N-Tier 플랫폼 환경에서 인포그래픽을 기반으로 N-스크린 융합 표현 계층의 설계

이명호
세명대학교 전자상거래학과

A Design of N-Screen Convergence Presentation Tier by using Infographics Based on N-Tier Platform

Myeong-Ho Lee
Department of eCommerce, Semyung University

요 약 IT의 환경은 컴퓨터와 인터넷을 물과 공기처럼 이용할 수 있고, 새로운 성장 동력이 될 수 있는 클라우드 컴퓨팅, 빅데이터 그리고 비즈니스 지능의 시대로 진화되고 있다. 그러나 다양한 상호작용과 인포그래픽 측면에서는 사용자의 요구가 점점 더 많아지고 있으며, 웹 브라우저가 제공하지 못하는 추가적인 기능의 필요성도 점점 많이 요구되고 있는 실정이다. 따라서 본 연구에서는 이러한 N-티어 플랫폼 환경에서 인포그래픽을 기반으로 N-스크린 융합 표현 계층의 설계를 제안하도록 한다.

• Key Words : 인포그래픽, MVC 디자인 패턴, 융합 표현 계층, N-티어 플랫폼

Abstract The environment of IT is, currently, on its developing process to the period of cloud computing, big data, and business intelligence which not only enable computer and internet to be utilized like the water or the air, but also be a new motivating force for its advance. In the respect of various interactions and Infographics, however, it is requiring more demands from its users, and additional functions which cannot be provided by the Web Browser. In this study, therefore, it will be suggested a design of N-screen convergence presentation tier by using infographics based on N-Tier platform.

• Key Words : Infographics, MVC Design Patterns, Convergence Presentation Tier, N-Tier Platform

1. 서론

급속도로 증가하고 있는 SNS, 웹로그, 소셜미디어, 이메일, 이미지, 동영상 등의 비정형 데이터 중심의 빅데이터 관심이 이제는 클라우드 컴퓨팅 환경의 시대를 예고하고 있다. 향후 웹 3.0은 인터넷 혁명의 파동에 대한 가설을 기반으로 N-스크린, 클라우드 컴퓨팅, 빅데이터, 디지털 콘텐츠, 스마트워크, 소셜 서비스 플랫폼, 스마트 디바이스, 모바일 웹앱, HTML5 등의 키워드를 중심으로 발전될 전망이다[1]. 이와 같은 플랫폼의 변화에 따라 IT 산업의 소프트웨어 분야에서도 배포 문제에 따른 소요 비용의 증가 때문에 웹 애플리케이션 기반으로 전환되고 있는 실정이다.

그러나 인터넷 기반의 웹 애플리케이션 시스템은 분산 컴퓨팅 시스템이지만 서버 측에 상당히 많은 컴퓨팅이 필요하게 되었고, 서비스가 다양화 될수록 복잡도가 크게 증가되고 있다. 또한 점차 다양한 상호 작용과 사용

*교신저자 : 이명호(mhlee@semyung.ac.kr)
접수일 2014년 9월 24일 수정일 2014년 11월 20일 게재확정일 2014년 12월 3일

자 인터페이스에서도 사용자의 요구가 점점 더 많아지고 있는 실정이다.

따라서 위와 같은 고객의 요구사항에 따라 웹 기반의 한계를 보완하고, 클라이언트-서버 환경의 장점을 살리면서 다양하고 복잡한 요구사항들을 수용하기 위하여 새로운 모델이 필요하게 되었다. 이러한 요구로 태어난 것이 바로 인포그래픽이다. 본 연구에서는 이러한 N-티어 플랫폼 기반에서 MVC(Model-View-Controller) 디자인 패턴을 활용한 N-스크린 융합 표현 계층의 인포그래픽스 설계를 제안하도록 한다.

2. 기존연구에 대한 고찰

2.1 인포그래픽

방대한 양의 정보들을 최소화하면서 원하는 정보를 제공할 수 있도록 콘텐츠의 최소화로 정보의 효율성을 갖고 광고하고자 하는 것이 인포그래픽이다[2]. 인포그래픽은 정보, 자료 또는 지식의 시각적 표현이다. 정보를 구체적, 표면적, 실용적으로 전달한다는 점에서 일반적인 그림이나 사진 등과는 구별된다. 복잡한 정보를 빠르고 명확하게 설명해야 하는 기호, 지도, 기술 문서 등에서 사용된다. 차트, 사실박스, 지도, 다이어그램, 흐름도, 로고, 달력, 일러스트레이션, 텔레비전 프로그램 편성표 등이 인포그래픽에 포함된다[3].

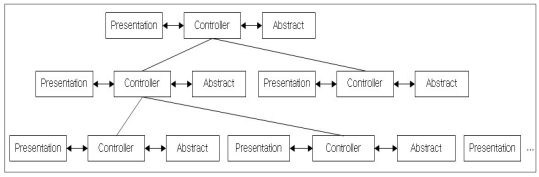
인포그래픽의 특징으로는 소비자가 직접 인포그래픽을 정적, 인터랙션, 모션 인포그래픽으로 제작할 수 있으며 차트나 지도 등으로 데이터를 기반으로 한 시각화 기능을 보여줄 수도 있다. 또한 폰트, 지도, 차트, 아이콘 등으로 특징에 맞는 콘텐츠로 차별화하여 주거나 국내외 알려진 소셜네트워크에 공유하는 특징도 있다.

2.2 MVC 디자인 패턴

MVC 디자인 패턴은 모델, 뷰, 컨트롤러의 세 가지 핵심적인 컴포넌트로 분리하여, 각각의 컴포넌트들이 고유의 업무를 처리하도록 하는 것이다[4]. 모델은 표현에 의존하지 않고 애플리케이션에서 필요로 하는 본질적인 모든 데이터와 비즈니스 로직을 캡슐화한다. 뷰는 시각적인 표현을 담당하는 것으로 사용자가 눈으로 볼 수 있고 사용자에 상호작용 하는 인터페이스이다. 컨트롤러는 사용자의 요청을 해석하고 요청을 수행하기 위해 필요에 따라 모델과 뷰 부분을 호출하여 모델과 뷰의 관계를 제어 해주는 역할을 담당한다. 이러한 MVC 디자인 패턴의 장점은 보여주는 것으로부터 데이터와 로직을 분리했기 때문에 같은 모델을 사용하는 다중 뷰를 두어 코드의 재사용성을 높일 수 있고, 구현, 테스트 및 유지보수가 쉬워지게 된다. 모델은 표현에 의존하지 않고 애플리케이션에서 필요로 하는 본질적인 모든 데이터와 비즈니스 로직을 캡슐화한다. 그리고 사용자가 뷰나 컨트롤러를 통해 데이터를 요청할 때, 뷰에 데이터를 주는 역할을 담당한다. 웹 애플리케이션에서는 이러한 방식의 통지가 불가능하거나 구현하기 어렵기 때문에 모델에 변경사항을 뷰로 밀어 넣는 푸시 방식 대신 클라이언트가 모델로부터 끌어오는 풀 방식을 사용한다[5].

2.3 PAC 모델

PAC(Presentation-Abstract-Controller)의 기본 구조는 표현-추상-컨트롤러에 의한 아키텍처 패턴이다. [Fig. 1]은 PAC의 계층 구조를 나타낸 것이다[6].



[Fig. 1] Hierarchical Structure of PAC

표현은 외부와 인터페이스를 담당하는 객체이고, 추상은 처리 대상인 모델이며, 컨트롤러는 처리를 제어하는 객체이다. MVC와 유사한 구조이지만 컨트롤러의 역할이 다르다. MVC의 컨트롤러는 마우스 이벤트나 키보드 입력 등의 데이터를 GUI 주변의 입력 장치를 추상화하기 위한 것이지만, PAC의 컨트롤러는 애플리케이션의 로직을 구현하기 위한 것이다. 따라서 MVC의 뷰와 컨트롤러는 PAC에서는 표현으로 구현된다. 최근에는 뷰와 컨트롤러는 위젯이라는 형으로 패키지된 부품으로 제공되는 경우가 많다.

PAC 패턴의 장점은 PAC의 계층을 정의한다는 점이다. 애플리케이션을 구성하는 각 층마다 PAC에 의한 구조가 있기 때문에 이것이 모여 애플리케이션 전체의 구조가 구성된다. 따라서 이 구조는 애플리케이션을 컴포넌트 기반으로 구축하는 경우에 아주 유용한 기능이다.

2.4 N-티어 플랫폼

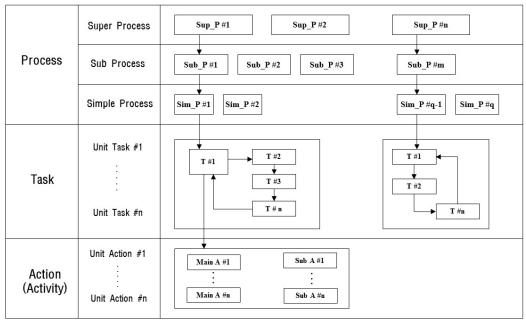
N-티어 플랫폼의 환경 구조는 대용량 분산 컴퓨팅 시스템 구조를 지향하고 있으며, 웹 브라우저가 운용되는 환경의 표현 티어, 웹 애플리케이션 및 비즈니스 로직이 동작되는 미들 티어 그리고 비즈니스 영역에서 발생하는 모든 정보를 보관 및 관리하는 데이터 티어인 EIS(Enterprise Information System)로 구성되어 있다. 표현 티어에는 최상위 레벨인 프로세스들을 관리하는 프로세스 핸들러와 프로세스를 구성하는 단위 작업들의 단위 작업 사용자 인터페이스 객체가 위치하게 설계한다. 그리고 각각의 단위 작업 사용자 인터페이스 객체에 상응하는 단위 작업 비즈니스 객체 컴포넌트가 미들 티어를 구성한다. 이 층의 단위 작업 컴포넌트는 EJB 컴포넌트로 구현되며, 기존과 마찬가지로 비즈니스 데이터를 다루고 처리하는 역할을 수행한다. 이 단위 작업 컴포넌트들은 실제 데이터 티어에 데이터 객체와 통신하게 된다. 지금까지 제안된 개발의 형태를 살펴보면, 각각의 단위 기능들을 나열하는 형태로 애플리케이션이 개발되었다. 지금도 대다수의 애플리케이션들은 이 같은 형태로 개발되고 있으며, 기반 기술이 OOP 기반이기는 하나 비즈니스와 프로세스의 변화에 유연하게 대처하는 데에는 한계를 가지고 있었다.

따라서 본 연구에서는 이 같은 한계를 극복하기 위해 비즈니스와 프로세스와 코드의 일치화를 목적으로 제안하는 표현 티어의 인포그래픽 사용자 인터페이스 영역 부분에 대한 설계 방법이다. 비즈니스의 프로세스를 조사하고, 다시 그 프로세스를 구성하는 작업들을 잘 도출하여 구성한다. 그리고 실제 이렇게 조사된 프로세스의 작업 단위별로 개발이 이루어지며, 최종적으로 이 작업들의 조합이 애플리케이션 시스템이 된다. 또한 일반적인 MVC 디자인 패턴을 그대로 사용한 것이 아니라 J2EE(Java EE) 환경에 맞는 구조로 변경하여 설계토록 한다[7].

3. 인포그래픽의 설계 구조

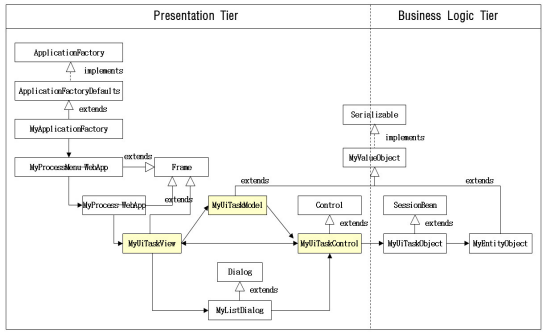
3.1 프레임워크의 설계

본 연구에서 제안한 인포그래픽을 위한 논리적인 구조는 [Fig. 2]와 같이 최상위 수준에는 비즈니스에 필요한 여러 가지의 프로세스가 존재한다. 각각의 프로세스는 그 프로세스를 구성하는 단위 업무인 단위 작업들을 가지고 있으며, 각 단위 작업에서는 정형화된 활동들이 수행된다.



[Fig. 2] Logical Architecture of Infographics

특정 프로세스의 특정 단위 작업을 추출하여 시스템을 구성했을 때, 인포그래픽스의 물리적인 구조는 [Fig. 3]과 같이 가장 상위에는 프로세스를 핸들링하는 프로세스 메뉴 웹앱과 프로세스 웹앱이 존재한다.



[Fig. 3] Physical Architecture of Infographics

프로세스 웹앱에 구성된 단위 작업이 표현 티어에 MVC 구조로 위치하게 된다. 중간 티어에는 기존의 설계 방법과 마찬가지로 엔티티 객체와 단위 작업 객체의 비즈니스 객체로 구성된다. 여기에 MVC 디자인 패턴 도입과 함께 새로운 가치 객체라는 것을 추가하였다. 이러한 단위 작업의 비즈니스 객체들은 EIS 티어의 ERP 데이터베이스 서버나 이 기종 데이터 서버와 통신을 하는 구조를 가지고 있다.

3.2 프로세스 핸들러

본 연구에서 제안한 인포그래픽 사용자 인터페이스를 위한 논리적인 구조는 최상위 수준에는 비즈니스에 필요한 여러 가지의 프로세스가 존재한다. 각각의 프로세스는 그 프로세스를 구성하는 단위 업무인 단위 작업들을 가지고 있으며, 각 단위 작업에서는 정형화된 활동들이 수행된다.

3.3 인포그래픽 객체의 설계

3.3.1 모델의 설계

패키지의 모델 클래스는 비즈니스 객체의 가치 객체의 값을 가지는 추상화 클래스로 컨트롤러의 요청인 처리 명령에 따라 가치 객체의 값을 받아 뷰에서 적절한 형식으로 값을 보여줄 수 있도록 값을 가지고 있다. 뷰의 변경된 데이터를 가지고 컨트롤러의 단위 할동에 따라 중간 티어의 가치 객체(Value Object : VO)에 데이터를 전송하는 역할을 수행한다. 이 추상 클래스의 속성은 Object 데이터형인 object와 selectedObject로 정의된다. object는 현재 작업 중인 작업의 속성값을 나타내고, selectedObject는 참조되는 작업의 속성값을 나타내기 위해 존재한다.

따라서 단위 작업의 인포그래픽 사용자 인터페이스에서는 Model 클래스를 상속 받는 단위 작업명에 Model 클래스를 첨부하여 생성하며, 비즈니스 로직을 수행하는 미들 티어의 가치 객체를 참조하도록 설계한다.

3.3.2 뷰와 ListDialog의 설계

일반적인 MVC 디자인 패턴에서는 모델의 값을 보여주는 다양한 형태의 뷰가 존재한다. 그러나 본 연구에서는 단위 작업당 오직 하나의 주(Main)가 되는 뷰를 가지도록 설계하며, 다양한 형식의 검색을 위해서는 하나의 ListDialog를 사용하도록 설계한다.

주문 Main 뷰와 주문 ListDialog를 생성하기 위해 슈퍼클래스가 되는 View 클래스는 showWindow (), addMainMenu(), actionPerformed(), closed Window(), getModel() 메서드를 구성하도록 설계한다. ViewExtends 클래스는 View 클래스를 상속받도록 한다.

ListDialog 클래스는 showWindow(), addDefault Components(), setViewFor(), actionPerformed(), closedWindow(), getModel() 메서드가 존재하도록 설계한다. ListDialogExtends 클래스는 ListDialog 클래스를 상속하도록 한다.

View 클래스는 눈에 보이는 뷰를 대표하는 클래스이다. 이 클래스는 Frame을 상속 받아 구현된다. 또한 사용자의 이벤트를 받기 위해 생성, 참조생성, 조회, 수정, 삭제, 선택, 저장, 종료와 같은 Main Menu Action을 정의하여 설계하도록 한다.

3.3.3 컨트롤러의 설계

컨트롤러 클래스는 Model과 View, ListDialog 사이에서 모든 단위 활동들을 수행하는 클래스로 Model, View, ListDialog를 생성하는 메서드와 Model의 값을 가져오거나 반환하는 메서드, 단위 활동를 처리하는 메서드 등을 구현하고 있다.

컨트롤러 클래스에 설계된 메서드는 createModel(), createView(), create ListDialog(), createVObject(), getModel(), setModel(), makeView(), deActivity(), addSelectListener(), fireSelectEvent() 메서드가 있다. addSubActivity()와 selected() 메서드를 가지는 ControllerExtends 클래스는 Controller 클래스를 상속받도록 한다.

컨트롤러 클래스는 생성자에서 애플리케이션, 단위 활동, 서브 작업 여부, 모델, 뷰, ListDialog, ValueObject, 단위 작업명의 정보를 받아 각 객체를 생성하며 Main 활동에 대한 처리 로직을 구현한다. 그 밖에 컨트롤러에서는 단위 작업과 단위 활동에 대한 기록을 하고, 예측하지 못한 시스템 장애 발생 시 복구를 할 수 있도록 객체를 직렬화하여 데이터베이스에 저장하는 역할도 수행한다. 각 단위 작업 명을 첨부한 컨트롤러 클래스는 컨트롤러 클래스를 상속받아 구현한다. 해당 단위 작업의 모델, 뷰, ListDialog를 생성해야 하기 때문에 멤버변수로 자신의 해당 클래스들을 선언하고 생성자의 매개변수로 사용한다. 이 때 생성자의 매개변수로 단위 활동 값과 서브 작업의 여부를 나타내는 값을 지정한다. 기본적으로 컨트롤러 클래스는 MainActivity에 대한 처리를 정의하고 있으며 각 단위 작업의 단위 작업명에 컨트롤러를 첨부한 것은 SubActivity에 대한 사항을 doSubActivity() 메서드에 구현하면 된다.

4. 결론

본 연구에서 제안한 인포그래픽 사용자 인터페이스의

설계는 개발 단위가 단위 작업이라는 단위로 세분화되기 때문에 개발 일정이나 역할 분담이 기존 개발 방법보다 수월해 질뿐만 아니라, 표준화된 단위 작업의 개발은 실제 개발 기간의 단축과 시스템 소프트웨어 품질의 향상을 가져온다. 단순한 객체의 재사용성을 넘어 단위 작업의 재사용을 할 수 있기 때문에, 대규모 프로젝트 시에 이미 개발된 단위 작업들을 이용하면 소프트웨어 생산성을 크게 향상시켜 줄 수 있다. 그리고 이 설계 방법은 비즈니스 프로세스와 시스템을 동기화시킨 형태이기 때문에 사용자들의 교육을 최소화 할 수 있고, 프로세스가 변경될 시에는 변경되는 부분의 단위 작업만 변경하면 되므로 빈번한 프로세스의 변화에 유연하게 대처할 수 있는 장점이 있다.

향후 상세 설계를 통한 실무 사례 연구와 모바일 웹 앱 기반의 e-커머스 및 m-커머스 구축에 대한 중간 티어의 인포그래픽 디자인 패턴 연구가 지속되어야 할 것이다.

References

[1] M.H. Lee and J.S. Han, “Comparison of Development Productivity of Spring 2.5 and EJB 3.0 with Lightweight Container Architecture,” The Society of Digital Policy & Management, Vol. 10, No. 3, pp. 137-142, 2012.

[2] M.Y. Kim, "Infographic," Gilbut, 2004.

[3] https://en.wikipedia.org/wiki/Infographic

[4] Krasner, G. E., Pope, S. T., "A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80," Journal of Object-Oriented Programming, Vol. 1, No. 3, 1988.

[5] M.H. Lee, “A Design of N-Tiers Platform for Building Enterprise Framework with Development Productivity,” The Society of Digital Policy & Management, Vol. 11, No. 10, pp. 411-417, 2013.

[6] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M., Sommerlad, P., Stal, M, “Pattern-Oriented Software Architecture, Volume 1: A System of Patterns,” John Wiley&Sons, 1996.

[7] Deepak Alur, Dan Malks, John Crupi, "Core J2EE Patterns: Best Practices and Design Strategies," Sun Microsystems Press, 2003.

저자소개

이 명 호(Myeong-Ho Lee) **[종신회원]**

- 11984년 2월 : 아주대학교 산업공학과(공학사)
- 1986년 2월 : 아주대학교 대학원 산업공학과(공학석사)
- 2001년 2월 : 아주대학교 대학원 산업공학과(공학박사)
- 2002년 3월 ~ 현재 세명대학교 전자상거래학과 교수

<관심분야> : 물류정보시스템, WAS 프로그래밍, 모니터링 시스템