OpenAPI를 이용한 외부컨텐츠 기반 동영상 매쉬업 서비스의 구현*

이 동 균** · 권 준 희***

Implementation of External-Contents based Video Mashup Service using OpenAPI

Lee, Dong Kyun · Kwon, Joon Hee

〈Abstract〉

The existing mashup service providers have induced users to participate in their own made services and have grown their services by user participations and contributing contents. Recent services based on Web2.0 are called by 'Open (Content) Service' which means anyone can provide new services to the others without any of their own contents. But unfortunately the most of these days services are not opened. They just using mashups for increasing their service's traffics. We introduce the way to make a simple service to provide searching videos present it with extracted keyword from external content without our own contents at all. this show that how to combine the content with another contents or services.

Key Words : Mashup, Video Mashup, Web2.0, OpenAPI, REST

Ⅰ. 서론

최근 웹2.0[1]을 기반으로 하는 개방형 서비스가 늘어나고 있다. 개방형 서비스는 해당 서비스 제공자가 제공하는 서비스나 콘텐츠를 외부에 노출시킨다. 사용자는 이러한 공개된 서비스나 콘텐츠를 이용하여 새로운 서비스를 만들고 제공할 수 있다. 현재 많은 개방형 서비스가 OpenAPI[2]를 기반으로 제공된다. OpenAPI란 자사의 API를 웹 서비스를 통해 외부로 공개한 것을 말하고 매쉬업(Mash-up)[3]은 두 가지 이상의 서로 다른 자원을 섞어서 완전히 새로운 자원을 만드는 것을 의미한다.

이는 사용자가 원하는 서비스를 제공하기 위해서 둘 이상의 공개된 서비스를 이용하여 새로운 콘텐츠나 그것을 제공하는 서비스를 창조하는 것이다. 이러한 방식은 기존에 존재하고 있는 콘텐츠를 다른 방식으로 가공하거나, 또 다른 콘텐츠와 섞어서 만드는 것이기 때문에 새로운 콘텐츠를 빠르고 쉽게 개발 할 수 있다. 또 관계없는 두 개 이상의 서비스를 섞어 창발적(emergence)[4]으로 새로운 서비스를 만들 수 있기 때문에 큰 관심을 받고 있다. 그러나 공개된 서비스나 콘텐츠를 기반으로 새로운 서비스를 만들기 때문에 서비스가 종속적인 문제점과 개발 한계를 가지는 문제점이 있다. 이 뿐만 아니라 현재 대부분의 매쉬업은 명확한 수익모델이 없어 주로 개인

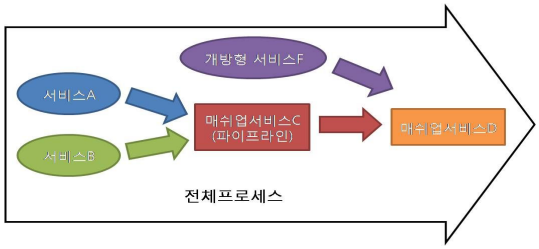
* 본 논문은 경기도 지역협력연구센터인 경기대학교 콘텐츠융합소프트웨어연구센터의 지원으로 수행한 연구 결과임.
** 경기대학교 컴퓨터과학과
*** 경기대학교 컴퓨터과학과(교신저자)

개발자 또는 비영리팀 주도로 이루어지고 있기 때문에 미완성이거나 프로토타입의 수준에서 그치는 경우도 많이 있고, 완성된 서비스라 하더라도 오랜 시간 안정적으로 서비스가 이루어지지 않는 경우가 많다. 무엇보다 가장 큰 문제는 개방형 서비스의 조합인, 각 매쉬업의 결과로 나온 서비스들이 더 이상 개방형 서비스의 형태를 지니지 않고 있다는 것이다. 즉, <그림 1>과 같은 양상을 보이게 되는데 이는 개방과 참여를 모토로 하는 웹2.0[1]의 정신에 어긋나 있다고 할 수 있다.

<그림 1> 현재 대부분의 매쉬업 서비스 형태

이러한 현상은 각 개발자의 개방형 서비스에 대한 인식부족과 기존 OpenAPI의 사용방법이었던 SOAP기반의 XML-RPC방식을 원인으로 들 수 있다. 이러한 개방형 서비스를 서비스 설계단계부터 유도하여 개선할 수 있는 방법으로 REST[5]모델이 있다. REST모델은 현재 데이터의 상태를 저장하지 않는 특성 때문에 외부컨텐츠를 사용하는 데 있어서 상당한 이점이 있다.

본 논문은 이러한 기존 매쉬업의 문제점을 해결하기 위해 REST모델을 기반으로 하는 매쉬업 서비스를 구현한다. 먼저 다른 서비스의 API 또는 데이터를 이용하여 매쉬업 서비스를 구현한다. 그리고 이렇게 구현된 서비스는 다시 자신의 결과를 REST기반의 API로 제공함으로써 다른 사용자들로 하여금 매쉬업된 데이터 또는 서비스를 재사용할 수 있도록 하여, 개방형 매쉬업 서비스를 제공한다. 이것은 웹2.0의 정신인 서비스와 데이터의 개방성뿐만 아니라 <그림 2>와 같이 프로세스와 프로세스의 사이에서 파이프라인의 역할을 하는 매쉬업 서비스를 구성하여 서비스단위의 모듈화 및 서비스의 재사용을 이룰 수 있다는 장점이 있다. 본 논문의 구성은 다음과 같다. 먼저 매쉬업의 개념을 알아보고, 개방형 매쉬업을 구성하기 위해 REST기반 OpenAPI에 대해서 알아본다. 그리고 이러한 오픈 서비스 기술들을 이용한 서비스들 중 구글 YouTube[7]와 그 OpenAPI를 소개한 뒤, 이를 이용하여 폐쇄형 영화관 예매 웹서비스인 CGV와 매쉬업하는 구현과정을 설명한다. 마지막으로 구현결과로 생긴 서비스인, 외부컨텐츠만을 이용하고 상태를 보존하지 않는 REST기반 서비스의 의의에 대해서 살펴본다.

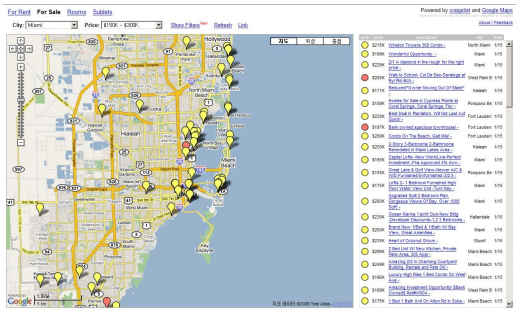


<그림 2> 개방형 매쉬업 서비스 형태

II. 관련연구

2.1 매쉬업(Mash-up)

매쉬업[3]이란 두 가지 이상의 서비스를 조합하여 새로운 서비스를 만들어 내는 것을 의미한다. 최근 매쉬업이 각광 받고 있는 이유 중 하나로 개발자나 기업이 다룰 수 없는 정보를 접근 가능케 해주고 있다는 점을 들 수 있다.



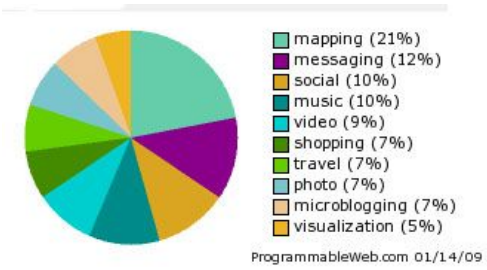
<그림 3> 하우징맵스(http://www.housingmaps.com/) 서비스

이러한 매쉬업의 대표적인 예로 <그림 3>의 하우징맵스[8]를 들 수 있다. 해당서비스는 Craigslist라는 부동산 가격 및 좌표정보제공 서비스를 구글의 개방형 위치정보 서비스인 구글맵스를 이용하여 매쉬업해 지도상에 표시해 줌으로써 사용자들에게 정보를 시각적이면서도 효과적으로 제공해주는 서비스이다. 기존에는 구글맵스에서 제공하는, GIS 등의 고도의 전문적인 지식과 장비가 필요한 서비스를 일반 사용자들이나 개발자들이 접하기가 쉽지 않았다. 하지만 구글맵스, 야후! 지도, 버추어어스 같은 GIS서비스 등이 개방형 서비스를 지원하면서, 이러한 서비스는 이제 일반인도 쉽게 접할 수 있는 현재와 같은 형태를 지니게 되었다.

2.2 동영상 관련 매쉬업 현황

본 논문에서는 세계적으로 가장 잘 알려진 동영상 서비스들 중 하나인 구글의 YouTube서비스의 OpenAPI를 이용했다. 본 절에서는 이러한 개방형 동영상 서비스를 이용한 매쉬업의 현황에 대해서 알아본다.

<그림 4>는 프로그래머블웹(ProgrammableWeb)[9]의 매쉬업 분포를 나타내고 있다.



<그림 4> ProgrammableWeb의 매쉬업 분포

<그림 4>에서 보이듯이 가장 많은 활용분야는 지도(mapping)분야이며 본 논문에서 다루고 있는 동영상(video)은 5위에 랭크되어 있다. 또한 <그림 4>에 보이듯 현재(2009. 1. 14) 프로그래머블웹에 등록된 매쉬업은 총 3647개이며 각 서비스는 하루에 3개꼴로 등록되어지고 있다. 프로그래머블웹은 매쉬업 포털 서비스로서 매쉬업에 사용되는 다양한 OpenAPI정보를 제공할 뿐만 아니라 매쉬업 카테고리를 제공하고 있다. 즉, 자신이 만든 매쉬업을 등록하고 홍보할 수도 있고 다른 사람이 만든 매쉬업을 검색할 수도 있는 서비스를 제공함으로써 매쉬업 관련 포털 서비스 중 가장 유명한 사이트가 되었다. 프로그래머블웹에서는 사용자들이 등록된 서비스에 점수(rate)를 매기고 이것을 이용해 인기도(popularity)로 정렬해 검색하는 서비스를 제공한다. 본 절에서는 가장 점수가 높은 몇 개의 서비스를 소개한다.

- Vdiddy

<그림 5>는 동영상 매쉬업 서비스인 Vdiddy의 화면을 보여준다. Vdiddy는 유튜브, 야후, 구글비디오, 마이스페이스, AOL 등의 개방형 동영상 서비스를 검색한다. 이러한 동영상 검색결과, 가장 많은 사용자가 시청한 비디오 10개를 뽑아서 화면에 제공한다. 또 비디오를 시청하면서 다른 사용자들과 비디오에 대한 의견을 나눌 수 있는 채팅로그 서비스도 함께 지원하고 있다.

<그림 5> 비디오 매쉬업 서비스 : Vdiddy

- Favreel.com

<그림 6>은 동영상 매쉬업 서비스인 Favreel.com의

화면을 보여준다. 이 서비스는 MTV API를 이용하여 MTV에서 제공하는 뮤직비디오를 플레이리스트로 만들어 다른 사용자들과 공유할 수 있는 환경을 제공한다. MTV는 음악전문 채널로써 현재 MTV API라는 OpenAPI를 제공하고 있다. 이는 OpenAPI 기술이 구글이나 야후!같은 인터넷 기술(IT)업체만의 고급기술이 아니라 단순히 콘텐츠기반의 사업을 하는 기업에서도 사용하는 범용적인 기술로 자리잡고 있다는 것을 방증한다.

<그림 6> 비디오 매쉬업 서비스 : Favreel

2.3 REST기반 OpenAPI

REST(REpresentational State Transfer)는 분산컴퓨팅 플랫폼 모델이며, 세계에서 가장 큰 분산 응용인 웹에서 사용하고 있는 웹 구조 스타일 모델이다. 현재의 웹의 기본 요소는 URI, HTTP, XML(HTML)이며, REST는 이러한 인터넷 표준만을 이용한다. REST에서 리소스의 식별은 URI로, 상태는 이를 표현된 문서(리소스)로써 HTTP를 통해 전달된다. 리소스의 내용은 XML로 기술하며, 리소스 탐색 및 참조에는 HTTP의 표준 메서드인 GET, PUT, POST, DELETE 등을 이용하는 것으로 분산 컴퓨팅을 모델링하고 있으며 이러한 REST권고안을 온전히 따르는 서비스를 RESTful한 서비스라고 한다[5,6].

REST는 분산 응용을 '상태 엔진'으로 본다. 클라이언트의 응용에 대한 요청은 응용의 현재 상태를 변화시키고, 이 변화된 상태는 클라이언트 요청에 대한 결과 정보를 포함하고 있다. 이 상태는 표현설명(Representation)을 갖는데 보통 한개 또는 여러 개의 문서로 표현되며(XML문서), 이 문서는 하이퍼텍스트로 다른 상태와의 연결고리(링크)를 갖는다. 즉, 응용의 상태들은 응용의 현재정보를 나타내며 하이퍼텍스트 문서들로 표현된다. 이 하이퍼텍스트 문서들은 리소스로써 유일무이한 URI를 갖는다. 링크는 새로운 상태로의 이동을 지시한다. 다음 상태로의 전환은 다음 상태 정보를 표현하는 리소스로의 이동을 가리키는 링크를 선택함으로써 이루어진다. 또한, REST에서 리소스의 전달은 그 표현인 문서 형태로 HTTP를 통해 이루어진다. 응용의 클라이언트는 응용에 서비스를 요청하고, 이 서비스의 요청 결과는 응용의 요청 결과 상태를 표현하는 문서로써 클라이언트에 전달된다. REST에서 리소스에 대한 오퍼레이션은 보편적이며 보다 단순한 GET(문서 가져오기), PUT(문서 쓰기), POST(문서에 추가 정보 전달하기), DELETE(문서 삭제), HEAD(문서의 헤더만을 가져오기) 등의 메서드만을 지원하며, 이러한 메서드는 HTTP에서 지원한다. 즉, RESTful한 서비스는 자원의 CRUD(Create, Retrieve, Update, Delete) 각 동작에 대해서 HTTP의 POST, GET, PUT, DELETE를 매칭해서 요청하는 것을 원칙으로 하고 있다[5,6].

하지만 현재 구글, 야후, 플리커, LastFM 등 세계에서 가장 영향력 있는 OpenAPI제공 업체들뿐만 아니라 국내의 네이버, 다음 등의 국내의 거대 OpenAPI제공 서비스들은, RESTful한 서비스를 제공하지 않고, XML을 이용한 데이터 교환과 URI를 이용한 상태의 표현 등의 RESTful서비스의 장점만을 차용한 REST기반 서비스를 제공하고 있다.

이절의 시작에서 설명했듯이 RESTful한 서비스란 자원의 CRUD(Create, Retrieve, Update, Delete) 요청에 대해 Http메소드(POST, PUT, GET, DELETE)를 각각 대응시키는 것을 원칙으로 한다. 그러나 REST기반 서비스는 POST와 GET 두가지만을 이용하여 자원에 대한 CRUD 요청을 모두 처리한다. OpenAPI서비스 제공자들이

REST기반 서비스를 채택한 이유는, 서비스 제공자들이 제공하는 서비스가 웹브라우저를 기반으로 하고 있다는 점과 웹 브라우저에서는 POST와 GET을 주로 이용하여 서비스하고 있는 것, 그리고 PUT, DELETE메소드를 추가로 지원하여 서비스를 구성할 경우 발생하는 서비스 복잡성의 상승을 고려하였을 때, RESTful이 REST기반에 비해 득보다 실이 많다고 판단되어진 것으로 보인다.

<표 1> RESTful 대 REST기반 URI

방식	요청 URI
RESTful	http://Service/get/name/fname/Dave/lname/Lee
	http://Service/get/name/Dave/Lee
REST기반	http://Service/get/name.xml?fname=Dave&lname=Lee

<표 1>은 RESTful한 URI와 REST기반 URI의 차이를 보여주고 있다. 이 표에서 보여주는 RESTful URI는 표준이 아니라 권고안이기 때문에 여러 가지의 표현 형태를 지닐 수 있으며 기본적으로 상태를 저장하지 않는 것을 원칙으로 하고 있기 때문에 URI에 데이터의 상태를 표현해야만 한다. 하지만 이로 인해 경우에 따라서 가독성이 저하될 수 있고 서비스의 설계자의 취향에 따라 각기 다른 체계의 URI를 사용하게 될 수도 있다. 즉, RESTful한 서비스의 단점인 기존의 프레임웍이나 개발방식을 수정해야 한다는 점과 모든 상태의 비지속성이, REST기반적인 서비스의 장점인 하위호환성과 서비스 사용과 설계의 용이함에 비해 이점이 부족하다. 이러한 문제로 인해 현재 대부분의 개방형 서비스의 OpenAPI는 RESTful URI요청보다 REST기반적인 URI요청을 선호하고 있다. 실제로 구글이나 야후, 다음, 네이버 등의 국내외 서비스들은 모두 REST기반적인 서비스를 제공하고 있다 [7,10,11,12].

2.4 기존 서비스의 문제점

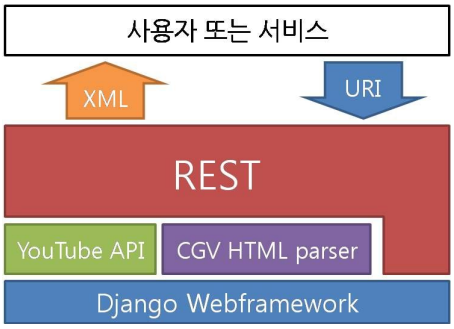
2.2절 에서 언급한 서비스들은 공개되어 있는 OpenAPI서비스들을 이용하여 새로운 콘텐츠를 제공할 수 있는 서비스를 만들었지만, 이것을 외부로 다시 노출시키는 어떠한 수단도 제공하고 있지 않다. 예를 들어 Favreel.com 서비스의 가장 큰 특징은 MTV API를 이용하여 가수이름으로 뮤직비디오를 찾는 기능과 뮤직비디오를 자신의 플레이리스트에 등록하여 다른 사용자와 공유할 수 있도록 하는 것이라고 할 수 있다. 이 때 MTV API의 결과를 외부에서도 이용할 수 있는 수단이 전혀 없다. 즉, Favreel.com 서비스는 <그림 1>의 형태를 가진 닫힌 서비스이다. 이는 사용자의 플레이리스트도 URI를 이용하여 접근은 가능하지만 XML 등의 데이터로 공개하지 않고 있으므로 사용자가 이것을 이용하려면 페이지 소스를 직접 파싱할 수 밖에 없으며, 이는 언제 변할지 모르는, View에 해당하는, HTML코드에 완전히 의존적인 형태의 서비스를 만들도록 함으로써 안정적인 서비스의 생산을 방해한다.

따라서 본 논문에서 구현한 서비스는, 서비스에서 이용한 모든 OpenAPI를 REST기반 URI를 제공함으로써 본 논문에서 구현한 서비스의 결과를 다른 서비스들에서 쉽게 재사용 가능하도록 한다. 이를 통해 <그림 2>와 같은 온전한 오픈서비스를 제공한다.

Ⅲ. 외부컨텐츠 기반 동영상 매쉬업 서비스의 구현

본 절에서 구현한 매쉬업 서비스는 영화 예매사이트인 CGV의 개봉예정작과 현재상영작에 대한 순위정보를 가져와 구글에서 제공하는 YouTube OpenAPI를 이용하여 매쉬업한다. 이 때, 개봉예정작은 개봉일순위, 현재상영작은 관람객순으로 정렬한다. 먼저 CGV홈페이지에서 현재 상영하고 있거나 상영예정중인 작품의 제목을 가져

와 YouTube API를 이용하여 키워드 검색한다. 검색결과 중 YouTube 검색엔진이 가장 연관이 높은 것으로 추천하는 동영상을 영화순위별로 정렬하여 화면에 표시해준다. 이 때 각 서비스 결과는 XML로 반환되며 REST기반의 URI를 통해 다른 서비스에서 쉽게 재이용할 수 있도록 한다.



<그림7> 본 논문에서 구현한 서비스 구조

<그림 7>은 본 논문에서 구현한 서비스의 구조를 나타낸다. Python을 기반으로 한 Django -Webframework를 사용하여 전체 웹서비스를 구성하였고, 비공개 서비스인 CGV에서 데이터를 가져오기 위해 HTML에서 데이터를 추출하는 parser를 만들었다. 외부와의 데이터교환은 REST를 기반으로 함으로써 URI를 통해 요청을 받고 XML로 데이터를 반환한다.

3.1 구글 YouTube OpenAPI를 이용한 매쉬업

구글 YouTube APIs가 기존 서비스들의 API와 다른 점은 Upload/Write API와 크롬리스(Chromeless) 플레이어 API에 있다. 기존 동영상 서비스들은 대부분 오픈되어 있지 않을뿐더러, 자신들이 제공하는 동영상 서비스를 이용하기 위해서는 서비스 이용 페이지에 접근하지 않으면 안되었다. 그리고 외부에서 동영상을 보기위해서는 해당 서비스가 제공하는 플레이어의 UI를 사용해야 하기 때문에 매쉬업서비스의 외관을 플레이어 UI에 맞추거나 혹은 디자인의 일관성을 버려야 했다.

그러나 YouTube API는 새로운 개념의 Upload/Write API를 지원하고 있다. 웹 사이트나 인터넷에 접속 가능한 소프트웨어를 개발하고 있는 사람은 누구라도 동영상을 YouTube에 직접 업로드할 수 있게 된다. 또 유저 코멘트나 레이팅, 동영상의 추천 기능이 개발자가 만든 매쉬업 서비스에서도 가능해진다. 이로써 콘텐츠를 공급받는 입장에서 콘텐츠의 재구성이나 파이프라이닝의 기능을 수행할 수 있어 다양한 형태의 서비스를 가능하게 한다. 또 크롬리스 플레이어 API는 동영상을 재생하는 플래시 플레이어의 개인화 및 제어도 가능하게 된다. 매쉬업 서비스의 디자인에 맞춰 화면에 표시할 플레이어의 색상이나 모양을 임의로 설정할 수 있다. 이는 서비스 개발자가 서비스 기능 뿐 아니라 디자인에 대해서도 독창성을 가질 수 있게 한다.

<그림 8> CGV의 현재개봉작 관람객순위

3.2 서비스의 구현결과

먼저 <그림 8>에서처럼 서비스 대상인 영화 예매서비스를 제공하는 CGV의 현재상영작 관람객순위를 가져온다. CGV는 개방형 서비스가 아니기 때문에 OpenAPI를 제공하지 않는다. 따라서 홈페이지의 소스코드로부터 해당 영화제목의 키워드를 얻는다.

<표 2>는 이렇게 얻어진 데이터를 본 절에서 구현한 매쉬업 서비스에서 재가공하여 RESTful URI로 외부에 노출시킨 예를 보인다. <표 2>의 URI로 요청된 정보는 <표 3>의 내용처럼 XML데이터로 구성되어 반환된다.

<표 2> 매쉬업 CGV영화순위 제공 요청 URI

데이터	요청 URI
상영예정	http://Service/get/plan/rank
상영 중	http://Service/get/playing/rank

<표 3> 매쉬업 영화순위 결과 XML

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp state="success">
  <movies category="plan">
    <movie>
      <rank> 순위 </rank>
      <title> 영화 제목 </title>
      <rate> 평점 </rate>
      <date> 개봉일자 </date>
    </movie>
  </movies>
</rsp>
```

해당 URI를 통해 얻은 영화제목 키워드를 통해 YouTube 키워드 검색을 한다. 서비스에서는 키워드 질의결과를 영화순위대로 정렬하여 화면에 보여준다. 이때 크롬리스 플레이어 API의 기능을 일부 이용하여 플레이어의 색상을 서비스 배경색상과 통일하여 매쉬업서비스 자체의 독창성을 부각시킨다.

키워드 검색 결과 또한 <표 4>와 같이 REST기반 URI로 구성해 본 매쉬업 서비스를 이용하여 다른 매쉬업 서비스를 구성할 수 있도록 한다. YouTube뒤 order는 검색결과를 영화순위로 정렬할 지 여부를 결정한다. 또 선택적 파라미터 값인 count는 반환될 video엘리먼트의 최댓값을 결정한다. 이렇게 정렬여부를 사용자가 URI로 선택할 수 있도록 함으로써 다른 매쉬업 서비스에서도 자신들이 다시 YouTube API에 대한 핸들러나 서비스를 구현하지 않고 본 논문에서 구현한 서비스를 이용할 수 있도록 하여 외부서비스의 프로세스에 파이프라인처럼 다른 서비스를 사용할 수 있도록 한다.

<표 4> 매쉬업 YouTube 키워드 검색 요청 URI

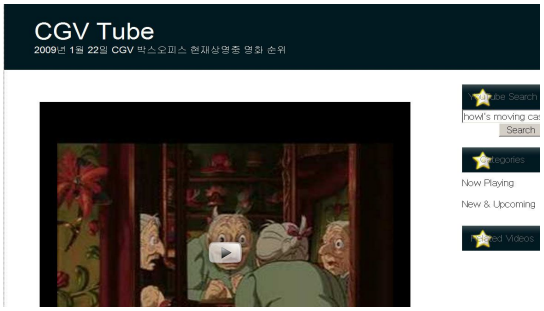
정렬방법	요청 URI
영화순위	http://Service/get/youtube/order/키워드[?count=개수]
유튜브 반환 결과	http://Service/get/youtube/키워드[?count=개수]

<표 5> 매쉬업 YouTube 키워드 검색 결과 XML

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp state="success">
  <videos keyword="키워드" order="1" count="1">
    <video>
      <title> 비디오 제목 </title>
      <rate> 평점 </rate>
      <url> 비디오 url </url>
    </video>
  </videos>
</rsp>
```

<그림 9>는 <표 4>에서 제공되는 URI를 서비스에 이용하여 특정 키워드로 YouTube 동영상을 검색한 결과화면이다. 키워드로 검색 시에는 YouTube 검색엔진에서 관련도가 높다고 추천하는 동영상을 순서대로 지정한 개수만큼 표시한다. 만약 사용자가 개수를 지정하지 않는다면 상위 1개만을 표시한다. <표 4>의 URI요청은 <표 5>와 같이 XML데이터로 변환되어 반환된다. <표 3>과 <표 5>의 XML데이터 반환방식은 REST모델의 표준방식

으로써 최근에는 자바스크립트 상에서 데이터 가공의 편이성을 고려하여 JSON[13]데이터 방식으로 반환하기도 한다. 본 논문에서는 REST모델을 이용하여 개방형 매쉬업을 구현하는 것이 목표이므로, 편이성보다 범용성을 중점으로 두어 XML형태의 데이터만을 반환한다.



<그림 9> "Howl's moving castle"로 키워드 검색화면

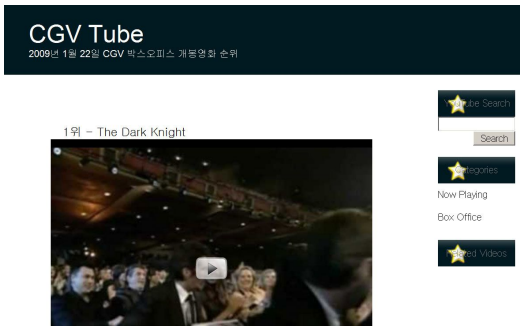
<그림 10>과 <그림 11>은 각각 위에서 설명한 내용을 바탕으로 실제 매쉬업 서비스를 구현한 화면이다. 먼저 CGV에서 가져온 영화순위별 키워드로 YouTube API를 통해 질의하여 결과를 가져온다. 이 때, 영화순위 데이터나 YouTube동영상 검색결과 데이터를 얻기 위해 다른 함수나 기능을 따로 구현하지 않고 <표 2>와 <표 3>의 URI를 통한 REST기반 웹서비스를 통해서 데이터를 가져와 사용한다. 이를 통해 구현된 서비스는 두 종류의 데이터를 조합하여, 영화순위별로 영화제목과 키워드 매칭되는 YouTube 동영상을 제공한다.

<그림 8>이 보이듯이 CGV는 영화관에서 운영하는 서비스로서 자사의 영화관에 대한 예매서비스만을 제공하는 폐쇄형 서비스이다. 이러한 폐쇄형 서비스의 특성상 사용자들은 현재 인기 있는 영화를 확인 위해서는 영화예매를 하지 않더라도 CGV서비스를 거쳐야만 한다. 그리고 이 영화와 관련된 동영상, 사진 등 다른 사용자들이 생성한 콘텐츠(UCC)들을 찾기 위해서는 다시 포털 사이트 등의 검색서비스로 이동해야 한다. 본 논문에서 구현한 서비스를 이용하면 <그림 10>과 <그림 11>처럼 영화순위를 한눈에 바로 확인 할 수 있고, YouTube API를 이용해 관련된 사용자 동영상도 볼 수 있다. 또한 크롬리스 플레이어 API를 이용하여 그 동영상과 관련된 동영상을 화면전환 없이 바로 볼 수도 있다. 즉, 외부콘텐츠만을 이용하여 CGV영화순위 확인 및 영화와 관련된 사용자 콘텐츠 검색을 하는 서비스를 구현했고, 영화순위와 사용자 콘텐츠 검색 기능을 REST기반으로 URI를 통해 외부로 노출시킴으로써 다른 매쉬업 서비스에서 쉽게 이용가능하게 하여 개방형 매쉬업 서비스를 구현하였다.

3.3 서비스의 활용방안

본 서비스는 2.4절에서 언급한 기존의 닫힌 서비스에 대한 문제점을 REST기반 URI를 이용하여 해결하였다.

사용자는 유튜브 동영상 검색결과를 REST기반 URI를 이용하여 XML데이터로 얻을 수 있다. 이 결과는 유튜브



<그림 10> 매쉬업 서비스에서 개봉 예정작 검색화면

<그림 11> 매쉬업 서비스에서 현재 상영작 검색화면

에서 반환하는 결과에서 항목을 단순화 된 것으로, 사용자는 본 서비스에서 제공하는 것 이상의 데이터가 필요하지 않다면 해당 API를 다시 구현할 필요가 없다. 또 기존에는 외부에서 접근할 수 없었던 CGV의 개봉예정작 순위와 현재상영작 순위에 접근하는 서비스를 구현하고 REST기반 URI를 이용하여 구현결과를 외부로 노출시켰다. 이를 통해 다른 영화순위 검색을 위한 서비스에서도 본 서비스에서 구현한 결과에 쉽게 접근할 수 있게 하여, 코드 및 서비스의 재사용성을 높일 수 있었다.

Ⅳ. 결론

웹2.0 이전의 서비스들은 대부분 폐쇄형 서비스였다. 그리고 웹2.0 초기의 서비스는 XML-RPC기반의 서비스로 제공되어 사용자들이 쉽게 접근하거나 사용하기가 힘들었다. 현재의 서비스는 진정한 의미의 개방형 서비스로 RESTful 또는 REST기반적인, REST기반 서비스를 표방한다. REST기반의 서비스는 사용자의 접근이 쉽고 직관적이며 상태를 저장하지 않는 특성 때문에 외부서비스로의 연결(Outgoing)이 자유롭다. 그러나 아직까지는 이러한 외부서비스와의 연결이 자유로운 특성을 잘 활용하지 못하고 있는 것이 현실이다.

두 가지 이상의 서비스를 조합해 새로운 서비스를 만드는 매쉬업 서비스들이 다양한 아이디어와 편리성, 유용성을 강조하며 나오지만 웹2.0의 근간이 되는 참여와 개방의 특성을 제대로 반영하지 않고 있다. 즉, 두 가지 이상의 개방형 서비스를 이용하여 하나의 폐쇄형 서비스를 구현해내는 회귀현상이 벌어지고 있는 것이다.

이에 본 논문에서는 개방형 서비스인 YouTube API와 폐쇄형 서비스인 CGV를 이용하여, REST모델 기반의 개방형 매쉬업 서비스를 구현하였다. 외부컨텐츠만을 이용해 서비스를 구성하여, 구현 측에서의 비용은 적게 들이고 YouTube의 방대한 비디오데이터를 영화데이터와 연동함으로써 저비용 고효율의 서비스를 구현했다.

서비스의 특성상 데이터의 상태를 서버에 저장할 필요가 없으므로 REST모델을 기반으로 서비스를 구성했다. 데이터의 상태가 URI를 통해 나타나는 것은 REST모델의 특성이다. 이를 통해 외부의 서비스에서 데이터의 상태에 구애받지 않고 자유롭게 구현된 매쉬업 서비스에서 제공하는 데이터를 이용할 수 있게 된다. 또한 데이터와 서비스의 매쉬업을 통해 가공된 데이터를 다시 REST방식의 URI로 외부에 노출함으로써 개방형 서비스를 제공한다. 즉, 구현된 매쉬업 서비스를 다른 서비스에서 이용 시, <그림 2>에서 보였던 전체 매쉬업 서비스 프로세스상의 파이프라이닝 효과를 얻을 수 있고, 이는 서비스의 재사용성과 모듈화를 높이는 효과를 얻을 수 있었다.

참고문헌

[1] O'Reilly,Tim "What Is Web2.0: Design Patterns and Business Models for the Next Generation of Software", www.oreilly.net/lpt/a/6228, 2005.

[2] OpenAPI, Wikipedia: The Free Encyclopedia, http://en.wikipedia.org/wiki/Open_API

[3] Mashup, Wikipedia: The Free Encyclopedia http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid), 2007.

[4] Emergence, Wikipedia: The Free Encyclopedia, http://en.wikipedia.org/wiki/Emergence

[5] Roger L. Costello, "Bulding Web Services the REST Way", http://www.xfront.com/REST-Web-Services.html

[6] 한창환・길준민・한연희, "REST웹 서비스를 이용한 데이터베이스 정보 획득", 한국정보기술학회논문지, 제6권 제4호, 2008, pp196~206.

[7] Google YouTube APIs and Tools, http://code.google.com/intl/ko-KR/apis/youtube/overview.html, 2009.

[8] HousingMaps, http://www.housingmaps.com
[9] ProgrammableWeb, http://www.programmableweb.com/mashups
[10] Yahoo! APIs, http://kr.open.gugi.yahoo.com/
[11] Daum APIs, http://dna.daum.net/apis/
[12] Naver APIs, http://dev.naver.com/openapi/
[13] JSON, http://www.json.org/

■ 저자소개 ■

이 동 균
Lee, Dong Kyun

2002년 3월~현재
경기대학교 전자계산학과 학사재학

관심분야 : 콘텐츠 컨버전스, 소프트웨어 공학, 애자일 방법론
E-mail : ldg55d@gmail.com

권 준 희
Kwon, Joon Hee

2003년 3월~현재
경기대학교 컴퓨터과학 전공 교수
2002년 2월 숙명여자대학교
컴퓨터과학과(이학박사)
1994년 2월 숙명여자대학교
전산학과(이학석사)
1992년 2월 숙명여자대학교 전산학과(이학사)

관심분야 : 유비쿼터스 컴퓨팅, 상황인식 컴퓨팅, 모바일 컴퓨팅, 데이터베이스
E-mail : kwonjh@kyonggi.ac.kr

논문접수일 : 2009년 1월 28일
수 정 일 : 2009년 2월 17일
게재확정일 : 2009년 2월 23일