In this lecture, we will discuss…

- What is OAuth 2
- Why use OAuth 2
- OAuth 2 data flow

What is OAuth 2

- OAuth stands for “Open Authorization”
- Open standard protocol that provides simple and secure authorization for different types of applications
- Allows providers to give access to users without any exchange of credentials
- *Authorization framework that enables applications to obtain limited access to user accounts on an HTTP service like Facebook, GitHub, Twitter etc…..*

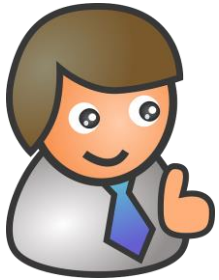
Password Approach: Problems

- Users have to share credentials
- Not secure and intrusive
- Hard to maintain when you authorize many apps
  - Change password at provider (Facebook) – clients need to be updated

OAuth 2 Approach

- Secure as no passwords are exchanged
- Uses tokens (next slide)
- Allows providers to give access to users without any exchange of credentials

OAuth 2 in Action: Movie Service



Mr. Ideas

MovieEditor Web Application

1. REGISTER APPLICATION

Movie Service

OAuth 2 in Action: Movie Service

MovieEditor Web Application

New Application

Name

Movie Client

Redirect uri

http://acme.com/auth/movies/callback

Use one li

Scopes

Separate scopes with spaces.

Submit Cancel

OAuth 2 in Action: Movie Service

MovieEditor Web Application

Application: Movie Client

Application Id:

4968cb91aa46a93eaaafb939370b013466152154220b02ac7fc455b78f1df3be

Secret:

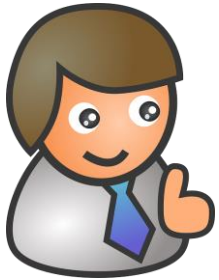
ec3743ecba039850473950dc141a1a7a59a5465487d262b62e7cc31e070f32ed

Scopes:

Callback urls:

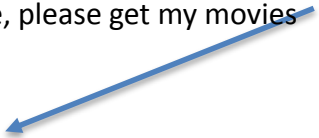
http://acme.com/auth/movies/callback Authorize

OAuth 2 in Action: Movie Service



Mr. Ideas

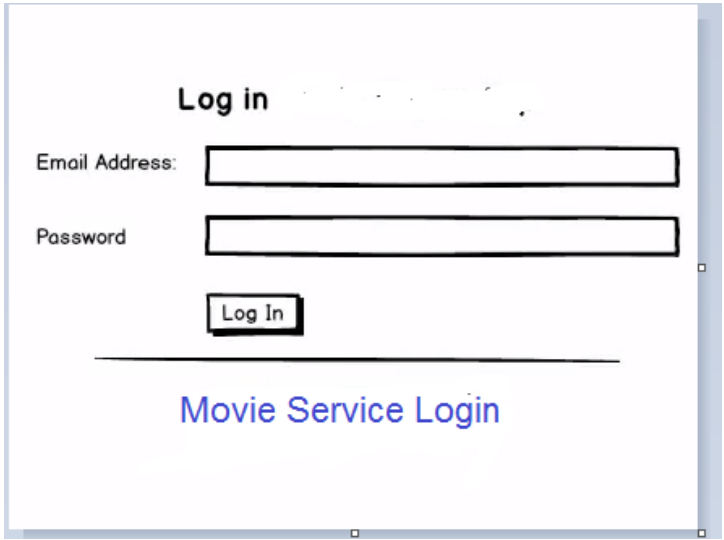
1. I want to work on a movie, please get my movies

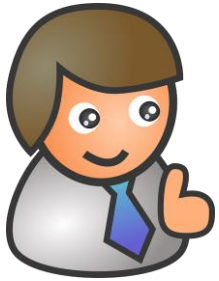


MovieEditor Web Application

Movie Service

OAuth 2 in Action: Movie Service

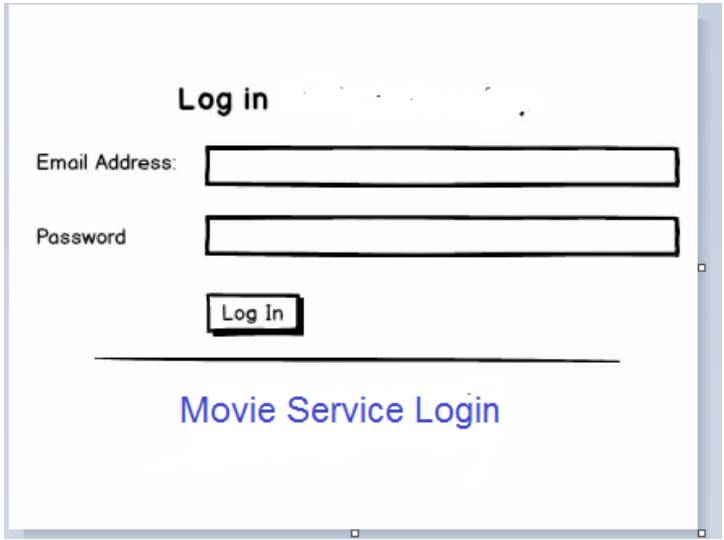


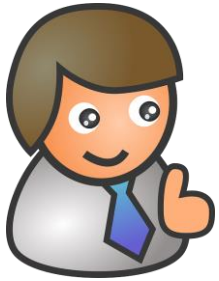


Mr. Ideas

Movie Service

OAuth 2 in Action: Movie Service



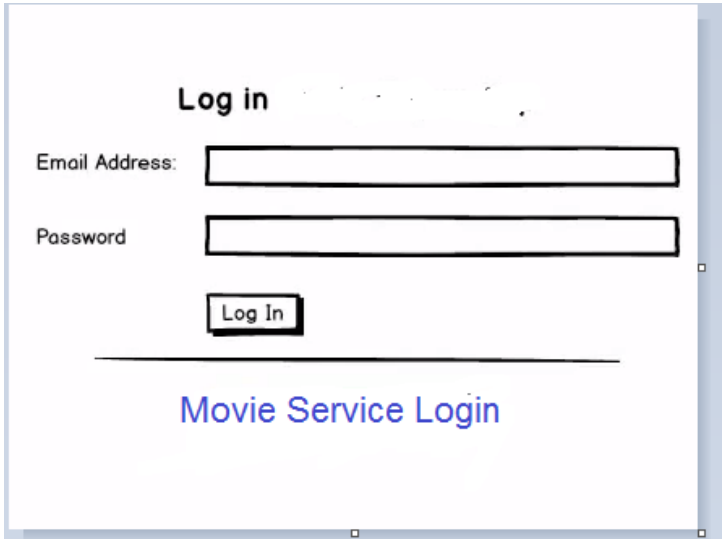


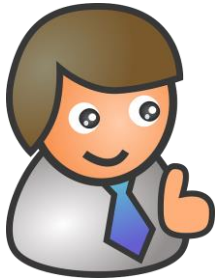
Mr. Ideas

app_id=12345
Redirect_uri = http://acme.com/auth/movies/callback
Scope = create_movies

Movie Service

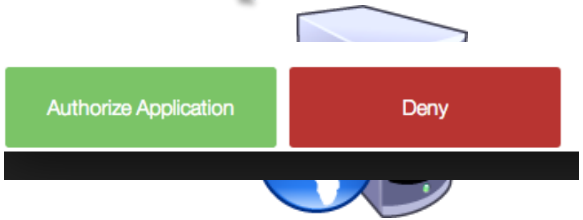
OAuth 2 in Action: Movie Service





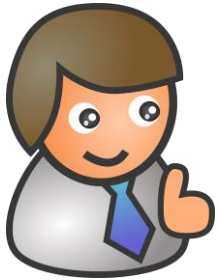
Mr. Ideas

YES - Authorize



Movie Service

OAuth 2 in Action: Movie Service



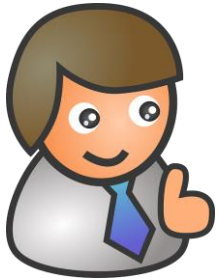
Mr. Ideas

MovieEditor Web Application

code = abcdef

Movie Service

OAuth 2 in Action: Movie Service



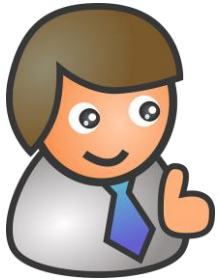
Mr. Ideas

MovieEditor Web Application

code=abcdef
app_id=XXX
secret_id=XXX

Movie Service

OAuth 2 in Action: Movie Service



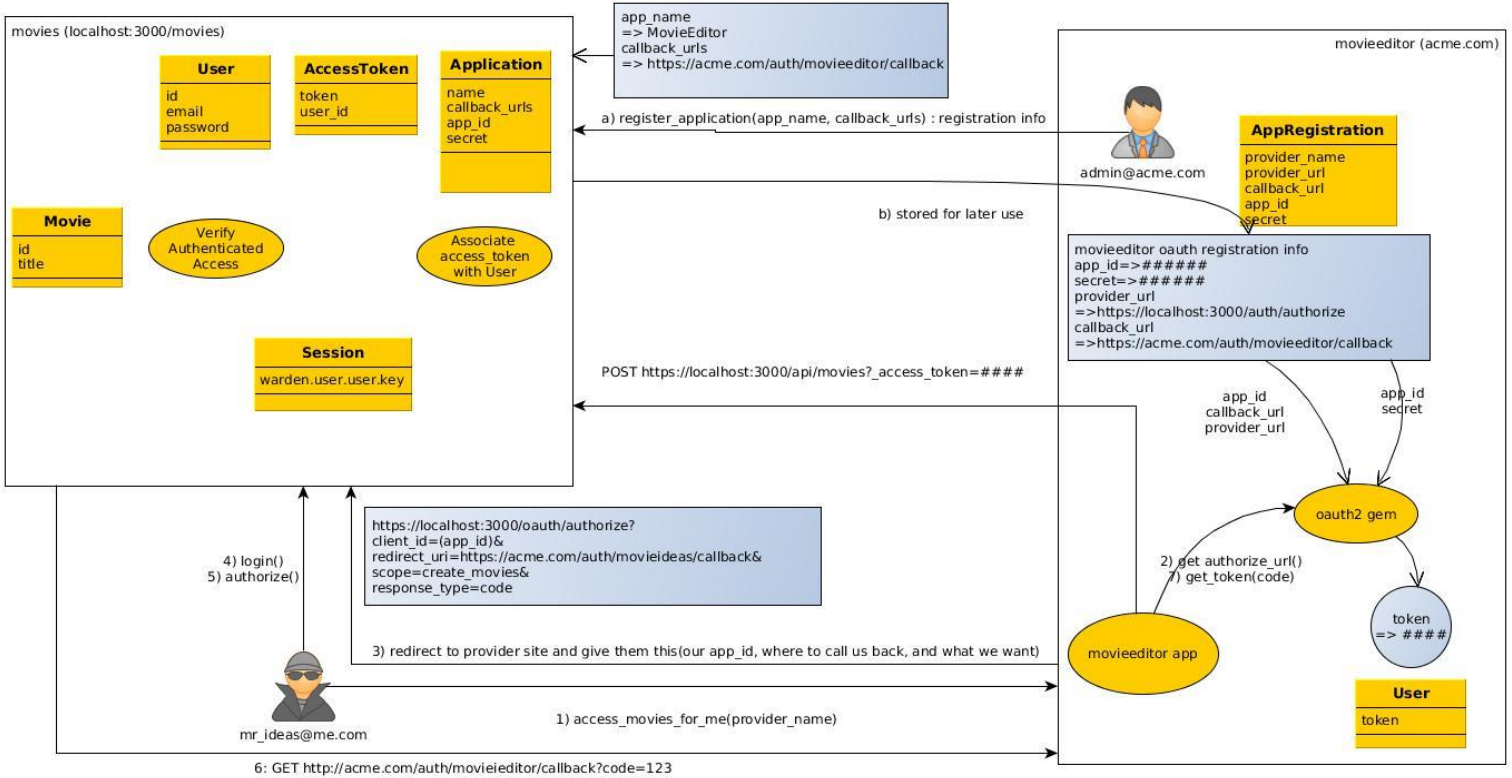
Mr. Ideas

MovieEditor Web Application

{ access_token: “efg123” }

Movie Service

Movie Service: Example



Summary

- OAuth 2.0 is the next evolution of the OAuth protocol which was originally created in late 2006

What's Next?

- OAuth 2 Movie Service – Workflow

In this lecture, we will discuss…

- ✧ Core Setup
- ✧ Resource Access
- ✧ Resource Controller
- ✧ Demo

Assembly

- ✧ Core Setup
  - Create new Rails application
    - `$ rails new oauth_movies`
    - `$ cd oauth_movies`
  - Add gems
    - `mongoid` and `httparty`

Assembly

- Core Setup
  - Integrate Mongoid
    - `$ rails g mongoid:config`
  - Define root URL
    - `$ rails g controller pages index`

```
# config/routes.rb
Rails.application.routes.draw do
  #get 'pages/index'
  root to: 'pages#index'
```

Resource Access

- HTML Controller → Movies
  - `$ rails g model Movie id title`
- Add Timestamp

Resource Controller

- Movies Controller
  - `$ rails g scaffold_controller Movies id title`
- **`config/routes.rb`**

```
# config/routes.rb
resources :movies
```

Resource Controller

- API Controller
  - Update the Gemfile with `responders` gem
    - Automatic marshalling
    - `gem 'responders', '~> 2.1', '>= 2.1.1'`
- Add Controller (`app/controller/api`) and update routes.rb

Demo

Demo – Test Drive

Summary

- ✧ Basic Setup of Resources

What's Next?

- ✧ Devise Integration

In this lecture, we will discuss…

- Devise
- Devise Configuration
- Devise Management
- Demo

Devise

- Devise is a popular authentication solution for Rails applications
- *A full-featured authentication solution which handles all of the controller logic and form views for you*

Configuration

- Gemfile
  - `gem 'devise', '~> 3.5', '>= 3.5.3'`
- Generate the Device configuration file using rails g

```
$ rails generate devise:install
      create  config/initializers/devise.rb
      create  config/locales/devise.en.yml
```

Configuration - URL

- Define a URL for generated e-mail messages to reference back to the server
  - Set to localhost:3000 (in development demo)

```
#config/environments/development.rb
  #devise options
  config.action_mailer.default_url_options = { host: 'localhost', port: 3000 }
```

User Model Class

- User Model class
  - hold accounts in your service
  - `rails g devise user`

```ruby
class User
  include Mongoid::Document
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable and :omniauthable
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :trackable, :validatable

  ## Database authenticatable
  field :email,              type: String, default: ""
  field :encrypted_password, type: String, default: ""
```

Devise Management

- Devise manages three (3) primary resources for our user:
  - login sessions - login/logout
  - passwords, and
  - registration data - email, optional fields

Demo

Summary

- ✧ Devise Integration to Movies app

What’s Next?

- ✧ Integrated Authentication

In this lecture, we will discuss…

- ✧ Integrate Sign-in and Authentication
- ✧ Doorkeeper integration
- ✧ Database configuration
- ✧ Demo

Integrating Authentication

- ✧ Integrate Sign-In and Authentication
  - API Base Controller (Write actions)

```
module Api
  class BaseController < ApplicationController
    before_action :authenticate_user!, except: [:index, :show ]
    before_action :user_signed_in?, except: [:index, :show ]
```

Verify Access

- ✧ DEMO (`/api` methods)
  - Verify access is still available to non-writable methods
  - Verify access is denied for writable methods

Demo

Doorkeeper

- Doorkeeper is an OAuth 2 provider for Rails
- It's built on top of Rails engines
- So far it supports all protocol flows

Doorkeeper: Gems

- `gem 'doorkeeper', '~> 3.1'`
- `gem "doorkeeper-mongodb", github: "doorkeeper-gem/doorkeeper-mongodb"`
- `gem 'oauth2', '~> 1.0'`

Doorkeeper: Configuration

- Install Doorkeeper
  - `rails g doorkeeper:install`
- Produces the following URI
  - `config/routes.rb`
    - `use_doorkeeper`

Doorkeeper: Database Configuration

- Configure the ORM and prepare the database

```
#config/initializers/doorkeeper.rb
Doorkeeper.configure do
  # Change the ORM that doorkeeper will use (needs plugins)
  #orm :active_record
  orm :mongoid5
```

- Install indexes (Mongoid)
  - `$ rake db:mongoid:create_indexes`

resource_owner_authenticator

- Update `resource_owner_authenticator` – to resolve User object based on what is stored in the session

```
resource_owner_authenticator do
  #fail "Please configure doorkeeper resource_owner_authenticator block located in #{__FILE__}"
  # Put your resource owner authentication logic here.
  # Example implementation:
  #   User.find_by_id(session[:user_id]) || redirect_to(new_user_session_url)
  user_key=session["warden.user.user.key"]
  user_id=user_key[0][0]  if user_key
  User.where(:id=>user_id).first || redirect_to(new_user_session_url)
end
```

Demo

Summary

- ✧ Doorkeeper integration with Movie Service

What’s Next?

- ✧ Registering Application with OAuth

In this lecture, we will discuss…

- ✧ Registration with OAuth
- ✧ Process Registration Results
- ✧ Build Authorization URL
- ✧ Access Application - Demo

Integrating Authentication

- Setup Registration between acme.com (Movie Editor App) and Movie Service and OAuth provider (embedded in Movie Service)
- MovieEditor (Acme.com) → signup for a new account
  - http://localhost:3000/oauth/applications/new

Registration

- Name: Acme Client
- Redirect URL: http://acme.com/auth/movies/callback
- Scopes: (blank)

Registration Results

- Application Id: XXXXXX
- Secret: XXXXXXX
- Callback URLs: http://acme.com/auth/movies/callback

Access Application

- Using `oauth2` gem, generate authorization URL
- Plug the authorization URL into a browser
  - Should get re-directed to the callback URL
- Generate access token
- Access application

Demo

Summary

- Register OAuth with Service and successful access of the application