<http://d2.naver.com/helloworld/24942>

# OAuth와 춤을

2012.03.14

|

96733

최근의 인터넷 서비스는 그 자체가 SaaS(Software as a Service)의 형태입니다. 서비스 중에서 사용자가 일부 필요한 것만 사용할 수 있게 한다는 것입니다. Facebook이나 트위터가 세상에 널리 퍼지게된 이유 중에 하나가 외부 서비스에서도 Facebook가 트위터의 일부 기능을 사용할 수 있게 한 것입니다. 외부 서비스와 연동되는 Facebook이나 트위터의 기능을 이용하기 위해 사용자가 반드시 Facebook이나 트위터에 로그인해야 하는 것이 아니라, 별도의 인증 절차를 거치면 다른 서비스에서 Facebook과 트위터의 기능을 이용할 수 있게 되는 것입니다. 이런 방식은 Facebook이나 트위터 같은 서비스 제공자뿐만 아니라 사용자와 여러 인터넷 서비스 업체 모두에 이익이 되는 생태계를 구축하는데 기여하게 될 것입니다.

이 방식에서 사용하는 인증 절차가 OAuth입니다. 이 기사에서는 OAuth의 탄생 배경과 OAuth 1.0의 인증 과정에 대해 간략하게 설명합니다.

## OAuth의 탄생과 사용

OAuth는 인증을 위한 오픈 스탠더드 프로토콜로, 사용자가 Facebook이나 트위터 같은 인터넷 서비스의 기능을 다른 애플리케이션(데스크톱, 웹, 모바일 등)에서도 사용할 수 있게 한 것이다.

OAuth의 탄생 이전에도 다른 애플리케이션에 사용자의 아이디와 암호가 노출되지 않도록 하면서 API 접근 위임(API Access Delegation)이 가능한 여러 인증 방법이 있었다. Google과 Yahoo!, AOL, Amazon 등에서는 각각의 인증 방식을 제작하여 사용했다.

OAuth 1.0이 나온 때는 2007년이며, 이후 보안 문제를 해결한 수정 버전인 OAuth 1.0 revision A 가 2008년에 나왔다.

OAuth의 시작은 2006년에 트위터의 개발자와 소셜 북마크 서비스인 Gnolia의 개발자가 만나 인증 방식을 논의한 때부터였다. 두 회사의 개발자들은 그때까지 API 접근 위임에 대한 표준안이 없다는 것을 알았다. 그래서 2007년 4월 인터넷에 OAuth 논의체를 만든 뒤 OAuth드래프트 제안서를 만들어 공유했다. 그 뒤에 이 활동을 지지하는 사람이 생기게 되었고, 2008년 IETF 모임(73회, 미네소타에서 개최)에서 OAuth가 IETF 표준안으로 관리되야 하는 가에 대한 논의가 있었다. 그리고 2010년에 OAuth 1.0 프로토콜 표준안이 RFC5849로 발표되었다.

2007년에 나온 OAuth 1.0은 비공식 논의체에 의해 최초로 만들어진 것이고, 2010년 IETF OAuth 워킹그룹에 의해 이 프로토콜이 IETF 표준 프로토콜로 발표된 것이다.

현재 나와 있는 OAuth 2.0은 드래프트 단계에 있는 것으로, IETF OAuth 워킹그룹이 주축이 되어 만든 것이다. OAuth 2.0은 OAuth 1.0과 호환되지 않지만, 인증 절차가 간략하다는 장점이 있다. 그래서 아직 최종안이 나오지 않았음에도 여러 인터넷 서비스에서 OAuth 2.0을 사용하고 있다. 다음 표는 대표적인 인터넷 서비스 기업에서 사용하는 OAuth의 버전을 정리한 것이다.

표 1 인터넷 서비스 기업이 사용하는 OAuth 버전

|  |  |
| --- | --- |
| 인터넷 서비스 기업 | OAuth 버전 |
| Facebook | 2.0 |
| Foursquare | 2.0 |
| Google | 2.0 |
| Microsoft (Hotmail, Messenger, Xbox) | 2.0 |
| LinkedIn | 2.0 |
| Daum(티스토리 | 2.0 |
| NHN (오픈API) | 1.0a |
| Daum(요즘, 오픈API) | 1.0a |
| MySpace | 1.0a |
| Netflix | 1.0a |
| 트위터 | 1.0a |
| Vimeo | 1.0a |
| Yahoo! | 1.0a |

## OAuth와 로그인

OAuth와 로그인은 반드시 분리해서 이해해야 한다. 일상 생활을 예로 들어 OAuth와 로그인의 차이를 설명해 보겠다.

사원증을 이용해 출입할 수 있는 회사를 생각해 보자. 그런데 외부 손님이 그 회사에 방문할 일이 있다. 회사 사원이 건물에 출입하는 것이 로그인이라면 OAuth는 방문증을 수령한 후 회사에 출입하는 것에 비유할 수 있다.

다음과 같은 절차를 생각해 보자.

* 나방문씨(외부 손님)가 안내 데스크에서 업무적인 목적으로 김목적씨(회사 사원)를 만나러 왔다고 말한다.
* 안내 데스크에서는 김목적씨에게 나방문씨가 방문했다고 연락한다.
* 김목적씨가 안내 데스크로 찾아와 나방문씨의 신원을 확인해 준다.
* 김목적씨는 업무 목적과 인적 사항을 안내 데스크에서 기록한다.
* 안내 데스크에서 나방문 씨에게 방문증을 발급해 준다.
* 김목적씨와 나방문씨는 정해진 장소로 이동해 업무를 진행한다.

위 과정은 방문증 발급과 사용에 빗대어 OAuth 발급 과정과 권한을 이해할 수 있도록 한 것이다. 방문증이란 사전에 정해진 곳만 다닐 수 있도록 하는 것이니, '방문증'을 가진 사람이 출입할 수 있는 곳과 '사원증'을 가진 사람이 출입할 수 있는 곳은 다르다. 역시 직접 서비스에 로그인한 사용자와 OAuth를 이용해 권한을 인증받은 사용자는 할 수 있는 일이 다르다.

OAuth에서 'Auth'는 'Authentication'(인증)뿐만 아니라 'Authorization'(허가) 또한 포함하고 있는 것이다. 그렇기 때문에 OAuth 인증을 진행할 때 해당 서비스 제공자는 '제 3자가 어떤 정보나 서비스에 사용자의 권한으로 접근하려 하는데 허용하겠느냐'라는 안내 메시지를 보여 주는 것이다. 다음의 화면 두 개는 각각 네이버와 미투데이에 접근 권한 요청이 있음을 알려 주는 화면이다. 미투데이의 경우에는 OAuth 인증 방식을 사용하지 않지만 인증 절차에 대한 예시 정도로 생각해 주기 바란다.

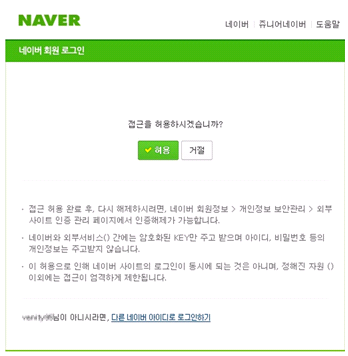


그림 1 네이버 OAuth 접근 권한 요청 화면

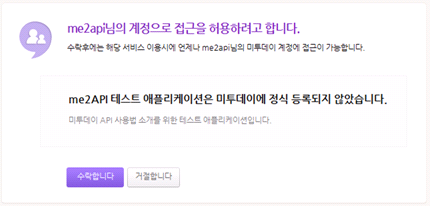


그림 2 미투데이 접근 권한 요청 화면

## OpenID와 OAuth

OpenID도 인증을 위한 표준 프로토콜이고 HTTP를 사용한다는 점에서는 OAuth와 같다. 그러나 OpenID와 OAuth의 목적은 다르다.

OpenID의 주요 목적은 인증(Authentication)이지만, OAuth의 주요 목적은 허가(Authorization)이다. 즉, OpenID를 사용한다는 것은 본질적으로 로그인하는 행동과 같다. OpenID는 OpenID Provider에서 사용자의 인증 과정을 처리한다. Open ID를 사용하는 여러 서비스(Relying Party)는 OpenID Provider에게 인증을 위임하는 것이다.

물론 OAuth에서도 인증 과정이 있다. 가령 Facebook의 OAuth를 이용한다면 Facebook의 사용자인지 인증하는 절차를 Facebook(Service Provider) 처리한다. 하지만 OAuth의 근본 목적은 해당 사용자의 담벼락(wall)에 글을 쓸 수 있는 API를 호출할 수 있는 권한이나, 친구 목록을 가져오는 API를 호출할 수 있는 권한이 있는 사용자인지 확인하는 것이다.

OAuth를 사용자 인증을 위한 방법으로 쓸 수 있지만, OpenID와 OAuth의 근본 목적은 다르다는 것을 알아야 한다.

## OAuth Dance, OAuth 1.0 인증 과정

OAuth를 이용하여 사용자를 인증을 하는 과정을 OAuth Dance라고 한다. 두 명이 춤을 추듯 정확하게 정보를 주고받는 과정을 재미있게 명명한 것이다.

OAuth를 이해하려면 몇 가지 용어를 먼저 알아 두어야 한다. 다음 표에 OAuth의 대표 용어를 정리해 보았다.

표 2 OAuth 1.0 대표 용어

|  |  |
| --- | --- |
| 용어 | 설명 |
| User | Service Provider에 계정을 가지고 있으면서, Consumer를 이용하려는 사용자 |
| Service Provider | OAuth를 사용하는 Open API를 제공하는 서비스 |
| Consumer | OAuth 인증을 사용해 Service Provider의 기능을 사용하려는 애플리케이션이나 웹 서비스 |
| Request Token | Consumer가 Service Provider에게 접근 권한을 인증받기 위해 사용하는 값. 인증이 완료된 후에는 Access Token으로 교환한다. |
| Access Token | 인증 후 Consumer가 Service Provider의 자원에 접근하기 위한 키를 포함한 값 |

다음은 OAuth 인증 과정을 그림으로 표현한 것이다.

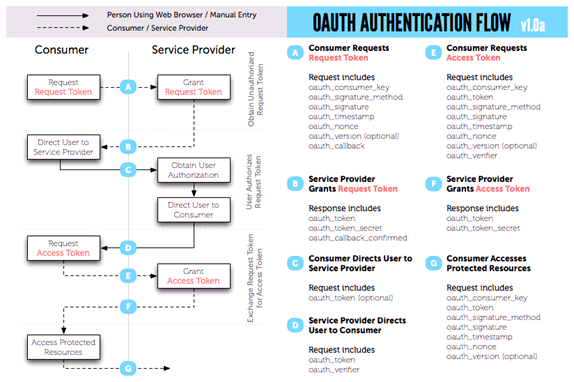


그림 3 OAuth 1.0a 인증 과정(원본 출처: <http://oauth.net/core/diagram.png>)

OAuth 인증 과정을 앞에서 설명한 회사 방문 과정과 연결하면 다음 표와 간다.

표 3 회사 방문 과정과 OAuth 인증 과정

|  |  |  |
| --- | --- | --- |
|  | 회사 방문 과정 | OAuth 인증 과정 |
| 1. | 나방문씨가 안내 데스크에서 업무적인 목적으로 김목적씨를 만나러 왔다고 말한다. | Request Token의 요청과 발급 |
| 2. | 안내 데스크에서는 김목적씨에게 나방문씨가 방문했다고 연락한다. | 사용자 인증 페이지 호출 |
| 3. | 김목적씨가 안내 데스크로 찾아와 나방문씨의 신원을 확인해 준다. | 사용자 로그인 완료 |
| 4. | 김목적씨는 업무 목적과 인적 사항을 안내 데스크에서 기록한다. | 사용자의 권한 요청 및 수락 |
| 5. | 안내 데스크에서 나방문 씨에게 방문증을 발급해 준다. | Access Token 발급 |
| 6. | 김목적씨와 나방문씨는 정해진 장소로 이동해 업무를 진행한다. | Access Token을 이용해 서비스 정보 요청 |

위의 표에 따르면, Access Token은 방문증이라고 이해할 수 있다. 이 방문증으로 사전에 허락된 공간에 출입할 수 있다. 마찬가지로 Access Token을 가지고 있는 Consumer는 사전에 호출이 허락된 Service Provider의 오픈 API를 호출할 수 있는 것이다.

### Request Token

OAuth에서 Consumer가 Request Token 발급을 요청하고 Service Provider가 Request Token을 발급하는 과정은 "저 나방문입니다. 김목적씨를 만날 수 있을까요?"라고 말하는 절차에 비유할 수 있다.

Request Token을 요청하는 Request 전문을 살펴보자. 다음은 네이버의 OAuth API로 Request Token을 요청하는 예이다.

GET /naver.oauth?mode=req\_req\_token&oauth\_callback=http://example.com/OAuthRequestToken.do&oauth\_consumer\_key=WEhGuJZWUasHg&oauth\_nonce=zSs4RFI7lakpADpSsv&oauth\_signature=wz9+ZO5OLUnTors7HlyaKat1Mo0=&oauth\_signature\_method=HMAC-SHA1&oauth\_timestamp=1330442419&oauth\_version=1.0 HTTP/1.1

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

Host: nid.naver.com

보기 쉽도록 위의 내용을 아래와 같이 매개변수를 기준으로 정리했다.

GET http://nid.naver.com/naver.oauth?mode=req\_req\_token&

oauth\_callback=http://example.com/OAuthRequestToken.do&

oauth\_consumer\_key=WEhGuJZWUasHg&

oauth\_nonce=zSs4RFI7lakpADpSsv&

oauth\_signature=wz9+ZO5OLUnTors7HlyaKat1Mo0=&

oauth\_signature\_method=HMAC-SHA1&

oauth\_timestamp=1330442419&

oauth\_version=1.0 HTTP/1.1

Request Token 발급 요청 시 사용하는 매개변수는 다음 표와 같다.

표 4 Request Token 발급 요청 시 사용하는 매개변수

|  |  |
| --- | --- |
| 매개변수 | 설명 |
| oauth\_callback | Service Provider가 인증을 완료한 후 리다이렉트할 Consumer의 웹 주소. 만약 Consumer가 웹 애플리케이션이 아니라 리다이렉트할 주소가 없다면 소문자로 'oob'(Out Of Band라는 뜻)를 값으로 사용한다. |
| oauth\_consumer\_key | Consumer를 구별하는 키 값. Service Provider는 이 키 값으로 Consumer를 구분한다. |
| oauth\_nonce | Consumer에서 임시로 생성한 임의의 문자열. oauth\_timestamp의 값이 같은 요청에서는 유일한 값이어야 한다. 이는 악의적인 목적으로 계속 요청을 보내는 것을 막기 위해서이다. |
| oauth\_signature | OAuth 인증 정보를 암호화하고 인코딩하여 서명 값. OAuth 인증 정보는 매개변수 중에서 oauth\_signature를 제외한 나머지 매개변수와 HTTP 요청 방식을 문자열로 조합한 값이다. 암화 방식은 oauth\_signature\_method에 정의된다. |
| oauth\_signature\_method | oauth\_signature를 암호화하는 방법. HMAC-SHA1, HMAC-MD5 등을 사용할 수 있다. |
| oauth\_timestamp | 요청을 생성한 시점의 타임스탬프. 1970년1월 1일 00시 00분 00초 이후의 시간을 초로 환산한 초 단위의 누적 시간이다. |
| oauth\_version | OAuth 사용 버전. 1.0a는 1.0이라고 명시하면 된다. |

### oauth\_signature 만들기

OAuth 1.0에서는 oauth\_signature를 생성하는 것이 가장 까다로운 단계이다. 당연히 Consumer와 Service Provider가 같은 암호화(signing) 알고리즘을 이용하여 oauth\_signature를 만들어야 한다.

oauth\_sinature는 다음과 같이 네 단계를 거쳐 만든다.

#### 1. 요청 매개변수를 모두 모은다.

oauth\_signature를 제외하고 'oauth\_'로 시작하는 OAuth 관련 매개변수를 모은다. POST body에서 매개변수를 사용하고 있다면 이 매개변수도 모아야 한다.

#### 2. 매개변수를 정규화(Normalize)한다.

모든 매개변수를 사전순으로 정렬하고 각각의 키(key)와 값(value)에 URL 인코딩(rfc3986)을 적용한다. URL 인코딩을 실시한 결과를 = 형태로 나열하고 각 쌍 사이에는 &을 넣는다. 이렇게 나온 결과 전체에 또 URL 인코딩을 적용한다.

#### 3. Signature Base String을 만든다.

HTTP method 명(GET 또는 POST), Consumer가 호출한 HTTP URL 주소(매개변수 제외), 정규화한 매개변수를 '&'를 사용해 결합한다. 즉 '[GET|POST] + & + [URL 문자열로 매개변수는 제외] + & + [정규화한 매개변수]' 형태가 된다. 이 예제에서는 '<http://nid.naver.com/naver.oauth>' 을 URL로 사용하고, 이 URL에 URL 인코딩을 적용한 값을 사용했다.

#### 4. 키 생성

3번 과정까지 거쳐 생성한 문자열을 암호화한다. 암호화할 때 Consumer Secret Key를 사용한다. Consumer Secret Key는 Consumer가 Service Provider에 사용 등록을 할 때 발급받은 값이다. HMAC-SHA1 등의 암호화 방법을 이용하여 최종적인 oauth\_signature를 생성한다.

### 사용자 인증 페이지의 호출

OAuth에서 사용자 인증 페이지를 호출하는 단계는 '안내데스크에서 김목적씨에게 방문한 손님이 있으니 안내 데스크로와서 확인을 요청하는 것'에 비유할 수 있다. Request Token을 요청하면, Service Provider는 Consumer에 Request Token으로 사용할 oauth\_token과 oauth\_token\_secret을 전달한다. Access Token을 요청할 때는 Request Token의 요청에 대한 응답 값으로 받은 oauth\_token\_secret을 사용한다. Consumer가 웹 애플리케이션이라면 HTTP 세션이나 쿠키 또는 DBMS 등에 oauth\_token\_secret를 저장해 놓아야 한다.

oauth\_token을 이용해 Service Provider가 정해 놓은 사용자 인증 페이지를 User에게 보여 주도록 한다. 네이버의 경우 OAuth용 사용자 인증 페이지의 주소는 다음과 같다.

https://nid.naver.com/naver.oauth?mode=auth\_req\_token

여기에 Request Token을 요청해서 반환받은 oauth\_token을 매개 변수로 전달하면 된다. 예를 들면 다음과 같은 URL이 만들어 지게 되는 것이다. 이 URL은 사용자 인증 화면을 가리킨다.

https://nid.naver.com/naver.oauth?mode=auth\_req\_token&oauth\_token=wpsCb0Mcpf9dDDC2

로그인 화면을 호출하는 단계까지가 '안내 데스크에서 김목적씨에게 전화하는 단계'라 보면 되겠다. 이제 김목적씨가 안내 데스크로 와서 나방문씨를 확인해야 한다. 정말 나방문씨가 맞는지 아닌지 확인하는 과정이 필요할 것이다. 이 과정이 OAuth에서는 Service Provider에서 User를 인증하는 과정이라고 볼 수 있다.

인증이 완료되면 앞에서 말한 바와 같이 어떤 권한을 요청하는 단계에 이르게 된다. "업무 약속이 있어 오셨으니 나방문씨가 출입할 수 있게 해 주세요"와 같은 것 말이다.

인증을 마치면 Consumer가 oauth\_callback에 지정한 URL로 리다이렉트한다. 이때 Service Provider는 새로운 oauth\_token과 oauth\_verifier를 Consumer에 전달한다. 이 값들은 Access Token을 요청할 때 사용한다.

### Access Token 요청하기

OAuth에서의 AccessToken은 나방문 씨에게 지급할 방문증과 같다.

Access Token을 요청하는 방법은 Request Token을 요청하는 방법과 거의 같다. 다만, 사용하는 매개변수의 종류가 약간 다르고 oauth\_signature를 생성할 때 사용하는 키가 다르다. Access Token 을 요청할 때에는 매개변수 oauth\_callback는 없고, oauth\_token와 oauth\_verifer가 있다.

Request Token 발급을 요청할 때에는 Consumer Secret Key를 사용해 oauth\_token\_secret를 생성했다. Access Token 발급을 요청할 때에는 Consumer Secret Key에 oauth\_token\_secret을 결합한 값(Consumer Secret Key + & + oauth\_token\_secret)을 사용해 oauth\_token\_secret를 생성한다. 암호화 키를 변경하여 보안을 더 강화하는 것이다.

Access Token 발급을 요청할 때 사용하는 매개변수는 다음 표와 같다.

표 5 Access Token 발급 요청을 위한 OAuth 매개변수

|  |  |
| --- | --- |
| **매개변수** | **설명** |
| oauth\_consumer\_key | Consumer를 구별하는 키 값. Service Provider는 이 키 값으로 Consumer를 구분한다. |
| oauth\_nonce | Consumer에서 임시로 생성한 임의의 문자열. oauth\_timestamp의 값이 같은 요청에서는 유일한 값이어야 한다. 이는 악의적인 목적으로 계속 요청을 보내는 것을 막기 위해서이다. |
| oauth\_signature | OAuth 인증 정보를 암호화하고 인코딩하여 서명 값. OAuth 인증 정보는 매개변수 중에서 oauth\_signature를 제외한 나머지 매개변수와 HTTP 요청 방식을 문자열로 조합한 값이다. 암화 방식은 oauth\_signature\_method에 정의된다. |
| oauth\_signature\_method | oauth\_signature를 암호화하는 방법. HMAC-SHA1, HMAC-MD5 등을 사용할 수 있다. |
| oauth\_timestamp | 요청을 생성한 시점의 타임스탬프. 1970년1월 1일 00시 00분 00초 이후의 시간을 초로 환산한 초 단위의 누적 시간이다. |
| oauth\_version | OAuth 사용 버전 |
| oauth\_verifier | Request Token 요청 시 oauth\_callback으로 전달받은 oauth\_verifier 값 |
| oauth\_token | Request Token 요청 시 oauth\_callback으로 전달받은 oauth\_token 값 |

위의 표에 정의한 매개변수를 상황에 맞게 정의한 다음 Access Token을 요청하면 oauth\_token과 oauth\_token\_secret을 전달받게 된다. Service Provider에 따라 사용자의 아이디나 프로필 정보 같은 것들이 반환되기도 한다.

### Access Token 사용하기

드디어 방문증이 발급됐다. 이제 출입문을 통과하는 일만 남았다. 방문증을 가지고 출입문을 통과한다는 것은 User의 권한으로 Service Provider의 기능을 사용하는 것과 비슷하다. 다시 말해, 권한이 필요한 오픈 API를 호출할 수 있게 되는 것이다.

가령 네이버 카페에서 게시판 목록을 가져온다고 한다면 호출해야 하는 URL은 다음과 같다.

http://openapi.naver.com/cafe/getMenuList.xml

특정 User의 권한을 가지고 카페 게시판 목록 반환 URL을 요청해야 해당 User가 가입한 카페의 게시판 목록을 반환받을 수 있을 것이다. 이 URL을 호출할 때는 OAuth 매개변수를 함께 전달해야 한다.

다음은 Access Token을 사용해 오픈 API를 요청하는 예이다. HTTP 헤더에 Authorization 필드를 두었고, Authorization 필드의 값 부분에 OAuth 매개변수 적는다. Access Token을 사용할 때는 GET이나 POST가 아닌 HEAD 방식을 사용한다.

POST /cafe/getMenuList.xml HTTP/1.1

Authorization: OAuth oauth\_consumer\_key="dpf43f3p2l4k3l03",oauth\_token="nSDFh734d00sl2jdk"

,oauth\_signature\_method="HMACSHA1",oauth\_timestamp="1379123202",oauth\_nonce="chapoH",oauth\_signature="MdpQcU8iPSUjWoN%2FUDMsK2sui9I%3D"

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

Host: http://openapi.naver.com

보기 쉽도록 Authorization 필드를 아래와 같이 매개변수를 기준으로 정리했다.

Authorization: OAuth oauth\_consumer\_key="dpf43f3p2l4k3l03",

oauth\_token="nSDFh734d00sl2jdk",

oauth\_signature\_method="HMACSHA1",

oauth\_timestamp="1379123202",

oauth\_nonce="csrrkjsd0OUhja",

oauth\_signature="MdpQcU8iPGGhytrSoN%2FUDMsK2sui9I%3D"

Access Token을 사용해 오픈 API를 호출할 때 사용하는 매개변수는 다음 표와 같다.

표 6 Access Token을 사용해 오픈 API를 호출 할 때 필요한 OAuth 매개변수

|  |  |
| --- | --- |
| 매개변수 | 설명 |
| oauth\_consumer\_key | Consumer를 구별하는 키 값. Service Provider는 이 키 값으로 Consumer를 구분한다. |
| oauth\_nonce | Consumer에서 임시로 생성한 임의의 문자열. oauth\_timestamp의 값이 같은 요청에서는 유일한 값이어야 한다. 이는 악의적인 목적으로 계속 요청을 보내는 것을 막기 위해서이다. |
| oauth\_signature | OAuth 인증 정보를 암호화하고 인코딩하여 서명 값. OAuth 인증 정보는 매개변수 중에서 oauth\_signature를 제외한 나머지 매개변수와 HTTP 요청 방식을 문자열로 조합한 값이다. 암화 방식은 oauth\_signature\_method에 정의된다. |
| oauth\_signature\_method | oauth\_signature를 암호화하는 방법. HMAC-SHA1, HMAC-MD5 등을 사용할 수 있다. |
| oauth\_timestamp | 요청을 생성한 시점의 타임스탬프. 1970년1월 1일 00시 00분 00초 이후의 시간을 초로 환산한 초 단위의 누적 시간이다. |
| oauth\_version | OAuth 버전 |
| oauth\_token | oauth\_callback으로 전달받은 oauth\_token |

**주의**

Access Token을 이용해 요청할 때, Service Provider에 따라 realm이라는 매개변수를 사용해야 하는 경우도 있다. realm은 optional 매개변수인데, WWW-Authenticate HTTP 헤더 필드에서 사용하는 값이다.

## OAuth 2.0

OAuth 1.0은 웹 애플리케이션이 아닌 애플리케이션에서는 사용하기 곤란하다는 단점이 있다. 또한 절차가 복잡하여 OAuth 구현 라이브러리를 제작하기 어렵고, 이런저런 복잡한 절차 때문에 Service Provider에게도 연산 부담이 발생한다.

OAuth 2.0은 이러한 단점을 개선한 것이다. OAuth 1.0과 호환성이 없고, 아직 최종안이 발표된 것은 아니지만 여러 인터넷 서비스 기업에서 OAuth 2.0을 사용하고 있다.

OAuth 2.0의 특징은 다음과 같다.

* 웹 애플리케이션이 아닌 애플리케이션 지원 강화
* 암호화가 필요 없음 HTTPS를 사용하고 HMAC을 사용하지 않는다.
* Siganature 단순화 정렬과 URL 인코딩이 필요 없다.
* Access Token 갱신 OAuth 1.0에서 Access Token을 받으면 Access Token을 계속 사용할 수 있었다. 트위터의 경우에는 Access Token을 만료시키지 않는다. OAuth 2.0에서는 보안 강화를 위해 Access Token의 Life-time을 지정할 수 있도록 했다.

이외에도 OAuth 2.0에서 사용하는 용어 체계는 OAuth 1.0과 완전히 다르다. 같은 목적의 다른 프로토콜이라고 이해는 것이 좋다. 하지만 아직 최종안이 나오지 않았기 때문에, 현재로서는 OAuth 2.0의 특징만 파악하는 것으로도 충분할 듯 하다.

## OAuth로 인한 인터넷 서비스의 변화

OAuth는 요즘의 인터넷 생태계의 주요 요소로 자리매김하고 있다.

신생 업체들은 고유의 인증 방식을 사용하기 보다는 Facebook이나 트위터의 인증을 이용하고 있다. 개발 비용과 운영 비용을 줄이는 효과도 있지만 Facebook이나 트위터를 통해 서비스를 홍보하는 효과를 만들 수도 있다. 그리고 Service Provider 입장에서는 핵심 기능을 더욱 공고히 하는 효과를 얻고 있다.

표준 인증 방법 하나가 인터넷 업계에 변화를 주고 있는 것이다.

[oauth](http://d2.naver.com/search/tag?keyword=oauth),[openid](http://d2.naver.com/search/tag?keyword=openid)



오창훈|NHN 소셜앱스서비스팀

2000년부터 웹마스터로 개발을 시작하게 되었고, 아직까지 치킨 장사 안하고 살아남아 자랑스럽게 개발을 하고 있습니다. 앞으로도 그러하겠지요.. 그래서 즐겁고 행복합니다. 전 개발과 관련된 것이면 무엇이든 알고 싶고, Hello World 한번 정도는 꼭 찍어보는 성격입니다. 그냥 만드는 것이 좋습니다. DIY, 프라모델, 캠핑을 취미로 삼고 있습니다. 예전엔 테크니컬리더로 프론트의 신기술 도입이나 적용 그리고 서비스의 최적화 작업을 주로 했었으며, 지금은 NHN에서 오픈소셜 플랫폼의 에반젤리스트 역할을 하고 있습니다.