```
package com.frostsowner.magic.weather;

import android.content.Context;
import android.util.Log;

import androidx.multidex.MultiDex;

import com.android.bugfix.logic.BugFixManager;
import com.evernote.android.job.JobConfig;
import com.evernote.android.job.JobManager;
import com.fm.commons.http.ContextHolder;
import com.fm.commons.logic.BeanFactory;
import com.fm.commons.thread.UIThread;
import com.fm.commons.util.ApkResources;
import com.fm.commons.util.DeviceInfoUtils;
import com.fm.commons.util.SystemUtils;
import com.frostsowner.magic.weather.activity.HomeTabActivity;
import com.frostsowner.magic.weather.base.BaseApplication;
import com.frostsowner.magic.weather.logic.CrashHandler;
import com.frostsowner.magic.weather.logic.LogicModule;
import com.frostsowner.magic.weather.net.AdSourceRequestManager;
import com.frostsowner.magic.weather.net.WeatherRequestManager;
import com.frostsowner.magic.weather.share.ShareConfig;
import com.frostsowner.magic.weather.topon.AtAdManager;
import com.frostsowner.magic.weather.upush.PushPlugin;
import com.frostsowner.magic.weather.utils.NetUtils;
import com.frostsowner.magic.weather.work.DemonsCreator;
import com.tencent.bugly.beta.Beta;

import java.util.HashMap;
import java.util.Map;

import static com.frostsowner.magic.weather.Constant.APP_INIT;
import static com.frostsowner.magic.weather.Constant.BUGLY_APP_ID;

public class AppApplication extends BaseApplication{

    @Override
    public void onCreate(){
        super.onCreate();
        UIThread.init();
        ContextHolder.set(this);

BeanFactory.getBean(AdSourceRequestManager.class).setUseTestServer(Cons
tant.TEST_SERVER_ENVIRONMENT);

BeanFactory.getBean(WeatherRequestManager.class).setUseTestServer(Const
ant.TEST_SERVER_ENVIRONMENT);
        if(isProperty("privacy")){
            CrashHandler.getInstance().init(this);
```

```
                    LogicModule.init();
                    UMConfig.preInit(this);
                    ShareConfig.set(this);
                    initActivityLife();
                    DLTaskConfig.init(this);
                    initBugly();
                    JobManager.create(this).addJobCreator(new DemonsCreator());
                    JobConfig.setLogcatEnabled(true);
                    BeanFactory.getBean(AtAdManager.class).init(this);
                    TrackPlugin.track(this,APP_INIT,null);
                    PushPlugin.initPush(this);
                    ShareConfig.initIds();
                }
            }

            private void initBugly(){
                BugFixManager bugFixManager =
        BeanFactory.getBean(BugFixManager.class);
                Beta.canShowUpgradeActs.add(HomeTabActivity.class);
                Beta.appChannel = Constant.CHANNEL;
                Beta.autoCheckUpgrade = false;
                Beta.autoCheckAppUpgrade = false;
                Beta.showInterruptedStrategy = true;
                Beta.upgradeCheckPeriod = 10*1000;
                Beta.upgradeDialogLayoutId = R.layout.dialog_upgrade_custom;
        //        Beta.tipsDialogLayoutId
                bugFixManager.init(this,BUGLY_APP_ID);
            }

            @Override
            protected void attachBaseContext(Context base){
                super.attachBaseContext(base);
                MultiDex.install(this);
            }

            private Map<String,String> getDeviceInfo(){
                Map<String,String> params = new HashMap<>();
                Constant.isEmulator = DeviceInfoUtils.isEmulator(this);
                params.put("emulator",Constant.isEmulator?"模拟器":"真机");
                if(DeviceInfoUtils.isCMCC()){
                    params.put("sim","中国移动");
                }
                else if(DeviceInfoUtils.isCMTC()){
                    params.put("sim","中国电信");
                }
                else if(DeviceInfoUtils.isCMUC()){
                    params.put("sim","中国联通");
                }
                params.put("device", SystemUtils.getDeviceInfo()+"
        "+SystemUtils.getSystemModel()+" Android
```

```
"+SystemUtils.getAndroidSystemVersion());
        params.put("root",DeviceInfoUtils.haveRoot()?"是":"否");
        params.put("network", NetUtils.getNetWorkName(this));
        StringBuilder builder = new StringBuilder();
        builder.append("客户端: "+params.get("emulator"));
        builder.append(", 手机卡 : "+params.get("sim"));
        builder.append(", 设备 : "+params.get("device"));
        builder.append(", 破解 : "+params.get("root"));
        builder.append(", 网络 : "+params.get("network"));
        params.put("info",builder.toString());
        if(ApkResources.isDebug()){
            Log.d("ijimu",builder.toString());
        }
        return params;
    }
}
package com.frostsowner.magic.weather.activity;

import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.FrameLayout;

import androidx.annotation.Nullable;

import com.evernote.android.job.JobConfig;
import com.evernote.android.job.JobManager;
import com.fm.commons.logic.BeanFactory;
import com.fm.commons.logic.LocalStorage;
import com.frostsowner.magic.weather.Constant;
import com.frostsowner.magic.weather.DLTaskConfig;
import com.frostsowner.magic.weather.R;
import com.frostsowner.magic.weather.TrackPlugin;
import com.frostsowner.magic.weather.UMConfig;
import com.frostsowner.magic.weather.dialog.PermissionAlertDialog;
import com.frostsowner.magic.weather.dialog.PrivacyDialog;
import com.frostsowner.magic.weather.domain.yunyida.YunWeather;
import com.frostsowner.magic.weather.event.ActionEvent;
import com.frostsowner.magic.weather.event.EventType;
import com.frostsowner.magic.weather.logic.CityManager;
import com.frostsowner.magic.weather.logic.CrashHandler;
import com.frostsowner.magic.weather.logic.LogicModule;
import com.frostsowner.magic.weather.logic.UserActionPatternManager;
import com.frostsowner.magic.weather.service.NotificationHandleService;
import com.frostsowner.magic.weather.service.WeatherService;
import com.frostsowner.magic.weather.service.WidgetHandleService;
import com.frostsowner.magic.weather.share.ShareConfig;
```

```
import com.frostsowner.magic.weather.topon.AtAdManager;
import com.frostsowner.magic.weather.upush.PushPlugin;
import com.frostsowner.magic.weather.utils.ConnectionUtils;
import com.frostsowner.magic.weather.utils.StringUtils;
import com.frostsowner.magic.weather.work.DemonsCreator;
import com.umeng.cconfig.UMRemoteConfig;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

import butterknife.BindView;

import static com.frostsowner.magic.weather.Constant.APP_INIT;
import static com.frostsowner.magic.weather.Constant.APP_START;
import static com.frostsowner.magic.weather.Constant.CHANNEL;
import static com.frostsowner.magic.weather.Constant.HOME_HAS_DEAD;
import static
com.frostsowner.magic.weather.Constant.NEED_REQUEST_AD_FLAG;
import static
com.frostsowner.magic.weather.Constant.SPLASH_GUIDE_AGREE;
import static com.frostsowner.magic.weather.Constant.SPLASH_GUIDE_SHOW;
import static com.frostsowner.magic.weather.Constant.UM_AD_FLAG_KEY;
import static
com.frostsowner.magic.weather.event.EventType.CHECK_UPGRADE;

public class SplashAdActivity extends AtSplashAdActivity{

    private LocalStorage localStorage;

    private boolean showGuide;

    @BindView(R.id.splash_container)
    FrameLayout splashContainer;

    private PrivacyDialog privacyDialog;
    private PermissionAlertDialog permissionAlertDialog;

    public boolean canJump = false;
    private boolean isLifeShowAd;
    private UserActionPatternManager userActionPatternManager;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        if (!this.isTaskRoot()) { // 判断当前 activity 是不是所在任务栈
的根
            Intent intent = getIntent();
            if (intent != null) {
```

```
                    String action = intent.getAction();
                    if (intent.hasCategory(Intent.CATEGORY_LAUNCHER) &&
Intent.ACTION_MAIN.equals(action)) {
                        Constant.SHOW_SPLASH_AD = false;
                        finish();
                        return;
                    }
                }
            }
            setContentView(R.layout.activity_splash_ad);
            localStorage = BeanFactory.getBean(LocalStorage.class);
            userActionPatternManager =
BeanFactory.getBean(UserActionPatternManager.class);
            isLifeShowAd =
getIntent().getBooleanExtra("isLifeShowAd",false);
            logger.info("init life show ad from onCreate, isLifeShowAd =
"+isLifeShowAd);
            permissionPrepare();
        }

        @Override
        protected void onNewIntent(Intent intent){
            super.onNewIntent(intent);
            setIntent(intent);
            isLifeShowAd =
getIntent().getBooleanExtra("isLifeShowAd",false);
            logger.info("init life show ad from onNewIntent, isLifeShowAd =
"+isLifeShowAd);
            permissionPrepare();
        }

        @Override
        protected void permissionPrepare(){
//          super.permissionPrepare();
            Constant.SHOW_SPLASH_AD = true;
            initViews();
        }

        private void initViews(){
            showGuide = !isHasShowPrivacy();
            if(showGuide){
                if(privacyDialog == null){
                    privacyDialog = new PrivacyDialog(this, v -> {
                        writeToApp("true");
                        TrackPlugin.track(this, SPLASH_GUIDE_AGREE, null);
                        UMConfig.init(getApplication());
                        CrashHandler.getInstance().init(this);
                        LogicModule.init();
                        ShareConfig.set(getApplication());
                        DLTaskConfig.init(getApplication());
```

```
                        JobManager.create(this).addJobCreator(new
DemonsCreator());
                        JobConfig.setLogcatEnabled(true);

BeanFactory.getBean(AtAdManager.class).init(getApplication());
                        TrackPlugin.track(this, APP_INIT, null);
                        PushPlugin.initPush(getApplication());
                        ShareConfig.initIds();
                        initRelaxFlag();
                    }, new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            System.exit(0);
                        }
                    });
                }
                if(!privacyDialog.isShowing()){
                    privacyDialog.show();
                    TrackPlugin.track(this,SPLASH_GUIDE_SHOW,null);
                    logInfo("privacyDialog show");
                }
            }else{
                UMConfig.init(getApplication());
                initRelaxFlag();
            }
        }

        private void showPermissionAlert(){
            if(permissionAlertDialog == null){
                permissionAlertDialog = new PermissionAlertDialog(this,
isDenied -> {
                    checkPermissions();
                }, new View.OnClickListener() {
                    @Override
                    public void onClick(View v){
                        initRelaxFlag();
                    }
                });
            }
            permissionAlertDialog.show();
            logInfo("permissionAlertDialog show");
        }

        private void initRelaxFlag(){
            if(localStorage == null)localStorage =
BeanFactory.getBean(LocalStorage.class);
            boolean isNeedRequestAdFlag =
localStorage.get(NEED_REQUEST_AD_FLAG,true);
//          loginFromServer();
            if(!isNeedRequestAdFlag){
```

```
                    logInfo("no need request ad flag");
                    Constant.SHOW_AD = true;
                    execute();
                    return;
                }
                if(!ConnectionUtils.isConnectionAvailable(this)){
                    execute();
                    return;
                }
                String adFlag =
        UMRemoteConfig.getInstance().getConfigValue(UM_AD_FLAG_KEY);
                Log.d("ijimu","splash um remote config value : "+adFlag+" ,
        channel : "+CHANNEL);
                if(StringUtils.isEmpty(adFlag)){
                    Constant.SHOW_AD = false;
                    execute();
                }else{
                    Constant.SHOW_AD = TextUtils.equals("开",adFlag);
                    execute();
                }
            }


            private void execute(){
                if(!Constant.SHOW_AD||Constant.isEmulator){
                    toMainView();
                }else{
                    executeAd();
                }
                TrackPlugin.track(this,APP_START,null);
            }

            private void executeAd(){
                showAtSplash(splashContainer);
            }

            @Override
            protected void onPause(){
                super.onPause();
                canJump = false;
            }

            @Override
            protected void onResume(){
                super.onResume();
                toMainView();
            }

            @Override
            public void toMainView(){
```

```
                if (canJump) {
                    if(!isLifeShowAd||HOME_HAS_DEAD){
                        startActivity(new Intent(this, HomeTabActivity.class));
                        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
                            startForegroundService(new Intent(getContext(),
NotificationHandleService.class));
                            startForegroundService(new Intent(getContext(),
WeatherService.class));
//                            startForegroundService(new Intent(getContext(),
LoginService.class));
                            startForegroundService(new Intent(getContext(),
WidgetHandleService.class));
                        }else{
                            startService(new Intent(getContext(),
NotificationHandleService.class));
                            startService(new Intent(getContext(),
WeatherService.class));
//                            startService(new Intent(getContext(),
LoginService.class));
                            startService(new Intent(getContext(),
WidgetHandleService.class));
                        }
                        if(userActionPatternManager == null){
                            userActionPatternManager =
BeanFactory.getBean(UserActionPatternManager.class);
                        }
                        userActionPatternManager.updateAppStartTime();
                    }else{
                        CityManager cityManager =
BeanFactory.getBean(CityManager.class);
                        YunWeather cityWeather =
cityManager.getCurrentCityWeather();
                        ActionEvent event = new
ActionEvent(EventType.WEATHER_REQUEST_SUCCESS);
                        if(cityWeather != null
&& !StringUtils.isEmpty(cityWeather.getCityName())){

event.setAttr("cityName",cityWeather.getCityName());
                            event.setAttr("fake","fake");
                        }
                        event.setAttr("reset","reset");
                        fire(event);
                    }
                    this.finish();
                } else {
                    canJump = true;
                }
            }

            @Override
```

```
    public void onBackPressed() {

    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if(privacyDialog!=null&&privacyDialog.isShowing()){
            privacyDialog.dismiss();
            privacyDialog = null;
        }
        if(permissionAlertDialog != null &&
permissionAlertDialog.isShowing()){
            permissionAlertDialog.dismiss();
            permissionAlertDialog = null;
        }
        Constant.SHOW_SPLASH_AD = false;
        fire(new ActionEvent(CHECK_UPGRADE));
    }

    private boolean isHasShowPrivacy(){
        FileInputStream fis = null;
        byte[] buffer = null;
        try {
            //获取文件输入流
            fis = openFileInput("privacy");
            //定义保存数据的数组
            buffer = new byte[fis.available()];
            //从输入流中读取数据
            fis.read(buffer);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try{
                if(fis!=null)fis.close();
                return buffer != null;
            }catch(IOException e){
                e.printStackTrace();
            }
        }
        return false;
    }

    private void writeToApp(String data){
        FileOutputStream out = null;
        try {
            out = openFileOutput("privacy",MODE_PRIVATE);
            out.write(data.getBytes());
```

```
                    out.flush();// 清理缓冲区的数据流
                    out.close();// 关闭输出流
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        package com.frostsowner.magic.weather.fragment;

        import android.content.Intent;
        import android.os.Build;
        import android.os.Bundle;
        import android.speech.tts.UtteranceProgressListener;
        import android.view.View;

        import androidx.annotation.NonNull;
        import androidx.recyclerview.widget.LinearLayoutManager;
        import androidx.recyclerview.widget.RecyclerView;

        import com.chad.library.adapter.base.entity.MultiItemEntity;
        import com.fm.commons.http.ContextHolder;
        import com.fm.commons.logic.BeanFactory;
        import com.fm.commons.logic.LocalStorage;
        import com.scwang.smartrefresh.layout.SmartRefreshLayout;
        import com.scwang.smartrefresh.layout.api.RefreshLayout;
        import com.scwang.smartrefresh.layout.listener.OnRefreshListener;
        import com.frostsowner.magic.weather.Constant;
        import com.frostsowner.magic.weather.R;
        import com.frostsowner.magic.weather.TrackPlugin;
        import com.frostsowner.magic.weather.activity.WanYearActivity;
        import com.frostsowner.magic.weather.adapter.CityWeatherAdapter;
        import com.frostsowner.magic.weather.domain.AdViewItem;
        import com.frostsowner.magic.weather.domain.AqiItem;
        import com.frostsowner.magic.weather.domain.BaseWeather;
        import com.frostsowner.magic.weather.domain.ConditionItem;
        import com.frostsowner.magic.weather.domain.ForecastItem;
        import com.frostsowner.magic.weather.domain.HourlyItem;
        import com.frostsowner.magic.weather.domain.LiveIndexItem;
        import com.frostsowner.magic.weather.domain.TopAlertItem;
        import com.frostsowner.magic.weather.domain.TwoDaysItem;
        import com.frostsowner.magic.weather.domain.WanYearItem;
        import com.frostsowner.magic.weather.domain.weather.CityWeather;
        import com.frostsowner.magic.weather.domain.yunyida.YunCondition;
        import com.frostsowner.magic.weather.domain.yunyida.YunWeather;
        import com.frostsowner.magic.weather.event.ActionEvent;
        import com.frostsowner.magic.weather.event.EventType;
        import com.frostsowner.magic.weather.impl.AdapterNotifyListener;
        import com.frostsowner.magic.weather.impl.RefreshStatusListener;
```

```
import com.frostsowner.magic.weather.logic.CityManager;
import com.frostsowner.magic.weather.logic.SimpleDateManager;
import com.frostsowner.magic.weather.logic.TTSManager;
import com.frostsowner.magic.weather.logic.UserManager;
import com.frostsowner.magic.weather.logic.WeatherItemExposure;
import com.frostsowner.magic.weather.logic.WeatherRefreshHandler;
import com.frostsowner.magic.weather.service.WeatherService;
import com.frostsowner.magic.weather.utils.ConnectionUtils;
import com.frostsowner.magic.weather.utils.StringUtils;
import com.frostsowner.magic.weather.utils.SubscribeUtils;
import com.frostsowner.magic.weather.widget.CustomRefreshHeader;

import org.greenrobot.eventbus.Subscribe;
import org.greenrobot.eventbus.ThreadMode;

import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;

import butterknife.BindView;
import rx.Subscription;
import rx.functions.Action0;

import static com.frostsowner.magic.weather.Constant.CONDITION_15_SHOW;
import static com.frostsowner.magic.weather.Constant.CONDITION_24_SHOW;
import static
com.frostsowner.magic.weather.Constant.CONDITION_LIVE_SHOW;
import static
com.frostsowner.magic.weather.Constant.CONDITION_NEWS_STICKY;
import static com.frostsowner.magic.weather.Constant.CONDITION_REFRESH;
import static
com.frostsowner.magic.weather.Constant.CONDITION_TODAY_SHOW;
import static
com.frostsowner.magic.weather.Constant.CONDITION_TTS_CLICK;
import static
com.frostsowner.magic.weather.Constant.CONDITION_YEAR_CLICK;
import static
com.frostsowner.magic.weather.Constant.DRAWING_UPDATE_OVERTIME;
import static com.frostsowner.magic.weather.Constant.NEWS_VIEW_SHOW;
import static
com.frostsowner.magic.weather.domain.BaseWeather.TYPE_BANNER_1;
import static
com.frostsowner.magic.weather.domain.BaseWeather.TYPE_BANNER_2;
import static
com.frostsowner.magic.weather.domain.BaseWeather.TYPE_NATIVE;
import static
com.frostsowner.magic.weather.event.EventType.WEATHER_REQUEST_SUCCESS;

public class CityWeatherFragment extends BaseAutoHeightFragment
```

```
        implements OnRefreshListener, AdapterNotifyListener ,
        RefreshStatusListener {

            public static CityWeatherFragment newInstance(String cityName){
                Bundle args = new Bundle();
                args.putString("cityName",cityName);
                CityWeatherFragment fragment = new CityWeatherFragment();
                fragment.setArguments(args);
                return fragment;
            }

            private final long UPDATE_DELAY = 1000*60*30;
            private final int NEWS_STICKY_OFFSET = 100;

            @BindView(R.id.swipe_container)
            SmartRefreshLayout swipeRefreshLayout;
            @BindView(R.id.weather_recyclerview)
            RecyclerView weatherRecyclerView;
            @BindView(R.id.background_view)
            View backgroundView;

            private String cityName;
            private CityManager cityManager;
            private SimpleDateManager simpleDateManager;
            private CityWeatherAdapter cityWeatherAdapter;
            private YunWeather cityWeather;
            private TTSManager ttsManager;
            private LocalStorage localStorage;
            private UserManager userManager;
            private WeatherItemExposure weatherItemExposure;
            private long updateTime;
            private boolean isSpeaking;
            private boolean forceLocation;
            private WeatherRefreshHandler weatherRefreshHandler;

            @Override
            public void onCreate(Bundle savedInstanceState){
                super.onCreate(savedInstanceState);
                Bundle bundle = getArguments();
                if(bundle != null){
                    cityName = bundle.getString("cityName");
                    logError("bundle cityName : "+cityName);
                }
            }

            @Override
            protected void initWidget(View root){
                super.initWidget(root);
                simpleDateManager =
        BeanFactory.getBean(SimpleDateManager.class);
```

```
            cityManager = BeanFactory.getBean(CityManager.class);
            weatherItemExposure =
        BeanFactory.getBean(WeatherItemExposure.class);
            localStorage = BeanFactory.getBean(LocalStorage.class);
            weatherRefreshHandler =
        BeanFactory.getBean(WeatherRefreshHandler.class);
            userManager = BeanFactory.getBean(UserManager.class);
            weatherItemExposure.init();
            cityWeatherAdapter = new CityWeatherAdapter(getActivity());
            cityWeatherAdapter.setTopViewHeight(getCenterViewHeight());
            cityWeatherAdapter.setAdapterNotifyListener(this);
            LinearLayoutManager layoutManager = new
        LinearLayoutManager(getContext()){
                @Override
                public boolean canScrollVertically(){
                    return !CONDITION_NEWS_STICKY;
                }
            };
            cityWeatherAdapter.setOnItemChildClickListener((adapter, view,
        position) -> {
                if(view.getId() == R.id.btn_speak){
                    checkSpeak();
                }
                if(view.getId() == R.id.simple_fate_root){
                    startActivity(new Intent(getContext(),
        WanYearActivity.class));

        TrackPlugin.track(getContext(),CONDITION_YEAR_CLICK,null);
                }
            });
            layoutManager.setOrientation(LinearLayoutManager.VERTICAL);
            layoutManager.setSmoothScrollbarEnabled(true);
            weatherRecyclerView.setLayoutManager(layoutManager);
            weatherRecyclerView.setAdapter(cityWeatherAdapter);
            swipeRefreshLayout.setOnRefreshListener(this);
            swipeRefreshLayout.setRefreshHeader(new
        CustomRefreshHeader(getContext(),this));
            swipeRefreshLayout.setEnableLoadMore(false);
            weatherRecyclerView.addOnScrollListener(new
        RecyclerView.OnScrollListener(){
                @Override
                public void onScrollStateChanged(@NonNull RecyclerView
        recyclerView, int newState){
                    super.onScrollStateChanged(recyclerView, newState);
                    itemViewShowReport();
                }

                @Override
                public void onScrolled(@NonNull RecyclerView recyclerView,
        int dx, int dy) {
```

```
                super.onScrolled(recyclerView, dx, dy);
                setScoreView();
//                newsSticky();
            }
        });

weatherRecyclerView.getViewTreeObserver().addOnGlobalLayoutListener(()
-> {
            if(weatherRecyclerView !=
null)weatherRecyclerView.stopScroll();
        });
        initTTS();
    }

    private void itemViewShowReport(){
        LinearLayoutManager linearLayoutManager =
(LinearLayoutManager)weatherRecyclerView.getLayoutManager();
        int position =
linearLayoutManager.findLastCompletelyVisibleItemPosition()+1;
//        Log.e("ijimu","city weather fragment position : " +
position);
        if(position == 2){
            if(weatherItemExposure.exposure(position)){

TrackPlugin.track(getContext(),CONDITION_TODAY_SHOW,null);
                logError("track condition");
            }
        }
        if(position == 4){
            if(weatherItemExposure.exposure(position)){
                TrackPlugin.track(getContext(),CONDITION_24_SHOW,null);
                logError("track 24 hourly");
            }
        }
        if(position == 6){
            if(weatherItemExposure.exposure(position)){
                TrackPlugin.track(getContext(),CONDITION_15_SHOW,null);
                logError("track 15 forecast");
            }
        }
        if(position == 10){
            if(weatherItemExposure.exposure(position)){

TrackPlugin.track(getContext(),CONDITION_LIVE_SHOW,null);
                logError("track live index");
            }
        }
    }

    private void newsSticky(){
```

```
            LinearLayoutManager linearLayoutManager = (LinearLayoutManager)
        weatherRecyclerView.getLayoutManager();
            int position =
        linearLayoutManager.findLastVisibleItemPosition();
            if(position == cityWeatherAdapter.getItemCount()-1){
                View view =
        linearLayoutManager.findViewByPosition(position);
                if(view != null && view.getTop() <= NEWS_STICKY_OFFSET){
                    CONDITION_NEWS_STICKY = true;
                    Map<String,String> params = new HashMap<>();
                    params.put("position","首页");
                    TrackPlugin.track(getContext(),NEWS_VIEW_SHOW,params);
                    logError("track news show");
                    fire(new
        ActionEvent(EventType.CONDITION_NEWS_STICKY_SHOW));
                    updateNewsStatus();
                }
            }
        }

        private void updateNewsStatus(){
            LinearLayoutManager linearLayoutManager = (LinearLayoutManager)
        weatherRecyclerView.getLayoutManager();

        linearLayoutManager.scrollToPositionWithOffset(CONDITION_NEWS_STICKY?ci
        tyWeatherAdapter.getItemCount()-1:0,0);

        weatherRecyclerView.setNestedScrollingEnabled(CONDITION_NEWS_STICKY);
            swipeRefreshLayout.setEnableRefresh(!CONDITION_NEWS_STICKY);
        }

        private void setScoreView(){
            int offsetY =
        weatherRecyclerView.computeVerticalScrollOffset();
            float alpha = 0;
            if(offsetY <= 0){
                alpha = 0;
            }
            else if(offsetY < getCenterViewHeight()){
                float scale = offsetY*1f/getCenterViewHeight();
                alpha = 1f*scale;
            }else{
                alpha = 1f;
            }
            backgroundView.setAlpha(alpha);
            ActionEvent event = new
        ActionEvent(EventType.WEATHER_VIEW_SCROLLING);
            event.setAttr("alpha",alpha);
            fire(event);
        }
```

```
    private void updateUI(boolean force){
        if(force || cityWeather == null){
            cityWeather = cityManager.getTarget(cityName);
            if(cityWeather != null &&
cityWeather.getCondition()!=null&&cityWeather.getForecast()!=null){
                List<MultiItemEntity> data = new LinkedList<>();
                data.add(createTopAlertItem(cityWeather));
                data.add(createConditionItem(cityWeather));
                data.add(createTwoDayItem(cityWeather));
                data.add(createHourlyItem(cityWeather));
                if(showLoadAd())data.add(new
AdViewItem(TYPE_BANNER_1));
                data.add(createForecastItem(cityWeather));
                if(showLoadAd())data.add(new AdViewItem(TYPE_NATIVE));
                data.add(createAqiItem(cityWeather));
//
if(showLoadAd())addData(data,createWanYearItem(cityWeather));
                data.add(createLiveIndexItem(cityWeather));
                if(showLoadAd())data.add(new
AdViewItem(TYPE_BANNER_2));
//                if(showLoadAd()&&Constant.AB_TEST_NEWS)data.add(new
AdViewItem(TYPE_NEWS));
                cityWeatherAdapter.setNewData(data);
                updateTime = System.currentTimeMillis();
                logInfo("updateUI force");
            }
        }else{
            cityWeatherAdapter.notifyDataSetChanged();
            logInfo("updateUI not force");
        }
    }

    private void addData(List<MultiItemEntity> data, BaseWeather item){
        if(item == null)return;
        data.add(item);
    }

    @Override
    public void onRefresh(@NonNull RefreshLayout refreshLayout){
        if(!ConnectionUtils.isConnectionAvailable(getContext())){
            showNotice("网络断开,请检查网络是否正确连接");
            return;
        }
        TrackPlugin.track(getContext(),CONDITION_REFRESH,null);
        boolean showFakeData =
weatherRefreshHandler.showFakeData(cityName);
        if(showFakeData){
            SubscribeUtils.doOnUIThreadDelayed(() -> {
                ActionEvent actionEvent = new
```

```
ActionEvent(WEATHER_REQUEST_SUCCESS);
                    actionEvent.setAttr("cityName",cityName);
                    actionEvent.setAttr("fake","fake");
                    fire(actionEvent);
                },500);
                logInfo("city weather fragment show fake data");
            }else{
                Intent intent = new Intent(getContext(),
WeatherService.class);
                intent.putExtra("cityName",cityName);
                if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
                    getContext().startForegroundService(intent);
                }else{
                    getContext().startService(intent);
                }
            }
            fire(new ActionEvent(EventType.WEATHER_REFRESHING));
        }

        @Override
        public void onNotify(){
            logError("cityWeatherFragment notify , cityName : "+cityName);
            itemViewShowReport();
        }

        @Subscribe(threadMode = ThreadMode.MAIN)
        public void onDataUpdate(ActionEvent event){
            if(event.getType() == WEATHER_REQUEST_SUCCESS){
                String cityName = event.getAttrStr("cityName");
                String fake = event.getAttrStr("fake");
                String reset = event.getAttrStr("reset");
                logInfo("CityWeatherFragment, cityName : "+cityName+" ,
fake : "+fake+" , reset : "+reset);
                YunWeather target = cityManager.getTarget(cityName);
                if(cityWeather!=null && cityWeather.isLocate() && target !=
null && target.isLocate()){
                    this.cityName = target.getCityName();
                }
                int currentPosition =
cityManager.getTargetIndex(this.cityName);
                int targetPosition = cityManager.getCurrentCityIndex();
                if(currentPosition != targetPosition)return;
                updateUI(StringUtils.isEmpty(fake));
                swipeRefreshLayout.finishRefresh(true);
            }
            else if(event.getType() ==
EventType.CONDITION_NEWS_STICKY_RELEASE){
                CONDITION_NEWS_STICKY = false;
                updateNewsStatus();
            }
```

```
            cityManager.getTargetIndex(this.cityName);
            cityManager.getCurrentCityIndex();
                    else if(event.getType() == EventType.CITY_WEATHER_SPEAK){
                        String targetCityName = event.getAttrStr("cityName");
                        int currentPosition =
            cityManager.getTargetIndex(this.cityName);
                        int targetPosition =
            cityManager.getTargetIndex(targetCityName);
                        if(currentPosition == targetPosition)return;
                        logInfo("city weather speak , current cityName
            "+cityName+" , target cityName : "+targetCityName);
                        isSpeaking = false;
                        if(ttsManager != null && ttsManager.isInit()){
                            ttsManager.stop();
                        }
                        cityWeatherAdapter.speakStatus(false);
                    }
                    else if(event.getType() == EventType.LOCATION_START){
                        int currentPosition =
            cityManager.getTargetIndex(this.cityName);
                        if(currentPosition == 0){
                            forceLocation = true;
                        }
                    }
                }

                @Override
                public void onResume() {
                    super.onResume();
                    if((cityWeather!= null && cityWeather.isLocate() &&
            System.currentTimeMillis() - updateTime >= UPDATE_DELAY) ||
                        updateTime == 0L || forceLocation){ // || !cityFakeRefresh
                        swipeRefreshLayout.autoRefresh();
                        addSubscription(SubscribeUtils.doOnUIThreadDelayed(new
            Action0(){
                            @Override
                            public void call(){
                                if(!swipeRefreshLayout.getState().isFinishing){
                                    swipeRefreshLayout.finishRefresh(false);
                                }
                            }
                        }, DRAWING_UPDATE_OVERTIME));
                        forceLocation = false;
                    }
                    initTTS();
                    updateUI(false);
                    setScoreView();
                    if(cityWeatherAdapter != null)cityWeatherAdapter.onResume();
                    logInfo("onResume , cityName : "+cityName);
                }
```

```
    @Override
    public void onPause() {
        super.onPause();
        if(cityWeatherAdapter != null)cityWeatherAdapter.onPause();
    }

    @Override
    public void refresh(int status){

    }

    private void initTTS(){
        if(ttsManager == null){
            ttsManager = new TTSManager(ContextHolder.get());
            ttsManager.init();
            ttsManager.addListener(new UtteranceProgressListener() {
                @Override
                public void onStart(String utteranceId) {
                    logInfo("tts speak start");
                    Subscription subscription =
SubscribeUtils.doOnUIThread(new Action0() {
                        @Override
                        public void call() {
                            if(cityWeatherAdapter !=
null)cityWeatherAdapter.speakStatus(true);
                        }
                    });
                    addSubscription(subscription);
                }

                @Override
                public void onDone(String utteranceId){
                    logInfo("tts speak done");
                    Subscription subscription =
SubscribeUtils.doOnUIThreadDelayed(new Action0() {
                        @Override
                        public void call() {
                            if(cityWeatherAdapter !=
null)cityWeatherAdapter.speakStatus(false);
                        }
                    },1000);
                    addSubscription(subscription);
                    ttsManager.stop();
                }

                @Override
                public void onError(String utteranceId) {
                    Subscription subscription =
SubscribeUtils.doOnUIThreadDelayed(new Action0() {
```

```
                            @Override
                            public void call() {
                                if(cityWeatherAdapter !=
null)cityWeatherAdapter.speakStatus(false);
                            }
                        },1000);
                        addSubscription(subscription);
                        ttsManager.stop();
                        logInfo("tts speak error : "+utteranceId);
                    }
                });
                if(cityWeatherAdapter !=
null)cityWeatherAdapter.speakStatus(false);
                isSpeaking = false;
            }
        }

        private void checkSpeak(){
            if(!ttsManager.isInit()){
                showNotice("语音播报初始化未成功");
                return;
            }
            if(cityManager.getCurrentCityWeather() == null){
                showNotice("等待当前天气数据加载...");
                return;
            }
            if(isSpeaking){
                ttsManager.stop();
                cityWeatherAdapter.speakStatus(false);
            }else{
                speakWeather(cityManager.getCurrentCityWeather());
                cityWeatherAdapter.speakStatus(true);
            }
            isSpeaking = !isSpeaking;
            TrackPlugin.track(getContext(),CONDITION_TTS_CLICK,null);
        }

        private void speakWeather(YunWeather cityWeather){
            if(cityWeather == null)return;
            StringBuffer buffer = new StringBuffer();
            buffer.append("天天提醒您,");
            buffer.append(cityWeather.getCityName());

            if(cityWeather.getCondition() != null){

                YunCondition baseCondition = cityWeather.getCondition();
                buffer.append("今日最高气温"+baseCondition.getTem1()+"度
"+",最低气温"+baseCondition.getTem2()+"度,");
                buffer.append("当前天气"+baseCondition.getWea());
                buffer.append(",气温"+baseCondition.getTem()+"度,");
```

```
buffer.append(baseCondition.getWin()+baseCondition.getWin_speed()+",");
            buffer.append(baseCondition.getAir_tips());
        }
        ActionEvent speakEvent = new
ActionEvent(EventType.CITY_WEATHER_SPEAK);
        speakEvent.setAttr("cityName",cityWeather.getCityName());
        fire(speakEvent);

        ttsManager.speak(buffer.toString());
//        ActionEvent event = new ActionEvent(EventType.TASK_COMPLETE);
//        event.setAttr("taskType", TaskType.AJ_TASK_TYPE_TTS);
//        fire(event);
        logInfo("自动转换天气信息 : "+buffer.toString());
        //"天天提醒您,地址当前天气晴，气温 29 度，今日最高气温 29 度最低
气温 17 度，西南风 3 级，略微偏热，注意衣物变化，紫外线强不适宜户外运动。
车辆尾号限行 0，5"
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        if(ttsManager != null){
            ttsManager.shutdown();
            ttsManager = null;
        }
    }

    private TopAlertItem createTopAlertItem(YunWeather cityWeather){
        TopAlertItem item = new TopAlertItem();
        if(cityWeather.getCondition() != null &&
cityWeather.getCondition().getAlarm() != null){
            item.setYunAlert(cityWeather.getCondition().getAlarm());
        }
        return item;
    }

    private ConditionItem createConditionItem(YunWeather cityWeather){
        ConditionItem item = new ConditionItem();
        if(cityWeather.getCondition() != null){
            item.setYunCondition(cityWeather.getCondition());
        }
        return item;
    }

    private TwoDaysItem createTwoDayItem(YunWeather cityWeather){
        TwoDaysItem item = new TwoDaysItem();
        if(cityWeather.getForecast() != null &&
            cityWeather.getForecast().getData() != null &&
            cityWeather.getForecast().getData().size() > 0){
```

```
                item.setForecastItems(cityWeather.getForecast().getData());
            }
            return item;
        }

        private HourlyItem createHourlyItem(YunWeather cityWeather){
            HourlyItem item = new HourlyItem();
            if(cityWeather.getCondition() != null &&
                cityWeather.getCondition().getHours() != null &&
                cityWeather.getCondition().getHours().size() > 0){
                item.setHours(cityWeather.getCondition().getHours());
            }
            if(cityWeather.getCondition() != null){
                item.setSunrise(cityWeather.getCondition().getSunrise());
                item.setSunset(cityWeather.getCondition().getSunset());
            }
            return item;
        }

        private ForecastItem createForecastItem(YunWeather cityWeather){
            ForecastItem item = new ForecastItem();
            if(cityWeather.getForecast() != null &&
                cityWeather.getForecast().getData() != null &&
                cityWeather.getForecast().getData().size() > 0){
                item.setForecastItems(cityWeather.getForecast().getData());
            }
            return item;
        }

        private AqiItem createAqiItem(YunWeather cityWeather){
            AqiItem item = new AqiItem();
            if(cityWeather.getCondition() != null &&
    cityWeather.getCondition().getAqi() != null){
                item.setYunAqi(cityWeather.getCondition().getAqi());
            }
            return item;
        }

        private LiveIndexItem createLiveIndexItem(YunWeather cityWeather){
            LiveIndexItem item = new LiveIndexItem();
            if(cityWeather.getCondition() != null &&
    cityWeather.getCondition().getZhishu()!= null){

    item.setYunLiveIndex(cityWeather.getCondition().getZhishu());
            }
            return item;
        }

        private WanYearItem createWanYearItem(CityWeather cityWeather){
            if(cityWeather.getWanYearData() != null){
```

```
                WanYearItem item = new WanYearItem();

item.setBaseWanYear(cityWeather.getWanYearData().getContent());
                return item;
            }
            return null;
        }

        private boolean showLoadAd(){
            if(Constant.isEmulator)return false;
            if(userManager.isLogin()&&userManager.isVip())return false;
            return Constant.SHOW_AD;
        }

        @Override
        protected int getLayout(){
            return R.layout.fragment_city_weather;
        }
    }
    package com.frostsowner.magic.weather.service;

    import android.content.Context;
    import android.content.Intent;
    import android.os.PowerManager;
    import android.text.TextUtils;
    import com.frostsowner.magic.weather.utils.SubscribeUtils;

    import org.greenrobot.eventbus.EventBus;
    import org.slf4j.Logger;
    import org.slf4j.LoggerFactory;

    import java.lang.reflect.Type;

    import rx.Observable;
    import rx.Subscription;
    import rx.android.schedulers.AndroidSchedulers;
    import rx.functions.Action0;
    import rx.functions.Action1;
    import rx.functions.Func0;
    import rx.functions.Func1;
    import rx.schedulers.Schedulers;

    import static com.frostsowner.magic.weather.Constant.ACTION_WIDGET;
    import static
    com.frostsowner.magic.weather.Constant.CONDITION_UPDATE_INTERVAL;
    import static
    com.frostsowner.magic.weather.Constant.SHORT_FORECAST_UPDATE_INTERVAL;

    public class WeatherService extends BaseWeatherService {

```

```
    private Logger logger = LoggerFactory.getLogger(getClass());

    private WeatherRequestManager requestManager;
    private CityManager cityManager;

    private boolean isRequesting = false;
    private PowerManager pm;

    @Override
    public void onCreate() {
        super.onCreate();
        requestManager =
BeanFactory.getBean(WeatherRequestManager.class);
        cityManager = BeanFactory.getBean(CityManager.class);
        pm = (PowerManager)getSystemService(Context.POWER_SERVICE);
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        startForeground();
        if(intent != null){
            String cityName = intent.getStringExtra("cityName");
            String action = intent.getStringExtra("notification");
            String widget = intent.getStringExtra("action");
            logInfo("get widget action : "+widget);
            boolean showNoticeAlert =
TextUtils.equals(widget,ACTION_WIDGET);
            if(StringUtils.isEmpty(cityName)){
                location(showNoticeAlert);
            }else{
                YunWeather cityWeather =
cityManager.getTarget(cityName);
                if(cityWeather == null){
                    cityWeather = cityManager.addData(cityName);
                    requestMJData(cityWeather,showNoticeAlert);
                }else{
                    if(cityWeather.isLocate()){
                        location(showNoticeAlert);
                    }else{
                        requestMJData(cityWeather,showNoticeAlert);
                    }
                }
            }
        }
        addSubscription(SubscribeUtils.doOnUIThreadDelayed(new
Action0(){
            @Override
            public void call() {
                isRequesting = false;
            }
```

```
            },1000*30));
            return super.onStartCommand(intent, flags, startId);
        }

        private void location(boolean showNoticeAlert){
            WeatherLocationClient.get().locate(new
    WeatherLocationClient.BDLocationCallBack(){
                @Override
                public void locate(BDLocation bdLocation){
                    if(bdLocation == null|| Constant.isEmulator){
                        YunWeather cityWeather = cityManager.getTarget(0);
                        if(cityWeather == null){
                            cityWeather = new YunWeather();
                            cityWeather.setCity("北京");
                            cityWeather.setLocate(true);
                            cityManager.addData("北京",true);
                        }
                        requestMJData(cityWeather,showNoticeAlert);
                        return;
                    }
                    String city = bdLocation.getCity();
                    String district = bdLocation.getDistrict();
                    String address = bdLocation.getAddrStr();
                    String semaAptag = bdLocation.getSemaAptag();
                    logInfo("locate log cityName : "+city+district);
                    YunWeather cityWeather =
    cityManager.getTarget(city+district);
                    if(cityWeather == null){
                        if(StringUtils.isEmpty(city)){
                            cityWeather = cityManager.addData("北京",true);
                        }else{
                            cityWeather =
    cityManager.addData(city,district,address,semaAptag,true);
                        }
                    }else{

    if(cityWeather.getCity().equals(city)&&cityWeather.getDistrict().equals
    (district)){
                            cityWeather.setCity(city);
                            cityWeather.setDistrict(district);
                            cityWeather.setAddress(address);
                            cityWeather.setSemaAptag(semaAptag);
                        }else{
                            cityWeather =
    cityManager.addData(city,district,address,semaAptag,true);
                        }
                    }
                    requestMJData(cityWeather,showNoticeAlert);
                }

```

```
                    @Override
                    public void failed(){
                        YunWeather cityWeather = cityManager.getTarget(0);
                        if(cityWeather == null){
                            cityWeather = cityManager.addData("北京",true);
                        }
                        ActionEvent actionEvent = new
ActionEvent(EventType.WEATHER_REQUEST_FAILED);
                        EventBus.getDefault().post(actionEvent);
                        requestMJData(cityWeather,showNoticeAlert);
                    }
                });
            }

            protected void requestMJData(YunWeather cityWeather,final boolean
showNoticeAlert){
                if(isRequesting){
                    logger.info("正在更新,无需重复请求");
                    return;
                }
                isRequesting = true;
                logInfo("请求天气数据 : "+cityWeather.getCity()+" ,
"+cityWeather.getDistrict()+" , "+cityWeather.getAddress()+" , cityId :
"+cityWeather.getCityId());
                Subscription subscription = Observable.defer(new
Func0<Observable<YunWeather>>() {
                    @Override
                    public Observable<YunWeather> call() {
                        return Observable.just(cityWeather);
                    }
                })
                .subscribeOn(Schedulers.io())
                .flatMap(new Func1<YunWeather, Observable<YunWeather>>(){
                    @Override
                    public Observable<YunWeather> call(YunWeather mjWeather) {
                        return requestCondition(mjWeather);
                    }
                })
                .flatMap(new Func1<YunWeather, Observable<YunWeather>>(){
                    @Override
                    public Observable<YunWeather> call(YunWeather mjWeather) {
                        return requestForecast(mjWeather);
                    }
                })
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(new Action1<YunWeather>() {
                    @Override
                    public void call(YunWeather cityData) {
                        cityManager.updateData(cityData);
                        updateWeatherNotification();
```

```
                    if(pm != null && pm.isInteractive()){
                        ActionEvent actionEvent = new
ActionEvent(EventType.WEATHER_REQUEST_SUCCESS);
                        actionEvent.setAttr("cityName",
cityWeather.getCityName());
                        actionEvent.setAttr("action",showNoticeAlert);
                        EventBus.getDefault().post(actionEvent);
                        logger.info("刷新天气");
                    }
                    logger.info("成功、 天气");
                    isRequesting = false;
                }
            }, new Action1<Throwable>() {
                @Override
                public void call(Throwable throwable) {
                    ActionEvent actionEvent = new
ActionEvent(EventType.WEATHER_REQUEST_FAILED);
                    actionEvent.setAttr("cityName",
cityWeather.getCityName());
                    EventBus.getDefault().post(actionEvent);
                    logger.info("失败、天气");
                    throwable.printStackTrace();
                    CrashHandler.getInstance().handleException(throwable);
                    isRequesting = false;
                }
            });
            addSubscription(subscription);
        }

        private Observable<YunWeather> requestCondition(YunWeather
cityWeather){
            YunCondition yunCondition = cityWeather.getCondition();
            if(yunCondition !=null && System.currentTimeMillis() -
yunCondition.getLastRequestTime() < CONDITION_UPDATE_INTERVAL){
                logger.info("do not need update conditionData data");
                return Observable.just(cityWeather);
            }
            String result =
requestManager.getYunYiDaWeather("v62",cityWeather.getCityId());
//            String result2 =
requestManager.getCondition(cityWeather.getCityId());
            logger.info("update condition data :  ");
//            logger.info("update conditionData data2 :  "+result2);
            Type type = new TypeToken<YunCondition>(){}.getType();
            YunCondition data = ResultUtil.getDomain(result, type);
            if (data != null) {
                data.setLastRequestTime(System.currentTimeMillis());
                cityWeather.setCondition(data);
                cityWeather.setUpdateTime(System.currentTimeMillis());
                TrackPlugin.track(this,"weatherApi",null);
```

```
            }
            return Observable.just(cityWeather);
        }

        private Observable<YunWeather> requestForecast(YunWeather
    cityWeather){
            YunForecast yunForecast = cityWeather.getForecast();
            if(yunForecast !=null && System.currentTimeMillis() -
    yunForecast.getLastRequestTime() < SHORT_FORECAST_UPDATE_INTERVAL){
                logger.info("do not need update forecast data");
                return Observable.just(cityWeather);
            }
            String result =
    requestManager.getYunYiDaWeather("v3",cityWeather.getCityId());
            logger.info("update forecast data : ");
            Type type = new TypeToken<YunForecast>(){}.getType();
            YunForecast data = ResultUtil.getDomain(result, type);
            if (data != null) {
                data.setLastRequestTime(System.currentTimeMillis());
                cityWeather.setForecast(data);
                cityWeather.setUpdateTime(System.currentTimeMillis());
                TrackPlugin.track(this,"weatherApi",null);
            }
            return Observable.just(cityWeather);
        }
    }
    package com.frostsowner.magic.weather.adapter;

    import android.content.Intent;
    import android.content.res.Resources;
    import android.graphics.Color;
    import android.graphics.Typeface;
    import android.graphics.drawable.Drawable;
    import android.os.Build;
    import android.util.DisplayMetrics;
    import android.util.Log;
    import android.util.SparseArray;
    import android.util.SparseBooleanArray;
    import android.view.View;
    import static
    com.frostsowner.magic.weather.domain.BaseWeather.TYPE_CONDITION;
     com.frostsowner.magic.weather.domain.BaseWeather.TYPE_TWO_DAYS;

    public class CityWeatherAdapter extends AtAdFragmentPagerAdapter {

        private int topViewHeight;

        private SparseArray<WeatherHourlyAdapter> hourlyAdapterArray;
        private SparseArray<LiveIndexAdapter> liveIndexAdapterArray;
        private SparseArray<WeatherWeekAdapter> forecastAdapterArray;
```

```
    private SparseBooleanArray forceUpdate;

    private SimpleDateFormat dateFormat;
    private SimpleDateFormat itemDayFormat;

    private boolean hourlySlip = false;
    private boolean forecastSlip = false;
    private NewsView newsView;
    private SpeakHolder speakHolder;

    private AdapterNotifyListener adapterNotifyListener;
    private NewsTypeManager newsTypeManager;
    private SimpleDateManager simpleDateManager;
    private UserManager userManager;

    public CityWeatherAdapter(FragmentActivity activity){
        super(activity);
        addItemType(TYPE_TOP_ALERT, R.layout.item_alert_top_view);
        addItemType(TYPE_CONDITION,R.layout.item_condition_view);
        addItemType(TYPE_TWO_DAYS,R.layout.item_two_days_view);
        addItemType(TYPE_24_HOURLY,R.layout.item_24_hourly_view);
        addItemType(TYPE_FORECAST,R.layout.item_forecast_view);
        addItemType(TYPE_AQI,R.layout.item_aqi_view);
        addItemType(TYPE_BANNER_1,R.layout.item_ad_banner_1_view);
        addItemType(TYPE_CALENDAR,R.layout.layout_simple_fate_1);
        addItemType(TYPE_BANNER_2,R.layout.item_ad_banner_2_view);
        addItemType(TYPE_LIVE,R.layout.item_live_index_view);
        addItemType(TYPE_NATIVE,R.layout.item_ad_native_view);
//
if(Constant.AB_TEST_NEWS)addItemType(TYPE_NEWS,com.shl.weather.baidu.R.
layout.layout_news);
        hourlyAdapterArray = new SparseArray<>();
        liveIndexAdapterArray = new SparseArray<>();
        forecastAdapterArray = new SparseArray<>();
        forceUpdate = new SparseBooleanArray();
        simpleDateManager =
BeanFactory.getBean(SimpleDateManager.class);
        newsTypeManager = BeanFactory.getBean(NewsTypeManager.class);
        userManager = BeanFactory.getBean(UserManager.class);
        dateFormat = simpleDateManager.getFormatMap("yyyy-MM-dd");
        itemDayFormat = simpleDateManager.getFormatMap("MM 月 dd 日");
    }

    public void setTopViewHeight(int value){
        this.topViewHeight = value;
    }

    public void setAdapterNotifyListener(AdapterNotifyListener
adapterNotifyListener){
        this.adapterNotifyListener = adapterNotifyListener;
```

```
        }

        @Override
        public void setNewData(@Nullable List<MultiItemEntity> data) {
            forceUpdate.clear();
            for(int i = 0; i < data.size();i++){
                int itemType = data.get(i).getItemType();
                forceUpdate.put(itemType,true);
            }
            super.setNewData(data);
        }

        @Override
        protected void convert(BaseViewHolder helper, MultiItemEntity
    item){
    //        Log.d("ijimu","convert, type : "+helper.getItemViewType());
            switch (helper.getItemViewType()){
                case TYPE_TOP_ALERT:
                    updateTopAlert(helper,item);
                    break;
                case TYPE_CONDITION:
                    updateCondition(helper,item);
                    break;
                case TYPE_TWO_DAYS:
                    updateTwoDays(helper,item);
                    break;
                case TYPE_24_HOURLY:
                    update24Hourly(helper,item);
                    break;
                case TYPE_FORECAST:
                    updateForecast(helper,item);
                    break;
                case TYPE_AQI:
                    updateAqi(helper,item);
                    break;
                case TYPE_CALENDAR:
                    updateWanYearView(helper,item);
                    break;
                case TYPE_LIVE:
                    updateLiveIndex(helper,item);
                    break;
                case TYPE_BANNER_1:
                    updateBanner1(helper);
                    break;
                case TYPE_BANNER_2:
                    updateBanner2(helper);
                    break;
                case TYPE_NATIVE:
                    updateNative(helper);
                    break;
```

```
                    case TYPE_NEWS:
                        updateNews(helper);
                        break;
                }
                if(adapterNotifyListener !=
            null)adapterNotifyListener.onNotify();
            }

            private void updateTopAlert(BaseViewHolder helper, MultiItemEntity
            item){
                boolean force = forceUpdate.get(item.getItemType());
                if(!force)return;
                TopAlertItem topAlertItem = (TopAlertItem)item;
                if(topViewHeight != 0){
                    LinearLayout topView = helper.getView(R.id.top_view);
                    RecyclerView.LayoutParams layoutParams =
            (RecyclerView.LayoutParams)topView.getLayoutParams();
                    layoutParams.height = topViewHeight;
                    topView.setLayoutParams(layoutParams);
                }

                TextView tvAlert = helper.getView(R.id.tv_alert);
                if(topAlertItem.getYunAlert() != null
            && !StringUtils.isEmpty(topAlertItem.getYunAlert().getAlarm_content()))
            {
                    tvAlert.setVisibility(View.VISIBLE);

            tvAlert.setText(topAlertItem.getYunAlert().getAlarm_content());
                    tvAlert.setSelected(true);
                    tvAlert.setOnClickListener(v -> {
                        Intent intent = new Intent();
                        intent.setClass(activity, WeatherAlertActivity.class);
                        activity.startActivity(intent);
                    });
                }else{
                    tvAlert.setVisibility(View.INVISIBLE);
                }
                forceUpdate.put(item.getItemType(),false);
            }

            private void updateCondition(BaseViewHolder helper, MultiItemEntity
            item){
                boolean force = forceUpdate.get(item.getItemType());
                if(!force)return;
                ConditionItem conditionItem = (ConditionItem)item;
                YunCondition yunCondition = conditionItem.getYunCondition();
                helper.addOnClickListener(R.id.btn_speak);
                LinearLayout baseWeatherView = helper.getView(R.id.item_root);
                if(yunCondition != null){
                    baseWeatherView.setVisibility(View.VISIBLE);
```

```
                }else{
                    baseWeatherView.setVisibility(View.INVISIBLE);
                    return;
                }
                RecyclerView.LayoutParams layoutParams =
            (RecyclerView.LayoutParams)baseWeatherView.getLayoutParams();
                layoutParams.height =
            (int)activity.getResources().getDimension(R.dimen.qb_px_200);
                baseWeatherView.setLayoutParams(layoutParams);
                baseWeatherView.setOnClickListener(new View.OnClickListener(){
                    @Override
                    public void onClick(View v) {
                        Intent intent = new Intent();
                        intent.setClass(activity,
            ConditionDetailsActivity.class);
                        activity.startActivity(intent);
                    }
                });

                TextView tvWeather = helper.getView(R.id.tv_weather);
                tvWeather.setText(yunCondition.getWea());

                TextView tvTemperature = helper.getView(R.id.tv_temperature);
                Typeface typeface =
            Typeface.createFromAsset(activity.getAssets(),"fonts/pf_tc_tiny.ttf");
                tvTemperature.setTypeface(typeface);
                tvTemperature.setText(yunCondition.getTem()+"° ");

                TextView tvWind = helper.getView(R.id.tv_wind);

            tvWind.setText(yunCondition.getWin()+yunCondition.getWin_speed());

                TextView tvHumidity = helper.getView(R.id.tv_humidity);
                tvHumidity.setText("湿度"+yunCondition.getHumidity());

                TextView tvTips = helper.getView(R.id.tv_tips);
                tvTips.setText(yunCondition.getAir_tips());
                tvTips.setSelected(true);

                TextView tvPreesure = helper.getView(R.id.tv_pressure);
                tvPreesure.setText("气压"+yunCondition.getPressure()+"hPa");

                View groupAqi = helper.getView(R.id.group_aqi);
                TextView tvAqi = helper.getView(R.id.tv_aqi);
                ImageView imgAqi = helper.getView(R.id.img_aqi);
                if(!StringUtils.isEmpty(yunCondition.getAir())){
                    groupAqi.setVisibility(View.VISIBLE);
                    String value = yunCondition.getAir();
                    String quality =
            WeatherUtils.getAqiValue(yunCondition.getAir());
```

```
                tvAqi.setText(quality+" "+value);
                if(quality.equals("优")){

            imgAqi.setColorFilter(activity.getResources().getColor(R.color.color_qu
            ality_1));
                }
                else if(quality.equals("良")){

            imgAqi.setColorFilter(activity.getResources().getColor(R.color.color_qu
            ality_2));
                }
                else if(quality.equals("轻度")){

            imgAqi.setColorFilter(activity.getResources().getColor(R.color.color_qu
            ality_3));
                }
                else if(quality.equals("中度")){

            imgAqi.setColorFilter(activity.getResources().getColor(R.color.color_qu
            ality_4));
                }
                else if(quality.equals("重度") || quality.equals("严重")){

            imgAqi.setColorFilter(activity.getResources().getColor(R.color.color_qu
            ality_5));
                }
                groupAqi.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        Intent intent = new Intent(activity,
            AqiDetailsActivity.class);
                        activity.startActivity(intent);
                        Map<String,String> params = new HashMap<>();
                        params.put("position","空气图标");
                        TrackPlugin.track(activity,AQI_CLICK,params);
                    }
                });
            }else{
                groupAqi.setVisibility(View.INVISIBLE);
            }
            VoiceWaveView voiceWaveView =
            helper.getView(R.id.voiceWaveView0);
            ImageView voiceClose = helper.getView(R.id.voice_close);
            setCurrentVoiceView(voiceWaveView,voiceClose);
            forceUpdate.put(item.getItemType(),false);
        }

        private void updateTwoDays(BaseViewHolder helper, MultiItemEntity
        item){
            boolean force = forceUpdate.get(item.getItemType());
```

```
            if(!force)return;
            TwoDaysItem twoDaysItem = (TwoDaysItem)item;
            View todayPanel = helper.getView(R.id.item_today_panel);
            View tomorrowPanel = helper.getView(R.id.item_tomorrow_panel);
            List<YunForecastItem> forecastData =
        twoDaysItem.getForecastItems();
            YunForecastItem todayForecast = null;
            YunForecastItem tomorrowForecast = null;
            for(int i = 0; i < forecastData.size();i++){
                YunForecastItem forecastItem = forecastData.get(i);
                String todayTime = simpleDateManager.transformTime("yyyy-
        MM-dd");
                String targetTime = forecastItem.getDate();
                if(todayTime.equals(targetTime)){
                    todayForecast = forecastItem;
                    if(i+1 < forecastData.size())tomorrowForecast =
        forecastData.get(i+1);
                    break;
                }
            }
            updateBottomWeather(todayPanel,"今天",todayForecast);
            updateBottomWeather(tomorrowPanel,"明天",tomorrowForecast);
            todayPanel.setOnClickListener(v -> activity.startActivity(new
        Intent(activity,ConditionDetailsActivity.class)));
            final YunForecastItem finalTomorrowForecast = tomorrowForecast;
            tomorrowPanel.setOnClickListener(v -> {
                ActionEvent event = new
        ActionEvent(EventType.CHANGE_SECEN_FORECAST);
                if(finalTomorrowForecast !=
        null)event.setAttr("time",finalTomorrowForecast.getDate());
                EventBus.getDefault().post(event);
            });
            forceUpdate.put(item.getItemType(),false);
        }

        private void updateBottomWeather(View root, String day,
        YunForecastItem baseForecast){
            if(baseForecast == null){
                root.setVisibility(View.INVISIBLE);
                return;
            }
            root.setVisibility(View.VISIBLE);
            TextView tvWeek = root.findViewById(R.id.tv_week);
            TextView tvTempMinMax =
        root.findViewById(R.id.tv_temp_min_max);
            ImageView iconWeather = root.findViewById(R.id.icon_weather);
            tvWeek.setText(day);

            TextView tvDay = root.findViewById(R.id.tv_day);

```

```
tvDay.setText(StringUtils.formatTransformStyle(baseForecast.getDate(),d
ateFormat,itemDayFormat));



tvTempMinMax.setText(baseForecast.getTem1()+"° /"+baseForecast.getTem2(
)+"° ");

iconWeather.setImageResource(WeatherResUtils.getYunWeatherDayIcon(baseF
orecast.getWea_img()));

        TextView tvWeather = root.findViewById(R.id.tv_weather);
        tvWeather.setVisibility(View.VISIBLE);
        tvWeather.setText(baseForecast.getWea());
    }

    private void update24Hourly(BaseViewHolder helper, MultiItemEntity
item){
        boolean force = forceUpdate.get(item.getItemType());
        if(!force)return;
        HourlyItem hourlyItem = (HourlyItem)item;
        HorizontalScrollView hourlyListView =
helper.getView(R.id.hourly_list_view);
        hourlyListView.scrollTo(PixelUtils.dp2px(activity,20),0);
//        hourlyView = root.findViewById(R.id.group_hourly);
        if(Build.VERSION.SDK_INT >= 23){
            hourlyListView.setOnScrollChangeListener(new
View.OnScrollChangeListener(){
                @Override
                public void onScrollChange(View v, int scrollX, int
scrollY, int oldScrollX, int oldScrollY) {
                    if(hourlyListView.getScrollX() > 0 && !hourlySlip){
                        hourlySlip = true;
                        Log.e("ijimu","hourly slip :
"+hourlyListView.getScrollX());

TrackPlugin.track(activity,CONDITION_24_SLIP,null);
                    }
                }
            });
        }
        if(hourlyItem.getHours() != null && hourlyItem.getHours() !=
null){
            hourlyListView.setVisibility(View.VISIBLE);
            List<YunHours> baseHourlyList = hourlyItem.getHours();
            RecyclerView recyclerView =
helper.getView(R.id.weather_hourly_recyclerview);
            WeatherHourlyAdapter adapter =
hourlyAdapterArray.get(helper.getLayoutPosition());"+forceUpdate);
            if(adapter == null){
                adapter = new WeatherHourlyAdapter(activity);
```

```
                    LinearLayoutManager linearLayoutManager = new
LinearLayoutManager(activity);

linearLayoutManager.setOrientation(LinearLayoutManager.HORIZONTAL);
                    recyclerView.setAdapter(adapter);
                    recyclerView.setLayoutManager(linearLayoutManager);

hourlyAdapterArray.put(helper.getLayoutPosition(),adapter);
                }
                adapter.setNewData(hourlyItem.getHours());
                forceUpdate.put(item.getItemType(),false);

                LineChart lineChart =
helper.getView(R.id.weather_hourly_chart_1);

                if(baseHourlyList == null)return;
                //chartviewdouble
                int size = baseHourlyList.size();
                RelativeLayout.LayoutParams layoutParams =
(RelativeLayout.LayoutParams)lineChart.getLayoutParams();
                int width = getItemWidth();
                layoutParams.leftMargin = width/20;
                layoutParams.rightMargin = width/20;
                lineChart.setLayoutParams(layoutParams);

                ArrayList<Entry> tempData = new ArrayList<>();
                for(int i = 0; i < size;i++){
                    YunHours yunHours = baseHourlyList.get(i);
                    Entry entry = new Entry();
                    float tempY = StringUtils.isEmpty(yunHours.getTem())?-
1000:Integer.parseInt(yunHours.getTem());
                    entry.setX(i);
                    entry.setY(tempY);
                    tempData.add(entry);
                }

setLineChart(lineChart,tempData,Color.parseColor("#ffffff"),Color.parse
Color("#1E70FF"),R.drawable.bg_blue_transform_vertical);
            }else{
                hourlyListView.setVisibility(View.GONE);
            }
            TextView riseInfoTv = helper.getView(R.id.rise_info);
            if(!StringUtils.isEmpty(hourlyItem.getSunrise())&&
               !StringUtils.isEmpty(hourlyItem.getSunset())){StringBuffer
buffer = new StringBuffer();
                buffer.append("日出\t");
                buffer.append(hourlyItem.getSunrise()+"/");
                buffer.append("日落\t");
                buffer.append(hourlyItem.getSunset());
                riseInfoTv.setText(buffer.toString());
```

```
                riseInfoTv.setVisibility(View.VISIBLE);
            }else{
                riseInfoTv.setVisibility(View.INVISIBLE);
            }
            forceUpdate.put(item.getItemType(),false);
        }

        private void setLineChart(LineChart lineChart,ArrayList<Entry>
    data,int circleColor,int circleHoleColor,int filterDrawable){
            lineChart.getLegend().setEnabled(false);
            lineChart.setDrawBorders(false);
            lineChart.setDrawGridBackground(false);
            lineChart.setScaleEnabled(false);
            lineChart.getXAxis().setDrawLabels(false);
            lineChart.getXAxis().setDrawAxisLine(false);
            lineChart.getXAxis().setDrawGridLines(false);
            lineChart.getAxisLeft().setDrawLabels(false);
            lineChart.getAxisLeft().setDrawGridLines(false);
            lineChart.getAxisLeft().setDrawAxisLine(false);
            lineChart.getAxisRight().setDrawGridLines(false);
            lineChart.getAxisRight().setDrawLabels(false);
            lineChart.getAxisRight().setDrawAxisLine(false);
            lineChart.getXAxis().setGranularity(1f);
            lineChart.getXAxis().setLabelCount(data.size(),true);
            lineChart.setTouchEnabled(false);
            lineChart.getDescription().setEnabled(false);
            lineChart.getXAxis().setAvoidFirstLastClipping(true);

            LineDataSet set1 = new LineDataSet(data,"");
            set1.setMode(LineDataSet.Mode.CUBIC_BEZIER);
            set1.setCircleColor(circleColor);
            set1.setCircleHoleColor(circleHoleColor);
            set1.setColor(circleHoleColor);
            set1.setValueTextColor(circleHoleColor);
            set1.setCircleRadius(4f);
            set1.setCircleHoleRadius(3f);
            set1.setDrawCircleHole(true);
            set1.setDrawCircles(true);
            set1.setLineWidth(2);
            set1.setFormLineWidth(1f);
            set1.setValueTextSize(15f);
            set1.setDrawFilled(true);
            set1.setValueFormatter(new ValueFormatter() {
                @Override
                public String getFormattedValue(float value){
                    return (int)value+"° ";
                }
            });
            Drawable drawable = ContextCompat.getDrawable(activity,
    filterDrawable);
```

```
            set1.setFillDrawable(drawable);
            ArrayList<ILineDataSet> dataSets = new ArrayList<>();
            dataSets.add(set1);
            lineChart.setData(new LineData(dataSets));
            lineChart.invalidate();
        }

        private void updateForecast(BaseViewHolder helper, MultiItemEntity
    item){
            boolean force = forceUpdate.get(item.getItemType());
            if(!force)return;
            ForecastItem forecastItem = (ForecastItem)item;
            HorizontalScrollView forecastListView =
    helper.getView(R.id.forecast_list_view);
            forecastListView.post(new Runnable() {
                @Override
                public void run() {

    forecastListView.scrollTo(PixelUtils.dp2px(activity,20),0);
                }
            });
            if(Build.VERSION.SDK_INT >= 23){
                forecastListView.setOnScrollChangeListener(new
    View.OnScrollChangeListener(){
                    @Override
                    public void onScrollChange(View v, int scrollX, int
    scrollY, int oldScrollX, int oldScrollY) {
                        if(forecastListView.getScrollX() > 0
    && !forecastSlip){
                            forecastSlip = true;
                            Log.e("ijimu","forecast slip :
    "+forecastListView.getScrollX());

    TrackPlugin.track(activity,CONDITION_15_SLIP,null);
                        }
                    }
                });
            }
            TextView btnMore = helper.getView(R.id.btn_more);
            btnMore.setOnClickListener(new View.OnClickListener(){
                @Override
                public void onClick(View v) {
                    ActionEvent event = new
    ActionEvent(EventType.CHANGE_SECEN_FORECAST);
                    EventBus.getDefault().post(event);
                    Map<String,String> params = new HashMap<>();
                    params.put("position","查看更多");
                    TrackPlugin.track(activity,CONDITION_15_CLICK,params);
                }
            });
```

```
            if(forecastItem.getForecastItems() != null &&
forecastItem.getForecastItems().size() > 0){
                forecastListView.setVisibility(View.VISIBLE);
                List<YunForecastItem> forecastItems =
forecastItem.getForecastItems();
                forecastItems =
forecastItems.subList(0,Math.min(forecastItems.size(),36));
                RecyclerView recyclerView =
helper.getView(R.id.weather_week_recyclerview);
                WeatherWeekAdapter adapter =
forecastAdapterArray.get(helper.getLayoutPosition());
                if(adapter == null){
                    adapter = new WeatherWeekAdapter(activity);
                    recyclerView.setAdapter(adapter);
                    LinearLayoutManager linearLayoutManager = new
LinearLayoutManager(activity);

linearLayoutManager.setOrientation(LinearLayoutManager.HORIZONTAL);
                    recyclerView.setLayoutManager(linearLayoutManager);
                    adapter.setNewData(forecastItems);

forecastAdapterArray.put(helper.getLayoutPosition(),adapter);
                }
                adapter.setNewData(forecastItems);

                if(forecastItems == null)return;
                int size = forecastItems.size();

                LinearLayout groupWeatherChart =
helper.getView(R.id.group_weather_chart);
                RelativeLayout.LayoutParams layoutParams =
(RelativeLayout.LayoutParams)groupWeatherChart.getLayoutParams();
                int width = getItemWidth();
                layoutParams.leftMargin = width/20;
                layoutParams.rightMargin = width/20;
                groupWeatherChart.setLayoutParams(layoutParams);

                LineChart lineChartMax =
helper.getView(R.id.weather_chart_max);
                LineChart lineChartMin =
helper.getView(R.id.weather_chart_min);

                ArrayList<Entry> dayTempData = new ArrayList<>();
                ArrayList<Entry> nightTempData = new ArrayList<>();
                for(int i = 0; i < size;i++){
                    YunForecastItem baseForecast = forecastItems.get(i);
                    dayTempData.add(new
Entry(i,StringUtils.isEmpty(baseForecast.getTem1())?0:Integer.parseInt(
baseForecast.getTem1())));
                    nightTempData.add(new
```

```
Entry(i,StringUtils.isEmpty(baseForecast.getTem2())?0:Integer.parseInt(
baseForecast.getTem2())));
            }

setLineChart(lineChartMax,dayTempData,Color.parseColor("#ffffff"),Color
.parseColor("#FFB41E"),R.drawable.bg_orange_transform_vertical);

setLineChart(lineChartMin,nightTempData,Color.parseColor("#ffffff"),Col
or.parseColor("#1E70FF"),R.drawable.bg_blue_transform_vertical);
        }else{
            forecastListView.setVisibility(View.GONE);
        }
        forceUpdate.put(item.getItemType(),false);
    }

    private void updateAqi(BaseViewHolder helper, MultiItemEntity
item){
        boolean force = forceUpdate.get(item.getItemType());
        if(!force)return;
        AqiItem aqiItem = (AqiItem)item;
        LinearLayout aqiParent = helper.getView(R.id.item_root);
        WaveView waveView = helper.getView(R.id.aqi_capsule);
        waveView.setShapeType(WaveView.ShapeType.CAPSULE);
        if(aqiItem.getYunAqi() != null){
            aqiParent.setVisibility(View.VISIBLE);
            YunAqi baseAqi = aqiItem.getYunAqi();
            String quality =
WeatherUtils.getAqiValue(baseAqi.getAir());
            helper.setText(R.id.tv_aqi_capsule_value,baseAqi.getAir());
            helper.setText(R.id.tv_aqi_capsule_quality,quality);
            helper.setVisible(R.id.btn_more,true);
            if(quality.equals("优")){
                waveView.setWaveColor(Color.parseColor("#2021CA49"),
                        Color.parseColor("#4021CA49"),
                        Color.parseColor("#8021CA49"));

helper.setTextColor(R.id.tv_aqi_capsule_name,activity.getResources().ge
tColor(R.color.color_quality_1));
            }
            else if(quality.equals("良")){
                waveView.setWaveColor(Color.parseColor("#20FEDD00"),
                        Color.parseColor("#40FEDD00"),
                        Color.parseColor("#80FEDD00"));

helper.setTextColor(R.id.tv_aqi_capsule_name,activity.getResources().ge
tColor(R.color.color_quality_2));
            }
            else if(quality.equals("轻度")){
                waveView.setWaveColor(Color.parseColor("#20FFA200"),
                        Color.parseColor("#40FFA200"),
```

```
                        Color.parseColor("#80FFA200"));

        helper.setTextColor(R.id.tv_aqi_capsule_name,activity.getResources().ge
        tColor(R.color.color_quality_3));
                    }
                    else if(quality.equals("中度")){
                        waveView.setWaveColor(Color.parseColor("#20FE1E1E"),
                                Color.parseColor("#40FE1E1E"),
                                Color.parseColor("#80FE1E1E"));

        helper.setTextColor(R.id.tv_aqi_capsule_name,activity.getResources().ge
        tColor(R.color.color_quality_4));
                    }
                    else if(quality.equals("重度")){
                        waveView.setWaveColor(Color.parseColor("#20A20050"),
                                Color.parseColor("#40A20050"),
                                Color.parseColor("#80A20050"));

        helper.setTextColor(R.id.tv_aqi_capsule_name,activity.getResources().ge
        tColor(R.color.color_quality_5));
                    }
                    else if(quality.equals("严重")){
                        waveView.setWaveColor(Color.parseColor("#2081022B"),
                                Color.parseColor("#4081022B"),
                                Color.parseColor("#8081022B"));

        helper.setTextColor(R.id.tv_aqi_capsule_name,activity.getResources().ge
        tColor(R.color.color_quality_6));
                    }
                    WaveHelper waveHelper = new WaveHelper(waveView);
                    waveHelper.start();

                    setAqiItem(helper.getView(R.id.item_pm25),"PM2.5(细颗粒
        物)",baseAqi.getPm25());
                    setAqiItem(helper.getView(R.id.item_pm10),"PM10(可吸入颗粒
        物)",baseAqi.getPm10());
                    setAqiItem(helper.getView(R.id.item_no2),"NO2(二氧化
        氮)",baseAqi.getNo2());
                    setAqiItem(helper.getView(R.id.item_so2),"SO2(二氧化
        硫)",baseAqi.getSo2());
                    setAqiItem(helper.getView(R.id.item_co),"CO(一氧化
        碳)",baseAqi.getCo());
                    setAqiItem(helper.getView(R.id.item_o3),"O3(臭氧指
        数)",baseAqi.getO3());

                    aqiParent.setOnClickListener(new View.OnClickListener(){
                        @Override
                        public void onClick(View v) {
                            Intent intent = new Intent(activity,
        AqiDetailsActivity.class);
```

```
                            activity.startActivity(intent);
                            Map<String,String> params = new HashMap<>();
                            params.put("position","查看更多");
                            TrackPlugin.track(activity,AQI_CLICK,params);
                        }
                    });
                }
                else{
                    waveView.setWaveColor(Color.parseColor("#20b4b4b4"),
                    Color.parseColor("#40b4b4b4"),
                    Color.parseColor("#80b4b4b4"));
                    helper.setVisible(R.id.btn_more,false);
                    helper.setText(R.id.tv_aqi_capsule_value,"0");
                    helper.setText(R.id.tv_aqi_capsule_quality,"无");

            helper.setTextColor(R.id.tv_aqi_capsule_name,Color.parseColor("#ffffff"
            ));
                    setAqiItem(helper.getView(R.id.item_pm25),"PM2.5(细颗粒
            物)","--");
                    setAqiItem(helper.getView(R.id.item_pm10),"PM10(可吸入颗粒
            物)","--");
                    setAqiItem(helper.getView(R.id.item_no2),"NO2(二氧化氮)","-
            -");
                    setAqiItem(helper.getView(R.id.item_so2),"SO2(二氧化硫)","-
            -");
                    setAqiItem(helper.getView(R.id.item_co),"CO(一氧化碳)","--
            ");
                    setAqiItem(helper.getView(R.id.item_o3),"O3(臭氧指数)","--
            ");
                }
                forceUpdate.put(item.getItemType(),false);
            }

            private void setAqiItem(View itemView,String name,String value){
                TextView tvName = itemView.findViewById(R.id.item_name);
                tvName.setText(name);
                TextView tvValue = itemView.findViewById(R.id.item_value);
                tvValue.setText(value);
            }

            private void updateWanYearView(BaseViewHolder helper,
            MultiItemEntity item){
                helper.addOnClickListener(R.id.simple_fate_root);
                Calendar calendar = Calendar.getInstance();
                calendar.setTime(new Date());
                int year = calendar.get(Calendar.YEAR);
                int month = calendar.get(Calendar.MONTH)+1;
                int day = calendar.get(Calendar.DAY_OF_MONTH);
                WanYearItem wanYearItem = (WanYearItem)item;
                BaseYear baseYear = wanYearItem.getBaseWanYear().getBaseYear();
```

```
            BaseMonth baseMonth =
        wanYearItem.getBaseWanYear().getBaseMonth();
            BaseDay baseDay = wanYearItem.getBaseWanYear().getBaseDay();
            if(baseMonth!= null && baseDay != null){
                helper.setText(R.id.tv_time_1,"农历
        "+baseMonth.getNongliyue()+baseDay.getNongliri());
            }
            if(baseYear != null && baseDay!=null && baseMonth!=null){

        helper.setText(R.id.tv_time_3,baseYear.getNongli()+"("+baseYear.getShux
        iang()+")"+"年 "+baseMonth.getSizhu_yuezhu()+"月
        "+baseDay.getSizhu_rizhu()+"日");
            }
            if(baseDay != null){

        helper.setText(R.id.tv_yi,baseDay.getRifen_yi().replaceAll("、","  "));

        helper.setText(R.id.tv_ji,baseDay.getRifen_ji().replaceAll("、","  "));
            }
            int weekCount =
        CalendarUtil.getWeekCountBetweenBothCalendar(year,1,1,year,month,day,1)
        ;
            int week = SupportUtils.getWeek(year,month,day);
            helper.setText(R.id.tv_time_2,"第"+weekCount+"周
        "+SupportUtils.weekNameTransform(week));
        }

        private void updateLiveIndex(BaseViewHolder helper,MultiItemEntity
        item){
            boolean force = forceUpdate.get(item.getItemType());
            if(!force)return;
            LiveIndexItem liveIndexItem = (LiveIndexItem)item;
            if(liveIndexItem.getYunLiveIndex() != null){
                RecyclerView recyclerView =
        helper.getView(R.id.live_index_recycler_view);
                LiveIndexAdapter liveIndexAdapter =
        liveIndexAdapterArray.get(helper.getLayoutPosition());
                if(liveIndexAdapter == null){
                    liveIndexAdapter = new
        LiveIndexAdapter(activity,LiveIndexAdapter.FRAGMENT,R.layout.item_live_
        index_1);
                    GridLayoutManager gridLayoutManager = new
        GridLayoutManager(activity,4);
                    recyclerView.setAdapter(liveIndexAdapter);
                    recyclerView.setLayoutManager(gridLayoutManager);

        liveIndexAdapterArray.put(helper.getLayoutPosition(),liveIndexAdapter);
                }
                List<AdLiveItem> adLiveItemList = new LinkedList<>();
                YunLiveIndex yunLiveIndex =
```

```
            liveIndexItem.getYunLiveIndex();
                        if(canShowLiveIndexItem(yunLiveIndex.getXiche())){

            adLiveItemList.add(createAdLiveItem(AdLiveItem.TYPE_LIVE,
            R.drawable.ic_wash_car, yunLiveIndex.getXiche().getLevel(), "洗车"));
                        }
                        if(canShowLiveIndexItem(yunLiveIndex.getChuanyi())){

            adLiveItemList.add(createAdLiveItem(AdLiveItem.TYPE_LIVE,
            R.drawable.ic_wear, yunLiveIndex.getChuanyi().getLevel(), "穿衣"));
                        }
                        if(canShowLiveIndexItem(yunLiveIndex.getZiwaixian())){

            adLiveItemList.add(createAdLiveItem(AdLiveItem.TYPE_LIVE,
            R.drawable.ic_sun_light, yunLiveIndex.getZiwaixian().getLevel(), "紫外
            线"));
                        }
                        if(canShowLiveIndexItem(yunLiveIndex.getGanmao())){

            adLiveItemList.add(createAdLiveItem(AdLiveItem.TYPE_LIVE,
            R.drawable.ic_cold, yunLiveIndex.getGanmao().getLevel(), "感冒"));
                        }
                        if(canShowLiveIndexItem(yunLiveIndex.getChenlian())){

            adLiveItemList.add(createAdLiveItem(AdLiveItem.TYPE_LIVE,
            R.drawable.ic_sport, yunLiveIndex.getChenlian().getLevel(), "运动"));
                        }
                        if(Constant.SHOW_AD&&!Constant.isEmulator&&!isVip()){
                        }
                        liveIndexAdapter.setNewData(adLiveItemList);
                    }
                    forceUpdate.put(item.getItemType(),false);
                }

                private AdLiveItem createAdLiveItem(int type, int iconId, String
            content, String label){
                    return createAdLiveItem(type,iconId,content,label,"");
                }

                private AdLiveItem createAdLiveItem(int type, int iconId, String
            content, String label,String contentColor){
                    AdLiveItem adLiveItem = new AdLiveItem();
                    adLiveItem.setType(type);
                    adLiveItem.setIconId(iconId);
                    adLiveItem.setContent(content);
                    adLiveItem.setLabel(label);
                    adLiveItem.setContentColor(StringUtils.isEmpty(contentColor)?
            Color.parseColor("#333333"):Color.parseColor(contentColor));
                    return adLiveItem;
                }
```

```
    private boolean canShowLiveIndexItem(YunLiveIndexItem indexItem){
        if(indexItem == null)return false;

return !StringUtils.isEmpty(indexItem.getLevel())&&!StringUtils.isEmpty
(indexItem.getTips());
    }

    private void updateBanner1(BaseViewHolder helper){
        boolean force = forceUpdate.get(helper.getItemViewType());
        if(!force)return;
        FrameLayout frameLayout =
helper.getView(R.id.banner_container);
        showNativeBanner1(frameLayout,helper.getLayoutPosition());
        forceUpdate.put(helper.getItemViewType(),false);
    }

    private void updateBanner2(BaseViewHolder helper){
        boolean force = forceUpdate.get(helper.getItemViewType());
        if(!force)return;
        FrameLayout frameLayout =
helper.getView(R.id.banner_container);
        showNativeBanner2(frameLayout,helper.getLayoutPosition());
        forceUpdate.put(helper.getItemViewType(),false);
    }

    private void updateNative(BaseViewHolder helper){
        boolean force = forceUpdate.get(helper.getItemViewType());
        if(!force)return;
        FrameLayout frameLayout = helper.getView(R.id.item_root);
        showNative(frameLayout,helper.getLayoutPosition());
        forceUpdate.put(helper.getItemViewType(),false);
    }

    private void updateNews(BaseViewHolder helper){
        LinearLayout linearLayout = helper.getView(R.id.item_root);
        if(newsView == null){
            newsView = new NewsView();
            newsView.initView(linearLayout);
            newsView.setViewPosition("首页");
        }
        newsView.initData(activity,newsTypeManager.getNewsItems());
    }

    @Override
    public void logInfo(String msg){

    }

    @Override
```

```
    public void onResume() {
        super.onResume();
        resumeAd(TYPE_BANNER_1);
        resumeAd(TYPE_BANNER_2);
        resumeAd(TYPE_NATIVE);
        resumeAd(TYPE_CONDITION);
        if(newsTypeManager.isNeedUpdate()){
            notifyDataSetChanged();
            newsTypeManager.setNeedUpdate(false);
        }
    }

    @Override
    public void onPause() {
        super.onPause();
        pauseAd(TYPE_BANNER_1);
        pauseAd(TYPE_BANNER_2);
        pauseAd(TYPE_NATIVE);
        pauseAd(TYPE_CONDITION);
    }

    private void resumeAd(int itemType){
        forceUpdate.put(itemType,true);
    }

    private void pauseAd(int itemType){
        forceUpdate.put(itemType,false);
    }

    private void setCurrentVoiceView(VoiceWaveView
voiceWaveView,ImageView voiceClose){
        if(speakHolder == null){
            speakHolder = new SpeakHolder();
            speakHolder.waveView = voiceWaveView;
            speakHolder.closeView = voiceClose;
            setVoiceWaveData(voiceWaveView);
            speakStatus(false);
        }
    }

    public void speakStatus(boolean isSpeak){
        if(speakHolder == null){
            logInfo("speakHolder null");
            return;
        }
        logInfo("speak status , isSpeak : "+isSpeak);
        if(speakHolder.waveView != null){
            if(isSpeak){
                speakHolder.waveView.setVisibility(View.VISIBLE);
                speakHolder.waveView.start();
```

```
                    }else{
                        speakHolder.waveView.stop();
                        speakHolder.waveView.setVisibility(View.INVISIBLE);
                    }
                }
                if(speakHolder.closeView != null){

speakHolder.closeView.setVisibility(isSpeak?View.INVISIBLE:View.VISIBLE
);
                }
            }

            public void shakeAnimation(){
                if(speakHolder == null)return;
                ViewAnimator.animate(speakHolder.closeView)
                        .scale(1f,1.05f,1.15f,1.05f,1f)
                        .duration(1500)
                        .startDelay(1000)
                        .repeatCount(5)
                        .repeatMode(ViewAnimator.REVERSE)
                        .start();
            }

            private void setVoiceWaveData(VoiceWaveView voiceWaveView){
                voiceWaveView.setDuration(200);
                voiceWaveView.addHeader(2);
                voiceWaveView.addBody(8);
                voiceWaveView.addBody(17);
                voiceWaveView.addBody(38);
                voiceWaveView.addBody(24);
                voiceWaveView.addBody(8);
                voiceWaveView.addBody(38);
                voiceWaveView.addBody(14);
                voiceWaveView.addBody(27);
                voiceWaveView.addBody(8);
                voiceWaveView.addFooter(2);
                voiceWaveView.setWaveMode(WaveMode.LEFT_RIGHT);
                voiceWaveView.setLineWidth(5f);
                voiceWaveView.setLineSpace(2f);
            }

            private static class SpeakHolder{
                public VoiceWaveView waveView;
                public ImageView closeView;
            }

            protected boolean isVip(){
                return userManager.isLogin()&&userManager.isVip();
            }

```

```
    @Override
    public void onAttachedToRecyclerView(RecyclerView recyclerView) {
        super.onAttachedToRecyclerView(recyclerView);
        FullSpanUtil.onAttachedToRecyclerView(recyclerView, this,
TYPE_NEWS);
    }

    @Override
    public void onViewAttachedToWindow(BaseViewHolder holder) {
        super.onViewAttachedToWindow(holder);
        FullSpanUtil.onViewAttachedToWindow(holder, this, TYPE_NEWS);
    }

    private int getItemWidth(){
        int screenW = getWidthHeight()[0] -
PixelUtils.dip2px(activity,20);
        return (screenW);
    }

    private int[] getWidthHeight(){
        Resources resources = activity.getResources();
        DisplayMetrics dm = resources.getDisplayMetrics();
        int width = dm.widthPixels;
        int height = dm.heightPixels;
        return new int[]{width,height};
    }
}
package com.frostsowner.magic.weather.activity;

import android.Manifest;
import android.app.AlertDialog;
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.net.Uri;
import android.os.Build;
import com.tencent.bugly.beta.Beta;
import com.zhy.android.percent.support.PercentRelativeLayout;

import org.greenrobot.eventbus.Subscribe;
import org.greenrobot.eventbus.ThreadMode;

import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import static com.frostsowner.magic.weather.Constant.SHOW_AD;

public class HomeTabActivity extends AtInsertAdActivity {

    private final String[] location_permissions = {
```

```
                Manifest.permission.ACCESS_COARSE_LOCATION,
                Manifest.permission.ACCESS_FINE_LOCATION
        };

        private final static String NAVIGATION_POWER_ANIMA =
    "tt_weather_navigation_power_animation";

        private final long BACK_PRESS_DELAYED = 2000;

        private int[][] navigationRes = {
                {R.drawable.ic_condition_2,R.drawable.ic_condition_1},
        };

        private String[] navigationNames = {
                "实况天气"
        };

        @BindView(R.id.view_pager)
        NoSwipeViewPager homeViewPager;

        @BindView(R.id.radio_line)
        View radioLine;

        @BindView(R.id.group_alert)
        View groupAlert;
        @BindView(R.id.tv_task_alert_msg)
        TextView tvTaskAlertMsg;

        @BindView(R.id.bottom_menu)
        PercentRelativeLayout bottomMenu;


    @BindViews({R.id.tab_condition,R.id.tab_forecast,R.id.tab_power,R.id.ta
    b_gift,R.id.tab_mine})
        List<View> navigationViews;

        private HomeAdapter homeAdapter;
        private LocalStorage localStorage;
        private long backPressTime;
        private AdSourceRequestManager adSourceRequestManager;
        private DeviceManager deviceManager;
        private UserManager userManager;
        private UserActionPatternManager userActionPatternManager;
        private NewGiftDialog newGiftDialog;
        private ViewAnimator animator;

        private long stayTime;

        @Override
        protected void onCreate(@Nullable Bundle savedInstanceState){
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home_tab);
        StatusBarUtil.setTranslucentForImageViewInFragment(this,
0,null);
        localStorage = BeanFactory.getBean(LocalStorage.class);
        adSourceRequestManager =
BeanFactory.getBean(AdSourceRequestManager.class);
        userManager = BeanFactory.getBean(UserManager.class);
        deviceManager = BeanFactory.getBean(DeviceManager.class);
        userActionPatternManager =
BeanFactory.getBean(UserActionPatternManager.class);
        deviceManager.init();
        if(deviceManager.isFirstInstall()){
            deviceManager.setInstall();
        }else{
            if(deviceManager.getFirstInstallTime() == 0){
                deviceManager.setInstall();
            }
            if(deviceManager.getInstalledDay() > 365*2){
                deviceManager.setInstall();
            }
        }
        boolean isNeedRequestAdFlag =
localStorage.get(NEED_REQUEST_AD_FLAG,true);
        if(!isNeedRequestAdFlag&&!SHOW_AD){
            SHOW_AD = true;
            Intent intent = new Intent(this,
WidgetHandleService.class);
            if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
                startForegroundService(intent);
            }else{
                startService(intent);
            }
        }
        initViews();
        handleAction();
        checkLocationSet();
        showApproval();
//        showImportAlertDialog();
        showAutoRemoveAdDialog();
        showInsertAd();
//        checkNotificationSet();
        Constant.HOME_HAS_DEAD = false;
        if(WeatherConfig.isNotificationSignAlertKey()){
            SignNotificationJob.schedule();
        }

if(ApkResources.isDebug(this)||Constant.TEST_SERVER_ENVIRONMENT){
            String builder = "APP 环境: debug = " +
ApkResources.isDebug(this) +
```

```
                "\n 基础服务器环境: " +
        (BeanFactory.getBean(AdSourceRequestManager.class).isUseTestServer() ?
        "测试服" : "正式服");
        //            showLongNotice(builder);
                }
        //        relaxFlagTest();
                Map<String,String> params = new HashMap<>();

        params.put("pattern",userActionPatternManager.getUserLevelName());
                TrackPlugin.track(this,USER_PATTERN,params);
                checkUpgrade();
            }

            private void showApproval(){
                boolean isClick = localStorage.get(APPROVAL_SHOW,false);
                if(isClick){
                    return;
                }
                SimpleDateManager simpleDateManager =
        BeanFactory.getBean(SimpleDateManager.class);
                String day =
        simpleDateManager.transformTime(System.currentTimeMillis(),"yyyy-MM-
        dd");
                boolean oneTime =
        localStorage.get(APPROVAL_DAY_TIME+day+"_1",false);
                boolean secondTime =
        localStorage.get(APPROVAL_DAY_TIME+day+"_2",false);
                if(!oneTime){
                    localStorage.put(APPROVAL_DAY_TIME+day+"_1",true);
                    return;
                }
                if(secondTime){
                    return;
                }
                showApprovalDialog("开发不易，好评鼓励我们一下吧~");
                localStorage.put(APPROVAL_DAY_TIME+day+"_2",true);
            }

            private void showImportAlertDialog(){
                if(SHOW_AD && !Constant.isEmulator
        && !deviceManager.isShowImportantAlert()){
                    ImportantAlertDialog dialog = new
        ImportantAlertDialog(this);
                    dialog.show();
                    deviceManager.setImportantShow();
                }
            }

            private void showAutoRemoveAdDialog(){
                if(!Constant.SHOW_AD&&!Constant.isEmulator)return;
```

```
        boolean isShowApproval = localStorage.get(APPROVAL_SHOW,false);
        if(!isShowApproval)return;
        long time = localStorage.get(AUTO_REMOVE_AD_SHOW,0L);
        if(time != 0 && System.currentTimeMillis() - time <
1000*60*60*24*3){
            return;
        }
        SimpleDateManager simpleDateManager =
BeanFactory.getBean(SimpleDateManager.class);
        String day =
simpleDateManager.transformTime(System.currentTimeMillis(),"yyyy-MM-
dd");
        boolean oneTime =
localStorage.get(AUTO_REMOVE_AD_DAY_TIME+day+"_1",false);
        boolean secondTime =
localStorage.get(AUTO_REMOVE_AD_DAY_TIME+day+"_2",false);
        if(!oneTime){
            localStorage.put(AUTO_REMOVE_AD_DAY_TIME+day+"_1",true);
            return;
        }
        if(secondTime){
            return;
        }
        fire(new ActionEvent(START_SHOW_REMOVE_AD_DIALOG));
        localStorage.put(AUTO_REMOVE_AD_DAY_TIME+day+"_2",true);
    }

    private void showFakeAdDialog(){
        if(!Constant.SHOW_AD&&!Constant.isEmulator)return;
        if(Build.VERSION.SDK_INT >= 23
&& !checkPermissionGranted(location_permissions))return;
        SimpleDateManager simpleDateManager =
BeanFactory.getBean(SimpleDateManager.class);
        String time =
simpleDateManager.transformTime(System.currentTimeMillis(),"yyyy-MM-
dd");
        boolean isFakeAdShow =
localStorage.get(FAKE_AD_WATCH_VIDEO+time,false);
        if(isFakeAdShow)return;
        addSubscription(SubscribeUtils.doOnUIThreadDelayed(new
Action0(){
            @Override
            public void call() {
                fire(new
ActionEvent(EventType.START_SHOW_FAKE_AD_DIALOG));
            }
        },8000));
        localStorage.put(FAKE_AD_WATCH_VIDEO+time,true);
    }

```

```
    @Override
    protected void onNewIntent(Intent intent){
        super.onNewIntent(intent);
        setIntent(intent);
        handleAction();
        logInfo("onNewIntent");
        Constant.HOME_HAS_DEAD = false;
    }

    private void handleAction(){
        if(getIntent() == null)return;
        String action = getIntent().getStringExtra("action");
        logInfo("home action : "+action);
        if(StringUtils.isEmpty(action)){
            return;
        }
        if(action.equals(ACTION_CONDITION)){
            Intent intent = new Intent();
            intent.setClass(this,ConditionDetailsActivity.class);
            startActivity(intent);
            TrackPlugin.track(this,NOTIFICATION_CLICK,null);
        }
        else if(action.equals(ACTION_15FORECAST)){
            Intent intent = new Intent();
            intent.setClass(this,ForecastDetailsActivity.class);
            startActivity(intent);
        }
        else if(action.equals(ACTION_HOME)){
            TrackPlugin.track(this,DESKTOP_WIDGET_CLICK,null);
        }
    }

    private void initViews(){
        homeViewPager.setReMeasure(false);
        homeAdapter = new HomeAdapter(getSupportFragmentManager());
        homeViewPager.setAdapter(homeAdapter);
        homeViewPager.setOffscreenPageLimit(homeAdapter.getCount());
        homeViewPager.setScrollFlag(false);

bottomMenu.setVisibility(View.GONE);//SHOW_AD&&!Constant.isEmulator?Vie
w.VISIBLE:View.GONE);

radioLine.setVisibility(View.GONE);//SHOW_AD&&!Constant.isEmulator?View
.VISIBLE:View.GONE);
    }

    private void initNavigation(){
        for(int i=0;i<navigationViews.size();i++){
            View view = navigationViews.get(i);
            view.setOnClickListener(navigationClick);
```

```
                }
                setNavigationVisible(0);
                View view = navigationViews.get(2);
                TextView itemName = view.findViewById(R.id.item_name);
                String key = NAVIGATION_POWER_ANIMA+dateFormat.format(new
            Date());
                boolean isShow = localStorage.get(key,false);
                if(userManager.isLogin()&&userManager.isVip()){
                    isShow = true;
                }
                if(isShow){
                    itemName.setVisibility(View.INVISIBLE);
                    return;
                }
                animator = ViewAnimator.animate(itemName)
                        .duration(900)
                        .startDelay(1000)
                        .scale(1f,1.03f,1.06f,1.1f,1.06f,1.03f)
                        .repeatCount(ViewAnimator.INFINITE)
                        .repeatMode(ViewAnimator.REVERSE)
                        .start();
            }

            private void setNavigationVisible(int position){
                for(int i=0;i<navigationViews.size();i++){
                    setNavigationItemVisible(i,i==position);
                }
            }

            private void setNavigationItemVisible(int position,boolean
            selected){
                View view = navigationViews.get(position);
                ImageView itemIcon = view.findViewById(R.id.item_icon);

            itemIcon.setImageDrawable(getResources().getDrawable(selected?navigatio
            nRes[position][1]:navigationRes[position][0]));
                TextView itemName = view.findViewById(R.id.item_name);
                if(position == 2){
                    itemName.setTextColor(0xffffffff);
                }else{
                    itemName.setTextColor(selected?0xff209CF4:0xffA49F9F);
                }
                itemName.setText(navigationNames[position]);
            }

            private View.OnClickListener navigationClick = new
            View.OnClickListener(){
                @Override
                public void onClick(View v) {
                    Map<String,String> params = new HashMap<>();
```

```
                switch (v.getId()){
                    case R.id.tab_condition:
                        homeViewPager.setCurrentItem(0,false);
                        setNavigationVisible(0);
                        params.clear();
                        params.put("click","实况天气");

        TrackPlugin.track(getContext(),HOME_NAVIGATION_CLICK,params);
        //                    startShowTaskAlert();
                        break;
                    case R.id.tab_forecast:
                        homeViewPager.setCurrentItem(1,false);
                        setNavigationVisible(1);
                        params.clear();
                        params.put("click","万年历");

        TrackPlugin.track(getContext(),HOME_NAVIGATION_CLICK,params);
                        showInsertAd();
                        break;
                    case R.id.tab_power:
                        homeViewPager.setCurrentItem(2,false);
                        setNavigationVisible(2);
                        if(animator != null){
                            animator.cancel();
                            animator = null;
                            View view = navigationViews.get(2);

        view.findViewById(R.id.item_name).setVisibility(View.INVISIBLE);
                            String key =
        NAVIGATION_POWER_ANIMA+dateFormat.format(new Date());
                            localStorage.put(key,true);
                        }
                        params.clear();
                        params.put("click","活动");

        TrackPlugin.track(getContext(),HOME_NAVIGATION_CLICK,params);
                        break;
                    case R.id.tab_gift:
                        if(deviceManager.isFirstInstall()){
                        }else{
                            homeViewPager.setCurrentItem(3,false);
                            setNavigationVisible(3);
                        }
                        params.clear();
                        params.put("click","福利");

        TrackPlugin.track(getContext(),HOME_NAVIGATION_CLICK,params);
                        break;
                    case R.id.tab_mine:
                        homeViewPager.setCurrentItem(4,false);
```

```
                        setNavigationVisible(4);
                        params.clear();
                        params.put("click","我的");

TrackPlugin.track(getContext(),HOME_NAVIGATION_CLICK,params);
                        params.clear();
                        params.put("position","底部导航");

TrackPlugin.track(getContext(),MINE_NAVIGATION_CLICK,params);
                        break;
                }
            }
        };

        private void checkLocationSet(){
            if((Build.VERSION.SDK_INT < 23 ||
checkPermissionGranted(location_permissions)) &&
LocationUtils.isLocServiceEnable(this))return;
            boolean locationAlert =
isProperty("tt_weather_location_guide_alert");
            if(locationAlert)return;
            LocationGuideDialog dialog = new LocationGuideDialog(this, new
View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    addPermission(location_permissions, 102, new
PermissionCallBack() {
                        @Override
                        public void result(boolean isGranted, List<String>
permission) {
                            if(!isGranted){
                                showNotice("申请失败");
                            }
                        }
                    });
                    checkPermissions();
                }
            });
            dialog.show();
            setProperty("tt_weather_location_guide_alert","true");
        }

        private void checkNotificationSet(){
            if(Build.VERSION.SDK_INT < Build.VERSION_CODES.KITKAT)return;
            if(NotificationUtils.isNotificationEnable(this))return;
            boolean isShow =
localStorage.get("tt_weather_notification_set_alert",false);
            if(isShow)return;
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setTitle("温馨提示");
```

```
            builder.setMessage("检测到您没有开启通知服务,可能会影响您的天气
        功能使用,是否跳转至设置页面");
            builder.setNegativeButton("取消",null);
            builder.setPositiveButton("立即前往", (dialog, which) -> {
                if (android.os.Build.VERSION.SDK_INT >=
        Build.VERSION_CODES.LOLLIPOP) {
                    Intent intent = new Intent();

        intent.setAction("android.settings.APP_NOTIFICATION_SETTINGS");
                    intent.putExtra("app_package", getPackageName());
                    intent.putExtra("app_uid", getApplicationInfo().uid);
                    startActivity(intent);
                } else if (android.os.Build.VERSION.SDK_INT ==
        Build.VERSION_CODES.KITKAT) {
                    Intent intent = new Intent();

        intent.setAction(Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
                    intent.addCategory(Intent.CATEGORY_DEFAULT);
                    intent.setData(Uri.parse("package:" +
        getPackageName()));
                    startActivity(intent);
                }
            });
            builder.show();
            localStorage.put("tt_weather_notification_set_alert",true);
        }

        private void showApprovalDialog(String content){
            ApprovalDialog approvalDialog = new ApprovalDialog(this,
        content, new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    try {
                        Uri uri = Uri.parse("market://details?id=" +
        HomeTabActivity.this.getApplication().getPackageName());
                        Intent intent1 = new Intent(Intent.ACTION_VIEW,
        uri);
                        intent1.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                        startActivity(intent1);
                    } catch (ActivityNotFoundException e) {
                        showNotice("无法打开应用市场");
                    }
                    localStorage.put(APPROVAL_SHOW,true);
                }
            }, new View.OnClickListener() {
                @Override
                public void onClick(View v) {

        BeanFactory.getBean(PluginManager.class).showFeedBack(getContext(),user
        Manager.getSoc());
```

```
                    }
                },null);
                approvalDialog.show();
            }

            @Subscribe(threadMode = ThreadMode.MAIN)
            public void onSceneChangeEvent(ActionEvent event){
                if(event.getType() == CHANGE_SECEN_GAME){
                    String position = event.getAttrStr("position");
                    Map<String,String> params = null;
                    if(!StringUtils.isEmpty(position)){
                        params = new HashMap<>();
                        params.put("position",position);
                    }

        TrackPlugin.track(getContext(),GAME_NAVIGATION_CLICK,params);
                    startActivity(new Intent(getContext(),GameActivity.class));
                }
                else if(event.getType() == CHANGE_SECEN_FORECAST){
                    String time = event.getAttrStr("time");
        else{
                        Intent intent = new
        Intent(this,ForecastDetailsActivity.class);
                        intent.putExtra("time",time);
                        startActivity(intent);
                }
                else if(event.getType() == CHANGE_SECEN_CONDITION){
                    homeViewPager.setCurrentItem(0,false);
                    setNavigationVisible(0);
                }
                else if(event.getType() == CHANGE_SECEN_POWER){
                    homeViewPager.setCurrentItem(2,false);
                    setNavigationVisible(2);
                }
                else if(event.getType() == CHECK_CALENDAR_PERMISSION){
                    final boolean isChecked = event.getAttrBoolean("checked");
                    addPermission(CALENDAR_PERMISSION, 102, (isGranted,
        permissions) -> {
                        if(isGranted){

        TrackPlugin.track(getContext(),GIFT_PERMISSION_GRATED,null);
                        }else{

        TrackPlugin.track(getContext(),GIFT_PERMISSION_REFUSE,null);
                        }
                        ActionEvent permissionEvent = new
        ActionEvent(GRANTED_CALENDAR_PERMISSION);
                        permissionEvent.setAttr("checked",isChecked);
                        permissionEvent.setAttr("isGranted",(isGranted));
                        fire(permissionEvent);
```

```
                    });
                    checkPermissions();
                }
                else if(event.getType() == PLATFORM_LOGIN_SUCCESS){
                    if(newGiftDialog != null){
                        newGiftDialog.dismiss();
                        newGiftDialog = null;
                    }
                    int position = event.getAttrInt("position");
                    if(position == 1){
                        homeViewPager.setCurrentItem(3,false);
                        setNavigationVisible(3);
                    }
                    if(userManager.isLogin()&&userManager.isVip()){
                       fire(new ActionEvent(VIP_STATUS_CHANGE));
                    }
                }
                else if(event.getType() == SHOW_LIVE_INDEX_DETAILS){
                    int id = event.getAttrInt("position");
                    int type = event.getAttrInt("type");
                    if(type != LiveIndexAdapter.FRAGMENT)return;
                    CityManager cityManager =
            BeanFactory.getBean(CityManager.class);
                    if(cityManager.getCurrentCityWeather() == null)return;
                    LiveIndexDetailsDialog dialog = new
            LiveIndexDetailsDialog(this,cityManager.getCurrentCityWeather(),id);
                    dialog.show();
                }
                else if(event.getType() == START_SHOW_APPROVAL_DIALOG){
                    String content = event.getAttrStr("content");
                    showApprovalDialog(content);
                }
                else if(event.getType() == EventType.WEATHER_REQUEST_SUCCESS){
                    showFakeAdDialog();
                }
                else if(event.getType() == EventType.CHECK_UPGRADE){
                    checkUpgrade();
                }
            }

            @Override
            protected void onResume() {
                super.onResume();
                stayTime = System.currentTimeMillis();
            }

            @Override
            protected void onPause() {
                super.onPause();
                long time = System.currentTimeMillis() - stayTime;
```

```
            if(stayTime == 0)time = 0;
            if(time<= 1000)return;
            Map<String,String> params = new HashMap<>();
            params.put("params",
        SupportUtils.transformTime((int)time/1000));
            TrackPlugin.track(this,APP_STAY_TIME,params);
        }

        private boolean isChecked = false;

        private void checkUpgrade(){
            if(isChecked)return;
            addSubscription(SubscribeUtils.doOnUIThreadDelayed(new
        Action0() {
                @Override
                public void call() {
                    logInfo("检测自动更新");
                    Beta.checkUpgrade(true,false);
                    isChecked = false;
                }
            },5000));
            isChecked = true;
        }

        @Override
        protected void onDestroy() {
            super.onDestroy();
            Constant.HOME_HAS_DEAD = true;
            BeanFactory.getBean(WeatherItemExposure.class).release();
        }

        @Override
        public void onBackPressed() {
            if(Constant.CONDITION_NEWS_STICKY){
                fire(new
        ActionEvent(EventType.CONDITION_NEWS_STICKY_RELEASE));
                Constant.CONDITION_NEWS_STICKY = false;
                return;
            }
            if(System.currentTimeMillis() - backPressTime >
        BACK_PRESS_DELAYED){
                showNotice("请再按一次确认退出");
                backPressTime = System.currentTimeMillis();
            }else{
                super.onBackPressed();
            }
        }
```