제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

Web 환경에서의 Globalization 컴포넌트의 설계 및 구현

김도형, 이유호
삼성 SDS
e-mail : dohyung1.kim@samsung.com

Design and Implementation of Globalization Component For Web

Do-Hyung Kim, You-Ho Lee
Samsung SDS

요 약

컴포넌트 기술은 재사용성을 기반으로 어플리케이션 개발의 생산성을 높일 수 있어 소프트웨어를 신속하고 효과적으로 개발할 수 있는 기술로써 채택되고 있으며, 특히 EJB 컴포넌트는 자바를 기반으로 하며 산업계에서 널리 이용되고 있는 기술이다. 이러한 컴포넌트의 유형은 크게 비즈니스 컴포넌트와 기능별 컴포넌트로 분류할 수 있으며, 비즈니스 컴포넌트는 특정 응용분야와의 상관성이 커짐에 따라 비즈니스 공용 컴포넌트, 비즈니스 핵심 컴포넌트, 비즈니스 응용 컴포넌트로 구분하고 있다. 본 논문은 일반 Web Application 에서 업무 구현상 필요성 혹은 솔루션 제품의 해외 판매 시 필요로 하는 비즈니스 공용 컴포넌트인 Globalization 컴포넌트의 개발과정과 실제 적용 가능성 및 효용성 측면을 제시한다

1. 서론

컴포넌트 소프트웨어 또는 컴포넌트웨어 (Componentware)란 의미는 일반적으로 객체지향 기술의 원리를 이용하여 제작한 소프트웨어 모듈을 의미하며, 기계 부품과 같이 소프트웨어도 하나 하나의 부품으로 제작한 다음 이를 조립해 보다 복잡한 소프트웨어를 만들 수 있을 것이라는 아이디어에서 시작한 개념이다. 이러한 컴포넌트 기술은 소프트웨어를 신속하고 효과적으로 개발할 수 있는 대안으로 인식되며 많은 사람들의 관심을 모으고 있다

특히, 최근 개발되는 시스템은 Web 기반의 Application 이 주류를 이루고 있으며, SW 업종의 회사는 국내 시장의 포화상태 및 과당경쟁에 대한 해결책으로 해외 SI 프로젝트 진출 및 솔루션의 해외 판매를 강화하고 있다. 이와 같은 시장상황의 변화에 따라 Globalization 혹은 Localization 기능의 필요성이 증대되고 있다. 물론 Globalization 기능은 DBMS 나 WAS (Web Application Server) 벤더 차원에서 제공하는 경우가 있으나, 본 논문에서는 Application 차원에서 공용 컴포넌트로 기능을 추출하여 Globalization 기능을 구현한 방법과 제작된 컴포넌트의 실제 배포 및 적용 사례를 통한 효용성 및 문제점 대하여 살펴보도록 하겠다.

구현 환경은 Web Application 을 대상으로 하였고, 최근 Enterprise Web Application 의 중요 Platform 인 .Net 과 J2EE 중 산업계에서 널리 이용되고 있는 기술인 J2EE 환경에서 EJB 컴포넌트로 구현 하였다.

SI 업체 혹은 솔루션개발 업체의 입장에서 앞에서 언급한 시장상황에 효과적으로 대응하고, 공용 기능의 회사차원에서 재사용을 통하여 생산성 및 효율성 향상을 목표로 하였다.

2. Globalization Component 추출

컴포넌트 추출을 위해서는 시스템에서 제공해야 할 기능을 기능적, 계층적으로 구분하여 의미있는 서비스들을 제공하는 Self contained 하고 다른 모듈과 독립적인 실행단위로 모듈화한다.

일반적으로 CBD (Component Based Development)

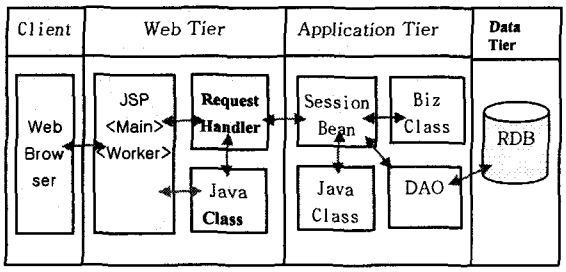
방법론에서 이야기하는 컴포넌트 추출 방법에는 1) 유즈케이스 분석을 통하여 공통으로 사용할 수 있는 컴포넌트를 도출하거나, 2) 분석/설계 과정을 통하여 공통으로 사용할 수 있는 UI 컨틀롤 및 기능을 도출하거나, 3) Entity 클래스간의 상관관계 분석을 통하여 비즈니스 컴포넌트를 도출 또는 4) 도메인 전반에 대한 지식을 가진 전문가가 컴포넌트에 대한 지식을 바탕으로 표준 비즈니스 기능 컴포넌트를 도출 할 수 있다.

본 논문에서 추출된 컴포넌트는 분석/설계 과정을 통하여 공용으로 사용할 필요가 있는 기능을 추출하였다.

3. *Globalization Component* 아키텍처

본 논문에서 설계 구현한 Globalization Component 의 아키텍처 구조는 J2EE 환경에서 주로 사용하는 MVC 아키텍처를 기본으로 하였으며, Web 계층의 복잡성을 해결하기 위하여 최근에 각광 받고 있는 MVC 모델 2 구조를 사용하였다. Application Tier 는 상대적으로 성숙된 환경인 EJB 표준환경을 사용하였고, 비교적 Persistence 처리가 단순하여 EntityBean 은 사용하지 않고 DAO 를 직접 사용하였다. 본 논문에서 EJB 로 설계 구현한 Globalization Component 의 기본적인 아키텍처 구조는 (그림 1)과 같으며 주요 구성 요소의 기능을 살펴보면 다음과 같다.

- **JSP:** Web Tier 의 JSP 는 Main JSP 와 Worker JSP 로 구성되어 있으며, Main JSP 는 화면을 처리하여 주는 역할을 하며, Worker JSP 는 저장 작업 수행시 해당 Request Handler Class 를 수행후 호출되며 서버 Application 작업에 에러가 있을 경우 메시지를 표시하거나 정상적으로 수행할경우 MainJSP 의 Script Function 을 호출한다.
- **Request Handler Class:** Request Handler 는 Client Browser 에서 Submit 한 Data 를 HttpServlet Request 객체로부터 전달 받아서 해당 Session Bean 을 호출하고 수행 결과를 Request Attribute 에 저장하고 Return 한다. Web Tier 의 MVC Model2 구현을 위한 Class 로써 Client Request 된 Data 처리를 전담 함으로써 화면 표시를 전담하는 JSP 와 Request 처리를 전담하는 Class 로 명확하게 기능을 구분할 수 있다.
- **MainServlet:** Web Tier 의 MVC Model2 구조를 구현하기 위한 Controller 로써, 클라이어트에서 작업을 Submit 하고 Web Controller 에 해당하는 MainServlet 이 구동되면 XML 파일에 정의된 자료를 이용하여 Request Handler 를 호출하고 후속으로 수행할 Main JSP 를 수행시킨다.
- **EJB 관련 Class:** *Session* 빈을 구현하기 위한 Home Interface, Remote Interface, EJB Class 로 구성되며 기능 구현을 위한 로직과 Transaction 을 관리한다.
- **DAO 관련 Class:** Data Access 및 처리를 전담하는 Class 이며, Interface Class, 구현(Implementation) Class, DAO Factory Class 로 구성된다. 이기종 DBMS 에 대한 유연한 대처를 위하여 DBMS 별로 Implementation Class 를 작성할 수 있도록 Factory Pattern 을 사용한다. VO Class 는 Table Column 과 밀접한 관계를 갖고 있고 Update 나 Select 해온 Data 를 갖고 있으며 Get, Set 메소드를 사용하여 값을 할당하고 구할 수 있다.

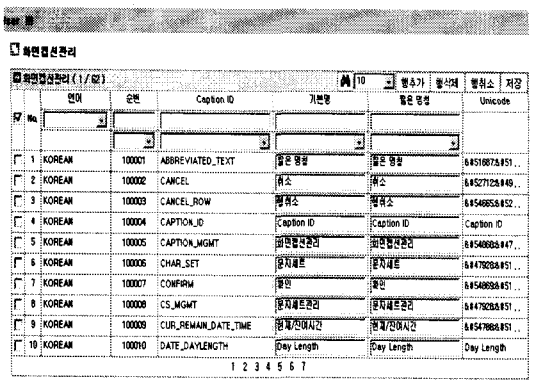


(그림 1) Globalization Component 아키텍처

4. Globalization Component 기능 및 특징

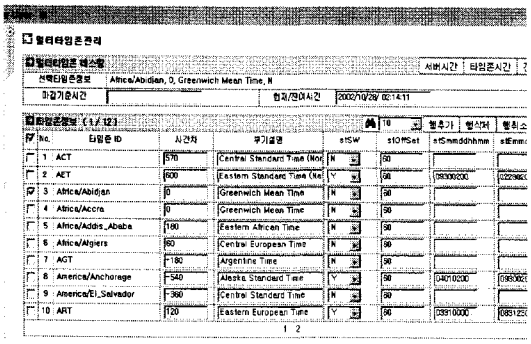
본 논문에서 설계 구현한 Globalization Component 는 멀티 랭귀지, 멀티 타임존, 멀티 포맷 컴포넌트로 구성되어 있으며, 각 컴포넌트의 주요 기능과 구현상 특징은 다음과 같다.

- **멀티 랭귀지 (Multi Language):** 멀티 랭귀지 컴포넌트는 다국어 사용 지원을 위하여 주요 관리 데이터를 화면캡션, 코드, 메시지정보를 대상으로 하였고, 관련데이터를 CRUD 할 수 있는 관리기능과 실제 Application 에서 호출하여 사용할 수 있는 기능으로 구성한다. 다국어 지원을 위한 데이터 관리 방식은 테이블로 관리하거나 언어별 Jsp 파일로 관리하는 방식이 있으나 본 논문에서는 테이블로 관리하는 방식으로 구현되었다. 테이블로 관리할 경우 장점은 캡션 관련 정보를 사이트별로 손쉽게 정의하여 관리하기 쉬우나 화면을 그리기 위하여 DB 를 Access 하므로 성능상으로는 약점이 있다. 다른 특징으로는 다국어 지원을 위한 데이터를 화면캡션, 코드, 메시지를 주요 관리 대상으로 선정하여 한 테이블에서 일관성 있게 관리할 수 있도록 하였다. 또한 Web Application 에서의 멀티 바이트 문자의 원활한 처리를 위하여 원어 데이터와 함께 유니코드 형태의 데이터로 동시에 저장한다. 상세 기능으로는 화면캡션, 메시지, 코드관리의 세가지 데이터에 대하여 멀티랭귀지 데이터를 입력, 수정, 삭제, 조회 할 수 있는 관리 기능과 유니코드 변환을 위한 기능을 제공하며, 일반 Application 에서 저장된 다국어 정보를 손쉽게 이용할 수 있는 API 를 제공한다.



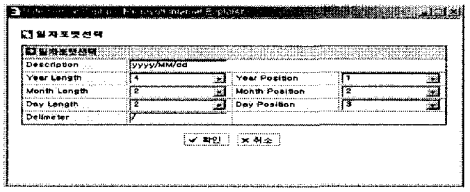
(그림 2) 멀티랭귀지 관리 화면

- **멀티 타임존 (Multi Time Zone):** 멀티 타임존 컴포넌트는 타임존 데이터를 CRUD 할 수 있는 관리 기능과 타임존 관련 기능 서비스를 제공한다. 구현 특징으로는 멀티 타임존 기능 구현을 위하여, Java 의 표준 타임존 정보를 사용하고, 시간 계산을 하기 위한 기본 클래스로 Calendar Class 의 메소드를 사용한다. Java 의 표준 타임존 정보를 사용하므로써 Java Platform 의존 경향이 있고 범용성이 떨어지는 약점을 가질 수 있다 상세 기능으로는 타임존 정보를 입력, 수정, 삭제, 조회 할 수 있는 관리 기능 및 화면을 제공하며, 타임존별 실시간 표시기능, 서머타임 처리 기능, 타임존별 현재 시간 정보, 타임존간의 시간변환, 차이시간/ 잔여시간 계산등의 기능을 제공한다.



(그림 3) 멀티 타임존 관리 화면

- **멀티 포맷 (Multi Format):** 멀티 포맷 컴포넌트는 국가별 혹은 시스템별로 다르게 사용할 수 있는 숫자, 일자 포맷에 대한 정의기능과 클라이언트와 서버의 포매팅, 디포매팅 기능을 제공한다. 구현상 특징으로는 지역별 혹은 시스템별 포맷형식(문자,숫자)을 시스템에 맞게 정의할 수 있어서, 비교적 환경변화에 유연하게 대처할 수 있다. 상세 기능을 살펴보면 숫자 포맷 적용을 위한 포맷 정의, 포매팅(클라이언트/서버), 입력데이타 체크(클라이언트), 디포매팅(클라이언트/서버) 기능을 제공하며, 일자 포맷 적용을 위한 포맷정의, 포매팅(클라이언트/서버), 입력데이타 체크(클라이언트), 디포매팅(클라이언트/서버), 칼렌더 기능을 제공한다.

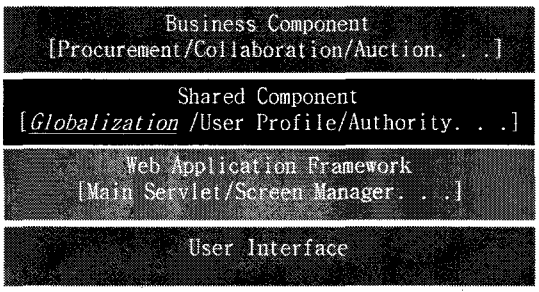


(그림 4) 일자포맷 정의 화면

5. 배포 및 사용

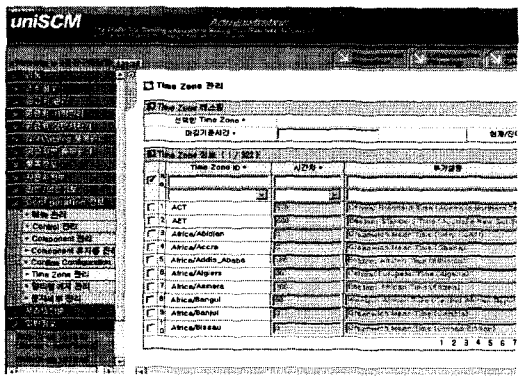
컴포넌트를 조립하여 소프트웨어를 개발하거나, 기존의 시스템에 통합하여 구축할 경우 제작된 컴포넌트에 대한 품질뿐만 아니라 사용자가 요구하는 명세(Specification)에 적합한 컴포넌트들을 조립하여 사용하는 것이 중요하다. 아래는 Globalization Component 를 실제 사용하는 3 가지 경우에 대해 설명한다.

- **Case 1:** Globalization Component 용 별도의 서비스 포트를 사용한 서버를 설치하여, 멀티랭귀지 멀티 타임존 등의 정보를 정의한다. 실제 일반 어플리케이션은 서비스를 이용하기 위하여 필요한 Class 나 JavaScript 파일을 배치시켜 사용할 수 있다. 컴포넌트 제공 기능을 위한 별도의 시스템을 설치 하는 개념이므로 약간의 사치스러운 배치일 수 있으나 기존 시스템과의 독립성이 좋다
- **Case 2:** 기존 Application 시스템의 서브시스템개념으로 별도의 Web Directory 에 배치하여 서브시스템간의 내비게이션으로 이동한다. 시스템 관리 서브시스템 정도에 해당하며, Case 1 보다는 Application 시스템과의 통합성은 높으나 독립성은 떨어진다.
- **Case 3:** Globalization Component 관리화면을 기존 Application 의 관리 서브시스템의 한 메뉴로 배치 할 수 있다. 기존 응용시스템과의 통합성은 가장 좋으나, 기존 시스템의 환경에 가장 많은 영향을 받는다.



(그림 5) SCM 에서의 Globalization Component Position

본 논문에서 설계 구현한 Globalization Component 는 c-SCM (Collaborative Supply Chain Management) 솔루션의 관리 서버시스템의 한 메뉴로 배치하여 (Case 3) 적용하였다. 해당 솔루션 제품은 국내 및 해외 판매를 목표로 하였고, 솔루션 제품의 특성상 사이트 적용시 수정 공수의 최소화가 중요하다. (그림 5)는 c-SCM 에 적용된 Globalization Component 의 Position 을 나타내며, (그림 6)은 c-SCM 관리 서버시스템의 한 메뉴로 배치된 멀티타임존 관리화면을 보여준다.



(그림 6) c-SCM 에 적용된 멀티타임존 관리화면

특히 비즈니스 컴포넌트는 이질적 사용환경에서의 (DBMS, WAS, Platform, OS 등) 독립성 확보가 양질의 컴포넌트로 평가될 수 있는 중요한 요소라고 할 수 있다. 환경에 대한 독립성이 커지면 그만큼 사용하기 쉽고, 재사용성이 높아지기 때문이다. 이론적으로 J2EE 환경을 기반으로 하고, 표준 SQL 을 사용하였다면 컴포넌트 수정없이 바로 수행 시킬 수 있으나, 실제 적용 환경에서는 다양한 벤더사가 제공하는 제품별 고유속성이 존재하므로 J2EE 표준환경을 준수한 컴포넌트 일지라도 일부 수정은 필요하다

제작된 Globalization Component 를 다른 환경의 WAS, DBMS 적용시 일부 수정사항에 대해 사례별로 살펴본다. 다른 환경으로 전환시 수정 폭은 매우 적어서 환경 변화에 따르는 투입 공수는 적다고 할 수 있다. 당연한 이야기이지만 전제조건은 그 제품을 잘 알고 있을때 가능하다. 아래의 수행 결과에서는 본 논문의 목적상 특정 제품은 언급하지 않고 원래의 타겟 환경을 ***A*** 로, 환경 변화시의 제품을 ***B***, ***C*** 등으로 표현한다.

- ***B*** **WAS 전환시 수정사항 사례:**
  - JSP 에서 AtrrayList 사용시 ***B*** WAS 에서 자동인식이 되지 않으므로 java.uitl.ArrayList import 추가.
  - XML 파서 사용시 WAS 에서 지원하는 JDK 버전 차이로 인한 관련 클래스 재 컴파일.
  - ***B*** WAS 고유의 Deployment Descriptor 반영을위하여 일부 정의 내용 수정
  - ***A*** WAS 에서는 오류로 인식하지 않으나, ***B*** WAS 특성상 Null 값 Assign 오류 발생 케이스 수정
- ***C*** **WAS 전환시 수정사항 사례:**
  - ***C*** WAS 특성상 con.setAutoCommit(false) 사용항목 제거
  - ***C*** WAS 고유의 Deployment Descriptor 반영을 위하여 일부 내용 수정
  - ***C*** WAS 고유의 Tablibrary 배치 방법으로 변경
  - JSP 에서 Tag 선언 방식을 ***C*** WAS 선언 방식으로 수정
- ***B*** **DBMS 전환시 수정사항 사례:**
  - 을 문자값으로 저장못함에 따르는 sql 문 수정
  - DB 변화에 따르는 대소문자 구별 환경변수 값 차이에서 오는 관련 내용 수정
- ***B*** **OS 전환시 수정사항 사례:**
  - JavaClass 에 대한 별도 컴파일 필요없이 ***A*** OS 배치환경 그대로 적용

6. 결론

본 논문에서는 기술 및 시장환경 변화에 따라서 필요로하는 Globalization 혹은 Localization 기능을 컴포넌트화 하였던 과정을 알아보았고, 실제 적용 사례를 통한 비즈니스 컴포넌트의 사용 가능성및 효용성에 대해서 살펴보았다. 특히 이질적 사용환경의 독립성 확보 및 재사용성 향상 측면에서 이론만큼 만족할 만한 결과는 아니나 이와 같은 노력을 지속적으로 이어감으로써 컴포넌트를 사용하여 SW 개발시 생산성, 품질, 납기에 대한 기대를 높여줄 수 있으리라 생각된다.

참고문헌

[1] Richard Monson-Haefel, "Enterprise Java Beans," 2nd ED., O'Reilly, 2000.
[2] Martin Fowller and Kendal Scott, UML Distilled, 2nd Edition, Addison-Wesley, 2000
[3] OMG Unified Modeling Language specification version 1.4, , OMG, 2001
[4] John Cheeseman and John Daniel, UML Component, Addison-Wesley, 2001, p88~95
[5] Rational Unified Process, Rational Software Corp. , 2001
[6] Ivar Jacobson, Grady Booch and James Rumbaough, The Unified Software Development Prpcess , Addison-Wesley, 1999
[7] Weblogic Reference Manual, "www.bea.com"
[8] Jeus Reference Manual, "www.tmax.co.kr"
[9] Jrun Reference Manual
[10] Oracle Reference Manual, www.oracle.com
[11] 강금석, "컴포넌트 추출방법에 대한 연구", 삼성 SDS Special Report, 2002