Token Based Authentication

Tokens

A single element of a programming language is known as a Token. A token could be a operator, operand, keywords and punctuation marks.

Ex: a+b=c

Here,

a,b = tokens

=,+ = tokens

In security systems, a [hard token](http://techterms.com/definition/hardtoken) is small card that displays an identification code used to log into a network. When the card user enters the correctpassword, the card will display the current ID needed to log into the network. This adds an extra level of protection to the network because the IDs change every few minutes. Security tokens also come in software versions, called [soft tokens](http://techterms.com/definition/softtoken).

Hard Tokens

* A hard token, sometimes called an "authentication token," is a hardware security device that is used to authorize a user. An common example of a hard token is a security card that gives a user access to different areas of building or allows him to log in to a computer system.
* Some hard tokens are used in combination with other security measures to further enhance security. For example, a username and password or a fingerprint scan may be required along with the hard token to gain access to a secure system.

Soft Tokens

* A soft token is a software-based [security token](http://searchsecurity.techtarget.com/definition/security-token) that generates a single-use login PIN.
* Soft tokens are an attempt to replicate the security advantages of authentication, while simplifying distribution and lowering costs. A smartphone soft token app performs the same task as a hardware-based security token.
* Like a hardware token, a smartphone provides an easy-to-protect and easy-to-remember location for secure login information on the device itself.
* Unlike a hardware token, smartphones are connected devices, which make them inherently less secure. The extent of their security largely depends on the devices operating system and client software.

Authentication

* The process of identifying an individual, usually based on the username and password.
* Authentication process just confirms that the credential is valid and is in the hands of the owner of the credential. Credential authentication does not reveal or confirm identity to the government website. It receives only a message confirming that your credential (username/password) was successfully validated by your bank.
* Authentication merely ensures that the individual is who he or she claims to be, but says nothing about the access rights of the individual.

Authorization

* It is the process of giving individual session[s](http://www.webopedia.com/TERM/A/access.html) to system objects based on their [identity](http://www.webopedia.com/TERM/I/identity.html).
* Authorization is the process of giving someone permission to do or have something. In multi-user computer systems, a system administrator defines for the system which users are allowed access to the system and what privileges of use.
* Assuming that someone has logged in to a computer operating system or application, the system or application may want to identify what resources the user can be given during this session.

Token Based Authentication

Token based authentication is prominent everywhere on the web nowadays. With most every web company using an API, tokens are the best way to handle authentication for multiple users.

There are some very important factors when choosing token based authentication for your application. The main reasons for tokens are:

1. Stateless and scalable servers

2. Mobile application ready

3. Pass authentication to other applications

4.Extra security

Any major API or web application that we have come across has most likely used tokens. Applications like Facebook, Twitter, Google+, GitHub, and so many more use tokens.

Token based authentication is **stateless**. We are not storing any information about our user on the server or in a session. It takes care of many of the problems with having to store information on the server.

No session information means your application can scale and add more machines as necessary without worrying about where a user is logged in.

This implementation can be done by following ways:

1. User Requests Access with Username / Password
2. Application validates credentials
3. Application provides a signed token to the client
4. Client stores that token and sends it along with every request
5. Server verifies token and responds with data

Every single request will require the token. This token should be sent in the HTTP header so that we keep with the idea of stateless HTTP requests. We will also need to set our server to accept requests from all domains using Access-Control-Allow-Origin.

Here’s an infographic to explain the process:

Once we have authenticated with our information and we have our token, we are able to do many things with this token.

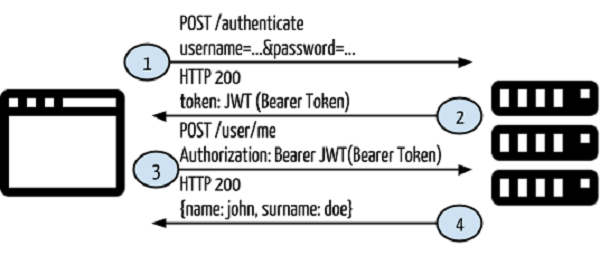
We could even create a permission based token and pass this along to a third-party application and they will be able to have access to our data.

In token-based authentication, cookies and sessions will not be used. A token will be used for authenticating a user for each request to the server.

It will use the following flow of control:

1. The user provides a **username** and **password** in the login form and clicks **Log In**.
2. After a request is made, validate the user on the backend by querying in the database. If the request is valid, create a token by using the user information fetched from the database, and then return that information in the response header so that we can store the token browser in local storage.
3. Provide token information in every request header for accessing restricted endpoints in the application.
4. If the token fetched from the request header information is valid, let the user access the specified end point, and respond with JSON or XML.

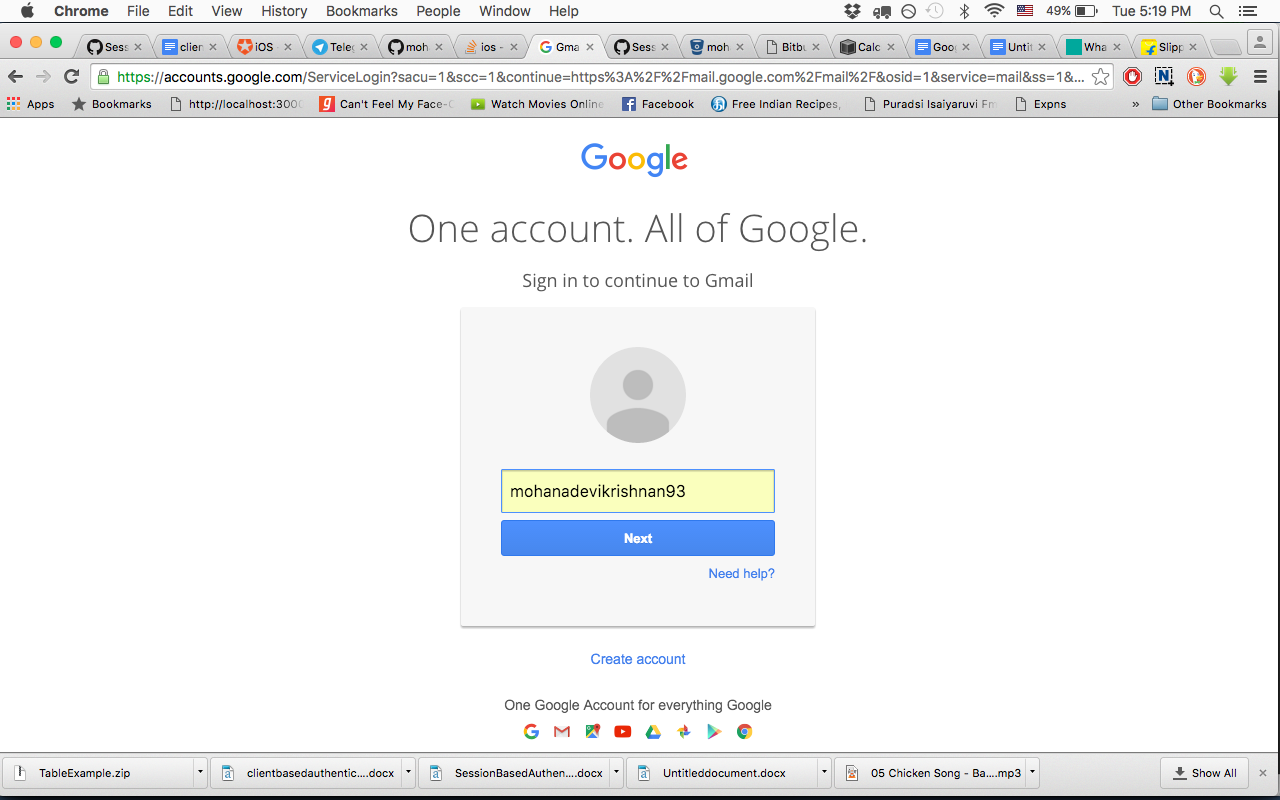
In this case, we have no returned session or cookie, and we have not returned any HTML content. That means that we can use this architecture for any client for a specific application. Architecture schema is explained in below.

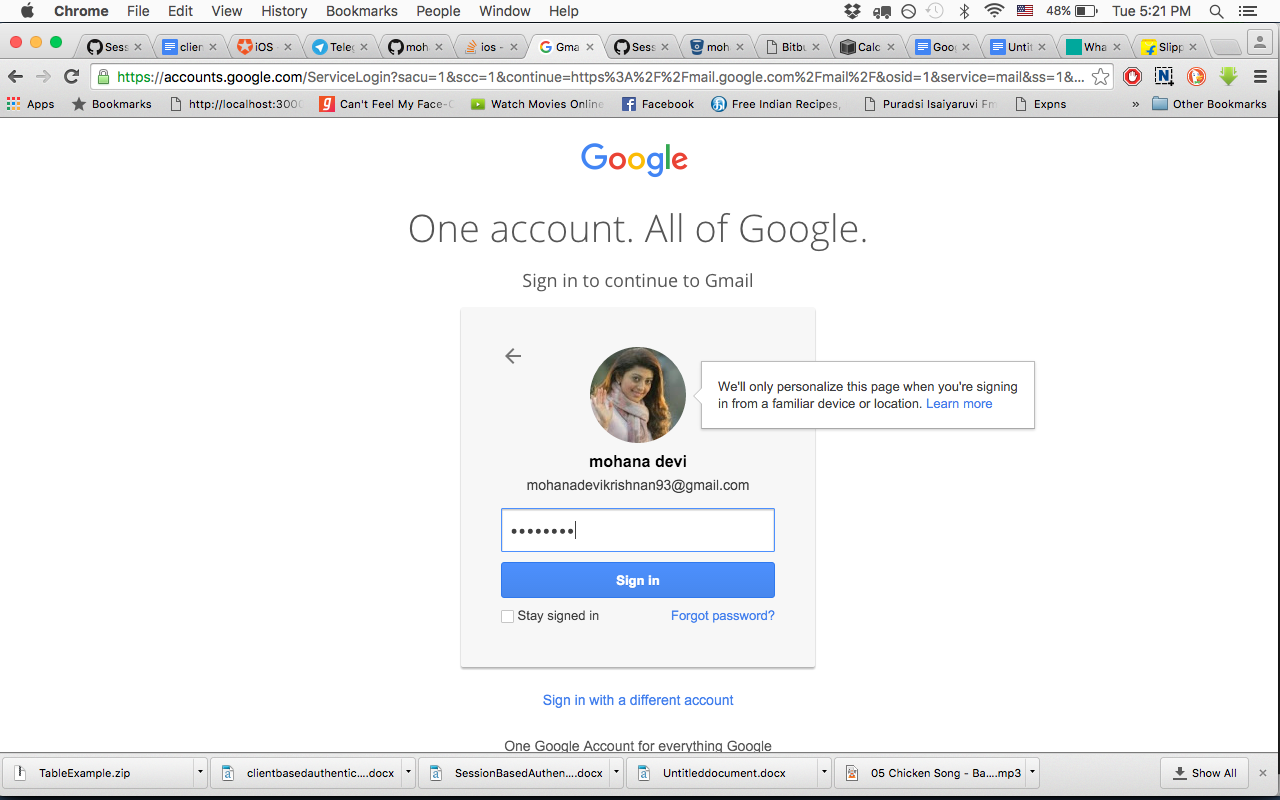


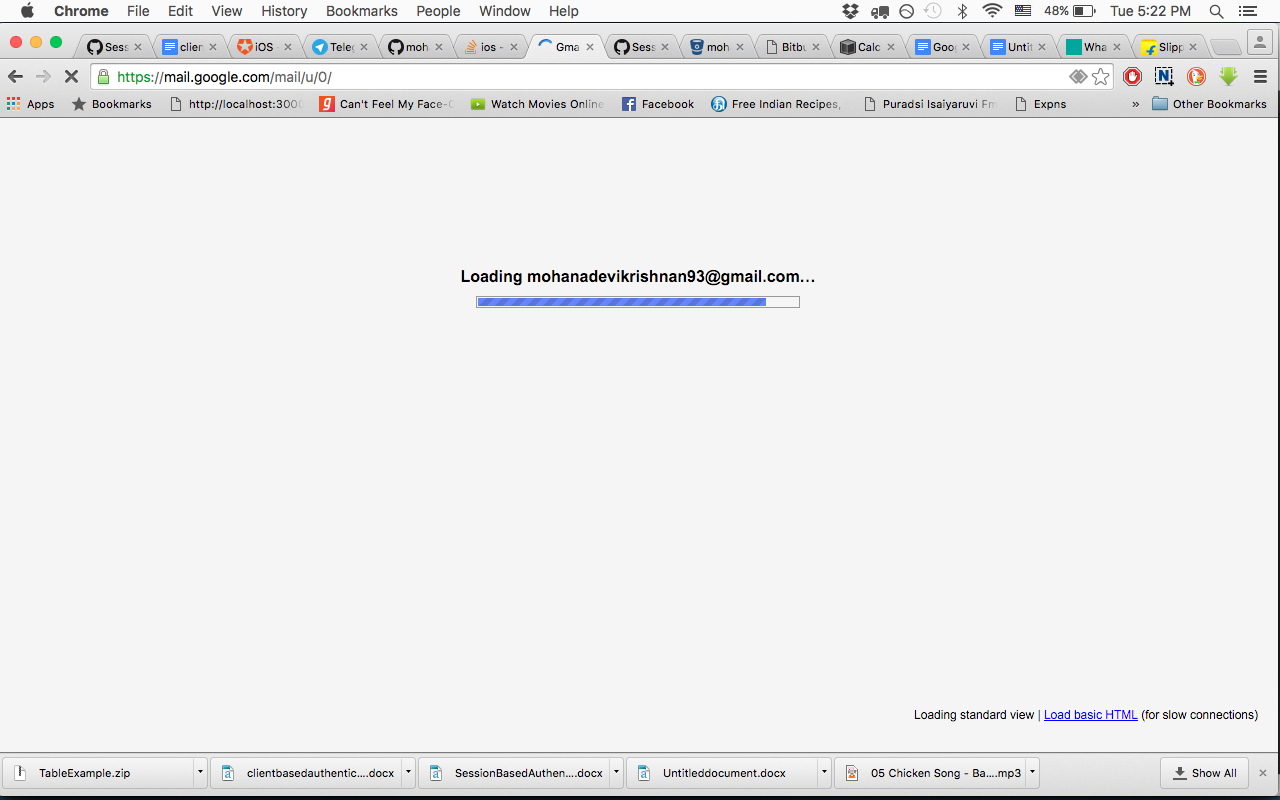
For ex. If we take a gmail account, it is also based on the client based authentication. It is explained by using the following example.

Step 1:

Give a username and password







Benefits

1. Stateless and scalable
2. Security
3. Extensibility
4. Multiple platforms and domains
5. Based on standards

Implementation