namespace Reddah.Web.UI.Controllers

{

using System.Xml;

using System.Web.Mvc;

using System.Globalization;

using System.Web.Security;

public class BaseController : Controller

{

protected string GetContentType(string path)

{

var doc = new XmlDocument();

var locale = CultureInfo.CurrentUICulture.Name;

if (!System.IO.File.Exists(HttpContext.Server.MapPath("~/App\_Data/" + path + "." + locale + ".xml")))

{

doc.Load(HttpContext.Server.MapPath("~/App\_Data/" + path + ".xml"));

}

else

{

doc.Load(HttpContext.Server.MapPath("~/App\_Data/" + path + "." + locale + ".xml"));

}

return doc.SelectSingleNode("ContentType/@Name").Value;

}

protected static string ErrorCodeToString(MembershipCreateStatus createStatus)

{

// See http://go.microsoft.com/fwlink/?LinkID=177550 for

// a full list of status codes.

switch (createStatus)

{

case MembershipCreateStatus.DuplicateUserName:

return "User name already exists. Please enter a different user name.";

case MembershipCreateStatus.DuplicateEmail:

return "A user name for that e-mail address already exists. Please enter a different e-mail address.";

case MembershipCreateStatus.InvalidPassword:

return "The password provided is invalid. Please enter a valid password value.";

case MembershipCreateStatus.InvalidEmail:

return "The e-mail address provided is invalid. Please check the value and try again.";

case MembershipCreateStatus.InvalidAnswer:

return "The password retrieval answer provided is invalid. Please check the value and try again.";

case MembershipCreateStatus.InvalidQuestion:

return "The password retrieval question provided is invalid. Please check the value and try again.";

case MembershipCreateStatus.InvalidUserName:

return "The user name provided is invalid. Please check the value and try again.";

case MembershipCreateStatus.ProviderError:

return "The authentication provider returned an error. Please verify your entry and try again. If the problem persists, please contact your system administrator.";

case MembershipCreateStatus.UserRejected:

return "The user creation request has been canceled. Please verify your entry and try again. If the problem persists, please contact your system administrator.";

default:

return "An unknown error occurred. Please verify your entry and try again. If the problem persists, please contact your system administrator.";

}

}

}

}

using System.Web.Mvc;

using System.Linq;

namespace Reddah.Web.UI.Controllers

{

public enum VerifySuccessState

{

NotVerified,

Success,

UserNotExist,

WrongToken,

AlreadyVerified

}

public class VerifyEmailController : Controller

{

private readonly log4net.ILog log = log4net.LogManager.GetLogger("VerifyEmailController");

public ActionResult Index(string userId, string emailToken)

{

log.Info(string.Format("userId:{0}, emailToken:{1}", userId, emailToken));

ViewBag.VerifySuccessState = VerifySuccessState.NotVerified;

ViewBag.UserId = userId;

ViewBag.EmailToken = emailToken;

using (var context = new reddahEntities1())

{

var intUserId = -1;

if (int.TryParse(userId, out intUserId))

{

var item = context.webpages\_Membership.FirstOrDefault(u => u.UserId == intUserId);

if (item != null)

{

if (item.IsConfirmed == true)

{

ViewBag.VerifySuccessState = VerifySuccessState.AlreadyVerified;

}

else

{

if (item.ConfirmationToken == emailToken)

{

item.IsConfirmed = true;

context.SaveChanges();

ViewBag.VerifySuccessState = VerifySuccessState.Success;

}

else

{

ViewBag.VerifySuccessState = VerifySuccessState.WrongToken;

}

}

}

else

{

ViewBag.VerifySuccessState = VerifySuccessState.UserNotExist;

}

}

else

{

ViewBag.VerifySuccessState = VerifySuccessState.UserNotExist;

}

}

return View("~/Views/Account/VerifyEmail.cshtml");

}

}

}

Reddah.service.ts

import { Injectable } from '@angular/core';

import { Observable, of } from 'rxjs';

import { HttpClient, HttpHeaders, HttpClientJsonpModule, HttpClientModule} from '@angular/common/http';

import {

Headers, Http, JsonpModule, XHRBackend, RequestOptions, RequestOptionsArgs, Jsonp,

JSONPBackend, URLSearchParams, QueryEncoder, ResponseContentType

} from '@angular/http';

import { catchError, map, tap } from 'rxjs/operators';

import { Article } from "./model/article";

import { UserProfileModel } from './model/UserProfileModel';

import { UserModel, QueryCommentModel, NewCommentModel, NewTimelineModel } from './model/UserModel';

import { Locale } from './model/locale';

import { FileTransfer, FileUploadOptions, FileTransferObject } from '@ionic-native/file-transfer/ngx';

import { File, FileEntry } from '@ionic-native/file/ngx';

import { LocalStorageService } from 'ngx-webstorage';

import { LoadingController, NavController, ModalController, ToastController, Platform } from '@ionic/angular';

import { CacheService } from 'ionic-cache';

import { TranslateService } from '@ngx-translate/core';

import { InAppBrowser } from '@ionic-native/in-app-browser/ngx';

import \* as moment from 'moment';

import { Camera, CameraOptions } from '@ionic-native/camera';

import { ImageResizer, ImageResizerOptions } from '@ionic-native/image-resizer';

import { Router } from '@angular/router';

@Injectable({

providedIn: 'root'

})

export class ReddahService {

constructor(

private http: HttpClient,

private localStorageService: LocalStorageService,

private jsonp: Jsonp,

private transfer: FileTransfer,

private file: File,

private toastController: ToastController,

private modalController: ModalController,

private platform: Platform,

private cacheService: CacheService,

private translate: TranslateService,

private iab: InAppBrowser,

private router: Router,

) { }

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private registersubUrl = 'https://login.reddah.com/api/pub/registersub';

registerSub(formData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.registersubUrl, formData)

.pipe(

tap(data => this.log('register sub')),

catchError(this.handleError('register sub', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private registerUrl = 'https://login.reddah.com/api/auth/register';

register(formData): Observable<any> {

return this.http.post<any>(this.registerUrl, formData)

.pipe(

tap(data => this.log('register')),

catchError(this.handleError('register', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private loginUrl = 'https://login.reddah.com/api/auth/sign';

login(userName: string, password: string): Observable<any> {

return this.http.post<any>(this.loginUrl, new UserModel(userName, password))

.pipe(

tap(data => this.log('login')),

catchError(this.handleError('login', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private getCommentsUrl = 'https://login.reddah.com/api/article/getcomments';

getComments(articleId: number): Observable<any> {

return this.http.post<any>(this.getCommentsUrl, new QueryCommentModel("", articleId))

.pipe(

tap(data => this.log('get comments')),

catchError(this.handleError('get comments', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private addCommentsUrl = 'https://login.reddah.com/api/article/addcomments';

addComments(articleId: number, parentId: number, content: string, uid : string): Observable<any> {

return this.http.post<any>(this.addCommentsUrl, new NewCommentModel(this.getCurrentJwt(), articleId, parentId, content, uid))

.pipe(

tap(data => this.log('add comment')),

catchError(this.handleError('add comment', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private shareToFriendUrl = 'https://login.reddah.com/api/chat/sharetofriend';

shareToFriend(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.shareToFriendUrl, formData)

.pipe(

tap(data => this.log('share to friend')),

catchError(this.handleError('share to friend', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private addPhotoCommentsUrl = 'https://login.reddah.com/api/chat/addphotocomments';

addPhotoComments(formData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.addPhotoCommentsUrl, formData)

.pipe(

tap(data => this.log('add photo comment')),

catchError(this.handleError('add photo comment', []))

);

}

private commentLikeUrl = 'https://login.reddah.com/api/article/commentlike';

commentLike(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.commentLikeUrl, formData)

.pipe(

tap(data => this.log('comment like')),

catchError(this.handleError('comment like', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private addTimelineUrl = 'https://login.reddah.com/api/article/addtimeline';

addTimeline(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

const httpOptions = {

headers: new HttpHeaders({

//'enctype': 'multipart/form-data; boundary=----WebKitFormBoundaryuL67FWkv1CA',

'Content-Type': 'multipart/form-data',

'Accept': 'application/json',

})

};

return this.http.post<any>(this.addTimelineUrl, formData)

.pipe(

tap(data => this.log('add timeline')),

catchError(this.handleError('add timeline', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private getMyTimelineUrl = 'https://login.reddah.com/api/article/getmytimeline';

getMyTimeline(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.getMyTimelineUrl, formData)

.pipe(

tap(data => this.log('get my timeline')),

catchError(this.handleError('get my timeline', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private getTimelineUrl = 'https://login.reddah.com/api/article/gettimeline';

getTimeline(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.getTimelineUrl, formData)

.pipe(

tap(data => this.log('get timeline')),

catchError(this.handleError('get timeline', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private getMaterialUrl = 'https://login.reddah.com/api/pub/getmaterial';

getMyMaterial(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.getMaterialUrl, formData)

.pipe(

tap(data => this.log('get my material')),

catchError(this.handleError('get my material', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private getReportUrl = 'https://login.reddah.com/api/admin/getreport';

getReport(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.getReportUrl, formData)

.pipe(

tap(data => this.log('get report')),

catchError(this.handleError('get report', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private getMyReportUrl = 'https://login.reddah.com/api/admin/getmyreport';

getMyReport(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.getMyReportUrl, formData)

.pipe(

tap(data => this.log('get my report')),

catchError(this.handleError('get my report', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private getUserInfoUrl = 'https://login.reddah.com/api/article/getuser';

getUserInfo(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.getUserInfoUrl, formData)

.pipe(

tap(data => this.log('get user info')),

catchError(this.handleError('get user info', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private timelineLikeUrl = 'https://login.reddah.com/api/article/like';

like(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.timelineLikeUrl, formData)

.pipe(

tap(data => this.log('set like')),

catchError(this.handleError('set like', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private searchUserUrl = 'https://login.reddah.com/api/article/searchuser';

searchUser(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.searchUserUrl, formData)

.pipe(

tap(data => this.log('search user')),

catchError(this.handleError('search user', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private addFriendUrl = 'https://login.reddah.com/api/article/addfriend';

addFriend(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.addFriendUrl, formData)

.pipe(

tap(data => this.log('add friend')),

catchError(this.handleError('add friend', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private changeNoteNameUrl = 'https://login.reddah.com/api/article/changenotename';

changeNoteName(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.changeNoteNameUrl, formData)

.pipe(

tap(data => this.log('change note name')),

catchError(this.handleError('change note name', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private changeSignatureUrl = 'https://login.reddah.com/api/article/changesignature';

changeSignature(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.changeSignatureUrl, formData)

.pipe(

tap(data => this.log('change signature')),

catchError(this.handleError('change signature', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private changePrivacyUrl = 'https://login.reddah.com/api/article/changeprivacy';

changePrivacy(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.changePrivacyUrl, formData)

.pipe(

tap(data => this.log('change privacy')),

catchError(this.handleError('change privacy', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private changeLocationUrl = 'https://login.reddah.com/api/article/changelocation';

changeLocation(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.changeLocationUrl, formData)

.pipe(

tap(data => this.log('change location')),

catchError(this.handleError('change location', []))

);

}

saveUserLocation(userName, location, lat, lng, tag=""){

let formData = new FormData();

formData.append("location", JSON.stringify(location));

formData.append("lat", JSON.stringify(lat));

formData.append("lng", JSON.stringify(lng));

formData.append("tag", tag);

this.changeLocation(formData)

.subscribe(result =>

{

if(result.Success==0){

this.localStorageService.store('userlocation\_'+userName, location.title);

this.localStorageService.store('userlocationjson\_'+userName, JSON.stringify(location));

}

else

alert(result.Message);

});

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

//type 0:girl, 1:boy

private getusersbylocation = 'https://login.reddah.com/api/article/getusersbylocation';

getUsersByLocation(type, latCenter, lngCenter, latLow, latHigh, lngLow, lngHigh, min=0){

let formData = new FormData();

formData.append('jwt', this.getCurrentJwt());

formData.append("type", JSON.stringify(type));

formData.append("latCenter", JSON.stringify(latCenter));

formData.append("lngCenter", JSON.stringify(lngCenter));

formData.append("latLow", JSON.stringify(latLow));

formData.append("latHigh", JSON.stringify(latHigh));

formData.append("lngLow", JSON.stringify(lngLow));

formData.append("lngHigh", JSON.stringify(lngHigh));

formData.append("min", JSON.stringify(min));

return this.http.post<any>(this.getusersbylocation, formData)

.pipe(

tap(data => this.log('get users by location')),

catchError(this.handleError('get users by location', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private changeNickNameUrl = 'https://login.reddah.com/api/article/changenickname';

changeNickName(formData: FormData): Observable<any> {

return this.service(formData, this.changeNickNameUrl, 'change nick name');

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private changeSexUrl = 'https://login.reddah.com/api/article/changesex';

changeSex(formData: FormData): Observable<any> {

return this.service(formData, this.changeSexUrl, 'change sex');

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private changePasswordUrl = 'https://login.reddah.com/api/auth/changepassword';

changePassword(formData: FormData): Observable<any> {

return this.service(formData, this.changePasswordUrl, 'change password');

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

service(formData: FormData, url, text): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(url, formData)

.pipe(

tap(data => this.log(text)),

catchError(this.handleError(text, []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private changeGroupChatTitleUrl = 'https://login.reddah.com/api/chat/changegroupchattitle';

changeGroupChatTitle(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.changeGroupChatTitleUrl, formData)

.pipe(

tap(data => this.log('change group chat title')),

catchError(this.handleError('change group chat title', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private removeFriendUrl = 'https://login.reddah.com/api/article/removefriend';

removeFriend(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.removeFriendUrl, formData)

.pipe(

tap(data => this.log('remove friend')),

catchError(this.handleError('remove friend', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private approveFriendUrl = 'https://login.reddah.com/api/article/approvefriend';

approveFriend(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.approveFriendUrl, formData)

.pipe(

tap(data => this.log('approve friend')),

catchError(this.handleError('approve friend', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private friendRequestsUrl = 'https://login.reddah.com/api/article/friendrequests';

friendRequests(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.friendRequestsUrl, formData)

.pipe(

tap(data => this.log('get friend requests')),

catchError(this.handleError('get friend requests', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private friendsUrl = 'https://login.reddah.com/api/article/friends';

getFriends(): Observable<any> {

let formData = new FormData();

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.friendsUrl, formData)

.pipe(

tap(data => this.log('get friends')),

catchError(this.handleError('get friends', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private focusPubsUrl = 'https://login.reddah.com/api/pub/focuspubs';

getFocusPubs(): Observable<any> {

let formData = new FormData();

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.focusPubsUrl, formData)

.pipe(

tap(data => this.log('get focus pubs')),

catchError(this.handleError('get focus pubs', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private subsUrl = 'https://login.reddah.com/api/pub/subs';

getSubs(): Observable<any> {

let formData = new FormData();

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.subsUrl, formData)

.pipe(

tap(data => this.log('get subs')),

catchError(this.handleError('get subs', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private updateUserPhotoUrl = 'https://login.reddah.com/api/article/updateuserphoto';

updateUserPhoto(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

const httpOptions = {

headers: new HttpHeaders({

'Content-Type': 'multipart/form-data',

'Accept': 'application/json',

})

};

return this.http.post<any>(this.updateUserPhotoUrl, formData)

.pipe(

tap(data => this.log('update user photo')),

catchError(this.handleError('update user photo', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private bookmarkUrl = 'https://login.reddah.com/api/article/bookmark';

bookmark(formData: FormData): Observable<any> {

return this.service(formData, this.bookmarkUrl, "set bookmark");

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private deleteBookmarkUrl = 'https://login.reddah.com/api/article/deletebookmark';

deleteBookmark(formData: FormData): Observable<any> {

return this.service(formData, this.deleteBookmarkUrl, "delete bookmark");

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private deleteArticleUrl = 'https://login.reddah.com/api/article/deletearticle';

deleteArticle(formData: FormData): Observable<any> {

return this.service(formData, this.deleteArticleUrl, "delete article");

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private deleteCommentUrl = 'https://login.reddah.com/api/article/deletecomment';

deleteComment(formData: FormData): Observable<any> {

return this.service(formData, this.deleteCommentUrl, "delete comment");

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private addPubArticleUrl = 'https://login.reddah.com/api/pub/addpubarticle';

addPubArticle(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.addPubArticleUrl, formData)

.pipe(

tap(data => this.log('add pub article')),

catchError(this.handleError('add pub article', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private addPubMiniUrl = 'https://login.reddah.com/api/pub/addpubmini';

addPubMini(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.addPubMiniUrl, formData)

.pipe(

tap(data => this.log('add pub article')),

catchError(this.handleError('add pub article', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private publishArticleUrl = 'https://login.reddah.com/api/pub/publisharticle';

publishArticle(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.publishArticleUrl, formData)

.pipe(

tap(data => this.log('pub article')),

catchError(this.handleError('pub article', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private publishMiniUrl = 'https://login.reddah.com/api/pub/publishprogram';

publishMini(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.publishMiniUrl, formData)

.pipe(

tap(data => this.log('pub mini')),

catchError(this.handleError('pub mini', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private getChatUrl = 'https://login.reddah.com/api/chat/getchat';

getChat(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.getChatUrl, formData)

.pipe(

tap(data => this.log('create chat')),

catchError(this.handleError('create chat', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private createGroupChatUrl = 'https://login.reddah.com/api/chat/creategroupchat';

createGroupChat(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

formData.append("title", this.translate.instant("Pop.GroupChatTitle"));

formData.append("annouce", this.translate.instant("Pop.GroupChatAnnouce"));

formData.append("update", this.translate.instant("Pop.GroupChatUpdate"));

return this.http.post<any>(this.createGroupChatUrl, formData)

.pipe(

tap(data => this.log('create group chat')),

catchError(this.handleError('create group chat', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private getGroupChatUrl = 'https://login.reddah.com/api/chat/getgroupchat';

getGroupChat(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.getGroupChatUrl, formData)

.pipe(

tap(data => this.log('get group chat')),

catchError(this.handleError('get group chat', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private getGroupListUrl = 'https://login.reddah.com/api/chat/getgrouplist';

getGroupList(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.getGroupListUrl, formData)

.pipe(

tap(data => this.log('get group chat list')),

catchError(this.handleError('get group chat list', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private deleteGroupChatUrl = 'https://login.reddah.com/api/chat/deletegroupchat';

deleteGroupChat(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.deleteGroupChatUrl, formData)

.pipe(

tap(data => this.log('delete group chat')),

catchError(this.handleError('delete group chat', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private addToGroupChatUrl = 'https://login.reddah.com/api/chat/addtogroupchat';

addToGroupChat(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

formData.append('add', this.translate.instant('Pop.AddToGrp'));

formData.append('kick', this.translate.instant('Pop.KickGrp'));

return this.http.post<any>(this.addToGroupChatUrl, formData)

.pipe(

tap(data => this.log('add to group chat')),

catchError(this.handleError('add to group chat', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private addAudioChatUrl = 'https://login.reddah.com/api/chat/addaudiochat';

addAudioChat(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.addAudioChatUrl, formData)

.pipe(

tap(data => this.log('add audio chat')),

catchError(this.handleError('add audio chat', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private messageunreadUrl = 'https://login.reddah.com/api/article/messageunread';

getMessageUnread(): Observable<any> {

let formData = new FormData();

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.messageunreadUrl, formData)

.pipe(

tap(data => this.log('get msg unread')),

catchError(this.handleError('get msg unread', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private messagesetreadUrl = 'https://login.reddah.com/api/article/messagesetread';

setMessageRead(): Observable<any> {

let formData = new FormData();

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.messagesetreadUrl, formData)

.pipe(

tap(data => this.log('set msg as read')),

catchError(this.handleError('set msg as read', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private getarticlebyidUrl = 'https://login.reddah.com/api/article/getarticlebyid';

getArticleById(formData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.getarticlebyidUrl, formData)

.pipe(

tap(data => this.log('get article by id')),

catchError(this.handleError('get article by id', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private getContactMessages = 'https://login.reddah.com/api/chat/getmessages';

getMessages(): Observable<any> {

let formData = new FormData();

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.getContactMessages, formData)

.pipe(

tap(data => this.log('get messages')),

catchError(this.handleError('get messages', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private log(message: string) {

//console.log(message);

}

public Locales = [

new Locale("zh-CN", "简体中文 (zh-CN)"),

new Locale("en-US", "English (en-US)"),

new Locale("es-ES", "Español (es-ES)"),

new Locale("ar-AE", " عربي ، (ar-AE)"),

new Locale("ru-RU", "Pусский язык (ru-RU)"),

new Locale("pt-PT", "Português (pt-PT)"),

new Locale("ja-JP", "日本語 (ja-JP)"),

new Locale("de-DE", "Deutsch (de-DE)"),

new Locale("fr-FR", "Français (fr-FR)"),

new Locale("ko-KR", "한국어 (ko-KR)"),

new Locale("it-IT", "Italiano (it-IT)"),

new Locale("el-GR", "Ελληνικά (el-GR)"),

new Locale("nl-NL", "Nederlands (nl-NL)"),

new Locale("th-TH", "ภาษาไทย (th-TH)"),

new Locale("zh-TW", "繁體中文 (zh-TW)"),

];

private publisherUrl = 'https://login.reddah.com/api/pub/getpublisher';

getPublishers(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.publisherUrl, formData)

.pipe(

tap(data => this.log('fetched publishers')),

catchError(this.handleError('getReddah publishers', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private setFocusUrl = 'https://login.reddah.com/api/pub/setfocus';

setFocus(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.setFocusUrl, formData)

.pipe(

tap(data => this.log('set focus pub')),

catchError(this.handleError('set focus pub', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private unFocusUrl = 'https://login.reddah.com/api/pub/unfocus';

unFocus(formData: FormData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.unFocusUrl, formData)

.pipe(

tap(data => this.log('unfocus pub')),

catchError(this.handleError('unfocus pub', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private articlesUrl = 'https://reddah.com/api/webapi/getarticles';

private userProfileModel: UserProfileModel;

getArticles(

loadedIds: Number[],

dislikeGroups: string[]=[],

dislikeUserNames: string[]=[],

locale: String, menu: String, keyword="", status=1, userName="", type=0): Observable<Article[]> {

this.userProfileModel = new UserProfileModel();

this.userProfileModel.LoadedIds = loadedIds;

this.userProfileModel.DislikeGroups = dislikeGroups;

this.userProfileModel.DislikeUserNames = dislikeUserNames;

this.userProfileModel.Locale = locale;

this.userProfileModel.Menu = menu;

this.userProfileModel.Token = "";

this.userProfileModel.Sub = "";

this.userProfileModel.User = userName;

this.userProfileModel.Keyword = keyword;

this.userProfileModel.Type = type;

this.userProfileModel.Status = status;

return this.http.post<Article[]>(this.articlesUrl, this.userProfileModel)//httpOptions)

.pipe(

tap(data => this.log('fetched subs')),

catchError(this.handleError('getReddahSubs', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private bookmarksUrl = 'https://login.reddah.com/api/article/getbookmarks';

getBookmarks(formData): Observable<any> {

formData.append('jwt', this.getCurrentJwt());

return this.http.post<any>(this.bookmarksUrl, formData)

.pipe(

tap(data => this.log('fetched bookmarks')),

catchError(this.handleError('getReddahBookmarks', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

setCurrentUser(userName: string){

this.localStorageService.store("Reddah\_CurrentUser",userName);

}

getCurrentUser(){

return this.localStorageService.retrieve("Reddah\_CurrentUser");

}

clearCurrentUser(){

this.localStorageService.clear("Reddah\_CurrentUser");

this.localStorageService.clear("Reddah\_CurrentJwt");

}

logoutClear(){

this.clearCurrentUser();

this.cacheService.clearAll();

}

setCurrentJwt(jwt: string){

this.localStorageService.store("Reddah\_CurrentJwt",jwt);

}

getCurrentJwt(){

return this.localStorageService.retrieve("Reddah\_CurrentJwt");

}

clearCurrentJwt(){

this.localStorageService.clear("Reddah\_CurrentJwt");

}

getCurrentLocale(){

let locale = this.localStorageService.retrieve("Reddah\_Locale");

if(locale==undefined||locale==null)

locale = "en-US";

return locale;

}

setLoginUserName(userName){

this.localStorageService.store("Reddah\_lastlogin\_username", userName);

}

getLoginUserName(){

return this.localStorageService.retrieve("Reddah\_lastlogin\_username");

}

getLastLoginUserEmail(){

let lastLoginUser = this.getLoginUserName();

return "";

}

private getSecurityTokenUrl = 'https://login.reddah.com/api/auth/generatetoken';

getSecurityToken(formData){

return this.http.post<any>(this.getSecurityTokenUrl, formData)

.pipe(

tap(data => this.log('get security token')),

catchError(this.handleError('get security token', []))

);

}

private resetPasswordUrl = 'https://login.reddah.com/api/auth/resetpassword';

resetPassword(formData){

return this.http.post<any>(this.resetPasswordUrl, formData)

.pipe(

tap(data => this.log('reset password')),

catchError(this.handleError('reset password', []))

);

}

/\*\*

\* Handle Http operation that failed.

\* Let the app continue.

\* @param operation - name of the operation that failed

\* @param result - optional value to return as the observable result

\*/

private handleError<T> (operation = 'operation', result?: T) {

return (error: any): Observable<T> => {

//alert(JSON.stringify(error));

let msg = error.message;

if(msg.indexOf("failure response")>0)

this.toast(this.translate.instant("Input.Error.NetworkError"), "danger")

if(msg.indexOf("ERR\_TIMED\_OUT")>0)

this.toast(this.translate.instant("Input.Error.ServiceError"), "danger")

// TODO: send the error to remote logging infrastructure

console.error(error); // log to console instead

// TODO: better job of transforming error for user consumption

this.log(`${operation} failed: ${error.message}`);

// Let the app keep running by returning an empty result.

return of(result as T);

};

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

options: any;

//d表示日线，m表示月线，w表示周线

getStock(stock:string, period='d'): Observable<any> {

let sohuStockApi = `https://q.stock.sohu.com/hisHq?code=cn\_${stock}&start=19900716&end=20200720&stat=1&order=A&period=${period}&rt=jsonp`;

// console.log(sohuStockApi);

const searchParams = new URLSearchParams();

searchParams.append('callback', 'JSONP\_CALLBACK');

if (!this.options) {

this.options = {headers: new Headers()};

}

this.options.headers.set('Content-Type', 'application/json:charset=UTF-8');

this.options.params = searchParams;

return this.jsonp.get(sohuStockApi, this.options).pipe(

tap(data => this.log('get stock')),

catchError(this.handleError('get stock', []))

);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

getNearby(lat, lon): Observable<any> {

let policy = 1;

let qqMapApi = `https://apis.map.qq.com/ws/geocoder/v1/?location=${lat},${lon}&output=jsonp&key=ARIBZ-BSK6D-2IL4Y-POZPV-ANU32-CIF56&poi\_options=address\_format=short;radius=5000;page\_size=100;page\_index=1;policy=${policy}&get\_poi=1`;

const searchParams = new URLSearchParams();

searchParams.append('callback', 'JSONP\_CALLBACK');

if (!this.options) {

this.options = {headers: new Headers()};

}

this.options.headers.set('Content-Type', 'application/json:charset=UTF-8');

this.options.params = searchParams;

return this.jsonp.get(qqMapApi, this.options).pipe(

tap(data => this.log('get nearby')),

catchError(this.handleError('get nearby', []))

);

}

getQqLocation(lat, lng): Observable<any> {

let qqTranslateApi = `https://apis.map.qq.com/ws/coord/v1/translate?locations=${lat},${lng}&output=jsonp&type=1&key=ARIBZ-BSK6D-2IL4Y-POZPV-ANU32-CIF56`;

const searchParams = new URLSearchParams();

searchParams.append('callback', 'JSONP\_CALLBACK');

if (!this.options) {

this.options = {headers: new Headers()};

}

this.options.headers.set('Content-Type', 'application/json:charset=UTF-8');

this.options.params = searchParams;

return this.jsonp.get(qqTranslateApi, this.options).pipe(

tap(data => this.log('get qq location')),

catchError(this.handleError('get qq location', []))

);

}

//util functions

//share with my timeline & user timeline

drawCanvasBackground(src){

src="assets/icon/timg.jpg";

//src= (<any>window).Ionic.WebView.convertFileSrc(src);

//cavas use local path, web url has cors issue.

var p = document.getElementById("mycontent");

var canvas = document.createElement('canvas');

var context = canvas.getContext("2d");

var img = new Image(3000,3);

img.src = src;

context.drawImage(img, 0, 0);

var imgData = context.getImageData(1, 1, img.width, 3);

//console.log(imgData.data)

//console.log(`r:${imgData.data[0]},g:${imgData.data[1]},b:${imgData.data[2]}`);

var canvas1 = document.createElement('canvas');

canvas1.style.position = "absolute";

canvas1.style.width = "100%";

canvas1.style.zIndex = "-100";

p.parentElement.appendChild(canvas1);

var ctx = canvas1.getContext("2d");

for(let i=0;i<90;i++){

ctx.putImageData(imgData, 0, 3\*i);

}

}

htmlDecode(text: string) {

var temp = document.createElement("div");

temp.innerHTML = text;

var output = temp.innerText || temp.textContent;

temp = null;

output = output.replace(/\"\/uploadPhoto/g, "\"\/\/\/reddah.com\/uploadPhoto");

output = output.replace(/\"\/\/\/reddah.com\/uploadPhoto/g, "\"https:\/\/reddah.com\/uploadPhoto");

output = output.replace(/\"\/\/\/login.reddah.com\/uploadPhoto/g, "\"https:\/\/login.reddah.com\/uploadPhoto");

return output;

}

subpost(str: string, n: number) {

var r = /[^\u4e00-\u9fa5]/g;

if (str.replace(r, "mm").length <= n) { return str; }

var m = Math.floor(n/2);

for (var i = m; i < str.length; i++) {

if (str.substr(0, i).replace(r, "mm").length >= n) {

return str.substr(0, i) + "...";

}

}

return str;

}

doubleByteLocale = ["zh-cn","zh-tw","ja-jp","ko-kr"];

summary(str: string, n: number, locale='en-US') {

if(locale==null)

locale='en-US';

locale = locale.toLowerCase();

if(!this.doubleByteLocale.includes(locale))

n = 2\*n;

str = this.htmlDecode(this.htmlDecode(str))

.replace(/<br>/g,"\n")

.replace(/<[^>]+>/g, "")

.replace(/\n\n/g,"<br>");

if(str.startsWith("<br>"))

str = str.replace("<br>", "");

return this.subpost(str, n);

}

summaryMsg(str: string, n: number, locale='en-US') {

return this.summary(str, n, locale).replace(/<[^>]+>/g, "");

}

playVideo(id: string) {

let v = document.querySelector('#video\_' + id)[0];

if(v){

if (v.paused) {

v.play();

} else {

v.pause();

}

}

}

/\*trustAsResourceUrl = function (url) {

return $sce.trustAsResourceUrl(url);

}\*/

QrUserKey = 'https://reddah.com/apk/reddah.apk?username=';

//appPhoto = {};

appCacheToOrg = {};

appData(cacheKey, userPhotoName="anonymous.png"){

let storedKey = cacheKey.replace("///","https://")

let result = this.localStorageService.retrieve(storedKey);

if(cacheKey.indexOf('userphoto\_')>-1){

if(this.platform.is('android')){

if(result)

return (<any>window).Ionic.WebView.convertFileSrc(result);

else

return "assets/icon/"+userPhotoName;

}

else{

let url = this.localStorageService.retrieve(cacheKey+"\_url");

if(url)

return url

return "assets/icon/"+userPhotoName;

}

}

else if(cacheKey.indexOf('cover\_')>-1){

if(this.platform.is('android')){

if(result)

return (<any>window).Ionic.WebView.convertFileSrc(result);

else

return "assets/icon/timg.jpg";

}

else{

let url = this.localStorageService.retrieve(cacheKey+"\_url");

if(url)

return url

return "assets/icon/timg.jpg";

}

}

else{//pure text

return result==null ? "": result;

}

}

appDataUserPhoto(cacheKey){

let result = this.localStorageService.retrieve(cacheKey);

if(result&&this.platform.is('cordova')){

return (<any>window).Ionic.WebView.convertFileSrc(result);

}

let url = this.localStorageService.retrieve(cacheKey+"\_url");

if(url)

return url.replace("///","https://");

return "assets/icon/anonymous.png";

}

appDataMap(cacheKey, url){

let result = this.localStorageService.retrieve(cacheKey);

if(result&&this.platform.is('cordova')){

return (<any>window).Ionic.WebView.convertFileSrc(result);

}

if(url)

return url.replace("///","https://");

return "assets/icon/anonymous.png";

}

async appData2(cacheKey){

let result = this.localStorageService.retrieve(cacheKey);

return await (<any>window).Ionic.WebView.convertFileSrc(result);

}

async appData1(cacheKey){

let result = this.localStorageService.retrieve(cacheKey);

if(cacheKey.indexOf('userphoto\_')>-1){

if(result&&this.platform.is('cordova')){

return await (<any>window).Ionic.WebView.convertFileSrc(result);

}

else{

return "assets/icon/anonymous.png";

}

}

else if(cacheKey.indexOf('cover\_')>-1){

if(result&&this.platform.is('cordova')){

return await (<any>window).Ionic.WebView.convertFileSrc(result);

}

else{

return "assets/icon/timg.jpg";

}

}

else{//pure text

return result==null ? "": result;

}

}

level2Cache(cacheKey){

if(this.platform.is('android')){

let storekey = cacheKey.replace("///","https://")

let preview = this.localStorageService.retrieve(storekey);

let org = this.localStorageService.retrieve(storekey.replace("\_reddah\_preview",""))

if(org){

return (<any>window).Ionic.WebView.convertFileSrc(org);

}

else if(preview)

{

return (<any>window).Ionic.WebView.convertFileSrc(preview);

}

else

{

return cacheKey.replace("///","https://");

}

}

else{

return cacheKey.replace("///","https://");

}

}

chatImageCache(cacheKey){

let preview = this.localStorageService.retrieve(cacheKey);

let org = this.localStorageService.retrieve(cacheKey.replace("\_reddah\_preview",""))

if(org)

return (<any>window).Ionic.WebView.convertFileSrc(org);

else if(preview)

return (<any>window).Ionic.WebView.convertFileSrc(preview);

else

{

if(this.platform.is('cordova'))

this.toFileCache(cacheKey);

return cacheKey;

}

}

getFileName(url){

let start = url.lastIndexOf('/')+1;

let end = url.indexOf('?');

end = (end==-1)?url.length:end;

return url.substring(start, end);

}

private fileTransfer: FileTransferObject;

toFileCache(webUrl, isVideo=false){

let deviceDirectory = this.getDeviceDirectory();

let cachedFilePath = this.localStorageService.retrieve(webUrl);

//alert("webUrl:"+webUrl+cachedFilePath);

if(cachedFilePath==null){

webUrl = webUrl.replace("///","https://");

let webFileName = this.getFileName(webUrl);

let targetUrl = deviceDirectory+"reddah/" + webFileName;

//alert(webFileName+"\_\_\_\_"+targetUrl)

this.fileTransfer = this.transfer.create();

this.fileTransfer.download(webUrl, targetUrl).then(

\_ => {

if(isVideo)

{

//this.localStorageService.store(webUrl,(<any>window).Ionic.WebView.convertFileSrc(targetUrl));

this.localStorageService.store(webUrl, targetUrl);

}

else{

this.localStorageService.store(webUrl, targetUrl);

this.appCacheToOrg[(<any>window).Ionic.WebView.convertFileSrc(targetUrl)] = webUrl;

}

//alert(webUrl)

},

\_ => {

//alert(JSON.stringify(\_))

console.log(JSON.stringify(\_)) });

}

}

getDeviceDirectory(){

let dir = this.file.externalRootDirectory;

if(this.platform.is('android'))

{

dir = this.file.externalRootDirectory;

}

else if(this.platform.is('ipad')||this.platform.is('iphone')||this.platform.is('ios')){

dir = this.file.cacheDirectory;

}

else {

}

return dir;

}

toImageCache(webUrl, cacheKey){

let deviceDirectory = this.getDeviceDirectory();

let cachedImagePath = this.localStorageService.retrieve(cacheKey);

let cacheImageName = "";

if(cachedImagePath!=null){

cacheImageName = cachedImagePath.replace(deviceDirectory+"reddah/","");

}

webUrl = webUrl.replace("///","https://");

let webImageName = this.getFileName(webUrl);

if(cachedImagePath==null||cacheImageName!=webImageName){

this.fileTransfer = this.transfer.create();

let targetUrl = deviceDirectory+"reddah/" + webImageName;

this.fileTransfer.download(webUrl, targetUrl).then(

\_ => {

this.localStorageService.store(cacheKey, targetUrl);

this.appCacheToOrg[(<any>window).Ionic.WebView.convertFileSrc(targetUrl)] = webUrl;

},

\_ => { console.log(JSON.stringify(\_)); });

}

}

toTextCache(text, cacheKey){

if(text!=null){

let cachedText = this.localStorageService.retrieve(cacheKey);

if(cachedText!=text){

this.localStorageService.store(cacheKey, text);

}

}

}

verifyImageFile(key){

let cachedPath = this.localStorageService.retrieve(key);

if(cachedPath&&this.platform.is('android')){

let fileName = this.getFileName(cachedPath);

let filePath = cachedPath.replace(fileName, "");

this.file.checkFile(filePath, fileName).then(

(files) => {

console.log("files found" + files)

}

).catch(

(err) => {

this.localStorageService.clear(key);

}

);

}

}

async getUserPhotos(userName, isTimeline=false){

if(userName==null)

return;

try{

this.verifyImageFile(`cover\_${userName}`);

this.verifyImageFile(`userphoto\_${userName}`);

//check from web

let formData = new FormData();

formData.append("targetUser", userName);

await this.getUserInfo(formData)

.subscribe(userInfo =>

{

if(userInfo.Cover!=null){

this.toImageCache(userInfo.Cover, `cover\_${userName}`);

this.localStorageService.store(`cover\_${userName}\_url`,userInfo.Cover);

}

if(userInfo.Photo!=null){

this.toImageCache(userInfo.Photo, `userphoto\_${userName}`);

this.localStorageService.store(`userphoto\_${userName}\_url`,userInfo.Photo)

}

if(userInfo.NickName!=null)

this.toTextCache(userInfo.NickName, `usernickname\_${userName}`);

if(userInfo.Sex!=null)

this.toTextCache(userInfo.Sex, `usersex\_${userName}`);

if(userInfo.Location!=null){

let locationTitle = JSON.parse(userInfo.Location).title;

this.toTextCache(locationTitle, `userlocation\_${userName}`);

this.toTextCache(userInfo.Location, `userlocationjson\_${userName}`);

}

if(userInfo.Admins!=null)

this.toTextCache(userInfo.Admins, `useradmins\_${userName}`);

if(userInfo.Point!=null)

this.toTextCache(userInfo.Point, `userpoint\_${userName}`);

if(userInfo.Signature!=null)

this.toTextCache(userInfo.Signature, `usersignature\_${userName}`);

if(userInfo.Email!=null)

this.toTextCache(userInfo.Email, `useremail\_${userName}`);

if(userInfo.Type!=null)

this.toTextCache(userInfo.Type, `usertype\_${userName}`);

if(userInfo.NoteName!=null)

this.toTextCache(userInfo.NoteName, `usernotename\_${userName}\_${this.getCurrentUser()}`);

if(userInfo.UserName!=this.getCurrentUser()){

this.toTextCache(userInfo.IsFriend?1:0, `userisfriend\_${userName}\_${this.getCurrentUser()}`);

}

if(userInfo.HideLocation!=null){

//alert(userInfo.HideLocation);

this.toTextCache(userInfo.HideLocation?1:0, `userhidelocation\_${userName}`);

//alert(this.appData(`userhidelocation\_${userName}`));

}

if(userInfo.AllowTenTimeline!=null)

{

//alert(userInfo.AllowTenTimeline);

this.toTextCache(userInfo.AllowTenTimeline?1:0, `userallowtentimeline\_${userName}`);

//alert(this.appData(`userallowtentimeline\_${userName}`));

}

});

}

catch(error){

alert(error)

}

}

GetCache(url){

if(this.platform.is('cordova')){

let org = this.localStorageService.retrieve(url);

if(org){

return (<any>window).Ionic.WebView.convertFileSrc(org);

}

}

else

{

return url;

}

}

getSortLetter(text, locale){

if(!String.prototype.localeCompare)

return null;

let letters = "\*abcdefghjklmnopqrstwxyz".split('');

let zh = "阿八嚓哒妸发旮哈讥咔垃痳拏噢妑七呥扨它穵夕丫帀".split('');

let result = '#';

letters.forEach((letter, i) => {

if((!zh[i-1] || zh[i-1].localeCompare(text, locale) <= 0) && text.localeCompare(zh[i], locale) == -1) {

result = letter;

}

});

return result;

}

complement(n) { return n < 10 ? '0' + n : n }

generateFileName() {

var date = new Date();

return date.getFullYear().toString() + this.complement(date.getMonth() + 1) + this.complement(date.getDate()) + this.complement(date.getHours()) + this.complement(date.getMinutes()) + this.complement(date.getSeconds());

}

getDisplayName(userName, count=15){

let currentUserName = this.getCurrentUser();

let name = this.appData('usernotename\_'+userName+"\_"+currentUserName) ? this.appData('usernotename\_'+userName+"\_"+currentUserName) :

(this.appData('usernickname\_'+userName) ? this.appData('usernickname\_'+userName) : userName);

return this.summary(name, count, this.getCurrentLocale());

}

getArray(n){

if(n==null||n<0)

n=0;

return new Array(n);

}

getSendTime(dateStr){

if(!dateStr)

return "";

let dateTimeStamp = Date.parse(dateStr.replace(/-/gi,"/"));

let result = "";

let second = 1000;

let minute = 1000 \* 60;

let hour = minute \* 60;

let day = hour \* 24;

let halfamonth = day \* 15;

let month = day \* 30;

let year = day \*365;

let localnow = new Date().getTime();

//to utc now

let offset = new Date().getTimezoneOffset()\*60\*1000;

let now = new Date(localnow+offset).getTime();

let diffValue = now - dateTimeStamp;

if(diffValue < 0){

if(diffValue>-1000\*60)

return this.translate.instant("Time.JustNow");

else

return result;

}

let yearC =diffValue/year;

let monthC =diffValue/month;

let weekC =diffValue/(7\*day);

let dayC =diffValue/day;

let hourC =diffValue/hour;

let minC =diffValue/minute;

let secC =diffValue/second;

let split = "";

if(!this.doubleByteLocale.includes(this.getCurrentLocale().toLowerCase())){

split=" ";

}

if(yearC>=1){

result=parseInt(yearC+"") + split +this.translate.instant("Time.YearsAgo");

}

else if(monthC>=1){

result=parseInt(monthC+"") + split + this.translate.instant("Time.MonthsAgo");

}

else if(weekC>=1){

result=parseInt(weekC+"") + split + this.translate.instant("Time.WeeksAgo");

}

else if(dayC>=1){

result=(parseInt(dayC+"")==1?this.translate.instant("Time.Yesterday"):parseInt(dayC+"") + split +this.translate.instant("Time.DaysAgo"));

}

else if(hourC>=1){

result=parseInt(hourC+"") + split +this.translate.instant("Time.HoursAgo");

}

else if(minC>=1){

result=parseInt(minC+"") + split +this.translate.instant("Time.MinutesAgo");

}

else if(secC>=1){

result=this.translate.instant("Time.JustNow");

}

return result;

}

getSendTimeShort(dateStr){

if(!dateStr)

return "";

let dateTimeStamp = Date.parse(dateStr.replace(/-/gi,"/"));

let result = "";

let second = 1000;

let minute = 1000 \* 60;

let hour = minute \* 60;

let day = hour \* 24;

let halfamonth = day \* 15;

let month = day \* 30;

let year = day \*365;

let localnow = new Date().getTime();

//to utc now

let offset = new Date().getTimezoneOffset()\*60\*1000;

let now = new Date(localnow+offset).getTime();

let diffValue = now - dateTimeStamp;

if(diffValue < 0){

if(diffValue>-1000\*60)

return this.translate.instant("Time.Today");

else

return dateStr;

}

let yearC =diffValue/year;

let monthC =diffValue/month;

let weekC =diffValue/(7\*day);

let dayC =diffValue/day;

let hourC =diffValue/hour;

let minC =diffValue/minute;

let secC =diffValue/second;

if(yearC>=1){

return dateStr;

}

else if(monthC>=1){

return dateStr;

}

else if(weekC>=1){

return dateStr;

}

else if(dayC>=1){

result=(parseInt(dayC+"")==1?this.translate.instant("Time.Yesterday"):dateStr);

}

else if(hourC>=1){

return this.translate.instant("Time.Today");

}

else if(minC>=1){

return this.translate.instant("Time.Today");

}

else if(secC>=1){

return this.translate.instant("Time.Today");

}

return dateStr;

}

getSendTimeChat(dateStr){

if(!dateStr)

return "";

let dateTimeStamp = Date.parse(dateStr.replace(/-/gi,"/"));

let result = "";

let second = 1000;

let minute = 1000 \* 60;

let hour = minute \* 60;

let day = hour \* 24;

let halfamonth = day \* 15;

let month = day \* 30;

let year = day \*365;

let localnow = new Date().getTime();

//to utc now

let offset = new Date().getTimezoneOffset()\*60\*1000;

let now = new Date(localnow+offset).getTime();

let diffValue = now - dateTimeStamp;

if(diffValue < 0){

if(diffValue>-1000\*60)

return this.utcToLocal(dateStr,'HH:mm');

else

return result;

}

let yearC =diffValue/year;

let monthC =diffValue/month;

let weekC =diffValue/(7\*day);

let dayC =diffValue/day;

let hourC =diffValue/hour;

let minC =diffValue/minute;

let secC =diffValue/second;

if(yearC>=1){

result=parseInt(yearC+"") + "" +this.translate.instant("Time.YearsAgo");

}

else if(monthC>=1){

result=parseInt(monthC+"") + "" + this.translate.instant("Time.MonthsAgo");

}

else if(weekC>=1){

result=parseInt(weekC+"") + "" + this.translate.instant("Time.WeeksAgo");

}

else if(dayC>=1){

result=(parseInt(dayC+"")==1?this.translate.instant("Time.Yesterday"):parseInt(dayC+"") + "" +this.translate.instant("Time.DaysAgo"));

}

else if(hourC>=1){

result=this.utcToLocal(dateStr,'HH:mm');

}

else if(minC>=1){

result=this.utcToLocal(dateStr,'HH:mm');

}

else if(secC>=1){

result=this.utcToLocal(dateStr,'HH:mm');

}

return result;

}

async loadFriends(){

let cachedContact = this.localStorageService.retrieve("Reddah\_Contacts");

if(!cachedContact){

let cacheKey = "this.reddah.getFriends";

let request = this.getFriends();

this.cacheService.loadFromObservable(cacheKey, request, "ContactPage")

.subscribe(contacts =>

{

this.localStorageService.store("Reddah\_Contacts", JSON.stringify(contacts));

});

}

}

getAllowedNames(groupName){

let result = [];

let cachedContact = this.localStorageService.retrieve("Reddah\_Contacts");

if(cachedContact){

let friends = JSON.parse(cachedContact).map(x=>x.Watch);

let groupNames = groupName.split(',')

groupNames.forEach((name)=>{

if(friends.includes(name)||name==this.getCurrentUser())

result.push({userName: name, displayName: this.getDisplayName(name)});

});

}

return result;

}

getAllowedComments(data){

let cachedContact = this.localStorageService.retrieve("Reddah\_Contacts");

if(cachedContact){

let friends = JSON.parse(cachedContact).map(x=>x.Watch);

return data.Comments.filter(c=>(friends.includes(c.UserName)||c.UserName==this.getCurrentUser())&&data.Seed==c.ParentId);

}

return [];

}

unReadMessage = [];

//unReadMessage(){

// return [{userName:'wind', type:0},{userName:'duck6686', type:0}];

//}

storeReadMessage(){

let readMessages = this.localStorageService.retrieve("Reddah\_ReadMessages");

if(!readMessages)

readMessages = []

readMessages = readMessages.concat(this.unReadMessage);

this.localStorageService.store("Reddah\_ReadMessages", readMessages);

this.unReadMessage = [];

}

getStoredMessage(){

let readMessages = this.localStorageService.retrieve("Reddah\_ReadMessages");

if(!readMessages)

readMessages = [];

return readMessages;

}

clearStoredMessage(){

this.localStorageService.clear("Reddah\_ReadMessages");

}

getRandomInt(max) {

return Math.floor(Math.random() \* Math.floor(max));

}

getRandomArray(m, n){

let a = [];

for(let i=0;i<n;i++)

{

a.push(i);

}

let r=[];

for (let j=1; j<=m; ++j)

{

var k = Math.floor((Math.random() \* (n-j)) + 1);

r.push(a[k]);

a[k] = a[n-j];

}

return r;

}

fontSizeMap = new Map()

.set(1,'12px')

.set(2,'13px')

.set(3,'14px')

.set(4,'15px')

.set(5,'16px')

.set(6,'17px')

.set(7,'18px')

.set(8,'19px')

.set(9,'20px')

.set(10,'21px');

async toast(message: string, color="dark") {

const toast = await this.toastController.create({

message: message,

showCloseButton: true,

position: 'top',

closeButtonText: this.translate.instant("Button.Close"),

duration: 3000,

color: color,

cssClass: "toast-style"

});

toast.present();

}

async addBookmark(articleId){

let formData = new FormData();

formData.append("ArticleId", JSON.stringify(articleId));

this.addBookmarkFormData(formData);

}

async addBookmarkFormData(formData){

let text = `${this.translate.instant("Pop.Marked")}:${this.translate.instant("Menu.About")}/${this.translate.instant("Menu.Mark")}`;

this.bookmark(formData).subscribe(result=>{

if(result.Success==0)

{

this.toast(text, "primary");

this.cacheService.clearGroup("BookmarkPage");

}

else{

alert(JSON.stringify(result.Message));

}

})

}

isLocal(videoUrl){

let key = videoUrl.toString().toLowerCase();

let localPath = this.localStorageService.retrieve(key);

return localPath!=null;

}

clearLocaleCache(){

this.localStorageService.clear("reddah\_articles");

this.localStorageService.clear("reddah\_article\_ids");

this.localStorageService.clear("reddah\_article\_groups");

this.localStorageService.clear("reddah\_article\_usernames");

this.cacheService.clearGroup("HomePage");

}

async ClearPub(){

this.localStorageService.clear("Reddah\_GroupedSubs");

this.localStorageService.clear("Reddah\_Subs");

this.cacheService.clearGroup("ManageSubsPage");

}

getJSON(text){

if(text==null||text=='undefined')

return "";

return JSON.parse(text);

}

openMini(webUrl, miniName){

let deviceDirectory = this.getDeviceDirectory();

this.fileTransfer = this.transfer.create();

let targetUrl = deviceDirectory+"reddah/mini/" + miniName;

this.fileTransfer.download(webUrl, targetUrl).then(

\_ => {

let localUrl = (<any>window).Ionic.WebView.convertFileSrc(targetUrl);

const browser = this.iab.create(localUrl, 'location=no');

browser.show();

},

\_ => { console.log(JSON.stringify(\_)) });

}

utcToLocal(str, format="YYYY-MM-DD HH:mm:ss"){

let localTime = moment.utc(str).toDate();

return moment(localTime).format(format).toString();

}

setRecent(user, type){

let recent = this.localStorageService.retrieve(`Reddah\_Recent\_${type}`);

if(!recent||recent.length==0){

recent = [];

recent.push(user);

}

else{

recent.forEach((item,index)=>{

if(item.UserId==user.UserId){

recent.splice(index, 1);

}

});

recent.unshift(user);

}

this.localStorageService.store(`Reddah\_Recent\_${type}`, recent);

}

loadRecent(type){

let recent = this.localStorageService.retrieve(`Reddah\_Recent\_${type}`);

if(!recent)

recent = [];

return recent;

}

hasClass(el: Element, name: string): any {

return new RegExp("(?:^|\\s+)" + name + "(?:\\s+|$)").test(el.className);

}

addClass(el: Element, name: string): void {

if (!this.hasClass(el, name)) {

el.className = el.className ? [el.className, name].join(" ") : name;

}

}

removeClass(el: Element, name: string): void {

if (this.hasClass(el, name)) {

el.className = el.className.replace(

new RegExp("(?:^|\\s+)" + name + "(?:\\s+|$)", "g"),

""

);

}

}

getHourString(){

let today = new Date();

let date = today.getFullYear()+'-'+(today.getMonth()+1)+'-'+today.getDate();

let time = today.getHours();

return date + " " + time;

}

getTimeString(){

let today = new Date();

let date = today.getFullYear()+'-'+(today.getMonth()+1)+'-'+today.getDate();

let time = today.getHours() + ":" + today.getMinutes();// + ":" + today.getSeconds();

return date + " " + time;

}

getRuffDistanceByDegree(degree){

if(degree<0)

degree=degree\*-1;

let meters = parseInt(111\*1000\*degree+"");

if(meters>1000){

return parseInt(meters/1000+"") + this.translate.instant("Pop.KM")

}

return meters + this.translate.instant("Pop.M");

}

async adjustImage(url, flag){

let imgId = this.makeItId(url+flag);

let image = document.getElementById(imgId);

if(image){

if(image.offsetHeight<image.offsetWidth)

{

image.style.height = "100%";

this.localStorageService.store(imgId+"adjusth", true);

}

else{

if(flag=="home"){

let result = this.localStorageService.retrieve(imgId+"adjusth");

if(result)

{

image.style.height = "100%";

}

else

{

image.style.height = "";

}

}

else{

image.style.height = "";

}

}

}

}

makeItId(text){

return text.replace(/[\/:,.%-]/g,'\_');

}

uuidv4() {

return 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g, function(c) {

var r = Math.random() \* 16 | 0, v = c == 'x' ? r : (r & 0x3 | 0x8);

return v.toString(16);

});

}

async takePhoto(photos, formData){

const options: CameraOptions = {

quality: 100,

destinationType: Camera.DestinationType.FILE\_URI,

encodingType: Camera.EncodingType.JPEG,

mediaType: Camera.MediaType.PICTURE,

correctOrientation: true

}

Camera.getPicture(options).then((imageData) => {

let data = {fileUrl: imageData, webUrl: (<any>window).Ionic.WebView.convertFileSrc(imageData)};

photos.push(data);

this.addPhotoToFormData(data, formData);

}, (err) => {

//alert(JSON.stringify(err));

});

}

async fromLibPhoto(photos, formData)

{

const options: CameraOptions = {

quality: 100,

destinationType: Camera.DestinationType.FILE\_URI,

encodingType: Camera.EncodingType.JPEG,

mediaType: Camera.MediaType.PICTURE,

sourceType: Camera.PictureSourceType.PHOTOLIBRARY,

correctOrientation: true

}

Camera.getPicture(options).then((imageData) => {

let data = {fileUrl: imageData, webUrl: (<any>window).Ionic.WebView.convertFileSrc(imageData)};

photos.push(data);

this.addPhotoToFormData(data, formData);

}, (err) => {

//alert(JSON.stringify(err));

});

}

addPhotoToFormData(photo, formData){

//append org photo form data

this.prepareData(photo.fileUrl, photo.fileUrl, formData);

//append preview photo form data

let orgFileName = photo.fileUrl.substring(photo.fileUrl.lastIndexOf('/')+1);

let fileExtention = orgFileName.substring(orgFileName.lastIndexOf('.'));

//remove ?\*\*\*\*

let removdQFileExtention = fileExtention.lastIndexOf('?')==-1 ?

fileExtention : fileExtention.replace(fileExtention.substring(fileExtention.lastIndexOf('?')),"");

let previewFileName = orgFileName.replace(fileExtention,"") + "\_reddah\_preview" + removdQFileExtention;

//alert(photo.fileUrl+"\_"+previewFileName);

let options = {

uri: photo.fileUrl,

folderName: 'reddah',

fileName: previewFileName,

quality: 20,

width: 800,

height: 800

} as ImageResizerOptions;

ImageResizer

.resize(options)

.then((filePath: string) => this.prepareData(filePath, photo.fileUrl+"\_reddah\_preview", formData))

.catch(e => alert(e));

}

prepareData(filePath, formKey, formData) {

this.file.resolveLocalFilesystemUrl(filePath)

.then(entry => {

( <FileEntry> entry).file(file => {

const reader = new FileReader();

reader.onloadend = () => {

//org image data

const imgBlob = new Blob([reader.result], {

type: file.type

});

formData.append(formKey, imgBlob, file.name);

};

reader.readAsArrayBuffer(file);

})

})

.catch(err => {

console.error(JSON.stringify(err));

});

}

checkPermission(permissionId){

let jwt = this.getCurrentJwt();

let parts = jwt.split('.');

let bodyEnc = parts[1];

if(!bodyEnc){

return false;

}

let bodyStr = atob(bodyEnc)

, body;

try{

body = JSON.parse(bodyStr);

}

catch(e){

body = {};

}

let exp = body.exp

, user= body.aud

;

if(this.isExpired(exp)){

return false;

}

//console.log(body)

if(body[permissionId]!=null)

return true;

return false;

}

isExpired(exp:number): boolean {

if(!exp) return true;

let now = Date.now();

return now >= exp\*1000;

}

preImageArray(imageSrcArray){

return imageSrcArray.map(item=>{

return {

webPreviewUrl: this.parseImage(item)

}

});

}

parseImage(url){

url = url.replace("///", "https://");

return url;

}

publishers = new Set<string>();

async checkIsToday(date){

let cur = new Date(date);

return cur.getDate()==new Date().getDate();

}

windowReload(){

if(this.platform.is('android')){

window.location.reload();

}

else{

this.modalController.dismiss()

this.router.navigate([''], {

queryParams: {

action: 'login'

}

});

}

}

}

En-US.json

{

"Surface":{

"Language": "Language",

"Login": "Login",

"Register": "Register"

},

"Login":{

"Title":"Login",

"NoAccount":"No Account?",

"RegisterNow":"Register Now",

"ForgotPassword":"Forgot Password",

"Loading":"Login..."

},

"Register":{

"Title":"Register",

"TitleSub":"Register publisher",

"TitleMini":"Register mini app",

"ReadAgree":"You have read and agreed",

"UserAgreement":"User Agreement,",

"Privacy":"Privacy Policy",

"And":"&",

"Content":"Content Policy",

"Loading":"Register...",

"Success":"Registration is successful. Please verify your email and login",

"Manage":"Manage"

},

"Input":{

"UserName": "User Name",

"Password": "Password",

"OldPassword": "Old Password",

"NewPassword": "New Password",

"ConfirmPassword": "Confirm Password",

"Email": "Email",

"Name":"Name",

"Description":"Description",

"Error":{

"UserNameEmpty":"Please input user name",

"UserNameTooLong":"User name should be less than 25 characters",

"SubNameTooShort":"Name should be at least 2 characters",

"PasswordEmpty":"Please input password",

"PasswordDifferent":"Password and confirm password are different",

"EmailEmpty":"Please input email",

"Agree":"Please agree with user agreement",

"SubNameEmpty":"Please input name",

"SubDescEmpty":"Please input description",

"TokenEmpty":"Please input security token",

"ResetResult":"Reset password successfully, please login",

"GetTokenFromEmail":"Please get security token from your Email",

"NetworkError":"No network connection",

"ServiceError":"The system service is being updated. Please try again later"

}

},

"Button":{

"Login": "Login",

"Register": "Register",

"Logout": "Logout",

"Back": "Back",

"Close": "Close",

"Loading":"loading...",

"Reset":"Forgot Password",

"Token":"Security Token",

"ChangePassword":"Change Password"

},

"Menu":{

"Message": "Message",

"Recommend": "Article",

"Earth":"Earth",

"Contact": "Friend",

"Find": "Explore",

"About": "Me",

"Publisher": "Publisher",

"MiniApp": "Mini App",

"NewFriends": "New Friend",

"GroupChat": "Group Chat",

"Mark": "Mark",

"Settings": "Settings",

"Header":{

"GroupChat":"New Chat",

"AddFriend":"Add Friend",

"Scan":"Scan",

"Feedback":"Feedback"

},

"Timeline":"Aperture",

"Shake":"Shake",

"Platform":"Open Platform"

},

"Search":{

"Title":"Search",

"Filter":"Filter by",

"Article":"Article",

"Aperture":"Aperture",

"Publisher":"Publisher",

"MiniApp":"Mini App",

"Recently":"Recently used",

"More":"More"

},

"About":{

"Account":"Account",

"Privacy":"Privacy",

"General":"General",

"About":"About",

"Photo":"Photo",

"Card":"Card",

"Location":"Location",

"Nickname":"Nickname",

"Sex":"Sex",

"Male":"Male",

"Female":"Female",

"Signature":"Signature",

"CurrentVersion": "Version",

"CheckUpgrade": "Check Latest Version",

"CurrentLocale": "Locale",

"ChangeLocale": "Change Locale",

"CurrentUser": "Current User"

},

"Time.YearsAgo": "year(s) ago",

"Time.MonthsAgo": "month(s) ago",

"Time.WeeksAgo": "week(s) ago",

"Time.Today": "Today",

"Time.Yesterday": "Yesterday",

"Time.DaysAgo": "days ago",

"Time.HoursAgo": "hour(s) ago",

"Time.MinutesAgo": "mins ago",

"Time.JustNow": "Just now",

"Article.Loading": "Please wait",

"Comment.Comment": "Comment",

"Comment.MoreComment":"More Comments",

"Comment.Like": "Like",

"Comment.Delete": "Delete",

"Comment.Tip": "Enter some comment here...",

"Confirm.Title": "Confirm",

"Confirm.Message": "Do you want to exit Reddah?",

"Confirm.ClearChatMessage": "Do you want to clear all the chat messages?",

"Confirm.LogoutMessage":"Do you want to logout?",

"Confirm.LeaveGroupChatMessage":"Do you want to quit and delete?",

"Confirm.DeleteMessage":"Do you want to delete?",

"Confirm.Yes": "Yes",

"Confirm.Cancel": "Cancel",

"Confirm.Submit": "Submit",

"Confirm.ByLatest":"Latest",

"Confirm.ByEarly":"Earliest",

"Confirm.ByLike":"Like",

"Pop":{

"NoRecommend":"Hide",

"Report":"Report",

"ReportReason":"Please input a concrete reason...",

"Nointerest":"Not Interested",

"ToFriend":"Share with friend",

"ToAperture":"Share to Aperture",

"ChangeCover":"Change cover",

"Note":"Note",

"TakePhoto":"Camera",

"Photo":"Take a photo",

"SelectPhoto":"photo from album",

"SelectVideo":"video from album",

"SawIt":"Seen",

"Trash":"Trash",

"Porn":"Porn",

"NoAuthor":"Anti",

"Dislike":"Dislike",

"Suggest":"Suggest",

"ContentWrong":"Content Wrong",

"CodeError":"Program Error",

"ContentInvalid":"Content Invalid",

"ContentOther":"Others",

"Marked":"Marked",

"HideLocation":"Hide Location",

"YourThoughts":"Your thoughts now",

"ReleaseToDel":"Release to delete",

"DragToDel":"Drag to delete",

"FeedbackThoughts":"Write down your issues and suggestions, we'll keep updating",

"Focus":"Focus",

"Unfocus":"Unfocus",

"EnterPub":"Enter",

"StillFocus":"Stay Focus",

"StillUnfocus":"Do not focus",

"UnfocusMessage":"You won't receive its messages any more",

"KM":"KM",

"M":"M",

"Year":"",

"Month":"",

"MsgCount":"Message",

"GroupChatTitle":"Group Name",

"GroupChatAnnouce":"Group Notice",

"GroupChatUpdate":"I created a group chat, let's talk!",

"AddToGrp":"invited to the group chat",

"KickGrp":"removed from the group chat",

"PressSpeak":"Hold to Talk",

"ReleaseSend":"Release to Send",

"FunImageLib":"Album",

"FunCamera":"Camera",

"FunVideoLib":"VideoLib",

"FunVideo":"Video"

},

"Common":{

"Distance":"Distance",

"Refresh":"Refresh",

"Choose":"Choose",

"Font":"Font Size",

"Font1":"Preview font size",

"Font2":"Adjust the slider below to change font size",

"Font3":"Font size in chat, article and aperture will be changed. \"Loveis something eternal; the aspect may change, but not theessence.\"",

"Cache":"Clear Cache",

"Save":"Save",

"FeatureNotOpen":"Features not open yet: Edit, Publish, etc.",

"Draft":"Draft",

"Test":"Test",

"Publish":"Publish",

"View":"View",

"Category":"Account Category",

"PubDesc":"It provides a new way of information dissemination for the media and individuals, and builds a better communication and management model with readers.",

"MiniDesc":"A new open capability can be easily accessed and disseminated in Reddah, and it has excellent experience in use.",

"HideMyLocation":"Do not display my location",

"AllowTenTimeline":"Allow stranger to view 10 Aperture",

"Developer":"Developer",

"ServicePhone":"Service Phone",

"Owner":"Owner",

"OwnerDesc":"Subject certification shall submit certificates certifying the legitimacy of institutions issued by relevant competent authorities, such as business licenses of enterprises, certificates of registration of legal persons of institutions, certificates of registration of legal persons of organs, certificates of registration of legal persons of social organizations, certificates of registration of private non-enterprise units, etc.",

"CacheClear":"Cache is clear",

"NewFriend":"New Friends",

"Accept":"Accept",

"Added":"Added",

"Accepted":"Accepted",

"Option":"Options",

"DeleteAndQuit":"Delete and quit",

"ClearChat":"Clear chat",

"Clear":"Clear",

"AllowTenMessage":"Strangers can only view 10 Aperture messages",

"NoPermissionMessage":"Strangers can't view Aperture messages",

"Details":"Details",

"ScanToAdd":"Scan the QR code to add friend",

"Original":"Original",

"ApplyFriend":"You need to send a application and wait for approval",

"ApplyFriendMe":"I am",

"Label":"Label",

"MaxDraft":"Ordinary users can save up to 3 drafts",

"WorkingCopyExist":"There's an unpublished version",

"MyPhoto":"My Material",

"AddPhoto":"Add Material",

"Copy":"Copy Image Address",

"UserNotExist":"User does not exist"

},

"Mail":{

"Register":{

"Title":"Reddah Account",

"Sub":"Verify your email address‏",

"ParaStart":"Please visit this link to verify your email address",

"ParaEnd":"After that, you can login. Thanks for using Reddah!"

},

"Reset":{

"Title":"Reddah Account",

"Sub":"Security token to reset password",

"ParaStart":"Please copy this token to reset your password",

"ParaEnd":"Thanks for using Reddah!"

}

},

"Service":{

"4": "Service Error, please try again later",

"1001":"User name must contain 6-18 letters and numbers. No special characters allowed",

"1002":"Email is invalid",

"1003":"User name exists",

"1004":"Email is registered",

"1005":"Please go to your mailbox to verify your email",

"1006":"User name or password is wrong",

"1007":"Email is not registered",

"1008":"Reset password failed, please check if security token is correct",

"1009":"Old password is wrong",

"1010":"Please select a material",

"1011":"No permission to delete article",

"1012":"No permission to delete comment"

}

}

namespace Reddah.Web.UI.Filters

{

[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, AllowMultiple = false, Inherited = true)]

public sealed class InitializeSimpleMembershipAttribute : ActionFilterAttribute

{

private static SimpleMembershipInitializer \_initializer;

private static object \_initializerLock = new object();

private static bool \_isInitialized;

public override void OnActionExecuting(ActionExecutingContext filterContext)

{

// Ensure ASP.NET Simple Membership is initialized only once per app start

LazyInitializer.EnsureInitialized(ref \_initializer, ref \_isInitialized, ref \_initializerLock);

}

private class SimpleMembershipInitializer

{

public SimpleMembershipInitializer()

{

Database.SetInitializer<UsersContext>(null);

try

{

WebSecurity.InitializeDatabaseConnection("DefaultConnection", "UserProfile", "UserId", "UserName", autoCreateTables: false);

}

catch (Exception ex)

{

throw new InvalidOperationException("The ASP.NET Simple Membership database could not be initialized. For more information, please see http://go.microsoft.com/fwlink/?LinkId=256588", ex);

}

}

}

}

}

using System;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

using System.Threading;

using System.Web.Mvc;

using WebMatrix.WebData;

using Reddah.Web.UI.Models;

using System.Web.Http;

using System.Threading.Tasks;

using System.Net.Http;

using System.Linq;

namespace Reddah.Web.UI.Filters

{

public class CrosHandler : DelegatingHandler

{

private const string \_origin = "Origin";

private const string \_accessControlRequestMethod = "Access-Control-Request-Method";

private const string \_accessControlRequestHeaders = "Access-Control-Request-Headers";

private const string \_accessControlAllowOrigin = "Access-Control-Allow-Origin";

private const string \_accessControlAllowMethods = "Access-Control-Allow-Methods";

private const string \_accessControlAllowHeaders = "Access-Control-Allow-Headers";

protected override Task<HttpResponseMessage> SendAsync(HttpRequestMessage request, System.Threading.CancellationToken cancellationToken)

{

bool isCrosRequest = request.Headers.Contains(\_origin);

bool isPreflightRequest = request.Method == HttpMethod.Options;

if (isCrosRequest)

{

Task<HttpResponseMessage> taskResult = null;

if (isPreflightRequest)

{

taskResult = Task.Factory.StartNew<HttpResponseMessage>(() =>

{

HttpResponseMessage response = new HttpResponseMessage(System.Net.HttpStatusCode.OK);

response.Headers.Add(\_accessControlAllowOrigin, request.Headers.GetValues(\_origin).FirstOrDefault());

string method = request.Headers.GetValues(\_accessControlRequestMethod).FirstOrDefault();

if (method != null)

{

response.Headers.Add(\_accessControlAllowMethods, method);

}

string headers = string.Join(", ", request.Headers.GetValues(\_accessControlRequestHeaders));

if (!string.IsNullOrEmpty(headers))

{

response.Headers.Add(\_accessControlAllowHeaders, headers);

}

return response;

}, cancellationToken);

}

else

{

taskResult = base.SendAsync(request, cancellationToken)

.ContinueWith<HttpResponseMessage>(t =>

{

var response = t.Result;

response.Headers.Add(\_accessControlAllowOrigin, request.Headers.GetValues(\_origin).FirstOrDefault());

return response;

});

}

return taskResult;

//return base.SendAsync(request, cancellationToken);

}

else

{

return base.SendAsync(request, cancellationToken);

}

}

}

}

using System;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

using System.Threading;

using System.Web.Mvc;

using WebMatrix.WebData;

using Reddah.Web.UI.Models;

namespace Reddah.Web.UI.Controllers

{

[Authorize]

[InitializeSimpleMembership]

public class AccountController : BaseController

{

private readonly log4net.ILog log = log4net.LogManager.GetLogger("AccountController");

//

// GET: /Account/Login

[AllowAnonymous]

public ActionResult Login(string returnUrl)

{

ViewBag.ReturnUrl = returnUrl;

return View();

}

//

// POST: /Account/Login

[AllowAnonymous]

[HttpPost]

[AjaxValidateAntiForgeryToken]

public JsonResult JsonLogin(LoginModel model, string returnUrl)

{

if (ModelState.IsValid)

{

if (WebSecurity.Login(model.UserName, model.Password, persistCookie: model.RememberMe))

{

return Json(new { success = true, redirect = returnUrl });

}

else

{

ModelState.AddModelError("", Resources.Resources.Login\_Msg\_WrongPW);

}

}

return Json(new { errors = GetErrorsFromModelState() });

}

[HttpPost, CaptchaVerify("Captcha is not valid")]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public ActionResult Login(LoginModel model, string returnUrl)

{

if (ModelState.IsValid && WebSecurity.Login(model.UserName, model.Password, persistCookie: model.RememberMe))

{

return RedirectToLocal(returnUrl);

}

// If we got this far, something failed, redisplay form

ModelState.AddModelError("", Resources.Resources.Login\_Msg\_WrongPW);

return View(model);

}

//

// POST: /Account/LogOff

[AllowAnonymous]

public ActionResult LogOff()

{

WebSecurity.Logout();

return Redirect("/"+ CultureInfo.CurrentUICulture.Name+"?logout=true");

}

//

// GET: /Account/Register

[AllowAnonymous]

public ActionResult Register()

{

return View();

}

//

// POST: /Account/Register

[HttpPost, CaptchaVerify("Captcha is not valid")]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public ActionResult Register(RegisterModel model)

{

if (ModelState.IsValid)

{

// Attempt to register the user

try

{

var verifyToken = WebSecurity.CreateUserAndAccount(

model.UserName,

model.Password,

new { Email=model.Email },

true);

using (var context = new reddahEntities1())

{

var user = context.UserProfiles.FirstOrDefault(u => u.UserName == model.UserName);

EmailHelper.Send(

new MailAddress("donotreply@reddah.com", "donotreply@reddah.com"),

new MailAddress(model.Email, model.UserName),

"[reddah] Verify your email address‏",

string.Format("Dear {0}:\r\n" +

"visit this link to verify your email address:\r\n" +

"http://www.reddah.com/en-US/VerifyEmail?Userid={1}&EmailToken={2}" +

"\r\nthanks for using the site!",

model.UserName, user.UserId, verifyToken)

);

}

WebSecurity.Login(model.UserName, model.Password);

return RedirectToAction("HomePage", "Home");

}

catch (MembershipCreateUserException e)

{

ModelState.AddModelError("", ErrorCodeToString(e.StatusCode));

}

}

// If we got this far, something failed, redisplay form

return View(model);

}

[AllowAnonymous]

[HttpPost]

[AjaxValidateAntiForgeryToken]

public JsonResult JsonRegister(RegisterModel model, string returnUrl)

{

if (ModelState.IsValid)

{

try

{

var verifyToken = WebSecurity.CreateUserAndAccount(

model.UserName,

model.Password,

new { Email = model.Email },

false);

using (var context = new reddahEntities1())

{

var user = context.UserProfiles.FirstOrDefault(u => u.UserName == model.UserName);

EmailHelper.Send(

new MailAddress("donotreply@reddah.com", "donotreply@reddah.com"),

new MailAddress(model.Email, model.UserName),

"[reddah] Verify your email address‏",

string.Format("Dear {0}:\r\n" +

"visit this link to verify your email address:\r\n" +

"http://www.reddah.com/en-US/VerifyEmail?Userid={1}&EmailToken={2}" +

"\r\nthanks for using the site!",

model.UserName, user.UserId, verifyToken)

);

}

if (WebSecurity.Login(model.UserName, model.Password, persistCookie: model.RememberMe))

{

return Json(new { success = true, redirect = returnUrl });

}

else

{

ModelState.AddModelError("", Resources.Resources.Login\_Msg\_WrongPW);

}

}

catch (MembershipCreateUserException e)

{

ModelState.AddModelError("", ErrorCodeToString(e.StatusCode));

}

}

return Json(new { errors = GetErrorsFromModelState() });

}

//

// POST: /Account/Disassociate

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Disassociate(string provider, string providerUserId)

{

string ownerAccount = OAuthWebSecurity.GetUserName(provider, providerUserId);

ManageMessageId? message = null;

// Only disassociate the account if the currently logged in user is the owner

if (ownerAccount == User.Identity.Name)

{

// Use a transaction to prevent the user from deleting their last login credential

using (var scope = new TransactionScope(TransactionScopeOption.Required, new TransactionOptions { IsolationLevel = IsolationLevel.Serializable }))

{

bool hasLocalAccount = OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));

if (hasLocalAccount || OAuthWebSecurity.GetAccountsFromUserName(User.Identity.Name).Count > 1)

{

OAuthWebSecurity.DeleteAccount(provider, providerUserId);

scope.Complete();

message = ManageMessageId.RemoveLoginSuccess;

}

}

}

return RedirectToAction("Manage", new { Message = message });

}

//

// GET: /Account/Manage

[HttpGet]

public ActionResult Manage(ManageMessageId? message)

{

ViewBag.StatusMessage = StatusMessage(message);

ViewBag.HasLocalPassword = OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));

ViewBag.ReturnUrl = Url.Action("Manage");

return View();

}

private string StatusMessage(ManageMessageId? message)

{

return

message == ManageMessageId.ChangePasswordSuccess ? "Your password has been changed."

: message == ManageMessageId.SetPasswordSuccess ? "Your password has been set."

: message == ManageMessageId.RemoveLoginSuccess ? "The external login was removed."

: "";

}

}

}

namespace Reddah.Web.UI.Controllers

{

using System;

using System.Linq;

using System.Web.Mvc;

using Reddah.Web.UI.ViewModels;

using Reddah.Web.UI.Filters;

using Reddah.Web.UI.Models;

using System.Threading;

using System.Web;

using Reddah.Web.UI.Utility;

using System.IO;

using Newtonsoft.Json;

//using System.Net.Http;

//using System.Web.Http;

//using System.Web.Http.Cors;

public class ArticlesController : BaseController

{

private string getViewByDevice(){

return Request.Browser.IsMobileDevice ?

"~/Views/Articles/ArticleList.mobile.cshtml" : "~/Views/Articles/ArticleList.cshtml";

}

public ActionResult Menu(string path, string page)

{

if (!System.IO.File.Exists(HttpContext.Server.MapPath("~/App\_Data/" + path + ".xml")))

{

return new HttpNotFoundResult();

}

else

{

var PresentationTemplate = GetContentType(path);

var presentationView = getViewByDevice();

int pageNo = 0;

int.TryParse(page, out pageNo);

return View(presentationView, new MenuArticleViewModel(path, PresentationTemplate, pageNo));

}

}

//ionic api

[HttpGet]

[AllowCors("\*","\*")]

public JsonResult GetSub(string sub, int count = 25)

{

//string sub = "PICS";

//int count = 25;

if(String.IsNullOrEmpty(sub))

sub = new SubArticleViewModel().GetRandomSub();

return Json(new SubArticleViewModel(sub, count).Articles, JsonRequestBehavior.AllowGet);

}

/\*public System.Net.Http.HttpResponseMessage GetSub(string sub, int count = 25)

{

//string sub = "PICS";

//int count = 25;

if (String.IsNullOrEmpty(sub))

sub = new SubArticleViewModel().GetRandomSub();

///return Json(new SubArticleViewModel(sub, count).Articles, JsonRequestBehavior.AllowGet);

return new System.Net.Http.HttpResponseMessage()

{

Content = new System.Net.Http.StringContent(Json(new SubArticleViewModel(sub, count).Articles, JsonRequestBehavior.AllowGet).ToString())

};

}\*/

public ActionResult ClassicArticle(string path, string category, string count)

{

if (!System.IO.File.Exists(HttpContext.Server.MapPath("~/App\_Data/" + path + ".xml")))

{

return new HttpNotFoundResult();

// ooopos page

//\*\*\*\*\*////////throw new Exception();//yellow page in dev machine

//throw new HttpException(404, "Application not found");//white page

//return new HttpNotFoundResult();// transparent page

//log

}

else

{

var PresentationTemplate = GetContentType(path);// get all content data here

var presentationView = string.Format("~/Views/Articles/{0}.cshtml", PresentationTemplate);

if (PresentationTemplate == "FamilyArticle")

{

return View(presentationView, new FamilyArticleViewModel(path));

}

else if (PresentationTemplate == "NavigationList")

{

return View(presentationView, new NavigationListViewModel(path));

}

else if (PresentationTemplate == "WizardSteps")

{

return View(presentationView, new WizardStepsViewModel(path));

}

else if (PresentationTemplate == "Log")

{

return View(presentationView, new DatabaseLogViewModel(path));

}

// SolutionArticle

presentationView = Request.Browser.IsMobileDevice ?

"~/Views/Articles/SolutionArticle.mobile.cshtml" : "~/Views/Articles/SolutionArticle.cshtml";

return View(presentationView, new SolutionArticleViewModel(path));

}

// // Hide the header and footer

// ControllerContext.HttpContext.Items["HideShellHeaderAndFooter"] = true;

// var contentData = cosmosClient.TryGetSiteRegionalContent(path);

// var SGSupportedTemplate = new List<string> { "Support-Generic - FamilyPage",

// "Support-Generic - SolutionArticlePage"

// };

// var SGWizardTemplateList = new List<string> { "Support-Generic - WizardPage",

// "Support-Generic - WizardSteps"

// };

// if (contentData.ContentItem.PresentationTemplate == "Redirect")

// {

// var targetUrl = contentData.ContentItem["RedirectDestination"].Value;

// contentData = cosmosClient.TryGetSiteRegionalContent(Helpers.RemoveLocaleFromPath(targetUrl));

// }

// if (contentData != null && contentData.ContentItem != null)

// {

// var presentationView = contentData.ContentItem.PresentationTemplate;

// if (SGSupportedTemplate.Contains(presentationView))

// {

// if (browserCapability.IsMobile && !browserCapability.IsTablet)

// {

// presentationView = "~/Views/SmartGlass/" +

// contentData.ContentItem.PresentationTemplate +

// "\_Companion.mobile.cshtml";

// }

// else

// {

// presentationView = "~/Views/SmartGlass/" +

// contentData.ContentItem.PresentationTemplate +

// "\_Companion.cshtml";

// }

// return View(presentationView, contentData);

// }

// if (SGWizardTemplateList.Contains(presentationView))

// {

// return CompanionWizard(locale, path, currentStepPath, currentStepIndex, currentSolutionIndex);

// }

// }

// throw new HttpException(404, "Companion : Article page not found");

//}

}

public ActionResult SubReddah(string sub, int count = 25)

{

return View(getViewByDevice(), new SubArticleViewModel(sub, count));

}

public ActionResult RandomSubReddah()

{

var subName = new SubArticleViewModel().GetRandomSub();

return Redirect(string.Format("/{0}/r/{1}", Thread.CurrentThread.CurrentCulture, subName));

}

public ActionResult UserArticles(string userName, string page)

{

var presentationView = getViewByDevice();

int pageNo = 0;

int.TryParse(page, out pageNo);

return View(presentationView, new UserArticleViewModel(userName, pageNo));

}

public ActionResult Comments(string group, string id, string count)

{

int articleId = -1;

if (int.TryParse(id, out articleId))

{

var presentationView = Request.Browser.IsMobileDevice ?

"~/Views/Articles/ArticleComment.mobile.cshtml" : "~/Views/Articles/ArticleComment.cshtml";

return View(presentationView, new ArticleCommentViewModel(group, articleId, count));

}

else

{

throw new HttpException(404, "Reddah : Article not found");

}

}

public ActionResult Error(string path)

{

var presentationView = string.Format("~/Views/Error.cshtml");

return View(presentationView);

}

[Authorize]

[HttpPost]

public JsonResult JsonVote(VoteModel model)

{

var errors = string.Empty;

if (model.ArticleId < 0)

{

errors = "Article does not exist";

}

if (string.IsNullOrEmpty(errors))

{

try

{

//todo: also update user info

using (var db = new reddahEntities1())

{

var article = db.Articles.FirstOrDefault(a => a.Id == model.ArticleId);

if (model.Value.ToLowerInvariant() == "up")

{

if(article.Up == null)

{

article.Up = 0;

}

article.Up += 1;

}

else

{

if (article.Down == null)

{

article.Down = 0;

}

article.Down += 1;

}

db.SaveChanges();

}

}

catch(Exception e)

{

errors = e.Message;

}

}

if (!string.IsNullOrEmpty(errors))

{

return Json(new { errors = errors });

}

return Json(new { success = true });

}

[Authorize]

[HttpPost]

public JsonResult JsonAddComment(CommentModel model)

{

var errors = string.Empty;

if (model.ArticleId < 0)

{

errors = "Article does not exist";

}

if (string.IsNullOrEmpty(errors))

{

try

{

using (var db = new reddahEntities1())

{

db.Comments.Add(new Comment()

{

ArticleId = model.ArticleId,

ParentId = model.ParentId,

Content = System.Web.HttpUtility.HtmlEncode(Helpers.HideSensitiveWords(Helpers.HideXss(model.Content))),

CreatedOn = DateTime.Now,

UserName = User.Identity.Name

});

var article = db.Articles.FirstOrDefault(a => a.Id == model.ArticleId);

if (article != null)

{

article.Count ++;

}

if (model.ParentId != -1)

{

var parentComment = db.Comments.FirstOrDefault(c => c.Id == model.ParentId);

if (parentComment != null)

{

parentComment.Count ++;

}

}

db.SaveChanges();

}

}

catch (Exception e)

{

errors = e.Message;

}

}

if (!string.IsNullOrEmpty(errors))

{

return Json(new { errors = errors });

}

return Json(new { success = true });

}

[Authorize]

[HttpPost]

public JsonResult JsonDelComment(CommentModel model)

{

var errors = string.Empty;

try

{

using (var db = new reddahEntities1())

{

db.Comments.FirstOrDefault(x => x.Id == model.ArticleId).Status = -1;

db.SaveChanges();

}

}

catch (Exception e)

{

errors = e.Message;

}

if (!string.IsNullOrEmpty(errors))

{

return Json(new { errors = errors });

}

return Json(new { success = true });

}

[Authorize]

[HttpPost]

public ContentResult Upload(HttpPostedFileBase upload)

{

string guid = Guid.NewGuid().ToString().Replace("-","");

string uploadedImagePath = "/uploadPhoto/";

string uploadImageServerPath = "~" + uploadedImagePath;

if (!upload.ContentType.Contains("image"))

{

return Content("<script type=\"text/javascript\">window.parent.CKEDITOR.tools.callFunction(window.parent.CKEDITOR.instances.Content.\_.filebrowserFn,'',"

+ "'File format Error: (must be .jpg/.gif/.bmp/.png)');</script>");

}

else

{

var fileFormat = upload.FileName.Substring(upload.FileName.LastIndexOf('.')).Replace(".","");

var fileName = Path.GetFileName(guid + "." + fileFormat);

var filePhysicalPath = Server.MapPath(uploadImageServerPath + "/" + fileName);

if (!Directory.Exists(Server.MapPath(uploadImageServerPath)))

{

Directory.CreateDirectory(Server.MapPath(uploadImageServerPath));

}

upload.SaveAs(filePhysicalPath);

var url = uploadedImagePath + fileName;

using (var db = new reddahEntities1())

{

UploadFile file = new UploadFile();

file.Guid = guid;

file.Format = fileFormat;

file.UserName = User.Identity.Name;

file.CreatedOn = DateTime.Now;

file.GroupName = "";

file.Tag = "";

db.UploadFiles.Add(file);

db.SaveChanges();

}

return Content("<script>window.parent.CKEDITOR.tools.callFunction(window.parent.CKEDITOR.instances.Content.\_.filebrowserFn, \"" + url + "\");</script>");

}

}

}

}

using CaptchaMvc.Attributes;

using Reddah.Web.UI.Utility;

using System;

using System.Linq;

using System.Web.Mvc;

using System.Globalization;

namespace Reddah.Web.UI.Controllers

{

public class SubmitController : Controller

{

public ActionResult Index(string group)

{

Article article = new Article();

if(!String.IsNullOrEmpty(group))

article.GroupName = Helpers.HtmlEncode(group.Replace(" ", "").Replace("，", ","));

return View("~/Views/Submit/Index.cshtml", article);

}

[HttpPost, CaptchaVerify("Captcha is not valid")]

[Authorize]

[ValidateAntiForgeryToken]

[ValidateInput(false)]

public ActionResult Index(Article article)

{

if (ModelState.IsValid)

{

TempData["Message"] = "Message: captcha is valid.";

using (var context = new reddahEntities1())

{

string longArticleGroupNames = Helpers.HtmlEncode(article.GroupName.Replace(" ", "").Replace("，",","));

String[] articleGroupNames = longArticleGroupNames.Split(',');

foreach(string articleGroupName in articleGroupNames)

{

if (context.Groups.FirstOrDefault(g => g.Name == articleGroupName.Trim()) == null)

{

context.Groups.Add(new Group

{

Name = articleGroupName.Trim(),

CreatedOn = DateTime.Now

});

}

}

context.Articles.Add(new Article

{

Title = Helpers.HtmlEncode(article.Title),

GroupName = longArticleGroupNames,

Content = Helpers.HtmlEncode(article.Content),

Abstract = Helpers.HtmlEncode(article.Abstract),

UserName = User.Identity.Name,

CreatedOn = DateTime.Now,

Locale = CultureInfo.CurrentUICulture.Name.ToLowerInvariant(),

Type = 0 //0 article, 1 timeline

});

context.SaveChanges();

}

return RedirectToRoute("New");

}

TempData["ErrorMessage"] = "Error: captcha is not valid.";

return View();

}

public ActionResult Edit(int id)

{

ViewBag.Id = id;

Article article = null;

using (var context = new reddahEntities1())

{

article = context.Articles.FirstOrDefault(x => x.Id == id);

article.Title = Helpers.HtmlDecode(article.Title);

article.Abstract = Helpers.HtmlDecode(article.Abstract);

article.Content = Helpers.HtmlDecode(article.Content);

}

if (!(User.Identity.Name.Equals(article.UserName) || Helpers.Acl(User.Identity.Name, PrivilegeList.EditPost)))

{

ViewBag.Error = "you can't edit other people's post!";

}

return View("~/Views/Submit/edit.cshtml", article);

}

[HttpPost, CaptchaVerify("Captcha is not valid")]

[Authorize]

[ValidateAntiForgeryToken]

[ValidateInput(false)]

public ActionResult Edit(Article article)

{

//current user can do this

//bool valid = (Helpers.HtmlDecode(User.Identity.Name) == Helpers.HtmlDecode(article.UserName));

if (ModelState.IsValid)// && valid)

{

TempData["Message"] = "Message: captcha is valid.";

using (var context = new reddahEntities1())

{

var existingArticle = context.Articles.FirstOrDefault(x => x.Id == article.Id);

if (User.Identity.Name.Equals(existingArticle.UserName) || Helpers.Acl(User.Identity.Name, PrivilegeList.EditPost))

{

string longArticleGroupNames = Helpers.HtmlEncode(article.GroupName.Replace(" ", "").Replace("，", ","));

String[] articleGroupNames = longArticleGroupNames.Split(',');

foreach (string articleGroupName in articleGroupNames)

{

if (context.Groups.FirstOrDefault(g => g.Name == articleGroupName.Trim()) == null)

{

context.Groups.Add(new Group

{

Name = articleGroupName.Trim(),

CreatedOn = DateTime.Now

});

}

}

existingArticle.Title = Helpers.HtmlEncode(article.Title);

existingArticle.GroupName = longArticleGroupNames;

existingArticle.Content = Helpers.HtmlEncode(article.Content);

existingArticle.Abstract = Helpers.HtmlEncode(article.Abstract);

existingArticle.LastUpdateOn = DateTime.Now;

context.SaveChanges();

}

else

{

throw new Exception("you do not have permission to edit the post!");

}

}

//return RedirectToRoute(new { controller = "ArticleComment", action = "comments", id = article.Id });

//http://localhost/en-US/r/world/comments/130/test2ddd/

return Redirect(string.Format("/{0}/r/{1}/comments/{2}/{3}", CultureInfo.CurrentUICulture.Name, Helpers.GetUrlTitle(article.GroupName), article.Id, Helpers.GetUrlTitle(article.Title)));

}

TempData["ErrorMessage"] = "Error: captcha is not valid.";

return View();

}

[Authorize]

[ValidateInput(false)]

public ActionResult Delete(Article article)

{

if (ModelState.IsValid)

{

TempData["Message"] = "Message: captcha is valid.";

using (var context = new reddahEntities1())

{

var existingArticle = context.Articles.FirstOrDefault(x => x.Id == article.Id);

if (!User.Identity.Name.Equals(existingArticle.UserName))

{

throw new Exception("you can't delete other people's post!");

}

context.Articles.Remove(existingArticle);

context.SaveChanges();

}

return RedirectToRoute(new { controller = "ArticleComment", action = "comments", id = article.Id });

//http://localhost/en-US/r/world/comments/130/test2ddd/

}

TempData["ErrorMessage"] = "Error: captcha is not valid.";

return View();

}

}

}

Chat-fire.page.html

<ion-item color="light" no-padding lines="none">

<ion-icon name="ios-arrow-back" (click)="close()" class="backbutton"></ion-icon>

<ion-label>{{title}}</ion-label>

<ion-icon slot="icon-only" slot="end" name="ios-more" (click)="option()"></ion-icon>

</ion-item>

<ion-content [scrollEvents]="true" (ionScroll)="onScroll($event)" no-padding

style="background-color: #eeeeee;--background: #eeeeee;" #pageTop (click)="chatbox.hide()">

<ion-refresher slot="fixed" (ionRefresh)="getMoreHistory($event)">

<ion-refresher-content refreshingSpinner="bubbles">

</ion-refresher-content>

</ion-refresher>

<ion-grid \*ngFor="let comment of messages">

<ion-row \*ngIf="comment.Type==0||comment.Type==1000">

<ion-col no-padding>

<div text-center class="chatTime" \*ngIf="reddah.checkIsToday(comment.CreatedOn)">{{reddah.utcToLocal(comment.CreatedOn,'HH:mm')}}</div>

<div text-center class="chatTime" \*ngIf="!reddah.checkIsToday(comment.CreatedOn)">{{reddah.utcToLocal(comment.CreatedOn,'MM-dd HH:mm')}}</div>

</ion-col>

</ion-row>

<ion-row \*ngIf="comment.Type==1000">

<ion-col no-padding>

<div text-center class="chatTime">{{reddah.htmlDecode(comment.Content)}}</div>

</ion-col>

</ion-row>

<ion-row \*ngIf="comment.Type==0||comment.Type==4">

<ion-col no-padding>

<div class="container-right" \*ngIf="userName==comment.UserName">

<img [src]="reddah.appData('userphoto\_'+comment.UserName)"

class="portrait" (click)="goUser(comment.UserName)">

<div class="arrow-box-right" \*ngIf="comment.Type==0">

<div style="margin-right:-45px;" \*ngIf="comment.Id==null">\*</div>

<div class="chat-box-me" [innerHTML]="reddah.htmlDecode(comment.Content) | safe: 'html'"></div>

<div class="chat-arrow-right" style="flex:auto 0;"></div>

</div>

<div class="arrow-box-right" \*ngIf="comment.Type==4">

<div class="chat-box-me-link">

<app-share-article-chat [abstract]="comment.Abstract"

[image]="comment.Content" [id]="comment.Duration"></app-share-article-chat>

</div>

<div class="chat-arrow-right-link" style="flex:auto 0;"></div>

</div>

<!--

<div class="arrow-box-right" \*ngIf="comment.Type<2">

<div class="chat-box-me" \*ngIf="comment.Type==0" [innerHTML]="reddah.htmlDecode(comment.Content) | safe: 'html'"></div>

<div class="chat-box-me-1" \*ngIf="comment.Type==1" (click)="play(comment.Content)">

<ion-icon slot="start" name="ios-wifi" size="large" style="width:20px;height:18px;transform: rotate(-90deg);margin:auto;"></ion-icon>

<div style="display: flex; line-height:38px;">

<div style="width:5px;" \*ngFor="let i of reddah.getArray(comment.Duration)">&nbsp;</div>

<div>{{comment.Duration<0?0:comment.Duration}}''</div>

</div>

</div>

<div class="chat-arrow-right" style="flex:auto 0;"></div>

</div>

<div class="arrow-box-right-photo" \*ngIf="comment.Type==2">

<div class="parent3-me">

<img onerror="this.src='assets/icon/noimage.jpg'"

[src]="reddah.chatImageCache(comment.Content)"

class="picture3" (click)="viewer(0, [comment.Content])">

</div>

</div>

<div class="arrow-box-right-photo" \*ngIf="comment.Type==3">

<div class="parent3-me">

<img onerror="this.src='assets/icon/noimage.jpg'"

[src]="reddah.chatImageCache(comment.Content.toLowerCase().replace('.mp4','.jpg'))"

class="picture3"

(click)="playVideo(comment)" [class.notdownload]="!reddah.isLocal(comment.Content)">

<div style="position: absolute;bottom:0;margin:0.6em;"><ion-icon name="videocam" color="light"></ion-icon></div>

</div>

</div>

<div class="arrow-box-right" \*ngIf="comment.Type==4">

<div class="chat-box-me-link">

<app-share-article-chat [abstract]="comment.Abstract" [image]="comment.ImageUrl" [id]="comment.Duration"></app-share-article-chat>

</div>

<div class="chat-arrow-right-link" style="flex:auto 0;"></div>

</div>

-->

</div>

<div class="container-left" \*ngIf="userName!=comment.UserName">

<img [src]="reddah.appData('userphoto\_'+comment.UserName)"

class="portrait" (click)="goUser(comment.UserName)">

<div style="display: flex;margin-left:5px;">

<div class="chat-arrow-left" style="flex:0 auto;" \*ngIf="comment.Type==0"></div>

<div class="chat-box-others" \*ngIf="comment.Type==0" [innerHTML]="reddah.htmlDecode(comment.Content) | safe: 'html'"></div>

<div class="chat-arrow-left" style="flex:0 auto;" \*ngIf="comment.Type==4"></div>

<div class="chat-box-others-link" \*ngIf="comment.Type==4">

<app-share-article-chat [abstract]="comment.Abstract"

[image]="comment.Content" [id]="comment.Duration"></app-share-article-chat>

</div>

<!--

<div class="chat-box-others-1" \*ngIf="comment.Type==1" (click)="play(comment.Content)">

<ion-icon name="ios-wifi" size="large" style="width:20px;height:18px;transform: rotate(90deg);margin:auto;"></ion-icon>

<div style="display: flex; line-height:38px;">

<div>{{comment.Duration<0?0:comment.Duration}}''</div>

<div style="width:5px;" \*ngFor="let i of reddah.getArray(comment.Duration)">&nbsp;</div>

</div>

</div>

<div class="parent3" \*ngIf="comment.Type==2">

<img onerror="this.src='assets/icon/noimage.jpg'"

[src]="reddah.chatImageCache(comment.Content)"

class="picture3" (click)="viewer(0, [comment.Content])">

</div>

<div class="parent3" \*ngIf="comment.Type==3">

<img onerror="this.src='assets/icon/noimage.jpg'"

[src]="reddah.chatImageCache(comment.Content.toLowerCase().replace('.mp4','.jpg'))"

class="picture3"

(click)="playVideo(comment)" [class.notdownload]="!reddah.isLocal(comment.Content)">

<div style="position: absolute;bottom:0;margin:0.6em;"><ion-icon name="videocam" color="light"></ion-icon></div>

</div>

<div class="chat-arrow-left" style="flex:0 auto;" \*ngIf="comment.Type==4"></div>

<div class="chat-box-others-link" \*ngIf="comment.Type==4">

<app-share-article-chat abstract="{{comment.Abstract}}" image="{{comment.ImageUrl}}" id="{{comment.Duration}}"></app-share-article-chat>

</div>

-->

</div>

</div>

</ion-col>

</ion-row>

</ion-grid>

</ion-content>

<app-chat-fire-box #chatbox [(selectedArticleId)]="chatId" selectedCommentId="-1"

(reloadComments)="childReloadComments($event)" (localComments)="childLocalComments($event)"></app-chat-fire-box>

Locale.page.html

<ion-item color="light" no-padding lines="none">

<ion-icon name="ios-arrow-back" (click)="close()" class="backbutton"></ion-icon>

<ion-label \*ngIf="reddah.getCurrentLocale()!='en-US'">{{ 'Surface.Language' | translate }}(Language)</ion-label>

<ion-label \*ngIf="reddah.getCurrentLocale()=='en-US'">Language</ion-label>

</ion-item>

<ion-content no-padding>

<ng-template ngFor let-locale [ngForOf]="reddah.Locales">

<ion-item (click)="changeLocale(locale.Name)" \*ngIf="locale.Name!=selectedLocale">

<ion-icon slot="start" name="radio-button-off"></ion-icon>

<ion-label>{{locale.Description}}</ion-label>

</ion-item>

<ion-item (click)="changeLocale(locale.Name)" color="primary" \*ngIf="locale.Name==selectedLocale">

<ion-icon slot="start" name="radio-button-on"></ion-icon>

<ion-label>{{locale.Description}}</ion-label>

</ion-item>

</ng-template>

</ion-content>

Search.page.html

<ion-item color="light" no-padding lines="none">

<ion-icon name="ios-arrow-back" (click)="close()" class="backbutton"></ion-icon>

<ion-input placeholder="{{keywordPlaceholder}}" #searchKeyword clearInput (keyup.enter)="search()" (ionChange)="onSearchchange()"></ion-input>

</ion-item>

<ion-content no-padding style="background-color: white;" #searchResult>

<ion-grid \*ngIf="showTopic">

<ion-row style="padding:10px;">

<ion-col text-center>

<ion-note>{{ 'Search.Filter' | translate }}</ion-note>

</ion-col>

</ion-row>

<ion-row \*ngFor="let row of topics" style="padding:10px;">

<ion-col \*ngFor="let col of row" text-center>

<ion-text (click)="chooseTopic(col)">{{col.name}}</ion-text>

</ion-col>

</ion-row>

</ion-grid>

<ion-grid \*ngIf="selectedTopicId==1">

<ion-row>

<div \*ngIf="firstLoading\_a" style="display: flex; justify-content: center;width:100%;margin:15px;">

<ion-spinner name="bubbles"></ion-spinner>

<ion-text class="loadingText">{{ 'Button.Loading' | translate }}</ion-text>

</div>

<span class="articleTab" \*ngIf="articles\_a.length>0">{{ 'Menu.Recommend' | translate }}</span>

</ion-row>

<ion-row \*ngFor="let article of articles\_a" (click)="view(article)">

<div>

<span class="article-title articleTitle">{{reddah.htmlDecode(article.Title)}}</span>

<ion-item>

<ion-avatar class="avatar-top" \*ngIf="article.ImageUrls.length>0">

<img class="BrowsePreviewImageOne" src="{{reddah.parseImage(article.ImageUrls[0])}}">

</ion-avatar>

<ion-label>

<p class="articleContent" text-wrap [innerHTML]="reddah.summary(article.Content, 100, article.Locale)"></p>

<ion-note class="user-name">{{reddah.getDisplayName(article.UserName, 20)}}</ion-note>

<ion-note class="articleFooterTime">{{reddah.getSendTime(article.CreatedOnOrg| date: 'yyyy-MM-dd HH:mm:ss')}}</ion-note>

</ion-label>

</ion-item>

</div>

</ion-row>

</ion-grid>

<ion-grid \*ngIf="selectedTopicId==2">

<ion-row>

<div \*ngIf="firstLoading\_t" style="display: flex; justify-content: center;width:100%;margin:15px;">

<ion-spinner name="bubbles"></ion-spinner>

<ion-text class="loadingText">{{ 'Button.Loading' | translate }}</ion-text>

</div>

<span class="articleTab" style="margin-left:3px;" \*ngIf="articles\_t.length>0">{{ 'Menu.Timeline' | translate }}</span>

</ion-row>

<ion-row \*ngFor="let article of articles\_t" (click)="goTsViewer(article)">

<div style="width:100%;border-bottom: 1px solid #eeeeee;padding: 10px 0;">

<span class="ts-article-title" [innerHTML]="reddah.summary(article.Title, 100, article.Locale)"></span>

<ion-item style="margin: 8px 0;" lines="none" \*ngIf="article.Content.split('$$$').length>0&&article.Content.split('$$$')[0].length>0">

<ion-thumbnail \*ngFor="let image of article.Content.split('$$$').splice(0,3);index as i" style="padding: 3px;">

<img src="{{reddah.parseImage(image)}}">

</ion-thumbnail>

</ion-item>

<ion-label class="articleTitle">

<ion-note class="user-name" style="margin-left:3px;">{{reddah.getDisplayName(article.UserName, 20)}}</ion-note>

<ion-note class="articleFooterTime">{{reddah.getSendTime(article.CreatedOnOrg| date: 'yyyy-MM-dd HH:mm:ss')}}</ion-note>

</ion-label>

</div>

</ion-row>

</ion-grid>

<ion-grid \*ngIf="selectedTopicId==3">

<ion-row>

<div \*ngIf="firstLoading\_p" style="display: flex; justify-content: center;width:100%;margin:15px;">

<ion-spinner name="bubbles"></ion-spinner>

<ion-text class="loadingText">{{ 'Button.Loading' | translate }}</ion-text>

</div>

<span class="articleTab" \*ngIf="users\_p&&users\_p.length>0">{{ 'Menu.Publisher' | translate }}</span>

</ion-row>

<ion-row \*ngIf="searchKeyword.value==null||searchKeyword.value.length==0&&users\_p\_recent.length>0">

<ion-item lines="none">

<ion-label>{{ 'Search.Recently' | translate }}</ion-label>

</ion-item>

</ion-row>

<ion-row \*ngIf="searchKeyword.value==null||searchKeyword.value.length==0">

<ion-item lines="none">

<div style="display: flex;flex-wrap: wrap;">

<div style="display: flex;flex-direction: column;margin-top:15px;" \*ngFor="let pub\_recent of users\_p\_recent">

<ion-avatar class="userPhoto" (click)="goPub(pub\_recent)">

<img \*ngIf="pub\_recent.Photo" src="{{reddah.appData('userphoto\_'+pub\_recent.UserName, 'photo.svg')}}" class="recentPortrait">

<ion-icon \*ngIf="!pub\_recent.Photo" name="ios-list" color="tertiary" class="sub recentPortrait"></ion-icon>

</ion-avatar>

<ion-label style="margin:3px 0 0 0;padding:3px 5px;width:75px;color:gray;" text-center>{{pub\_recent.NickName}}</ion-label>

</div>

</div>

</ion-item>

</ion-row>

<ion-row \*ngFor="let pub of users\_p" (click)="goPub(pub)">

<ion-item>

<ion-avatar slot="start" class="contactUserPhoto">

<img src="{{reddah.appData('userphoto\_'+pub.UserName, 'photo.svg')}}" class="contactPortrait">

</ion-avatar>

<ion-label style="margin-left:10px;">

<div class="user-name" (press)="showChangeMenu($event, contact)">{{pub.NickName}}</div>

<ion-note color="dark">{{pub.Signature}}</ion-note>

</ion-label>

</ion-item>

</ion-row>

</ion-grid>

<ion-grid \*ngIf="selectedTopicId==4">

<ion-row>

<div \*ngIf="firstLoading\_m" style="display: flex; justify-content: center;width:100%;margin:15px;">

<ion-spinner name="bubbles"></ion-spinner>

<ion-text class="loadingText">{{ 'Button.Loading' | translate }}</ion-text>

</div><span class="articleTab" \*ngIf="users\_m&&users\_m.length>0">{{ 'Menu.MiniApp' | translate }}</span>

</ion-row>

<ion-row \*ngIf="searchKeyword.value==null||searchKeyword.value.length==0&&users\_m\_recent.length>0">

<ion-item lines="none">

<ion-label>{{ 'Search.Recently' | translate }}</ion-label>

</ion-item>

</ion-row>

<ion-row \*ngIf="searchKeyword.value==null||searchKeyword.value.length==0">

<ion-item lines="none">

<div style="display: flex;flex-wrap: wrap;">

<div style="display: flex;flex-direction: column;margin-top:15px;" \*ngFor="let pub\_recent of users\_m\_recent">

<ion-avatar class="userPhoto" (click)="goMini(pub\_recent)">

<img \*ngIf="pub\_recent.Photo" src="{{reddah.appData('userphoto\_'+pub\_recent.UserName, 'photo.svg')}}" class="recentPortrait">

<ion-icon \*ngIf="!pub\_recent.Photo" name="infinite" color="tertiary" class="mini recentPortrait"></ion-icon>

</ion-avatar>

<ion-label style="margin:0;padding:3px 5px;width:75px;color:gray;" text-center>{{pub\_recent.NickName}}</ion-label>

</div>

</div>

</ion-item>

</ion-row>

<ion-row \*ngFor="let pub of users\_m" (click)="goMini(pub)">

<ion-item>

<ion-avatar slot="start" class="contactUserPhoto">

<img src="{{reddah.appData('userphoto\_'+pub.UserName, 'photo.svg')}}" class="contactPortrait">

</ion-avatar>

<ion-label style="margin-left:10px;">

<div class="user-name" (press)="showChangeMenu($event, contact)">{{pub.NickName}}</div>

<ion-note color="dark">{{pub.Signature}}</ion-note>

</ion-label>

</ion-item>

</ion-row>

</ion-grid>

<ion-grid \*ngIf="selectedTopicId==0">

<ion-row>

<div \*ngIf="firstLoading\_a" style="display: flex; justify-content: center;width:100%;margin:15px;">

<ion-spinner name="bubbles"></ion-spinner>

<ion-text class="loadingText">{{ 'Button.Loading' | translate }}</ion-text>

</div>

<span class="articleTab" \*ngIf="articles\_a.length>0">{{ 'Menu.Recommend' | translate }}</span>

</ion-row>

<ion-row \*ngFor="let article\_a of articles\_a" (click)="view(article\_a)">

<div>

<span class="article-title articleTitle">{{reddah.htmlDecode(article\_a.Title)}}</span>

<ion-item>

<ion-avatar class="avatar-top" \*ngIf="article\_a.ImageUrls.length>0">

<img class="BrowsePreviewImageOne" src="{{reddah.parseImage(article\_a.ImageUrls[0])}}">

</ion-avatar>

<ion-label>

<p class="articleContent" text-wrap [innerHTML]="reddah.summary(article\_a.Content, 100, article\_a.Locale)"></p>

<ion-note class="user-name">{{reddah.getDisplayName(article\_a.UserName, 20)}}</ion-note>

<ion-note class="articleFooterTime">{{reddah.getSendTime(article\_a.CreatedOnOrg| date: 'yyyy-MM-dd HH:mm:ss')}}</ion-note>

</ion-label>

</ion-item>

</div>

</ion-row>

<ion-row>

<span class="articleTab" \*ngIf="articles\_a.length>0" (click)="more(0)">{{ 'Search.More' | translate }}>></span>

</ion-row>

<ion-row>

<div \*ngIf="firstLoading\_t" style="display: flex; justify-content: center;width:100%;margin:15px;">

<ion-spinner name="bubbles"></ion-spinner>

<ion-text class="loadingText">{{ 'Button.Loading' | translate }}</ion-text>

</div>

<span class="articleTab" \*ngIf="articles\_t.length>0">{{ 'Menu.Timeline' | translate }}</span>

</ion-row>

<ion-row \*ngFor="let article\_t of articles\_t" (click)="goTsViewer(article\_t)">

<div style="width:100%;border-bottom: 1px solid #eeeeee;padding: 10px 0;">

<span class="ts-article-title" [innerHTML]="reddah.summary(article\_t.Title, 100, article\_t.Locale)"></span>

<ion-item style="margin: 8px 0;" lines="none" \*ngIf="article\_t.Content.split('$$$').length>0&&article\_t.Content.split('$$$')[0].length>0">

<ion-thumbnail \*ngFor="let image of article\_t.Content.split('$$$').splice(0,3);index as i" style="padding: 3px;">

<img src="{{reddah.parseImage(image)}}">

</ion-thumbnail>

</ion-item>

<ion-label class="articleTitle">

<ion-note class="user-name">{{reddah.getDisplayName(article\_t.UserName, 20)}}</ion-note>

<ion-note class="articleFooterTime">{{reddah.getSendTime(article\_t.CreatedOnOrg| date: 'yyyy-MM-dd HH:mm:ss')}}</ion-note>

</ion-label>

</div>

</ion-row>

<ion-row>

<span class="articleTab" \*ngIf="articles\_t.length>0" (click)="more(1)">{{ 'Search.More' | translate }}>></span>

</ion-row>

</ion-grid>

<ion-infinite-scroll threshold="100px" (ionInfinite)="loadData($event)" \*ngIf="selectedTopicId!=0">

<ion-infinite-scroll-content loadingSpinner="bubbles" loadingText="{{ 'Button.Loading' | translate }}">

</ion-infinite-scroll-content>

</ion-infinite-scroll>

</ion-content>

Share-article.component.html

<div (click)="viewArticle()" class="container">

<ion-item lines="none" color="light" no-padding style="padding-bottom:5px;">

<ion-avatar class="avatar-top sharePhoto" \*ngIf="image">

<img [src]="reddah.parseImage(image)" class="sharePortrait">

</ion-avatar>

<ion-avatar class="avatar-top sharePhotoNone" \*ngIf="!image">

<ion-icon name="ios-link" color="dark" class="sharePortraitNone"></ion-icon>

</ion-avatar>

<ion-label class="marginleft" text-wrap class="label-style">

<span>{{reddah.htmlDecode(abstract)}}</span><br>

<ion-note color="dark"></ion-note>

</ion-label>

</ion-item>

</div>

Mytimeline.page.html

{

"Surface":{

"Language": "Language",

"Login": "Login",

"Register": "Register"

},

"Login":{

"Title":"Login",

"NoAccount":"No Account?",

"RegisterNow":"Register Now",

"ForgotPassword":"Forgot Password",

"Loading":"Login..."

},

"Register":{

"Title":"Register",

"TitleSub":"Register publisher",

"TitleMini":"Register mini app",

"ReadAgree":"You have read and agreed",

"UserAgreement":"User Agreement,",

"Privacy":"Privacy Policy",

"And":"&",

"Content":"Content Policy",

"Loading":"Register...",

"Success":"Registration is successful. Please verify your email and login",

"Manage":"Manage"

},

"Input":{

"UserName": "User Name",

"Password": "Password",

"OldPassword": "Old Password",

"NewPassword": "New Password",

"ConfirmPassword": "Confirm Password",

"Email": "Email",

"Name":"Name",

"Description":"Description",

"Error":{

"UserNameEmpty":"Please input user name",

"UserNameTooLong":"User name should be less than 25 characters",

"SubNameTooShort":"Name should be at least 2 characters",

"PasswordEmpty":"Please input password",

"PasswordDifferent":"Password and confirm password are different",

"EmailEmpty":"Please input email",

"Agree":"Please agree with user agreement",

"SubNameEmpty":"Please input name",

"SubDescEmpty":"Please input description",

"TokenEmpty":"Please input security token",

"ResetResult":"Reset password successfully, please login",

"GetTokenFromEmail":"Please get security token from your Email",

"NetworkError":"No network connection",

"ServiceError":"The system service is being updated. Please try again later"

}

},

"Button":{

"Login": "Login",

"Register": "Register",

"Logout": "Logout",

"Back": "Back",

"Close": "Close",

"Loading":"loading...",

"Reset":"Forgot Password",

"Token":"Security Token",

"ChangePassword":"Change Password"

},

"Menu":{

"Message": "Message",

"Recommend": "Article",

"Earth":"Earth",

"Contact": "Friend",

"Find": "Explore",

"About": "Me",

"Publisher": "Publisher",

"MiniApp": "Mini App",

"NewFriends": "New Friend",

"GroupChat": "Group Chat",

"Mark": "Mark",

"Settings": "Settings",

"Header":{

"GroupChat":"New Chat",

"AddFriend":"Add Friend",

"Scan":"Scan",

"Feedback":"Feedback"

},

"Timeline":"Aperture",

"Shake":"Shake",

"Platform":"Open Platform"

},

"Search":{

"Title":"Search",

"Filter":"Filter by",

"Article":"Article",

"Aperture":"Aperture",

"Publisher":"Publisher",

"MiniApp":"Mini App",

"Recently":"Recently used",

"More":"More"

},

"About":{

"Account":"Account",

"Privacy":"Privacy",

"General":"General",

"About":"About",

"Photo":"Photo",

"Card":"Card",

"Location":"Location",

"Nickname":"Nickname",

"Sex":"Sex",

"Male":"Male",

"Female":"Female",

"Signature":"Signature",

"CurrentVersion": "Version",

"CheckUpgrade": "Check Latest Version",

"CurrentLocale": "Locale",

"ChangeLocale": "Change Locale",

"CurrentUser": "Current User"

},

"Time.YearsAgo": "year(s) ago",

"Time.MonthsAgo": "month(s) ago",

"Time.WeeksAgo": "week(s) ago",

"Time.Today": "Today",

"Time.Yesterday": "Yesterday",

"Time.DaysAgo": "days ago",

"Time.HoursAgo": "hour(s) ago",

"Time.MinutesAgo": "mins ago",

"Time.JustNow": "Just now",

"Article.Loading": "Please wait",

"Comment.Comment": "Comment",

"Comment.MoreComment":"More Comments",

"Comment.Like": "Like",

"Comment.Delete": "Delete",

"Comment.Tip": "Enter some comment here...",

"Confirm.Title": "Confirm",

"Confirm.Message": "Do you want to exit Reddah?",

"Confirm.ClearChatMessage": "Do you want to clear all the chat messages?",

"Confirm.LogoutMessage":"Do you want to logout?",

"Confirm.LeaveGroupChatMessage":"Do you want to quit and delete?",

"Confirm.DeleteMessage":"Do you want to delete?",

"Confirm.Yes": "Yes",

"Confirm.Cancel": "Cancel",

"Confirm.Submit": "Submit",

"Confirm.ByLatest":"Latest",

"Confirm.ByEarly":"Earliest",

"Confirm.ByLike":"Like",

"Pop":{

"NoRecommend":"Hide",

"Report":"Report",

"ReportReason":"Please input a concrete reason...",

"Nointerest":"Not Interested",

"ToFriend":"Share with friend",

"ToAperture":"Share to Aperture",

"ChangeCover":"Change cover",

"Note":"Note",

"TakePhoto":"Camera",

"Photo":"Take a photo",

"SelectPhoto":"photo from album",

"SelectVideo":"video from album",

"SawIt":"Seen",

"Trash":"Trash",

"Porn":"Porn",

"NoAuthor":"Anti",

"Dislike":"Dislike",

"Suggest":"Suggest",

"ContentWrong":"Content Wrong",

"CodeError":"Program Error",

"ContentInvalid":"Content Invalid",

"ContentOther":"Others",

"Marked":"Marked",

"HideLocation":"Hide Location",

"YourThoughts":"Your thoughts now",

"ReleaseToDel":"Release to delete",

"DragToDel":"Drag to delete",

"FeedbackThoughts":"Write down your issues and suggestions, we'll keep updating",

"Focus":"Focus",

"Unfocus":"Unfocus",

"EnterPub":"Enter",

"StillFocus":"Stay Focus",

"StillUnfocus":"Do not focus",

"UnfocusMessage":"You won't receive its messages any more",

"KM":"KM",

"M":"M",

"Year":"",

"Month":"",

"MsgCount":"Message",

"GroupChatTitle":"Group Name",

"GroupChatAnnouce":"Group Notice",

"GroupChatUpdate":"I created a group chat, let's talk!",

"AddToGrp":"invited to the group chat",

"KickGrp":"removed from the group chat",

"PressSpeak":"Hold to Talk",

"ReleaseSend":"Release to Send",

"FunImageLib":"Album",

"FunCamera":"Camera",

"FunVideoLib":"VideoLib",

"FunVideo":"Video"

},

"Common":{

"Distance":"Distance",

"Refresh":"Refresh",

"Choose":"Choose",

"Font":"Font Size",

"Font1":"Preview font size",

"Font2":"Adjust the slider below to change font size",

"Font3":"Font size in chat, article and aperture will be changed. \"Loveis something eternal; the aspect may change, but not theessence.\"",

"Cache":"Clear Cache",

"Save":"Save",

"FeatureNotOpen":"Features not open yet: Edit, Publish, etc.",

"Draft":"Draft",

"Test":"Test",

"Publish":"Publish",

"View":"View",

"Category":"Account Category",

"PubDesc":"It provides a new way of information dissemination for the media and individuals, and builds a better communication and management model with readers.",

"MiniDesc":"A new open capability can be easily accessed and disseminated in Reddah, and it has excellent experience in use.",

"HideMyLocation":"Do not display my location",

"AllowTenTimeline":"Allow stranger to view 10 Aperture",

"Developer":"Developer",

"ServicePhone":"Service Phone",

"Owner":"Owner",

"OwnerDesc":"Subject certification shall submit certificates certifying the legitimacy of institutions issued by relevant competent authorities, such as business licenses of enterprises, certificates of registration of legal persons of institutions, certificates of registration of legal persons of organs, certificates of registration of legal persons of social organizations, certificates of registration of private non-enterprise units, etc.",

"CacheClear":"Cache is clear",

"NewFriend":"New Friends",

"Accept":"Accept",

"Added":"Added",

"Accepted":"Accepted",

"Option":"Options",

"DeleteAndQuit":"Delete and quit",

"ClearChat":"Clear chat",

"Clear":"Clear",

"AllowTenMessage":"Strangers can only view 10 Aperture messages",

"NoPermissionMessage":"Strangers can't view Aperture messages",

"Details":"Details",

"ScanToAdd":"Scan the QR code to add friend",

"Original":"Original",

"ApplyFriend":"You need to send a application and wait for approval",

"ApplyFriendMe":"I am",

"Label":"Label",

"MaxDraft":"Ordinary users can save up to 3 drafts",

"WorkingCopyExist":"There's an unpublished version",

"MyPhoto":"My Material",

"AddPhoto":"Add Material",

"Copy":"Copy Image Address",

"UserNotExist":"User does not exist"

},

"Mail":{

"Register":{

"Title":"Reddah Account",

"Sub":"Verify your email address‏",

"ParaStart":"Please visit this link to verify your email address",

"ParaEnd":"After that, you can login. Thanks for using Reddah!"

},

"Reset":{

"Title":"Reddah Account",

"Sub":"Security token to reset password",

"ParaStart":"Please copy this token to reset your password",

"ParaEnd":"Thanks for using Reddah!"

}

},

"Service":{

"4": "Service Error, please try again later",

"1001":"User name must contain 6-18 letters and numbers. No special characters allowed",

"1002":"Email is invalid",

"1003":"User name exists",

"1004":"Email is registered",

"1005":"Please go to your mailbox to verify your email",

"1006":"User name or password is wrong",

"1007":"Email is not registered",

"1008":"Reset password failed, please check if security token is correct",

"1009":"Old password is wrong",

"1010":"Please select a material",

"1011":"No permission to delete article",

"1012":"No permission to delete comment"

}

}

Home.page.html

<app-header title="Menu.Recommend"></app-header>

<ion-content #pageTop style="background-color: white;">

<ion-refresher slot="fixed" (ionRefresh)="dragDown($event)">

<ion-refresher-content refreshingSpinner="bubbles">

</ion-refresher-content>

</ion-refresher>

<ion-spinner name="bubbles" \*ngIf="firstLoad" style="margin:15px 50%;transform: translateX(-50%);"></ion-spinner>

<ng-template ngFor let-article [ngForOf]="articles" id="homepage" style="color: #636468;font-family: 'Segoe UI', Arial, sans-serif;">

<div class="postModuleLiner" \*ngIf="article.VideoUrl.length>0">

<div>

<span [class.articleTextRead]="article.Read" class="articleTitle" (click)="view(article)">{{reddah.htmlDecode(article.Title)}}</span>

<div class="postClear"></div>

<video id="video\_{{article.Id}}" [poster]="" [src]="article.VideoUrl" controls>

your browser does not support playing video

</video>

<div class="postClear"></div>

</div>

<div class="listInfo" style="color:#636468;display:flex;justify-content: space-between;">

<div style="display: flex;" (click)="goUser(article.UserName)">

<span class="user-name" style="margin-right:10px;display: inline-block;vertical-align: middle;line-height: 1.5em;">

{{reddah.getDisplayName(article.UserName)}}

</span>

<ion-icon color="medium" name="text"></ion-icon><span style="padding-left:3px;display: inline-block;vertical-align: middle;line-height: 1.5em;">{{article.Comments}}</span>

</div>

<div style="display: flex;" (click)="dislike($event, article)">

<ion-icon color="medium" size="small" name="close"></ion-icon>

</div>

</div>

</div>

<div class="postModuleLiner" \*ngIf="article.VideoUrl.length==0">

<div \*ngIf="article.ImageUrls.length>=3" (click)="view(article)">

<span [class.articleTextRead]="article.Read" class="articleTitle">{{reddah.htmlDecode(article.Title)}}</span>

<div class="postClear"></div>

<ion-card-content no-padding>

<div class="parent">

<div class="container" \*ngFor="let articleUrl of article.ImageUrls;index as i">

<img [class.left-radius]="i==0" [class.right-radius]="i==2"

[id]="reddah.makeItId(articleUrl+'home')"

(load)="reddah.adjustImage(articleUrl, 'home')"

onerror="this.src='/assets/icon/noimage.jpg'"

[src]="reddah.level2Cache(articleUrl)"

class="picture">

</div>

</div>

</ion-card-content>

</div>

<div \*ngIf="article.ImageUrls.length>=3" class="listInfo" style="color:#636468;display:flex;justify-content: space-between;">

<div style="display: flex;" (click)="goUser(article.UserName)">

<span class="user-name" style="margin-right:10px;display: inline-block;vertical-align: middle;line-height: 1.5em;">

{{reddah.getDisplayName(article.UserName)}}

</span>

<ion-icon color="medium" name="text"></ion-icon><span style="padding-left:3px;display: inline-block;vertical-align: middle;line-height: 1.5em;">{{article.Comments}}</span>

</div>

<div style="display: flex;" (click)="dislike($event, article)">

<ion-icon color="medium" size="small" name="close"></ion-icon>

</div>

</div>

<div \*ngIf="article.ImageUrls.length>0&&article.ImageUrls.length<3" (click)="view(article)">

<div class="listImage">

<div class="listImageWrapper">

<img class="BrowsePreviewImageOne"

[id]="reddah.makeItId(article.ImageUrls[0]+'home')"

(load)="reddah.adjustImage(article.ImageUrls[0], 'home')"

onerror="this.src='/assets/icon/noimage.jpg'"

[src]="reddah.level2Cache(article.ImageUrls[0])">

</div>

</div>

<span [class.articleTextRead]="article.Read" class="articleTitle">{{reddah.htmlDecode(article.Title)}}</span><br>

<div [class.articleTextRead]="article.Read">{{reddah.htmlDecode(reddah.summary(article.Content, 100))}}</div>

</div>

<div \*ngIf="article.ImageUrls.length>0&&article.ImageUrls.length<3" class="listInfo" style="color:#636468;display:flex;justify-content: space-between;">

<div style="display: flex;" (click)="goUser(article.UserName)">

<span class="user-name" style="margin-right:10px;display: inline-block;vertical-align: middle;line-height: 1.5em;">

{{reddah.getDisplayName(article.UserName)}}

</span>

<ion-icon color="medium" name="text"></ion-icon><span style="padding-left:3px;display: inline-block;vertical-align: middle;line-height: 1.5em;">{{article.Comments}}</span>

</div>

<div style="display: flex;" (click)="dislike($event, article)">

<ion-icon color="medium" size="small" name="close"></ion-icon>

</div>

</div>

<div \*ngIf="article.ImageUrls.length==0" (click)="view(article)">

<span [class.articleTextRead]="article.Read" class="articleTitle">{{reddah.htmlDecode(article.Title)}}</span><br>

<div [class.articleTextRead]="article.Read">{{reddah.htmlDecode(reddah.summary(article.Content, 100))}}</div>

</div>

<div class="listInfo" \*ngIf="article.ImageUrls.length==0">

<span \*ngFor="let group of article.GroupName.split(',')" (click)="goSearch(group)" style="padding-right:5px;">

<u style="border-bottom: 1px solid #ff3030;text-decoration: none;color:#107c10;margin-bottom:2px;">{{group.trim()}}</u>

</span>

</div>

<div \*ngIf="article.ImageUrls.length==0" class="listInfo" style="color:#636468;display:flex;justify-content: space-between;">

<div style="display: flex;">

<span (click)="goUser(article.UserName)" class="user-name" style="margin-right:10px;display: inline-block;vertical-align: middle;line-height: 1.5em;">

{{reddah.getDisplayName(article.UserName)}}

</span>

<ion-icon (click)="view(article)" color="medium" name="text"></ion-icon>

<span (click)="view(article)" style="padding-left:3px;display: inline-block;vertical-align: middle;line-height: 1.5em;">{{article.Comments}}</span>

</div>

<div style="display: flex;" (click)="dislike($event, article)">

<ion-icon color="medium" size="small" name="close"></ion-icon>

</div>

</div>

</div>

</ng-template>

<ion-infinite-scroll threshold="100px" (ionInfinite)="loadData($event)">

<ion-infinite-scroll-content loadingSpinner="bubbles" loadingText="{{ 'Button.Loading' | translate }}">

</ion-infinite-scroll-content>

</ion-infinite-scroll>

</ion-content>

@using Reddah.Web.UI.Models

@using Reddah.Web.UI.Resources

@using Reddah.Web.UI.Utility

@using System.Globalization

@{

var locale = CultureInfo.CurrentUICulture.Name;

int index = (GetNextPageNo() - 1) \* 25 + 1;

int nextPrevIndex = 1;

}

@functions{

public bool ShowPrev()

{

return GetNextPageNo() > 1;

}

public int GetPrevPageNo()

{

var pageNo = Request.Params["page"];

int newPageNo = 0;

if (string.IsNullOrWhiteSpace(pageNo))

{

newPageNo = 0;

}

else

{

if (int.TryParse(pageNo, out newPageNo))

{

newPageNo -= 1;

}

}

return newPageNo;

}

public int GetNextPageNo()

{

var pageNo = Request.Params["page"];

int newPageNo = 1;

if (string.IsNullOrWhiteSpace(pageNo))

{

newPageNo = 1;

}

else

{

if (int.TryParse(pageNo, out newPageNo))

{

newPageNo += 1;

}

}

return newPageNo;

}

}

<link href="/Content/Post.css" rel="stylesheet" />

@\*<h1 class="HomePageModular">@Model.ArticleTitle</h1>\*@

<div class="rightBox">

@\*Html.Partial("~/Views/Shared/Controls/LoginRegister/JsonLogin.cshtml", new Reddah.Web.UI.Models.LoginModel())\*@

@if (Model.RightBoxModules != null)

{

int j = 0;

foreach (var rightBoxModule in Model.RightBoxModules)

{

if (!String.IsNullOrEmpty(rightBoxModule.Title) && !String.IsNullOrEmpty(rightBoxModule.ArticleUrl))

{

if (j == 3)

{

break;

}

<div class="HomePageRightBoxModule">

<table>

<tr>

<td>

<a href="/@locale/@rightBoxModule.ArticleUrl"><img alt="@rightBoxModule.ImageUrl" src="@rightBoxModule.ImageUrl" /></a>

</td>

</tr>

<tr>

<td class="rightBoxModule">

<h2><a href="/@locale/@rightBoxModule.ArticleUrl">@rightBoxModule.Title</a></h2>

<div>@rightBoxModule.Description</div>

</td>

</tr>

</table>

</div>

j++;

}

}

}

</div>

<div class="postBlock" id="postBlock">

@if (Model.NextPrevBox != null)

{

<div class="upcomingLinks" ng-controller="nextprevbox" id="upcomingLinks">

@foreach (var article in Model.NextPrevBox)

{

if (!String.IsNullOrEmpty(article.Title) && !String.IsNullOrEmpty(article.Abstract) && !String.IsNullOrEmpty(article.ArticleUrl))

{

<div class="upcomingPost" id="upcomingPost-@nextPrevIndex" ng-init="showPost()">

<span class="voterank"></span>

<div class="votecell">

<div class="vote">

@if (Request.IsAuthenticated)

{

<a class="vote-up-off" ng-click="vote(@article.Id, 'up', @(article.Up-article.Down))">up vote</a>

<span itemprop="upvoteCount" class="vote-count-post" id="vote-count-post-@article.Id">-</span>

<a class="vote-down-off" ng-click="vote(@article.Id, 'down', @(article.Up-article.Down))">down vote</a>}

else

{

<a class="vote-up-off" ng-click="showPanel()">up vote</a>

<span itemprop="upvoteCount" class="vote-count-post">-</span>

<a class="vote-down-off" ng-click="showPanel()">down vote</a>

}

</div>

</div>

<div class="listInfoBlock">

@if (!String.IsNullOrEmpty(article.ImageUrl))

{

<div class="listImage">

<div class="listImageWrapper">

<img ng-click="showImage('upcomingLinks', @nextPrevIndex)" class="BrowsePreviewImage" onerror="this.style.display='none'" width="193" src="@article.ImageUrl" alt="picture">

</div>

</div>

}

<h2><a ng-click="toggleArticle('upcomingLinks', @nextPrevIndex)">@Helpers.HtmlDecode(article.Title)</a> <span class="listInfo" style="display:none;">(source)</span></h2>

<div>@Helpers.HtmlDecode(article.Abstract)</div>

<div class="listInfo">submitted @article.CreatedOn ago by <a href="/@locale/user/@article.UserName">@article.UserName</a> to <a href="/@locale/r/@Helpers.HtmlDecode(article.GroupName)"><u>@Helpers.HtmlDecode(article.GroupName)</u></a></div>

<div class="listInfo"><a href="/@locale/r/@article.GroupName/comments/@article.Id/@Helpers.GetUrlTitle(Helpers.HtmlDecode(article.Title))/" target="\_blank">@article.Comments @Resources.Article\_Comments</a> <!--share save hide report--></div>

<div class="postClear"></div>

<img id="org\_image\_@nextPrevIndex" class="orgImage" onerror="if (this.src != '/content/images/noimage.jpg') this.src = '/content/images/noimage.jpg';" width="100%" src="@article.ImageUrl" alt="picture">

<div id="org\_article\_@nextPrevIndex" class="orgArticle">@Html.Raw(Helpers.HtmlDecode(article.Content))</div>

</div>

</div>

}

nextPrevIndex++;

}

<div class="nextprev">

<img class="buttonPrevNext" src="/content/images/arrow-left.png" ng-click="prev(@Model.NextPrevBox.Count)" />

<img class="buttonPrevNext" src="/content/images/arrow-right.png" ng-click="next(@Model.NextPrevBox.Count)" />

</div>

<div class="nextprevHelp" title="@Resources.Article\_WhatThisTip">@Resources.Article\_WhatThis</div>

</div>

}

@if (Model.TrendingSubs != null)

{

<div class="trendingSubs">

@Resources.Article\_TrendingSubReddah

@foreach (var trendingSub in Model.TrendingSubs)

{

<a href="/@locale/r/@trendingSub"><u>@trendingSub</u></a>

}

</div>

}

<div class="postClear"></div>

@foreach (var article in Model.Articles)

{

if (!String.IsNullOrEmpty(article.Title) && !String.IsNullOrEmpty(article.Abstract) && !String.IsNullOrEmpty(article.ArticleUrl))

{

<div class="post">

<span class="voterank">@index</span>

<div class="votecell">

<div class="vote">

@if (Request.IsAuthenticated)

{

<a id="vote-up-@article.Id" class="vote-up-off" ng-click="vote(@article.Id, 'up', @(article.Up-article.Down),'post', @DateTime.Now.Second)">up vote</a>

<span itemprop="upvoteCount" class="vote-count-post" id="vote-count-post-@article.Id">@(article.Up - article.Down)</span>

<a id="vote-down-@article.Id" class="vote-down-off" ng-click="vote(@article.Id, 'down', @(article.Up-article.Down),'post', @DateTime.Now.Second)">down vote</a>}

else

{

<a class="vote-up-off" ng-click="showPanel()">up vote</a>

<span itemprop="upvoteCount" class="vote-count-post">@(article.Up - article.Down)</span>

<a class="vote-down-off" ng-click="showPanel()">down vote</a>

}

</div>

</div>

<div class="postModuleLiner">

@if (article.ImageUrls != null)

{

if (article.ImageUrls.Length > 1)

{

<h2><a ng-click="toggleArticle('post', @index)">@Helpers.HtmlDecode(article.Title)</a> <span class="listInfo" style="display:none;">(source)</span></h2>

<div class="postClear"></div>

}

for (int i = 0; i < article.ImageUrls.Length; i++)

{

<div class="listImage">

<div class="listImageWrapper">

<img ng-click="showImage('post', @index)" class="BrowsePreviewImage" onerror="this.style.display='none'" width="193" src="@article.ImageUrls[i]" alt="picture">

</div>

</div>

}

}

@if (article.ImageUrls != null && article.ImageUrls.Length < 2)

{

<h2><a ng-click="toggleArticle('post', @index)">@Helpers.HtmlDecode(article.Title)</a> <span class="listInfo" style="display:none;">(source)</span></h2>

}

<div>@Helpers.HtmlDecode(article.Abstract)</div>

<div class="listInfo">

<a href="/@locale/r/@Helpers.HtmlDecode(article.GroupName)"><u>@Helpers.HtmlDecode(article.GroupName)</u></a>

submitted @article.CreatedOn ago by <a href="/@locale/user/@article.UserName">@article.UserName</a>

</div>

<div class="listInfo"><a href="/@locale/r/@article.GroupName/comments/@article.Id/@Helpers.GetUrlTitle(Helpers.HtmlDecode(article.Title))/" target="\_blank">@article.Comments @Resources.Article\_Comments</a> <!--share save hide report--></div>

<div class="postClear"></div>

<img id="org\_image\_@index" class="orgImage" onerror="if (this.src != '/content/images/noimage.jpg') this.src = '/content/images/noimage.jpg';" width="100%" src="@article.ImageUrl" alt="picture">

<div id="org\_article\_@index" class="orgArticle">@Html.Raw(Helpers.HtmlDecode(article.Content))</div>

</div>

</div>

<div class="postClear"></div>

}

index++;

}

<span class="footermenu">

@Resources.Article\_More

@if (ShowPrev())

{

<a class="nextbutton" href="?page=@GetPrevPageNo()">< @Resources.Article\_Prev </a><span class="separator"></span>

}

<a class="nextbutton" href="?page=@GetNextPageNo()">@Resources.Article\_Next > </a>

@Resources.Article\_OrTry <a class="nextbutton" href="/@locale/random"> @Resources.Article\_RandomSubReddah</a>

</span>

</div>

@if (Model.Articles.Count < 5)

{

<div id="box" style="height:100vh;"></div>

}

namespace Reddah.Web.UI.Utility

{

using System;

using System.Linq;

using System.Net.Mail;

using log4net;

public static class EmailHelper

{

private static readonly ILog log = log4net.LogManager.GetLogger("EmailHelper");

public static void Send(

MailAddress from,

MailAddress to,

string subject,

string body

)

{

try

{

var verifyEmailPassword = string.Empty;

using (var context = new reddahEntities1())

{

verifyEmailPassword = context.Settings.FirstOrDefault(s => s.Key == "VerifyEmailPassword").Value;

}

SmtpClient client = new SmtpClient();

client.Host = "smtp.reddah.com";

client.Credentials = new System.Net.NetworkCredential(from.Address, verifyEmailPassword);

client.EnableSsl = false;

MailMessage message = new MailMessage(from, to);

message.Subject = subject;

message.Body = body;

client.Send(message);

}

catch (Exception e)

{

log.Error(e.Message);

}

}

}

}

namespace Reddah.Web.UI.Utility

{

using System;

using System.Linq;

using System.Globalization;

using System.Text.RegularExpressions;

using System.Web.Helpers;

using System.Collections.Generic;

using System.Runtime.InteropServices;

public enum PrivilegeList

{

EditPost=1,

DeletePost=2,

DeleteComment=3

}

public static class Helpers

{

public static bool Acl(string userName, PrivilegeList privilegeList)

{

using (var db = new reddahEntities1())

{

var query = (from user in db.Set<UserProfile>()

join userInRole in db.Set<webpages\_UsersInRoles>() on user.UserId equals userInRole.UserId

join privilegeInRole in db.Set<webpages\_PrivilegesInRoles>() on userInRole.RoleId equals privilegeInRole.RoleId

join priviege in db.Set<webpages\_Privileges>() on privilegeInRole.PrivilegeId equals priviege.PrivilegeId

join role in db.Set<webpages\_Roles>() on userInRole.RoleId equals role.RoleId

where user.UserName == userName && priviege.PrivilegeId == (int)privilegeList

select new

{

user.UserId,

PrivilegeName = priviege.PrivilegeId

}).Take(1);

if (query.Count() > 0)

return true;

}

return false;

}

public static string Display(bool flag)

{

return flag ? "flex" : "";

}

public static string GetColor(int n)

{

return n == 0 ? "black" : "darkgreen";

}

public static string RemoveLocaleFromPath(string originalPath)

{

String path = String.Empty;

if (!String.IsNullOrEmpty(originalPath))

{

path = originalPath.TrimStart('/');

//Remove locale if found

if (originalPath.Length > 5)

{

Regex rgx = new Regex("^[A-Z]{2}-[A-Z]{2}/", RegexOptions.IgnoreCase);

path = rgx.Replace(path, "");

}

}

return path;

}

public static bool IsValidLocale(string localeToTest)

{

if (!String.IsNullOrEmpty(localeToTest))

{

// Get and enumerate all cultures

var cultures = CultureInfo.GetCultures(CultureTypes.AllCultures);

return cultures.Any(elt => elt.ToString().Equals(localeToTest));

}

return false;

}

public static string GetRelativeUrl(string url)

{

return url.StartsWith("/uploadPhoto", StringComparison.InvariantCultureIgnoreCase)?url.Replace("/uploadPhoto",

"///reddah.com/uploadPhoto") : url;

}

public static string GetVideoSrc(string content)

{

var matches = Regex.Match(System.Web.HttpUtility.HtmlDecode(content), "<video.+?src=[\"'](.+?)[\"'].\*?>", RegexOptions.IgnoreCase);

string matchString = string.IsNullOrWhiteSpace(matches.Groups[1].Value) ?

"" : matches.Groups[1].Value;

return GetRelativeUrl(matchString);

}

public static string GetVideoPoster(string content)

{

var matches = Regex.Match(System.Web.HttpUtility.HtmlDecode(content), "<video.+?poster=[\"'](.+?)[\"'].\*?>", RegexOptions.IgnoreCase);

string matchString = string.IsNullOrWhiteSpace(matches.Groups[1].Value) ?

"" : matches.Groups[1].Value;

return GetRelativeUrl(matchString);

}

public static string GetFirstImageSrc(string content)

{

var matches = Regex.Match(System.Web.HttpUtility.HtmlDecode(content), "<img.+?src=[\"'](.+?)[\"'].\*?>", RegexOptions.IgnoreCase);

string matchString = string.IsNullOrWhiteSpace(matches.Groups[1].Value) ?

"" : matches.Groups[1].Value;

return GetRelativeUrl(matchString);

}

public static string[] GetFirstImageSrc(string content, int n)

{

List<string> list = new List<string>();

var matches = Regex.Matches(System.Web.HttpUtility.HtmlDecode(content), "<img.+?src=[\"'](.+?)[\"'].\*?>", RegexOptions.IgnoreCase);

var count = 0;

foreach (Match m in matches)

{

if (count < n)

{

if(!string.IsNullOrWhiteSpace(m.Groups[1].Value))

{

list.Add(GetRelativeUrl(m.Groups[1].Value));

count++;

}

}

else

break;

}

return list.ToArray<string>();

}

public static string TimeAgoNullable(Nullable<System.DateTime> dt)

{

if (dt == null)

return null;

else

return dt.Value.ToShortDateString() + " " + dt.Value.ToShortTimeString();

}

public static string TimeAgo(DateTime dt)

{

TimeSpan span = DateTime.Now - dt;

return TimeAgo(span);

}

private static string TimeAgo(TimeSpan span)

{

if (span.Days > 365)

{

int years = (span.Days / 365);

if (span.Days % 365 != 0)

years += 1;

return String.Format(Resources.Resources.List\_Time\_Format,

years, years == 1 ? Resources.Resources.List\_Time\_Year : Resources.Resources.List\_Time\_Years);

}

if (span.Days > 30)

{

int months = (span.Days / 30);

if (span.Days % 31 != 0)

months += 1;

return String.Format(Resources.Resources.List\_Time\_Format,

months, months == 1 ? Resources.Resources.List\_Time\_Month : Resources.Resources.List\_Time\_Months);

}

if (span.Days > 0)

return String.Format(Resources.Resources.List\_Time\_Format,

span.Days, span.Days == 1 ? Resources.Resources.List\_Time\_Day : Resources.Resources.List\_Time\_Days);

if (span.Hours > 0)

return String.Format(Resources.Resources.List\_Time\_Format,

span.Hours, span.Hours == 1 ? Resources.Resources.List\_Time\_Hour : Resources.Resources.List\_Time\_Hours);

if (span.Minutes > 0)

return String.Format(Resources.Resources.List\_Time\_Format,

span.Minutes, span.Minutes == 1 ? Resources.Resources.List\_Time\_Minute : Resources.Resources.List\_Time\_Minutes);

if (span.Seconds > 5)

return String.Format(Resources.Resources.List\_Time\_Format, span.Seconds, Resources.Resources.List\_Time\_Seconds);

if (span.Seconds <= 5)

return Resources.Resources.List\_Time\_Just;

return string.Empty;

}

public static string GetShortTitle(string title)

{

var urlTitle = string.Empty;

if (!String.IsNullOrWhiteSpace(title))

{

urlTitle = title;

}

else

{

return "article";

}

//urlTitle = Regex.Replace(urlTitle, @"[^A-Za-z0-9 ]+", "");

//urlTitle = urlTitle.Trim().Replace(" ", "-");

return SubString(urlTitle, Math.Min(50, urlTitle.Length - 1));

}

public static string GetUrlTitle(string title)

{

var urlTitle = string.Empty;

if (!String.IsNullOrWhiteSpace(title))

{

urlTitle = title;

}

else

{

return "reddah\_article";

}

urlTitle = urlTitle.Trim().Replace(" ", "-");

string[] invalid = { "<", ">", "\*", "%", "&", ":", "/", "." };

foreach (string item in invalid) {

urlTitle = urlTitle.Replace(item, "-");

}

string[] opts = { ",", ";", "'", "(", ")"};

foreach (string item in opts)

{

urlTitle = urlTitle.Replace(item, "");

}

urlTitle = urlTitle.Replace("--", "-");

if (urlTitle.IndexOf('-') == 0 && urlTitle.Length>1)

urlTitle.Remove(1);

urlTitle = SubString(urlTitle, Math.Min(50, urlTitle.Length));

if (urlTitle.LastIndexOf('-') == urlTitle.Length - 1)

urlTitle = SubString(urlTitle, urlTitle.Length - 1);

return urlTitle;

}

public static string GetAntiForgeryToken()

{

string cookieToken, formToken;

AntiForgery.GetTokens(null, out cookieToken, out formToken);

return cookieToken + ":" + formToken;

}

public static string HideSensitiveWords(string text)

{

var re = new Regex(@"\b(\w+)\b", RegexOptions.Compiled);

var replacements = new Dictionary<string, string>(StringComparer.OrdinalIgnoreCase)

{

{"fuck", "duck"},

{"bitch", "beach"},

{"asshole", "niceweather"},

{"操他妈","#$@" },

{"操你妈","#\*@" },

{"你妈逼","\*@O" },

};

return re.Replace(text, match => replacements.ContainsKey(match.Groups[1].Value) ?

replacements[match.Groups[1].Value] : match.Groups[1].Value);

}

public static string HideXss(string text)

{

var re = new Regex(@"\b(\w+)\b", RegexOptions.Compiled);

var replacements = new Dictionary<string, string>(StringComparer.OrdinalIgnoreCase)

{

{"<applet>", "(applet)"},

{"<body>", "(body)"},

{"<embed>", "(embed)"},

{"<frame>", "(frame)"},

{"<script>", "(script)"},

{"<frameset>", "(frameset)"},

{"<html>", "(html)"},

{"<iframe>", "(iframe)"},

{"<img>", "(img)"},

{"<style>", "(style)"},

{"<layer>", "(layer)"},

{"<link>", "(link)"},

{"<ilayer>", "(ilayer)"},

{"<meta>", "(meta)"},

{"<object>", "(object)"},

{"&lt;script&gt;", "(script)"},

};

return re.Replace(text, match => replacements.ContainsKey(match.Groups[1].Value) ?

replacements[match.Groups[1].Value] : match.Groups[1].Value);

}

public static string HtmlEncode(string text)

{

return System.Web.HttpUtility.HtmlEncode(HideXss(HideSensitiveWords(text)));

}

public static string HtmlDecode(string text)

{

return System.Web.HttpUtility.HtmlDecode(HideXss(text));

}

public static string ReplaceHtmlTag(string html)

{

string stroutput = html;

Regex regex = new Regex(@"<[^>]+>|</[^>]+>");

stroutput = regex.Replace(stroutput, "");

return stroutput;

}

public static string ToAudioString(string html)

{

html = HtmlDecode(html);

html = System.Web.HttpUtility.HtmlDecode(html);

return SubString(ReplaceHtmlTag(html), Math.Min(500, html.Length - 1));

}

public static string SubString(string str, int len)

{

string result = string.Empty;

if (string.IsNullOrEmpty(str))

{

return result;

}

int byteLen = System.Text.Encoding.Default.GetByteCount(str);

int charLen = str.Length;

int byteCount = 0;

int pos = 0;

if (byteLen > len)

{

for (int i = 0; i < charLen; i++)

{

if (Convert.ToInt32(str.ToCharArray()[i]) > 255)

{

byteCount += 2;

}

else

{

byteCount += 1;

}

if (byteCount > len)

{

pos = i;

break;

}

else if (byteCount == len)

{

pos = i + 1; break;

}

}

if (pos >= 0)

{

result = str.Substring(0, pos);

}

}

else { result = str; }

return result;

}

//public static string GetLocalizedPath(string originalPath)

//{

// String path = RemoveLocaleFromPath(originalPath);

// var culture = new CultureInfo("en-US");

// return string.Join("/", culture.Name, path);

//}

public static string ToSimplified(string source)

{

return Microsoft.VisualBasic.Strings.StrConv(source, Microsoft.VisualBasic.VbStrConv.SimplifiedChinese, 0);

}

public static string ToTraditional(string source)

{

return Microsoft.VisualBasic.Strings.StrConv(source, Microsoft.VisualBasic.VbStrConv.TraditionalChinese, 0);

}

}

}

angular.module("reddahApp", [])

.controller("appCtrl", ['$scope', 'loginSvc', function ($scope, loginSvc) {

$scope.subpost = function (str, n) {

var r = /[^\u4e00-\u9fa5]/g;

if (str.replace(r, "mm").length <= n) { return str; }

var m = Math.floor(n / 2);

for (var i = m; i < str.length; i++) {

if (str.substr(0, i).replace(r, "mm").length >= n) {

return str.substr(0, i) + "...";

}

}

return str;

};

$scope.toTop = function () {

$("html,body").animate({ scrollTop: 0 }, 500);

};

$scope.htmlDecode = function (text) {

var temp = document.createElement("div");

temp.innerHTML = text;

var output = temp.innerText || temp.textContent;

temp = null;

return output;

};

$scope.urlDecode = function (str) {

var s = "";

if (str.length == 0) return "";

s = $scope.htmlDecode(str).replace(/ /g, "-");

s = s.replace(/--/g, "-");

s = s.replace(/[^a-zA-Z0-9,\-\u4e00-\u9fa5]/g, "");

return s;

};

$scope.summary = function (str, n) {

text = $scope.htmlDecode(str).replace(/<[^>]+>/g, "");

return $scope.htmlDecode($scope.subpost(text, n));

};

$scope.verify = null;

$scope.loginModel = { UserName: "", Password: "", RememberMe: false, ReturnUrl: window.location.href, Error: "", Token: "", Locale: "en-us" };

$scope.showPanel = function () {

$('div#BroswerOpacify').show();

$('div#LoginRegisterPanel').show();

var toAlign = $('div#LoginRegisterPanel');

toAlign.css('top', (($(window).height() - toAlign.outerHeight()) / 2) + $(window).scrollTop() + 'px');

toAlign.css('left', (($(window).width() - toAlign.outerWidth()) / 2) + $(window).scrollLeft() + 'px');

};

$scope.data = {

current: "1" // 1代表Login，2代表Register

};

$scope.showPanelLoginOnly = function (n='1') {

$scope.data.current = n;

$('div#BroswerOpacify').show();

$('div#LoginRegisterPanel').show();

var toAlign = $('div#LoginRegisterPanel');

toAlign.css('top', (($(window).height()) / 2) + $(window).scrollTop() + 'px');

};

$scope.hidePanel = function () {

$('div#BroswerOpacify').hide();

$('div#LoginRegisterPanel').hide();

}

$scope.showImage = function (box, index) {

if ($('.' + box + ' img#org\_image\_' + index).attr("src") === "/content/images/noimage.jpg")

return;

$('.' + box + ' img#org\_image\_' + index).toggle();

}

$scope.toggleArticle = function (box, index) {

$('.' + box + ' div#org\_article\_' + index).toggle();

}

$scope.vote = function (articleId, value, orgCount, page, tick) {

$scope.voteModel = {

ArticleId: articleId,

Value: value,

Locale: 'en-us'

};

if ($scope.verify == null)

$scope.verify = articleId + tick;

else {

if ($scope.verify == articleId + tick) {

return;

}

$scope.verify = articleId + tick;

}

loginSvc.vote($scope.voteModel).then(function (data) {

if (data.success == true) {

var newCount = value == "up" ? orgCount + 1 : orgCount - 1;

$('.' + page + ' span#vote-count-post-' + articleId).text(newCount);

if (page == "articleContent" || page == "articleContentMobile") {

$('#voteButton').hide();

$('#voteInfo').hide();

$('#voteThanks').show();

}

}

else {

var str = '';

for (var error in data.errors) {

str += data.errors[error] + '\n';

}

alert(str);

}

})

}

}])

angular.module("reddahApp")

.controller("loginCtrl", ['$scope', 'loginSvc', function ($scope, loginSvc) {

$scope.actions =

{

setCurrent: function (param) {

$scope.data.current = param;

}

}

$scope.loginOnEnter = function (e) {

var keycode = window.event ? e.keyCode : e.which;

if (keycode == 13) {

$scope.login();

}

};

$scope.loadingLogin = false;

$scope.loginModel = {

UserName: "",

Password: "",

RememberMe: false,

ReturnUrl: window.location.href,

Error: "",

Token: "",

Locale: "en-us"

};

$scope.login = function () {

$scope.loadingLogin = true;

$scope.loginModel.Token = $scope.antiForgeryToken;

$scope.loginModel.Locale = $scope.locale;

loginSvc.login($scope.loginModel).then(function (data) {

if (data.success == true) {

if (data.redirect.indexOf("?")!=-1)

data.redirect += "&login=true";

else

data.redirect += "?login=true";

window.location.replace(data.redirect);

}

else {

$scope.loadingLogin = false;

var str = '';

for (var error in data.errors) {

str += data.errors[error] + '\n';

}

$scope.loginModel.Error = str;

}

})

};

$scope.loadingRegister = false;

$scope.registerModel = {

UserName: "",

Password: "",

ConfirmPassword: "",

Email: "",

RememberMe: false,

ReturnUrl: window.location.href,

Error: "",

Token: "",

Locale: "en-us"

};

$scope.register = function () {

$scope.loadingRegister = true;

$scope.registerModel.Token = $scope.antiForgeryToken;

$scope.registerModel.Locale = $scope.locale;

loginSvc.register($scope.registerModel).then(function (data) {

if (data.success == true) {

window.location.replace(data.redirect);

}

else {

$scope.loadingRegister = false;

var str = '';

for (var error in data.errors) {

str += data.errors[error] + '\n';

}

$scope.registerModel.Error = str;

}

})

};

}])

.controller("nextprevbox", ['$scope', function ($scope) {

$scope.currentIndex = 1;

$scope.next = function (total) {

if ($scope.currentIndex < total) {

$scope.currentIndex++;

}

else {

$scope.currentIndex = 1;

}

$scope.showPost();

}

$scope.prev = function (total) {

if ($scope.currentIndex > 1) {

$scope.currentIndex--;

}

else

{

$scope.currentIndex = total;

}

$scope.showPost();

}

$scope.showPost = function () {

$('div.upcomingPost').hide();

$('div#upcomingPost-' + $scope.currentIndex).show();

};

}])

.factory("loginSvc", ['$http', '$q', function ($http, $q) {

return {

login: function (data) {

var defer = $q.defer();

$http({

method: 'post',

url: '/' + data.Locale + '/Account/JsonLogin',

data: data,

headers: { 'RequestVerificationToken': data.Token }

}).success(function (data, status, headers, config) {

defer.resolve(data);

}).error(function (data, status, headers, config) {

defer.reject(data)

});

return defer.promise;

},

register: function (data) {

var defer = $q.defer();

$http({

method: 'post',

url: '/' + data.Locale + '/Account/JsonRegister',

data: data,

headers: { 'RequestVerificationToken': data.Token }

}).success(function (data, status, headers, config) {

defer.resolve(data);

}).error(function (data, status, headers, config) {

defer.reject(data)

});

return defer.promise;

},

vote: function (data) {

var defer = $q.defer();

$http({

method: 'post',

url: '/' + data.Locale + '/Articles/JsonVote',

data: data

}).success(function (data, status, headers, config) {

defer.resolve(data);

}).error(function (data, status, headers, config) {

defer.reject(data)

});

return defer.promise;

},

}

}])

<ion-tabs \*ngIf="isAuthenticated()" #myTabs>

<!--appSwipetab (ionTabsDidChange)="ionTabsDidChange($event)" (tabChange)="onTabChange($event)"-->

<ion-tab tab="message">

<ion-router-outlet name="message"></ion-router-outlet>

</ion-tab>

<ion-tab tab="contact">

<ion-router-outlet name="contact"></ion-router-outlet>

</ion-tab>

<ion-tab tab="earth">

<ion-router-outlet name="earth"></ion-router-outlet>

</ion-tab>

<ion-tab tab="find">

<ion-router-outlet name="find"></ion-router-outlet>

</ion-tab>

<ion-tab tab="about" #about>

<ion-router-outlet name="about"></ion-router-outlet>

</ion-tab>

<ion-tab tab="home">

<ion-router-outlet name="home"></ion-router-outlet>

</ion-tab>

<ion-tab-bar slot="bottom">

<ion-tab-button tab="home" href="/tabs/(home:home)" (click)="change()">

<ion-icon name="book" style="width: 20%;position: relative;"></ion-icon>

<ion-label class="tab">{{ 'Menu.Recommend' | translate }}</ion-label>

</ion-tab-button>

<ion-tab-button tab="find" href="/tabs/(find:find)" (click)="change()">

<div style="position:absolute;right:20px;top:10px;" [ngClass]="{'new-point':isIos&&reddah.unReadMessage.length>0}"></div>

<ion-icon name="compass" style="width:20%;position: relative;"></ion-icon>

<ion-label class="tab">{{ 'Menu.Find' | translate }}</ion-label>

<ion-badge class="tab-count" color="danger" \*ngIf="!isIos&&reddah.unReadMessage.length>0">{{reddah.unReadMessage.length}}</ion-badge>

</ion-tab-button>

<ion-tab-button (click)="openEarth()">

<ion-icon name="globe" style="width: 80px;height:100%;position: absolute;"></ion-icon>

</ion-tab-button>

<ion-tab-button tab="contact" href="/tabs/(contact:contact)" (click)="change()">

<ion-icon name="ios-people" style="width: 20%;position: relative;"></ion-icon>

<ion-label class="tab">{{ 'Menu.Contact' | translate }}</ion-label>

</ion-tab-button>

<ion-tab-button tab="about" href="/tabs/(about:about)" (click)="change('about')">

<ion-icon name="person" style="width: 20%;position: relative;"></ion-icon>

<ion-label class="tab">{{ 'Menu.About' | translate }}</ion-label>

</ion-tab-button>

</ion-tab-bar>

</ion-tabs>

import { Component, OnInit, ViewChild } from '@angular/core';

import { AuthService } from '../auth.service';

import { Platform, Events } from '@ionic/angular';

import { LocalStorageService } from 'ngx-webstorage';

import { ModalController } from '@ionic/angular';

import { StatusBar } from '@ionic-native/status-bar/ngx';

import { Router, ActivatedRoute, Params } from '@angular/router';

import { ReddahService } from '../reddah.service';

import { SwipeTabDirective } from '../swipe-tab.directive';

import { Tabs } from '@ionic/angular';

import { EarthPage } from './earth/earth.page';

import { MapPage } from '../map/map.page';

@Component({

selector: 'app-tabs',

templateUrl: 'tabs.page.html',

styleUrls: ['tabs.page.scss']

})

export class TabsPage implements OnInit {

@ViewChild(SwipeTabDirective) swipeTabDirective: SwipeTabDirective;

@ViewChild('myTabs') tabRef: Tabs;

@ViewChild('about') about;

isAndroid = false;

isIos = false;

constructor(

private authService: AuthService,

private platform: Platform,

private localStorageService: LocalStorageService,

private modalController: ModalController,

private statusBar: StatusBar,

private router: Router,

public reddah: ReddahService,

public events: Events,

)

{

this.events.subscribe('tab:clicked', (data) =>

{

try{

let index = data['tab'];

let lat = data['lat'];

let lng = data['lng'];

//this.tabRef.select(this.map);

this.router.navigateByUrl(`/tabs/(map:map)?lat=${lat}&lng=${lng}`);

}catch(e){}

});

}

ngOnInit(){

let locale = this.localStorageService.retrieve("Reddah\_Locale");

if(locale==null){

}

if(this.platform.is('android')){

this.isAndroid = true;

}

else if(this.platform.is('iphone')||this.platform.is('ipad')||this.platform.is('ios')){

this.isIos = true;

}

}

ionTabsDidChange($event) {

//console.log('[TabsPage] ionTabsDidChange, $event: ', $event);

this.swipeTabDirective.onTabInitialized($event.tab);

}

onTabChange($event) {

//console.log('[TabsPage] onTabChange, $event: ', $event);

this.tabRef.select($event);

}

change(page=null){

if(page=="about"){

this.about.checked = false;

}

/\*if(page=="about"){

this.statusBar.overlaysWebView(true);

this.statusBar.backgroundColorByHexString("#ffffff");

this.localStorageService.store("statusbar\_bgcolor", "white");

this.statusBar.styleDefault();

}

else{

let lastcolor = this.localStorageService.retrieve("statusbar\_bgcolor");

if(lastcolor!="#eeeeee"){

this.localStorageService.store("statusbar\_bgcolor", "#eeeeee");

this.statusBar.overlaysWebView(true);

this.statusBar.backgroundColorByHexString("#eeeeee");

this.statusBar.styleDefault();

}

}\*/

this.reddah.getMessageUnread().subscribe(data=>{

if(data.Success==0){

this.reddah.unReadMessage = data.Message;

}

});

}

logout() {

this.authService.logout();

}

isAuthenticated() {

return this.authService.authenticated();

}

ionViewDidEnter(){

//this.subscription = this.platform.backButton.subscribe(()=>{

// navigator['app'].exitApp();

//});

}

ionViewWillLeave(){

//this.subscription.unsubscribe();

}

isLocaleSet(){

let locale = this.localStorageService.retrieve("Reddah\_Locale");

if(locale==undefined||locale==null)

return false;

return true;

}

async openEarth(){

let yearslater = false;

if(yearslater){

const modal = await this.modalController.create({

component: EarthPage,

componentProps: {},

cssClass: "modal-fullscreen",

});

await modal.present();

}

else{

const modal = await this.modalController.create({

component: MapPage,

componentProps: {

//lat: this.config.lat,

//lng: this.config.lng

},

cssClass: "modal-fullscreen",

});

await modal.present();

}

}

}using System.ComponentModel.DataAnnotations;

using System.Data.Entity;

using System.Web.Mvc;

namespace Reddah.Web.UI.Models

{

public class UsersContext : DbContext

{

public UsersContext()

: base("reddahEntities1")

{

}

public DbSet<UserProfile> UserProfiles { get; set; }

}

public class RegisterExternalLoginModel

{

[Required]

[Display(Name = "User name")]

public string UserName { get; set; }

public string ExternalLoginData { get; set; }

}

public class LocalPasswordModel

{

[Required]

[DataType(DataType.Password)]

[Display(Name = "Current password")]

public string OldPassword { get; set; }

[Required]

[StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]

[DataType(DataType.Password)]

[Display(Name = "New password")]

public string NewPassword { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Confirm new password")]

[Compare("NewPassword", ErrorMessage = "The new password and confirmation password do not match.")]

public string ConfirmPassword { get; set; }

}

public class LoginModel

{

[Required(ErrorMessageResourceType = typeof(Resources.Resources), ErrorMessageResourceName = "Login\_Msg\_UserNameRequire")]

[Display(Name = "User name")]

public string UserName { get; set; }

[Required(ErrorMessageResourceType = typeof(Resources.Resources), ErrorMessageResourceName = "Login\_Msg\_PasswordRequire")]

[DataType(DataType.Password)]

[Display(Name = "Password")]

public string Password { get; set; }

[Display(Name = "Login\_RememberMe", ResourceType = typeof(Resources.Resources))]

public bool RememberMe { get; set; }

}

public class RegisterModel

{

[Required(ErrorMessageResourceType = typeof(Resources.Resources), ErrorMessageResourceName = "Login\_Msg\_UserNameRequire")]

[Display(Name = "User name")]

public string UserName { get; set; }

[Required(ErrorMessageResourceType = typeof(Resources.Resources), ErrorMessageResourceName = "Login\_Msg\_PasswordRequire")]

[StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]

[DataType(DataType.Password)]

[Display(Name = "Password")]

public string Password { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Confirm password")]

[Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]

public string ConfirmPassword { get; set; }

[Required(ErrorMessageResourceType = typeof(Resources.Resources), ErrorMessageResourceName = "Login\_Msg\_EmailRequire")]

[Display(Name = "Email")]

public string Email { get; set; }

//[Display(Name = "Remember me?")]

[Display(Name = "Login\_RememberMe", ResourceType = typeof(Resources.Resources))]

public bool RememberMe { get; set; }

}

public class ExternalLogin

{

public string Provider { get; set; }

public string ProviderDisplayName { get; set; }

public string ProviderUserId { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace Reddah.Web.UI.Models

{

public class ArticlePreview

{

public int Id { get; set; }

public string Title { get; set; }

public string Description { get; set; }

public string ImageUrl { get; set; }

public string[] ImageUrls { get; set; }

public string VideoUrl { get; set; }

public string VideoPoster { get; set; }

public string ArticleUrl { get; set; }

public int Count { get; set; }

public int Up { get; set; }

public int Down { get; set; }

public int Comments { get; set; }

public string Abstract { get; set; }

public string CreatedOn { get; set; }

public DateTime CreatedOnOrg { get; set; }

public string UserName { get; set; }

public string GroupName { get; set; }

public string Content { get; set; }

public string Locale { get; set; }

public DateTime? LastUpdateOn { get; set; }

public Nullable<int> Type { get; set; }

public int? Ref { get; set; }

}

public class VoteModel {

[Required]

[Display(Name = "Article Id")]

public int ArticleId { get; set; }

[Required]

[Display(Name = "Vote Value")]

public string Value { get; set; }

}

public class ArticleComment

{

public int Id { get; set; }

public int ParentId { get; set; }

public string Content{get;set;}

public string UserName {get;set;}

public int Count { get; set; }

public int Up { get; set; }

public int Down { get; set; }

}

public class SeededComments

{

public int Seed { get; set; }

public IList<Comment> Comments { get; set; }

}

public class CommentModel

{

[Required]

[Display(Name = "Article ID")]

public int ArticleId { get; set; }

[Required]

[Display(Name = "Parent Id")]

public int ParentId { get; set; }

[Required]

[Display(Name = "Comment Content")]

public string Content { get; set; }

}

public class UserProfileModel

{

public int[] LoadedIds;

public string[] DislikeGroups;

public string[] DislikeUserNames;

public string Locale;

public string Token;

public string Menu;

public string Sub;

public string User;

public string Keyword;

public int Type;//1 timeline, others: article

public int Status;//1 published, 0: draft

//public Dictionary<string, int> habits;

}

public class JsonGroup

{

public string Name;

public string Locale;

public string Desc;

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Reddah.Web.Login

{

public class BaseModel

{

public string Jwt { get; set; }

}

public class CommentQuery : BaseModel

{

public int ArticleId { get; set; }

}

public class NewComment : BaseModel

{

public int ArticleId { get; set; }

public int ParentId { get; set; }

public string Content { get; set; }

public string Uid { get; set; }

}

public class NewTimeline : BaseModel

{

public string Thoughts { get; set; }

public string Content { get; set; }

public string Location { get; set; }

}

public class SeededComments

{

public int Seed { get; set; }

public IList<AdvancedComment> Comments { get; set; }

}

public class GroupChatSeededComments

{

public Article Group { get; set; }

public IList<AdvancedComment> Comments { get; set; }

}

public class AdvancedComment : Comment

{

public string UserNickName { get; set; }

public string UserPhoto { get; set; }

public Nullable<int> UserSex { get; set; }

}

public class AdvancedTimeline : Article

{

public string UserNickName { get; set; }

public string UserPhoto { get; set; }

public Nullable<int> UserSex { get; set; }

}

public class AdvancedUserFriend : UserFriend

{

public string UserNickName { get; set; }

public string UserPhoto { get; set; }

public Nullable<int> UserSex { get; set; }

public string Signature { get; set; }

}

public class AdvancedPub : Article

{

public string UserName { get; set; }

public string UserNickName { get; set; }

public string Signature { get; set; }

public string UserPhoto { get; set; }

public string UserCover { get; set; }

public string Email { get; set; }

}

public class AdvancedUserArticle: UserArticle

{

public string Title { get; set; }

public string OrgUserName { get; set; }

public string PreviewPhoto { get; set; }

public Article article { get; set; }

}

public class AdvancedUserProfile : UserProfile

{

public decimal? Distance { get; set; }

}

public class AdvancedArticle: Article

{

public string ImageUrl { get; set; }

public AdvancedArticle(Article article)

{

this.Id = article.Id;

this.Title = article.Title;

this.Content = article.Content;

this.Abstract = article.Abstract;

this.CreatedOn = article.CreatedOn;

this.Up = article.Up;

this.Down = article.Down;

this.Count = article.Count;

this.UserName = article.UserName;

this.GroupName = article.GroupName;

this.Locale = article.Locale;

this.LastUpdateOn = article.LastUpdateOn;

this.Type = article.Type;

this.Ref = article.Ref;

}

}

}

using System.Data.Entity;

namespace Reddah.Web.Login

{

public class UsersContext : DbContext

{

public UsersContext()

: base("reddahEntities")

{

}

public DbSet<UserProfile> UserProfiles { get; set; }

}

public class UserModel

{

public string UserName { get; set; }

public string Password { get; set; }

}

public class JwtUser

{

public string User { get; set; }

public string Allow { get; set; }

}

public class UserInfo

{

public int Type { get; set; }

public string UserName { get; set; }

public string NickName { get; set; }

public int Sex { get; set; }

public string Photo { get; set; }

public string Location { get; set; }

public string Admins { get; set; }

public int Point { get; set; }

public string Signature { get; set; }

public string Cover { get; set; }

public string Email { get; set; }

public string NoteName { get; set; }

public bool IsFriend { get; set; }

public bool HideLocation { get; set; }

public bool AllowTenTimeline { get; set; }

}

}