Contents

[Introduction 2](#_Toc474698044)

[Json Web Token 2](#_Toc474698045)

[C’est quoi JWT? 2](#_Toc474698046)

[Quand on l’utilise? 2](#_Toc474698047)

[Structure du JWT 3](#_Toc474698048)

[Fonctionnement du JWT 3](#_Toc474698049)

[OAUTH2 3](#_Toc474698050)

[Introduction 3](#_Toc474698051)

[Composants 3](#_Toc474698052)

[Flux 4](#_Toc474698053)

[Oauth2 et authentification 5](#_Toc474698054)

[OpenID connect 5](#_Toc474698055)

[Introduction 5](#_Toc474698056)

[Flux 5](#_Toc474698057)

[Configuration de la démonstration 5](#_Toc474698058)

[Installer Netbeans 8.2 5](#_Toc474698059)

[Installer MySQL database 5](#_Toc474698060)

[Configuration du Google API 6](#_Toc474698061)

[Installer Tomcat, serveur d’application 6](#_Toc474698062)

[Java Web application 7](#_Toc474698063)

# Introduction

Ce document représente un ensemble de nouvelles technologies qui sont conçues pour travailler ensemble et construire la base de sécurité pour écrire des applications modernes.

Les directions des sécurités en train de changer, car les gens utilisent de plus en plus les appareils mobiles qui en quelques sorte supportent les protocoles lourds comme SAML et WS star (ces protocoles sont utilisés dans les entreprises « business to business » et rarement pour les « business to customer »)

Ces protocoles sont des protocoles d’entreprises, de base XML, compliqués, et de couplage étroit (tightly coupled). Cependant, ils ne sont pas la meilleur solution pour ces appareils, pour cela la communauté technique, et pour réduire la complexité d’utilisation de ces protocoles, a inventé des protocoles légers comme l’OAUTH2 et OPENID Connect.

# Json Web Token

## C’est quoi JWT?

JSON Web Token (JWT) est un standard ouvert qui définit un moyen compact et autonome de transmission sécurisée d'informations entre les parties en tant qu'objet JSON. Ces informations peuvent être vérifiées et approuvées car elles sont signées numériquement. Les JWT peuvent être signés à l'aide d'une clé secret (avec l'algorithme HMAC) ou d'une paire de clés publiques ou privées en utilisant RSA.

**Compact**: En raison de leur taille plus petite, les JWT peuvent être envoyés via une URL, un paramètre POST ou à l'intérieur d'un en-tête HTTP. Plus la taille est petite signifie que la transmission est rapide.

**Self-Contained**: La charge utile contient toutes les informations requises sur l'utilisateur, évitant de devoir interroger la base de données plus d'une fois.

Donc, le but d'un jeton de sécurité (Token) est vraiment de produire une structure de données qui contient des informations sur l'émetteur et peut-être sur le destinataire et sur le sujet que ce jeton décrit.

## Quand on l’utilise?

JWT est utilisé dans l’authentification et dans l’échange d’information:

**Authentification :** c’est-à-dire, pour que l’utilisateur soit valide par l’application, le token récupéré par le serveur de ressource contient toutes les informations nécessaires de l’utilisateur connecté.

**Echange d’information :** d’une manière sécurisée, car le token récupéré sera signé et les expéditeurs peuvent désigner le message en utilisant une clé publique ou privée, et est alors sûre que le contenu n’a pas été altéré.

## Structure du JWT

JWT se compose de 3 parties séparées par un point, qui sont:

* Entête (Header)

Entête se compose généralement de deux parties: le type du jeton, qui est JWT, et l'algorithme de hachage utilisé

* Charge Utile (Payload)

La 2ème partie contient les réclamations (claims), qui sont les informations de l’utilisateur et les métadonnées supplémentaires.

* Signature

Pour créer la partie signature, vous devez prendre l'entête codée, la charge utile encodée, une clé secret, l'algorithme spécifié dans l'entête, et signer cela.

Par exemple, si vous souhaitez utiliser l'algorithme HMAC SHA256, la signature sera créée de la façon suivante:  
RSASHA256( base64UrlEncode(header) + "." + base64UrlEncode(payload))

## Fonctionnement du JWT

L'utilisateur utilise le web browser pour être authentifié, alors L'utilisateur envoie une demande de « post » qui contient ses informations d'identification, (Mot de passe et nom d’utilisateur). Le serveur répond et génère le token et le retourne au l’utilisateur.

Ce token est sauvegardé dans un stockage local comme le cookie. Après ça et durant chaque demande (request), token sera envoyé dans l’autorisation « Header » et c’est le rôle du serveur pour vérifier s’il est authentifié, et envoyer alors la réponse.

# OAUTH2

## Introduction

Oauth2 est un protocole pour l’autorisation, il distingue les clients et les propriétaires de ressources.

L’autorisation, c’est à dire obtenir les permissions nécessaires d’accéder les services ou bien d’exécuter n’importe qu’elle fonction (Ajouter, Modifier..), mais l’utilisateur connecté n’est pas vérifié par l’application elle-même.

Exemple: valley parking ; Le chauffeur avait la clé de la voiture et il peut la conduire mais la voiture ne peut pas vérifier l’identité du chauffeur.

## Composants

Oauth2 est définit par quatre rôles:

**Propriétaire de la ressource** *(Resource owner)***,** Une entité capable d'accorder l'accès à une ressource protégée. Lorsque le propriétaire de la ressource est une personne, il est désigné comme un utilisateur final. (Navigateur web, IE, chrome).

**Serveur de ressources** *(Resource server*),Le serveur hébergeant les ressources protégées capable d'accepter et de répondre aux demandes de ressources protégées à l'aide de jetons d'accès.

**Application client** *(Client Application)* effectuant des demandes de ressources protégées au nom du propriétaire de la ressource et avec son autorisation. Le terme «client» n'implique aucune caractéristique d'implémentation particulière (ex. navigateur, ..)

**Serveur d'autorisation** *(Authorization server)* Le serveur émet des jetons d'accès au client après avoir authentifié avec succès le propriétaire de la ressource et obtenu l'autorisation. L'interaction entre le serveur d'autorisation et le serveur de ressources dépasse la portée de cette spécification.

Le serveur d'autorisation peut être le même que le serveur de ressources ou une entité distincte. Un serveur d'autorisation unique peut délivrer des jetons d'accès acceptés par plusieurs serveurs de ressources.

Le propriétaire de la ressource utilise un client pour accéder à la ressource en son nom. Pour cela, le client doit avoir une clé d'accès (ou un jeton d'accès). Cette clé d'accès limitée est donnée par le serveur d'autorisation.

## Flux

Oauth2 possède 2 types de flux :

* **Flux avec interaction utilisateur**
  + Flux du code d'autorisation
    1. Demande d'autorisation
    2. Demande de jeton
    3. Accès à la ressource
  + Flux implicite
    1. Demande de jeton
    2. Accès à la ressource
* **Flux sans interaction de l’utilisateur**
  + Flux du mot de passe du propriétaire de ressource
    1. Demandez un jeton avec les informations d'identification du propriétaire de ressource
    2. Accès à la ressource
    3. C'est ce qui se rapproche des protocoles d'entreprise - comme WS-Trust.
    4. Le client lui-même recueille les informations d'identification de l'utilisateur (c'est-à-dire le mot de passe ou la clé principale). Le client utilise alors cette clé pour accéder à la ressource.
  + Flux du mot de passe du client
    1. Communication client-service
    2. Demande de jeton avec les informations d'identification du client
    3. Accès à la ressource
    4. Aucun détenteur de ressources n'est impliqué

## Oauth2 et authentification

Comme on a mentionné, Oauth2 est pour l’autorisation, et il n'a pas une manière standard d'identifier les utilisateurs aux applications, mais les développeurs ont l’avait utilisé comme une mauvaise technologie d’authentification. Mais comment on peut identifier l’utilisateur dans l’application en ordre d’authentifier l’utilisateur d’accéder ou limiter l’accès aux fonctionnalités de l’application ?

La solution était par l’ajout d’un nouveau paramètre dans le jeton (token) qui est le tokenId, cette technologie s’appelle « OpenID Connect ».

# OpenID connect

## Introduction

OpenId connect complète ouath2, c’est une couche d'identité simple au-dessus du protocole OAuth2. Il permet aux clients de vérifier l'identité de l'utilisateur final en fonction de l'authentification effectuée par un serveur d'autorisation, ainsi que d'obtenir des informations du profil de base sur l'utilisateur final de manière interopérable.

## Flux

C’est la même comme l’oauth2 mais il différencie dans l’étape où nous récupérons la réponse du jeton. Il apparait alors ici la présence d'un nouveau jeton "id\_token". Ce jeton est destiné au client; Il est utilisé par le client pour identifier (c'est-à-dire authentifier) l'utilisateur. Le client doit alors valider cet id\_token.

# Configuration de la démonstration

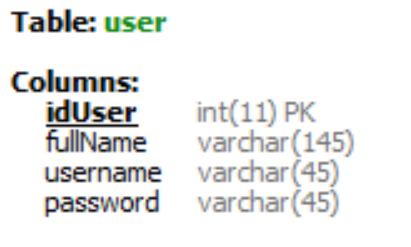
Cette application est construite sur le Platform java EE, dans l’environnement du développement NetBeans 8.2. Pour que cette application fonctionne, il faut suivre les étapes suivantes :

## Installer Netbeans 8.2

1. Visiter le web site <https://netbeans.org/community/releases/82/install.html> pour installer IDE.

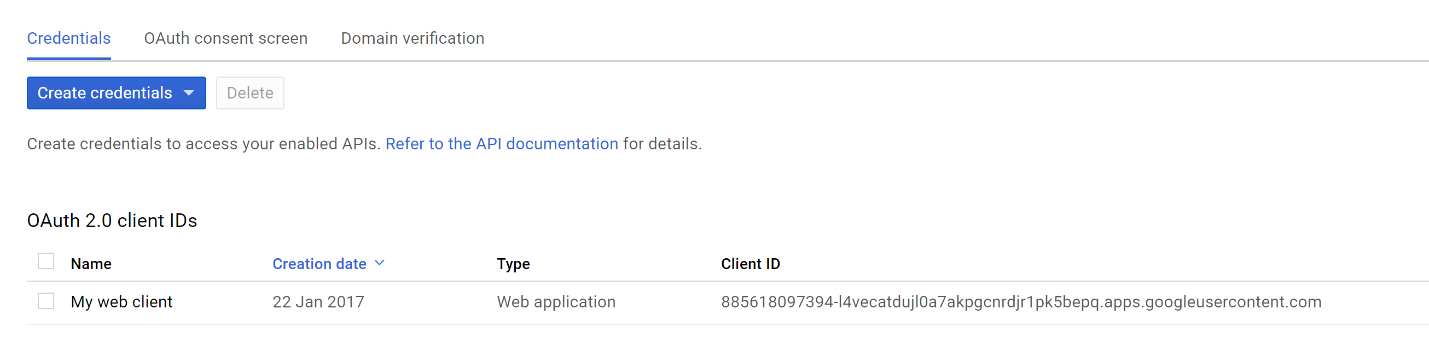
## Installer MySQL database

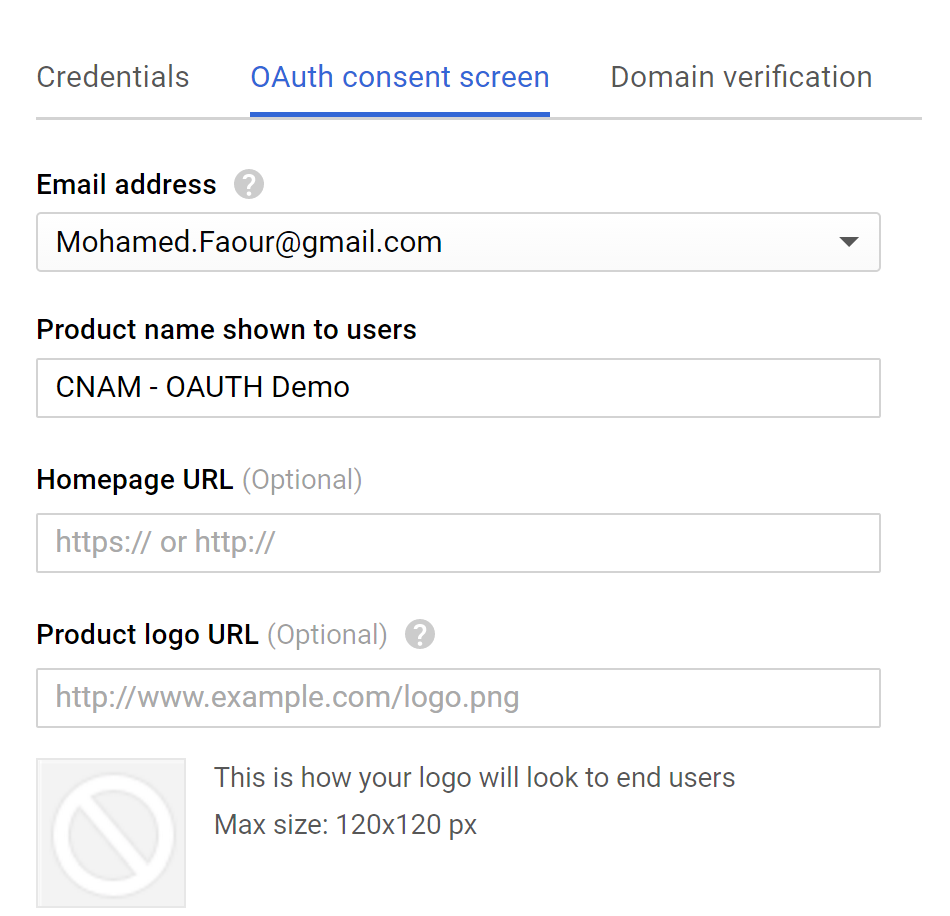
1. Installer MySql du site suivant : <https://dev.mysql.com/downloads/installer/>
2. Créer une base de données nommeée customouath
3. Créer un nouveau tableau « user » qui contient 4 colonnes (iduser, fullName,username, password) comme ci-dessous :



## Configuration du Google API

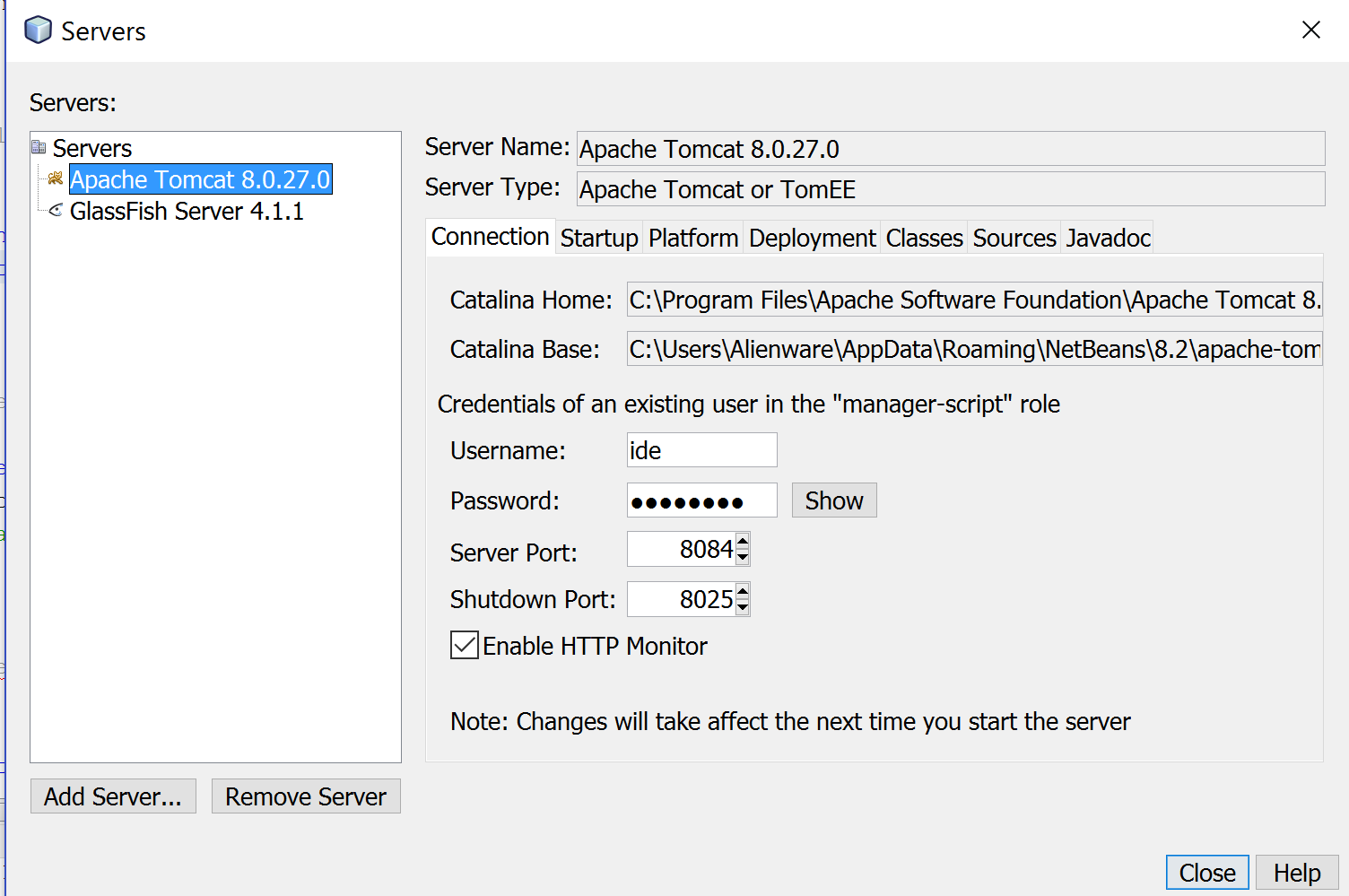
1. Télécharger les librairies Google du site suivant : <https://developers.google.com/identity/protocols/OAuth2>
2. Créer une application dans google comme ci-dessous :
   * Accédez à l'url suivante <https://console.developers.google.com/apis/credentials>
   * Créer un nouveau « OAuth Client » Id et sélectionner « web application »
   * Entrez votre URL de redirection dans la page « Credentials »
   * Changer le champ « Product name » pour être « CNAM-OAUTH Demo » dans la page OAUTH consent screen.
   * Apres vous aurez un Client ID et Secret Code.
   * Ces paramètres doivent être utilisés dans l’application.





## Installer Tomcat, serveur d’application

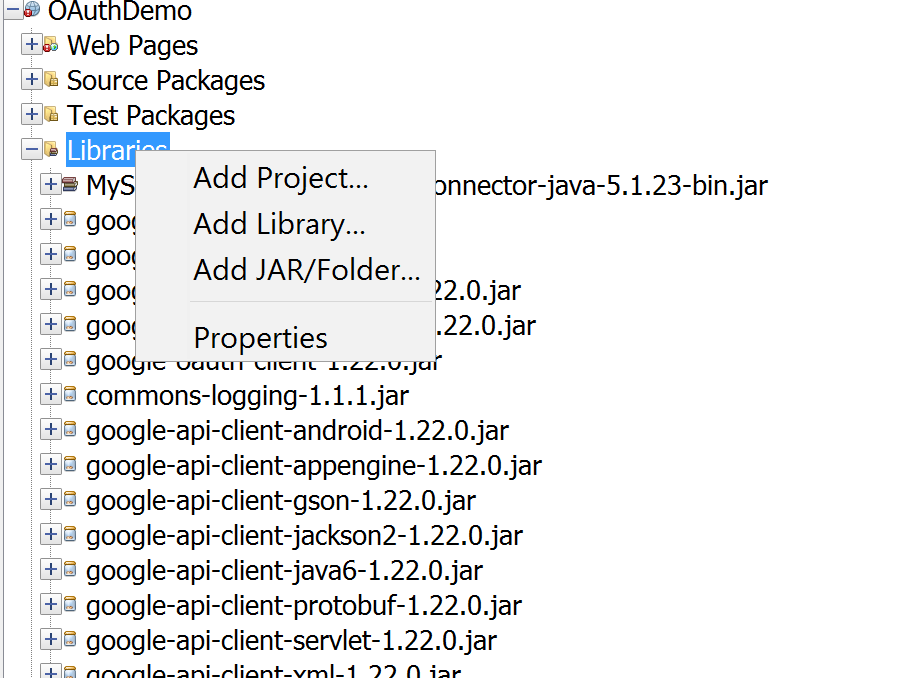
1. Ajouter tomcat à netbeans : <http://wiki.netbeans.org/AddExternalTomcat>
2. Pour vérifier la configuration, cliquer tools>> servers
3. Il faut voir tomcat dans la liste comme ci-dessous :

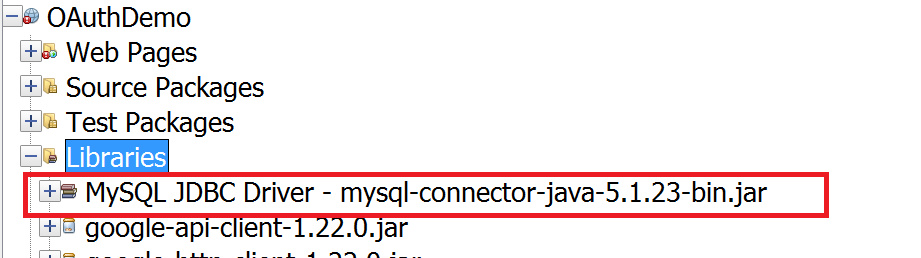


## 

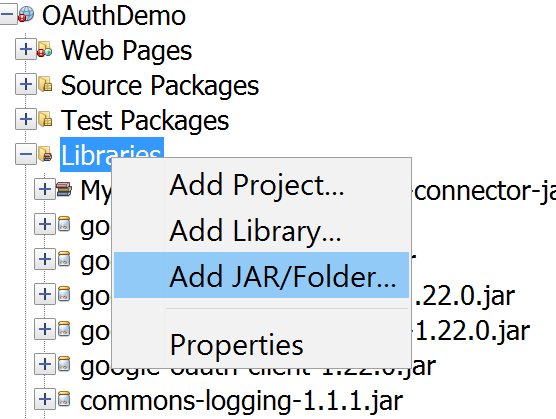
## Java Web application

1. Ouvrir Netbeans IDE.
2. Obtenir la dernière version de github, <https://github.com/mfaour/OauthDemo>
3. Ajouter la librairie MySQL JDBC driver au projet





1. Ajouter la librairie google (jar files) au projet (fichiers téléchargés du site google)



1. Dans le fichier login data access object (logindao.java) modifier les variables URL, username et le password pour connecter au MySQL database.
2. Exécuter le projet ou appuyer sur F6
3. Le navigateur s'ouvrira et affichera la page suivante

