DOI: 10.3745/KIPSTD.2010.17D.1.009

클라이언트 기반 매시업 페이지에서 다중 비동기 서비스 호출

이 은 정†

요 약

웹서비스의 매시업이 새로운 소프트웨어 개발방법으로 주목받고 있다. 본 논문에서는 하나의 페이지가 여러 서비스 메소드를 비동기 호출의 방식으로 인터페이스하는 클라이언트 매시업을 고려한다. 비동기 웹서비스 호출의 컬백이 사용자 인터페이스와 병행하여 수행되는 경우 컬백은 사용자 인터페이스의 연산들과 메모리와 화면을 공유한다. 또한 사용자가 서비스 요청의 회신이 도착하기 전에 다른 요청을 전송하는 다중 호출이 가능하다면 공유 자원에 대한 병행 처리의 문제는 더 복잡해진다.

이 문제를 해결하기 위한 본 논문의 기여는 다음과 같다. 첫째, RESTful 서비스의 매시업 페이지를 사용자 액션과 컬백으로 모델링하고, 매시업 페이지에서 다중 요청의 유형을 제시하였다. 둘째, 컬백과 사용자 액션의 충돌을 공유자원의 측면에서 정의하고 병행 가능한 컬백의 조건을 제시하였다. 셋째, 공유 자원에 대한 충돌이 가능한 컬백을 순차화하여 수행하는 방법을 제시하였다. 마지막으로 제안된 컬백의 병행가능성 검사와 순차화 방법을 XForms 언어에 적용하고 브라우저에서 구현하여 사용자 경험을 향상시킬 수 있음을 실험적으로 증명하였다.

키워드 : 웹서비스, 비동기서비스호출, REST, 컬백

Multiple Asynchronous Requests on a Client-based Mashup Page

Eunjung Lee†

ABSTRACT

Web service mashup bacomes one of the important web application development methods. This paper considers a client based mashup, where a page interfaces many service methods asynchronously. Browser systems execute callbacks when the corresponding reply arrives, possibly concurrent to user interface actions. In this case, callbacks and user interface actions share data memory and screen. Moreover, when the user is able to send another request before the previous ones have replied, the shared resource problem becomes more complicated.

In order to solve the multiple requests problem, our contributions are as follows. First, we modeled a mashup page with user actions and callbacks, and we presented several types of callbacks. Secondly, concurrency condition is defined between callbacks and user actions in terms of shared resources, and the test method is presented. Also, we proposed the serialization approach to guarantee the safe execution of callbacks. Finally, we applied the proposed concurrency condition on XForms language and extended the XForms browser to implement the proposed approach. The prototype implementation showed that the proposed approach helps enhancing user experience on mashup pages.

Keywords : Web Services, Asynchronous Service Calls, REST, Callbacks

1. 서 론

매시업은 웹상의 자원과 서비스를 재사용하여 새로운 기능과 서비스를 가지는 어플리케이션을 개발하는 방법을 지칭한다[1,2]. 공개 웹서비스의 활발한 보급과 함께 서비스를 연결한 매시업이 새로운 소프트웨어 개발 방법으로 주목받고 있다[3-5]. 특히 클라이언트에서 서비스를 조합하고 연결하는 클라이언트 기반 매시업이 가볍고 쉽게 개발 가능하며 사용자의 필요에 따라 맞춤형 서비스를 제공할 수 있어 향후 웹어플리케이션의 대표적인 형태가 될 것으로 기대된다. 매시업의 개발을 위한 방법론과 도구들이 활발하게 연구되고 있다[3, 6-8].

한편 REST는 웹서비스를 위한 가벼운 프로토콜로서

※ 본 연구는 경기도의 경기도지역협력연구센터사업[컨텐츠융합소프트웨어연구센터]의 일환으로 수행하였음.
† 종신회원 : 경기대학교 컴퓨터과학과 부교수
논문접수 : 2009년 9월 1일
수 정 일 : 1차 2009년 12월 3일
심사완료 : 2009년 12월 17일

SOAP과 함께 웹서비스 제공을 위한 표준으로 인정받고 있다[9]. REST 기술과 Ajax 기술을 바탕으로 하는 비동기 웹서비스 통신이 일반적으로 매시업 페이지의 구축 방법으로 가장 많이 사용되고 있다[10, 11]. 비동기 웹서비스 통신은 사용자가 서버의 회신을 기다리지 않고 다른 작업을 할 수 있어 사용자의 대기 시간을 줄여줄 뿐 아니라 서버의 부담을 경감시키는데도 도움이 된다.

웹상에서 사용되고 있는 다양한 형태의 매시업을 분석, 분류하거나 개발 도구를 비교를 통한 연구가 활발하게 진행되고 있다[2, 12]. 또한 매시업되는 대상을 컴포넌트로 보고 조합에 의한 새로운 소프트웨어 개발 방법과 아키텍처를 제시하기도 한다[3, 8]. 클라이언트 매시업에 대한 연구는 데이터와 사용자 인터페이스 분야에서 활발하지만, 상대적으로 서비스 호출이나 페이지의 제어 부분은 아직 많이 다루어지지 않는 것으로 보인다. 본 논문에서는 비동기 서비스 호출의 컬백이 수행되는 방식을 살펴보고 컬백이 통신 스래드에 의해 실행되는 경우에 사용자 인터페이스 동작과 병행 수행됨으로 인해 발생하는 문제를 살펴본다. 또한 여러 개의 요청이 한꺼번에 전송되는 경우 그 회신의 도착 순서를 예측할 수 없어 컬백 간에 데이터 경쟁 관계가 발생할 수 있다.

비동기 웹서비스 요청의 컬백은 회신된 결과를 로컬 메모리에 저장하고 사용자에게 결과를 알려준다. 그 외에도 뷰를 수정하거나 포커스를 이동하는 등 사용자가 결과를 이용하여 추가적인 작업을 할 수 있도록 지원하는 사용자 인터페이스 연산을 수행하기도 한다. 그러므로 컬백은 데이터 및 사용자 인터페이스 자원을 사용자 액션과 공유하여 사용한다. 그로 인한 문제를 밝히기 위해 본 논문에서는 컬백의 병행 가능성을 정의하고 검사하는 방법을 제시한다. 제안된 검사 방법을 XForms 언어에 적용하여 병행 가능한 컬백을 찾는 알고리즘을 보였다.

충돌 가능한(병행 가능하지 않은) 컬백을 사용자가 요청할 때까지 지연시키는 순차화 방법을 제안하고, 제안된 방법을 검증하기 위하여 XForms 브라우저에 이 기능을 확장 구현한다. 확장된 시스템에서는 회신 메시지에 대해 충돌 가능성이 있는 컬백은 대기열에 추가하고 사용자가 요청할 때 수행시킨다.

이 논문은 다음과 같은 순서로 구성된다. 2장에서는 관련 연구를 살펴보고 3장에서는 비동기 요청의 컬백을 지원하는 매시업 페이지의 모델을 살펴본다. 4장에서는 컬백의 병행 가능성을 소개하고 이를 검사하는 방법을 소개하였다. 5장에서는 제안된 병행 가능성 검사 방법을 XForms 언어에 적용하였고 충돌 가능한 컬백의 순차화 방법을 XForms 언어와 브라우저 시스템에 구현한 결과를 살펴본다. 또한 사례 서비스를 통해 제안된 방법이 사용자에게 유용한 인터페이스를 제공함을 보였다. 6장에서는 결론을 맺는다.

2. 관련 연구

웹 2.0 환경의 변화와 매시업 활용의 급속한 증가로 인해 매시업은 최근 문헌 상에서 활발하게 연구되고 있다. 매시업에 대한 정의와 이론적인 개념의 정립을 시도하는 논문들이 많이 발표되었다[1, 5, 12, 14]. 이러한 시도를 통해 용어와 개념이 정립되면서 매시업은 소프트웨어 아키텍처의 새로운 형태로 자리잡게 되었다.

또한 인터넷 상에서 보급된 매시업 기술의 현황을 분석하고 분류하려는 노력이 많이 있다. Wong 등은 인터넷 상의 매시업의 형태를 다양하게 분류, 분석하였으며[8], Abeiteboul 등은 매시업에서 조합의 형태(glue pattern)을 제시하였다[6]. Yu 등은 매시업의 생태 시스템 연구를 통해 통계적으로 조사하였으며, 주요 서비스 제공자가 전체 매시업의 대부분을 차지하는 분포를 제시하였다[13]. 또한 매시업 개발 도구를 비교 평가한 연구도 있다[2, 12].

매시업에 대한 분류 방식은 다음과 같이 몇 가지로 나누어진다[12]. 매시업은 일어나는 장소에 따라 서버 기반 및 클라이언트 기반으로 나누어 볼 수 있고, 어플리케이션의 구성 요소에 따라 모델/뷰/컨트롤러의 수준으로 분류하기도 한다. 또한 여러 서비스 메소드를 모아서 지원하는 묶음 형식과 이전 메소드의 결과가 다음 메소드의 입력으로 사용되는 흐름 형식으로 분류하기도 한다.

특히 매시업 시스템이 데이터와 UI, 그리고 프로세스의 어느 부분에 초점을 맞추고 있느냐에 따라 매시업을 분류할 수 있다[12, 14]. 데이터 수준의 매시업 기술은 여러 서비스에서 제공되는 리소스를 연결하고 데이터 간의 구조적, 의미적 차이를 고려하여 통합된 데이터 모델을 제공하는 데 목적이 있다[12]. 한편 프로세스 수준의 매시업은 각 서비스가 제공하는 기능을 WS-BPML 같은 프로세스 정의를 통해 연결하는 것으로 흔히 SOA 방식의 엔터프라이즈 서비스 개발이나 워크플로우 시스템에서 많이 연구된다[5]. 한편 프리젠테이션 수준에서 개별 어플리케이션의 사용자 인터페이스를 하나의 페이지에서 통합하는 방법에 대한 연구도 있다[15].

매시업을 컴포넌트 단위의 요소를 조합해서 새로운 어플리케이션을 만드는 웹 아키텍춰로 보는 관점이 있다. 일부 논문에서는 매시업의 대상이 되는 기본 요소를 매시렛(mashlet)이라 하고 이것을 블랙박스 형태로 조립하는 매시업 소프트웨어 개발 방법을 제시하였다[6, 7]. 또한 Taivalsaari 등은 Mashware의 개념을 소개하며 매시업 기반의 어플리케이션을 RIA의 다음 단계 형태라고 했다[3]. Perez 등은 여러 개의 위젯을 포함하는 웹어플리케이션에서 통신 부담을 최소화하도록 페이지를 구성하는 알고리즘을 제안하였다[16].

그 이외에도 매시업 어플리케이션을 개발하기 위한 코드 생성 방법[7, 17], RIA의 접근에 기반한 모델 기반의 개발 방법론[18, 19], 효과적인 서비스 조합을 위한 시맨틱 웹 기반의 접근 방법[20, 21] 등 다양한 연구가 있다.

매시업 클라이언트는 웹페이지의 형태로 브라우저에서 실행되는데 대부분이 비동기 통신 방식을 지원한다[22, 23]. 대부분의 브라우저는 자바스크립트 API인 Ajax의 XmlHttpRequest 클래스를 통하여 비동기 통신을 지원한다 [10]. 이 클래스의 객체는 별도의 스래드로 수행되어 메시지를 전송한 후 응답

을 기다리다가 미리 등록된 컬백을 호출하여 회신 처리에 필요한 일을 수행한다. 한편 아파치 SOAP 프로젝트에서는 Axis 웹서비스 프레임워크를 개발하였는데, 여기서는 컬백 또는 폴링을 이용한 비동기 웹서비스 호출을 지원한다 [24]. 그러나 현재의 비동기 웹서비스 지원 방법에서는 클라이언트 시스템이 회신을 하나씩 처리하는 것을 가정하고 있다. 즉 여러 개의 요청이 전송되어 그 회신이 임의의 순서로 도착하는 상다가 처리하지 못한다. 또한 컬백에 의한 동작이 진행 중인 사용자의 작업과 충돌을 일으킬 가능성을 포함하고 있다. 이러한 문제는 통신이나 서버 측면에서 논의되었으나[25, 26] 매시업 클라이언트에서 비동기 웹서비스 통신과 컬백 처리 방식에 대해서 연구한 결과는 아직 발표된 바가 없다. 매시업 페이지에서 여러 메소드를 조합하면서 비동기적인 요청이 동시에 발생할 가능성은 페이지의 설계에 큰 영향을 미칠 수 있어 매시업의 연구에서 중요한 부분이라 할 수 있다.

3. 비동기 서비스 호출을 지원하는 매시업 페이지의 동작

3.1 매시업 페이지 모델

클라이언트 기반 매시업은 웹 페이지에서 서비스를 호출하고 회신을 처리하며, 개별 서비스의 데이터와 사용자 인터페이스를 조합하여 제공한다 일반적으로 웹페이지로 개발되는 클라이언트 기반의 매시업은 서버의 개발이 필요하지 않으므로 가볍고 비교적 쉽게 개발가능하며 사용자가 필요한 기능을 모아 개별화된 페이지를 구성할 수 있다는 장점을 가진다.

매시업 대상을 서비스 메소드라고 볼 때 여러 메소드의 인터페이스를 제공하는 묶음 형식의 매시업 페이지를 고려해 보자. 이러한 페이지에서 메소드 간 결합의 정도에 따라 (그림 1)과 같이 페이지를 분류해 볼 수 있다. (가) 각 메소드마다 별도의 페이지가 존재했던 웹 2.0 이전의 형태에서 (나) 초기적인 매시업 형태인 느슨한 결합 유형의 페이지, 그리고 (다) 강한 결합에 의해 데이터나 사용자 인터페이스 수준에서 상호 결합된 형태의 페이지 유형이다. 강한 결합 페이지 개발을 위한 대상 또는 조합 방법 등이 많이 연구되고 있으나 아직 실용화되기 어렵고 코드 복잡도나 개발 비용 등으로 인해 실제 인터넷 환경에서는 느슨한 결합의 유형이 대부분이다. 클라이언트 매시업을 위해 제공되는 개발 도구들도 느슨한 결합의 형태를 주 대상으로 한다 [12].

한편 웹서비스 프로토콜은 SOAP, REST, XML-RPC 등 다양한데, 이중에서 REST 프로토콜은 SOAP 기반의 프로토콜에 비해 가볍고 잘 정의된 구조를 가져 성능 및 개발 생산성이 높다. 이러한 이유로 인터넷 상에서 가장 많은 서비스에서 채택되고 있으며, Open API라고 불리는 웹서비스 분야의 실질적 표준으로 자리잡았다[11]. REST 방식은 미리 정의된 메소드 타입과 입력 및 결과 데이터 타입으로 인해 매시업 페이지의 개발에 중요한 지침을 제공한다. 본 논문에서는 REST 방식 메소드를 느슨한 형태로 결합한 매시업 페이지를 대상으로 한다.

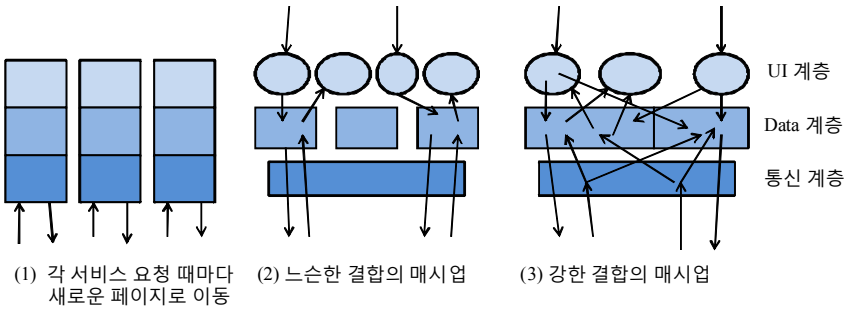
3.2 비동기 요청에 대한 컬백의 동작

클라이언트 매시업 페이지는 메소드 호출과 회신 결과 처리를 위한 인터페이스를 제공한다. 비동기 서비스 호출의 경우 페이지의 동작은 사용자가 입력 매개변수 데이터를 준비하고 서비스 요청을 전송하는 사용자 액션 부분과 회신 결과를 처리하는 컬백 부분으로 나누어 볼 수 있다.

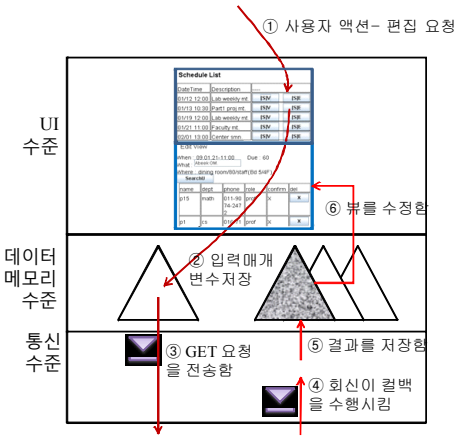
사용자 액션은 페이지 내의 여러 뷰 중 하나를 선택하고 지역 데이터를 확인하거나 원격 서비스 호출을 위한 입력 매개변수를 수정 또는 작성하여 전송하게 된다. 컬백은 회신된 결과 데이터가 있는 경우 그것을 지정된 로컬 메모리 영역에 저장하고 그 결과를 사용자가 확인할 수 있도록 화면의 일정 부분을 갱신하기도 한다. 그러므로 사용자 액션과 컬백은 클라이언트의 메모리와 화면을 공유하면서 동시에 수행된다. (그림 2)는 사용자 액션(하향 화살표)와 컬백(상향 화살표)에 의한 페이지의 동작을 보여주고 있다.

한편 페이지에서 수행되는 연산은 데이터를 읽거나 쓰는 데이터 연산과 사용자 인터페이스에 의해 화면을 일부 또는 전체를 수정하는 화면 연산이 있다. 또한 서비스 요청의 전송과 수신을 담당하는 통신 연산이 있다.

매시업 페이지에서 사용자는 서비스 요청을 전송한 후 회신을 기다릴 수도 있고 회신이 오기 전에 전송을 취소하거나 다른 작업을 진행할 수 있다. 다른 작업을 하면서 새로운 요청을 전송하는 것도 가능한데, 클라이언트 페이지에서 전송의 회신이 오기 전에 다른 요청을 전송하는 것을 *다중*



(그림 1) 웹서비스 메소드의 접근 및 매시업 방식에 따른 페이지 구조



(그림 2) 매시업 페이지에서 사용자 액션과 컬백의 연산 수행 과정

요청이라고 부른다. 회신에 의한 컬백은 사용자 인터페이스와는 별도의 스래드에서 실행된다고 가정할 수 있으며 사용자 인터페이스 스래드와 동시에 수행된다. 사용자 액션은 통신 연산, 데이터 연산, 화면 연산의 연속을, 컬백은 데이터 연산, 화면 연산을 트리거할 수 있다. 그러므로 사용자 액션과 컬백은 (1) 데이터 연산의 동시 수행에 의해 경쟁조건이 될 수 있고, (2) 사용자의 화면 작업과 컬백의 화면 연산이 충돌을 일으킬 수 있다.

이러한 충돌 가능성 때문에 상용화된 페이지에서는 보통 다중 요청을 허용하지 않는다. 그러나 다중 연산의 허용이 꼭 필요한 경우가 있는데 예를 들면 서비스 회신이 매우 길어지거나 기타 사유로 지연되는 경우, 또는 비연결 환경에서 작업할 때 (여러 개 서비스 요청에 필요한 작업을 마치고 네트워크에 접속되면 일괄 처리하려는 경우) 등이 그러한 경우다. 그러므로 가능한 경우 다중 요청을 허용하는 것이 사용자 경험의 향상을 위해 도움이 될 수 있다.

어플리케이션의 특성과 목적에 따라 동기적으로 통신해야 하는 서비스 메소드가 있다. 예를 들면 예약이나 결재 등 트랜잭션 방식으로 실행되어야 하거나 문제가 발생했을 때 사용자가 즉시 대처할 수 있어야 하는 서비스들이 있다. 이런 경우는 회신이 도착하기 전에 사용자가 다른 요청을 보낼 수 없도록 막아야 한다.

4. 다중 요청 컬백의 병행가능성 검사 방법

서비스 메소드의 컬백이 다른 사용자 작업이나 컬백과 동시에 수행된다면 이들에 의해 트리거되는 데이터 연산 및 사용자 인터페이스 연산이 충돌을 일으킬 수 있다. 이 절에서는 컬백이 충돌 가능한가(또는 병행 가능한가) 여부를 공유 자원에 대한 연산을 토대로 검사하는 방법을 살펴본다. 또한 충돌이 가능한 경우 컬백을 순차화하는 방법을 제시한다.

매시업 페이지의 액션 처리 과정을 모델링하기 위하여 페이지를 뷰와 액션으로 다음과 같이 정의한다. 페이지 P는 뷰의 집합 V^P를 가지며 서비스 인터페이스를 위해 메소드들이 공유하는 메모리 집합 M^P을 유지한다. 또한 이 페이지에서 발생하는 액션의 집합을 A라 하고, 페이지 P에서 사용되는 연산의 집합을 O^P라 하자. 그러므로 매시업 페이지는 $P = (V^P, M^P, A^P, O^P)$로 나타낼 수 있다. 이하에서는 문맥 상 분명할 경우 P는 생략한다.

사용자 액션과 컬백을 각각 A_{user}와 $A_{callback}$으로 표시하며 $A = A_{user} \cup A_{callback}$, $A_{user} \cap A^P_{callback} = \varnothing$이다. 사용자 액션과 컬백은 페이지에서 연속된 연산을 실행시킨다. 먼저 개별 연산의 효과를 다음과 같이 메모리 및 화면 리소스에 대해 정의한다. 통신의 전송 및 수신 연산은 여기서는 다루지 않는다.

[정의 1] 매시업 페이지 P = (V, M, A, O)에 대해 P가 차지한 화면 영역을 R이라 하자. 주어진 연산 $o \in O$에 대해 연산의 효과 함수는 다음과 같이 정의된다.

i) $read(o) \subset M$는 연산 o가 읽은 메모리 영역이다.
ii) $write(o) \subset M$는 연산 o가 쓴 메모리 영역이다.
iii) $hold(o) \subset R$는 연산의 수행동안 차지하고 있는 화면 영역이다.
iv) $update(o) \subset R$는 연산 o가 수정하는 화면 영역을 나타낸다.

여기서 $data\_access(o) = read(o) \cup write(o)$, $ui_access(o) = hold(o) \cup update(o)$로 정의한다. 본 논문에서는 화면 영역 R을 메소드 간의 공유 자원으로 보고 연산이 접근하는 화면의 부분을 표시하는 update(o)와 hold(o) 함수를 정의하였다. hold(o)는 사용자가 작업을 하기 위해 바뀌지 않고 유지되어야 하는 화면 영역을 나타낸다. 예를 들면 사용자가 장기게임에서 다음 수를 생각하는 동안 현재의 화면 영역이 유지되어야 하는 것이 그러한 경우이다. 또한 update 함수는 해당 연산이 수정하는 화면 영역으로 예를 들어 다른 뷰로 포커스를 이동시키는 연산은 그 뷰가 차지한 화면 영역을 수정하게 된다.

연산의 효과 함수를 이용하여 두 연산이 경쟁관계가 될 조건을 다음과 같이 정의한다.

(1) $o_1, o_2 \in O$에 대해 데이터 경쟁 조건은 다음과 같다.

$data\_access(o_1) \cap write(o_2) \neq \varnothing$ 또는 $write(o_1) \cap data\_access(o_2) \neq \varnothing$,

(2) $o_1, o_2 \in O$에 대해 화면 경쟁 경쟁 조건은 다음과 같다.

$ui\_access(o_1) \cap update(o_2) \neq \varnothing$ 또는 $ui\_access(o_2) \cap update(o_1) \neq \varnothing$

[정의 2] 두 연산 o_1, o_2가 안전하게 동시에 수행될 수 있을 때 병행 가능하다고 하고 $o_1 \parallel o_2$로 표시한다.

[보조정리 1] 주어진 연산 o1, o2에 대해 o1과 o2가 데이터 경쟁 관계도 아니고 화면 경쟁 관계도 아니면 $o_1 \parallel o_2$이다.

다음으로 이벤트에 의한 액션의 병행가능성을 살펴본다.

사용자 이벤트와 컬백에 의해 발생되는 액션을 각각 a와 g라 할 때 이들을 연속된 연산으로 다음과 같이 표시할 수 있다.

$$\alpha = (a_1, a_2, ..., a_n),\ \gamma = (b_1, b_2, ..., b_m),$$

여기서 $a_i, b_j \in O$, $0 \le i \le n$, $0 \le j \le m$이다. 이 때 연산의 효과 함수 write를 액션의 연산열에 대해 확장하면, 위와 같은 a에 대해 $write(\alpha) = \cup_{0\le i\le n} write(a_i)$, 연산의 집합 $X \subset O$에 대해 $write(X) = \cup_{X\subset O} write(u)$라고 쓸 수 있다. 연산의 효과를 나타내는 read, hold, update 함수도 동일하게 확장된다. 또한 두 액션의 병행 가능성을 실행되는 연산들에 대해 다음과 같이 정의할 수 있다.

[정의 3] 사용자 액션 $\alpha = (a_1, a_2, ..., a_n)$와 컬백 $\gamma = (b_1, b_2, ..., b_m)$에 대해 모든 i, j에 대해 $a_i \parallel b_j$라면 병행가능하다고 하고 $\alpha \parallel \gamma$라고 표시한다.

사용자 액션과 컬백의 병행 가능성은 공유하는 데이터 영역과 화면 영역을 통해 검사할 수 있다. 주어진 컬백이 데이터를 저장하는 메모리 영역과 사용자 액션이 사용하는 메모리 영역이 겹치지 않는다면 데이터 경쟁 관계는 발생하지 않는다. 매시업 페이지에서 서비스 메소드의 입력 매개변수를 위한 메모리 영역과 결과 데이터를 저장하는 메모리 영역이 서로 겹치지 않게 설계할 수 있다. 그러면 사용자 액션은 입력 매개변수 영역에 기록하고 컬백은 결과 데이터 저장 영역에 기록하므로 서로 겹치지 않는 것이 보장된다.

한편 컬백 g가 화면을 전혀 수정하지 않거나 독점적으로 할당된 영역만을 수정하는 경우는 화면 자원에 대한 경쟁 관계가 발생하지 않는다. 컬백이 독점하는 화면 영역의 예로는 날씨나 로그인 상태 등을 표시하는 부분창의 경우를 들 수 있다. 이러한 영역은 보통 백그라운드로 수행되는 서비스 메소드 호출을 통해 수정되며, 사용자 액션에 의해 수정되지 않는다.

[보조정리 2] *γ가 컬백이고 U가 사용자 액션에 의해 수행되는 모든 연산의 집합이라고 하자. $write(U) \cap write(\gamma) = \varnothing$이고 $ui_access(U) \cap update(\gamma) = \varnothing$이면 γ는 병행가능하다.*

컬백은 회신이 도착한 시점에 수행되는데, 일반적으로 통신 스레드가 담당한다. 이때 컬백은 하나씩 처리된다고 가정하면 동시에 수행되지는 않지만, 컬백 액션 간의 순서에 따라 경쟁 상태는 가능하다. 두 개의 컬백이 순서에 상관없이 수행될 수 있을 때 이를 직렬 가능하다고 한다.

[보조정리 3] *두 컬백 γ_1과 γ_2에 대래 $update(\gamma_1) \cap update(\gamma_2) = \varnothing$이고 $write(\gamma_1) \cap write(\gamma_2) = \varnothing$이면 γ_1과 γ_2는 직렬 가능하다.*

[증명] 컬백의 연산이 결과 데이터를 저장하고 그것을 바탕으로 다른 데이터 영역을 수정하거나 화면 영역을 갱신하는데 이 때 $read(\gamma_1)$이나 $read(\gamma_2)$은 다른 컬백이 저장한 결과 데이터를 접근하지 않는다고 가정할 수 있다. 그러므로 write 만으로 데이터 경쟁 조건을 검사할 수 있다. 또한 컬백은 차지하고 있는 화면 영역이 없으므로 $hold(\gamma_1) = hold(\gamma_2) = \varnothing$이라고 가정할 수 있다. 그러므로 update 만으로 화면 경쟁 조건을 검사할 수 있다. □

이상의 논의로부터 매시업 페이지에서 어떤 컬백 액션의 수행이 다른 액션들과 충돌하지 않고 안전하게 병행수행될 수 있는 조건을 다음과 같이 기술할 수 있다.

[정리 1] *$P = (V, M, A, O)$가 클라이언트 매시업 페이지이고 g는 컬백이라 하자. γ가 안전하게 병행 수행되기 위해서는 다음의 두 조건을 만족하여야 한다.*

(i) γ가 모든 사용자 액션 α에 대해 병행가능하고

(ii) γ가 다른 컬백에 대해 순차화가능하다.

이상의 결과는 매시업 페이지의 설계에 유용한 패턴을 제공할 수 있다. 즉 매시업 페이지에서 지원할 웹서비스 메소드는 요구분석 단계에서 다음과 같은 세 가지로 나눌 수 있다.

(1) 동기적으로 수행되어야 하는 경우,
(2) 병행성이 보장되어야 하는 경우,
(3) 컬백이 다른 액션과 충돌 가능한 경우

이러한 메소드의 분류는 어플리케이션과 서비스의 성격에 따라 설계 이전 단계에서 결정된다. 그러면 각각의 요구조건에 맞게 데이터 및 화면 설계를 할 수 있다. 즉 (1)의 경우는 다중 요청을 허용하지 않아야 하고, (2)의 경우는 정리 1의 조건을 만족하도록 데이터 및 화면의 배타적으로 설계해야 한다. (3) 기타의 경우에 대해서는 동기적으로 수행시키거나 별도의 정책을 사용하여 안전한 호출을 보장해야 할 것이다.

충돌을 일으킬 수 있는 컬백의 다중 요청을 허용하기 위해 본 논문에서는 컬백의 순차화 방법을 제안한다. 사용자 액션은 순차화된다고 가정할 수 있다. 이를 이용하여 본 논문에서는 컬백이 충돌 가능성이 있는 경우 사용자의 요청이 있을 때까지 지연시키는 방법을 제안한다. 사용자가 명시적으로 요청할 때까지 컬백을 하나씩 수행시키면 순차화시키는 것이 가능하다. 그러면 충돌 문제를 해결할 수 있으며 컬백 간의 경쟁 상태도 해결될 수 있다.

5. 설계 및 구현

5.1 XForms 언어에서 컬백의 병행 가능성 검사

앞 장에서 살펴본 컬백의 병행가능 조건을 이용하면 다중 요청에 의해 병행하여 수행할 수 있는 컬백을 검사할 수 있다. 여기서는 정리 1의 조건에 의해 병행가능하지 않은 컬백은 충돌가능하다고 본다.

여기서는 병행가능한 컬백을 검사하는 제안된 방법을 XForms 언어에 대해 적용한다. XForms 언어는 XML 데이터 모델을 가지며 웹서비스 인터페이스를 가지는 페이지를 개발하는데 적합한 언어이다[27]. 또한 표준에서 통신 방식을 지정하지 않으므로 다중 비동기 통신을 지원할 수 있고 잘 정의된 액션 요소들을 이용하여 클라이언트의 동작이 정의된다.

```
<submit submission="edit">                                          ①
  <label>[S]Edit</label>
  <setvalue ref="instance('query')/param/edit_param/schedule/sid"
            value="./sid"/>
</submit>

<submission id="edit" type="epilog"
      method="get"                                                  ②
      ref="instance('query')/param/edit_param/schedule/sid"
      action="http://203.249.21.224:3000/edit_schedule.xml"
      replace="instance"                                            ③
      instance="query"
      target="/param/schedule_param/schedule">
  <setfocus control="Edit"/>                                        ④
</submission>
```

(그림 3) 편집 데이터 요청을 위한 XForms 코드 부분

클라이언트 시스템에서 컬백 중 사용자 작업과 병행 가능한 연산을 결정하려면 컬백에 의해 트리거되는 연산의 수행 결과를 분석하고, 사용자 액션에 의한 연산의 효과를 계산할 수 있어야 한다. 자바스크립트 같은 범용 개발 언어에서는 연산의 효과를 모두 계산하기 어려운 반면 XForms 언어는 데이터와 사용자 인터페이스의 효과가 미리 정의된 연산 요소들을 이용하여 클라이언트 시스템의 동작을 기술한다.

XForms 페이지에서 지역 데이터 모델 M은 여러 개의 XML 인스턴스를 가지는데 본 논문에서는 입력 매개변수 데이터를 위한 인스턴스 트리와 서비스 회신의 결과 데이터를 저장하는 인스턴스 트리는 구별되게 설계하였다. 입력 매개변수 데이터 영역과 회신 결과 데이터 영역을 각각 M_{input}과 M_{output}으로 표시하면 다음의 관계가 만족한다.

$$M = M_{input} \cup M_{output},\ M_{input} \cap M_{output} = \varnothing$$

또한 서로 다른 서비스 메소드의 결과 저장 영역은 다른 XML 인스턴스를 사용하도록 설계하였다.

XForms 페이지의 프리젠테이션은 group 요소로 정의되는 뷰가 화면 구성의 단위가 된다. 각 뷰는 화면에 보여질 렌더링 정보와 UI 컨트롤을 가진다. 여기서는 연산의 화면 효과를 간단히 하기 위해 각 뷰가 화면 영역 전체를 차지하고 뷰가 카드 형태로 쌓여 있는 사용자 인터페이스 환경을 가정한다. group 요소는 자식으로 사용자 컨트롤을 가진다.

사용자 컨트롤 중에서 사용자 액션을 제공하는 trigger와 submit 요소가 있다. trigger는 데이터 및 UI 연산들을 수행시키고 submit는 데이터 및 UI 연산 수행 후 마지막에 서비스 메소드를 요청하는 통신 연산을 수행시킨다. 수행될 연산은 trigger 및 submit의 자식 요소로 나타나며 XForms 연산 요소로 기술된다. 이 때 요청할 서비스 메소드를 표시하기 위해 submit 요소는 대응하는 submission 요소의 아이디를 속성으로 가진다. XForms 언어에서 서버와의 통신을 기술하는 submission 요소는 서비스 접속에 필요한 정보와 회신이 도착했을 때 결과 데이터를 저장할 곳, 그리고 컬백 시 수행될 연산을 기술한다. (그림 3)의 코드에서 submit 요소의 자식인 ①번의 setvalue 요소는 사용자 액션에 의해 수행되는 연산이다. 한편 submission 요소의 속성으로 기술된 ③의 결과 저장 연산과 뷰를 이동하는 ④번 연산은 회신이 도착한 후 수행되는 컬백 부분이다.

XForms 언어에서 연산은 미리 정의된 요소들로 기술된다. 각 연산은 데이터 효과나 화면 효과를 정의한 표준에 따라 동작하므로 다음 표와 같이 write 및 update 함수의 값을 구할 수 있다. 여기서 p는 XML 인스턴스에 대한 XPath 경로이다[28].

<표 1>의 연산 요소 효과 함수를 바탕으로 컬백의 병행 가능 조건을 찾는 알고리즘이 [알고리즘 1]과 같다. write_UI는 모든 사용자 인터페이스 연산이 기록하는 데이터 영역의 합이다. 이 알고리즘은 페이지를 읽으면서 정적으로 write_UI와 모든 submission에 대해 write(s)를 구할 수 있다. 그러므로 실행 시에는 각 submission에 대해 병행가능 여부가 결정되어 있다.

〈표 1〉 XForms 연산 요소의 write 및 update 범위

요소	연산의 매개변수	write(o)	update(o)
setvalue	p, value	p	
insert	p, tag, value	p	
delete	p	p	
setfocus	group id		해당 view영역
refresh			해당 화면 영역

[알고리즘 1]
입력 : 모든 submission 요소 s, XForms 페이지 코드 P
결과 : 병행가능한 submission 집합 CS
(1) body 요소에 대해
(1-1) write_UI = ∅,
(1-2) body의 후손인 모든 연산 u에 대해
(1-2-1) write_UI = write_UI U write(u)
(2) submission 요소 s에 대해
(2-1) write(s) = s의 target 속성 부분트리, update(s) = ∅,
(2-2) s의 모든 자식 연산 o에 대해
(2-2-1) write(s) = write(s) ∪ write(o),
(2-2-2) update(s) = update(s) ∪ update(o).
(2-3) write(s) ∩ write_UI = ∅이고 update(s) = ∅이면 s는 병행가능함.

5.2 브라우저에서 컬백 순차화 기능의 구현

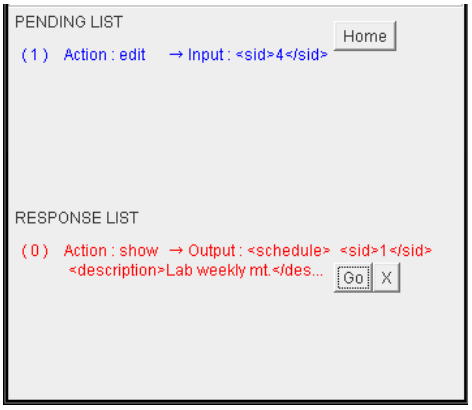
4장에서는 충돌 가능한 컬백의 안전한 수행을 보장하기 위해서 사용자가 요청할 때 수행하는 순차화 방식을 제안하였다. 순차화 방식은 브라우저에서 지원해야 하는 기능인데, 본 연구팀에서 기존에 개발한 XForms 브라우저에 이 기능을 확장하였다[29]. XForms 페이지에서 비동기 서비스 요청 중 컬백이 충돌 가능한 것을 구분하여 회신이 도착하면 별도의 메시지 큐에 넣어 두었다가 사용자의 요청이 있을 때 컬백을 수행시킨다.

브라우저 시스템의 비동기 통신 부분을 Java API 중 XmlHttpRequest 클래스를 확장하여 다음과 같이 구현하였다[30]. 사용자가 서비스 전송을 요청하면 전송 메시지를 만들어 확장 클래스 객체를 생성하고 스래드를 발생시켜 회신을 기다린다. 회신이 도착하면 다음과 같이 세 가지 경우로

나누어 수행한다.

(1) 해당 컬백이 병행가능하면 수행하고 종료함
(2) 해당 컬백이 데이터 연산만 병행가능하면 회신결과를 지정된 메모리 영역에 저장한다.
(3) 미수행된 컬백 연산이 남아있는 경우 메시지를 회신 메시지큐에 넣는다.
(4) 컬백 알림창의 갱신을 요청한다.

사용자 인터페이스에서는 컬백 알림창을 두어 회신메시지 큐의 대기중인 메시지를 사용자에게 보여주고 사용자가 특정 메시지를 선택하여 컬백을 요청하면 수행시키는 역할을 한다. 수행이 끝났거나, 사용자가 삭제를 요청한 경우 해당 메시지를 큐에서 삭제한다. (그림 4)는 컬백 알림창을 보여준다. 상단은 전송 후 회신을 기다리고 있는 pending list이고 하단은 회신 후 컬백의 수행을 기다리는 회신 메시지 큐의 리스트이다. 회신 큐의 각 메시지는 패널에서 두 개의 버튼을 가진다. 하나는 컬백의 수행을 시작하게 하는 버튼이고 하나는 해당 메시지를 큐에서 지우는 버튼이다.



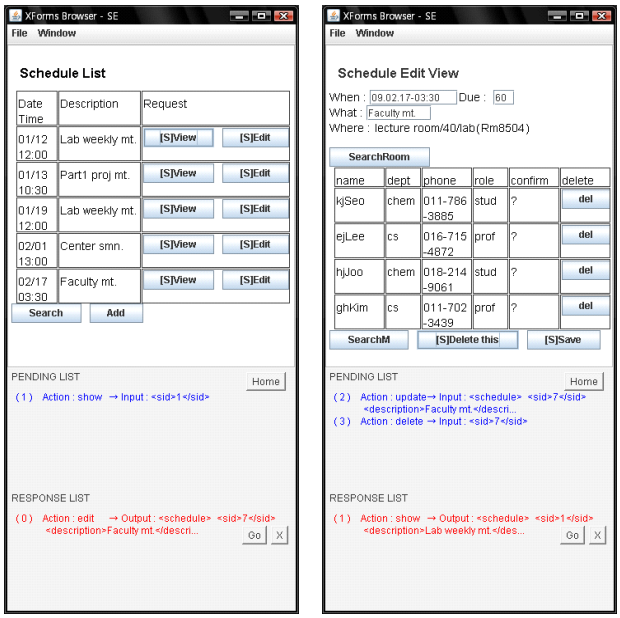
(그림 4) 컬백 알림 창

5.3 사례 시나리오

본 논문에서는 확장된 브라우저의 기능을 데모하기 위하여 일정 관리 서비스를 개발하였다. 개발된 웹서비스는 서버 데이터베이스의 일정 정보에 대해 REST 방식의 데이터 처리 메소드를 제공한다. 구현된 XForms 페이지는 (그림 5)와 같이 여러 개의 뷰로 구성된다. 화면 (a)는 일정 리스트 뷰로서 검색(Search)과 레코드 별 상세보기([S]View) 및 편집([S]Edit) 연산, 그리고 추가(Add) 연산을 제공한다. 화면 (b)는 일정의 편집 뷰인데, 여기서는 수정([S]Update)이나 삭제([S]Delete) 서비스의 호출이 제공된다. 버튼의 레이블에서 [S]는 서비스 메소드를 호출하는 submit 컨트롤임을 표시한다. 상세보기 및 편집 연산은 서버에 해당 세부 데이터를 요청하는 서비스 호출이다.

레일즈 프레임워크를 이용하여[31] 개발된 서버에서는 메시지를 일정기간 동안 지연시키도록 하여 다중 비동기 통신을 시뮬레이션했다. 뷰 하단의 컬백 알림창에는 회신 큐가 나타나는데, 요청 메시지의 상태가 pending 또는 responsed로 구별되어 표시된다. 회신 큐의 각 메시지는 [Go] 버튼을 가져서 사용자가 해당 메시지에 대한 컬백의 수행을 지시할 수 있다. 뷰의 아래쪽 패널은 회신 큐를 보여준다.

(그림 5)의 (a)는 'show' 요청이 대기 상태이고 'eidt' 요청은 응답 상태임을 보여준다. 여기에서 사용자가 'eidt' 메시지의 [Go] 버튼을 누르면 화면은 (b)의 뷰로 이동하여 사용자가 회신 데이터를 이용하여 편집할 수 있게 된다. 이 뷰에서 'update' 요청이 새로 보내져 대기 리스트에 추가되었다. 그 동안 'show' 요청에 대한 회신이 도착하여 그 메시지는 회신 리스트로 이동되었다.



(a) (b)

(그림 5) 일정 서비스 클라이언트의 화면

5. 결 론

본 논문에서는 클라이언트 측 매시업 페이지에서 발생하는 다중요청 및 컬백의 문제를 살펴보았다. 매시업 페이지에서 컬백의 수행이 메모리 및 화면을 사용자 액션과 공유하면서 동시성 문제가 발생함을 제시하였고 정형화된 방법으로 병행 가능성을 정의하였다. 또한 주어진 페이지에서 서비스 요청에 대해 컬백의 병행가능성을 검사하는 방법을 제시하였다.

제안된 병행 가능성 검사 방법을 XForms 언어에 적용하여 충돌 가능한 컬백을 분리하고 충돌 가능한 컬백을 순차화시키는 방법을 XForms 브라우저에 확장구현하였다. 개발된 프로토타입은 사용자가 회신 리스트를 통해 필요할 때 컬백을 수행시킬 수 있어 유용한 인터페이스를 제공한다.

컬백의 동시수행으로 인한 문제는 잘 알려져 있는 문제이다. 기존의 매시업 페이지는 데이터 및 화면 충돌을 가능성 때문에 일반적으로 다중 요청을 허용하지 않고 있다. 또한

사용자 인터페이스와 컬백의 동시 수행으로 인한 문제는 테스트를 통해 방지 또는 튜닝하는 방식으로 개발되어 많은 비용이 소모된다. 본 논문에서는 이 문제를 정형화된 방법으로 접근하여 병행 가능성을 제시하였고 그를 통해 매시업 페이지의 설계 단계에서 서비스 메소드들의 메모리 및 화면 사용에 활용할 수 있는 성질을 제시하였다. 또한 컬백의 순차화 방식을 충돌 가능성이 있는 경우에만 적용하여 사용자로 하여금 다중 요청의 컬백 실행을 제어할 수 있게 하므로서 유용한 인터페이스를 제공할 수 있을 것으로 기대된다

참 고 문 헌

[1] J. Yu, et al, "Understanding mashup development," IEEE Internet computing, Vol.12, Issue5, pp.44-52, 2008.

[2] A. Koschmider, et al., "Elucidating the Mashup Hype: Definition, Challenges, Methodical Guide and Tools for Mashups," MEM 2009, pp.1-9, 2009.

[3] Taivalsaari, "Mashware:the future of web applications is software," Sun Labs Technical Report TR-2009-181, Feb., 2009.

[4] S. Auer, et al., "Dbpedia: A nucleus for a web of open data," ISWS 2008.

[5] A. Jhingran, "Enterprise information mashups: integrating information, simply," pp.3-4, Proc. of VLDB, 2006.

[6] S.Abiteboul, et al. "Matchup: Autocompletion for mashups," ICDE'2009, 2009.

[7] R. Ennals, M.Garofalakis, "MashMaker: mashups for the masses," Proc. ICMD'2007, pp.1116-1118, 2007.

[8] J.Wong, J.Hong, "Marmite: Towards end-user programming for the web," CHI'07, 2007.

[9] R.T. Fielding, Architectural Styles and the Design of Network-Based Software Architectures, doctoral dissertation, Dept. of Computer Science, Univ. of Calif., Irvine, 2000.

[10] A. Holdener III, *Ajax, the definitive guide*, O'Reilly, 2008.

[11] Richardson, L., Ruby, S. (2007). *RESTful Web Services*, O'Reilly Media, Inc.

[12] G.Lorenzo, et al. "Data integration in mashups," ACM Sigmod Record, Vol.38, Issue1, pp.59-66, 2009.

[13] S.Yu, J.Woodard, "Innovation in the programmable web: characterizing the mashup ecosystem," LNCS 5472, pp.136-147, 2009.

[14] N.Zang, et al., "Mashups: who? what? why?" pp.3171-3176, CHI'08, 2008.

[15] Linaje, M; Preciado, J.C:; Sanchez-Figueroa, F.: A Method for Model-Based Design of Rich Internet Application Interactive User Interfaces. In 7th International Conference on Web Engineering(ICWE), 2007.

[16] S.Perez, O.Diaz, S.Melia, J.Gomez, "Facing interaction-rich RIAs: the orchestration model," In 8th International Conference on Web Engineering(ICWE), 2008.

[17] T.Fischer, et al., "Towards an Automatic Service Composition for Generation of User-Sensitive Mashups," ABIS'08, 2008.

[18] Bozzon, A.; Comai,,S.; Fraternali,P; Toffetti Carughi, G. "Conceptual modeling and code generation for rich internet applications," ICWE2006, Menlo Park, California, USA(2006).

[19] P. Dolog and J. Stage. "Designing interaction spaces for rich internet applications with UML," In 7th International Conference on Web Engineering(ICWE), 2007.

[20] 김진한, 이병정, "웹서비스와 OpenAPI를 사용한 SOA 기반 동적 서비스 합성 프레임워크," 정보과학회 논문지: 소프트웨어 및응용 제 36권 3호, pp.187-199, 2009.

[21] 진훈, 김인철, 반응형 계획에 기초한 자동화된 시맨틱 웹서비스의 조합, 한국정보처리학회논문지, 14권, 3호, pp.199-214, 2007. 6.

[22] Firefox browser, http://www.mozilla.com/firefox/.

[23] XForms processor from FormsPlayer, www.formsplayer.com.

[24] Apache group, "Axis web services," http://ws.apache.org/axis/.

[25] M.Brambilla, et al., "Managing asynchronous web services interactions," pp.80, ICWS'04, 2004.

[26] Juha Puustjarvi, "Concurrency control in web service orchestration," CIT'2008.

[27] XForms 1.1 W3C Candidate Recommandation, www.w3.org/TR/xforms11/.

[28] World-Wide Web Consortium standards including XML, XML Schema, and XPath.

[29] 이은정, 유가연, "XForms 지원 브라우저를 이용한 모바일 오픈 API 플랫폼 개발," 정보처리학회 학술발표대회논문집, 14집 2권, pp.444-447, 2007년 추계학술발표대회, 목포대학교, 2007.

[30] Java2Script, http://j2s.sourceforge.net/.

[31] Dave Thomas, David Heinemeier Hansson: *Agile Web Development with Rails*, Second Edition, Pragmatic Bookshelf, 2006.

이 은 정

e-mail : ejlee@kyonggi.ac.kr

1988년 서울대학교 계산통계학과(학사)

1990년 한국과학기술원 전자계산학과(공학석사)

1994년 한국과학기술원 전자계산학과(공학박사)

1994년~2000년 전자통신연구원 선임연구원

2001년~현 재 경기대학교 컴퓨터과학과 부교수

관심분야 : XML 처리 기술, 웹서비스