**天天休闲生活软件源程序**

package com.zxl.casual.living;

import android.app.Application;

import com.zxl.casual.living.utils.CommonUtils;

import com.zxl.casual.living.utils.EventBusUtils;

import com.tencent.mm.opensdk.openapi.WXAPIFactory;

import com.zxl.common.DebugUtil;

/\*\*

\* Created by zxl on 2018/9/6.

\*/

public class MyApplication extends Application {

@Override

public void onCreate() {

super.onCreate();

DebugUtil.IS\_DEBUG = DebugUtil.STATE\_OPEN;

EventBusUtils.init();

}

}

package com.zxl.casual.living;

import android.Manifest;

import android.content.Context;

import android.content.pm.PackageManager;

import android.graphics.Color;

import android.os.Bundle;

import android.os.Handler;

import android.os.Message;

import android.support.annotation.NonNull;

import android.support.v4.app.ActivityCompat;

import android.support.v4.app.Fragment;

import android.support.v4.app.FragmentTransaction;

import android.support.v4.content.ContextCompat;

import android.support.v4.widget.DrawerLayout;

import android.support.v7.app.ActionBarDrawerToggle;

import android.support.v7.app.AppCompatActivity;

import android.support.v7.widget.Toolbar;

import android.view.View;

import android.widget.FrameLayout;

import android.widget.ImageView;

import android.widget.Toast;

import com.bumptech.glide.Glide;

import com.zxl.casual.living.event.BackSelectLeftMenuEvent;

import com.zxl.casual.living.event.LocatePermissionSuccessEvent;

import com.zxl.casual.living.event.RequestLocatePermissionEvent;

import com.zxl.casual.living.event.SelectLeftMenuEvent;

import com.zxl.casual.living.fragment.AccountFragment;

import com.zxl.casual.living.fragment.CheckVersionFragment;

import com.zxl.casual.living.fragment.CollectQSBKFragment;

import com.zxl.casual.living.fragment.LeftMenuFragment;

import com.zxl.casual.living.fragment.QSBKFragment;

import com.zxl.casual.living.fragment.TaoBaoAnchorFragment;

import com.zxl.casual.living.http.data.TodayWeather;

import com.zxl.casual.living.http.data.TodayWeatherResponseBean;

import com.zxl.casual.living.utils.CommonUtils;

import com.zxl.casual.living.utils.Constants;

import com.zxl.casual.living.utils.EventBusUtils;

import com.zxl.casual.living.utils.SharePreUtils;

import com.tencent.mm.opensdk.modelmsg.SendMessageToWX;

import com.tencent.mm.opensdk.modelmsg.WXMediaMessage;

import com.tencent.mm.opensdk.modelmsg.WXTextObject;

import com.zxl.common.DebugUtil;

import org.greenrobot.eventbus.Subscribe;

import org.greenrobot.eventbus.ThreadMode;

import java.util.ArrayList;

import java.util.List;

import java.util.Stack;

public class MainActivity extends AppCompatActivity {

private static final String TAG = "MainActivity";

private Context mContext;

private static final int MSG\_CANCEL\_CLICK\_BACK\_TO\_FINISH = 1;

private String[] permissions = new String[]{

Manifest.permission.WRITE\_EXTERNAL\_STORAGE,

Manifest.permission.READ\_EXTERNAL\_STORAGE,

Manifest.permission.ACCESS\_COARSE\_LOCATION,

Manifest.permission.ACCESS\_FINE\_LOCATION,

// Manifest.permission.WRITE\_SETTINGS

};

private Toolbar mToolbar;

private DrawerLayout mDrawerLayout;

private FrameLayout mLeftMenuView;

private ActionBarDrawerToggle mActionBarDrawerToggle;

private LeftMenuFragment mLeftMenuFragment;

// private QSBKFragment mQSBKFragment;

// private TaoBaoAnchorFragment mTaoBaoAnchorFragment;

// private AccountFragment mAccountFragment;

private List<Fragment> mContentFragments = new ArrayList<>();

private Stack<Integer> mLeftMenuPositionStack = new Stack<>();

private boolean isClickBackToFinish = false;

private Handler mHandler = new Handler(){

@Override

public void handleMessage(Message msg) {

switch (msg.what){

case MSG\_CANCEL\_CLICK\_BACK\_TO\_FINISH:

isClickBackToFinish = false;

break;

}

}

};

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

DebugUtil.d(TAG,"onCreate");

CommonUtils.regToWX(this);

mContext = this;

EventBusUtils.register(this);

mToolbar = findViewById(R.id.custom\_tool\_bar);

mDrawerLayout = findViewById(R.id.drawer\_layout);

mLeftMenuView = findViewById(R.id.left\_menu\_view);

mToolbar.setTitle("Toolbar");//设置Toolbar标题

mToolbar.setTitleTextColor(Color.parseColor("#ffffff")); //设置标题颜色

setSupportActionBar(mToolbar);

getSupportActionBar().setHomeButtonEnabled(true); //设置返回键可用

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

getSupportActionBar().setDisplayShowTitleEnabled(false);

mActionBarDrawerToggle = new ActionBarDrawerToggle(this,mDrawerLayout,mToolbar,R.string.drawer\_open,R.string.drawer\_close){

@Override

public void onDrawerOpened(View drawerView) {

super.onDrawerOpened(drawerView);

}

@Override

public void onDrawerClosed(View drawerView) {

super.onDrawerClosed(drawerView);

}

};

mActionBarDrawerToggle.syncState();

mDrawerLayout.addDrawerListener(mActionBarDrawerToggle);

showLeftMenuFragment();

initContentFragments();

showContentFragment(Constants.LEFT\_MENU\_POSITION\_0);

mLeftMenuPositionStack.push(Constants.LEFT\_MENU\_POSITION\_0);

mLeftMenuFragment.setToolbar(mToolbar);

// String s = "{\"code\": 0, \"address\_info\": \"南京市\", \"today\_weather\": {\"now\_time\": \"17:50 实况\", \"temperature\": \"22\", \"is\_w\": 1, \"simple\_content\": \"周一 阴转小雨 18/24°C\", \"wind\_direction\": \"东风\", \"air\_quality\": \"67良\", \"humidity\": \"64%\", \"humidity\_icon\_css\": {\"width\": \"24\", \"height\": \"24\", \"background\_position\_x\": \"-2\", \"img\": \"http://i.tq121.com.cn/i/weather2015/city/iconall.png\", \"background\_position\_y\": \"-289\"}, \"is\_limit\": 0, \"wind\_icon\_css\": {\"width\": \"24\", \"height\": \"24\", \"background\_position\_x\": \"-36\", \"img\": \"http://i.tq121.com.cn/i/weather2015/city/iconall.png\", \"background\_position\_y\": \"-291\"}, \"air\_quality\_icon\_css\": {\"width\": \"24\", \"height\": \"24\", \"background\_position\_x\": \"-2\", \"img\": \"http://i.tq121.com.cn/i/weather2015/city/iconall.png\", \"background\_position\_y\": \"-314\"}, \"is\_h\": 1, \"wind\_value\": \"2级\", \"is\_pol\": 1, \"temperature\_icon\_css\": {\"background\_position\_y1\": \"-137\", \"background\_position\_y2\": \"-142\", \"img\": \"http://i.tq121.com.cn/i/weather2015/city/iconall.png\", \"width2\": \"15\", \"width1\": \"15\", \"background\_position\_x2\": \"-35\", \"background\_position\_x1\": \"-35\", \"height1\": \"8\", \"height2\": \"57.5938\"}}, \"city\_name\": \"南京市\", \"today\_weather\_detail\": [{\"sun\_icon\_css\": {\"width\": \"22\", \"height\": \"22\", \"background\_position\_x\": \"-2\", \"img\": \"http://i.tq121.com.cn/i/weather2015/city/iconall.png\", \"background\_position\_y\": \"-269\"}, \"is\_sun\_up\": 0, \"temperature\": \"18\", \"title\": \"8日夜间\", \"wind\_direction\": \"东风\", \"wind\_icon\_css\": {\"width\": \"24\", \"height\": \"25\", \"background\_position\_x\": \"-82\", \"img\": \"http://i.tq121.com.cn/i/weather2015/city/iconall.png\", \"background\_position\_y\": \"-69\"}, \"sun\_time\": \"日落 17:43\", \"weather\": \"阴\", \"wind\_value\": \"3-4级\", \"weather\_icon\_css\": {\"width\": \"80\", \"height\": \"80\", \"background\_position\_x\": \"-160\", \"img\": \"https://i.tq121.com.cn/i/weather2015/png/blue80.png\", \"background\_position\_y\": \"-320\"}}, {\"sun\_icon\_css\": {\"width\": \"22\", \"height\": \"22\", \"background\_position\_x\": \"-33\", \"img\": \"http://i.tq121.com.cn/i/weather2015/city/iconall.png\", \"background\_position\_y\": \"-269\"}, \"is\_sun\_up\": 1, \"temperature\": \"24\", \"title\": \"9日白天\", \"wind\_direction\": \"西北风\", \"wind\_icon\_css\": {\"width\": \"24\", \"height\": \"25\", \"background\_position\_x\": \"-82\", \"img\": \"http://i.tq121.com.cn/i/weather2015/city/iconall.png\", \"background\_position\_y\": \"-188\"}, \"weather\_desc\": \"天空阴沉\", \"weather\": \"小雨\", \"sun\_time\": \"日出 06:03\", \"wind\_value\": \"4-5级\", \"weather\_icon\_css\": {\"width\": \"80\", \"height\": \"80\", \"background\_position\_x\": \"-560\", \"img\": \"https://i.tq121.com.cn/i/weather2015/png/blue80.png\", \"background\_position\_y\": \"0\"}}], \"desc\": \"success\"}";

// try {

// TodayWeatherResponseBean todayWeatherResponseBean = CommonUtils.mGson.fromJson(s, TodayWeatherResponseBean.class);

// DebugUtil.d(TAG, "onCreate::todayWeatherResponseBean = " + todayWeatherResponseBean);

// }catch (Exception e){

// e.printStackTrace();

// }

}

private void showLeftMenuFragment() {

FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();

if(null == mLeftMenuFragment){

mLeftMenuFragment = (LeftMenuFragment) Fragment.instantiate(mContext,"com.zxl.casual.living.fragment.LeftMenuFragment");

fragmentTransaction.add(R.id.left\_menu\_view,mLeftMenuFragment);

}else{

fragmentTransaction.show(mLeftMenuFragment);

}

fragmentTransaction.commit();

}

private void initContentFragments(){

mContentFragments.clear();

QSBKFragment mQSBKFragment = (QSBKFragment) Fragment.instantiate(mContext,"com.zxl.casual.living.fragment.QSBKFragment");

mContentFragments.add(mQSBKFragment);

TaoBaoAnchorFragment mTaoBaoAnchorFragment = (TaoBaoAnchorFragment) Fragment.instantiate(mContext,"com.zxl.casual.living.fragment.TaoBaoAnchorFragment");

mContentFragments.add(mTaoBaoAnchorFragment);

CollectQSBKFragment mCollectQSBKFragment = (CollectQSBKFragment) Fragment.instantiate(mContext,"com.zxl.casual.living.fragment.CollectQSBKFragment");

mContentFragments.add(mCollectQSBKFragment);

AccountFragment mAccountFragment = (AccountFragment) Fragment.instantiate(mContext,"com.zxl.casual.living.fragment.AccountFragment");

mContentFragments.add(mAccountFragment);

CheckVersionFragment mCheckVersionFragment = (CheckVersionFragment) Fragment.instantiate(mContext,"com.zxl.casual.living.fragment.CheckVersionFragment");

mContentFragments.add(mCheckVersionFragment);

}

private void showContentFragment(int index){

for(int i = 0; i < mContentFragments.size(); i++){

FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();

Fragment fragment = mContentFragments.get(i);

DebugUtil.d(TAG,"showContentFragment::i = " + i);

if(i == index){

DebugUtil.d(TAG,"showContentFragment::index = " + index + "::isAdded = " + fragment.isAdded());

DebugUtil.d(TAG,"showContentFragment::fragment = " + fragment);

if(fragment.isAdded()){

fragmentTransaction.show(fragment);

}else{

fragmentTransaction.add(R.id.container\_view,fragment);

fragmentTransaction.show(fragment);

}

fragmentTransaction.commit();

}

}

for(int i = 0; i < mContentFragments.size(); i++){

FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();

Fragment fragment = mContentFragments.get(i);

if(i != index){

fragmentTransaction.hide(fragment);

}

fragmentTransaction.commit();

}

}

private void requestLocatePermission() {

if (PackageManager.PERMISSION\_GRANTED != ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_COARSE\_LOCATION)

|| PackageManager.PERMISSION\_GRANTED != ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_FINE\_LOCATION)

|| PackageManager.PERMISSION\_GRANTED != ContextCompat.checkSelfPermission(this, Manifest.permission.READ\_EXTERNAL\_STORAGE)

|| PackageManager.PERMISSION\_GRANTED != ContextCompat.checkSelfPermission(this, Manifest.permission.WRITE\_EXTERNAL\_STORAGE)) {

ActivityCompat.requestPermissions(this, permissions, 1);

} else {

}

}

@Override

public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {

super.onRequestPermissionsResult(requestCode, permissions, grantResults);

boolean isPermissionOk = true;

if (requestCode == 1) {

for (int result : grantResults) {

if (result != PackageManager.PERMISSION\_GRANTED) {

isPermissionOk = false;

break;

}

}

}

if(isPermissionOk){

EventBusUtils.post(new LocatePermissionSuccessEvent());

}

DebugUtil.d(TAG,"onRequestPermissionsResult::isPermissionOk = " + isPermissionOk);

}

@Override

public void onBackPressed() {

DebugUtil.d(TAG,"onBackPressed::mLeftMenuPositionStack = " + mLeftMenuPositionStack);

if(mDrawerLayout.isDrawerOpen(mLeftMenuView)){

mDrawerLayout.closeDrawer(mLeftMenuView);

}else{

int position = Constants.LEFT\_MENU\_POSITION\_0;

if(mLeftMenuPositionStack.size() > 0){

position = mLeftMenuPositionStack.get(mLeftMenuPositionStack.size() - 1);

if(mContentFragments.get(position) instanceof QSBKFragment){

QSBKFragment qsbkFragment = (QSBKFragment) mContentFragments.get(position);

boolean isNeedHand = qsbkFragment.onBackPressed();

if(isNeedHand){

return;

}

}

}

if(mLeftMenuPositionStack.size() > 1){

mLeftMenuPositionStack.pop();

position = mLeftMenuPositionStack.get(mLeftMenuPositionStack.size() - 1);

if(position == Constants.LEFT\_MENU\_POSITION\_2 && SharePreUtils.getInstance(mContext).getUserInfo() == null){

mLeftMenuPositionStack.pop();

position = mLeftMenuPositionStack.get(mLeftMenuPositionStack.size() - 1);

}

showContentFragment(position);

EventBusUtils.post(new BackSelectLeftMenuEvent(position));

}else{

if(mLeftMenuPositionStack.size() == 1 && mLeftMenuPositionStack.get(0) == Constants.LEFT\_MENU\_POSITION\_0){

if(!isClickBackToFinish){

isClickBackToFinish = true;

Toast.makeText(mContext,"再按一次退出",Toast.LENGTH\_SHORT).show();

mHandler.sendEmptyMessageDelayed(MSG\_CANCEL\_CLICK\_BACK\_TO\_FINISH,1500);

}else {

super.onBackPressed();

}

}else{

mLeftMenuPositionStack.clear();

mLeftMenuPositionStack.push(Constants.LEFT\_MENU\_POSITION\_0);

showContentFragment(Constants.LEFT\_MENU\_POSITION\_0);

EventBusUtils.post(new BackSelectLeftMenuEvent(Constants.LEFT\_MENU\_POSITION\_0));

}

}

}

}

@Override

protected void onDestroy() {

super.onDestroy();

EventBusUtils.unregister(this);

mDrawerLayout.removeDrawerListener(mActionBarDrawerToggle);

}

@Subscribe(threadMode = ThreadMode.MAIN)

public void onRequestDoLocateEvent(RequestLocatePermissionEvent event){

requestLocatePermission();

}

@Subscribe(threadMode = ThreadMode.MAIN)

public void onSelectLeftMenuEvent(SelectLeftMenuEvent event){

if(event.mPosition == Constants.LEFT\_MENU\_POSITION\_0){

mLeftMenuPositionStack.clear();

}else{

int index = -1;

for(int i = 0; i < mLeftMenuPositionStack.size(); i++){

if(mLeftMenuPositionStack.get(i).intValue() == event.mPosition){

index = i;

break;

}

}

if(index > -1){

mLeftMenuPositionStack.remove(index);

}

}

mLeftMenuPositionStack.push(event.mPosition);

showContentFragment(event.mPosition);

mDrawerLayout.closeDrawer(mLeftMenuView,true);

}

}

package com.zxl.casual.living.fragment;

import android.graphics.Color;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.support.v7.widget.GridLayoutManager;

import android.support.v7.widget.RecyclerView;

import android.support.v7.widget.Toolbar;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.LinearLayout;

import android.widget.TextView;

import com.zxl.casual.living.R;

import com.zxl.casual.living.custom.view.TodayWeatherView;

import com.zxl.casual.living.event.BackSelectLeftMenuEvent;

import com.zxl.casual.living.event.LoginSuccessEvent;

import com.zxl.casual.living.event.LogoutSuccessEvent;

import com.zxl.casual.living.event.SelectLeftMenuEvent;

import com.zxl.casual.living.http.data.UserInfoResponseBean;

import com.zxl.casual.living.utils.Constants;

import com.zxl.casual.living.utils.EventBusUtils;

import com.zxl.casual.living.utils.SharePreUtils;

import com.zxl.common.DebugUtil;

import org.greenrobot.eventbus.Subscribe;

import org.greenrobot.eventbus.ThreadMode;

/\*\*

\* Created by zxl on 2018/9/20.

\*/

public class LeftMenuFragment extends BaseFragment {

private static final String TAG = "LeftMenuFragment";

private static final String[] MENU\_TITLE\_ARRAY = new String[]{"笑话","美女","收藏","账号","版本更新"};

private View mContentView;

private TodayWeatherView mTodayWeatherView;

private RecyclerView mRecyclerView;

private LeftMenuAdapter mLeftMenuAdapter;

private Toolbar mToolbar;

private int mSelectedPosition = 0;

@Nullable

@Override

public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

DebugUtil.d(TAG,"onCreateView");

EventBusUtils.register(this);

mContentView = inflater.inflate(R.layout.fragment\_left\_menu,null, false);

mTodayWeatherView = mContentView.findViewById(R.id.left\_menu\_today\_weather\_view);

mRecyclerView = mContentView.findViewById(R.id.left\_menu\_recycler\_view);

mLeftMenuAdapter = new LeftMenuAdapter();

GridLayoutManager gridLayoutManager = new GridLayoutManager(mActivity,2);

// gridLayoutManager.setOrientation(GridLayoutManager.VERTICAL);

mRecyclerView.setLayoutManager(gridLayoutManager);

mRecyclerView.setAdapter(mLeftMenuAdapter);

if(mToolbar != null){

mTodayWeatherView.setToolbar(mToolbar);

}

setSelectedPosition(0);

mContentView.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

}

});

return mContentView;

}

public void setToolbar(Toolbar toolbar){

DebugUtil.d(TAG,"setToolbar");

mToolbar = toolbar;

if(mTodayWeatherView != null){

mTodayWeatherView.setToolbar(toolbar);

}

}

public void setSelectedPosition(int position){

mSelectedPosition = position;

mLeftMenuAdapter.notifyDataSetChanged();

}

class LeftMenuAdapter extends RecyclerView.Adapter<LeftMenuViewHolder>{

@NonNull

@Override

public LeftMenuViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

View view = LayoutInflater.from(mActivity).inflate(R.layout.item\_left\_menu\_view,parent,false);

return new LeftMenuViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull LeftMenuViewHolder holder, final int position) {

holder.mItemLeftMenuTitleTv.setText(MENU\_TITLE\_ARRAY[position]);

if(position == mSelectedPosition){

holder.mItemLeftMenuContentLl.setSelected(true);

}else{

holder.mItemLeftMenuContentLl.setSelected(false);

}

UserInfoResponseBean userInfoResponseBean = SharePreUtils.getInstance(mActivity).getUserInfo();

if(userInfoResponseBean == null && position == Constants.LEFT\_MENU\_POSITION\_2){

holder.mItemView.setEnabled(false);

holder.mItemLeftMenuContentLl.setBackgroundColor(Color.parseColor("#BDBDBD"));

}else{

holder.mItemView.setEnabled(true);

holder.mItemLeftMenuContentLl.setBackgroundResource(R.drawable.s\_item\_left\_menu);

holder.mItemView.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

setSelectedPosition(position);

EventBusUtils.post(new SelectLeftMenuEvent(position));

}

});

}

}

@Override

public long getItemId(int position) {

return position;

}

@Override

public int getItemCount() {

return MENU\_TITLE\_ARRAY.length;

}

}

class LeftMenuViewHolder extends RecyclerView.ViewHolder{

public View mItemView;

public LinearLayout mItemLeftMenuContentLl;

public TextView mItemLeftMenuTitleTv;

public LeftMenuViewHolder(View itemView) {

super(itemView);

mItemView = itemView;

mItemLeftMenuContentLl = mItemView.findViewById(R.id.item\_left\_menu\_content\_ll);

mItemLeftMenuTitleTv = mItemView.findViewById(R.id.item\_left\_menu\_title\_tv);

}

}

@Override

public void onDestroyView() {

super.onDestroyView();

EventBusUtils.unregister(this);

}

@Subscribe(threadMode = ThreadMode.MAIN)

public void onBackSelectLeftMenuEvent(BackSelectLeftMenuEvent event){

setSelectedPosition(event.mPosition);

}

@Subscribe(threadMode = ThreadMode.MAIN)

public void onLoginSuccessEvent(LoginSuccessEvent event){

mLeftMenuAdapter.notifyDataSetChanged();

}

@Subscribe(threadMode = ThreadMode.MAIN)

public void onLogoutSuccessEvent(LogoutSuccessEvent event){

mLeftMenuAdapter.notifyDataSetChanged();

}

}

package com.zxl.casual.living.fragment;

import android.content.ClipboardManager;

import android.content.Context;

import android.content.Intent;

import android.graphics.Color;

import android.os.Bundle;

import android.os.Handler;

import android.os.Message;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.support.v4.widget.SwipeRefreshLayout;

import android.support.v7.widget.LinearLayoutManager;

import android.support.v7.widget.RecyclerView;

import android.text.TextUtils;

import android.view.KeyEvent;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.LinearLayout;

import android.widget.TextView;

import android.widget.Toast;

import com.bumptech.glide.Glide;

import com.zxl.casual.living.QSBKActivity;

import com.zxl.casual.living.QSBKDetailActivity;

import com.zxl.casual.living.R;

import com.zxl.casual.living.custom.view.CustomScaleView;

import com.zxl.casual.living.http.HttpUtils;

import com.zxl.casual.living.http.data.QSBKElement;

import com.zxl.casual.living.http.data.QSBKElementList;

import com.zxl.casual.living.http.data.ResponseBaseBean;

import com.zxl.casual.living.http.data.UserInfoResponseBean;

import com.zxl.casual.living.http.listener.NetRequestListener;

import com.zxl.casual.living.utils.CommonUtils;

import com.zxl.casual.living.utils.SharePreUtils;

import com.tencent.mm.opensdk.modelmsg.SendMessageToWX;

import com.zxl.common.DebugUtil;

import java.util.ArrayList;

import java.util.List;

import retrofit2.Call;

import retrofit2.http.GET;

import retrofit2.http.Query;

/\*\*

\* Created by zxl on 2018/9/20.

\*/

public class QSBKFragment extends BaseFragment {

private static final String TAG = "QSBKFragment";

private static final int MSG\_FIRST\_LOAD\_START = 1;

private static final int MSG\_FIRST\_LOAD\_SUCCESS = 2;

private static final int MSG\_FIRST\_LOAD\_ERROR = 3;

private static final int MSG\_LOAD\_START = 4;

private static final int MSG\_LOAD\_SUCCESS = 5;

private static final int MSG\_LOAD\_ERROR = 6;

private View mContentView;

private View mLoadingView;

private View mLoadErrorView;

private Button mBtnErrorRefresh;

private CustomScaleView mCustomScaleView;

private RecyclerView mRecyclerView;

private QSBKAdapter mQSBKAdapter;

private SwipeRefreshLayout mSwipeRefreshLayout;

// private Retrofit mRetrofit;

// private IQueryQSBK mIQueryQSBK;

private int mCurrentPage = 1;

private int mTotalPage = Integer.MAX\_VALUE;

private int mPageCount = 10;

private boolean isLoading = false;

private Handler mHandler = new Handler(){

@Override

public void handleMessage(Message msg) {

switch (msg.what){

case MSG\_FIRST\_LOAD\_START:

mRecyclerView.setVisibility(View.GONE);

mLoadingView.setVisibility(View.VISIBLE);

mLoadErrorView.setVisibility(View.GONE);

break;

case MSG\_FIRST\_LOAD\_SUCCESS:

mRecyclerView.setVisibility(View.VISIBLE);

mLoadingView.setVisibility(View.GONE);

mLoadErrorView.setVisibility(View.GONE);

List<QSBKElement> mFirstTemp = (List<QSBKElement>) msg.obj;

mQSBKAdapter.setData(mFirstTemp);

isLoading = false;

mSwipeRefreshLayout.setRefreshing(false);

break;

case MSG\_FIRST\_LOAD\_ERROR:

mRecyclerView.setVisibility(View.GONE);

mLoadingView.setVisibility(View.GONE);

mLoadErrorView.setVisibility(View.VISIBLE);

isLoading = false;

mSwipeRefreshLayout.setRefreshing(false);

break;

case MSG\_LOAD\_START:

mRecyclerView.setVisibility(View.VISIBLE);

mLoadingView.setVisibility(View.GONE);

mLoadErrorView.setVisibility(View.GONE);

break;

case MSG\_LOAD\_SUCCESS:

mRecyclerView.setVisibility(View.VISIBLE);

mLoadingView.setVisibility(View.GONE);

mLoadErrorView.setVisibility(View.GONE);

List<QSBKElement> mTemp = (List<QSBKElement>) msg.obj;

mQSBKAdapter.addData(mTemp);

isLoading = false;

mSwipeRefreshLayout.setRefreshing(false);

break;

case MSG\_LOAD\_ERROR:

mRecyclerView.setVisibility(View.VISIBLE);

mLoadingView.setVisibility(View.GONE);

mLoadErrorView.setVisibility(View.GONE);

mQSBKAdapter.setLoadDataState(QSBKActivity.CalculateAdapter.LOAD\_DATA\_ERROR\_STATE);

isLoading = false;

mSwipeRefreshLayout.setRefreshing(false);

break;

}

}

};

@Nullable

@Override

public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

mContentView = inflater.inflate(R.layout.fragment\_qsbk,null);

mLoadingView = mContentView.findViewById(R.id.qsbk\_loading\_view);

mLoadErrorView = mContentView.findViewById(R.id.qsbk\_load\_error\_view);

mBtnErrorRefresh = mLoadErrorView.findViewById(R.id.load\_error\_btn);

mBtnErrorRefresh.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

loadData(true,1);

}

});

mCustomScaleView = mContentView.findViewById(R.id.custom\_scale\_img);

mCustomScaleView.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

mCustomScaleView.setVisibility(View.GONE);

}

});

mRecyclerView = mContentView.findViewById(R.id.recycler\_view);

LinearLayoutManager mLinearLayoutManager = new LinearLayoutManager(mActivity);

mRecyclerView.setLayoutManager(mLinearLayoutManager);

mQSBKAdapter = new QSBKAdapter();

mRecyclerView.setAdapter(mQSBKAdapter);

mSwipeRefreshLayout = mContentView.findViewById(R.id.swipe\_refresh\_layout);

mSwipeRefreshLayout.setColorSchemeColors(Color.parseColor("#3F51B5"),Color.parseColor("#303F9F"),Color.parseColor("#FF4081"));

mSwipeRefreshLayout.setRefreshing(false);

mSwipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {

@Override

public void onRefresh() {

mSwipeRefreshLayout.setRefreshing(true);

loadData(true,1);

}

});

// OkHttpClient mOkHttpClient = new OkHttpClient.Builder().build();

// mRetrofit = new Retrofit.Builder()

// //.baseUrl("http://www.zxltest.cn/")

// .baseUrl("http://118.25.178.69/")

// .client(mOkHttpClient)

// .addConverterFactory(GsonConverterFactory.create())

// .build();

// mIQueryQSBK = mRetrofit.create(IQueryQSBK.class);

return mContentView;

}

@Override

public void onActivityCreated(@Nullable Bundle savedInstanceState) {

super.onActivityCreated(savedInstanceState);

loadData(true, 1);

}

public boolean onBackPressed(){

if(mCustomScaleView.getVisibility() == View.VISIBLE){

mCustomScaleView.setVisibility(View.GONE);

return true;

}

return false;

}

public void loadData(final boolean isFirstLoad, final int page){

if(isLoading){

return;

}

isLoading = true;

if(isFirstLoad){

mHandler.sendEmptyMessage(MSG\_FIRST\_LOAD\_START);

}else{

mHandler.sendEmptyMessage(MSG\_LOAD\_START);

}

UserInfoResponseBean userInfoResponseBean = SharePreUtils.getInstance(mActivity).getUserInfo();

HttpUtils.getInstance().getQSBK(mActivity, page, (userInfoResponseBean != null ? userInfoResponseBean.user\_id : ""),new NetRequestListener() {

@Override

public void onSuccess(ResponseBaseBean responseBaseBean) {

QSBKElementList mQSBKElementList = (QSBKElementList) responseBaseBean;

mCurrentPage = mQSBKElementList.current\_page;

if(isFirstLoad){

Message message = mHandler.obtainMessage();

message.what = MSG\_FIRST\_LOAD\_SUCCESS;

message.obj = mQSBKElementList.result;

message.sendToTarget();

}else{

Message message = mHandler.obtainMessage();

message.what = MSG\_LOAD\_SUCCESS;

message.obj = mQSBKElementList.result;

message.sendToTarget();

}

}

@Override

public void onNetError() {

if(isFirstLoad){

mHandler.sendEmptyMessage(MSG\_FIRST\_LOAD\_ERROR);

}else{

mHandler.sendEmptyMessage(MSG\_LOAD\_ERROR);

}

}

@Override

public void onNetError(Throwable e) {

if(isFirstLoad){

mHandler.sendEmptyMessage(MSG\_FIRST\_LOAD\_ERROR);

}else{

mHandler.sendEmptyMessage(MSG\_LOAD\_ERROR);

}

}

@Override

public void onServerError(ResponseBaseBean responseBaseBean) {

if(isFirstLoad){

mHandler.sendEmptyMessage(MSG\_FIRST\_LOAD\_ERROR);

}else{

mHandler.sendEmptyMessage(MSG\_LOAD\_ERROR);

}

}

});

}

public void doForCollect(final int operator, final int position, final QSBKElement qsbkElement){

UserInfoResponseBean userInfoResponseBean = SharePreUtils.getInstance(mActivity).getUserInfo();

if(userInfoResponseBean != null){

HttpUtils.getInstance().collectQSBK(mActivity, operator, userInfoResponseBean.user\_id, CommonUtils.mGson.toJson(qsbkElement), new NetRequestListener() {

@Override

public void onSuccess(ResponseBaseBean responseBaseBean) {

if(QSBKElement.QSBK\_COLLECT\_OPERATOR\_COLLECT == operator){

if(position < mQSBKAdapter.getData().size() && TextUtils.equals(mQSBKAdapter.getData().get(position).author\_id, qsbkElement.author\_id)){

mQSBKAdapter.getData().get(position).is\_collect = true;

mQSBKAdapter.notifyItemChanged(position);

}

Toast.makeText(mActivity,"已收藏",Toast.LENGTH\_SHORT).show();

}

if(QSBKElement.QSBK\_COLLECT\_OPERATOR\_CANCEL == operator){

if(position < mQSBKAdapter.getData().size() && TextUtils.equals(mQSBKAdapter.getData().get(position).author\_id, qsbkElement.author\_id)){

mQSBKAdapter.getData().get(position).is\_collect = false;

mQSBKAdapter.notifyItemChanged(position);

}

Toast.makeText(mActivity,"已取消收藏",Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onNetError() {

Toast.makeText(mActivity,mActivity.getResources().getString(R.string.no\_network\_tip),Toast.LENGTH\_SHORT).show();

}

@Override

public void onNetError(Throwable e) {

Toast.makeText(mActivity,mActivity.getResources().getString(R.string.network\_error\_tip),Toast.LENGTH\_SHORT).show();

}

@Override

public void onServerError(ResponseBaseBean responseBaseBean) {

Toast.makeText(mActivity,responseBaseBean.desc,Toast.LENGTH\_SHORT).show();

}

});

}else{

Toast.makeText(mActivity,"请先登录",Toast.LENGTH\_SHORT).show();

}

}

public class QSBKAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder>{

public static final int LOADING\_DATA\_STATE = 1;

public static final int LOAD\_DATA\_SUCCESS\_STATE =2 ;

public static final int LOAD\_DATA\_ERROR\_STATE = 3;

private static final int CONTENT\_TYPE = 1;

private static final int FOOT\_TYPE = 2;

private int mCurrentState = LOAD\_DATA\_SUCCESS\_STATE;

private List<QSBKElement> mQSBKElements = new ArrayList<>();

public void setData(List<QSBKElement> elements){

mQSBKElements.clear();

mQSBKElements.addAll(elements);

mCurrentState = LOAD\_DATA\_SUCCESS\_STATE;

notifyDataSetChanged();

}

public void addData(List<QSBKElement> elements){

mQSBKElements.addAll(elements);

mCurrentState = LOAD\_DATA\_SUCCESS\_STATE;

notifyDataSetChanged();

}

public List<QSBKElement> getData(){

return mQSBKElements;

}

public void setLoadDataState(int state){

mCurrentState = state;

notifyDataSetChanged();

}

@Override

public int getItemViewType(int position) {

if(position == getItemCount() - 1 && mCurrentPage < mTotalPage - 1){

return FOOT\_TYPE;

}

return CONTENT\_TYPE;

}

@NonNull

@Override

public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

switch (viewType){

case CONTENT\_TYPE:

View mItemView = LayoutInflater.from(mActivity).inflate(R.layout.item\_qsbk\_view, parent, false);

return new QSBKViewHolder(mItemView);

case FOOT\_TYPE:

View mItemFootView = LayoutInflater.from(mActivity).inflate(R.layout.item\_qsbk\_foot\_view, parent, false);

return new QSBKFootViewHolder(mItemFootView);

}

return null;

}

@Override

public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, final int position) {

QSBKViewHolder mQSBKViewHolder = null;

if(holder instanceof QSBKViewHolder){

mQSBKViewHolder = (QSBKViewHolder) holder;

}

QSBKFootViewHolder mQSBKFootViewHolder = null;

if(holder instanceof QSBKFootViewHolder){

mQSBKFootViewHolder = (QSBKFootViewHolder) holder;

}

if(position == getItemCount() - 1 && mCurrentPage < mTotalPage - 1 && mQSBKFootViewHolder != null){

switch (mCurrentState){

case LOAD\_DATA\_SUCCESS\_STATE:

mQSBKFootViewHolder.mLoadErrorView.setVisibility(View.GONE);

mQSBKFootViewHolder.mLoadingView.setVisibility(View.VISIBLE);

loadData(false,mCurrentPage + 1);

break;

case LOADING\_DATA\_STATE:

mQSBKFootViewHolder.mLoadErrorView.setVisibility(View.GONE);

mQSBKFootViewHolder.mLoadingView.setVisibility(View.VISIBLE);

break;

case LOAD\_DATA\_ERROR\_STATE:

mQSBKFootViewHolder.mLoadErrorView.setVisibility(View.VISIBLE);

mQSBKFootViewHolder.mLoadingView.setVisibility(View.GONE);

View mBtnErrorRefresh = mQSBKFootViewHolder.mLoadErrorView.findViewById(R.id.load\_error\_btn);

mBtnErrorRefresh.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

mQSBKAdapter.setLoadDataState(QSBKAdapter.LOADING\_DATA\_STATE);

loadData(false, mCurrentPage + 1);

}

});

break;

}

}else{

if(mQSBKViewHolder != null){

final QSBKElement mQSBKElement = mQSBKElements.get(position);

Glide.with(mActivity).load(mQSBKElement.author\_head\_img).into(mQSBKViewHolder.mAuthorHeadImg);

mQSBKViewHolder.mAuthorNameTv.setText(mQSBKElement.author\_name);

if(mQSBKElement.is\_collect){

mQSBKViewHolder.mCollectImg.setImageResource(R.mipmap.collect\_select\_icon);

}else{

mQSBKViewHolder.mCollectImg.setImageResource(R.mipmap.collect\_cancel\_icon);

}

if(mQSBKElement.isAnonymity()){

mQSBKViewHolder.mAuthorSexAgeLl.setVisibility(View.GONE);

}else{

mQSBKViewHolder.mAuthorSexAgeLl.setVisibility(View.VISIBLE);

if(mQSBKElement.author\_sex == QSBKElement.SEX\_MAN){

mQSBKViewHolder.mAuthorSexTv.setText("男");

mQSBKViewHolder.mAuthorSexTv.setTextColor(Color.parseColor("#0000ff"));

mQSBKViewHolder.mAuthorAgeTv.setTextColor(Color.parseColor("#0000ff"));

}else if(mQSBKElement.author\_sex == QSBKElement.SEX\_FEMALE){

mQSBKViewHolder.mAuthorSexTv.setText("女");

mQSBKViewHolder.mAuthorSexTv.setTextColor(Color.parseColor("#aa00aa"));

mQSBKViewHolder.mAuthorAgeTv.setTextColor(Color.parseColor("#aa00aa"));

}

mQSBKViewHolder.mAuthorAgeTv.setText(mQSBKElement.author\_age+"岁");

}

mQSBKViewHolder.mContentTv.setText(mQSBKElement.content);

if(mQSBKElement.hasThumb()){

mQSBKViewHolder.mThumbImg.setVisibility(View.VISIBLE);

Glide.with(mActivity).load(mQSBKElement.thumb).into(mQSBKViewHolder.mThumbImg);

mQSBKViewHolder.mThumbImg.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

mCustomScaleView.setUrl(mQSBKElement.thumb);

}

});

}else{

mQSBKViewHolder.mThumbImg.setVisibility(View.GONE);

}

mQSBKViewHolder.mVoteNumberTv.setText(String.valueOf(mQSBKElement.vote\_number));

mQSBKViewHolder.mCommentNumberTv.setText(String.valueOf(mQSBKElement.comment\_number));

mQSBKViewHolder.mItemView.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String mQSBKElementStr = CommonUtils.mGson.toJson(mQSBKElement);

Intent mIntent = new Intent(mActivity, QSBKDetailActivity.class);

mIntent.putExtra(QSBKDetailActivity.EXTRA\_QSBK\_ELEMENT, mQSBKElementStr);

startActivity(mIntent);

}

});

mQSBKViewHolder.mContentTv.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

String mQSBKElementStr = CommonUtils.mGson.toJson(mQSBKElement);

Intent mIntent = new Intent(mActivity, QSBKDetailActivity.class);

mIntent.putExtra(QSBKDetailActivity.EXTRA\_QSBK\_ELEMENT, mQSBKElementStr);

startActivity(mIntent);

}

});

mQSBKViewHolder.mContentTv.setOnLongClickListener(new View.OnLongClickListener() {

@Override

public boolean onLongClick(View view) {

ClipboardManager cmb = (ClipboardManager) mActivity.getSystemService(Context.CLIPBOARD\_SERVICE);

cmb.setText(mQSBKElement.content); //将内容放入粘贴管理器,在别的地方长按选择"粘贴"即可

cmb.getText();//获取粘贴信息

Toast.makeText(mActivity,"复制成功",Toast.LENGTH\_SHORT).show();

return true;

}

});

mQSBKViewHolder.mShareWechatFriendImg.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if(mQSBKElement.hasThumb()){

CommonUtils.shareWXBitmap(mActivity,mQSBKElement.thumb,SendMessageToWX.Req.WXSceneTimeline);

}else{

CommonUtils.shareWXText(mQSBKElement.content,mActivity.getPackageName(),SendMessageToWX.Req.WXSceneTimeline);

}

}

});

mQSBKViewHolder.mShareWechatFriendsImg.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if(mQSBKElement.hasThumb()){

CommonUtils.shareWXBitmap(mActivity,mQSBKElement.thumb,SendMessageToWX.Req.WXSceneSession);

}else{

CommonUtils.shareWXText(mQSBKElement.content,mActivity.getPackageName(),SendMessageToWX.Req.WXSceneSession);

}

}

});

mQSBKViewHolder.mCollectImg.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if(!mQSBKElement.is\_collect){

doForCollect(QSBKElement.QSBK\_COLLECT\_OPERATOR\_COLLECT, position, mQSBKElement);

}else{

doForCollect(QSBKElement.QSBK\_COLLECT\_OPERATOR\_CANCEL, position, mQSBKElement);

}

}

});

}

}

}

@Override

public int getItemCount() {

return mQSBKElements.size() + (mCurrentPage < mTotalPage - 1 ? 1 : 0);

}

}

public class QSBKViewHolder extends RecyclerView.ViewHolder{

public View mItemView;

public ImageView mAuthorHeadImg;

public ImageView mThumbImg;

public TextView mAuthorNameTv;

public TextView mAuthorSexTv;

public TextView mAuthorAgeTv;

public TextView mContentTv;

public TextView mVoteNumberTv;

public TextView mCommentNumberTv;

public ImageView mShareWechatFriendImg;

public ImageView mShareWechatFriendsImg;

public ImageView mCollectImg;

public LinearLayout mAuthorSexAgeLl;

public QSBKViewHolder(View itemView) {

super(itemView);

mItemView = itemView;

mAuthorHeadImg = mItemView.findViewById(R.id.author\_head\_img);

mThumbImg = mItemView.findViewById(R.id.thumb\_img);

mAuthorNameTv = mItemView.findViewById(R.id.author\_name\_tv);

mAuthorSexTv = mItemView.findViewById(R.id.author\_sex\_tv);

mAuthorAgeTv = mItemView.findViewById(R.id.author\_age\_tv);

mContentTv = mItemView.findViewById(R.id.content\_tv);

mVoteNumberTv = mItemView.findViewById(R.id.vote\_number\_tv);

mCommentNumberTv = mItemView.findViewById(R.id.comment\_number\_tv);

mAuthorSexAgeLl = mItemView.findViewById(R.id.author\_sex\_age\_ll);

mShareWechatFriendImg = mItemView.findViewById(R.id.share\_wechat\_friend\_img);

mShareWechatFriendsImg = mItemView.findViewById(R.id.share\_wechat\_friends\_img);

mCollectImg = mItemView.findViewById(R.id.collect\_img);

}

}

public class QSBKFootViewHolder extends RecyclerView.ViewHolder{

private View mItemView;

private View mLoadingView;

private View mLoadErrorView;

public QSBKFootViewHolder(View itemView) {

super(itemView);

mItemView = itemView;

mLoadingView = mItemView.findViewById(R.id.loading\_view);

mLoadErrorView = mItemView.findViewById(R.id.load\_error\_view);

}

}

}

package com.zxl.casual.living.fragment;

import android.graphics.Color;

import android.os.Bundle;

import android.os.Handler;

import android.os.Message;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.support.v4.widget.SwipeRefreshLayout;

import android.support.v7.widget.LinearLayoutManager;

import android.support.v7.widget.RecyclerView;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.LinearLayout;

import android.widget.TextView;

import com.bumptech.glide.Glide;

import com.zxl.casual.living.R;

import com.zxl.casual.living.http.HttpUtils;

import com.zxl.casual.living.http.data.ResponseBaseBean;

import com.zxl.casual.living.http.data.TaoBaoAnchor;

import com.zxl.casual.living.http.data.TaoBaoAnchorListResponseBean;

import com.zxl.casual.living.http.listener.NetRequestListener;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* Created by zxl on 2018/9/20.

\*/

public class TaoBaoAnchorFragment extends BaseFragment {

private static final String TAG = "TaoBaoAnchorFragment";

private static final int MSG\_FIRST\_LOAD\_START = 1;

private static final int MSG\_FIRST\_LOAD\_SUCCESS = 2;

private static final int MSG\_FIRST\_LOAD\_ERROR = 3;

private static final int MSG\_LOAD\_START = 4;

private static final int MSG\_LOAD\_SUCCESS = 5;

private static final int MSG\_LOAD\_ERROR = 6;

private View mContentView;

private View mLoadingView;

private View mLoadErrorView;

private Button mBtnErrorRefresh;

private RecyclerView mRecyclerView;

private TaoBaoAnchorAdapter mTaoBaoAnchorAdapter;

private SwipeRefreshLayout mSwipeRefreshLayout;

private int mCurrentPage = 1;

private int mTotalPage = Integer.MAX\_VALUE;

private int mPageCount = 10;

private boolean isLoading = false;

private Handler mHandler = new Handler(){

@Override

public void handleMessage(Message msg) {

switch (msg.what){

case MSG\_FIRST\_LOAD\_START:

mRecyclerView.setVisibility(View.GONE);

mLoadingView.setVisibility(View.VISIBLE);

mLoadErrorView.setVisibility(View.GONE);

break;

case MSG\_FIRST\_LOAD\_SUCCESS:

mRecyclerView.setVisibility(View.VISIBLE);

mLoadingView.setVisibility(View.GONE);

mLoadErrorView.setVisibility(View.GONE);

List<TaoBaoAnchor> mFirstTemp = (List<TaoBaoAnchor>) msg.obj;

mTaoBaoAnchorAdapter.setData(mFirstTemp);

isLoading = false;

mSwipeRefreshLayout.setRefreshing(false);

break;

case MSG\_FIRST\_LOAD\_ERROR:

mRecyclerView.setVisibility(View.GONE);

mLoadingView.setVisibility(View.GONE);

mLoadErrorView.setVisibility(View.VISIBLE);

isLoading = false;

mSwipeRefreshLayout.setRefreshing(false);

break;

case MSG\_LOAD\_START:

mRecyclerView.setVisibility(View.VISIBLE);

mLoadingView.setVisibility(View.GONE);

mLoadErrorView.setVisibility(View.GONE);

break;

case MSG\_LOAD\_SUCCESS:

mRecyclerView.setVisibility(View.VISIBLE);

mLoadingView.setVisibility(View.GONE);

mLoadErrorView.setVisibility(View.GONE);

List<TaoBaoAnchor> mTemp = (List<TaoBaoAnchor>) msg.obj;

mTaoBaoAnchorAdapter.addData(mTemp);

isLoading = false;

mSwipeRefreshLayout.setRefreshing(false);

break;

case MSG\_LOAD\_ERROR:

mRecyclerView.setVisibility(View.VISIBLE);

mLoadingView.setVisibility(View.GONE);

mLoadErrorView.setVisibility(View.GONE);

mTaoBaoAnchorAdapter.setLoadDataState(TaoBaoAnchorAdapter.LOAD\_DATA\_ERROR\_STATE);

isLoading = false;

mSwipeRefreshLayout.setRefreshing(false);

break;

}

}

};

@Nullable

@Override

public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

mContentView = inflater.inflate(R.layout.fragment\_taobao\_anchor,null);

mLoadingView = mContentView.findViewById(R.id.taobao\_anchor\_loading\_view);

mLoadErrorView = mContentView.findViewById(R.id.taobao\_anchor\_error\_view);

mBtnErrorRefresh = mLoadErrorView.findViewById(R.id.load\_error\_btn);

mBtnErrorRefresh.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

loadData(true,1);

}

});

mRecyclerView = mContentView.findViewById(R.id.recycler\_view);

LinearLayoutManager mLinearLayoutManager = new LinearLayoutManager(mActivity);

mRecyclerView.setLayoutManager(mLinearLayoutManager);

mTaoBaoAnchorAdapter = new TaoBaoAnchorAdapter();

mRecyclerView.setAdapter(mTaoBaoAnchorAdapter);

mSwipeRefreshLayout = mContentView.findViewById(R.id.swipe\_refresh\_layout);

mSwipeRefreshLayout.setColorSchemeColors(Color.parseColor("#3F51B5"),Color.parseColor("#303F9F"),Color.parseColor("#FF4081"));

mSwipeRefreshLayout.setRefreshing(false);

mSwipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {

@Override

public void onRefresh() {

mSwipeRefreshLayout.setRefreshing(true);

loadData(true,1);

}

});

return mContentView;

}

@Override

public void onActivityCreated(@Nullable Bundle savedInstanceState) {

super.onActivityCreated(savedInstanceState);

loadData(true, 1);

}

public void loadData(final boolean isFirstLoad, final int page){

if(isLoading){

return;

}

isLoading = true;

if(isFirstLoad){

mHandler.sendEmptyMessage(MSG\_FIRST\_LOAD\_START);

}else{

mHandler.sendEmptyMessage(MSG\_LOAD\_START);

}

HttpUtils.getInstance().getTaoBaoAnchor(mActivity, page, new NetRequestListener() {

@Override

public void onSuccess(ResponseBaseBean responseBaseBean) {

TaoBaoAnchorListResponseBean taoBaoAnchorListResponseBean = (TaoBaoAnchorListResponseBean) responseBaseBean;

mCurrentPage = taoBaoAnchorListResponseBean.current\_page;

if(isFirstLoad){

Message message = mHandler.obtainMessage();

message.what = MSG\_FIRST\_LOAD\_SUCCESS;

message.obj = taoBaoAnchorListResponseBean.taobao\_anchor\_list;

message.sendToTarget();

}else{

Message message = mHandler.obtainMessage();

message.what = MSG\_LOAD\_SUCCESS;

message.obj = taoBaoAnchorListResponseBean.taobao\_anchor\_list;

message.sendToTarget();

}

}

@Override

public void onNetError() {

if(isFirstLoad){

mHandler.sendEmptyMessage(MSG\_FIRST\_LOAD\_ERROR);

}else{

mHandler.sendEmptyMessage(MSG\_LOAD\_ERROR);

}

}

@Override

public void onNetError(Throwable e) {

if(isFirstLoad){

mHandler.sendEmptyMessage(MSG\_FIRST\_LOAD\_ERROR);

}else{

mHandler.sendEmptyMessage(MSG\_LOAD\_ERROR);

}

}

@Override

public void onServerError(ResponseBaseBean responseBaseBean) {

if(isFirstLoad){

mHandler.sendEmptyMessage(MSG\_FIRST\_LOAD\_ERROR);

}else{

mHandler.sendEmptyMessage(MSG\_LOAD\_ERROR);

}

}

});

}

public class TaoBaoAnchorAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder>{

public static final int LOADING\_DATA\_STATE = 1;

public static final int LOAD\_DATA\_SUCCESS\_STATE =2 ;

public static final int LOAD\_DATA\_ERROR\_STATE = 3;

private static final int CONTENT\_TYPE = 1;

private static final int FOOT\_TYPE = 2;

private int mCurrentState = LOAD\_DATA\_SUCCESS\_STATE;

private List<TaoBaoAnchor> mTaoBaoAnchors = new ArrayList<>();

public void setData(List<TaoBaoAnchor> elements){

mTaoBaoAnchors.clear();

mTaoBaoAnchors.addAll(elements);

mCurrentState = LOAD\_DATA\_SUCCESS\_STATE;

notifyDataSetChanged();

}

public void addData(List<TaoBaoAnchor> elements){

mTaoBaoAnchors.addAll(elements);

mCurrentState = LOAD\_DATA\_SUCCESS\_STATE;

notifyDataSetChanged();

}

public void setLoadDataState(int state){

mCurrentState = state;

notifyDataSetChanged();

}

@Override

public int getItemViewType(int position) {

if(position == getItemCount() - 1 && mCurrentPage < mTotalPage - 1){

return FOOT\_TYPE;

}

return CONTENT\_TYPE;

}

@NonNull

@Override

public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

switch (viewType){

case CONTENT\_TYPE:

View mItemView = LayoutInflater.from(mActivity).inflate(R.layout.item\_taobao\_anchor\_view, parent, false);

return new TaoBaoAnchorViewHolder(mItemView);

case FOOT\_TYPE:

View mItemFootView = LayoutInflater.from(mActivity).inflate(R.layout.item\_taobao\_anchor\_foot\_view, parent, false);

return new TaoBaoAnchorFootViewHolder(mItemFootView);

}

return null;

}

@Override

public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, int position) {

TaoBaoAnchorViewHolder mTaoBaoAnchorViewHolder = null;

if(holder instanceof TaoBaoAnchorViewHolder){

mTaoBaoAnchorViewHolder = (TaoBaoAnchorViewHolder) holder;

Glide.with(mActivity).load(mTaoBaoAnchors.get(position).anchor\_img).into(mTaoBaoAnchorViewHolder.mItemTaobaoAnchorImg);

Glide.with(mActivity).load(mTaoBaoAnchors.get(position).anchor\_vflag).into(mTaoBaoAnchorViewHolder.mItemTaobaoAnchorVflag);

mTaoBaoAnchorViewHolder.mItemTaobaoAnchorName.setText(mTaoBaoAnchors.get(position).anchor\_name);

mTaoBaoAnchorViewHolder.mItemTaobaoAnchorFansCount.setText(mTaoBaoAnchors.get(position).fans\_count);

}

TaoBaoAnchorFootViewHolder mTaoBaoAnchorFootViewHolder = null;

if(holder instanceof TaoBaoAnchorFootViewHolder){

mTaoBaoAnchorFootViewHolder = (TaoBaoAnchorFootViewHolder) holder;

}

if(position == getItemCount() - 1 && mCurrentPage < mTotalPage - 1 && mTaoBaoAnchorFootViewHolder != null){

switch (mCurrentState){

case LOAD\_DATA\_SUCCESS\_STATE:

mTaoBaoAnchorFootViewHolder.mLoadErrorView.setVisibility(View.GONE);

mTaoBaoAnchorFootViewHolder.mLoadingView.setVisibility(View.VISIBLE);

loadData(false,mCurrentPage + 1);

break;

case LOADING\_DATA\_STATE:

mTaoBaoAnchorFootViewHolder.mLoadErrorView.setVisibility(View.GONE);

mTaoBaoAnchorFootViewHolder.mLoadingView.setVisibility(View.VISIBLE);

break;

case LOAD\_DATA\_ERROR\_STATE:

mTaoBaoAnchorFootViewHolder.mLoadErrorView.setVisibility(View.VISIBLE);

mTaoBaoAnchorFootViewHolder.mLoadingView.setVisibility(View.GONE);

View mBtnErrorRefresh = mTaoBaoAnchorFootViewHolder.mLoadErrorView.findViewById(R.id.load\_error\_btn);

mBtnErrorRefresh.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

mTaoBaoAnchorAdapter.setLoadDataState(TaoBaoAnchorAdapter.LOADING\_DATA\_STATE);

loadData(false, mCurrentPage + 1);

}

});

break;

}

}else{

if(mTaoBaoAnchorViewHolder != null){

final TaoBaoAnchor mTaoBaoAnchor = mTaoBaoAnchors.get(position);

}

}

}

@Override

public int getItemCount() {

return mTaoBaoAnchors.size() + (mCurrentPage < mTotalPage - 1 ? 1 : 0);

}

}

public class TaoBaoAnchorViewHolder extends RecyclerView.ViewHolder{

public View mItemView;

public ImageView mItemTaobaoAnchorImg;

public ImageView mItemTaobaoAnchorVflag;

public TextView mItemTaobaoAnchorName;

public TextView mItemTaobaoAnchorFansCount;

public TaoBaoAnchorViewHolder(View itemView) {

super(itemView);

mItemView = itemView;

mItemTaobaoAnchorImg = mItemView.findViewById(R.id.item\_taobao\_anchor\_img);

mItemTaobaoAnchorVflag = mItemView.findViewById(R.id.item\_taobao\_anchor\_vflag);

mItemTaobaoAnchorName = mItemView.findViewById(R.id.item\_taobao\_anchor\_name);

mItemTaobaoAnchorFansCount = mItemView.findViewById(R.id.item\_taobao\_anchor\_fans\_count);

}

}

public class TaoBaoAnchorFootViewHolder extends RecyclerView.ViewHolder{

private View mItemView;

private View mLoadingView;

private View mLoadErrorView;

public TaoBaoAnchorFootViewHolder(View itemView) {

super(itemView);

mItemView = itemView;

mLoadingView = mItemView.findViewById(R.id.loading\_view);

mLoadErrorView = mItemView.findViewById(R.id.load\_error\_view);

}

}

}

package com.zxl.casual.living.fragment;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.support.design.widget.TextInputEditText;

import android.support.design.widget.TextInputLayout;

import android.support.v7.widget.CardView;

import android.text.TextUtils;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Toast;

import com.zxl.casual.living.R;

import com.zxl.casual.living.event.LoginSuccessEvent;

import com.zxl.casual.living.event.LogoutSuccessEvent;

import com.zxl.casual.living.http.HttpUtils;

import com.zxl.casual.living.http.data.ResponseBaseBean;

import com.zxl.casual.living.http.data.UserInfoResponseBean;

import com.zxl.casual.living.http.listener.NetRequestListener;

import com.zxl.casual.living.utils.CommonUtils;

import com.zxl.casual.living.utils.EventBusUtils;

import com.zxl.casual.living.utils.SharePreUtils;

import com.zxl.common.DebugUtil;

import java.util.regex.Pattern;

/\*\*

\* Created by zxl on 2018/9/21.

\*/

public class AccountFragment extends BaseFragment {

private static final String TAG = "AccountFragment";

private static final int CLICK\_UNKNOWN\_STATE = 0;

private static final int CLICK\_REGISTER\_STATE = 1;

private static final int CLICK\_LOGIN\_STATE = 2;

private static final int LOGIN\_SUCCESS\_STATE = 3;

private View mContentView;

private View mLoadingView;

private View mLoadErrorView;

private TextInputLayout mUserNameTextInputLayout;

private TextInputEditText mUserNameTextInputEditText;

private TextInputLayout mPassWordTextInputLayout;

private TextInputEditText mPassWordTextInputEditText;

private TextInputLayout mPhoneNumberTextInputLayout;

private TextInputEditText mPhoneNumberTextInputEditText;

private TextInputLayout mNickNameTextInputLayout;

private TextInputEditText mNickNameTextInputEditText;

private CardView mRegisterCardView;

private CardView mLoginCardView;

private CardView mLogoutCardView;

private CardView mCancelCardView;

private int mClickState = CLICK\_UNKNOWN\_STATE;

private boolean isRegistering = false;

private boolean isLogining = false;

private View.OnClickListener mOnClickListener = new View.OnClickListener() {

@Override

public void onClick(View v) {

switch (v.getId()){

case R.id.register\_card\_view:

if(mClickState == CLICK\_REGISTER\_STATE){

register();

}else{

mClickState = CLICK\_REGISTER\_STATE;

}

break;

case R.id.login\_card\_view:

if(mClickState == CLICK\_LOGIN\_STATE){

login();

}else{

mClickState = CLICK\_LOGIN\_STATE;

}

break;

case R.id.logout\_card\_view:

mClickState = CLICK\_UNKNOWN\_STATE;

SharePreUtils.getInstance(mActivity).saveUserInfo(null);

EventBusUtils.post(new LogoutSuccessEvent());

break;

case R.id.cancel\_card\_view:

mClickState = CLICK\_UNKNOWN\_STATE;

break;

}

doForkState(mClickState);

}

};

@Nullable

@Override

public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

DebugUtil.d(TAG,"onCreateView");

mContentView = inflater.inflate(R.layout.fragment\_account,null);

mLoadingView = mContentView.findViewById(R.id.loading\_view);

mLoadErrorView = mContentView.findViewById(R.id.load\_error\_view);

mUserNameTextInputLayout = mContentView.findViewById(R.id.user\_name\_input\_l);

mUserNameTextInputEditText = mContentView.findViewById(R.id.user\_name\_input\_et);

mPassWordTextInputLayout = mContentView.findViewById(R.id.pass\_word\_input\_l);

mPassWordTextInputEditText = mContentView.findViewById(R.id.pass\_word\_input\_et);

mPassWordTextInputLayout.setPasswordVisibilityToggleEnabled(true);

mPhoneNumberTextInputLayout = mContentView.findViewById(R.id.phone\_number\_input\_l);

mPhoneNumberTextInputEditText = mContentView.findViewById(R.id.phone\_number\_input\_et);

mNickNameTextInputLayout = mContentView.findViewById(R.id.nick\_name\_input\_l);

mNickNameTextInputEditText = mContentView.findViewById(R.id.nick\_name\_input\_et);

mRegisterCardView = mContentView.findViewById(R.id.register\_card\_view);

mLoginCardView = mContentView.findViewById(R.id.login\_card\_view);

mLogoutCardView = mContentView.findViewById(R.id.logout\_card\_view);

mCancelCardView = mContentView.findViewById(R.id.cancel\_card\_view);

mRegisterCardView.setOnClickListener(mOnClickListener);

mLoginCardView.setOnClickListener(mOnClickListener);

mLogoutCardView.setOnClickListener(mOnClickListener);

mCancelCardView.setOnClickListener(mOnClickListener);

return mContentView;

}

@Override

public void onActivityCreated(@Nullable Bundle savedInstanceState) {

super.onActivityCreated(savedInstanceState);

UserInfoResponseBean userInfoResponseBean = SharePreUtils.getInstance(mActivity).getUserInfo();

if(userInfoResponseBean != null){

doForkState(LOGIN\_SUCCESS\_STATE);

}else{

mRegisterCardView.setVisibility(View.VISIBLE);

mLoginCardView.setVisibility(View.VISIBLE);

mLogoutCardView.setVisibility(View.GONE);

}

}

public void doForkState(int state){

switch (state){

case CLICK\_UNKNOWN\_STATE:

mUserNameTextInputLayout.setVisibility(View.GONE);

mPassWordTextInputLayout.setVisibility(View.GONE);

mPhoneNumberTextInputLayout.setVisibility(View.GONE);

mNickNameTextInputLayout.setVisibility(View.GONE);

initInputContent(state);

mRegisterCardView.setVisibility(View.VISIBLE);

mLoginCardView.setVisibility(View.VISIBLE);

mLogoutCardView.setVisibility(View.GONE);

mCancelCardView.setVisibility(View.GONE);

break;

case CLICK\_REGISTER\_STATE:

mUserNameTextInputLayout.setVisibility(View.VISIBLE);

mPassWordTextInputLayout.setVisibility(View.VISIBLE);

mPhoneNumberTextInputLayout.setVisibility(View.VISIBLE);

mNickNameTextInputLayout.setVisibility(View.VISIBLE);

initInputContent(state);

mRegisterCardView.setVisibility(View.VISIBLE);

mLoginCardView.setVisibility(View.GONE);

mLogoutCardView.setVisibility(View.GONE);

mCancelCardView.setVisibility(View.VISIBLE);

break;

case CLICK\_LOGIN\_STATE:

mUserNameTextInputLayout.setVisibility(View.VISIBLE);

mPassWordTextInputLayout.setVisibility(View.VISIBLE);

mPhoneNumberTextInputLayout.setVisibility(View.GONE);

mNickNameTextInputLayout.setVisibility(View.GONE);

mRegisterCardView.setVisibility(View.GONE);

mLoginCardView.setVisibility(View.VISIBLE);

mLogoutCardView.setVisibility(View.GONE);

mCancelCardView.setVisibility(View.VISIBLE);

break;

case LOGIN\_SUCCESS\_STATE:

mUserNameTextInputLayout.setVisibility(View.VISIBLE);

mPassWordTextInputLayout.setVisibility(View.GONE);

mPhoneNumberTextInputLayout.setVisibility(View.VISIBLE);

mNickNameTextInputLayout.setVisibility(View.VISIBLE);

initInputContent(state);

mRegisterCardView.setVisibility(View.GONE);

mLoginCardView.setVisibility(View.GONE);

mLogoutCardView.setVisibility(View.VISIBLE);

mCancelCardView.setVisibility(View.GONE);

break;

}

}

package com.zxl.casual.living.receiver;

import android.app.DownloadManager;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.text.TextUtils;

import com.zxl.casual.living.utils.DownloadUtils;

import com.zxl.casual.living.utils.SharePreUtils;

import com.zxl.common.DebugUtil;

/\*\*

\* Created by zxl on 2018/10/9.

\*/

public class DownloadReceiver extends BroadcastReceiver {

private static final String TAG = "DownloadReceiver";

@Override

public void onReceive(Context context, Intent intent) {

DebugUtil.d(TAG , "onReceive::action = " + intent.getAction());

if (TextUtils.equals(intent.getAction(), DownloadManager.ACTION\_DOWNLOAD\_COMPLETE)) {

long id = intent.getLongExtra(DownloadManager.EXTRA\_DOWNLOAD\_ID, -1);

if (SharePreUtils.getInstance(context).getDownloadId() == id) {

DownloadUtils.installApk(context, id);

}

} else if (TextUtils.equals(intent.getAction(), DownloadManager.ACTION\_NOTIFICATION\_CLICKED)) {

// DownloadManager manager = (DownloadManager) context.getSystemService(Context.DOWNLOAD\_SERVICE);

//获取所有下载任务Ids组

//long[] ids = intent.getLongArrayExtra(DownloadManager.EXTRA\_NOTIFICATION\_CLICK\_DOWNLOAD\_IDS);

////点击通知栏取消所有下载

//manager.remove(ids);

//Toast.makeText(context, "下载任务已取消", Toast.LENGTH\_SHORT).show();

//处理 如果还未完成下载，用户点击Notification ，跳转到下载中心

Intent viewDownloadIntent = new Intent(DownloadManager.ACTION\_VIEW\_DOWNLOADS);

viewDownloadIntent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

context.startActivity(viewDownloadIntent);

}

}

}

package com.zxl.casual.living.http;

import com.zxl.casual.living.http.data.CityInfoListResponseBean;

import com.zxl.casual.living.http.data.QSBKElementList;

import com.zxl.casual.living.http.data.ResponseBaseBean;

import com.zxl.casual.living.http.data.TaoBaoAnchorListResponseBean;

import com.zxl.casual.living.http.data.TodayWeatherResponseBean;

import com.zxl.casual.living.http.data.UpdateInfoResponseBean;

import com.zxl.casual.living.http.data.UserInfoResponseBean;

import okhttp3.ResponseBody;

import retrofit2.Call;

import retrofit2.http.GET;

import retrofit2.http.Query;

import rx.Observable;

package com.zxl.casual.living.custom.view;

import android.Manifest;

import android.content.Context;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.graphics.Color;

import android.location.Address;

import android.location.Geocoder;

import android.location.Location;

import android.location.LocationListener;

import android.location.LocationManager;

import android.os.Bundle;

import android.support.v7.widget.Toolbar;

import android.text.TextUtils;

import android.util.AttributeSet;

import android.view.LayoutInflater;

import android.view.View;

import android.widget.Button;

import android.widget.LinearLayout;

import android.widget.TextView;

import com.baidu.location.BDAbstractLocationListener;

import com.baidu.location.BDLocation;

import com.baidu.location.LocationClient;

import com.baidu.location.LocationClientOption;

import com.zxl.casual.living.CityInfoListActivity;

import com.zxl.casual.living.R;

import com.zxl.casual.living.event.LocatePermissionSuccessEvent;

import com.zxl.casual.living.event.RequestLocatePermissionEvent;

import com.zxl.casual.living.event.SelectCityEvent;

import com.zxl.casual.living.http.HttpUtils;

import com.zxl.casual.living.http.data.ResponseBaseBean;

import com.zxl.casual.living.http.data.TodayWeatherResponseBean;

import com.zxl.casual.living.http.listener.NetRequestListener;

import com.zxl.casual.living.utils.EventBusUtils;

import com.zxl.common.DebugUtil;

import org.greenrobot.eventbus.Subscribe;

import org.greenrobot.eventbus.ThreadMode;

import java.io.IOException;

import java.util.List;

import java.util.Locale;

/\*\*

\* Created by zxl on 2018/9/5.

\*/

public class TodayWeatherView extends CardView {

private static final String TAG = "TodayWeatherView";

private Context mContext;

private LocationManager mLocationManager;

private LocationListener mLocationListener = new LocationListener() {

@Override

public void onLocationChanged(Location location) {

DebugUtil.d(TAG, "onLocationChanged::isNeedLoadData = " + isNeedLoadData + "::location = " + location);

if (isNeedLoadData) {

mLocationInfo = location.getLatitude() + "," + location.getLongitude();

// getDataFromNetByLocation(mLocationInfo);

}

}

@Override

public void onStatusChanged(String provider, int status, Bundle extras) {

}

@Override

public void onProviderEnabled(String provider) {

}

@Override

public void onProviderDisabled(String provider) {

}

};

private LocationClient mLocationClient = null;

private BDAbstractLocationListener mBdAbstractLocationListener = new BDAbstractLocationListener() {

@Override

public void onReceiveLocation(BDLocation bdLocation) {

DebugUtil.d(TAG,"mBdAbstractLocationListener::onReceiveLocation::bdLocation = " + bdLocation);

//此处的BDLocation为定位结果信息类，通过它的各种get方法可获取定位相关的全部结果

//以下只列举部分获取经纬度相关（常用）的结果信息

//更多结果信息获取说明，请参照类参考中BDLocation类中的说明

double latitude = bdLocation.getLatitude(); //获取纬度信息

double longitude = bdLocation.getLongitude(); //获取经度信息

float radius = bdLocation.getRadius(); //获取定位精度，默认值为0.0f

DebugUtil.d(TAG,"mBdAbstractLocationListener::onReceiveLocation::latitude = " + latitude + "::longitude = " + longitude);

String coorType = bdLocation.getCoorType();

//获取经纬度坐标类型，以LocationClientOption中设置过的坐标类型为准

int errorCode = bdLocation.getLocType();

//获取定位类型、定位错误返回码，具体信息可参照类参考中BDLocation类中的说明

String addr = bdLocation.getAddrStr(); //获取详细地址信息

String country = bdLocation.getCountry(); //获取国家

String province = bdLocation.getProvince(); //获取省份

String city = bdLocation.getCity(); //获取城市

String district = bdLocation.getDistrict(); //获取区县

String street = bdLocation.getStreet(); //获取街道信息

DebugUtil.d(TAG,"mBdAbstractLocationListener::onReceiveLocation::addr = " + addr + "::city = " + city);

mLocationClient.stop();

HttpUtils.getInstance().getZHTianQiByCity(mContext, city , new NetRequestListener() {

@Override

public void onSuccess(ResponseBaseBean responseBaseBean) {

DebugUtil.d(TAG,"onSuccess::responseBaseBean = " + responseBaseBean);

TodayWeatherResponseBean todayWeatherResponseBean = (TodayWeatherResponseBean) responseBaseBean;

mLoadingView.setVisibility(GONE);

mLoadErrorView.setVisibility(GONE);

mTodayWeatherContentView.setVisibility(VISIBLE);

setDataToView(todayWeatherResponseBean);

isLoading = false;

isNeedLoadData = false;

}

@Override

public void onNetError() {

DebugUtil.d(TAG,"onNetError");

mLoadingView.setVisibility(GONE);

mLoadErrorView.setVisibility(VISIBLE);

mTodayWeatherContentView.setVisibility(INVISIBLE);

mLoadErrorTv.setText(R.string.no\_network\_tip);

isLoading = false;

}

@Override

public void onNetError(Throwable e) {

DebugUtil.d(TAG,"onNetError::e = " + e);

mLoadingView.setVisibility(GONE);

mLoadErrorView.setVisibility(VISIBLE);

mTodayWeatherContentView.setVisibility(INVISIBLE);

mLoadErrorTv.setText(mContext.getResources().getString(R.string.network\_error\_tip, ""));

isLoading = false;

}

@Override

public void onServerError(ResponseBaseBean responseBaseBean) {

DebugUtil.d(TAG,"onServerError::responseBaseBean = " + responseBaseBean);

mLoadingView.setVisibility(GONE);

mLoadErrorView.setVisibility(VISIBLE);

mTodayWeatherContentView.setVisibility(INVISIBLE);

mLoadErrorTv.setText(mContext.getResources().getString(R.string.server\_error\_tip, responseBaseBean.desc));

isLoading = false;

}

});

}

};

private LocationClientOption mLocationClientOption = new LocationClientOption();

private String mLocationInfo = "";

private View mContentView;

private View mTodayWeatherContentView;

private View mLoadingView;

private View mLoadErrorView;

private TextView mLoadErrorTv;

private Button mLoadErrorBtn;

private LinearLayout mAddressInfoLl;

private TextView mAddressInfo;

private TextView mNowTimeTv;

private TodayWeatherTemperatureView mTodayWeatherTemperatureView;

private TextView mTemperatureTv;

private TodayWeatherHumidityIconView mTodayWeatherHumidityIconView;

private TextView mHumidityTv;

private TodayWeatherWindIconView mTodayWeatherWindIconView;

private TextView mWindTv;

private TodayWeatherAirQualityIconView mTodayWeatherAirQualityIconView;

private TextView mAirQualityTv;

private LinearLayout mLimitContentLl;

private TodayWeatherLimitIconView mTodayWeatherLimitIconView;

private TextView mLimitTv;

private TextView mTodayWeatherDetail1TitleTv;

private TodayWeatherDetailIconView mTodayWeatherDetail1IconView;

private TextView mTodayWeatherDetail1TemperatureTv;

private TextView mTodayWeatherDetail1WeatherTv;

private TextView mTodayWeatherDetail1WeatherDescTv;

private TodayWeatherDetailWindIconView mTodayWeatherDetail1WindIconView;

private TextView mTodayWeatherDetail1WindTv;

private TodayWeatherDetailSunIconView mTodayWeatherDetail1SunIconView;

private TextView mTodayWeatherDetail1SunTimeTv;

private TextView mTodayWeatherDetail2TitleTv;

private TodayWeatherDetailIconView mTodayWeatherDetail2IconView;

private TextView mTodayWeatherDetail2TemperatureTv;

private TextView mTodayWeatherDetail2WeatherTv;

private TextView mTodayWeatherDetail2WeatherDescTv;

private TodayWeatherDetailWindIconView mTodayWeatherDetail2WindIconView;

private TextView mTodayWeatherDetail2WindTv;

private TodayWeatherDetailSunIconView mTodayWeatherDetail2SunIconView;

private TextView mTodayWeatherDetail2SunTimeTv;

private Toolbar mToolbar;

private boolean isLoading = false;

private boolean isNeedLoadData = true;

public TodayWeatherView(@NonNull Context context) {

super(context);

init(context);

}

public TodayWeatherView(@NonNull Context context, @Nullable AttributeSet attrs) {

super(context, attrs);

init(context);

}

public TodayWeatherView(@NonNull Context context, @Nullable AttributeSet attrs, int defStyleAttr) {

super(context, attrs, defStyleAttr);

init(context);

}

private void init(Context context) {

DebugUtil.d(TAG, "init");

mContext = context;

mContentView = LayoutInflater.from(context).inflate(R.layout.today\_weather\_view, this);

mTodayWeatherContentView = mContentView.findViewById(R.id.today\_weather\_content\_view);

mLoadingView = mContentView.findViewById(R.id.today\_weather\_loading\_view);

mLoadErrorView = mContentView.findViewById(R.id.today\_weather\_load\_error\_view);

mLoadErrorTv = mLoadErrorView.findViewById(R.id.load\_error\_tv);

mLoadErrorBtn = mLoadErrorView.findViewById(R.id.load\_error\_btn);

mAddressInfoLl = mContentView.findViewById(R.id.address\_info\_ll);

mAddressInfo = mContentView.findViewById(R.id.address\_info);

mNowTimeTv = mContentView.findViewById(R.id.now\_time\_tv);

mTodayWeatherTemperatureView = mContentView.findViewById(R.id.today\_weather\_temperature\_view);

mTemperatureTv = mContentView.findViewById(R.id.temperature\_tv);

mTodayWeatherHumidityIconView = mContentView.findViewById(R.id.today\_weather\_humidity\_icon\_view);

mHumidityTv = mContentView.findViewById(R.id.humidity\_tv);

mTodayWeatherWindIconView = mContentView.findViewById(R.id.today\_weather\_wind\_icon\_view);

mWindTv = mContentView.findViewById(R.id.wind\_tv);

mTodayWeatherAirQualityIconView = mContentView.findViewById(R.id.today\_weather\_air\_quality\_icon\_view);

mAirQualityTv = mContentView.findViewById(R.id.air\_quality\_tv);

mLimitContentLl = mContentView.findViewById(R.id.limit\_content\_ll);

mTodayWeatherLimitIconView = mContentView.findViewById(R.id.today\_weather\_limit\_icon\_view);

mLimitTv = mContentView.findViewById(R.id.limit\_tv);

mTodayWeatherDetail1TitleTv = mContentView.findViewById(R.id.today\_weather\_detail\_1\_title\_tv);

mTodayWeatherDetail1IconView = mContentView.findViewById(R.id.today\_weather\_detail\_1\_icon\_view);

mTodayWeatherDetail1TemperatureTv = mContentView.findViewById(R.id.today\_weather\_detail\_1\_temperature\_tv);

mTodayWeatherDetail1WeatherTv = mContentView.findViewById(R.id.today\_weather\_detail\_1\_weather\_tv);

mTodayWeatherDetail1WeatherDescTv = mContentView.findViewById(R.id.today\_weather\_detail\_1\_weather\_desc\_tv);

mTodayWeatherDetail1WindIconView = mContentView.findViewById(R.id.today\_weather\_detail\_1\_wind\_icon\_view);

mTodayWeatherDetail1WindTv = mContentView.findViewById(R.id.today\_weather\_detail\_1\_wind\_tv);

mTodayWeatherDetail1SunIconView = mContentView.findViewById(R.id.today\_weather\_detail\_1\_sun\_icon\_view);

mTodayWeatherDetail1SunTimeTv = mContentView.findViewById(R.id.today\_weather\_detail\_1\_sun\_time\_tv);

mTodayWeatherDetail2TitleTv = mContentView.findViewById(R.id.today\_weather\_detail\_2\_title\_tv);

mTodayWeatherDetail2IconView = mContentView.findViewById(R.id.today\_weather\_detail\_2\_icon\_view);

mTodayWeatherDetail2TemperatureTv = mContentView.findViewById(R.id.today\_weather\_detail\_2\_temperature\_tv);

mTodayWeatherDetail2WeatherTv = mContentView.findViewById(R.id.today\_weather\_detail\_2\_weather\_tv);

mTodayWeatherDetail2WeatherDescTv = mContentView.findViewById(R.id.today\_weather\_detail\_2\_weather\_desc\_tv);

mTodayWeatherDetail2WindIconView = mContentView.findViewById(R.id.today\_weather\_detail\_2\_wind\_icon\_view);

mTodayWeatherDetail2WindTv = mContentView.findViewById(R.id.today\_weather\_detail\_2\_wind\_tv);

mTodayWeatherDetail2SunIconView = mContentView.findViewById(R.id.today\_weather\_detail\_2\_sun\_icon\_view);

mTodayWeatherDetail2SunTimeTv = mContentView.findViewById(R.id.today\_weather\_detail\_2\_sun\_time\_tv);

mLoadErrorBtn.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

//getDataFromNetByLocation(mLocationInfo);

doLocate();

}

});

mAddressInfoLl.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

Intent intent = new Intent(mContext, CityInfoListActivity.class);

intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

mContext.startActivity(intent);

}

});

boolean isNeedStartRequestPermissionActivity = false;

if (PackageManager.PERMISSION\_GRANTED != ContextCompat.checkSelfPermission(mContext, Manifest.permission.WRITE\_EXTERNAL\_STORAGE)

|| PackageManager.PERMISSION\_GRANTED != ContextCompat.checkSelfPermission(mContext, Manifest.permission.READ\_EXTERNAL\_STORAGE)

|| PackageManager.PERMISSION\_GRANTED != ContextCompat.checkSelfPermission(mContext, Manifest.permission.ACCESS\_COARSE\_LOCATION)

|| PackageManager.PERMISSION\_GRANTED != ContextCompat.checkSelfPermission(mContext, Manifest.permission.ACCESS\_FINE\_LOCATION)) {

isNeedStartRequestPermissionActivity = true;

}

DebugUtil.d(TAG,"init::isNeedStartRequestPermissionActivity = " + isNeedStartRequestPermissionActivity);

if (isNeedStartRequestPermissionActivity) {

EventBusUtils.post(new RequestLocatePermissionEvent());

} else {

doLocate();

// mLocationInfo = "31.950454,118.809312";

// getDataFromNetByLocation(mLocationInfo);

}

}

// private void getDataFromNetByLocation(String l) {

// DebugUtil.d(TAG,"getDataFromNetByLocation::isLoading = " + isLoading);

// if (isLoading) {

// return;

// }

// isLoading = true;

//

// mLoadingView.setVisibility(VISIBLE);

// mLoadErrorView.setVisibility(GONE);

// mTodayWeatherContentView.setVisibility(INVISIBLE);

//

// HttpUtils.getInstance().getZHTianQiByLocation(mContext, l, new NetRequestListener() {

// @Override

// public void onSuccess(ResponseBaseBean responseBaseBean) {

// DebugUtil.d(TAG,"onSuccess::responseBaseBean = " + responseBaseBean);

//

// TodayWeatherResponseBean todayWeatherResponseBean = (TodayWeatherResponseBean) responseBaseBean;

//

// mLoadingView.setVisibility(GONE);

// mLoadErrorView.setVisibility(GONE);

// mTodayWeatherContentView.setVisibility(VISIBLE);

//

// setDataToView(todayWeatherResponseBean);

//

// isLoading = false;

// isNeedLoadData = false;

// }

//

// @Override

// public void onNetError() {

// DebugUtil.d(TAG,"onNetError");

// mLoadingView.setVisibility(GONE);

// mLoadErrorView.setVisibility(VISIBLE);

// mTodayWeatherContentView.setVisibility(INVISIBLE);

// mLoadErrorTv.setText(R.string.no\_network\_tip);

//

// isLoading = false;

// }

//

// @Override

// public void onNetError(Throwable e) {

// DebugUtil.d(TAG,"onNetError::e = " + e);

// mLoadingView.setVisibility(GONE);

// mLoadErrorView.setVisibility(VISIBLE);

// mTodayWeatherContentView.setVisibility(INVISIBLE);

// mLoadErrorTv.setText(mContext.getResources().getString(R.string.network\_error\_tip, ""));

//

// isLoading = false;

// }

//

// @Override

// public void onServerError(ResponseBaseBean responseBaseBean) {

// DebugUtil.d(TAG,"onServerError::responseBaseBean = " + responseBaseBean);

// mLoadingView.setVisibility(GONE);

// mLoadErrorView.setVisibility(VISIBLE);

// mTodayWeatherContentView.setVisibility(INVISIBLE);

// mLoadErrorTv.setText(mContext.getResources().getString(R.string.server\_error\_tip, responseBaseBean.desc));

//

// isLoading = false;

// }

// });

// }

private void getDataFromNetByCity(final String city) {

DebugUtil.d(TAG,"getDataFromNetByCity::isLoading = " + isLoading);

if (isLoading) {

return;

}

isLoading = true;

mLoadingView.setVisibility(VISIBLE);

mLoadErrorView.setVisibility(GONE);

mTodayWeatherContentView.setVisibility(INVISIBLE);

HttpUtils.getInstance().getZHTianQiByCity(mContext, city, new NetRequestListener() {

@Override

public void onSuccess(ResponseBaseBean responseBaseBean) {

DebugUtil.d(TAG,"onSuccess::responseBaseBean = " + responseBaseBean);

TodayWeatherResponseBean todayWeatherResponseBean = (TodayWeatherResponseBean) responseBaseBean;

mLoadingView.setVisibility(GONE);

mLoadErrorView.setVisibility(GONE);

mTodayWeatherContentView.setVisibility(VISIBLE);

setDataToView(todayWeatherResponseBean);

isLoading = false;

isNeedLoadData = false;

}

@Override

public void onNetError() {

DebugUtil.d(TAG,"onNetError");

mLoadingView.setVisibility(GONE);

mLoadErrorView.setVisibility(VISIBLE);

mTodayWeatherContentView.setVisibility(INVISIBLE);

mLoadErrorTv.setText(R.string.no\_network\_tip);

setToolbarTitle("获取"+city+"天气失败");

isLoading = false;

}

@Override

public void onNetError(Throwable e) {

DebugUtil.d(TAG,"onNetError::e = " + e);

mLoadingView.setVisibility(GONE);

mLoadErrorView.setVisibility(VISIBLE);

mTodayWeatherContentView.setVisibility(INVISIBLE);

mLoadErrorTv.setText(mContext.getResources().getString(R.string.network\_error\_tip, ""));

setToolbarTitle("获取"+city+"天气失败");

isLoading = false;

}

@Override

public void onServerError(ResponseBaseBean responseBaseBean) {

DebugUtil.d(TAG,"onServerError::responseBaseBean = " + responseBaseBean);

mLoadingView.setVisibility(GONE);

mLoadErrorView.setVisibility(VISIBLE);

mTodayWeatherContentView.setVisibility(INVISIBLE);

mLoadErrorTv.setText(mContext.getResources().getString(R.string.server\_error\_tip, responseBaseBean.desc));

setToolbarTitle("获取"+city+"天气失败");

isLoading = false;

}

});

}

private void setDataToView(TodayWeatherResponseBean todayWeatherResponseBean) {

setToolbarTitle(todayWeatherResponseBean.today\_weather.simple\_content);

mAddressInfo.setText(todayWeatherResponseBean.address\_info);

mNowTimeTv.setText(todayWeatherResponseBean.today\_weather.now\_time);

mTodayWeatherTemperatureView.setTodayWeatherTemperatureIconCss(todayWeatherResponseBean.today\_weather.temperature\_icon\_css);

mTemperatureTv.setText(todayWeatherResponseBean.today\_weather.temperature + "°C");

if(todayWeatherResponseBean.today\_weather.is\_h == 1){

mHumidityTv.setVisibility(VISIBLE);

mTodayWeatherHumidityIconView.setVisibility(VISIBLE);

mHumidityTv.setText("相对湿度 " + todayWeatherResponseBean.today\_weather.humidity);

mTodayWeatherHumidityIconView.setTodayWeatherHumidityIconCss(todayWeatherResponseBean.today\_weather.humidity\_icon\_css);

}else {

mHumidityTv.setVisibility(INVISIBLE);

mTodayWeatherHumidityIconView.setVisibility(INVISIBLE);

}

if(todayWeatherResponseBean.today\_weather.is\_w == 1) {

mTodayWeatherWindIconView.setVisibility(VISIBLE);

mWindTv.setVisibility(VISIBLE);

mTodayWeatherWindIconView.setTodayWeatherHumidityIconCss(todayWeatherResponseBean.today\_weather.wind\_icon\_css);

mWindTv.setText(todayWeatherResponseBean.today\_weather.wind\_direction + " " + todayWeatherResponseBean.today\_weather.wind\_value);

}else{

mTodayWeatherWindIconView.setVisibility(INVISIBLE);

mWindTv.setVisibility(INVISIBLE);

}

if(todayWeatherResponseBean.today\_weather.is\_pol == 1) {

mTodayWeatherAirQualityIconView.setVisibility(VISIBLE);

mAirQualityTv.setVisibility(VISIBLE);

mTodayWeatherAirQualityIconView.setTodayWeatherHumidityIconCss(todayWeatherResponseBean.today\_weather.air\_quality\_icon\_css);

mAirQualityTv.setText(todayWeatherResponseBean.today\_weather.air\_quality);

}else{

mTodayWeatherAirQualityIconView.setVisibility(INVISIBLE);

mAirQualityTv.setVisibility(INVISIBLE);

}

if(todayWeatherResponseBean.today\_weather.is\_limit == 1){

mLimitContentLl.setVisibility(VISIBLE);

mTodayWeatherLimitIconView.setTodayWeatherHumidityIconCss(todayWeatherResponseBean.today\_weather.limit\_icon\_css);

mLimitTv.setText(todayWeatherResponseBean.today\_weather.limit\_content);

}else{

mLimitContentLl.setVisibility(INVISIBLE);

}

mTodayWeatherDetail1TitleTv.setText(todayWeatherResponseBean.today\_weather\_detail.get(0).title);

mTodayWeatherDetail1IconView.setTodayWeatherDetailIconCss(todayWeatherResponseBean.today\_weather\_detail.get(0).weather\_icon\_css);

mTodayWeatherDetail1TemperatureTv.setText(todayWeatherResponseBean.today\_weather\_detail.get(0).temperature + "°C");

mTodayWeatherDetail1WeatherTv.setText(todayWeatherResponseBean.today\_weather\_detail.get(0).weather);

String weatherDesc1 = todayWeatherResponseBean.today\_weather\_detail.get(0).weather\_desc;

if (!TextUtils.isEmpty(weatherDesc1)) {

mTodayWeatherDetail1WeatherDescTv.setVisibility(VISIBLE);

mTodayWeatherDetail1WeatherDescTv.setText(weatherDesc1);

} else {

mTodayWeatherDetail1WeatherDescTv.setVisibility(GONE);

}

mTodayWeatherDetail1WindIconView.setTodayWeatherHumidityIconCss(todayWeatherResponseBean.today\_weather\_detail.get(0).wind\_icon\_css);

mTodayWeatherDetail1WindTv.setText(todayWeatherResponseBean.today\_weather\_detail.get(0).wind\_direction + " " + todayWeatherResponseBean.today\_weather\_detail.get(0).wind\_value);

mTodayWeatherDetail1SunIconView.setTodayWeatherHumidityIconCss(todayWeatherResponseBean.today\_weather\_detail.get(0).sun\_icon\_css);

mTodayWeatherDetail1SunTimeTv.setText(todayWeatherResponseBean.today\_weather\_detail.get(0).sun\_time);

mTodayWeatherDetail2TitleTv.setText(todayWeatherResponseBean.today\_weather\_detail.get(1).title);

mTodayWeatherDetail2IconView.setTodayWeatherDetailIconCss(todayWeatherResponseBean.today\_weather\_detail.get(1).weather\_icon\_css);

mTodayWeatherDetail2TemperatureTv.setText(todayWeatherResponseBean.today\_weather\_detail.get(1).temperature + "°C");

mTodayWeatherDetail2WeatherTv.setText(todayWeatherResponseBean.today\_weather\_detail.get(1).weather);

String weatherDesc2 = todayWeatherResponseBean.today\_weather\_detail.get(1).weather\_desc;

if (!TextUtils.isEmpty(weatherDesc2)) {

mTodayWeatherDetail2WeatherDescTv.setVisibility(VISIBLE);

mTodayWeatherDetail2WeatherDescTv.setText(weatherDesc2);

} else {

mTodayWeatherDetail2WeatherDescTv.setVisibility(GONE);

}

mTodayWeatherDetail2WindIconView.setTodayWeatherHumidityIconCss(todayWeatherResponseBean.today\_weather\_detail.get(1).wind\_icon\_css);

mTodayWeatherDetail2WindTv.setText(todayWeatherResponseBean.today\_weather\_detail.get(1).wind\_direction + " " + todayWeatherResponseBean.today\_weather\_detail.get(1).wind\_value);

mTodayWeatherDetail2SunIconView.setTodayWeatherHumidityIconCss(todayWeatherResponseBean.today\_weather\_detail.get(1).sun\_icon\_css);

mTodayWeatherDetail2SunTimeTv.setText(todayWeatherResponseBean.today\_weather\_detail.get(1).sun\_time);

}

public void doLocate() {

DebugUtil.d(TAG,"doLocate");

mLoadingView.setVisibility(VISIBLE);

mLoadErrorView.setVisibility(GONE);

mTodayWeatherContentView.setVisibility(INVISIBLE);

// if (ActivityCompat.checkSelfPermission(mContext, Manifest.permission.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED &&

// ActivityCompat.checkSelfPermission(mContext, Manifest.permission.ACCESS\_COARSE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {

// // TODO: Consider calling

// // ActivityCompat#requestPermissions

// // here to request the missing permissions, and then overriding

// // public void onRequestPermissionsResult(int requestCode, String[] permissions,

// // int[] grantResults)

// // to handle the case where the user grants the permission. See the documentation

// // for ActivityCompat#requestPermissions for more details.

// return;

// }

//

// DebugUtil.d(TAG,"can doLocate");

//

// mLocationManager = (LocationManager) mContext.getSystemService(Context.LOCATION\_SERVICE);

// if (mLocationManager.isProviderEnabled(LocationManager.NETWORK\_PROVIDER)) {

// DebugUtil.d(TAG, "NETWORK\_PROVIDER");

// mLocationManager.requestLocationUpdates(LocationManager.NETWORK\_PROVIDER, 6 \* 60 \* 1000, 10, mLocationListener);

// }else if(mLocationManager.isProviderEnabled(LocationManager.GPS\_PROVIDER)){

// DebugUtil.d(TAG,"GPS\_PROVIDER");

// mLocationManager.requestLocationUpdates(LocationManager.GPS\_PROVIDER, 6 \* 60 \* 1000, 10, mLocationListener);

// }else{

// //将手机位置服务中--基于网络的位置服务关闭后，则获取不到数据

// DebugUtil.d(TAG,"NETWORK\_PROVIDER不可用，无法获取GPS信息!");

// }

//声明LocationClient类

mLocationClient = new LocationClient(mContext.getApplicationContext());

//注册监听函数

mLocationClient.registerLocationListener(mBdAbstractLocationListener);

mLocationClientOption.setLocationMode(LocationClientOption.LocationMode.Hight\_Accuracy);

//可选，设置定位模式，默认高精度

//LocationMode.Hight\_Accuracy：高精度；

//LocationMode. Battery\_Saving：低功耗；

//LocationMode. Device\_Sensors：仅使用设备；

mLocationClientOption.setCoorType("bd09ll");

//可选，设置返回经纬度坐标类型，默认GCJ02

//GCJ02：国测局坐标；

//BD09ll：百度经纬度坐标；

//BD09：百度墨卡托坐标；

//海外地区定位，无需设置坐标类型，统一返回WGS84类型坐标

mLocationClientOption.setScanSpan(60000);

//可选，设置发起定位请求的间隔，int类型，单位ms

//如果设置为0，则代表单次定位，即仅定位一次，默认为0

//如果设置非0，需设置1000ms以上才有效

mLocationClientOption.setOpenGps(true);

//可选，设置是否使用gps，默认false

//使用高精度和仅用设备两种定位模式的，参数必须设置为true

mLocationClientOption.setLocationNotify(true);

//可选，设置是否当GPS有效时按照1S/1次频率输出GPS结果，默认false

mLocationClientOption.setIgnoreKillProcess(false);

//可选，定位SDK内部是一个service，并放到了独立进程。

//设置是否在stop的时候杀死这个进程，默认（建议）不杀死，即setIgnoreKillProcess(true)

mLocationClientOption.SetIgnoreCacheException(false);

//可选，设置是否收集Crash信息，默认收集，即参数为false

mLocationClientOption.setWifiCacheTimeOut(5\*60\*1000);

//可选，V7.2版本新增能力

//如果设置了该接口，首次启动定位时，会先判断当前Wi-Fi是否超出有效期，若超出有效期，会先重新扫描Wi-Fi，然后定位

mLocationClientOption.setEnableSimulateGps(false);

//可选，设置是否需要过滤GPS仿真结果，默认需要，即参数为false

mLocationClientOption.setIsNeedAddress(true);

mLocationClient.setLocOption(mLocationClientOption);

//mLocationClient为第二步初始化过的LocationClient对象

//需将配置好的LocationClientOption对象，通过setLocOption方法传递给LocationClient对象使用

//更多LocationClientOption的配置，请参照类参考中LocationClientOption类的详细说明

mLocationClient.start();

//mLocationClient为第二步初始化过的LocationClient对象

//调用LocationClient的start()方法，便可发起定位请求

}

@Override

protected void onAttachedToWindow() {

super.onAttachedToWindow();

DebugUtil.d(TAG,"onAttachedToWindow");

EventBusUtils.register(this);

}

@Override

protected void onDetachedFromWindow() {

super.onDetachedFromWindow();

DebugUtil.d(TAG,"onDetachedFromWindow");

EventBusUtils.unregister(this);

mLocationClient.unRegisterLocationListener(mBdAbstractLocationListener);

}

@Subscribe(threadMode = ThreadMode.MAIN)

public void onLocatePermissionSuccessEvent(LocatePermissionSuccessEvent event){

doLocate();

}

@Subscribe(threadMode = ThreadMode.MAIN)

public void onSelectCityEvent(SelectCityEvent event){

setToolbarTitle("获取"+event.mCityInfo.city\_name+"天气中...");

getDataFromNetByCity(event.mCityInfo.city\_name);

}

public void setToolbar(Toolbar toolbar){

mToolbar = toolbar;

}

public void setToolbarTitle(String s){

if(mToolbar != null){

//mToolbar.setTitle(s);

TextView tv = mToolbar.findViewById(R.id.toolbar\_title);

tv.setText(s);

}

}

}

package com.zxl.casual.living.custom.view;

import android.content.Context;

import android.graphics.Bitmap;

import android.graphics.Canvas;

import android.graphics.Paint;

import android.graphics.Rect;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.util.AttributeSet;

import com.bumptech.glide.request.target.SimpleTarget;

import com.bumptech.glide.request.transition.Transition;

import com.zxl.casual.living.GlideApp;

import com.zxl.casual.living.http.data.TodayWeatherAirQualityIconCss;

import com.zxl.common.DebugUtil;

/\*\*

\* Created by zxl on 2018/9/5.

\*/

public class TodayWeatherAirQualityIconView extends android.support.v7.widget.AppCompatImageView {

private static final String TAG = "TodayWeatherAirQualityIconView";

private Context mContext;

private TodayWeatherAirQualityIconCss mTodayWeatherAirQualityIconCss;

private Bitmap mBitmap;

private SimpleTarget<Bitmap> mBitmapTarget = new SimpleTarget<Bitmap>() {

@Override

public void onResourceReady(@NonNull Bitmap resource, @Nullable Transition<? super Bitmap> transition) {

DebugUtil.d(TAG,"TodayWeatherDetailIconView = " + TodayWeatherAirQualityIconView.this);

DebugUtil.d(TAG,"width = " + (Integer.valueOf(mTodayWeatherAirQualityIconCss.width)));

DebugUtil.d(TAG,"height = " + (Integer.valueOf(mTodayWeatherAirQualityIconCss.height)));

DebugUtil.d(TAG,"background\_position\_x = " + (-Integer.valueOf(mTodayWeatherAirQualityIconCss.background\_position\_x)));

DebugUtil.d(TAG,"background\_position\_y = " + (-Integer.valueOf(mTodayWeatherAirQualityIconCss.background\_position\_y)));

mBitmap = Bitmap.createBitmap(Integer.valueOf(mTodayWeatherAirQualityIconCss.width),Integer.valueOf(mTodayWeatherAirQualityIconCss.height), Bitmap.Config.ARGB\_8888);

Canvas canvas = new Canvas(mBitmap);

Rect rectRes = new Rect(

-Integer.valueOf(mTodayWeatherAirQualityIconCss.background\_position\_x),

-Integer.valueOf(mTodayWeatherAirQualityIconCss.background\_position\_y),

-Integer.valueOf(mTodayWeatherAirQualityIconCss.background\_position\_x) + Integer.valueOf(mTodayWeatherAirQualityIconCss.width),

-Integer.valueOf(mTodayWeatherAirQualityIconCss.background\_position\_y) + Integer.valueOf(mTodayWeatherAirQualityIconCss.height));

Rect rectDst = new Rect(0,0,mBitmap.getWidth(),mBitmap.getHeight());

Paint paint = new Paint();

// paint.setStyle(Paint.Style.STROKE);

// paint.setAntiAlias(true);

canvas.drawBitmap(resource,rectRes,rectDst,paint);

paint.reset();

paint = null;

setImageBitmap(mBitmap);

}

};

public TodayWeatherAirQualityIconView(Context context) {

super(context);

init(context);

}

public TodayWeatherAirQualityIconView(Context context, @Nullable AttributeSet attrs) {

super(context, attrs);

init(context);

}

public TodayWeatherAirQualityIconView(Context context, @Nullable AttributeSet attrs, int defStyleAttr) {

super(context, attrs, defStyleAttr);

init(context);

}

private void init(Context context){

DebugUtil.d(TAG,"init");

mContext = context;

}

public void setTodayWeatherHumidityIconCss(TodayWeatherAirQualityIconCss iconCss){

mTodayWeatherAirQualityIconCss = iconCss;

GlideApp.with(mContext).asBitmap().load(mTodayWeatherAirQualityIconCss.img).into(mBitmapTarget);

}

@Override

protected void onAttachedToWindow() {

super.onAttachedToWindow();

DebugUtil.d(TAG,"onAttachedToWindow::width = " + getWidth());

DebugUtil.d(TAG,"onAttachedToWindow::height = " + getHeight());

}

@Override

protected void onDetachedFromWindow() {

super.onDetachedFromWindow();

DebugUtil.d(TAG,"onDetachedFromWindow");

if(mBitmap != null){

if(!mBitmap.isRecycled()){

mBitmap.recycle();

}

mBitmap = null;

}

}

}

package com.zxl.casual.living.custom.view;

import android.content.Context;

import android.graphics.Bitmap;

import android.graphics.Canvas;

import android.graphics.Paint;

import android.graphics.PorterDuff;

import android.graphics.PorterDuffXfermode;

import android.graphics.Rect;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.util.AttributeSet;

import com.bumptech.glide.request.target.SimpleTarget;

import com.bumptech.glide.request.transition.Transition;

import com.zxl.casual.living.GlideApp;

import com.zxl.casual.living.http.data.TodayWeatherTemperatureIconCss;

import com.zxl.common.DebugUtil;

/\*\*

\* Created by zxl on 2018/9/5.

\*/

public class TodayWeatherTemperatureView extends android.support.v7.widget.AppCompatImageView {

private static final String TAG = "TodayWeatherTemperatureView";

private Context mContext;

private TodayWeatherTemperatureIconCss mTodayWeatherTemperatureIconCss;

private Bitmap mBottomBitmap;

private Bitmap mTopBitmap;

private SimpleTarget<Bitmap> mBottomBitmapTarget = new SimpleTarget<Bitmap>() {

@Override

public void onResourceReady(@NonNull Bitmap resource, @Nullable Transition<? super Bitmap> transition) {

mBottomBitmap = Bitmap.createBitmap(80,-Integer.valueOf(mTodayWeatherTemperatureIconCss.background\_position\_y2) - 4, Bitmap.Config.ARGB\_8888);

Canvas canvas = new Canvas(mBottomBitmap);

Rect rectRes = new Rect(0,-Integer.valueOf(mTodayWeatherTemperatureIconCss.background\_position\_y1) - 4,80,-Integer.valueOf(mTodayWeatherTemperatureIconCss.background\_position\_y1) - Integer.valueOf(mTodayWeatherTemperatureIconCss.background\_position\_y2) - 8);

Rect rectDst = new Rect(0,0,mBottomBitmap.getWidth(),mBottomBitmap.getHeight());

Paint paint = new Paint();

paint.setStyle(Paint.Style.STROKE);

paint.setAntiAlias(true);

canvas.drawBitmap(resource,rectRes,rectDst,paint);

paint.reset();

paint = null;

GlideApp.with(mContext).asBitmap().load(mTodayWeatherTemperatureIconCss.img).into(mTopBitmapTarget);

}

};

private SimpleTarget<Bitmap> mTopBitmapTarget = new SimpleTarget<Bitmap>() {

@Override

public void onResourceReady(@NonNull Bitmap resource, @Nullable Transition<? super Bitmap> transition) {

double temperatureHeight = Double.valueOf(mTodayWeatherTemperatureIconCss.height2);

mTopBitmap = Bitmap.createBitmap(14, (int) (-Integer.valueOf(mTodayWeatherTemperatureIconCss.background\_position\_y1) - 38 - temperatureHeight), Bitmap.Config.ARGB\_8888);

Canvas canvas = new Canvas(mTopBitmap);

Rect rectRes = new Rect(36, (int) temperatureHeight,50, -Integer.valueOf(mTodayWeatherTemperatureIconCss.background\_position\_y1) - 38);

Rect rectDst = new Rect(0,0,mTopBitmap.getWidth(), mTopBitmap.getHeight());

canvas.drawBitmap(resource,rectRes,rectDst,new Paint());

canvas = new Canvas(mBottomBitmap);

rectRes = new Rect(0,0,mTopBitmap.getWidth(), mTopBitmap.getHeight());

rectDst = new Rect(mBottomBitmap.getWidth()/2 - 4,0,mBottomBitmap.getWidth()/2 + mTopBitmap.getWidth() - 4, mTopBitmap.getHeight() - 2);

Paint paint = new Paint();

paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC\_OVER));

canvas.drawBitmap(mTopBitmap,rectRes,rectDst,paint);

DebugUtil.d(TAG,"mTopBitmapTarget::width = " + getWidth());

DebugUtil.d(TAG,"mTopBitmapTarget::height = " + getHeight());

setImageBitmap(mBottomBitmap);

}

};

public TodayWeatherTemperatureView(Context context) {

super(context);

init(context);

}

public TodayWeatherTemperatureView(Context context, @Nullable AttributeSet attrs) {

super(context, attrs);

init(context);

}

public TodayWeatherTemperatureView(Context context, @Nullable AttributeSet attrs, int defStyleAttr) {

super(context, attrs, defStyleAttr);

init(context);

}

private void init(Context context){

DebugUtil.d(TAG,"init");

mContext = context;

}

public void setTodayWeatherTemperatureIconCss(TodayWeatherTemperatureIconCss iconCss){

mTodayWeatherTemperatureIconCss = iconCss;

GlideApp.with(mContext).asBitmap().load(mTodayWeatherTemperatureIconCss.img).into(mBottomBitmapTarget);

}

@Override

protected void onAttachedToWindow() {

super.onAttachedToWindow();

DebugUtil.d(TAG,"onAttachedToWindow::width = " + getWidth());

DebugUtil.d(TAG,"onAttachedToWindow::height = " + getHeight());

}

@Override

protected void onDetachedFromWindow() {

super.onDetachedFromWindow();

DebugUtil.d(TAG,"onDetachedFromWindow");

if(mTopBitmap != null){

if(!mTopBitmap.isRecycled()){

mTopBitmap.recycle();

}

mTopBitmap = null;

}

if(mBottomBitmap != null){

if(!mBottomBitmap.isRecycled()){

mBottomBitmap.recycle();

}

mBottomBitmap = null;

}

}

}

package com.zxl.casual.living.http.data;

/\*\*

\* Created by zxl on 2018/9/5.

\*/

public class TodayWeather {

/\*

"now\_time":"17:40 实况",

"temperature":"24",

"is\_w":1,

"simple\_content":"周一 多云转阴 21/26°C",

"wind\_direction":"东北风",

"air\_quality":"暂无数据",

"humidity":"83%",

"humidity\_icon\_css":Object{...},

"is\_limit":0,

"wind\_icon\_css":Object{...},

"is\_h":1,

"wind\_value":"2级",

"is\_pol":0,

"temperature\_icon\_css":Object{...}

},

\*/

public String now\_time = "";

public String temperature = "";

public String wind\_direction = "";

public String air\_quality = "";

public String humidity = "";

public String wind\_value = "";

public String limit\_content = "";

public String simple\_content = "";

public int is\_w;

public int is\_h;

public int is\_pol;

public int is\_limit;

public TodayWeatherTemperatureIconCss temperature\_icon\_css;

public TodayWeatherHumidityIconCss humidity\_icon\_css;

public TodayWeatherWindIconCss wind\_icon\_css;

public TodayWeatherAirQualityIconCss air\_quality\_icon\_css;

public TodayWeatherLimitIconCss limit\_icon\_css;

@Override

public String toString() {

return "TodayWeather{" +

"now\_time='" + now\_time + '\'' +

", temperature='" + temperature + '\'' +

", wind\_direction='" + wind\_direction + '\'' +

", air\_quality='" + air\_quality + '\'' +

", humidity='" + humidity + '\'' +

", wind\_value='" + wind\_value + '\'' +

", limit\_content='" + limit\_content + '\'' +

", simple\_content='" + simple\_content + '\'' +

", is\_w=" + is\_w +

", is\_h=" + is\_h +

", is\_pol=" + is\_pol +

", is\_limit=" + is\_limit +

", temperature\_icon\_css=" + temperature\_icon\_css +

", humidity\_icon\_css=" + humidity\_icon\_css +

", wind\_icon\_css=" + wind\_icon\_css +

", air\_quality\_icon\_css=" + air\_quality\_icon\_css +

", limit\_icon\_css=" + limit\_icon\_css +

'}';

}

}

package com.zxl.casual.living.utils;

import android.content.Context;

import android.content.pm.PackageInfo;

import android.content.pm.PackageManager;

import android.content.res.Resources;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.os.AsyncTask;

import android.os.Environment;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.view.View;

import com.bumptech.glide.Glide;

import com.bumptech.glide.request.FutureTarget;

import com.bumptech.glide.request.target.SimpleTarget;

import com.bumptech.glide.request.target.Target;

import com.bumptech.glide.request.transition.Transition;

import com.zxl.casual.living.R;

import com.google.gson.Gson;

import com.google.gson.GsonBuilder;

import com.tencent.mm.opensdk.modelbase.BaseReq;

import com.tencent.mm.opensdk.modelmsg.SendMessageToWX;

import com.tencent.mm.opensdk.modelmsg.WXImageObject;

import com.tencent.mm.opensdk.modelmsg.WXMediaMessage;

import com.tencent.mm.opensdk.modelmsg.WXTextObject;

import com.tencent.mm.opensdk.openapi.IWXAPI;

import com.tencent.mm.opensdk.openapi.WXAPIFactory;

import com.zxl.common.DebugUtil;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

public class CommonUtils {

public static final String TAG = "CommonUtils";

/\*\* 判断是否是快速点击 \*/

private static long lastClickTime;

public static Gson mGson = new GsonBuilder().setDateFormat("yyyy-MM-dd HH:mm:ss").create();

public static IWXAPI mIwxapi;

public static String getVersionName(Context context){

PackageManager pm = context.getPackageManager();

try {

PackageInfo packageInfo = pm.getPackageInfo(context.getPackageName(), 0);

return packageInfo.versionName;

} catch (PackageManager.NameNotFoundException e) {

e.printStackTrace();

}

return "";

}

public static int getVersionCode(Context context){

PackageManager pm = context.getPackageManager();

try {

PackageInfo packageInfo = pm.getPackageInfo(context.getPackageName(), 0);

return packageInfo.versionCode;

} catch (PackageManager.NameNotFoundException e) {

e.printStackTrace();

}

return 0;

}

public static int px2dip(int pxValue){

final float scale = Resources.getSystem().getDisplayMetrics().density;

return (int) (pxValue / scale + 0.5f);

}

public static float dip2px(float dipValue){

final float scale = Resources.getSystem().getDisplayMetrics().density;

return (dipValue \* scale + 0.5f);

}

public static int screenWidth(){

return Resources.getSystem().getDisplayMetrics().widthPixels;

}

public static int screenHeight(){

return Resources.getSystem().getDisplayMetrics().heightPixels;

}

public static boolean isFastDoubleClick() {

long time = System.currentTimeMillis();

long timeD = time - lastClickTime;

DebugUtil.d(TAG,"isFastDoubleClick::timeD = " + timeD);

if (0 < timeD && timeD < 500) {

return true;

}

lastClickTime = time;

return false;

}

public static final void regToWX(Context context){

mIwxapi = WXAPIFactory.createWXAPI(context,Constants.WX\_APP\_ID,true);

mIwxapi.registerApp(Constants.WX\_APP\_ID);

}

public static final void sendWXMessage(BaseReq baseReq){

mIwxapi.sendReq(baseReq);

}

public static final void shareWXText(String content,String desc,int scene){

//SendMessageToWX.Req.WXSceneTimeline 设置发送到朋友圈

//SendMessageToWX.Req.WXSceneSession 设置发送给朋友

WXTextObject wxTextObject = new WXTextObject();

wxTextObject.text = content;

WXMediaMessage wxMediaMessage = new WXMediaMessage();

wxMediaMessage.mediaObject = wxTextObject;

wxMediaMessage.description = content;

SendMessageToWX.Req req = new SendMessageToWX.Req();

req.transaction = String.valueOf(System.currentTimeMillis());

req.message = wxMediaMessage;

req.scene = scene;

CommonUtils.sendWXMessage(req);

}

public static final void shareWXBitmap(final Context context, final String bitmapUrl, final int scene){

new AsyncTask<Void, Integer, String>() {

@Override

protected String doInBackground(Void... params) {

File file = null;

try {

FutureTarget<File> future = Glide

.with(context)

.load(bitmapUrl)

.downloadOnly(Target.SIZE\_ORIGINAL, Target.SIZE\_ORIGINAL);

file = future.get();

// 首先保存图片

File pictureFolder = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY\_PICTURES).getAbsoluteFile();

File appDir = new File(pictureFolder ,"test\_weather");

if (!appDir.exists()) {

appDir.mkdirs();

}

String fileName = bitmapUrl.substring(bitmapUrl.lastIndexOf("/"));

File destFile = new File(appDir, fileName);

FileInputStream fis = new FileInputStream(file);

FileOutputStream fos = new FileOutputStream(destFile);

byte buffer[] = new byte[1024];

int count = 0;

while((count = fis.read(buffer)) != -1){

fos.write(buffer,0,count);

}

fos.close();

fis.close();

return destFile.getPath();

} catch (Exception e) {

DebugUtil.d(TAG, e.getMessage());

}

return "";

}

@Override

protected void onPostExecute(String path) {

DebugUtil.d(TAG,"onPostExecute::path = " + path);

Bitmap thumbBmp = WXUtil.createBitmapThumbnail(path, Constants.THUMB\_SIZE, Constants.THUMB\_SIZE);

WXImageObject wxImageObject = new WXImageObject();

wxImageObject.imagePath = path;

WXMediaMessage wxMediaMessage = new WXMediaMessage();

wxMediaMessage.mediaObject = wxImageObject;

wxMediaMessage.thumbData = WXUtil.bmpToByteArray(thumbBmp,true);

wxMediaMessage.title = "title";

wxMediaMessage.description = "description";

SendMessageToWX.Req req = new SendMessageToWX.Req();

req.transaction = String.valueOf(System.currentTimeMillis())+"img";

req.message = wxMediaMessage;

req.scene = scene;

CommonUtils.sendWXMessage(req);

}

@Override

protected void onProgressUpdate(Integer... values) {

super.onProgressUpdate(values);

}

}.execute();

}

}

package com.zxl.casual.living.utils;

import android.app.DownloadManager;

import android.content.Context;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.database.Cursor;

import android.net.Uri;

import android.os.Build;

import android.os.Environment;

import android.os.ParcelFileDescriptor;

import android.provider.Settings;

import android.support.v4.content.FileProvider;

import com.zxl.common.DebugUtil;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.IOException;

/\*\*

\* Created by zxl on 2018/10/9.

\*/

public class DownloadUtils {

private static final String TAG = "DownloadUtils";

public static final boolean checkDownloadEnable(Context context){

int state = context.getPackageManager().getApplicationEnabledSetting("com.android.providers.downloads");

if(state == PackageManager.COMPONENT\_ENABLED\_STATE\_DISABLED ||

state == PackageManager.COMPONENT\_ENABLED\_STATE\_DISABLED\_USER ||

state == PackageManager.COMPONENT\_ENABLED\_STATE\_DISABLED\_UNTIL\_USED){

return false;

}

return true;

}

public static final void openDownloadEnableSettings(Context context){

Intent intent = new Intent(Settings.ACTION\_APPLICATION\_DETAILS\_SETTINGS);

intent.setData(Uri.parse("com.android.providers.downloads"));

context.startActivity(intent);

}

public static final void removeDownloadApk(Context context){

}

public static final void download(Context context,String url){

Uri uri = Uri.parse(url);

DownloadManager.Request request = new DownloadManager.Request(uri);

request.setAllowedNetworkTypes(DownloadManager.Request.NETWORK\_MOBILE | DownloadManager.Request.NETWORK\_WIFI);

request.setNotificationVisibility(DownloadManager.Request.VISIBILITY\_VISIBLE\_NOTIFY\_COMPLETED);

request.setDestinationInExternalPublicDir(Environment.DIRECTORY\_DOWNLOADS,Constants.UPDATE\_APP\_NAME);

request.setTitle("更新");

request.setDescription("下载更新包");

request.setMimeType("application/vnd.android.package-archive");

request.allowScanningByMediaScanner();

request.setVisibleInDownloadsUi(true);

DownloadManager downloadManager = (DownloadManager) context.getSystemService(Context.DOWNLOAD\_SERVICE);

downloadManager.remove(SharePreUtils.getInstance(context).getDownloadId());

long id = downloadManager.enqueue(request);

SharePreUtils.getInstance(context).saveDownloadId(id);

}

public static final void installApk(Context context, long downloadApkId) {

DownloadManager dManager = (DownloadManager) context.getSystemService(Context.DOWNLOAD\_SERVICE);

Intent install = new Intent(Intent.ACTION\_VIEW);

Uri downloadFileUri = dManager.getUriForDownloadedFile(downloadApkId);

DebugUtil.d(TAG, "installApk::downloadFileUri = " + downloadFileUri);

if (downloadFileUri != null) {

DebugUtil.d(TAG, "installApk::uri = " + downloadFileUri.toString());

install.setDataAndType(downloadFileUri, "application/vnd.android.package-archive");

if ((Build.VERSION.SDK\_INT >= 24)) {//判读版本是否在7.0以上

install.addFlags(Intent.FLAG\_GRANT\_READ\_URI\_PERMISSION); //添加这一句表示对目标应用临时授权该Uri所代表的文件

install.addFlags(Intent.FLAG\_GRANT\_WRITE\_URI\_PERMISSION);

}

install.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

if (install.resolveActivity(context.getPackageManager()) != null) {

context.startActivity(install);

} else {

DebugUtil.d(TAG, "自动安装失败，请手动安装");

}

} else {

DebugUtil.d(TAG, "download error");

}

}

//检查下载状态

public static final int checkStatus(Context context, long id) {

DownloadManager downloadManager = (DownloadManager) context.getSystemService(Context.DOWNLOAD\_SERVICE);

DownloadManager.Query query = new DownloadManager.Query();

//通过下载的id查找

query.setFilterById(id);

Cursor c = downloadManager.query(query);

int status = DownloadManager.STATUS\_SUCCESSFUL;

if (c.moveToFirst()) {

status = c.getInt(c.getColumnIndex(DownloadManager.COLUMN\_STATUS));

switch (status) {

//下载暂停

case DownloadManager.STATUS\_PAUSED:

break;

//下载延迟

case DownloadManager.STATUS\_PENDING:

break;

//正在下载

case DownloadManager.STATUS\_RUNNING:

break;

//下载完成

case DownloadManager.STATUS\_SUCCESSFUL:

//下载完成安装APK

break;

//下载失败

case DownloadManager.STATUS\_FAILED:

break;

}

}

c.close();

return status;

}

}

package com.zxl.casual.living.utils;

import java.io.ByteArrayOutputStream;

import java.io.File;

import java.io.IOException;

import java.io.InputStream;

import java.io.RandomAccessFile;

import java.net.HttpURLConnection;

import java.net.MalformedURLException;

import java.net.URL;

import java.net.URLConnection;

import java.security.MessageDigest;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Bitmap.CompressFormat;

import android.graphics.Canvas;

import android.graphics.Matrix;

import android.graphics.Rect;

import android.util.Log;

import com.zxl.common.DebugUtil;

public class WXUtil {

private static final String TAG = "SDK\_Sample.Util";

public static byte[] bmpToByteArray(final Bitmap bmp, final boolean needRecycle) {

ByteArrayOutputStream output = new ByteArrayOutputStream();

bmp.compress(CompressFormat.PNG, 100, output);

if (needRecycle) {

bmp.recycle();

}

byte[] result = output.toByteArray();

try {

output.close();

} catch (Exception e) {

e.printStackTrace();

}

return result;

}

//压缩图片

public static Bitmap createBitmapThumbnail(Bitmap bitMap, int viewWidth, int viewHeight) {

int width = bitMap.getWidth();

int height = bitMap.getHeight();

// 设置想要的大小

int newWidth = viewWidth;

int newHeight = viewHeight;

// 计算缩放比例

float scaleWidth = ((float) newWidth) / width;

float scaleHeight = ((float) newHeight) / height;

// 取得想要缩放的matrix参数

Matrix matrix = new Matrix();

matrix.postScale(scaleWidth, scaleHeight);

// 得到新的图片

Bitmap newBitMap = Bitmap.createBitmap(bitMap, 0, 0, width, height, matrix, true);

return newBitMap;

}

//压缩图片

public static Bitmap createBitmapThumbnail(String path, int viewWidth, int viewHeight) {

BitmapFactory.Options options = new BitmapFactory.Options();

options.inJustDecodeBounds = true;

BitmapFactory.decodeFile(path,options);

int width = options.outWidth;

int height = options.outHeight;

DebugUtil.d(TAG,"createBitmapThumbnail::width = " + width + "::height = " + height);

// 设置想要的大小

int newWidth = viewWidth;

int newHeight = viewHeight;

// 计算缩放比例

int scaleWidth = newWidth / width;

int scaleHeight = newHeight / height;

// 取得想要缩放的matrix参数

Matrix matrix = new Matrix();

matrix.postScale(scaleWidth, scaleHeight);

float realWidth = options.outWidth;

float realHeight = options.outHeight;

DebugUtil.d(TAG,"真实图片高度：" + realHeight + "宽度:" + realWidth);

// 计算缩放比&nbsp;&nbsp;&nbsp; &nbsp;&nbsp;&nbsp;

int scale = (int) ((realHeight > realWidth ? realHeight : realWidth) / 100);

if (scale <= 0)

{

scale = 1;

}

options.inSampleSize = scale;

options.inJustDecodeBounds = false;

DebugUtil.d(TAG,"createBitmapThumbnail::inSampleSize = " + options.inSampleSize);

Bitmap bitmap = BitmapFactory.decodeFile(path,options);

DebugUtil.d(TAG,"createBitmapThumbnail::bitmap = " + bitmap.getWidth() + "--->"+bitmap.getHeight());

return bitmap;

}

public static byte[] getHtmlByteArray(final String url) {

URL htmlUrl = null;

InputStream inStream = null;

try {

htmlUrl = new URL(url);

URLConnection connection = htmlUrl.openConnection();

HttpURLConnection httpConnection = (HttpURLConnection)connection;

int responseCode = httpConnection.getResponseCode();

if(responseCode == HttpURLConnection.HTTP\_OK){

inStream = httpConnection.getInputStream();

}

} catch (MalformedURLException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

byte[] data = inputStreamToByte(inStream);

return data;

}

public static byte[] inputStreamToByte(InputStream is) {

try{

ByteArrayOutputStream bytestream = new ByteArrayOutputStream();

int ch;

while ((ch = is.read()) != -1) {

bytestream.write(ch);

}

byte imgdata[] = bytestream.toByteArray();

bytestream.close();

return imgdata;

}catch(Exception e){

e.printStackTrace();

}

return null;

}

public static byte[] readFromFile(String fileName, int offset, int len) {

if (fileName == null) {

return null;

}

File file = new File(fileName);

if (!file.exists()) {

Log.i(TAG, "readFromFile: file not found");

return null;

}

if (len == -1) {

len = (int) file.length();

}

Log.d(TAG, "readFromFile : offset = " + offset + " len = " + len + " offset + len = " + (offset + len));

if(offset <0){

Log.e(TAG, "readFromFile invalid offset:" + offset);

return null;

}

if(len <=0 ){

Log.e(TAG, "readFromFile invalid len:" + len);

return null;

}

if(offset + len > (int) file.length()){

Log.e(TAG, "readFromFile invalid file len:" + file.length());

return null;

}

byte[] b = null;

try {

RandomAccessFile in = new RandomAccessFile(fileName, "r");

b = new byte[len];

in.seek(offset);

in.readFully(b);

in.close();

} catch (Exception e) {

Log.e(TAG, "readFromFile : errMsg = " + e.getMessage());

e.printStackTrace();

}

return b;

}

private static final int MAX\_DECODE\_PICTURE\_SIZE = 1920 \* 1440;

public static Bitmap extractThumbNail(final String path, final int height, final int width, final boolean crop) {

BitmapFactory.Options options = new BitmapFactory.Options();

try {

options.inJustDecodeBounds = true;

Bitmap tmp = BitmapFactory.decodeFile(path, options);

if (tmp != null) {

tmp.recycle();

tmp = null;

}

Log.d(TAG, "extractThumbNail: round=" + width + "x" + height + ", crop=" + crop);

final double beY = options.outHeight \* 1.0 / height;

final double beX = options.outWidth \* 1.0 / width;

Log.d(TAG, "extractThumbNail: extract beX = " + beX + ", beY = " + beY);

options.inSampleSize = (int) (crop ? (beY > beX ? beX : beY) : (beY < beX ? beX : beY));

if (options.inSampleSize <= 1) {

options.inSampleSize = 1;

}

// NOTE: out of memory error

while (options.outHeight \* options.outWidth / options.inSampleSize > MAX\_DECODE\_PICTURE\_SIZE) {

options.inSampleSize++;

}

int newHeight = height;

int newWidth = width;

if (crop) {

if (beY > beX) {

newHeight = (int) (newWidth \* 1.0 \* options.outHeight / options.outWidth);

} else {

newWidth = (int) (newHeight \* 1.0 \* options.outWidth / options.outHeight);

}

} else {

if (beY < beX) {

newHeight = (int) (newWidth \* 1.0 \* options.outHeight / options.outWidth);

} else {

newWidth = (int) (newHeight \* 1.0 \* options.outWidth / options.outHeight);

}

package com.zxl.casual.living.http;

import android.content.Context;

import android.net.ConnectivityManager;

import android.net.NetworkInfo;

import com.zxl.casual.living.http.data.CityInfoListResponseBean;

import com.zxl.casual.living.http.data.QSBKElementList;

import com.zxl.casual.living.http.data.ResponseBaseBean;

import com.zxl.casual.living.http.data.TaoBaoAnchorListResponseBean;

import com.zxl.casual.living.http.data.TodayWeatherResponseBean;

import com.zxl.casual.living.http.data.UpdateInfoResponseBean;

import com.zxl.casual.living.http.data.UserInfoResponseBean;

import com.zxl.casual.living.http.listener.NetRequestListener;

import com.zxl.casual.living.utils.CommonUtils;

import com.zxl.casual.living.utils.Constants;

import com.zxl.common.DebugUtil;

import java.io.IOException;

import java.util.concurrent.TimeUnit;

import okhttp3.OkHttpClient;

import okhttp3.ResponseBody;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

import retrofit2.Retrofit;

import retrofit2.adapter.rxjava.RxJavaCallAdapterFactory;

import retrofit2.converter.gson.GsonConverterFactory;

import rx.Observable;

import rx.Subscriber;

import rx.android.schedulers.AndroidSchedulers;

import rx.schedulers.Schedulers;

/\*\*

\* Created by zxl on 2018/9/5.

\*/

public class HttpUtils {

private static final String TAG = "HttpUtils";

private static HttpUtils mHttpUtils;

private static Object mLock = new Object();

private static Retrofit mRetrofit;

private static HttpAPI mHttpAPI;

private HttpUtils(){

OkHttpClient.Builder okBuilder = new OkHttpClient.Builder();

okBuilder.connectTimeout(1, TimeUnit.MINUTES);

okBuilder.readTimeout(1,TimeUnit.MINUTES);

OkHttpClient okHttpClient = okBuilder.build();

Retrofit.Builder retrofitBuilder = new Retrofit.Builder();

Retrofit mRetrofit = retrofitBuilder

// .baseUrl("http://www.zxltest.cn/cgi\_server/")

.baseUrl(Constants.WEATHER\_BASE\_URL)

.addConverterFactory(GsonConverterFactory.create())

.addCallAdapterFactory(RxJavaCallAdapterFactory.create())

.client(okHttpClient)

.build();

mHttpAPI = mRetrofit.create(HttpAPI.class);

}

public static HttpUtils getInstance(){

DebugUtil.d(TAG,"getInstance");

if(null == mHttpUtils){

synchronized (mLock){

if(null == mHttpUtils){

mHttpUtils = new HttpUtils();

}

}

}

return mHttpUtils;

}

public void getZHTianQiByLocation(Context context, String l, final NetRequestListener listener){

DebugUtil.d(TAG,"getZHTianQiByLocation::l = " + l);

if(isNetworkAvailable(context)){

Observable<TodayWeatherResponseBean> observable = mHttpAPI.getZHTianQiByLocation(l);

observable.subscribeOn(Schedulers.io())

.observeOn(AndroidSchedulers.mainThread())

.subscribe(new Subscriber<ResponseBaseBean>() {

@Override

public void onCompleted() {

DebugUtil.d(TAG,"getZHTianQiByLocation::onCompleted");

}

@Override

public void onError(Throwable e) {

DebugUtil.d(TAG,"getZHTianQiByLocation::onError::e = " + e);

if(listener != null){

listener.onNetError(e);

}

}

@Override

public void onNext(ResponseBaseBean responseBaseBean) {

DebugUtil.d(TAG,"getZHTianQiByLocation::onNext::responseBaseBean = " + responseBaseBean);

if(responseBaseBean.code == 0){

if(listener != null){

listener.onSuccess(responseBaseBean);

}

}else{

if(listener != null){

listener.onServerError(responseBaseBean);

}

}

}

});

}else{

DebugUtil.d(TAG,"getZHTianQiByLocation::net work error");

if(listener != null){

listener.onNetError();

}

}

}

package com.zxl.casual.living.common;import android.os.Handler;import android.os.HandlerThread;import android.os.Looper;import com.zxl.casual.living.event.UploadLogFileEvent;import com.zxl.casual.living.http.HttpUtils;import com.zxl.casual.living.http.listener.NetRequestListener;import com.zxl.casual.living.http.retrofit.FileRequestBody;import com.zxl.casual.living.http.retrofit.RetrofitCallback;import com.zxl.casual.living.utils.Constants;import com.zxl.casual.living.utils.EventBusUtils;import com.zxl.common.DebugUtil;import org.greenrobot.eventbus.EventBus;import java.io.File;import okhttp3.MediaType;import okhttp3.MultipartBody;import okhttp3.RequestBody;import retrofit2.Call;import retrofit2.Response;/\*\* \* Created by zxl on 2018/11/15. \*/public class UploadLogFileTask { private static final String TAG = "UploadLogFileTask"; private static boolean isUploading = false; private static NetRequestListener mNetRequestListener = null; private static HandlerThread mHandlerThread = null; private static Handler mHandler = null; private static RetrofitCallback mRetrofitCallback = new RetrofitCallback() { @Override public void onSuccess(Call call, Response response) { DebugUtil.d(TAG,"RetrofitCallback::onSuccess"); } @Override public void onLoading(long total, long progress) { DebugUtil.d(TAG,"RetrofitCallback::onLoading = " + (progress \* 1.0 / total)); try { Thread.sleep(10); } catch (InterruptedException e) { e.printStackTrace(); } if(total == progress){ File logDir = new File(Constants.APP\_CRASH\_PATH); File[] logFiles = logDir.listFiles(); if(logFiles != null && logFiles.length > 0){ File file = logFiles[0]; file.delete(); } mHandler.removeCallbacks(mTask); mHandler.postDelayed(mTask,500); } EventBusUtils.post(new UploadLogFileEvent(total,progress)); } @Override public void onFailure(Call call, Throwable t) { DebugUtil.d(TAG,"RetrofitCallback::onLoading"); } }; private static Runnable mTask = new Runnable() { @Override public void run() { File logDir = new File(Constants.APP\_CRASH\_PATH); File[] logFiles = logDir.listFiles(); if(logFiles != null && logFiles.length > 0){ File file = logFiles[0]; DebugUtil.d(TAG,"uploadFile::file = " + file.getName()); MultipartBody.Builder builder = new MultipartBody.Builder(); RequestBody requestBody = RequestBody.create(MediaType.parse("image/png"), file); builder.addFormDataPart("file", file.getName(), requestBody); builder.setType(MultipartBody.FORM); MultipartBody multipartBody = builder.build(); FileRequestBody fileRequestBody = new FileRequestBody(multipartBody,mRetrofitCallback); HttpUtils.getInstance().uploadFile(fileRequestBody,mNetRequestListener); }else{ isUploading = false; } DebugUtil.d(TAG,"mTask::isUploading = " + isUploading); } }; public static void start(){ DebugUtil.d(TAG,"start::isUploading = " + isUploading); if(isUploading){ return; } isUploading = true; if(mHandlerThread == null){ mHandlerThread = new HandlerThread(TAG); mHandlerThread.start(); mHandler = new Handler(mHandlerThread.getLooper()); } mHandler.postDelayed(mTask,500); } public static void setNetRequestListener(NetRequestListener listener){ mNetRequestListener = listener; } public static boolean isStart(){ return isUploading; }}