OmniAuth

# Qu’est-ce que c’est OmniAuth ?

Aujourd’hui nous avons plein de sites à s’inscrire. Ce n’est vraiment pas possible de mémoire tous les noms d’utilisateur et mots de passe. Donc le site offert une autre façon de s’inscrire en utilisant un compte d’un autre site comme Google, Facebook, twitter etc. En ce moment-là, OmniAuth joue le rôle d’obtenir l’authentication d’autre site (authentication fournisseur) pour le nouveau compte.

« OmniAuth est une librairie qui standardise multifournisseur authentication pour web les applications. Il est une Ruby authentication framework qui sert à simplifier de »

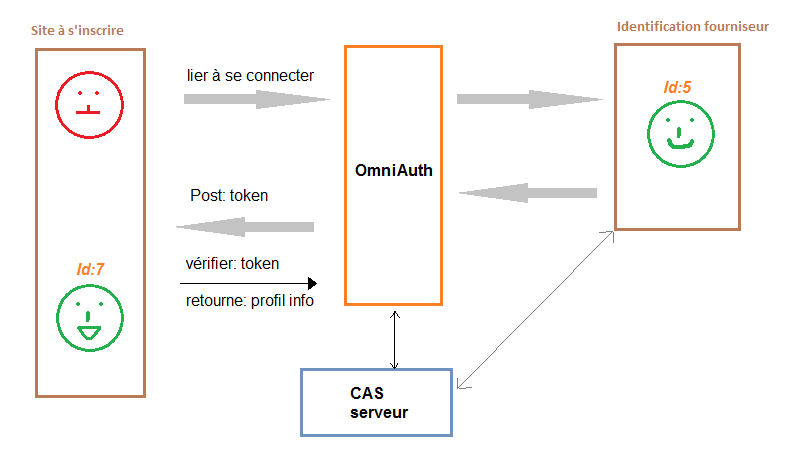
# Qu’est-ce que c’est Authentication ?

Authentication est un identification token/ticket.

Un site peut avoir plusieurs authentication fournisseurs. Un utilisateur peut avoir plusieurs authentications.

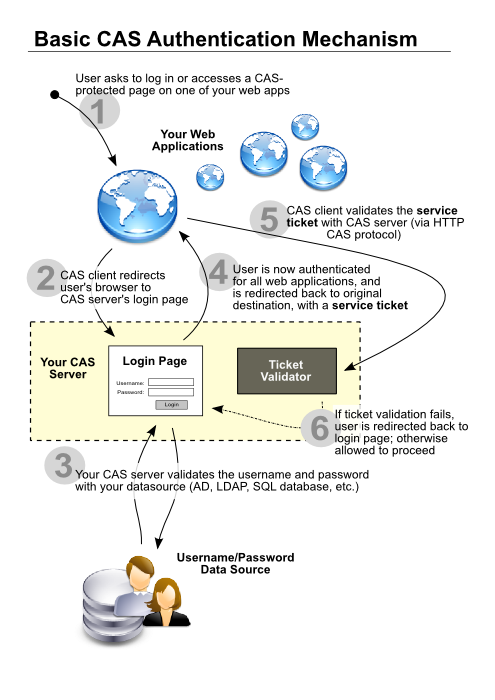
# Comment ça fonctionne ?

OmniAuth manie l’autorisation pour la 3ème partie. Par exemple, si tu veux ton application à intégrer avec Twitter. Tu veux accéder quelque 3ème partie service ou ressource comme un utilisateur sans mdp. L’application demande Twitter pour l’accès. L’utilisateur autorise (sur Twitter) puis l’app a l’accès.



# CAS Authentication Mécanisme

Source lien : https://rubycas.github.io/

CAS centralise authentication qui permet de s’inscrire un serveur unique(toutes les applications sont configurées à un serveur unique). Mais il ne manie pas l’autorisation. OmniAuth manie l’autorisation pour la 3ème partie.

Ruby on Rails

# Introduction

Rails est un framework de développement web écrit en Ruby. Les contraints du RoR sont:

1. *Le DRY(Don’t Repeat Yourself) :* Rails réduit le code duplicatif
2. *Convention over configuration* : Rails se base uniquement sur des conventions, non sur de la configuration.
3. ***REST*** : c’est le meilleur design pattern pour le développement d’applications web. Il permet d’organiser les application autours de verbes HTTP standards.

Les environnement du RoR: Ruby, Rubygems, SQLLite/MySQL/PostgrSQL

# Composants de Rails

## Active Controller

Il gère les contrôleurs de l’application. Les services rendus par le composant sont la gestion de session, le rendu de templates, et la gestion de redirection.

## Active View

Gère la partie vue de vos applications. Il permet de créer du HTML et du XML. Les

## Active Record

C’est la base de modèles dans l’application Rails. Il fournit l’indépendance vis-à-vis des base de données, la base des fonctionnalités CRUD( Create-Retrieve-Update-Delete), et la capacité à relier les modèles entre eux par exemple.

## Active Resource

Ce composant fournit un framework pour gérer la connexion entre des objets métier et des services web RESTful. Il implémente une façon de mapper des resources web à des objets locaux avec une sémantique CRUD.

## Active Support

C’est une collection extensive de classes utilitaires et d’extensions Ruby standards utilisées dans l’application.

## Modèle MVC

Ruby on Rails est un MVC Framework complet qui peut séparer efficacement la manière de transfert de ressource et la base de données.

* Controller : le Controlller reçoit le résultat que le Route analyse, détermine laquelle controller va être appliquée et mettre le controller à exécuter certain action. C’est-à-dire, un « Http Request » est envoi par l’utilisateur(Browser) à « Routing ». Un fichier « route.rb »(Routes) assigne certain « Action Controller ». Après l’action du controller va être appelée. En le même temps, le controller exchange des données avec Model et View.

**Dispatcher**

**Routes**

**Web Server**

**Browser**

**Controller**

**View**

**Model**

**MySql**

HTTP

HTML

Html template est rempli de données de controller

Html rempli est retourné au Browser

* View : View est la page HTML qui est remplie des données via controller.
* Model : il fournit une interface de sauvegarder et interroger la base de données.

# P:\Jing\rails-flow-mapping.pngMapping URLs du Rails

Figure 1: Mapping flow du Rails

L’url cartographie à ses actions respectivement comme le fig. 2 . Chaque url est chargé par une action du controller. L’action est une méthode qui manie les urls, les interactions avec la base de données et rendre une HTML page.

Dans l’exemple du fig.2, on a des actions index, add, mark, unmark, et delete du controller

## REST

Les principes de base de REST sont les suivants:

1. l’utilisation d’URL pour représenter des ressources.
2. l’utilisation de verbes HTTP pour toutes les opérations nécessaire(GET, POST, PUT, et DELETE)
3. l’utilisation des standards HTML et XML

# Projet Rails

## le répertoire du projet Rails

* **app/** : contient le code Rails de l’application(modèles, vues, contrôleurs, assistants)
* **config/** : contient le code de configuration de l’application
* **db/** : contient le code de description de la base de données
* **doc/** : contient la documentation du code de l'application
* **lib/** : contient les modules d'extensions éventuels de l'application
* **log/** : contient les logs de l'application
* **public/** : contient les fichiers statiques de l'application, accessibles aux navigateurs web (images, javascript, css)
* **script/** : contient les scripts Rails nécessaires au développement
* **test/** : contient les tests unitaires de l'application
* **tmp/** : contient les fichiers temporaires générés par l'application
* **vendor/** : contient les gems et plugins
* **README**: une brève description de l’application
* **Rakefile**: tâche utiles accessibles par le biais de la commande rake
* **Gemfile** : les gems requis(les dépendances) par l’application