Mashed Login Access Control System 설계 및 구현*

한재일, Nguyen Hong An, 임승용, 정기용, 김영만O
국민대학교 컴퓨터공학부
{jhan, hongan, chitos, admin, ymkim}@kookmin.ac.kr

Design and Implementation of Mashed Login Access Control System

Jaeil Han, Nguyen Hong An, Suengyoung Lim, Kiyong Jung, Youngman Kim
School of Computer Science, Kookmin University

요 약

현재 다양하고 전문적인 웹 서비스 들이 사용자를 위해 제공되고 있다. 이러한 서비스는 사용자의 인증을 거친 뒤 확인된 사용자에 한해 서비스를 제공하는 것이 보통이다. 이 서비스들은 각각의 서비스 별로 인증 시스템을 가지고 있기 때문에 사용자들은 매번 회원가입과을 해야하는 등 많은 불편함을 겪고 있다. 본 논문에서는 OpenID 프로토콜과 OAuth 프로토콜을 기반으로 다양한 인증 방식들을 래핑해 사용자가 기존에 보유하고 있는 ID를 이용해 인증할 수 있도록 통합된 인터페이스를 제공하는 인증 솔루션인 Mashed Login 접속 관리 시스템의 설계 및 구현에 대해 논한다.

1. 서 론

인터넷의 서비스들이 전문화됨에 따라 다양한 서비스를 제공하는 수많은 웹 사이트들이 생겨나 운영되고 있다. 사용자들은 자신의 취향에 맞거나 필요한 서비스를 제공하는 웹 사이트들을 선별하여 선택적으로 사용해야 하는데, 이런 서비스는 대부분 회원 가입을 요구한다. 때문에 사용자는 매 사이트별로 새로 ID를 발급받아야 한다.

이 때문에 사용자를 인증하는 여러 가지 방법들이 활발히 논의되고 있다. 사용자들이 여러 사이트에 매번 가입해 인증정보(ID와 Password)를 관리해야 하는 불편함을 해결하기 위한 OpenID 프로토콜이 대표적이다. 또한 서비스 제공자와 서드파티 어플리케이션 사이에서 사용자를 인증하고 개인정보를 보호하기 위한 OAuth 보안 프로토콜도 널리 쓰인다. 이 외에 각 웹 사이트에서 독자적으로 구축한 회원 시스템에 대한 API를 제공하는 경우도 있다.

본 논문에서는 이러한 대표적인 인증 시스템인 OpenID 솔루션과 OAuth 보안 프로토콜을 중심으로 그 외의 독립적인 인증 솔루션들을 래핑하여 하나의 인터페이스로 제공하여 다양한 인증 방식에 대응할 수 있는 Mashed Login 접속관리 시스템을 제안한다.

2장에서는 바탕이 되는 OpenID와 OAuth 데이터 엑세스 보안 프로토콜에 대한 전반적인 소개를, 3장에서는 본 논문에서 주제로 삼고 있는 Mashed Login 접속관리 시스템의 설계와 구현에 대한 소개를 하겠다.

*본 연구는 한국콘텐츠 진흥원의 2009년도 2차 문화콘텐츠산업 기술지원 사업의 지원을 받아 수행된 결과임

2. 관련 연구

2.1 OpenID [1][2][3]

2.1.1 용어 정의

OpenID 표준문서(OpenID Authentication 2.0)에 정의된 용어들은 다음과 같다.

User(사용자)는 OpenID 서비스를 이용하는 사용주체로 End-User로 정의되어 있다.

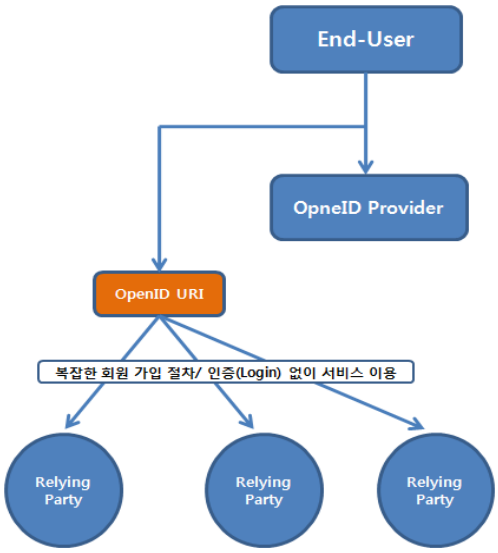
Consumer는 OpenID 서비스를 통해서 자신의 사이트에 사용자가 로그인을 할 수 있도록 서비스를 제공하는 사이트 측을 말한다. 쉽게 말해 사용자가 필요한 서비스를 제공하는 사이트 측을 말한다. RP 즉, Relying Party라고도 한다.

Provider는 사용자에게 OpenID를 발급해주고 사용자가 그 OpenID를 사용할 때마다 사용자를 인증해주는 서버 측을 말한다. OP 즉, OpenID Provider라고도 한다.

2.1.2 OpenID 서비스

OpenID 서비스는 하나의 URI의 형태를 가지는 식별자를 사용자에게 제공한다. 사용자는 다른 웹 사이트나 어플리케이션을 이용할 때 해당 사이트나 어플리케이션에 매번 가입하는 대신 발급받은 OpenID를 이용해 인증을 받을 수 있다.

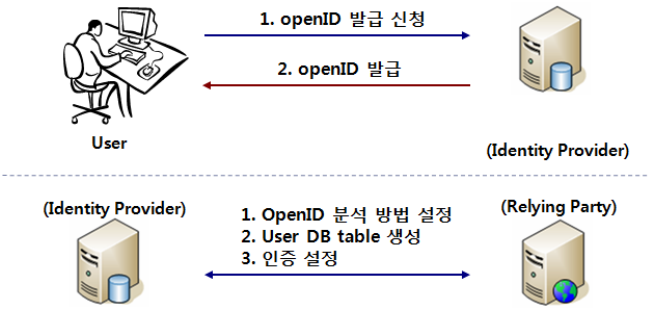
[그림1]에서는 위에서 설명한 OpenID의 유용성을 도식화해서 표현하고 있다. 사용자의 입장에서는 한 번 OP에 가입해서 OpenID를 발급 받으면 이후부터는 별도의 회원가입 절차 없이 RP들의 서비스를 이용할 수 있다. 따라서 사용자는 불필요한 회원 가입을 줄일 수 있을 뿐만 아니라 여러 개의 ID와 Password를 관리해야 하는 수고도 덜 수 있다.



[그림1] OpenID서비스의 구성

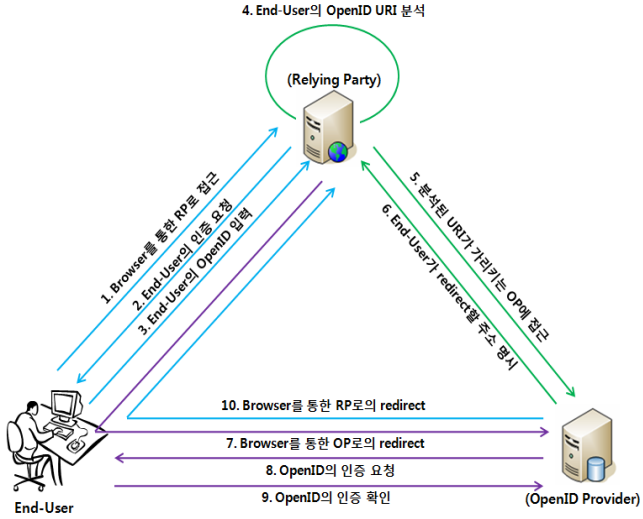
2.1.3 OpenID 서비스 인증 절차

OpenID 서비스를 이용하기 위해서 사용자는 OP에 회원 가입을 하여 OP로부터 URI형태의 OpenID를 발급 받아야 한다. RP는 사전에 사용자의 OpenID인증 요청에 대응할 수 있는 서비스 절차를 설정해야 한다. 이 과정은 [그림2]에 표현되어 있다.



[그림2] OpenID 액터간 관계

OpenID 인증을 절차는 다음과 같다. 최초로 사용자는 웹 브라우저를 통해 이용하고자 하는 웹 사이트(RP)에 접근을 한다. 웹 사이트 측에서는 사용자의 접근이 확인되면 인증을 요청하는데 이것에 대응해서 사용자는 자신이 가지고 있는 OpenID를 입력함으로써 ID의 인증을 요청한다. RP는 사용자가 입력한 OpenID URI를 분석해 OP의 URL을 찾아내, 이 URL을 통해 OP에 접근하여 사용자가 비밀번호를 입력하는 등의 인증과정을 진행할 페이지의 주소를 가져온다. RP는 이 주소를 사용자의 웹 브라우저에 제공해 사용자를 해당 URL로 유도한다. 이 때, RP는 사용자의 OpenID URI와 RP가 인증을 처리할 콜백 URL을 함께 웹 브라우저에 제공하는데 이 주소는 웹 브라우저가 OP에 접근할 때 OP에게 제공된다. 사용자는 웹 브라우저를 통해 RP가 제공한 OP의 주소로 리다이렉트한다. 사용자의 접근을 허용한 OP는 사용자가 가지고 있는 OpenID의 인증을 요청하게 되고 사용자는 이것에 답하여 인증을 완료한다. 사용자가 제시한 ID의 인증이 완료되면 OP는 받았던 콜백 URL로 리다이렉트 시는데 이 때 파라미터로 인증의 성공 여부 등을 함께 전송한다. RP는 이를 해석해 사용자가 서비스를 이용할 수 있도록 허용한다. 이 과정은 [그림3]에 나타나 있다.



[그림3]OpenID 서비스 진행과정

2.2 OAuth [4]

2.2.1 용어 정의

OAuth 보안 프로토콜의 표준스펙(OAuth Core 1.0)에 정의된 용어들은 다음과 같다.

Service Provider는 OAuth 프로토콜을 이용해서 웹 어플리케이션의 접근을 허락해 주는 측으로 OpenID의 OP에 연관 지어서 생각할 수 있다.

Consumer도 OpenID와 동일하게 OAuth 프로토콜을 이용하는 주체인 웹 사이트나 웹 어플리케이션을 말한다.

User는 서비스를 이용하는 주체이다.

Request Token은 User의 인증을 Service Provider에게 인증을 요청하고, 인증된 주체가 최초의 User가 맞는지를 확인해 주는 도구로 Consumer가 Service Provider에게 요청을 한다.

Access Token은 Consumer가 Service Provider에 인증된 User의 개인 정보나 기타 서비스에 접근을 할 수 있는 key의 역할을 하는 도구이다.

Consumer Key와 Consumer Secret은 2.2.3절에서 함께 설명한다.

2.2.2 OAuth 프로토콜

OAuth는 웹 어플리케이션(Consumer)이 서비스 제공자에 의해 보호되어 있는 사용자의 개인정보에 접근할 수 있도록 해주는 보안 프로토콜이다.

데이터 엑세스 솔루션인 OAuth 프로토콜은 웹 어플리케

이션이 서비스 제공자에 인증이 완료된 User의 개인정보 또는 User가 이용하고 있는 개인 서비스에 접근을 할 수 있도록 웹 어플리케이션과 서비스 제공자 사이의 인증을 유도한다. 이때, OAuth는 Consumer Key와 Consumer Secret, Request Token과 Access Token을 이용하여 인증방식의 보안을 강화하고 있다.

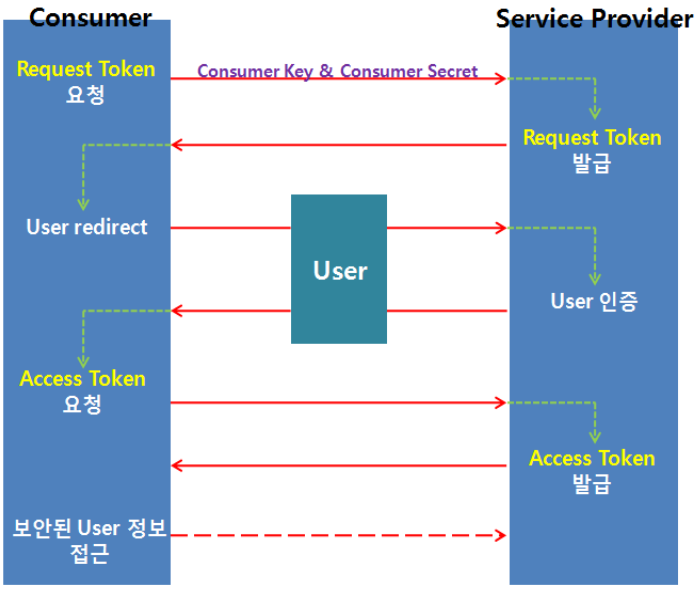
2.2.3 OAuth 프로토콜 인증 절차

OAuth 프로토콜을 이용하기 위해서 사전에 웹 어플리케이션은 서비스 제공자로 부터 발급받은 Consumer Key와 Consumer Secret을 보유하고 있어야 한다. 이 두 가지 값들은 서비스 제공자에 웹 어플리케이션이 인증을 받은 대임을 증명한다.

웹 어플리케이션은 서비스 제공자에게 미리 발급받은 Consumer Key와 Consumer Secret를 이용해 자신을 인증함과 동시에 Request Token의 발급을 요청한다. 서비스 제공자는 접근해 온 Consumer의 Key 값들을 확인하여 인증된 웹 어플리케이션이라고 판단하고 Request Token을 발급해 준다. Request Token이 발급 되면 웹 어플리케이션은 사용자를 서비스 제공자로 리다이렉트시키고 사용자는 서비스 제공자 측에서 인증을 받는다. 인증이 완료 되면 서비스 제공자는 사용자를 다시 웹 어플리케이션에게 리다이렉트시키는데, 이 때 인증된 사용자라는 사실을 증명하기 위하여 Request Token과 동일한 값을 함께 전송한다. 웹 어플리케이션은 콜백된 사용자가 가지고 온 값과 Request Token을 비교하여 최초 인증을 위해 리다이렉트했던 사용자인지를 판단하고 인증된 사용자로 판단되면 사용자의 개인정보 혹은 개인 서비스에 접근하기 위해 서비스 제공자에게 Access Token의 발급을 요청한다. 이에 서비스 제공자는 웹 어플리케이션의 요청을 받아들여 사용자의 정보에 접근할 수 있는 key 역할을 하는 Access Token을 발급해준다. 웹 어플리케이션은 발급받은 Access Token을 이용해서 User의 정보가 필요할 때에 서비스 제공자에 접근하여 User의 정보를 열람 할 수 있다. 이 과정은 [그림4]에 나타나 있다.

2.2.4 사용자 확인

OAuth 프로토콜의 결과물은 매번 랜덤하게 생성되는 Access Token이기 때문에 이것만으로는 사용자의 아이덴티티를 확인할 수가 없다. 이를 확인하기 위해서는 서비스 제공자가 사용자의 고유한 아이덴티티를 확인할 수 있는 수단을 제공해 주어야 한다. 이 방법은 서비스 제공자별로 상이하기 때문에 추가적인 처리가 필요하다. 일례로 Twitter는 Access Token을 전송할 때 파라메터로 사용자의 ID를 함께 전송하기 때문에 이를 바로 이용할 수 있으나, YouTube는 별도의 API를 호출해 사용자의 아이디를 확인해야 한다.

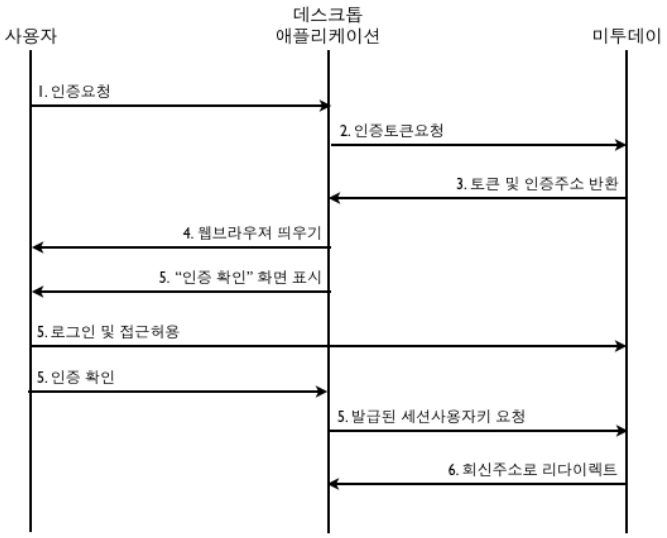


[그림4] OAuth 프로토콜의 인증 절차

2.3 사이트 종속적 Open API

2.3.1 개요

위에서 소개된 두가지 프로토콜은 사용자 인증을 위해 널리 쓰이고 있는 프로토콜이다. 그러나 이외에도 특정 사이트에서 자신의 사이트에 종속적인 Open API를 제공하는 경우가 많이 존재하는데, 이 API에서 해당 사이트의 회원 인증을 위한 메소드를 공개 지원한다면 이를 이용할 수 있다. 이 경우 대부분 OAuth와 유사한 형태와 절차를 갖는데, 대표적인 예로 me2DAY에서 제공하는 me2API를 들 수 있다.[5]



[그림5] me2DAY 웹 인증 절차[6]

2.3.2 me2API

[그림5]는 me2DAY에서 API로 제공하고 있는 '웹 기반 쉬운 인증'의 개요이다. OAuth와 마찬가지로 사용을 위

해서는 Application Key를 발급받아야 하며, 절차 역시 OAuth와 매우 유사함을 확인할 수 있다.

3. Mashed Login Access Control System 설계 및 구현

3.1 개요

Mashed Login Access Control System은 OpenID와 OAuth 및 기타 API를 통합한 로그인 환경을 제공해주는 시스템이다. 이 시스템을 적용한 웹 애플리케이션에는 이용자가 별도의 회원가입 없이 OpenID 혹은 자신이 기존에 가입하였던 사이트의 ID를 이용해 로그인할 수 있다.

3.2 설계

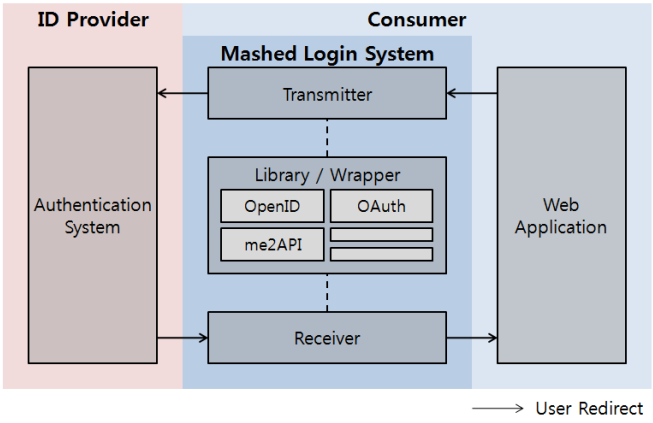
본 시스템은 웹 어플리케이션과 다양한 ID 제공자간의 통합된 인터페이스를 제공한다. 웹 어플리케이션 쪽에서 동작하지만 사용자의 로그인이 승인되었는지 여부만을 독립적으로 확인하므로 어플리케이션과의 직접적인 의존관계가 없다. 다양한 인증 프로토콜 및 API와의 호환을 위해 해당 프로토콜 및 API와 호환되는 라이브러리를 하나의 일관적인 인터페이스로 래핑하여 구현한다.

사용자가 사용 가능한 ID는 크게 OpenID와 그 외 사이트의 ID 두 가지로 나뉜다. 사용자가 자신만의 완전한 OpenID URL을 가지고 있는 경우에는 URL을 입력받아 인증 절차를 밟게끔 하면 된다. 그 외 특정 사이트에서 API를 제공해 줄 경우 해당 사이트에 가입된 ID로 로그인이 가능하다.

예를 들면, Flickr 등의 사이트는 OpenID 프로토콜로, Daum, Twitter, YouTube 등의 사이트는 OAuth 프로토콜로 회원들의 인증 절차를 제공한다. OpenID 프로토콜을 제공하는 사이트의 ID는 는 별다른 절차 없이 사용할 수 있으나, OAuth를 사용하는 사이트의 경우 별도로 해당 사이트에서 Consumer Key와 Consumer Secret를 발급받아 등록하고, 사용자의 아이덴티리를 확인할 수 있는 수단을 적용한 경우 사용 가능하다. 이 외에도 me2DAY처럼 사이트에 종속적인 Open API를 제공하는 사이트가 있는데, 이 경우에는 해당 API에 대한 라이브러리와 래퍼 클래스를 추가하면 사용 가능하다.[7][8][9][10][11]

인터페이스는 Transmit와 Receive 등 2개의 오퍼레이션을 갖는다. Transmit은 ID 제공자가 어느 것인지를 판단해 알맞은 클래스의 객체를 생성해 이를 세션에 저장하고 사용자를 인증 절차를 진행하는 ID 제공자의 사이트로 보낸다. Receive는 인증절차를 완료하고 콜백돼 돌아오는 사용자가 리다이렉트되었던 사용자와 동일한지 여부와 ID 제공자의 종류를 세션을 통해 확인하고 URL을 해석해 인증절차를 마무리한다.

현재 구현된 시스템에는 OpenID와 OAuth및 me2API에 래핑된 인터페이스가 존재하여 이를 이용하는 사이트에 대한 통합된 로그인 환경을 제공한다.



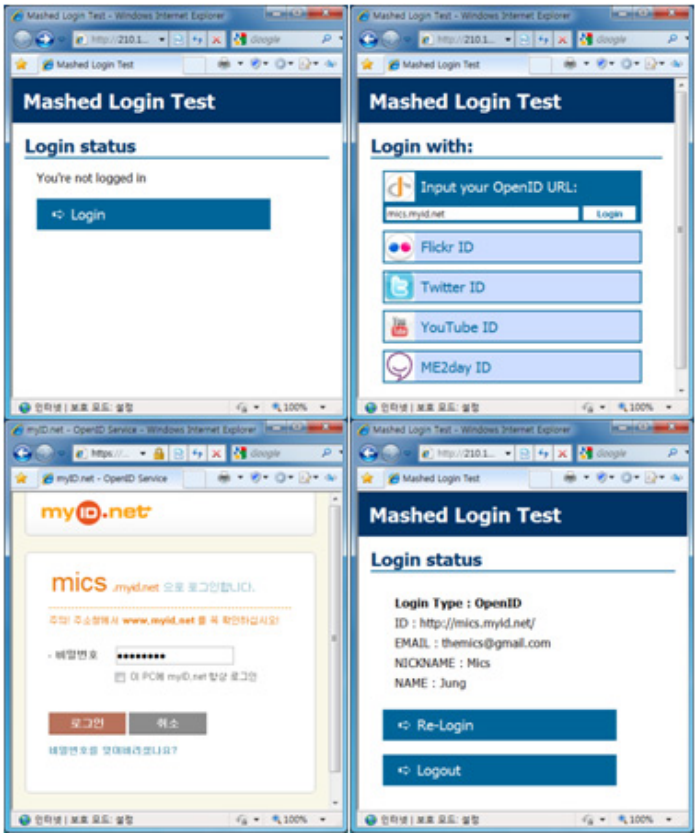
[그림6] 시스템 개요

3.3 구현 환경

본 시스템은 Java언어로 구현되었으며, Transmitter와 Receiver는 Servlet으로 작성되어 Jetty와 Tomcat 6.0에서 테스트하였다. 래핑된 라이브러리는 openid4java, net.oauth, me2day-api등이다.[12][13][14]

3.4 샘플 사이트

Mashed Login Access Control System을 적용한 Mashed Login 사이트를 함께 제작하였다. 이 사이트에는 OpenID를 비롯해 Flickr, Twitter, YouTube, me2DAY의 ID를 이용해 Sigle Sign On(SSO) 방식으로 로그인이 가능하다.



[그림7] Mashed Login 사이트

4. 결론

현재 웹에는 독창적이고 새로운 서비스들이 우후죽순처럼 생겨나고 있다. 하지만 그 유용성에도 불구하고 큰 주목을 받지 못하고 사라져 버리는 서비스들이 많다. 그 이유는 사용자들이 새로운 서비스에 가입하는 것을 꺼리는 것이 원인 중 하나이다. 이는 대부분 가입절차의 귀찮음과 신규 서비스에 대한 불신감 때문인데, 따라서 OpenID처럼 개인정보와 인증정보를 분리하는 대안적인 프로토콜이 등장하게 되었다.

본 연구에서는 OpenID와 OAuth를 비롯한 다양한 인증 프로토콜과 Open API를 이용하여 통합된 로그인 시스템을 구축하였고, 이를 이용하면 사용자가 신규회원가입 없이 새로운 서비스에 손쉽게 접근 하는데 도움을 줄 수 있다. 특히 Naver나 Daum처럼 기존에 많은 회원을 보유하고 있는 사이트들의 Open API를 적용하면 기존의 대형 사이트에서 가지고 있는 회원을 흡수할 수 있기 때문에 신생 서비스에 큰 도움이 된다.

하지만 실제로 대형 서비스에서 API를 공개하는 사례가 많지 않기에 이런 환경의 확대가 쉽지 않은 실정이다. 기존의 대형 서비스들이 독점적 지위를 유지하려고 하는 대신 API를 공개하여야 이를 이용한 창의적인 서비스들이 많이 나올 수 있고, 궁극적으로 바람직한 웹 생태계 조성에 일조할 수 있을 것이다.

참고문헌

[1] 신동휘, 전인경, 정현철, 계층적 ID를 통한 OpenID 보안 기능 강화 연구, 2009 한국컴퓨터종합학술대회 논문집, pp 10-14, 2009
[2] OpenID Authentication 2.0, http://openid.net/specs/openid-authentication-2_0.html
[3] OpenID: an actually distributed identity system, http://openid.co.kr
[4] OAuth Core 1.0 Specification, http://oauth.net/core/1.0/
[5] me2API, http://codian.springnote.com/pages/86001
[6] me2API - 웹 기반 쉬운인증, http://codian.springnote.com/pages/1645274
[7] Federated Login for Google Account Users, http://code.google.com/intl/ko/apis/accounts/docs/OpenID.html
[8] Flicker API, http://www.flickr.com/services/api/
[9] Daum OAuth 소개, https://apis.daum.net/oauth/main/welcome
[10] Twitter API Wiki, http://apiwiki.twitter.com/
[11] API Overview Guide - YouTube APIs and Tools, http://code.google.com/intl/ko-KR/apis/youtube/getting_started.html
[12] openid4java, http://code.google.com/p/openid4java
[13] oauth, http://code.google.com/p/oauth
[14] me2day-api, http://code.google.com/p/me2day-api