Accessing Twitter Resources Using OAUTH

Contributed by Aminur Rashid
Thursday, 03 May 2012
Last Updated Thursday, 03 May 2012

OAuth is an Authorization scheme to protect resources. It is widely used to share private resources (e.g. photos, videos, contact lists) stored on one site with another site without having to hand out their credentials. Twitter is one of such website, that exposes its resources as Rest Web services. These resources are protected resources and can be accessed against proper Authorization.

User will need the access if he is building a twitter Application. Following steps explains the simplest use case to access friend's or one's own timeline.

1).Register an application with twitter.

Go to twitter's development resource and register your Application. [URL : https://dev.twitter.com/apps/new]

Once the application is registered, note down the Consumer key and Consumer Secret for your application. For the Demo, I am using dummyConsumerKey/Secret which should be replaced with your working Consumer key and Consumer Secret

2).Get the Request Token

http://api.twitter.com/oauth/request_token is twitter URL to get the request token. (see docs). Access to the URL is protected and we need OAUTH header to access the url. Following utility class will be used across the Application to generate the OAUTH Authorization header. You will need Apache codec jar for Base64 encoding.

```
import java.io.UnsupportedEncodingException;

import java.net.URLEncoder;

import java.util.Collections;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;
import java.util.TreeMap;
import java.util.regex.Pattern;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

import org.apache.commons.codec.binary.Base64;


public class HMACSHAUtil
{
  private static final String TIMESTAMP = "oauth_timestamp";
  private static final String SIGN_METHOD = "oauth_signature_method";
  private static final String SIGNATURE = "oauth_signature";
  private static final String CONSUMER_KEY = "oauth_consumer_key";
  private static final String VERSION = "oauth_version";
  private static final String NONCE = "oauth_nonce";
  private static final String TOKEN = "oauth_token";
  private static String CHARSET = "UTF-8";
  private static Timer timer = new Timer();
  private static final String EMPTY_STRING = "";
  private static final Map<String, String> ENCODING_RULES;
  private static final String AMPERSAND_SEPARATED_STRING = "%s&%s&%s";
  private static final String HMAC_SHA1 = "HmacSHA1";
  private static final String CARRIAGE_RETURN = "\r\n";
  private static final String VERIFIER = "oauth_verifier";
  static
  {
```

```
  Map<String, String> rules = new HashMap<String, String>();
  rules.put("*", "%2A");
  rules.put("+", "%20");
  rules.put("%7E", "~");
  ENCODING_RULES = Collections.unmodifiableMap(rules);
}

public HMACSHAUtil()
{
  super();
}

static String OAUTH1Header(String oauth_token, String consumer_key, String httpMethod,
                           String url, String consumerSecret, String oauth_token_secret,
                           String verifier)
  throws Exception
{
  Map<String, String> oauthParameters = new TreeMap<String, String>();
  oauthParameters.put(TOKEN, oauth_token);
  oauthParameters.put(TIMESTAMP, getTimestampInSeconds());
  oauthParameters.put(NONCE, getNonce());
  oauthParameters.put(CONSUMER_KEY, consumer_key);
  oauthParameters.put(SIGN_METHOD, "HMAC-SHA1");
  oauthParameters.put(VERSION, "1.0"); /**/
  if (verifier != null && !verifier.isEmpty())
  {
    oauthParameters.put(VERIFIER, verifier);
  }
  oauthParameters.put(VERSION, "1.0");


  String params = encode(asFormUrlEncodedString(oauthParameters));
  String baseString = String.format(AMPERSAND_SEPARATED_STRING, encode(httpMethod),
                                    encode(getSanitizedUrl(url)), params);
  String signature = doSign(baseString,encode(consumerSecret)+'&'+encode(oauth_token_secret));

  oauthParameters.put(SIGNATURE, signature);
  return generateOAuth1Header(oauthParameters);
}

private static String generateOAuth1Header(Map<String, String> parameters)
{
  StringBuffer header = new StringBuffer(parameters.size() * 20);
  header.append("OAuth ");
  for (String key: parameters.keySet())
  {
    if (header.length() > "OAuth ".length())
    {
      header.append(", ");
    }
    header.append(String.format("%s=\"%s\"", key, encode(parameters.get(key))));
  }
  return header.toString();
}

private static String doSign(String toSign, String keyString)
  throws Exception
{
  SecretKeySpec key = new SecretKeySpec((keyString).getBytes(CHARSET), HMAC_SHA1);
  Mac mac = Mac.getInstance(HMAC_SHA1);
  mac.init(key);
  byte[] bytes = mac.doFinal(toSign.getBytes(CHARSET));
  return new String(Base64.encodeBase64(bytes)).replace(CARRIAGE_RETURN, EMPTY_STRING);
}
```

```
static class Timer
{
  Long getMilis()
  {
    return System.currentTimeMillis();
  }

  Integer getRandomInteger()
  {
    return new Random().nextInt();
  }
}

private static String getNonce()
{
  Long ts = getTs();
  return String.valueOf(ts + timer.getRandomInteger());
}

private static String getTimestampInSeconds()
{
  return String.valueOf(getTs());
}

private static String getSanitizedUrl(String url)
{
  return url.replaceAll("\\?.*", "").replace("\\:\\d{4}", "");
}

private static Long getTs()
{
  return timer.getMilis() / 1000;
}

private static String asFormUrlEncodedString(Map<String, String> oauthParameters)
{
  if (oauthParameters.size() == 0)
    return EMPTY_STRING;

  StringBuilder builder = new StringBuilder();
  for (String p: oauthParameters.keySet())
  {
    builder.append('&').append(encode(p).concat("=").concat(encode(oauthParameters.get(p))));
  }
  return builder.toString().substring(1);
}

private static String applyRule(String encoded, String toReplace, String replacement)
{
  return encoded.replaceAll(Pattern.quote(toReplace), replacement);
}

private static String encode(String plain)
{
  String encoded = "";
  try
  {
    encoded = URLEncoder.encode(plain, CHARSET);
  }
  catch (UnsupportedEncodingException uee)
  {
    throw new RuntimeException("Charset not found while encoding string: " + CHARSET, uee);
  }
  for (Map.Entry<String, String> rule: ENCODING_RULES.entrySet())
  {
```

```
      encoded = applyRule(encoded, rule.getKey(), rule.getValue());
    }
    return encoded;
  }
}
```

Let us generate the Oauth Header to access the url http://api.twitter.com/oauth/request_token.

```
String requestTokenOauthHeader = HMACSHAUtil.OAUTH1Header("","dummyConsumerKey", "POST",
                                        "http://api.twitter.com/oauth/request_token",
                                         "dummyConsumerSecret", "",null);
```

Send a POST request to the URL with HttpHeader Authorization=requestTokenOauthHeader received above. I used Oracle's Jdeveloper's HTTP Analyzer to send the request. .

The response from the URL will contain oauth_token and oauth_token_secret.

```
oauth_token=3aJJJaXPXWv8F986T9gHhfOMH3RPSYNvD4Pb8Ch33f8Y
 &oauth_token_secret=AIxZUtKby0Qh1wRWGEj8OP81f799eAThm8BNXMNW5U0&oauth_callback_confirmed=true
```

In a browser go to following URL :https://api.twitter.com/oauth/authorize?oauth_token=<token_recieved_in_step_above>
e.g https://api.twitter.com/oauth/authorize?oauth_token=3aJJJaXPXWv8F986T9gHhfOMH3RPSYNvD4Pb8Ch33f8Y

Allow access to Application. If your application has a callback registered, then the request will be redirected to your call back URL as
http://Callbackurl?oauth_token=3aJJJaXPXWv8F986T9gHhfOMH3RPSYNvD4Pb8Ch33f8Y&oauth_verifier=Z05Nn4j78qbnuYnuDQz4iJDDGagasda9

3).Get the Access Token

Now we have to make a request to http://api.twitter.com/oauth/access_token to get the access token.(See docs).

For this let us generate OAUTH header. (using the oauth_token, oauth_token_secret, oauth_verifier obtained earlier)

```
String accessTokenOauthHeader =
    HMACSHAUtil.OAUTH1Header("3aJJJaXPXWv8F986T9gHhfOMH3RPSYNvD4Pb8Ch33f8Y",
                             "dummyConsumerKey", "POST",
                             "http://api.twitter.com/oauth/access_token",
                             "dummyConsumerSecret",
                             "AIxZUtKby0Qh1wRWGEj8OP81f799eAThm8BNXMNW5U0",
                             "Z05Nn4j78qbnuYnuDQz4iJDDGagasda9");
```

Now send a POST request to http://api.twitter.com/oauth/access_token. Make sure you add httpHeader as Authorization : accessTokenOauthHeader returned above.

Response will be something like
auth_token=6253282-eWudHldSbIaelX7swmsiHImEL4KinwaGloHANdrY&oauth_token_secret=2EEfA6BG3ly3sR3RjE0IBSnlQu4ZrUzPiYKmrkVU

Once you have received the access token, you can use till the time its be revoked by the user.

4).Get access to friend's timeline.
You can now make requests to protected resources e.g http://api.twitter.com/1/statuses/friends_timeline.xml (see doc).

Generate the Oauth HTTP Header for the URL http://api.twitter.com/1/statuses/friends_timeline.xml

```
String httpHeaderForFriendsTimeLine =
      HMACSHAUtil.OAUTH1HeaderVersion2("6253282-eWudHldSbIaelX7swmsiHImEL4KinwaGloHANdrY",
                        "dummyConsumerKey",  "GET",
                        "http://api.twitter.com/1/statuses/friends_timeline.xml",
                        "dummyConsumerSecret",
                        "2EEfA6BG3ly3sR3RjE0IBSnlQu4ZrUzPiYKmrkVU",null);
```

Access the URL http://api.twitter.com/1/statuses/friends_timeline.xml passing the Http Header Authorization:httpHeaderForFriendsTimeLine and get the data.

User can apply the same concept when developing Apps for website that uses OAuth (e.g Facebook, Twitter, Flicker etc).