-keys ( primary-key+ ) >
<!ELEMENT primary-key ( #PCDATA ) >
<!ELEMENT fields ( field+ ) >
<!ELEMENT field EMPTY >
<!ATTLIST field name NMTOKEN #REQUIRED >
<!ATTLIST field colname NMTOKEN #REQUIRED >
<!ATTLIST field display NMTOKEN #REQUIRED >
<!ATTLIST field type ( int|long |double |java.lang.String|java.util.Date) #REQUIRED >
<!ATTLIST field list ( false | true ) #REQUIRED >
<!ATTLIST field size NMTOKEN #IMPLIED >
<!ATTLIST field scale NMTOKEN #IMPLIED >
```

그림 4 autocode.dtd

그림 4의 DTD를 근간으로 XML 파일을 작성하여 Model 2 기반 프로토타입 빌더의 입력으로 사용하며 이 XML 파일을 해석하는 Parser를 작성이 요구된다.

4.3 XML Input으로 Model 2 Builder 구현

4.3.1 Model 2 기반 빌더의 동작 구조

생성 정보 분석 단계에서는 XML 파일을 읽어 들여 ConfigReader 클래스에서 Table 클래스의 배열형을 생성하여 CodeInfo 클래스로 리턴 한다. XML Input으로 Model 2 기반 프로토타입을 생성하기 위한 빌더의 동작 구조는 그림 5과 같다.

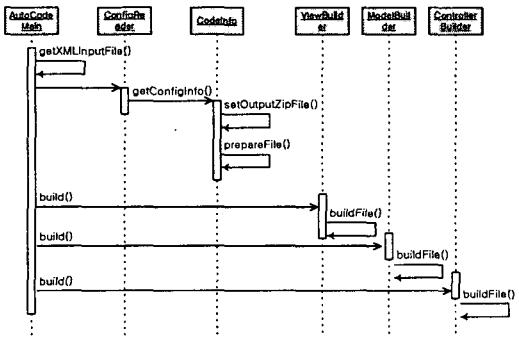

그림 5 Model 2 기반 빌더의 동작 구조

4.3.2 Model 2 빌더의 클래스 다이어그램

소스코드 생성 단계는 일관성 있는 구조를 위해 모든 빌더 클래스는 Builder 클래스를 상속하여 작성된다. 생성되는 코드는 MessageFormat에 의해 채워져서 작성되도록 템플릿(template) 파일로 저장되어 사용된다. 그림 6은 Model 2 빌더의 클래스 다이어그램이다.

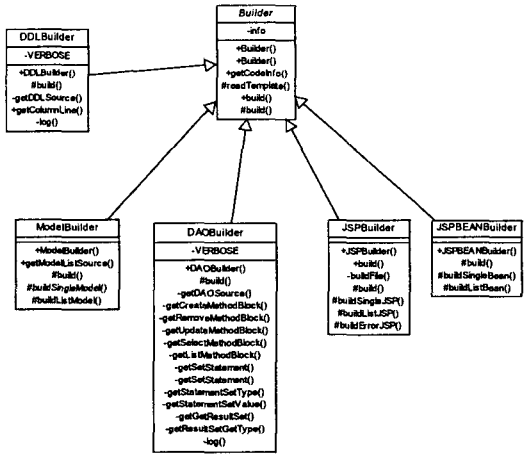

그림 6 Model 2 빌더의 클래스 다이어그램

5. 결론 및 연구 방향

본 연구에서는 웹 애플리케이션을 개발할 때 자주 이용되는 컴포넌트와 관련 요소들을 자동 생성하기 위해 환경에 적합한 Model 2 프레임웍을 제시하고 이 프레임웍에 기반하여 소프트웨어 컴포넌트와 관련 요소들을 생성하는 빌더를 설계하고 구현하였다. 본 연구에서 설계고 구현한 Model 2 기반 프로토타입 자동생성 도구는 웹 애플리케이션 개발시에 활용되어 생산성 향상에 기여할 수 있다. 연구 방향으로는 웹 애플리케이션에 패턴에 대한 적용 및 전체 시스템의 성능을 평가하는 부분을 보완하는 것이 필요하다.

참고문헌

[1] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, Pattern-Oriented Software Architecture - A System of Patterns, Wiley and Sons, 1996.

[2] Gamma, Helm, Johnson, Vlissides, Design Pattern. Addison-Wesley, 1995.

[3] Berg, Daniel J. and Fritzinger, Steven. Advanced Techniques for Java Developers, Wiley. 1998.

[4] Mowbray, Thomas J. and Ruh, William A. Inside CORBA: Distributed Object Standards and Applications, Addison Wesley, 1997.

[5] Orfali, R, Hashley, D., and Edwards, J. The Essential Distributed Object Survival Guide. Wiley. 1996.