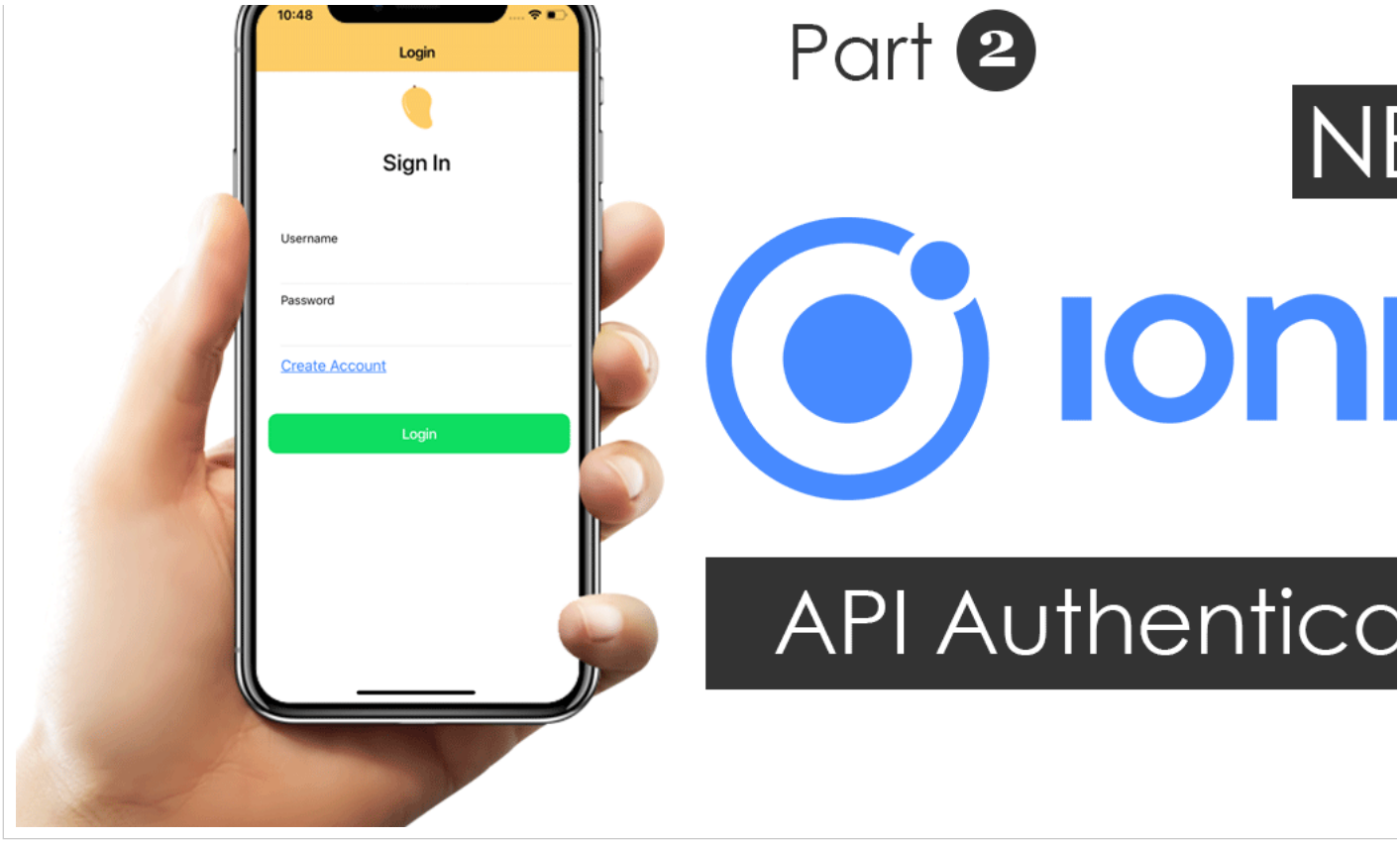
TUESDAY, SEPTEMBER 10, 2019

Ionic 5 and Angular 8: Restful API User Authentication Login and Signup using Guard and Resolver

ANDROID ANGULAR API IONIC IOS JAVASCRIPT MOBILE RESTFUL

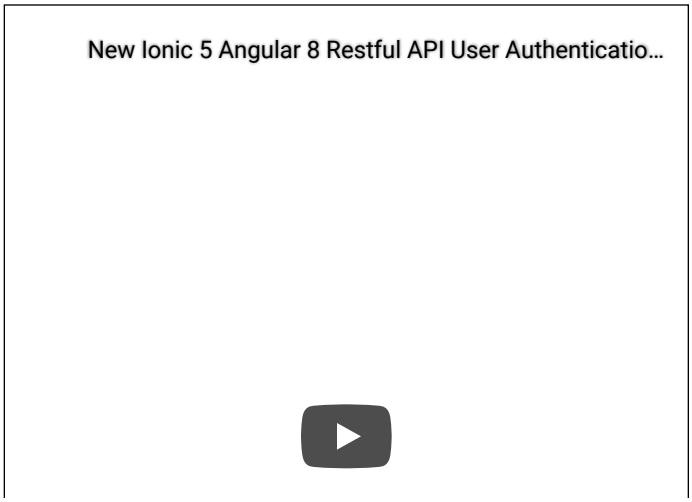
This is a continuation of my previous article creating an Ionic Angular project with welcome and tabs home page. Today's post explains how to implement login authentication system for your Ionic Angular application with guards and resolvers. It will show you how to log in with a user and store the user data and protect the routes, so it deals with token-based authentication. Every user details will be stored in an external database and a PHP based API is used in the backend for handling this authentication.



Live Demo

Ionic 5 Angular Welcome Page
Download this project on Github

Video Tutorial



```
production: false,
apiUrl: 'http://localhost:8084/'
};
```

environment.prod.ts

Production configuration file. Here set your production API endpoint. Production build `ng build --prod` command automatically maps with live API.

```
export const environment = {
production: true,
apiUrl: 'https://api.thewallscript.com/restful/'
};
```

Create Services

We have to generate following services for connecting APIs.

Http Service

Create a Http Service using Ionic generate command.

```
new-ionic-angular$ ionic generate service services/http
> ng generate service services/http
CREATE src/app/services/http.service.spec.ts (323 bytes)
CREATE src/app/services/http.service.ts (133 bytes)
```

http.service.ts

Create a post method to communicate post request with APIs using HttpClient.

```
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { environment } from '../../environments/environment';

@Injectable({
providedIn: 'root'
})
export class HttpService {
constructor(private http: HttpClient) {}

post(serviceName: string, data: any) {
const headers = new HttpHeaders();
const options = { headers: headers, withCredintials: false };
const url = environment.apiUrl + serviceName;

return this.http.post(url, JSON.stringify(data), options);
}
}
```

app.module.ts

Import the HttpClient module in application level.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { RouteReuseStrategy } from '@angular/router';

import { IonicModule, IonicRouteStrategy } from '@ionic/angular';
import { SplashScreen } from '@ionic-native/splash-screen/ngx';
```

```
import { HttpClientModule } from '@angular/common/http';

@NgModule({
declarations: [AppComponent],
entryComponents: [],
imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule, HttpClientModule],
providers: [
StatusBar,
SplashScreen,
HttpClientModule,
{ provide: RouteReuseStrategy, useClass: IonicRouteStrategy }
],
bootstrap: [AppComponent]
})
export class AppModule {}
```

Storage

Using Capacitor Storage API to store the data with key-value.

```
$ ionic generate service services/storage
> ng generate service services/storage
CREATE src/app/services/storage.service.spec.ts (338 bytes)
CREATE src/app/services/storage.service.ts (136 bytes)
```

storage.service.ts

Here you will find the store, get, removeItem and clear methods for handling the storage. This Strage API uses the mobile SQL lite database, for browser localStorage.

```
import { Injectable } from '@angular/core';
import { Plugins } from '@capacitor/core';
const { Storage } = Plugins;
@Injectable({
providedIn: 'root'
})
export class StorageService {
constructor() {}

// Store the value
async store(storageKey: string, value: any) {
const encryptedValue = btoa(escape(JSON.stringify(value)));
await Storage.set({
key: storageKey,
value: encryptedValue
});
}

// Get the value
async get(storageKey: string) {
const ret = await Storage.get({ key: storageKey });
return JSON.parse(unescape(atob(ret.value)));
}

async removeStorageItem(storageKey: string) {
await Storage.remove({ key: storageKey });
}
```

```
}
}
```

auth-constants.ts

Create this file under the config folder. This is a reference for auth storage key.

```
export classAuthConstants {
public static readonly AUTH = 'userData'
};
```

Auth Service

```
$ ionic generate service services/auth
> ng generate service services/auth
CREATE src/app/services/auth.service.spec.ts (323 bytes)
CREATE src/app/services/auth.service.ts (133 bytes)
```

auth.service.ts

Auth service is a controller to connect the APIs.

```
import { Injectable } from '@angular/core';
import { Router } from '@angular/router';
import { Observable } from 'rxjs';
import { HttpService } from './http.service';
import { StorageService } from './storage.service';

@Injectable({
providedIn: 'root'
})
export class AuthService {
constructor(
private httpService: HttpService,
private storageService: StorageService,
private router: Router
) {}

login(postData: any): Observable<any> {
return this.httpService.post('login', postData);
}

signup(postData: any): Observable<any> {
return this.httpService.post('signup', postData);
}

logout() {
this.storageService.removeStorageItem(AuthConstants.AUTH).then(res => {
this.router.navigate(['/login']);
});
}
}
```

Login Page

login.page.ts

Here loginAction method will validate the input values.

```
import { StorageService } from './../../services/storage.service';

@Component({
selector: 'app-login',
templateUrl: './login.page.html',
styleUrls: ['./login.page.scss']
})
export class LoginPage implements OnInit {
postData = {
username: '',
password: ''
};

constructor(
private router: Router,
private authService: AuthService,
private storageService: StorageService
) {}

ngOnInit() {}

validateInputs() {
let username = this.postData.username.trim();
let password = this.postData.password.trim();
return (
this.postData.username &&
this.postData.password &&
username.length > 0 &&
password.length > 0
);
}

loginAction() {
if (this.validateInputs()) {
this.authService.login(this.postData).subscribe(
(res: any) => {
if (res.userData) {
// Storing the User data.
this.storageService.store(AuthConstants.AUTH, res.userData);
this.router.navigate(['home/feed']);
} else {
console.log('incorrect password.');
}
},
(error: any) => {
console.log('Network Issue.');
}
);
} else {
console.log('Please enter email/username or password.');
}
}
}
```

login.page.html

Bind the postData object with input values.

```
<ion-header>
```

```
<ion-content padding='true'>
<div class="center">
<img src="assets/images/logo.png" class="smallLogo"/>
<h1>Sign In</h1>
</div>
<form>
<ion-list>
<ion-item>
<ion-label position="stacked">Username</ion-label>
<ion-input autocomplete="off" type="text" name="username" [(ngModel)]="postData.username"></ion-input>
</ion-item>

<ion-item>
<ion-label position="stacked">Password</ion-label>
<ion-input autocomplete="off" type="password" name="password" [(ngModel)]="postData.password"></ion-input>
</ion-item>

<ion-item lines='none'>
<a routerLink='/signup'>Create Account</a>
</ion-item>
</ion-list>
<ion-button expand="block" share="round" color="success" (click)="loginAction()">Login</ion-button>
</form>
</ion-content>a
```

Toast Service

Create a toast service for showing the login alerts.

```
$ ionic generate service services/toast
> ng generate service services/toast
CREATE src/app/services/toast.service.spec.ts (328 bytes)
CREATE src/app/services/toast.service.ts (134 bytes)
[OK] Generated service!
```

toast.service.ts

Using Ionic ToastController presenting the alerts.

```
import { Injectable } from '@angular/core';
import { ToastController } from '@ionic/angular';

@Injectable({
providedIn: 'root'
})
export class ToastService {
constructor(public toastController: ToastController) {}

async presentToast(infoMessage: string) {
const toast = await this.toastController.create({
message: infoMessage,
duration: 2000
});
toast.present();
}
}
```

login.page.ts

```
import { AuthConstants } from '../../config/auth-constants';
import { AuthService } from './../../services/auth.service';
import { StorageService } from './../../services/storage.service';
import { ToastService } from './../../services/toast.service';

@Component({
selector: 'app-login',
templateUrl: './login.page.html',
styleUrls: ['./login.page.scss']
})
export class LoginPage implements OnInit {
postData = {
username: '',
password: ''
};

constructor(
private router: Router,
private authService: AuthService,
private storageService: StorageService,
private toastService: ToastService
) {}

ngOnInit() {}

validateInputs() {
console.log(this.postData);
let username = this.postData.username.trim();
let password = this.postData.password.trim();
return (
this.postData.username &&
this.postData.password &&
username.length > 0 &&
password.length > 0
);
}

loginAction() {
if (this.validateInputs()) {
this.authService.login(this.postData).subscribe(
(res: any) => {
if (res.userData) {
// Storing the User data.
this.storageService.store(AuthConstants.AUTH, res.userData);
this.router.navigate(['home/feed']);
} else {
this.toastService.presentToast('Incorrect username and password.');
}
},
(error: any) => {
this.toastService.presentToast('Network Issue.');
}
);
} else {
this.toastService.presentToast(
'Please enter username or password.'
);
}
}
```

Follow the same like login.

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AuthConstants } from './../../config/auth-constants';
import { AuthService } from './../../services/auth.service';
import { StorageService } from './../../services/storage.service';
import { ToastService } from './../../services/toast.service';

@Component({
selector: 'app-signup',
templateUrl: './signup.page.html',
styleUrls: ['./signup.page.scss']
})
export class SignupPage implements OnInit {
postData = {
username: '',
email: '',
password: ''
};

constructor(
private authService: AuthService,
private toastService: ToastService,
private storageService: StorageService,
private router: Router
) {}

ngOnInit() {}

validateInputs() {
console.log(this.postData);
let username = this.postData.username.trim();
let password = this.postData.password.trim();
let email = this.postData.email.trim();
return (
this.postData.username &&
this.postData.password &&
this.postData.email &&
username.length > 0 &&
email.length > 0 &&
password.length > 0
);
}

signAction() {
if (this.validateInputs()) {
this.authService.signup(this.postData).subscribe(
(res: any) => {
if (res.userData) {
// Storing the User data.
this.storageService
.store(AuthConstants.AUTH, res.userData)
.then(res => {
this.router.navigate(['home/feed']);
});
} else {
this.toastService.presentToast(
'Data alreay exists, please enter new details.'
```

```
this.toastService.presentToast('Network Issue.');
}
);
} else {
this.toastService.presentToast(
'Please enter email, username or password.'
);
}
}
}
```

signup.page.html

Bind the postData object value.

```
<ion-header>
<ion-toolbar color="mango">
<ion-title>Registration</ion-title>
</ion-toolbar>
</ion-header>

<ion-content padding='true'>
<div class="center">
<img src="assets/images/logo.png" class="smallLogo"/>
<h1>Create Account</h1>
</div>
<form>
<ion-list>
<ion-item>
<ion-label position="stacked">Email</ion-label>
<ion-input autocomplete="off" name="email" type="email"></ion-input>
</ion-item>
<ion-item>
<ion-label position="stacked">Username</ion-label>
<ion-input autocomplete="off" name="username" type="text"></ion-input>
</ion-item>

<ion-item>
<ion-label position="stacked">Password</ion-label>
<ion-input autocomplete="off" name="password" type="password"></ion-input>
</ion-item>

<ion-item lines='none'>
Already have an account? <a routerLink='/login'>Sign in.</a>
</ion-item>
</ion-list>
<ion-button expand="block" share="round" color="success" (click)="signupAction()">Registration</ion-button>
</form>
</ion-content>
```

Run the mock server

Create a mock server for local development. You will find the more information Mock REST Backend Server for Angular and React Applications.

```
$cd server
$node server.js
```

Home Guard

```
$ ionic generate guard guards/home
> ng generate guard guards/home
CREATE src/app/guards/home.guard.spec.ts (346 bytes)
CREATE src/app/guards/home.guard.ts (248 bytes)
[OK] Generated guard!
```

home.guard.ts

Implement the canActivate method to resolve the promise values with Auth storage value.

```
import { Injectable } from '@angular/core';
import { CanActivate, Router } from '@angular/router';
import { AuthConstants } from '../config/auth-constants';
import { StorageService } from '../services/storage.service';

@Injectable({
providedIn: 'root'
})
export class HomeGuard implements CanActivate {
constructor(public storageService: StorageService, public router: Router) {}
canActivate(): Promise<boolean> {
return new Promise(resolve => {
this.storageService
.get(AuthConstants.AUTH)
.then(res => {
if (res) {
resolve(true);
} else {
this.router.navigate(['login']);
resolve(false);
}
})
.catch(err => {
resolve(false);
});
});
}
}
```

Index Guard

If auth data is present redirect to home route.

```
$ ionic generate guard guards/index
> ng generate guard guards/index
CREATE src/app/guards/index.guard.spec.ts (352 bytes)
CREATE src/app/guards/index.guard.ts (249 bytes)
[OK] Generated guard!
```

```
aimport { Injectable } from '@angular/core';
import { CanActivate, Router } from '@angular/router';
import { AuthConstants } from '../config/auth-constants';
import { StorageService } from '../services/storage.service';

@Injectable({
providedIn: 'root'
```

```
return new Promise(resolve => {
this.storageService
.get(AuthConstants.AUTH)
.then(res => {
if (res) {
this.router.navigate(['home/feed']);
resolve(false);
} else resolve(true);
})
.catch(err => {
resolve(true);
});
});
}
}
```

home.router.ts

Impore the home guard and connect with routes.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomePage } from './home.page';
import { HomeGuard } from '../guards/home.guard';

const routes: Routes = [
{
path: 'home',
component: HomePage,
canActivate: [HomeGuard],
children: [
{
path: 'feed',
loadChildren: () =>
import('../pages/feed/feed.module').then(m => m.FeedPageModule)
},
{
path: 'messages',
loadChildren: () =>
import('../pages/messages/messages.module').then(
m => m.MessagesPageModule
)
},
{
path: 'notifications',
loadChildren: () =>
import('../pages/notifications/notifications.module').then(
m => m.NotificationsPageModule
)
},
{
path: 'settings',
loadChildren: () =>
import('../pages/settings/settings.module').then(
m => m.SettingsPageModule
)
}
]
}
```

```
exports: [RouterModule]
})
export class HomeRouter {}
```

index.router.ts

Import the index guard and implement the canActivate.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { IndexPage } from './index.page';
import { IndexGuard } from '../guards/index.guard';

const routes: Routes = [
{
path: '',
component: IndexPage,
canActivate: [IndexGuard],
children: [
{
path: '',
loadChildren: () =>
import('../pages/welcome/welcome.module').then(
m => m.WelcomePageModule
)
},
{
path: 'login',
loadChildren: () =>
import('../pages/login/login.module').then(m => m.LoginPageModule)
},
{
path: 'signup',
loadChildren: () =>
import('../pages/signup/signup.module').then(m => m.SignupPageModule)
}
]
}
];

@NgModule({
imports: [RouterModule.forChild(routes)],
exports: [RouterModule]
})
export class IndexRouter {}
```

Resolver

User data resolver for accessing the user data.

auth.service.ts

Create a getUserData method to get the user data from the storage. Update the behavior subject value with next method.

```
import { Injectable } from '@angular/core';
import { Router } from '@angular/router';
import { BehaviorSubject, Observable } from 'rxjs';
import { AuthConstants } from './../config/auth-constants';
import { HttpService } from './http.service';
```

```
})
export class AuthService {
userData$ = new BehaviorSubject<any>([]);

constructor(
private httpService: HttpService,
private storageService: StorageService,
private router: Router
) {}

getUserData() {
this.storageService.get(AuthConstants.AUTH).then(res => {
this.userData$.next(res);
});
}

login(postData: any): Observable<any> {
return this.httpService.post('login', postData);
}

signup(postData: any): Observable<any> {
return this.httpService.post('signup', postData);
}

logout() {
this.storageService.removeStorageItem(AuthConstants.AUTH).then(res => {
this.userData$.next('');
this.router.navigate(['/login']);
});
}
}
```

user-data.resolver.ts

Create a resolver class and call the getUserData method.

```
import { Injectable } from '@angular/core';
import { AuthService } from '../services/auth.service';

@Injectable({
providedIn: 'root'
})
export class UserDataResolver {
constructor(private authService: AuthService) {}

resolve() {
console.log('Hello');
return this.authService.getUserData();
}
}
```

home.router.ts

Include the resolver for home router.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomePage } from './home.page';
import { HomeGuard } from '../guards/home.guard';
```

```
path: 'home',
component: HomePage,
canActivate: [HomeGuard],
resolve:{
userData: UserDataResolver
},
children: [
{
path: 'feed',
loadChildren: () =>
import('../pages/feed/feed.module').then(m => m.FeedPageModule)
},
{
path: 'messages',
loadChildren: () =>
import('../pages/messages/messages.module').then(
m => m.MessagesPageModule
)
},
{
path: 'notifications',
loadChildren: () =>
import('../pages/notifications/notifications.module').then(
m => m.NotificationsPageModule
)
},
{
path: 'settings',
loadChildren: () =>
import('../pages/settings/settings.module').then(
m => m.SettingsPageModule
)
}
]
}
];

@NgModule({
imports: [RouterModule.forChild(routes)],
exports: [RouterModule]
})
export class HomeRouter {}
```

feed.page.ts

Subscribe to the behavior subject value to get the user data.

```
import { Component, OnInit } from '@angular/core';
import { AuthService } from './../../services/auth.service';

@Component({
selector: 'app-feed',
templateUrl: './feed.page.html',
styleUrls: ['./feed.page.scss']
})
export class FeedPage implements OnInit {
public authUser: any;
constructor(private auth: AuthService) {}
```

```
this.authUser = res;
});
}
}
```

feed.page.html

Print thel user data value.

```
<ion-header>
<ion-toolbar>
<ion-title>Feed</ion-title>
</ion-toolbar>
</ion-header>

<ion-content>
<ion-item>
<h2> Welcome to {{ authUser?.name }}</h2>
</ion-item>
</ion-content>
```

package.json

Include the script for using npm commands.

```
"scripts": {
"ng": "ng",
"start": "ng serve",
"build": "ng build --prod",
"test": "ng test",
"lint": "ng lint",
"e2e": "ng e2e",
"ionic-build": "ionic build --prod",
"ios-add": "ionic capacitor add ios",
"android-add": "ionic capacitor add android",
"ios-open": "ionic capacitor open ios",
"android-open": "ionic capacitor open android",
"ios-copy": "ionic capacitor copy ios",
"android-copy": "ionic capacitor copy android"
},
```

Building a Mobile Application

You have to create a production build for generating mobile applications.

Production Build

```
$ ionic build --prod
```

Build iOS App

Following commands for executing Xcode build, watch the video tutorial you will understand more.

```
$ npm run ios-add
$ npm run ios-open
```

Build Android App

Open Android build using Android SDK

Project Updates

If you want to update your project changes.

```
$ npm run ios-copy
$ npm run android-copy
```


Related Posts

Getting Started with Ionic React and Capacitor

Ionic 3 and Angular 4: Insert and Delete with Token Based Restful API

Mock REST Backend Server for Angular and React Applications.

Angular 8 Multi Tenants Architecture

React JS and PHP Restful API User Authentication for Login and Signup.

ReactJS Welcome Page with Routing Tutorial

Create a RESTful API using NodeJS and MySQL Database

Ionic Electron Desktop App

Angular 8 Lazy Load Routing.

Angular Routing with Lazy Loading Design Pattern

1	5
2	6
3	7
4	8

1 comments:

Enter your comment...

Comment as: khaliljedda4@g Sign out

Publish Preview Notify me

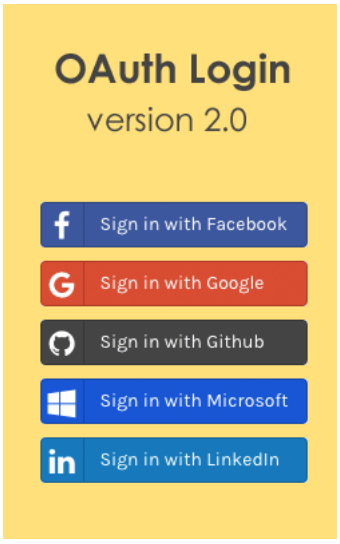
Newer Post Home Older Post

SRINIVAS TAMADA
Entrepreneur, Blogger,
Thinker & UI Architect
srinivas [at] 9lessons.info
I love the WEB - **Indian**
Invented Wall Script
Lives in Atlanta - USA

Most Popular Posts

Categories

Angular	J2EE
React JS	HTML5
jQuery	Ionic
JavaScript	MySQL
Tutorials	Ajax
Web Design	PHP



Recent Posts

New Ionic 5 Angular 8 Display, Update and Delete Records with RxJS

Ionic 5 and Angular 8: Restful API User Authentication Login and Signup using Guard and Resolver

Mock REST Backend Server for Angular and React Applications.

Ionic 5 and Angular 8: Create a Welcome Page with Tabs Home Pages.

Microsoft Azure Virtual Machines Setup with Ubuntu and XAMPP PHP Server

Google Cloud VM Instance Setup with Ubuntu and XAMPP PHP Server

Angular 8 Multi Tenants Architecture

30+ Web Tools and Services Reviewed For You

41721937