제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003. 5)

웹 애플리케이션의 특징과 개발 방법의 제안

윤 준수, 정 용주

단국대학교 전자컴퓨터학부

mirjunsu@cs.dankook.ac.kr, chungy@cs.dankook.ac.kr

The Characteristics of Web Applications and the Proposal of Their Development Methodology

Junsu Yoon, Yongju Chung

Dept. of Computer Science, Dankook University

요 약

최근 몇 년간 웹 애플리케이션의 개발은 다른 애플리케이션의 개발보다 사회적으로 더욱 활발하였다. 이것은 우리 사회에 많이 보급되어 손쉽게 접근할 수 있는 인터넷과 또 그 인터넷 안에 있는 웹의 편리함에 그 이유가 있을 것이다. 그러나 이러한 활발한 웹 애플리케이션 개발에 비하여 웹 애플리케이션의 개발 방법 자체에 대한 연구는 거의 전무한 상태이다. 따라서 대학 교육에서도 그 일정한 지침이 없는 상태이다. 이러한 개발 방법에 대한 미진한 연구는 아마도 다른 애플리케이션에 비하여 웹 애플리케이션의 개발이 시작된 지 오래되지 않은 원인도 있겠지만 또 웹 애플리케이션이 가지는 특성에도 기인할 것이다. 즉, 웹은 보통 MVC(Model-View-Controller) 모델의 뷰(view)가 중심이 되고 컨트롤 부분이나 모델 부분은 상대적으로 적기 때문이다. 그래서 그 구현이 다른 일반 애플리케이션의 구현보다 비교적 개념이 쉽고 간단하다. 그러나 그 구현이 비교적 간단하다 하더라도 정형화된 개발 모델이 있으면 애플리케이션의 생산성은 향상될 뿐만이 아니라 학교 교육에도 큰 도움이 될 것이다. 즉 학생들에게 정형화 된 모델을 교육하는 경우 그것은 학생들의 웹 애플리케이션 개발에 큰 도움이 되는 것이다. 실제로 개발 모델을 교육시키고 프로젝트를 하는 경우와 그렇지 않은 경우에 그 완성도에서 크나큰 차이를 보였다. 정형화된 모델을 교육받은 학생들은 그렇지 않은 학생들에 비하여 프로젝트의 완성도도 높을 뿐만 아니라 그 구성이나 일관성에서 훨씬 좋은 결과를 보였다. 본 논문에서는 이러한 웹 애플리케이션을 위한 개발 방법을 제시하고 그에 앞서 클라이언트-서버 모델이 웹이라는 환경에서 가질 수 있는 특성과 작업에 대하여 분석한다. 그리고 본 논문에서는 그 예를 JSP로 보여준다.

1. 서론

지난 몇 년간 웹 애플리케이션의 개발은 사회적으로 다른 여타 소프트웨어의 개발보다 월등하였다. 이것은 웹의 편리함과 또 그만큼 사회적 요구와 그 필요가 많았기 때문인데, 이러한 시스템들로는 각 종 온라인 시스템들, 각 종 사이버 시스템들 등이 대부분 모두 웹을 사용하여 개발된 것들이다. 예를 들면, 기본적으로 자료 탐색을 위한 웹 브라우저는 물론 온라인 주식 투자, 온라인 뱅킹(on-line banking), 온라인 철도 혹은 비행기표 예약 시스템, 대학의 온라인 수강신청, 온라인 상담 등 웹을 이용한 사례는 다른 소프트웨어의 개발보다 최근 몇 년간은 훨씬 활발하였다. 이것은 아마도 첫째 우리 사회의 발달된 IT 문화, 즉 높은 PC의 보급율과 손쉬운 인터넷의 접근 때문일 것이다. 즉 우리 사회에서 회사나 관공서는 물론 일반 가정에서도 PC는 많이 사용하고 있는 실정이고 또한 그러한 PC들은 대부분 인터넷에 연결이 가능하기 때문이다. 그리고 두 번째 중요한 이유는 웹의 편리성과 그 발전 때문일 것이다. 현재의 웹은 CGI 방식이 가능한 이 후로 초창기의 웹과 달리 서버 안의 일반 프로그램의 실행이 가능할 뿐만이 아니라 그 애플리케이션의 작성이 서버측 스크립트(server-side script) 언어를 사용하여 간편하여졌기 때문일 것이다. 즉 CGI 방식과 달리 스크립트 언어를 사용함으로 웹 애플리케이션 개발이 용이해졌기 때문이다. 또한 장바구니 기능이나 세션의 활용 등으로 그 편리성이 더하였기 때문일 것이다. 이런 이유들로 웹 애플리케이션은 다른 애플리케이션들과 달리 많이 개발이 되었다.

그러나 이러한 활발한 웹 애플리케이션 개발에도 불구하고 아직 체계화된 웹 애플리케이션 개발 방법에 대한 연구는 그 활발함에 비례하여 거의 전무한 상태이다. 이것은 웹 애플리케이션의 작성이 스크립트 언어를 사용하기 시작한 것이 얼마 되지 않아서이기도 하겠지만 또 다른 이유로는 다른 소프트웨어에 비하여 짧은 역사로 아직은 사회적 요구 때문에 시스템 개발에만 너무 편중이 된 것도 있을 것이다. 그러나 무엇보다도 웹 애플리케이션의 개발 방법에 대한 연구가 미진한 것은 웹 애플리케이션의 인식에 기인하는 듯하다. 웹 애플리케이션은 과거 CGI 방식만을 의존할 때는 일반 소프트웨어 개발 방법이 그대로 적용되면 되었기 때문이다. 즉, CGI 방식은 그 실행이 서버에서 일반 프로그래밍 언어의 프로그램들로 작동되므로 일반적인 소프트웨어 개발 방법론이 적용되면 되었던 것이다. 이것은 MVC(Model-View-Controller) 모델의 뷰(view) 부분을 완전히 분리하는 방식이므로 당연한 결론이었다. 그러나 웹 애플리케이션의 개발이 최근에는 스크립트 언어 방식을 많이 사용하므로 큰 변화는 아니나 상황이 바뀌었다. 즉 스크립트 언어 방식은 MVC 모델 방식에서 뷰와 컨트롤을 완전히 분리하기도 하지만 많은 경우 뷰와 컨트롤이 한 페이지 안에 들어간다. 즉 뷰-컨트롤(view-control) 작성이 자주 생기는 것이다. 이것은 일반 소프트웨어 개발 방법을 그대로 적용하기는 무리가 있다. 왜냐하면 웹 애플리케이션은 아무래도 뷰가 중심이 되어 뷰 위주로 생각을 하여야 하기 때문이고 또 한 가지 이유는 일반 프로그램 작성과 달리 클라이언트-서버 방식에 의존하므로 그 조건을 역시 만족하여야 하기 때문이다. 즉 뷰 위주로 자주 컨트롤이 함께 하는 구조로 또한 그 처리는 요청을 먼저 하여 처리는 그 요청이 있은 후에 하여

야 하는 방식인 것이다. 이것은 CGI 방식에서 뷰를 완전히 분리하여 순수히 일반 소프트웨어 개발 방법을 적용하는 것과는 차이가 있다. 그래서 본 논문에서는 이러한 상황 아래에서의 웹 애플리케이션 개발 방법의 한 모델을 제시한다. 그러나 웹 애플리케이션 개발 방법의 하나를 제시하기는 하지만 일반적인 모든 웹 애플리케이션에 적용하기에는 아직 무리가 있다. 왜냐하면 웹 애플리케이션은 특성상 MVC의 모델(M) 부분이 간단할 수도 있지만 대단히 복잡할 수도 있기 때문이다. 즉, 보통의 웹사이트-회원 관리제 정도의 웹사이트-이라면 그 DB나 파일의 구성이 그리 복잡하지는 않을 것이다. 그러나 그 웹 애플리케이션이 관공서나 큰 기업의 인트넷과 연결된 경우라면 모델 부분이 복잡할 것이고 그에 따라 컨트롤 부분 역시 복잡한 양상을 나타낼 것이기 때문이다. 이런 경우에는 모델 부분과 컨트롤 부분의 연결을 위한 또 다른 구조가 나타나기에 또 다른 상황으로 바뀔 것이다. 그래서 본 논문에서는 보통의 웹사이트, 즉 일반적인 웹 애플리케이션으로 국한시키고 그에 대한 방법론을 제시한다.

본 논문에서의 제시한 방법이 비록 간단한 구조의 DB나 파일 구조의 애플리케이션을 위한 방법이기는 하지만 경우에 따라서는 여러 경우에 적용이 가능할 것으로 보인다. 즉 DB나 파일들이 비록 그들이, 즉 DB의 테이블이나 파일들이, 많다 하더라도 작성될 웹 애플리케이션의 웹 페이지들과 직접적으로 혹은 하나 혹은 둘 정도의 모듈과 연동하여 실행된다면 본 논문에서의 방법이 적용이 가능할 것으로 생각된다. 또한 과거 기업체나 관공서의 소프트웨어들이 초기 비 전산학 전공 개발인들에 의하여 개발되어 뒤에 결국 다시 재작성 되는 사례를 비추어 보면 이러한 정형화 된 개발 방법에 대한 연구는, 비록 간단하지만, 시급한 것 같다.

본 논문의 구성은 다음과 같다. 2절 "스크립트 언어와 웹 애플리케이션"에서는 스크립트 언어의 작동 환경과 그러한 스크립트 언어의 작성상의 특징 그리고 일반적인 웹 애플리케이션들이 되기 위한 작업의 특징들을 기술한다. 그리고 3절 "웹 개발 방법"에서는 웹 애플리케이션 개발을 위한 일반적인 방법을 제시하고 4절은 결론과 더 필요한 연구 과제에 대하여 기술한다.

2. 웹 애플리케이션과 스크립트 언어

2.1 웹 애플리케이션 작업들의 특징과 장단점

일반적으로 웹 애플리케이션들은(앞서도 기술하였듯이 여기서는 일반인들을 상대로 하는 회원 관리 정도의 웹사이트를 의미한다) 인트라넷과 같은 특수한 용도의 것들을 제외하고는 대체로 일반인들을 상대로 한다. 그리고 웹은 클라이언트-서버 모델을 따른다. 클라이언트-서버 모델은 클라이언트(고객)가 있고 그 클라이언트의 요청이 오면 서비스하는 서버가 있는 일의 모델을 의미한다. 웹이 클라이언트-서버 모델을 따르므로 웹에서는 항상 요청이 먼저 있다. 그리고 그 클라이언트로부터의 요청에 서버는 응답을 한다. (그래서 HTTP 프로토콜에도 두 종류, 즉 요청 HTTP와 응답 HTTP 두 종류가 있다.)

이러한 클라이언트-서버 모델을 따르는 웹 애플리케이션으로 작성할 수 있는 일의 특징으로는 여러 가지들이 있겠으나 중요한 특징 두 가지는 다음과 같다.

첫째, 최소한 어느 정도의 수의 클라이언트는 확보하고 있어야 한다. 즉 그 애플리케이션의 사용자 수가 어느 정도는 되어야 한다는 것인데 이 작업이 상업적인 사이트를 위한 것이라면 당연한 것이겠으나 비상업적이라 하여도 클라이언트의 수는 어느 정도 되어야 한다. 클라이언트 수가 적은데 굳이 웹 애플리케이션으로 작성하기도 그렇고 작성하였다 하여도 24시간 수행되는 웹 서버를 이용한다는 것은 자원의 낭비이다. 차라리 각 컴퓨터에 필요한 애플리케이션을 적재하여 사용하는 것이 더 나은 정책이다.

둘째, 클라이언트-서버 모델의 기본 특징 중의 하나인 서비스 시간이 짧아야 한다는 것은 웹에서도 동일하다. 식당 점원이 아니면 가게 점원이 하나의 고객을 위해서 하루 종일 서비스하지는 않는다. 웹에서도 서버에서의 작업 처리 시간이 짧아야 한다. 더구나 일반인들을 대상으로 하는 경우는 서버에서의 시간이 길면 왜면을 한다. 즉, 많은 루프가 있거나 아니면 많은 계산을 한다든지 하는 식으로 처리 시간이 길면 웹 애플리케이션으로 부적당하다.

이러한 두 가지 조건을 만족하는 작업이라면 웹 애플리케이션으로 적당하다. 즉 소수가 아닌 다수의 사람들이 공통으로 사용할 수 있는 작업이고 또 서버에서의 처리 시간이 짧다면 그것은 웹 애플리케이션으로 작성할 가치가 있다는 것이다. 그러나 인트라넷과 같은 행정 시스템 아니면 어떠한 상담 시스템과 같은 것들은 여기서 예외로 하여야 한다. 왜냐하면 그러한 애플리케이션의 클라이언트들은 그 애플리케이션을 필요에 의해서 사용하기 때문이다. 그들은 서버의 처리 시간이 길어도, 그들 자신이 필요하기에, 기다린다. 그러나 일반적인 웹사이트를 위한 작업이라면 두 번째 조건 역시 무시할 수 없다.

이러한 조건이 만족된 웹 애플리케이션이 장점들은 다음과 같다.

첫째, 클라이언트-서버 모델의 근본 특징으로 자원의 관리가 효율적이다. 여러 클라이언트들에게 필요한 파일이나 자원들이 하나의 서버에 저장이 되므로 그 관리가 수월하다. 여러 클라이언트들의 컴퓨터에 저장하는 경우보다 간편한 것이다. 자료 노출에 대해서도 필요한 자료만 클라이언트에게 보여주면 된다.

둘째, 컴퓨터의 처리 부담을 감소시킨다. 일의 능률이나 편리성 측면에서 보면 가장 이상적인 방식은 필요한 소프트웨어를 모든 클라이언트의 컴퓨터에 직접 저장하여 사용하는 것일 것이다. 그러나 항시 사용하지 않고 간단한 작업만

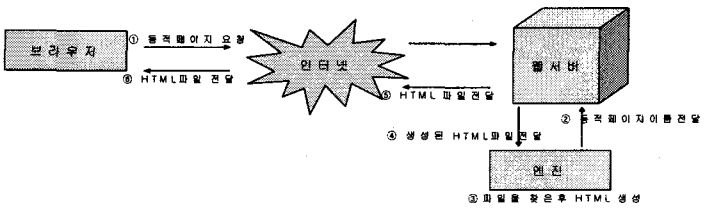

그림 1. 엔진을 사용하여 동적 HTML파일의 생성 과정. CGI 방식과의 차이는 엔진이라는 것이 더 필요하다.

을 하는 소프트웨어라면 오히려 컴퓨터에게 부담이 된다. 간단한 처리를 위하여 DBMS를 비롯한 여러 소프트웨어를 시동시켜야 되기 때문이다. 그러나 서버에게 24시간 작동시키고 그리고 클라이언트에게는 필요한 작업만 서버에게 지시하게 하면, 통신의 부담은 있지만 무시할만하고, 서버와 클라이언트의 작업 분담으로 클라이언트 측 컴퓨터의 부담이 주는 것이다.

셋째, 경제적인 측면에서도 이득이 된다. 서비스에 필요한 DBMS나 처리 소프트웨어들을 클라이언트들 모두에게 제공하면 그것은 클라이언트에게 큰 부담이 된다. 이러한 DBMS나 소프트웨어를 서버에 저장하고 모든 클라이언트들이 공통으로 그 부담을 나누면 그 부담은 훨씬 줄어든다.

넷째, 웹 애플리케이션은 인터넷에서 작동되므로 플랫폼에 무관하게 사용할 수 있다. 클라이언트는 요청을 HTTP로 하고 서버도 역시 HTTP로 하기 때문이다.

그러나 이러한 클라이언트-서버 모델의 웹 애플리케이션은 웹이라는 매체를 사용하므로 단점들도 생긴다.

첫째, 서버의 의존도가 크다. 서버가 멈추면 모든 클라이언트들을 위한 서비스가 멈추게 된다.

둘째, 클라이언트 수가 어느 정도 이상이 되고 그들이 공통으로 사용하기에 자원의 노출 위험이 있다.

셋째, 웹은 인터넷을 사용하기에 대상 클라이언트의 크기를 예측하기가 어렵다. 인터넷은 국내는 물론 전세계적으로도 사용이 가능하기 때문이다.

2.2 스크립트 언어 방식의 실행 환경과 특징

웹 애플리케이션은 초기에는 CGI 방식으로 작성하였다. 그러나 웹 애플리케이션 작성의 최근 추세가 서버측 스크립트(server-side script) 언어 방식을 많이 사용하므로 과거의 CGI 방식과는 다른 개념으로 생각하여야 한다. 서버측 스크립트 언어 방식으로 사용되는 대표적인 스크립트 언어들로는 ASP, JSP, PHP들을 들 수 있는데 이 세 언어 모두 문서 안에 직접 일반 프로그래밍 언어나 혹은 일반 프로그래밍 언어 수준의 언어를 사용하여 코딩 할 수 있다. 이것은 뷰와 컨트롤을 함께 작성할 수 있다는 것이다. 실제로 많은 웹 애플리케이션의 웹 페이지 안에서 뷰와 컨트롤이 함께 작성이 되고 디자인과 프로그래밍 작업을 분리하는데 중점을 둔 JSP 조차도 웹 페이지 안에서 프로그램 코드 부분을 완전히 분리하기는 어렵다는 것을 여러 책에서 지적하고 있다. 이것은 비록 웹에서 컨트롤이 하이퍼링크를 많이 사용하기는 하지만 스크립트 언어 방식에서는 컨트롤을 뷰와 함께 작성하는 것이 편리한 경우가 많다는 것이다. 즉, 과거 CGI 방식처럼 MVC의 뷰와 컨트롤-모델 부분을 완전히 분리하여 접근하던 방식과는 다른 것이다. 더구나 웹 애플리케이션이 가지는 특성, 엄밀한 의미에서는 제약 조건, 즉 먼저 요청이 있고 그에 따른 응답이 있는 상호 응답(interactive)식인 클라이언트-서버 처리 방식마저 고려를 하면 분명히 다른 개발 모델이 필요하다.

스크립트 언어 방식 이전의 CGI 방식은 직접 웹 서버에서 일반 프로그램을 실행하여 HTML 파일을 만든다. 이 CGI 방식은, 잘 알려져 있듯이, 표현이 어려운 단점이 있었다. 즉, 웹 브라우저에 보여줄 문서를 일반 프로그래밍 언어로 표현하여야 하는 어려움이 있었다. 이것은 문서를 작성하는데 문서 작성용 언어가 아닌 일반 프로그래밍 언어를 사용하는 것이다. 그래서 이러한 표현의 어려움을 없애기 위하여 개발된 것이 서버측 스크립트 언어들이다. 서버측 스크립트 언어로 많이 사용되는 것들이 ASP, JSP, PHP 들이다. 이들은 모두 HTML과 같이 나름대로의 태그를 사용하여 문서를 표현하는 것이다. 그리고 이들은 모두 인터프리트된다. 그래서 이들 모두 인터프리터나 혹은 각 페이지들의 관리나 수행을 위해서 각각 엔진들이 필요하다.(그림1) 예를 들면 ASP의 경우는 ASP 엔진, JSP의 경우는 JSP 엔진(혹은 콘테이너), PHP의 경우는 PHP 엔진(PHP 4.0 이후는 ZEND) 들이 각각 필요하다.

이 세 언어는 순수한 문서 작성용 언어인 HTML과 달리 작성할 문서 안에 일반 처리 코드를 넣을 수 있는 태그들이 있다. 즉, 예를 들면, ASP난 JSP의 경우는 <% ... %> 태그, PHP의 경우는 <? .... ?> 태그들을 사용하여 문서 모양이 아닌 처리 코드를 넣는 것이다. 이들은 이 태그 안에 어떠한 계산이나 혹은 함수의 호출 등 일반적인 작업을 할 수 있는 것이다. 즉 이것은 MVC 모델의 컨트롤 부분이 이 태그들 사이에 들어갈 수 있다는 것이다. 그리고 실제로 이 태그들은 웹 문서에서 많이 사용되는 태그의 하나이다.

3. 웹 애플리케이션의 개발 방법

웹을 사용하여 애플리케이션을 개발하려면 일반적인 시스템 개발과는 다른 고려할 점이 많이 있다. 웹 시스템은 기본적으로 클라이언트와 서버간에 상호 반응적으로 뷰를 중심으로, 필요한 경우 컨트롤을 뷰에 포함시켜야 하기 때문이다. 보통의 시스템은 한 컴퓨터 안에서 실행되고 그 안에서 결과가 나온다. 그러나 웹을 사용하는 경우는 클라이언트 쪽으로 전송되어 나타나는 것, 또 서버에서 작동되는 것, 그리고 상호간에 자료를 주고받으니 양쪽을 고려하여야 하기 때문이다.

다음은 웹 애플리케이션 개발을 위한 한 방법이다.

첫 번째 단계로 구축하려는 웹 시스템의 기능들을 나열한다. 예를 들어서 정치 선전을 위한 웹사이트라면 회원 관리가 필요할 것이고 또한 개인 의견을 보여 줄 전자 게시판이 필요할 것이다. 또 홈쇼핑의 경우도 회원 관리가 필요할 것이고 또 품목들을 나타낼 품목 전시 기능이 필요할 것이다. 그리고 품목 전시 기능에서 여러 품목들을 살 수 있는 장바구니 기능이 필요하면 그 기능 기술에 첨가하여 추가할 기능을 표시한다. 게시판의 경우도 응답, 재응답 등 이 필요하면 그러한 추가 기능을 표시한다.

두 번째 단계에서는 작성한 기능을 기반으로 필요한 페이지들을 적는다. 예를 들면, 우선은 홈페이지가 필요할 것이고, 회원 관리를 위해서는 로그인 페이지, 신입 회원 가입 페이지 등이 필요할 것이고 전자 게시판을 위해서는 그 전자 게시판을 위한 페이지가 필요할 것이다. 이때 각 페이지별로 코딩시 사용할 페이지 이름을 함께 적어준다. 그리고 모든 필요한 페이지들이 찾아지면 다음에는 그들간의 관계를 화살표나 기타의 방식으로 나타내어 전체 페이지간의 관계를 표시하는 일종의 전체 조감도를 추가로 작성한다.(그림 2)

세 번째 단계에서는 각 페이지 별로 세부적인 디자인 단계이다. 클라이언트에게 보여줄 웹 페이지를 디자인하는 것으로 탑다운(top-down) 방식으로 홈페이지부터 디자인하는 것이 좋다. 클라이언트는 홈페이지부터 방문을 하기 때문이다. 그래야 페이지들 간의 관계에서 빠트리는 일이 없다. 이 단계에서는 페이지 안의 각 요소들의 이름을 정하는 것이 좋다. 예를 들어서 브라우저 안에서 사람 이름을 입력할 텍스트 영역이면 그 부분을 "name"이라고 한다든지, 배경에 이미지가 들어간다면 그 이미지 이름을 역시 프로그램 코딩 작업시 사용할 이름으로 정한다.

네 번째 단계에서는 앞 단계까지 페이지들 간의 관계와

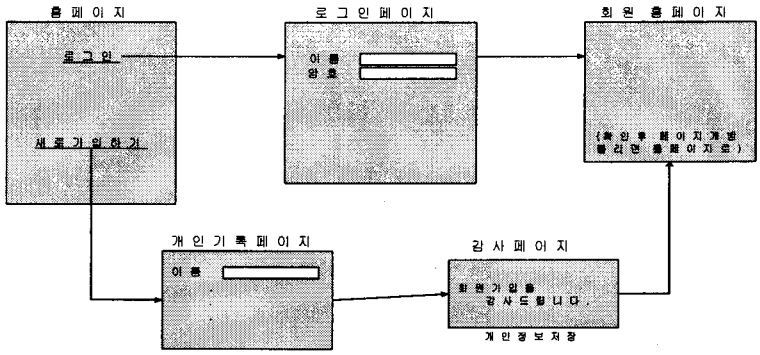

그림 2 페이지 별 관계도의 예

각 페이지의 디자인이 끝나면 페이지마다 하여야 할 추가적인 작업을 명시한다. 웹 페이지들 간에는 보통 하이퍼링크로 이루어져 있다. 그래서 사실 두 번째 단계와 세 번째 단계에서 작성된 것만으로도 작업 표시가 많이 표시된 상태이다. 예를 들어 어떤 두 페이지간에 화살표로 표시하였다면, 어떤 페이지에서 어떤 것을 클릭하면 또 다른 어떤 페이지가 나타난다는 자체가 페이지에서 하여야 할 작업이다. 그러나 경우에 따라서는 추가적으로 하여야 할 경우가 있다. 예를 들면 신입 회원의 경우, 신입 회원이 자신의 정보를 모두 입력하고 "전송" 버튼을 누르면 DB에 기록 후 다른 페이지를 보여주어야 한다. 여기에 DB에 기록하는 작업 등을 추가하는 것이다. 그러나 이때 주의 할 것은 웹 프로그래밍은 클라이언트-서버 방식이므로 추가적인 작업은 다음 페이지에서 이루어진다는 것이다. 웹 서버 측에서는 클라이언트의 요청이 있어야 작업을 하기 때문이다. 그래서 추가적인 작업은 다음 페이지에 기록하는 것이다. 그래서 이런 경우 처리하는 방식이 두 가지가 있다. 하나는 요청에 대한 처리를 위한 페이지 하나를 새로이 만드는 것이고 다른 방법은 처리 후 보여 줄 페이지에서 처리를 하는 방법이다. 앞의 방법은 처리할 루틴이 간단하지 않은 경우이고 뒤의 방법은 처리할 루틴이 간단한 경우이다. 앞의 방법을 사용하는 경우에는 새로 마련된 페이지의 끝에 JSP 경우는 <forward> 액션 태그를 사용하여 페이지 이전을 하고 뒤의 경우에는 임시 매개 변수를 주어 그 값의 세팅 여부로 뒷 페이지에서 처리 여부를 결정한다.

다섯째, 구현 단계로 각 페이지들을 구현한다. 각 페이지들을 구현할 때는 스크립트 언어에 초보인 경우는 스크립트 언어로 표시할 부분은 비워 놓고 먼저 순수하게 바탕이 될 HTML 파일을 먼저 만든다. 즉, 스크립트 언어 태그들이 들어갈 부분은 주석으로 적당한 표시와 함께 비워 놓는다. 그리고 그 HTML 파일로부터 비워둔 부분에 필요한 기능들을 스크립트 태그를 사용하여 코딩해 나간다.

마지막으로 전체 테스트 단계이다.

4. 결론

본 논문에서는 웹 애플리케이션 개발 방법의 하나를 제시하였다. 그에 앞서 웹 애플리케이션으로 구현될 수 있는 작업의 특징들을 기술하였다. 그러나 여기서의 애플리케이션은 뷰 중심의 애플리케이션을 의미하였고 가령 MVC의 모델(M)과 컨트롤(C)이 큰 경우에는 뷰와 컨트롤의 관계가 복잡하지 않은 경우로 한정하였다.(실제로 많은 웹 애플리케이션들이 그와 같은 구조이다.) 이러한 웹 애플리케이션은 일반 소프트웨어 개발 방법보다는 비교적 간단하다. 왜냐하면 뷰가 중심이고 그 뷰를 기준으로 애플리케이션들을 구현하면 되기 때문이다. 그러나 이 개발 방법이 비록 간단한 개념이나 대학 강의 시간에서는 큰 효과를 보았다. 즉, 이 개발 방법을 강의한 클래스와 그렇지 않은 클래스에서 많은 차이를 보였다. 이 이유는 아마도 학생들이 뷰와 컨트롤을 혼합하는 프로그램 개발이 처음이기에 전체적인 구성 제작에 혼란이 있기 때문인 것 같았다. 실험은 두 개의 클래스에는 이 개발 방법을 강의를 하고 두 개의 클래스에는 이 방법의 강의 없이 프로젝트를 주었다. 프로젝트는 DB를 사용하는 회원 관리와 전자 게시판, 로그인 등이 있는 학부 학생 일인에게는 비교적 큰 프로젝트였는데 이 개발 방법을 강의하지 않은 클래스들에서는 프로젝트의 완성도가 모두 60% 정도였으나 그러나 이 개발 방법을 강의한 클래스에서는 두 클래스 모두 80%를 넘는 완성도를 보였다.(일부 모듈, 즉 DB의 코넥션 풀을 위한 모듈은 수업 시간에 제공하였다.) 더구나 프로젝트의 구성도 강의한 클래스에서는 훨씬 구조적이고 일관성 있게 구성을 하였으나 강의를 하지 않은 클래스에서는 일부 결과물들을 제외하고는 그 구성도가 훨씬 떨어지고 심지어 복잡한 양상으로 구성 파악이 어렵기까지 하였다. 결과물들의 수행 안정도도 강의를 한 클래스들에서는 훨씬 안정적으로 실행되었다.

참고 문헌

[1] Roger S. Pressman, "Software Engineering", 2nd ed., McGraw-Hill, 1987

[2] I. Sommerville, "Software Engineering", 5th ed., Addison-Wesley, 1998

[3] Lincoln D. Stein, "How to Set up and Maintain a World Wide Web Site", Addison-Wesley, 1995

[4] JavaServer Pages, Specfications 1.2, Sun-microsystems. 2001

[5] Hans Bergstein, "JavaServer Pages", O' Reilly, 2001

[6] GAl Shachor 외, "JSP Tag Libraries", Manning, 2001

[7] Neil Jenkins 외, "Client/Server Unleashed", SAM.net

[8] Rice Darnell," HTML 4 Unleashed", Sams.net, 1997

[9] 정 용주, "서블릿으로 설명하는 JSP" 생능 출판사, 2003

[10] 양 회석 외, "퍼펙트 JSP"

[11] 김 지훈, "JSP", 정보 게이트

[12] 권 경희, "웹 엔지니어링", 배움터, 2001

[13] http://htmlhelp.org/faq/cgifaq.1.html

[14]http://searchvb.techtarget.com/sDefinition/