

MISC06 - Portable Class Library oltre ogni limite



Nicolò Carandini

n.carandini@tpcware.com - @tpcware

<http://blogs.ugidotnet.org/Nick60/>

[http://blog\(tpcware.com/](http://blog(tpcware.com/)

Grazie a



Sponsor



Visual Basic tips&tricks



Agenda

- Twitter authentication
- User Sign-in with Twitter
- OAuth 1.0
- Portable Class Library
- The HMAC-SHA1 Cryptography dilemma
- PCL Contrib
- Demo

Twitter authentication (API 1.1)

- ***Application-user authentication***

- Required for “Sign-in”
- User context
- Use OAuth 1.0
- SSL not required

- ***Application-only***

- Use OAuth 2.0
- SSL absolutely required
- No user context



OAuth 1.0

- Identify your request as coming from both the user performing the request *and* your application that's working on behalf of the user
- Provide a "post mark" describing the time the "letter" was sent and the actual contents of the envelope

Access Token
c/o Consumer Key

```
OAuth oauth_nonce="oElnnMTOI2vqvlfXM56aBlAf5noGD0AQR3Fmi706x",
oauth_signature_method="HMAC-SHA1", oauth_timestamp="1272325550",
oauth_consumer_key="GDDmIQH6ihtmlUvng82g", oauth_token="819797-
Ixg8aYUDRmykzVKrgolhXSq67TEa5ruc4GJC2rWimw",
oauth_signature="yOahq5m0YjDDifixHaXEsW9D%2BX0%3D",
oauth_version="1.0"
```

http://api.twitter.com/1/statuses/user_timeline.json



Authorizing a request

```
POST /1/statuses/update.json?include_entities=true HTTP/1.1
Accept: */*
Connection: close
User-Agent: OAuth gem v0.4.4
Content-Type: application/x-www-form-urlencoded
Authorization:
  OAuth oauth_consumer_key="xvz1levFS4wEEPTGEFPHBog",
  oauth_nonce="kYjzVBB8Y0ZFabxSWbWovY3uYSQ2pTgmZeNu2VS4cg",
  oauth_signature="tnnArxj06cWHq44gCs1OSKk%2FjLY%3D",
  oauth_signature_method="HMAC-SHA1",
  oauth_timestamp="1318622958",
  oauth_token="370773112-GmHxMAgYyLbNETIKZeRNFsMKPR9EyMZeS9weJAEb",
  oauth_version="1.0"
Content-Length: 76
Host: api.twitter.com

status=Hello%20Ladies%20%2b%20Gentlemen%2c%20a%20signed%20OAuth%20request%21
```

Request parameters

- **oauth_consumer_key**

The consumer key identifies which application is making the request.

- **oauth_nonce**

Is a unique token your application should generate for each unique request.

- **oauth_signature**

Contains a value which is generated by running all of the other request parameters and two secret values through a signing algorithm.

- **oauth_signature_method**

Always equal to "HMAC-SHA1" for any request sent to the Twitter API.

- **oauth_timestamp**

Indicates when the request was created.

- **oauth_token**

Represents a user's permission to share access to their account with your application.

- **oauth_version**

Always equal to "1.0" for any authorized request sent to the Twitter API.

Creating a signature

- Collect the HTTP request method and Base URL
- Collect parameters (raw values, not the URL encoded) from the URL querystring and the request body
- Include all OAuth parameters
- Encode these values into a single string (following OAuth rules)
- Create the signature base string (following OAuth rules)
- Getting a signing key (Consumer secret and eventually OAuth token secret)
- Calculating the signature (with the HMAC-SHA1 hashing algorithm)

Encode the parameter string

- Percent encode every key and value that will be signed.
- Sort the list of parameters alphabetically by encoded key.
- For each key/value pair:
 - Append the encoded key to the output string.
 - Append the '=' character to the output string.
 - Append the encoded value to the output string.
 - If there are more key/value pairs remaining, append a '&' character to the output string.



Creating the signature base string

- Convert the HTTP Method to uppercase and set the output string equal to this value.
- Append the '&' character to the output string.
- Percent encode the URL and append it to the output string.
- Append the '&' character to the output string.
- Percent encode the parameter string and append it to the output string



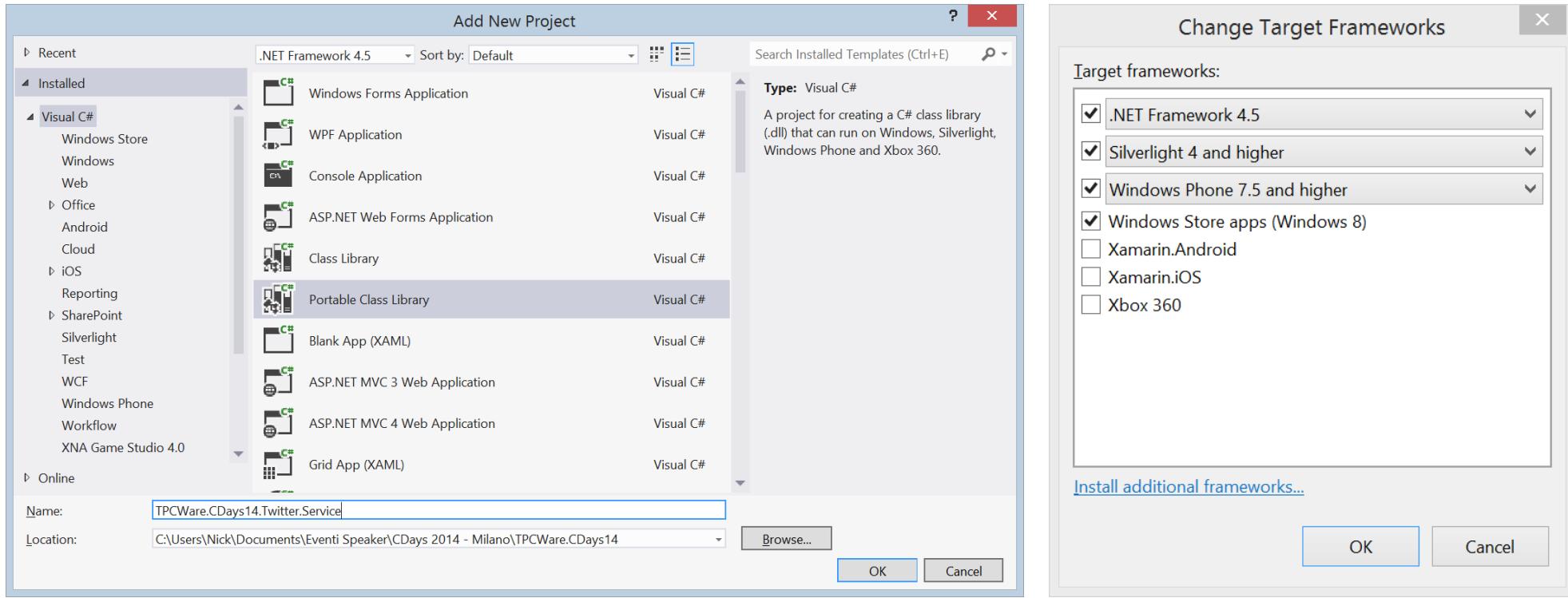
Calculating the signature

- The signature is calculated by passing the signature base string and signing key to the HMAC-SHA1 hashing algorithm.
- The output of the HMAC signing function is a binary string. This needs to be base64 encoded to produce the signature string.



Portable Class Library

- Proviamo a implementare OAuth 1.0 in una Portable Class Library



Redmond, we have a problem!



Non c'è la crittografia tra le librerie delle Portable Class Library!



Portable Class Library Contrib

CodePlex Project Hosting for C# .NET

Portable Class Library Contrib

HOME

SOURCE CODE

Page Info

Change History (all pages)

Project Description

Portable Class Libraries Contrib (PclContrib) is a library that helps you share code between different .NET platforms when using the Portable Class Library feature introduced in Visual Studio 2012, enables you to share common code between different .NET platforms.

The Portable Class Library feature, available in Visual Studio 2012, enables you to share common code between different .NET platforms.

When converting existing projects over to Portable Class Libraries, you may find that some APIs are not provided by every platform. PclContrib provides versions and adapters of these APIs enabling you to share common code between different .NET platforms.



How-To PCL Contrib (step 1)

- Scaricare la solution PclContrib da <http://pclcontrib.codeplex.com>
- Creare nella nostra solution una cartella PclContrib e copiarci dentro i seguenti file dalla cartella Bin/Releases:
 - Portable.Runtime.dll
 - Portable.Reflection.dll
 - Portable.Data.dll
 - Portable.Security.Cryptography.dll
 - Portable.Phone.dll
 - Portable.Store.dll
 - Portable.Desktop.dll
 - Portable.Silverlight.dll



How-To PCL Contrib (step 2)

- Aggiungere al nostro progetto PCL i riferimenti a:
 - Portable.Runtime.dll
 - Portable.Security.Cryptography.dll
- Creare un progetto Windows Phone 7.1 (oppure 8.0)
- Aggiungere al nostro progetto WP i riferimenti a:
 - <nostro progetto PCL>
 - Portable.Phone.dll



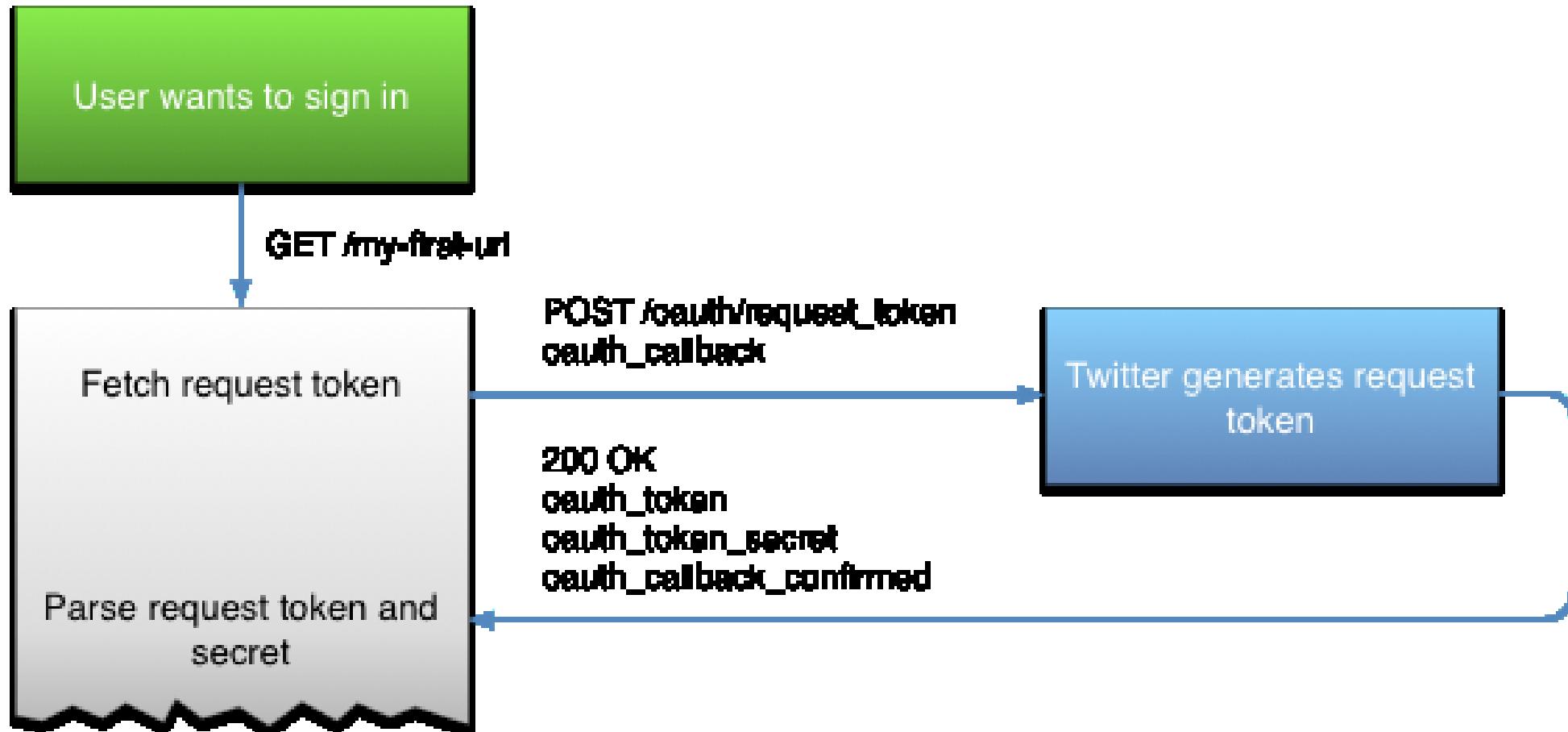
demo

OAuth 1.0



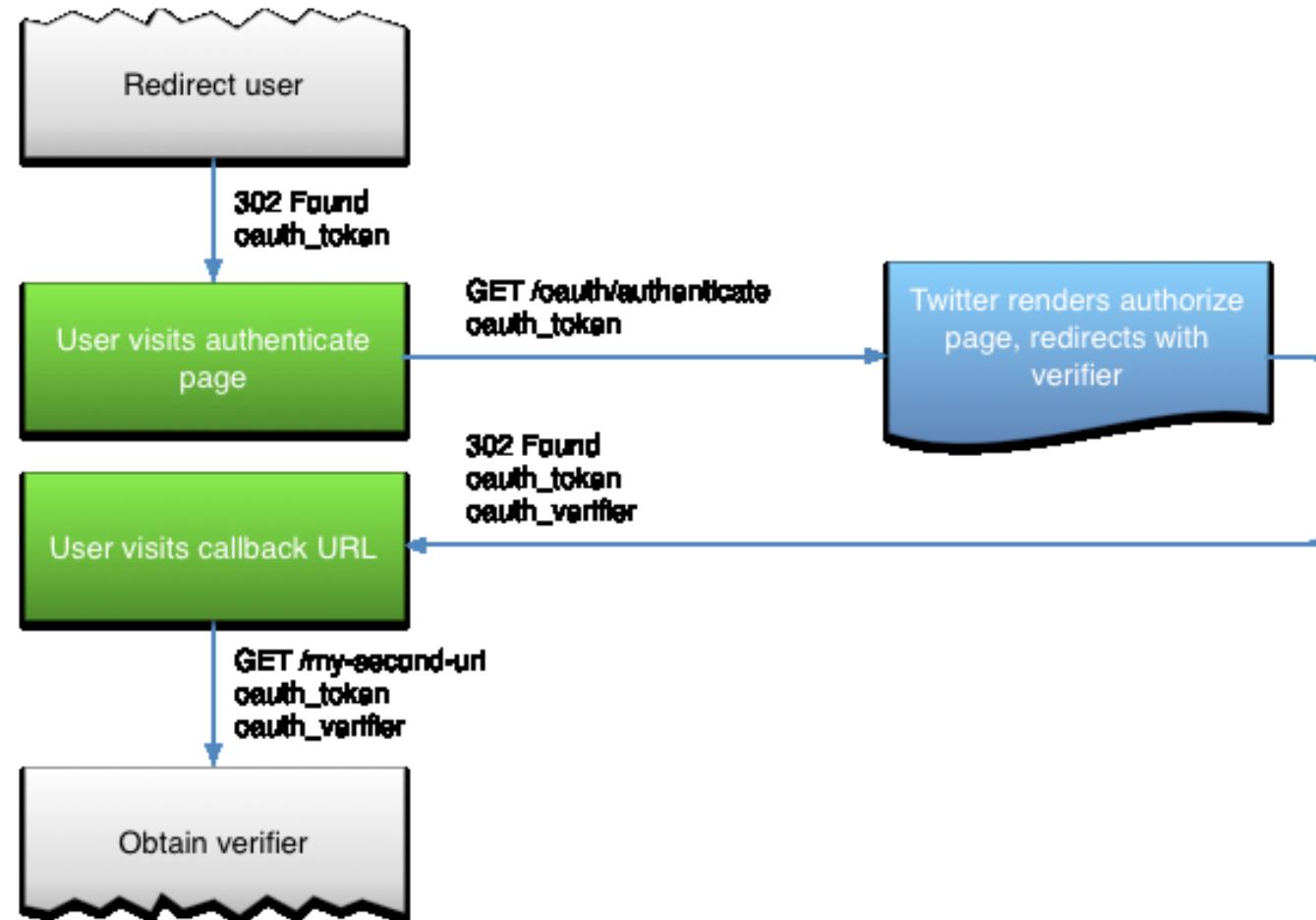
User Sign-in with Twitter

- *Step 1: Obtaining a request token*



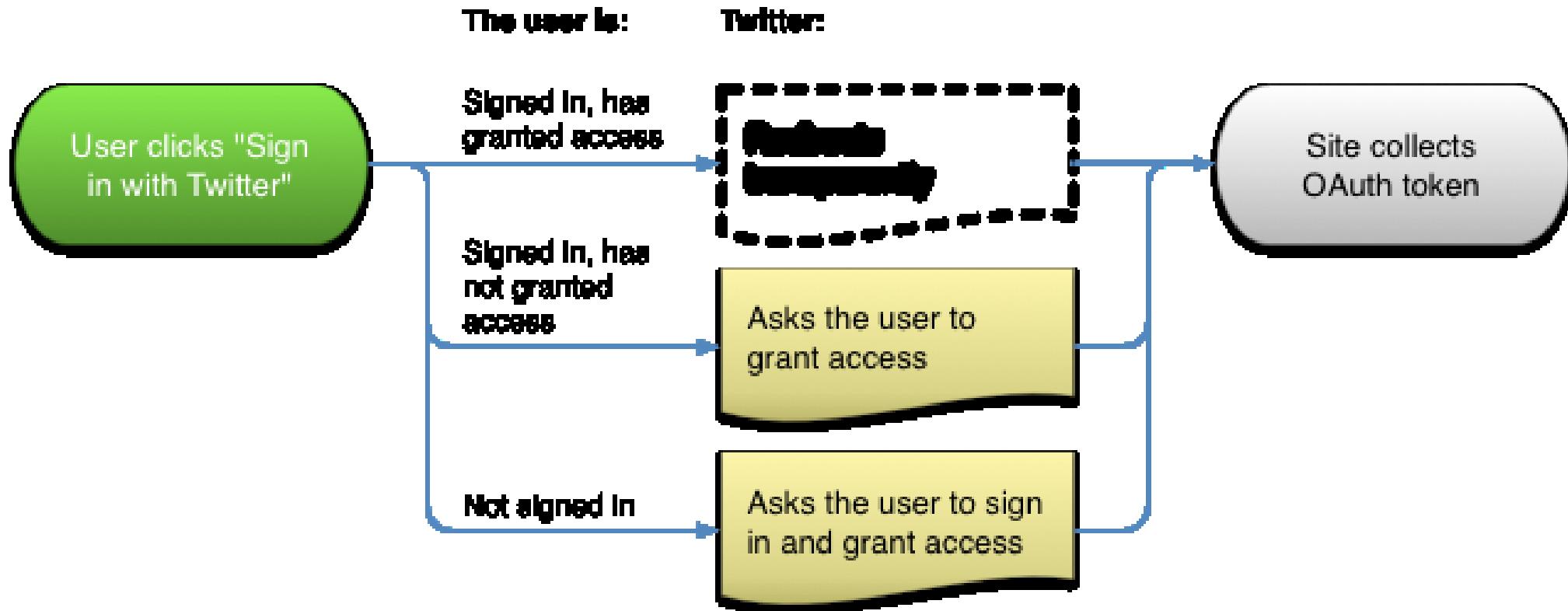
User Sign-in with Twitter

- *Step 2: Redirecting the user*



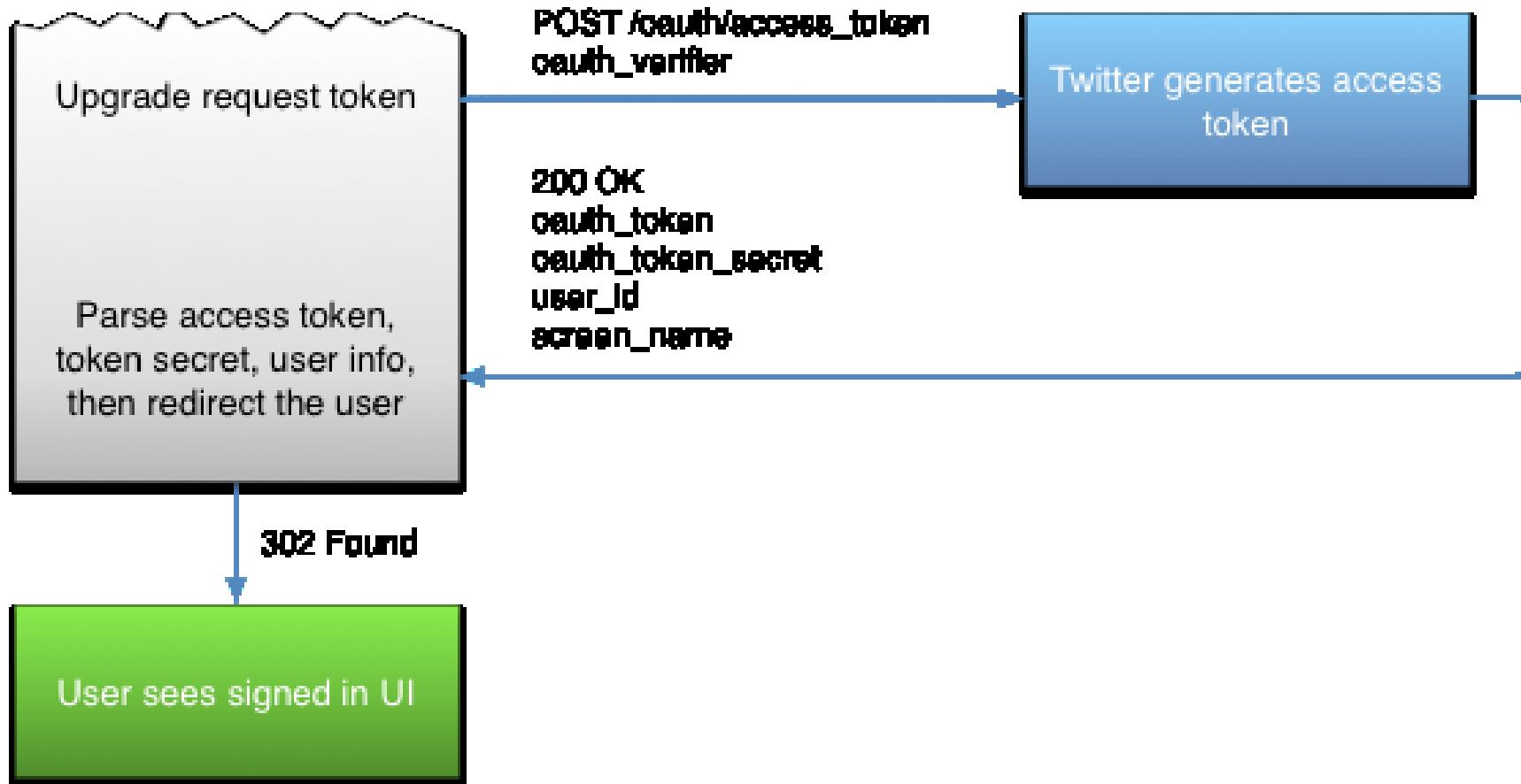
User Sign-in with Twitter

- *Step 2: possible results*



User Sign-in with Twitter

- *Step 3: Converting the request token to an access token*



demo

Twitter Sign-in



Q&A

Tutto il materiale di questa sessione su
<http://www.communitydays.it/>

Lascia il feedback su questa sessione,
potrai essere estratto per i nostri premi!

Seguici su

Twitter @CommunityDaysIT

Facebook <http://facebook.com/cdaysit>

#CDays14

