

```
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(Constant.TEST_SERVER_ENVIRONMENT);

        BeanFactory.getBean(WeatherRequestManager.class).setUseTestServer(Constant.TEST_SERVER_ENVIRONMENT);

        if(isProperty("privacy")){
            CrashHandler.getInstance().init(this);
        }
    }
}
```

```
1         LogicModule.init();
2         UMConfig.preInit(this);
3         ShareConfig.set(this);
4         initActivityLife();
5         DLTaskConfig.init(this);
6         initBugly();
7         JobManager.create(this).addJobCreator(new DemonsCreator());
8         JobConfig.setLogcatEnabled(true);
9         BeanFactory.getBean(AtAdManager.class).init(this);
10        TrackPlugin.track(this, APP_INIT, null);
11        PushPlugin.initPush(this);
12        ShareConfig.initIds();
13    }
14 }
15
16     private void initBugly() {
17         BugFixManager bugFixManager =
18         BeanFactory.getBean(BugFixManager.class);
19         Beta.canShowUpgradeActs.add(HomeTabActivity.class);
20         Beta.appChannel = Constant.CHANNEL;
21         Beta.autoCheckUpgrade = false;
22         Beta.autoCheckAppUpgrade = false;
23         Beta.showInterruptedStrategy = true;
24         Beta.upgradeCheckPeriod = 10*1000;
25         Beta.upgradeDialogLayoutId = R.layout.dialog_upgrade_custom;
26         // Beta.tipsDialogLayoutId
27         bugFixManager.init(this, BUGLY_APP_ID);
28     }
29
30     @Override
31     protected void attachBaseContext(Context base) {
32         super.attachBaseContext(base);
33         MultiDex.install(this);
34     }
35
36     private Map<String, String> getDeviceInfo() {
37         Map<String, String> params = new HashMap<>();
38         Constant.isEmulator = DeviceInfoUtils.isEmulator(this);
39         params.put("emulator", Constant.isEmulator?"模拟器":"真机");
40         if(DeviceInfoUtils.isCMCC()) {
41             params.put("sim", "中国移动");
42         }
43         else if(DeviceInfoUtils.isCMTC()) {
44             params.put("sim", "中国电信");
45         }
46         else if(DeviceInfoUtils.isCMUC()) {
47             params.put("sim", "中国联通");
48         }
49         params.put("device", SystemUtils.getDeviceInfo()+"
50 "+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;
```

```
1      import com.frostowner.magic.weather.topon.AtAdManager;
2      import com.frostowner.magic.weather.upush.PushPlugin;
3      import com.frostowner.magic.weather.utils.ConnectionUtils;
4      import com.frostowner.magic.weather.utils.StringUtils;
5      import com.frostowner.magic.weather.work.DemonsCreator;
6      import com.umeng.cconfig.UMRemoteConfig;
7
8      import java.io.FileInputStream;
9      import java.io.FileNotFoundException;
10     import java.io.FileOutputStream;
11     import java.io.IOException;
12
13     import butterknife.BindView;
14
15     import static com.frostowner.magic.weather.Constant.APP_INIT;
16     import static com.frostowner.magic.weather.Constant.APP_START;
17     import static com.frostowner.magic.weather.Constant.CHANNEL;
18     import static com.frostowner.magic.weather.Constant.HOME_HAS_DEAD;
19     import static
20     com.frostowner.magic.weather.Constant.NEED_REQUEST_AD_FLAG;
21     import static
22     com.frostowner.magic.weather.Constant.SPLASH_GUIDE_AGREE;
23     import static com.frostowner.magic.weather.Constant.SPLASH_GUIDE_SHOW;
24     import static com.frostowner.magic.weather.Constant.UM_AD_FLAG_KEY;
25     import static
26     com.frostowner.magic.weather.event.EventType.CHECK_UPGRADE;
27
28     public class SplashAdActivity extends AtSplashAdActivity{
29
30         private LocalStorage localStorage;
31
32         private boolean showGuide;
33
34         @BindView(R.id.splash_container)
35         FrameLayout splashContainer;
36
37         private PrivacyDialog privacyDialog;
38         private PermissionAlertDialog permissionAlertDialog;
39
40         public boolean canJump = false;
41         private boolean isLifeShowAd;
42         private UserActionPatternManager userActionPatternManager;
43
44         @Override
45         protected void onCreate(@Nullable Bundle savedInstanceState) {
46             super.onCreate(savedInstanceState);
47             if (!this.isTaskRoot()) { // 判断当前 activity 是不是所在任务栈
48                 的根
49                 Intent intent = getIntent();
50                 if (intent != null) {
```

```
1         String action = intent.getAction();
2         if (intent.hasCategory(Intent.CATEGORY_LAUNCHER) &&
3             Intent.ACTION_MAIN.equals(action)) {
4             Constant.SHOW_SPLASH_AD = false;
5             finish();
6             return;
7         }
8     }
9 }
10 setContentView(R.layout.activity_splash_ad);
11 localStorage = BeanFactory.getBean(LocalStorage.class);
12 userActionPatternManager =
13 BeanFactory.getBean(UserActionPatternManager.class);
14 isLifeShowAd =
15 getIntent().getBooleanExtra("isLifeShowAd", false);
16 logger.info("init life show ad from onCreate, isLifeShowAd =
17 "+isLifeShowAd);
18 permissionPrepare();
19 }
20
21 @Override
22 protected void onNewIntent(Intent intent) {
23     super.onNewIntent(intent);
24     setIntent(intent);
25     isLifeShowAd =
26 getIntent().getBooleanExtra("isLifeShowAd", false);
27     logger.info("init life show ad from onNewIntent, isLifeShowAd =
28 "+isLifeShowAd);
29     permissionPrepare();
30 }
31
32 @Override
33 protected void permissionPrepare() {
34 //     super.permissionPrepare();
35     Constant.SHOW_SPLASH_AD = true;
36     initView();
37 }
38
39 private void initView() {
40     showGuide = !isHasShowPrivacy();
41     if(showGuide) {
42         if(privacyDialog == null) {
43             privacyDialog = new PrivacyDialog(this, v -> {
44                 writeToApp("true");
45                 TrackPlugin.track(this, SPLASH_GUIDE_AGREE, null);
46                 UMConfig.init(getApplication());
47                 CrashHandler.getInstance().init(this);
48                 LogicModule.init();
49                 ShareConfig.set(getApplication());
50                 DLTaskConfig.init(getApplication());
```

```
1          JobManager.create(this).addJobCreator(new
2      DemonsCreator());
3          JobConfig.setLogcatEnabled(true);
4
5      BeanFactory.getBean(AtAdManager.class).init(getApplication());
6          TrackPlugin.track(this, APP_INIT, null);
7          PushPlugin.initPush(getApplication());
8          ShareConfig.initIds();
9          initRelaxFlag();
10         }, new View.OnClickListener() {
11             @Override
12             public void onClick(View v) {
13                 System.exit(0);
14             }
15         });
16     }
17     if(!privacyDialog.isShowing()){
18         privacyDialog.show();
19         TrackPlugin.track(this, SPLASH_GUIDE_SHOW, null);
20         logInfo("privacyDialog show");
21     }
22     }else{
23         UMConfig.init(getApplication());
24         initRelaxFlag();
25     }
26 }
27
28 private void showPermissionAlert() {
29     if(permissionAlertDialog == null) {
30         permissionAlertDialog = new PermissionAlertDialog(this,
31 isDenied -> {
32             checkPermissions();
33             }, new View.OnClickListener() {
34                 @Override
35                 public void onClick(View v) {
36                     initRelaxFlag();
37                 }
38             });
39     }
40     permissionAlertDialog.show();
41     logInfo("permissionAlertDialog show");
42 }
43
44 private void initRelaxFlag() {
45     if(localStorage == null) localStorage =
46 BeanFactory.getBean(LocalStorage.class);
47     boolean isNeedRequestAdFlag =
48 localStorage.get(NEED_REQUEST_AD_FLAG, true);
49 //     loginFromServer();
50     if(!isNeedRequestAdFlag) {
```

```
1         logInfo("no need request ad flag");
2         Constant.SHOW_AD = true;
3         execute();
4         return;
5     }
6     if(!ConnectionUtils.isConnectionAvailable(this)){
7         execute();
8         return;
9     }
10    String adFlag =
11    UMRemoteConfig.getInstance().getConfigValue(UM_AD_FLAG_KEY);
12    Log.d("ijimu","splash um remote config value : "+adFlag+" ,
13    channel : "+CHANNEL);
14    if(StringUtils.isEmpty(adFlag)){
15        Constant.SHOW_AD = false;
16        execute();
17    }else{
18        Constant.SHOW_AD = TextUtils.equals("开",adFlag);
19        execute();
20    }
21 }
22
23
24 private void execute() {
25     if(!Constant.SHOW_AD||Constant.isEmulator){
26         toMainView();
27     }else{
28         executeAd();
29     }
30     TrackPlugin.track(this,APP_START,null);
31 }
32
33 private void executeAd() {
34     showAtSplash(splashContainer);
35 }
36
37 @Override
38 protected void onPause() {
39     super.onPause();
40     canJump = false;
41 }
42
43 @Override
44 protected void onResume() {
45     super.onResume();
46     toMainView();
47 }
48
49 @Override
50 public void toMainView() {
```

```
1         if (canJump) {
2             if (!isLifeShowAd || HOME_HAS_DEAD) {
3                 startActivity(new Intent(this, HomeTabActivity.class));
4                 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
5                     startForegroundService(new Intent(getContext(),
6 NotificationHandleService.class));
7                     startForegroundService(new Intent(getContext(),
8 WeatherService.class));
9                     // startForegroundService(new Intent(getContext(),
10 LoginService.class));
11                     startForegroundService(new Intent(getContext(),
12 WidgetHandleService.class));
13                 } else {
14                     startService(new Intent(getContext(),
15 NotificationHandleService.class));
16                     startService(new Intent(getContext(),
17 WeatherService.class));
18                     // startService(new Intent(getContext(),
19 LoginService.class));
20                     startService(new Intent(getContext(),
21 WidgetHandleService.class));
22                 }
23                 if (userActionPatternManager == null) {
24                     userActionPatternManager =
25 BeanFactory.getBean(UserActionPatternManager.class);
26                 }
27                 userActionPatternManager.updateAppStartTime();
28             } else {
29                 CityManager cityManager =
30 BeanFactory.getBean(CityManager.class);
31                 YunWeather cityWeather =
32 cityManager.getCurrentCityWeather();
33                 ActionEvent event = new
34 ActionEvent(EventType.WEATHER_REQUEST_SUCCESS);
35                 if (cityWeather != null
36 && !StringUtils.isEmpty(cityWeather.getCityName())) {
37
38 event.setAttr("cityName", cityWeather.getCityName());
39                 event.setAttr("fake", "fake");
40             }
41             event.setAttr("reset", "reset");
42             fire(event);
43         }
44         this.finish();
45     } else {
46         canJump = true;
47     }
48 }
49
50 @Override
```



```
1      public void onBackPressed() {
2
3      }
4
5      @Override
6      protected void onDestroy() {
7          super.onDestroy();
8          if(privacyDialog!=null&&privacyDialog.isShowing()){
9              privacyDialog.dismiss();
10             privacyDialog = null;
11         }
12         if(permissionAlertDialog != null &&
13 permissionAlertDialog.isShowing()){
14             permissionAlertDialog.dismiss();
15             permissionAlertDialog = null;
16         }
17         Constant.SHOW_SPLASH_AD = false;
18         fire(new ActionEvent(CHECK_UPGRADE));
19     }
20
21     private boolean isHasShowPrivacy() {
22         FileInputStream fis = null;
23         byte[] buffer = null;
24         try {
25             //获取文件输入流
26             fis = openFileInput("privacy");
27             //定义保存数据的数组
28             buffer = new byte[fis.available()];
29             //从输入流中读取数据
30             fis.read(buffer);
31         } catch (FileNotFoundException e) {
32             e.printStackTrace();
33         } catch (IOException e) {
34             e.printStackTrace();
35         } finally {
36             try{
37                 if(fis!=null)fis.close();
38                 return buffer != null;
39             }catch(IOException e){
40                 e.printStackTrace();
41             }
42         }
43         return false;
44     }
45
46     private void writeToApp(String data) {
47         FileOutputStream out = null;
48         try {
49             out = openFileOutput("privacy",MODE_PRIVATE);
50             out.write(data.getBytes());
```

```
1         out.flush(); // 清理缓冲区的数据流
2         out.close(); // 关闭输出流
3     } catch (FileNotFoundException e) {
4         e.printStackTrace();
5     } catch (IOException e) {
6         e.printStackTrace();
7     }
8 }
9 }
10 package com.frostsowner.magic.weather.fragment;
11
12 import android.content.Intent;
13 import android.os.Build;
14 import android.os.Bundle;
15 import android.speech.tts.UtteranceProgressListener;
16 import android.view.View;
17
18 import androidx.annotation.NonNull;
19 import androidx.recyclerview.widget.LinearLayoutManager;
20 import androidx.recyclerview.widget.RecyclerView;
21
22 import com.chad.library.adapter.base.entity.MultiItemEntity;
23 import com.fm.commons.http.ContextHolder;
24 import com.fm.commons.logic.BeanFactory;
25 import com.fm.commons.logic.LocalStorage;
26 import com.scwang.smartrefresh.layout.SmartRefreshLayout;
27 import com.scwang.smartrefresh.layout.api.RefreshLayout;
28 import com.scwang.smartrefresh.layout.listener.OnRefreshListener;
29 import com.frostsowner.magic.weather.Constant;
30 import com.frostsowner.magic.weather.R;
31 import com.frostsowner.magic.weather.TrackPlugin;
32 import com.frostsowner.magic.weather.activity.WanYearActivity;
33 import com.frostsowner.magic.weather.adapter.CityWeatherAdapter;
34 import com.frostsowner.magic.weather.domain.AdViewItem;
35 import com.frostsowner.magic.weather.domain.AqiItem;
36 import com.frostsowner.magic.weather.domain.BaseWeather;
37 import com.frostsowner.magic.weather.domain.ConditionItem;
38 import com.frostsowner.magic.weather.domain.ForecastItem;
39 import com.frostsowner.magic.weather.domain.HourlyItem;
40 import com.frostsowner.magic.weather.domain.LiveIndexItem;
41 import com.frostsowner.magic.weather.domain.TopAlertItem;
42 import com.frostsowner.magic.weather.domain.TwoDaysItem;
43 import com.frostsowner.magic.weather.domain.WanYearItem;
44 import com.frostsowner.magic.weather.domain.weather.CityWeather;
45 import com.frostsowner.magic.weather.domain.yunyida.YunCondition;
46 import com.frostsowner.magic.weather.domain.yunyida.YunWeather;
47 import com.frostsowner.magic.weather.event.ActionEvent;
48 import com.frostsowner.magic.weather.event.EventType;
49 import com.frostsowner.magic.weather.impl.AdapterNotifyListener;
50 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
```

```
1      implements OnRefreshListener, AdapterNotifyListener ,
2      RefreshStatusListener {
3
4          public static CityWeatherFragment newInstance(String cityName) {
5              Bundle args = new Bundle();
6              args.putString("cityName", cityName);
7              CityWeatherFragment fragment = new CityWeatherFragment();
8              fragment.setArguments(args);
9              return fragment;
10         }
11
12         private final long UPDATE_DELAY = 1000*60*30;
13         private final int NEWS_STICKY_OFFSET = 100;
14
15         @BindView(R.id.swipe_container)
16         SmartRefreshLayout swipeRefreshLayout;
17         @BindView(R.id.weather_recyclerview)
18         RecyclerView weatherRecyclerView;
19         @BindView(R.id.background_view)
20         View backgroundView;
21
22         private String cityName;
23         private CityManager cityManager;
24         private SimpleDateManager simpleDateManager;
25         private CityWeatherAdapter cityWeatherAdapter;
26         private YunWeather cityWeather;
27         private TTSManager ttsManager;
28         private LocalStorage localStorage;
29         private UserManager userManager;
30         private WeatherItemExposure weatherItemExposure;
31         private long updateTime;
32         private boolean isSpeaking;
33         private boolean forceLocation;
34         private WeatherRefreshHandler weatherRefreshHandler;
35
36         @Override
37         public void onCreate(Bundle savedInstanceState) {
38             super.onCreate(savedInstanceState);
39             Bundle bundle = getArguments();
40             if(bundle != null) {
41                 cityName = bundle.getString("cityName");
42                 logError("bundle cityName : "+cityName);
43             }
44         }
45
46         @Override
47         protected void initView(View root) {
48             super.initWidget(root);
49             simpleDateManager =
50             BeanFactory.getBean(SimpleDateManager.class);
```

```
1         cityManager = BeanFactory.getBean(CityManager.class);
2         weatherItemExposure =
3     BeanFactory.getBean(WeatherItemExposure.class);
4         localStorage = BeanFactory.getBean(LocalStorage.class);
5         weatherRefreshHandler =
6     BeanFactory.getBean(WeatherRefreshHandler.class);
7         userManager = BeanFactory.getBean(UserManager.class);
8         weatherItemExposure.init();
9         cityWeatherAdapter = new CityWeatherAdapter(getActivity());
10        cityWeatherAdapter.setTopViewHeight(getCenterViewHeight());
11        cityWeatherAdapter.setAdapterNotifyListener(this);
12        LinearLayoutManager layoutManager = new
13    LinearLayoutManager(getContext()) {
14            @Override
15            public boolean canScrollVertically() {
16                return !CONDITION_NEWS_STICKY;
17            }
18        };
19        cityWeatherAdapter.setOnItemChildClickListener((adapter, view,
20    position) -> {
21            if(view.getId() == R.id.btn_speak) {
22                checkSpeak();
23            }
24            if(view.getId() == R.id.simple_fate_root) {
25                startActivity(new Intent(getContext(),
26    WanYearActivity.class));
27
28        TrackPlugin.track(getContext(), CONDITION_YEAR_CLICK, null);
29            }
30        });
31        layoutManager.setOrientation(LinearLayoutManager.VERTICAL);
32        layoutManager.setSmoothScrollbarEnabled(true);
33        weatherRecyclerView.setLayoutManager(layoutManager);
34        weatherRecyclerView.setAdapter(cityWeatherAdapter);
35        swipeRefreshLayout.setOnRefreshListener(this);
36        swipeRefreshLayout.setRefreshHeader(new
37    CustomRefreshHeader(getContext(), this));
38        swipeRefreshLayout.setEnableLoadMore(false);
39        weatherRecyclerView.addOnScrollListener(new
40    RecyclerView.OnScrollListener() {
41            @Override
42            public void onScrollStateChanged(@NonNull RecyclerView
43    recyclerView, int newState) {
44                super.onScrollStateChanged(recyclerView, newState);
45                itemViewShowReport();
46            }
47
48            @Override
49            public void onScrolled(@NonNull RecyclerView recyclerView,
50    int dx, int dy) {
```

```
1         super.onScrolled(recyclerView, dx, dy);
2         setScoreView();
3         //         newsSticky();
4     }
5     });
6
7     weatherRecyclerView.getViewTreeObserver().addOnGlobalLayoutListener(()
8     -> {
9         if(weatherRecyclerView !=
10        null)weatherRecyclerView.stopScroll();
11        });
12        initTTS();
13    }
14
15    private void itemViewShowReport() {
16        LinearLayoutManager layoutManager =
17        (LinearLayoutManager)weatherRecyclerView.getLayoutManager();
18        int position =
19        layoutManager.findLastCompletelyVisibleItemPosition()+1;
20        //        Log.e("ijimu","city weather fragment position : " +
21        position);
22        if(position == 2){
23            if(weatherItemExposure.exposure(position)){
24
25                TrackPlugin.track(getContext(),CONDITION_TODAY_SHOW,null);
26                logError("track condition");
27            }
28        }
29        if(position == 4){
30            if(weatherItemExposure.exposure(position)){
31                TrackPlugin.track(getContext(),CONDITION_24_SHOW,null);
32                logError("track 24 hourly");
33            }
34        }
35        if(position == 6){
36            if(weatherItemExposure.exposure(position)){
37                TrackPlugin.track(getContext(),CONDITION_15_SHOW,null);
38                logError("track 15 forecast");
39            }
40        }
41        if(position == 10){
42            if(weatherItemExposure.exposure(position)){
43
44                TrackPlugin.track(getContext(),CONDITION_LIVE_SHOW,null);
45                logError("track live index");
46            }
47        }
48    }
49
50    private void newsSticky() {
```

```
1         LinearLayoutManager layoutManager = (LinearLayoutManager)
2         weatherRecyclerView.getLayoutManager();
3         int position =
4         layoutManager.findLastVisibleItemPosition();
5         if(position == cityWeatherAdapter.getItemCount()-1){
6             View view =
7             layoutManager.findViewByPosition(position);
8             if(view != null && view.getTop() <= NEWS_STICKY_OFFSET){
9                 CONDITION_NEWS_STICKY = true;
10                Map<String,String> params = new HashMap<>();
11                params.put("position","首页");
12                TrackPlugin.track(getContext(),NEWS_VIEW_SHOW,params);
13                logError("track news show");
14                fire(new
15                ActionEvent(EventType.CONDITION_NEWS_STICKY_SHOW));
16                updateNewsStatus();
17            }
18        }
19    }
20
21    private void updateNewsStatus(){
22        LinearLayoutManager layoutManager = (LinearLayoutManager)
23        weatherRecyclerView.getLayoutManager();
24
25        layoutManager.scrollToPositionWithOffset(CONDITION_NEWS_STICKY?ci
26        tyWeatherAdapter.getItemCount()-1:0,0);
27
28        weatherRecyclerView.setNestedScrollingEnabled(CONDITION_NEWS_STICKY);
29        swipeRefreshLayout.setEnableRefresh(!CONDITION_NEWS_STICKY);
30    }
31
32    private void setScoreView(){
33        int offsetY =
34        weatherRecyclerView.computeVerticalScrollOffset();
35        float alpha = 0;
36        if(offsetY <= 0){
37            alpha = 0;
38        }
39        else if(offsetY < getCenterViewHeight()){
40            float scale = offsetY*1f/getCenterViewHeight();
41            alpha = 1f*scale;
42        }else{
43            alpha = 1f;
44        }
45        backgroundView.setAlpha(alpha);
46        ActionEvent event = new
47        ActionEvent(EventType.WEATHER_VIEW_SCROLLING);
48        event.setAttr("alpha",alpha);
49        fire(event);
50    }
```

```
1
2     private void updateUI(boolean force){
3         if(force || cityWeather == null){
4             cityWeather = cityManager.getTarget(cityName);
5             if(cityWeather != null &&
6 cityWeather.getCondition() != null && cityWeather.getForecast() != null){
7                 List<MultiItemEntity> data = new LinkedList<>();
8                 data.add(createTopAlertItem(cityWeather));
9                 data.add(createConditionItem(cityWeather));
10                data.add(createTwoDayItem(cityWeather));
11                data.add(createHourlyItem(cityWeather));
12                if(showLoadAd()) data.add(new
13 AdViewItem(TYPE_BANNER_1));
14                data.add(createForecastItem(cityWeather));
15                if(showLoadAd()) data.add(new AdViewItem(TYPE_NATIVE));
16                data.add(createAqiItem(cityWeather));
17                //
18                if(showLoadAd()) addData(data, createWanYearItem(cityWeather));
19                data.add(createLiveIndexItem(cityWeather));
20                if(showLoadAd()) data.add(new
21 AdViewItem(TYPE_BANNER_2));
22                //                if(showLoadAd() && Constant.AB_TEST_NEWS) data.add(new
23 AdViewItem(TYPE_NEWS));
24                cityWeatherAdapter.setNewData(data);
25                updateTime = System.currentTimeMillis();
26                logInfo("updateUI force");
27            }
28        } else {
29            cityWeatherAdapter.notifyDataSetChanged();
30            logInfo("updateUI not force");
31        }
32    }
33
34    private void addData(List<MultiItemEntity> data, BaseWeather item){
35        if(item == null) return;
36        data.add(item);
37    }
38
39    @Override
40    public void onRefresh(@NonNull RefreshLayout refreshLayout){
41        if(!ConnectionUtils.isConnectionAvailable(getContext())){
42            showNotice("网络断开, 请检查网络是否正确连接");
43            return;
44        }
45        TrackPlugin.track(getContext(), CONDITION_REFRESH, null);
46        boolean showFakeData =
47 weatherRefreshHandler.showFakeData(cityName);
48        if(showFakeData){
49            SubscribeUtils.doOnUIThreadDelayed(() -> {
50                ActionEvent actionEvent = new
```



```
1      ActionEvent(WEATHER_REQUEST_SUCCESS);
2          actionEvent.setAttr("cityName",cityName);
3          actionEvent.setAttr("fake","fake");
4          fire(actionEvent);
5      },500);
6      logInfo("city weather fragment show fake data");
7  }else{
8      Intent intent = new Intent(getApplicationContext(),
9  WeatherService.class);
10         intent.putExtra("cityName",cityName);
11         if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
12             getContext().startForegroundService(intent);
13         }else{
14             getContext().startService(intent);
15         }
16     }
17     fire(new ActionEvent(EventType.WEATHER_REFRESHING));
18 }
19
20 @Override
21 public void onNotify(){
22     logError("cityWeatherFragment notify , cityName : "+cityName);
23     itemViewShowReport();
24 }
25
26 @Subscribe(threadMode = ThreadMode.MAIN)
27 public void onDataUpdate(ActionEvent event){
28     if(event.getType() == WEATHER_REQUEST_SUCCESS){
29         String cityName = event.getAttrStr("cityName");
30         String fake = event.getAttrStr("fake");
31         String reset = event.getAttrStr("reset");
32         logInfo("CityWeatherFragment, cityName : "+cityName+" ,
33 fake : "+fake+" , reset : "+reset);
34         YunWeather target = cityManager.getTarget(cityName);
35         if(cityWeather!=null && cityWeather.isLocate() && target !=
36 null && target.isLocate()){
37             this.cityName = target.getCityName();
38         }
39         int currentPosition =
40 cityManager.getTargetIndex(this.cityName);
41         int targetPosition = cityManager.getCurrentCityIndex();
42         if(currentPosition != targetPosition)return;
43         updateUI(StringUtils.isEmpty(fake));
44         swipeRefreshLayout.finishRefresh(true);
45     }
46     else if(event.getType() ==
47 EventType.CONDITION_NEWS_STICKY_RELEASE){
48         CONDITION_NEWS_STICKY = false;
49         updateNewsStatus();
50     }
```

```
1      cityManager.getTargetIndex(this.cityName);
2      cityManager.getCurrentCityIndex();
3      else if(event.getType() == EventType.CITY_WEATHER_SPEAK) {
4          String targetCityName = event.getAttrStr("cityName");
5          int currentPosition =
6      cityManager.getTargetIndex(this.cityName);
7          int targetPosition =
8      cityManager.getTargetIndex(targetCityName);
9          if(currentPosition == targetPosition)return;
10         logInfo("city weather speak , current cityName
11 "+cityName+" , target cityName : "+targetCityName);
12         isSpeaking = false;
13         if(ttsManager != null && ttsManager.isInit()){
14             ttsManager.stop();
15         }
16         cityWeatherAdapter.speakStatus(false);
17     }
18     else if(event.getType() == EventType.LOCATION_START) {
19         int currentPosition =
20     cityManager.getTargetIndex(this.cityName);
21         if(currentPosition == 0) {
22             forceLocation = true;
23         }
24     }
25 }
26
27 @Override
28 public void onResume() {
29     super.onResume();
30     if((cityWeather!= null && cityWeather.isLocate() &&
31 System.currentTimeMillis() - updateTime >= UPDATE_DELAY) ||
32     updateTime == 0L || forceLocation){ // || !cityFakeRefresh
33         swipeRefreshLayout.autoRefresh();
34         addSubscription(SubscribeUtils.doOnUIThreadDelayed(new
35 Action0() {
36             @Override
37             public void call() {
38                 if(!swipeRefreshLayout.getState().isFinishing) {
39                     swipeRefreshLayout.finishRefresh(false);
40                 }
41             }
42             }, DRAWING_UPDATE_OVERTIME));
43             forceLocation = false;
44         }
45         initTTS();
46         updateUI(false);
47         setScoreView();
48         if(cityWeatherAdapter != null)cityWeatherAdapter.onResume();
49         logInfo("onResume , cityName : "+cityName);
50     }
```

```
1
2         @Override
3         public void onPause() {
4             super.onPause();
5             if(cityWeatherAdapter != null)cityWeatherAdapter.onPause();
6         }
7
8         @Override
9         public void refresh(int status){
10
11     }
12
13     private void initTTS(){
14         if(ttsManager == null){
15             ttsManager = new TTSManager(ContextHolder.get());
16             ttsManager.init();
17             ttsManager.addListener(new UtteranceProgressListener() {
18                 @Override
19                 public void onStart(String utteranceId) {
20                     logInfo("tts speak start");
21                     Subscription subscription =
22 SubscribeUtils.doOnUIThread(new Action0() {
23                     @Override
24                     public void call() {
25                         if(cityWeatherAdapter !=
26 null)cityWeatherAdapter.speakStatus(true);
27                     }
28                 });
29                 addSubscription(subscription);
30             }
31
32             @Override
33             public void onDone(String utteranceId){
34                 logInfo("tts speak done");
35                 Subscription subscription =
36 SubscribeUtils.doOnUIThreadDelayed(new Action0() {
37                 @Override
38                 public void call() {
39                     if(cityWeatherAdapter !=
40 null)cityWeatherAdapter.speakStatus(false);
41                 }
42                 },1000);
43                 addSubscription(subscription);
44                 ttsManager.stop();
45             }
46
47             @Override
48             public void onError(String utteranceId) {
49                 Subscription subscription =
50 SubscribeUtils.doOnUIThreadDelayed(new Action0() {
```

```
1         @Override
2         public void call() {
3             if(cityWeatherAdapter !=
4 null)cityWeatherAdapter.speakStatus(false);
5             }
6             },1000);
7             addSubscription(subscription);
8             ttsManager.stop();
9             logInfo("tts speak error : "+utteranceId);
10        }
11    });
12        if(cityWeatherAdapter !=
13 null)cityWeatherAdapter.speakStatus(false);
14        isSpeaking = false;
15    }
16 }
17
18 private void checkSpeak() {
19     if(!ttsManager.isInit()){
20         showNotice("语音播报初始化未成功");
21         return;
22     }
23     if(cityManager.getCurrentCityWeather() == null){
24         showNotice("等待当前天气数据加载...");
25         return;
26     }
27     if(isSpeaking){
28         ttsManager.stop();
29         cityWeatherAdapter.speakStatus(false);
30     }else{
31         speakWeather(cityManager.getCurrentCityWeather());
32         cityWeatherAdapter.speakStatus(true);
33     }
34     isSpeaking = !isSpeaking;
35     TrackPlugin.track(getContext(),CONDITION_TTS_CLICK,null);
36 }
37
38 private void speakWeather(YunWeather cityWeather){
39     if(cityWeather == null)return;
40     StringBuffer buffer = new StringBuffer();
41     buffer.append("天天提醒您,");
42     buffer.append(cityWeather.getCityName());
43
44     if(cityWeather.getCondition() != null){
45
46         YunCondition baseCondition = cityWeather.getCondition();
47         buffer.append("今日最高气温"+baseCondition.getTem1()+"度
48 "+", 最低气温"+baseCondition.getTem2()+"度,");
49         buffer.append("当前天气"+baseCondition.getWea());
50         buffer.append(", 气温"+baseCondition.getTem()+"度,");
```

```
1
2      buffer.append(baseCondition.getWin()+baseCondition.getWin_speed()+",");
3          buffer.append(baseCondition.getAir_tips());
4      }
5      ActionEvent speakEvent = new
6      ActionEvent(EventType.CITY_WEATHER_SPEAK);
7          speakEvent.setAttr("cityName",cityWeather.getCityName());
8          fire(speakEvent);
9
10         ttsManager.speak(buffer.toString());
11     //      ActionEvent event = new ActionEvent(EventType.TASK_COMPLETE);
12     //      event.setAttr("taskType", TaskType.AJ_TASK_TYPE_TTS);
13     //      fire(event);
14     logInfo("自动转换天气信息 : "+buffer.toString());
15     // "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
16     气温 17 度, 西南风 3 级, 略微偏热, 注意衣物变化, 紫外线强不宜户外运动。
17     车辆尾号限行 0, 5"
18     }
19
20     @Override
21     public void onDestroy() {
22         super.onDestroy();
23         if(ttsManager != null){
24             ttsManager.shutdown();
25             ttsManager = null;
26         }
27     }
28
29     private TopAlertItem createTopAlertItem(YunWeather cityWeather){
30         TopAlertItem item = new TopAlertItem();
31         if(cityWeather.getCondition() != null &&
32         cityWeather.getCondition().getAlarm() != null){
33             item.setYunAlert(cityWeather.getCondition().getAlarm());
34         }
35         return item;
36     }
37
38     private ConditionItem createConditionItem(YunWeather cityWeather){
39         ConditionItem item = new ConditionItem();
40         if(cityWeather.getCondition() != null){
41             item.setYunCondition(cityWeather.getCondition());
42         }
43         return item;
44     }
45
46     private TwoDaysItem createTwoDayItem(YunWeather cityWeather){
47         TwoDaysItem item = new TwoDaysItem();
48         if(cityWeather.getForecast() != null &&
49             cityWeather.getForecast().getData() != null &&
50             cityWeather.getForecast().getData().size() > 0){
```

```
1         item.setForecastItems(cityWeather.getForecast().getData());
2     }
3     return item;
4 }
5
6 private HourlyItem createHourlyItem(YunWeather cityWeather){
7     HourlyItem item = new HourlyItem();
8     if(cityWeather.getCondition() != null &&
9         cityWeather.getCondition().getHours() != null &&
10        cityWeather.getCondition().getHours().size() > 0){
11         item.setHours(cityWeather.getCondition().getHours());
12     }
13     if(cityWeather.getCondition() != null){
14         item.setSunrise(cityWeather.getCondition().getSunrise());
15         item.setSunset(cityWeather.getCondition().getSunset());
16     }
17     return item;
18 }
19
20 private ForecastItem createForecastItem(YunWeather cityWeather){
21     ForecastItem item = new ForecastItem();
22     if(cityWeather.getForecast() != null &&
23        cityWeather.getForecast().getData() != null &&
24        cityWeather.getForecast().getData().size() > 0){
25         item.setForecastItems(cityWeather.getForecast().getData());
26     }
27     return item;
28 }
29
30 private AqiItem createAqiItem(YunWeather cityWeather){
31     AqiItem item = new AqiItem();
32     if(cityWeather.getCondition() != null &&
33        cityWeather.getCondition().getAqi() != null){
34         item.setYunAqi(cityWeather.getCondition().getAqi());
35     }
36     return item;
37 }
38
39 private LiveIndexItem createLiveIndexItem(YunWeather cityWeather){
40     LiveIndexItem item = new LiveIndexItem();
41     if(cityWeather.getCondition() != null &&
42        cityWeather.getCondition().getZhishu() != null){
43
44         item.setYunLiveIndex(cityWeather.getCondition().getZhishu());
45     }
46     return item;
47 }
48
49 private WanYearItem createWanYearItem(CityWeather cityWeather){
50     if(cityWeather.getWanYearData() != null){
```

```
1         WanYearItem item = new WanYearItem();
2
3         item.setBaseWanYear(cityWeather.getWanYearData().getContent());
4         return item;
5     }
6     return null;
7 }
8
9     private boolean showLoadAd() {
10         if(Constant.isEmulator)return false;
11         if(userManager.isLogin()&&userManager.isVip())return false;
12         return Constant.SHOW_AD;
13     }
14
15     @Override
16     protected int getLayout() {
17         return R.layout.fragment_city_weather;
18     }
19 }
20 package com.frostsowner.magic.weather.service;
21
22 import android.content.Context;
23 import android.content.Intent;
24 import android.os.PowerManager;
25 import android.text.TextUtils;
26 import com.frostsowner.magic.weather.utils.SubscribeUtils;
27
28 import org.greenrobot.eventbus.EventBus;
29 import org.slf4j.Logger;
30 import org.slf4j.LoggerFactory;
31
32 import java.lang.reflect.Type;
33
34 import rx.Observable;
35 import rx.Subscription;
36 import rx.android.schedulers.AndroidSchedulers;
37 import rx.functions.Action0;
38 import rx.functions.Action1;
39 import rx.functions.Func0;
40 import rx.functions.Func1;
41 import rx.schedulers.Schedulers;
42
43 import static com.frostsowner.magic.weather.Constant.ACTION_WIDGET;
44 import static
45 com.frostsowner.magic.weather.Constant.CONDITION_UPDATE_INTERVAL;
46 import static
47 com.frostsowner.magic.weather.Constant.SHORT_FORECAST_UPDATE_INTERVAL;
48
49 public class WeatherService extends BaseWeatherService {
50
```

```
1         private Logger logger = LoggerFactory.getLogger(getClass());
2
3         private WeatherRequestManager requestManager;
4         private CityManager cityManager;
5
6         private boolean isRequesting = false;
7         private PowerManager pm;
8
9         @Override
10        public void onCreate() {
11            super.onCreate();
12            requestManager =
13        BeanFactory.getBean(WeatherRequestManager.class);
14            cityManager = BeanFactory.getBean(CityManager.class);
15            pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
16        }
17
18        @Override
19        public int onStartCommand(Intent intent, int flags, int startId) {
20            startForeground();
21            if(intent != null){
22                String cityName = intent.getStringExtra("cityName");
23                String action = intent.getStringExtra("notification");
24                String widget = intent.getStringExtra("action");
25                logInfo("get widget action : "+widget);
26                boolean showNoticeAlert =
27        TextUtils.equals(widget, ACTION_WIDGET);
28                if(StringUtils.isEmpty(cityName)){
29                    location(showNoticeAlert);
30                }else{
31                    YunWeather cityWeather =
32        cityManager.getTarget(cityName);
33                    if(cityWeather == null){
34                        cityWeather = cityManager.addData(cityName);
35                        requestMJData(cityWeather, showNoticeAlert);
36                    }else{
37                        if(cityWeather.isLocate()){
38                            location(showNoticeAlert);
39                        }else{
40                            requestMJData(cityWeather, showNoticeAlert);
41                        }
42                    }
43                }
44            }
45            addSubscription(SubscribeUtils.doOnUIThreadDelayed(new
46        Action0() {
47                @Override
48                public void call() {
49                    isRequesting = false;
50                }
51            })
```



```
1         }, 1000 * 30));
2         return super.onStartCommand(intent, flags, startId);
3     }
4
5     private void location(boolean showNoticeAlert) {
6         WeatherLocationClient.get().locate(new
7         WeatherLocationClient.BDLocationCallBack() {
8             @Override
9             public void locate(BDLocation bdLocation) {
10                 if (bdLocation == null || Constant.isEmulator) {
11                     YunWeather cityWeather = cityManager.getTarget(0);
12                     if (cityWeather == null) {
13                         cityWeather = new YunWeather();
14                         cityWeather.setCity("北京");
15                         cityWeather.setLocate(true);
16                         cityManager.addData("北京", true);
17                     }
18                     requestMJData(cityWeather, showNoticeAlert);
19                     return;
20                 }
21                 String city = bdLocation.getCity();
22                 String district = bdLocation.getDistrict();
23                 String address = bdLocation.getAddrStr();
24                 String semaAptag = bdLocation.getSemaAptag();
25                 logInfo("locate log cityName : " + city + district);
26                 YunWeather cityWeather =
27                 cityManager.getTarget(city + district);
28                 if (cityWeather == null) {
29                     if (StringUtils.isEmpty(city)) {
30                         cityWeather = cityManager.addData("北京", true);
31                     } else {
32                         cityWeather =
33                         cityManager.addData(city, district, address, semaAptag, true);
34                     }
35                 } else {
36
37                     if (cityWeather.getCity().equals(city) && cityWeather.getDistrict().equals
38                     (district)) {
39                         cityWeather.setCity(city);
40                         cityWeather.setDistrict(district);
41                         cityWeather.setAddress(address);
42                         cityWeather.setSemaAptag(semaAptag);
43                     } else {
44                         cityWeather =
45                         cityManager.addData(city, district, address, semaAptag, true);
46                     }
47                 }
48                 requestMJData(cityWeather, showNoticeAlert);
49             }
50         }
```

```
1         @Override
2         public void failed() {
3             YunWeather cityWeather = cityManager.getTarget(0);
4             if(cityWeather == null) {
5                 cityWeather = cityManager.addData("北京", true);
6             }
7             ActionEvent actionEvent = new
8 ActionEvent(EventType.WEATHER_REQUEST_FAILED);
9             EventBus.getDefault().post(actionEvent);
10            requestMJData(cityWeather, showNoticeAlert);
11        }
12    });
13    }
14
15    protected void requestMJData(YunWeather cityWeather, final boolean
16 showNoticeAlert) {
17        if(isRequesting) {
18            logger.info("正在更新, 无需重复请求");
19            return;
20        }
21        isRequesting = true;
22        logInfo("请求天气数据 : "+cityWeather.getCity()+" ,
23 "+cityWeather.getDistrict()+" , "+cityWeather.getAddress()+" , cityId :
24 "+cityWeather.getCityId());
25        Subscription subscription = Observable.defer(new
26 Func0<Observable<YunWeather>>() {
27            @Override
28            public Observable<YunWeather> call() {
29                return Observable.just(cityWeather);
30            }
31        })
32        .subscribeOn(Schedulers.io())
33        .flatMap(new Func1<YunWeather, Observable<YunWeather>>() {
34            @Override
35            public Observable<YunWeather> call(YunWeather mjWeather) {
36                return requestCondition(mjWeather);
37            }
38        })
39        .flatMap(new Func1<YunWeather, Observable<YunWeather>>() {
40            @Override
41            public Observable<YunWeather> call(YunWeather mjWeather) {
42                return requestForecast(mjWeather);
43            }
44        })
45        .observeOn(AndroidSchedulers.mainThread())
46        .subscribe(new Action1<YunWeather>() {
47            @Override
48            public void call(YunWeather cityData) {
49                cityManager.updateData(cityData);
50                updateWeatherNotification();
51            }
52        });
53    }
```

```
1         if (pm != null && pm.isInteractive()) {
2             ActionEvent actionEvent = new
3             ActionEvent(EventType.WEATHER_REQUEST_SUCCESS);
4             actionEvent.setAttr("cityName",
5             cityWeather.getCityName());
6             actionEvent.setAttr("action", showNoticeAlert);
7             EventBus.getDefault().post(actionEvent);
8             logger.info("刷新天气");
9         }
10        logger.info("成功、天气");
11        isRequesting = false;
12    }
13    }, new ActionListener<Throwable>() {
14        @Override
15        public void call(Throwable throwable) {
16            ActionEvent actionEvent = new
17            ActionEvent(EventType.WEATHER_REQUEST_FAILED);
18            actionEvent.setAttr("cityName",
19            cityWeather.getCityName());
20            EventBus.getDefault().post(actionEvent);
21            logger.info("失败、天气");
22            throwable.printStackTrace();
23            CrashHandler.getInstance().handleException(throwable);
24            isRequesting = false;
25        }
26    });
27    addSubscription(subscription);
28    }
29
30    private Observable<YunWeather> requestCondition(YunWeather
31    cityWeather) {
32        YunCondition yunCondition = cityWeather.getCondition();
33        if (yunCondition != null && System.currentTimeMillis() -
34        yunCondition.getLastRequestTime() < CONDITION_UPDATE_INTERVAL) {
35            logger.info("do not need update conditionData data");
36            return Observable.just(cityWeather);
37        }
38        String result =
39        requestManager.getYunYiDaWeather("v62", cityWeather.getCityId());
40        // String result2 =
41        requestManager.getCondition(cityWeather.getCityId());
42        logger.info("update condition data : ");
43        // logger.info("update conditionData data2 : "+result2);
44        Type type = new TypeToken<YunCondition>().getType();
45        YunCondition data = ResultUtil.getDomain(result, type);
46        if (data != null) {
47            data.setLastRequestTime(System.currentTimeMillis());
48            cityWeather.setCondition(data);
49            cityWeather.setUpdateTime(System.currentTimeMillis());
50            TrackPlugin.track(this, "weatherApi", null);
```

```
1         }
2         return Observable.just(cityWeather);
3     }
4
5     private Observable<YunWeather> requestForecast(YunWeather
6 cityWeather){
7         YunForecast yunForecast = cityWeather.getForecast();
8         if(yunForecast !=null && System.currentTimeMillis() -
9 yunForecast.getLastRequestTime() < SHORT_FORECAST_UPDATE_INTERVAL){
10             logger.info("do not need update forecast data");
11             return Observable.just(cityWeather);
12         }
13         String result =
14 requestManager.getYunYiDaWeather("v3",cityWeather.getCityId());
15         logger.info("update forecast data : ");
16         Type type = new TypeToken<YunForecast>() {}.getType();
17         YunForecast data = ResultUtil.getDomain(result, type);
18         if (data != null) {
19             data.setLastRequestTime(System.currentTimeMillis());
20             cityWeather.setForecast(data);
21             cityWeather.setUpdateTime(System.currentTimeMillis());
22             TrackPlugin.track(this, "weatherApi", null);
23         }
24         return Observable.just(cityWeather);
25     }
26 }
27 package com.frostowner.magic.weather.adapter;
28
29 import android.content.Intent;
30 import android.content.res.Resources;
31 import android.graphics.Color;
32 import android.graphics.Typeface;
33 import android.graphics.drawable.Drawable;
34 import android.os.Build;
35 import android.util.DisplayMetrics;
36 import android.util.Log;
37 import android.util.SparseArray;
38 import android.util.SparseBooleanArray;
39 import android.view.View;
40 import static
41 com.frostowner.magic.weather.domain.BaseWeather.TYPE_CONDITION;
42 com.frostowner.magic.weather.domain.BaseWeather.TYPE_TWO_DAYS;
43
44 public class CityWeatherAdapter extends AtAdFragmentPagerAdapter {
45
46     private int topViewHeight;
47
48     private SparseArray<WeatherHourlyAdapter> hourlyAdapterArray;
49     private SparseArray<LiveIndexAdapter> liveIndexAdapterArray;
50     private SparseArray<WeatherWeekAdapter> forecastAdapterArray;
```

```
1      private SparseBooleanArray forceUpdate;
2
3      private SimpleDateFormat dateFormat;
4      private SimpleDateFormat itemDayFormat;
5
6      private boolean hourlySlip = false;
7      private boolean forecastSlip = false;
8      private NewsView newsView;
9      private SpeakHolder speakHolder;
10
11     private AdapterNotifyListener adapterNotifyListener;
12     private NewsTypeManager newsTypeManager;
13     private SimpleDateManager simpleDateManager;
14     private UserManager userManager;
15
16     public CityWeatherAdapter(FragmentActivity activity) {
17         super(activity);
18         addItemType(TYPE_TOP_ALERT, R.layout.item_alert_top_view);
19         addItemType(TYPE_CONDITION, R.layout.item_condition_view);
20         addItemType(TYPE_TWO_DAYS, R.layout.item_two_days_view);
21         addItemType(TYPE_24_HOURLY, R.layout.item_24_hourly_view);
22         addItemType(TYPE_FORECAST, R.layout.item_forecast_view);
23         addItemType(TYPE_AQI, R.layout.item_aqi_view);
24         addItemType(TYPE_BANNER_1, R.layout.item_ad_banner_1_view);
25         addItemType(TYPE_CALENDAR, R.layout.layout_simple_fate_1);
26         addItemType(TYPE_BANNER_2, R.layout.item_ad_banner_2_view);
27         addItemType(TYPE_LIVE, R.layout.item_live_index_view);
28         addItemType(TYPE_NATIVE, R.layout.item_ad_native_view);
29     }
30     if (Constant.AB_TEST_NEWS) addItemType(TYPE_NEWS, com.shl.weather.baidu.R.
31     layout.layout_news);
32     hourlyAdapterArray = new SparseArray<>();
33     liveIndexAdapterArray = new SparseArray<>();
34     forecastAdapterArray = new SparseArray<>();
35     forceUpdate = new SparseBooleanArray();
36     simpleDateManager =
37     BeanFactory.getBean(SimpleDateManager.class);
38     newsTypeManager = BeanFactory.getBean(NewsTypeManager.class);
39     userManager = BeanFactory.getBean(UserManager.class);
40     dateFormat = simpleDateManager.getFormatMap("yyyy-MM-dd");
41     itemDayFormat = simpleDateManager.getFormatMap("MM 月 dd 日");
42 }
43
44     public void setTopViewHeight(int value) {
45         this.topViewHeight = value;
46     }
47
48     public void setAdapterNotifyListener(AdapterNotifyListener
49     adapterNotifyListener) {
50         this.adapterNotifyListener = adapterNotifyListener;
```

```
1      }
2
3      @Override
4      public void setNewData(@Nullable List<MultiItemEntity> data) {
5          forceUpdate.clear();
6          for(int i = 0; i < data.size();i++){
7              int itemType = data.get(i).getItemType();
8              forceUpdate.put(itemType, true);
9          }
10         super.setNewData(data);
11     }
12
13     @Override
14     protected void convert(BaseViewHolder helper, MultiItemEntity
15     item){
16         //      Log.d("ijimu","convert, type : "+helper.getItemViewType());
17         switch (helper.getItemViewType()) {
18             case TYPE_TOP_ALERT:
19                 updateTopAlert(helper, item);
20                 break;
21             case TYPE_CONDITION:
22                 updateCondition(helper, item);
23                 break;
24             case TYPE_TWO_DAYS:
25                 updateTwoDays(helper, item);
26                 break;
27             case TYPE_24_HOURLY:
28                 update24Hourly(helper, item);
29                 break;
30             case TYPE_FORECAST:
31                 updateForecast(helper, item);
32                 break;
33             case TYPE_AQI:
34                 updateAqi(helper, item);
35                 break;
36             case TYPE_CALENDAR:
37                 updateWanYearView(helper, item);
38                 break;
39             case TYPE_LIVE:
40                 updateLiveIndex(helper, item);
41                 break;
42             case TYPE_BANNER_1:
43                 updateBanner1(helper);
44                 break;
45             case TYPE_BANNER_2:
46                 updateBanner2(helper);
47                 break;
48             case TYPE_NATIVE:
49                 updateNative(helper);
50                 break;
```

```
1         case TYPE_NEWS:
2             updateNews(helper);
3             break;
4         }
5         if(adapterNotifyListener !=
6 null)adapterNotifyListener.onNotify();
7     }
8
9     private void updateTopAlert(BaseViewHolder helper, MultiItemEntity
10 item){
11         boolean force = forceUpdate.get(item.getItemType());
12         if(!force)return;
13         TopAlertItem topAlertItem = (TopAlertItem)item;
14         if(topViewHeight != 0){
15             LinearLayout topView = helper.getView(R.id.top_view);
16             RecyclerView.LayoutParams layoutParams =
17 (RecyclerView.LayoutParams)topView.getLayoutParams();
18             layoutParams.height = topViewHeight;
19             topView.setLayoutParams(layoutParams);
20         }
21
22         TextView tvAlert = helper.getView(R.id.tv_alert);
23         if(topAlertItem.getYunAlert() != null
24 && !StringUtils.isEmpty(topAlertItem.getYunAlert().getAlarm_content()))
25     {
26         tvAlert.setVisibility(View.VISIBLE);
27
28         tvAlert.setText(topAlertItem.getYunAlert().getAlarm_content());
29         tvAlert.setSelected(true);
30         tvAlert.setOnClickListener(v -> {
31             Intent intent = new Intent();
32             intent.setClass(activity, WeatherAlertActivity.class);
33             activity.startActivity(intent);
34         });
35     }else{
36         tvAlert.setVisibility(View.INVISIBLE);
37     }
38     forceUpdate.put(item.getItemType(), false);
39 }
40
41 private void updateCondition(BaseViewHolder helper, MultiItemEntity
42 item){
43     boolean force = forceUpdate.get(item.getItemType());
44     if(!force)return;
45     ConditionItem conditionItem = (ConditionItem)item;
46     YunCondition yunCondition = conditionItem.getYunCondition();
47     helper.addOnClickListener(R.id.btn_speak);
48     LinearLayout baseWeatherView = helper.getView(R.id.item_root);
49     if(yunCondition != null){
50         baseWeatherView.setVisibility(View.VISIBLE);
```

```
1         }else{
2             baseWeatherView.setVisibility(View.INVISIBLE);
3             return;
4         }
5         RecyclerView.LayoutParams layoutParams =
6         (RecyclerView.LayoutParams)baseWeatherView.getLayoutParams();
7         layoutParams.height =
8         (int)activity.getResources().getDimension(R.dimen.qb_px_200);
9         baseWeatherView.setLayoutParams(layoutParams);
10        baseWeatherView.setOnClickListener(new View.OnClickListener() {
11            @Override
12            public void onClick(View v) {
13                Intent intent = new Intent();
14                intent.setClass(activity,
15                ConditionDetailsActivity.class);
16                activity.startActivity(intent);
17            }
18        });
19
20        TextView tvWeather = helper.getView(R.id.tv_weather);
21        tvWeather.setText(yunCondition.getWea());
22
23        TextView tvTemperature = helper.getView(R.id.tv_temperature);
24        Typeface typeface =
25        Typeface.createFromAsset(activity.getAssets(),"fonts/pf_tc_tiny.ttf");
26        tvTemperature.setTypeface(typeface);
27        tvTemperature.setText(yunCondition.getTem()+"° ");
28
29        TextView tvWind = helper.getView(R.id.tv_wind);
30
31        tvWind.setText(yunCondition.getWin()+yunCondition.getWin_speed());
32
33        TextView tvHumidity = helper.getView(R.id.tv_humidity);
34        tvHumidity.setText("湿度"+yunCondition.getHumidity());
35
36        TextView tvTips = helper.getView(R.id.tv_tips);
37        tvTips.setText(yunCondition.getAir_tips());
38        tvTips.setSelected(true);
39
40        TextView tvPreesure = helper.getView(R.id.tv_pressure);
41        tvPreesure.setText("气压"+yunCondition.getPressure()+"hPa");
42
43        View groupAqi = helper.getView(R.id.group_aqi);
44        TextView tvAqi = helper.getView(R.id.tv_aqi);
45        ImageView imgAqi = helper.getView(R.id.img_aqi);
46        if(!StringUtils.isEmpty(yunCondition.getAir())) {
47            groupAqi.setVisibility(View.VISIBLE);
48            String value = yunCondition.getAir();
49            String quality =
50            WeatherUtils.getAqiValue(yunCondition.getAir());
```



```
1         tvAqi.setText(quality+" "+value);
2         if(quality.equals("优")){
3
4             imgAqi.setColorFilter(activity.getResources().getColor(R.color.color_qu
5             ality_1));
6         }
7         else if(quality.equals("良")){
8
9             imgAqi.setColorFilter(activity.getResources().getColor(R.color.color_qu
10            ality_2));
11        }
12        else if(quality.equals("轻度")){
13
14            imgAqi.setColorFilter(activity.getResources().getColor(R.color.color_qu
15            ality_3));
16        }
17        else if(quality.equals("中度")){
18
19            imgAqi.setColorFilter(activity.getResources().getColor(R.color.color_qu
20            ality_4));
21        }
22        else if(quality.equals("重度") || quality.equals("严重")){
23
24            imgAqi.setColorFilter(activity.getResources().getColor(R.color.color_qu
25            ality_5));
26        }
27        groupAqi.setOnClickListener(new View.OnClickListener() {
28            @Override
29            public void onClick(View v) {
30                Intent intent = new Intent(activity,
31                AqiDetailsActivity.class);
32                activity.startActivity(intent);
33                Map<String,String> params = new HashMap<>();
34                params.put("position","空气图标");
35                TrackPlugin.track(activity,AQI_CLICK,params);
36            }
37        });
38    }else{
39        groupAqi.setVisibility(View.INVISIBLE);
40    }
41    VoiceWaveView voiceWaveView =
42    helper.getView(R.id.voiceWaveView0);
43    ImageView voiceClose = helper.getView(R.id.voice_close);
44    setCurrentVoiceView(voiceWaveView,voiceClose);
45    forceUpdate.put(item.getItemType(),false);
46    }
47
48    private void updateTwoDays(BaseViewHolder helper, MultiItemEntity
49    item){
50        boolean force = forceUpdate.get(item.getItemType());
```

```
1         if(!force)return;
2         TwoDaysItem twoDaysItem = (TwoDaysItem)item;
3         View todayPanel = helper.getView(R.id.item_today_panel);
4         View tomorrowPanel = helper.getView(R.id.item_tomorrow_panel);
5         List<YunForecastItem> forecastData =
6         twoDaysItem.getForecastItems();
7         YunForecastItem todayForecast = null;
8         YunForecastItem tomorrowForecast = null;
9         for(int i = 0; i < forecastData.size();i++){
10             YunForecastItem forecastItem = forecastData.get(i);
11             String todayTime = simpleDateManager.transformTime("yyyy-
12 MM-dd");
13             String targetTime = forecastItem.getDate();
14             if(todayTime.equals(targetTime)){
15                 todayForecast = forecastItem;
16                 if(i+1 < forecastData.size())tomorrowForecast =
17 forecastData.get(i+1);
18                 break;
19             }
20         }
21         updateBottomWeather(todayPanel,"今天",todayForecast);
22         updateBottomWeather(tomorrowPanel,"明天",tomorrowForecast);
23         todayPanel.setOnClickListener(v -> activity.startActivity(new
24 Intent(activity,ConditionDetailsActivity.class)));
25         final YunForecastItem finalTomorrowForecast = tomorrowForecast;
26         tomorrowPanel.setOnClickListener(v -> {
27             ActionEvent event = new
28 ActionEvent(EventType.CHANGE_SECEN_FORECAST);
29             if(finalTomorrowForecast !=
30 null)event.setAttr("time",finalTomorrowForecast.getDate());
31             EventBus.getDefault().post(event);
32         });
33         forceUpdate.put(item.getItemType(),false);
34     }
35
36     private void updateBottomWeather(View root, String day,
37 YunForecastItem baseForecast){
38         if(baseForecast == null){
39             root.setVisibility(View.INVISIBLE);
40             return;
41         }
42         root.setVisibility(View.VISIBLE);
43         TextView tvWeek = root.findViewById(R.id.tv_week);
44         TextView tvTempMinMax =
45 root.findViewById(R.id.tv_temp_min_max);
46         ImageView iconWeather = root.findViewById(R.id.icon_weather);
47         tvWeek.setText(day);
48
49         TextView tvDay = root.findViewById(R.id.tv_day);
50
```

```
1      tvDay.setText(StringUtils.formatTransformStyle(baseForecast.getDate(), dateFormat, itemDayFormat));
2
3
4
5      tvTempMinMax.setText(baseForecast.getTem1()+"° /"+baseForecast.getTem2(
6      )+"° ");
7
8      iconWeather.setImageResource(WeatherResUtils.getYunWeatherDayIcon(baseForecast.getWea_img()));
9
10
11         TextView tvWeather = root.findViewById(R.id.tv_weather);
12         tvWeather.setVisibility(View.VISIBLE);
13         tvWeather.setText(baseForecast.getWea());
14     }
15
16     private void update24Hourly(BaseViewHolder helper, MultiItemEntity item) {
17
18         boolean force = forceUpdate.get(item.getItemType());
19         if(!force)return;
20         HourlyItem hourlyItem = (HourlyItem)item;
21         HorizontalScrollView hourlyListView =
22     helper.getView(R.id.hourly_list_view);
23         hourlyListView.scrollTo(PixelUtils.dp2px(activity, 20), 0);
24         // hourlyView = root.findViewById(R.id.group_hourly);
25         if(Build.VERSION.SDK_INT >= 23) {
26             hourlyListView.setOnScrollChangeListener(new
27     View.OnScrollChangeListener() {
28                 @Override
29                 public void onScrollChange(View v, int scrollX, int
30     scrollY, int oldScrollX, int oldScrollY) {
31                     if(hourlyListView.getScrollX() > 0 && !hourlySlip){
32                         hourlySlip = true;
33                         Log.e("ijimu", "hourly slip :
34     "+hourlyListView.getScrollX());
35
36     TrackPlugin.track(activity, CONDITION_24_SLIP, null);
37                     }
38                 }
39             });
40         }
41         if(hourlyItem.getHours() != null && hourlyItem.getHours() !=
42     null) {
43             hourlyListView.setVisibility(View.VISIBLE);
44             List<YunHours> baseHourlyList = hourlyItem.getHours();
45             RecyclerView recyclerView =
46     helper.getView(R.id.weather_hourly_recyclerview);
47             WeatherHourlyAdapter adapter =
48     hourlyAdapterArray.get(helper.getLayoutPosition());"+forceUpdate);
49             if(adapter == null) {
50                 adapter = new WeatherHourlyAdapter(activity);
```

```
1          LinearLayoutManager layoutManager = new
2      LinearLayoutManager(activity);
3
4      layoutManager.setOrientation(LinearLayoutManager.HORIZONTAL);
5          recyclerView.setAdapter(adapter);
6          recyclerView.setLayoutManager(layoutManager);
7
8      hourlyAdapterArray.put(helper.getLayoutPosition(), adapter);
9          }
10         adapter.setNewData(hourlyItem.getHours());
11         forceUpdate.put(item.getItemType(), false);
12
13         LineChart lineChart =
14     helper.getView(R.id.weather_hourly_chart_1);
15
16         if(baseHourlyList == null)return;
17         //chartviewdouble
18         int size = baseHourlyList.size();
19         RelativeLayout.LayoutParams layoutParams =
20     (RelativeLayout.LayoutParams)lineChart.getLayoutParams();
21         int width = getItemWidth();
22         layoutParams.leftMargin = width/20;
23         layoutParams.rightMargin = width/20;
24         lineChart.setLayoutParams(layoutParams);
25
26         ArrayList<Entry> tempData = new ArrayList<>();
27         for(int i = 0; i < size;i++){
28             YunHours yunHours = baseHourlyList.get(i);
29             Entry entry = new Entry();
30             float tempY = StringUtils.isEmpty(yunHours.getTem())?-
31     1000:Integer.parseInt(yunHours.getTem());
32             entry.setX(i);
33             entry.setY(tempY);
34             tempData.add(entry);
35         }
36
37         setLineChart(lineChart, tempData, Color.parseColor("#ffffff"), Color.parse
38     Color("#1E70FF"), R.drawable.bg_blue_transform_vertical);
39         }else{
40             hourlyListView.setVisibility(View.GONE);
41         }
42         TextView riseInfoTv = helper.getView(R.id.rise_info);
43         if(!StringUtils.isEmpty(hourlyItem.getSunrise())&&
44             !StringUtils.isEmpty(hourlyItem.getSunset())) {StringBuffer
45     buffer = new StringBuffer();
46             buffer.append("日出\t");
47             buffer.append(hourlyItem.getSunrise()+" / ");
48             buffer.append("日落\t");
49             buffer.append(hourlyItem.getSunset());
50             riseInfoTv.setText(buffer.toString());
```

```
1         riseInfoTv.setVisibility(View.VISIBLE);
2     }else{
3         riseInfoTv.setVisibility(View.INVISIBLE);
4     }
5     forceUpdate.put(item.getItemType(), false);
6 }
7
8     private void setLineChart(LineChart lineChart, ArrayList<Entry>
9 data, int circleColor, int circleHoleColor, int filterDrawable) {
10         lineChart.getLegend().setEnabled(false);
11         lineChart.setDrawBorders(false);
12         lineChart.setDrawGridBackground(false);
13         lineChart.setScaleEnabled(false);
14         lineChart.getXAxis().setDrawLabels(false);
15         lineChart.getXAxis().setDrawAxisLine(false);
16         lineChart.getXAxis().setDrawGridLines(false);
17         lineChart.getAxisLeft().setDrawLabels(false);
18         lineChart.getAxisLeft().setDrawGridLines(false);
19         lineChart.getAxisLeft().setDrawAxisLine(false);
20         lineChart.getAxisRight().setDrawGridLines(false);
21         lineChart.getAxisRight().setDrawLabels(false);
22         lineChart.getAxisRight().setDrawAxisLine(false);
23         lineChart.getXAxis().setGranularity(1f);
24         lineChart.getXAxis().setLabelCount(data.size(), true);
25         lineChart.setTouchEnabled(false);
26         lineChart.getDescription().setEnabled(false);
27         lineChart.getXAxis().setAvoidFirstLastClipping(true);
28
29         LineDataSet set1 = new LineDataSet(data, "");
30         set1.setMode(LineDataSet.Mode.CUBIC_BEZIER);
31         set1.setCircleColor(circleColor);
32         set1.setCircleHoleColor(circleHoleColor);
33         set1.setColor(circleHoleColor);
34         set1.setValueTextColor(circleHoleColor);
35         set1.setCircleRadius(4f);
36         set1.setCircleHoleRadius(3f);
37         set1.setDrawCircleHole(true);
38         set1.setDrawCircles(true);
39         set1.setLineWidth(2);
40         set1.setFormLineWidth(1f);
41         set1.setValueTextSize(15f);
42         set1.setDrawFilled(true);
43         set1.setValueFormatter(new ValueFormatter() {
44             @Override
45             public String getFormattedValue(float value) {
46                 return (int)value+"° ";
47             }
48         });
49         Drawable drawable = ContextCompat.getDrawable(activity,
50 filterDrawable);
```

```
1         set1.setFillDrawable(drawable);
2         ArrayList<ILineDataSet> dataSets = new ArrayList<>();
3         dataSets.add(set1);
4         lineChart.setData(new LineData(dataSets));
5         lineChart.invalidate();
6     }
7
8     private void updateForecast(BaseViewHolder helper, MultiItemEntity
9 item) {
10         boolean force = forceUpdate.get(item.getItemType());
11         if(!force)return;
12         ForecastItem forecastItem = (ForecastItem)item;
13         HorizontalScrollView forecastListView =
14 helper.getView(R.id.forecast_list_view);
15         forecastListView.post(new Runnable() {
16             @Override
17             public void run() {
18
19 forecastListView.scrollTo(PixelUtils.dp2px(activity, 20), 0);
20             }
21         });
22         if(Build.VERSION.SDK_INT >= 23) {
23             forecastListView.setOnScrollChangeListener(new
24 View.OnScrollChangeListener() {
25                 @Override
26                 public void onScrollChange(View v, int scrollX, int
27 scrollY, int oldScrollX, int oldScrollY) {
28                     if(forecastListView.getScrollX() > 0
29 && !forecastSlip) {
30                         forecastSlip = true;
31                         Log.e("ijimu", "forecast slip :
32 "+forecastListView.getScrollX());
33
34 TrackPlugin.track(activity, CONDITION_15_SLIP, null);
35                     }
36                 }
37             });
38         }
39         TextView btnMore = helper.getView(R.id.btn_more);
40         btnMore.setOnClickListener(new View.OnClickListener() {
41             @Override
42             public void onClick(View v) {
43                 ActionEvent event = new
44 ActionEvent(EventType.CHANGE_SECEN_FORECAST);
45                 EventBus.getDefault().post(event);
46                 Map<String, String> params = new HashMap<>();
47                 params.put("position", "查看更多");
48                 TrackPlugin.track(activity, CONDITION_15_CLICK, params);
49             }
50         });
```

```
1         if(forecastItem.getForecastItems() != null &&
2 forecastItem.getForecastItems().size() > 0){
3             forecastListView.setVisibility(View.VISIBLE);
4             List<YunForecastItem> forecastItems =
5 forecastItem.getForecastItems();
6             forecastItems =
7 forecastItems.subList(0, Math.min(forecastItems.size(), 36));
8             RecyclerView recyclerView =
9 helper.getView(R.id.weather_week_recyclerview);
10            WeatherWeekAdapter adapter =
11 forecastAdapterArray.get(helper.getLayoutPosition());
12            if(adapter == null){
13                adapter = new WeatherWeekAdapter(activity);
14                recyclerView.setAdapter(adapter);
15                LinearLayoutManager layoutManager = new
16 LinearLayoutManager(activity);
17
18                layoutManager.setOrientation(LinearLayoutManager.HORIZONTAL);
19                recyclerView.setLayoutManager(layoutManager);
20                adapter.setNewData(forecastItems);
21
22 forecastAdapterArray.put(helper.getLayoutPosition(), adapter);
23            }
24            adapter.setNewData(forecastItems);
25
26            if(forecastItems == null)return;
27            int size = forecastItems.size();
28
29            LinearLayout groupWeatherChart =
30 helper.getView(R.id.group_weather_chart);
31            RelativeLayout.LayoutParams layoutParams =
32 (RelativeLayout.LayoutParams)groupWeatherChart.getLayoutParams();
33            int width = getItemWidth();
34            layoutParams.leftMargin = width/20;
35            layoutParams.rightMargin = width/20;
36            groupWeatherChart.setLayoutParams(layoutParams);
37
38            LineChart lineChartMax =
39 helper.getView(R.id.weather_chart_max);
40            LineChart lineChartMin =
41 helper.getView(R.id.weather_chart_min);
42
43            ArrayList<Entry> dayTempData = new ArrayList<>();
44            ArrayList<Entry> nightTempData = new ArrayList<>();
45            for(int i = 0; i < size;i++){
46                YunForecastItem baseForecast = forecastItems.get(i);
47                dayTempData.add(new
48 Entry(i,StringUtils.isEmpty(baseForecast.getTem1())?0:Integer.parseInt(
49 baseForecast.getTem1())));
50                nightTempData.add(new
```

```
1      Entry(i, StringUtils.isEmpty(baseForecast.getTem2())?0:Integer.parseInt(
2      baseForecast.getTem2())));
3          }
4
5      setLineChart(lineChartMax, dayTempData, Color.parseColor("#ffffff"), Color
6      .parseColor("#FFB41E"), R.drawable.bg_orange_transform_vertical);
7
8      setLineChart(lineChartMin, nightTempData, Color.parseColor("#ffffff"), Col
9      or.parseColor("#1E70FF"), R.drawable.bg_blue_transform_vertical);
10         }else{
11             forecastListView.setVisibility(View.GONE);
12         }
13         forceUpdate.put(item.getItemType(), false);
14     }
15
16     private void updateAqi(BaseViewHolder helper, MultiItemEntity
17     item){
18         boolean force = forceUpdate.get(item.getItemType());
19         if(!force)return;
20         AqiItem aqiItem = (AqiItem)item;
21         LinearLayout aqiParent = helper.getView(R.id.item_root);
22         WaveView waveView = helper.getView(R.id.aqi_capsule);
23         waveView.setShapeType(WaveView.ShapeType.CAPSULE);
24         if(aqiItem.getYunAqi() != null){
25             aqiParent.setVisibility(View.VISIBLE);
26             YunAqi baseAqi = aqiItem.getYunAqi();
27             String quality =
28             WeatherUtils.getAqiValue(baseAqi.getAir());
29             helper.setText(R.id.tv_aqi_capsule_value, baseAqi.getAir());
30             helper.setText(R.id.tv_aqi_capsule_quality, quality);
31             helper.setVisible(R.id.btn_more, true);
32             if(quality.equals("优")){
33                 waveView.setWaveColor(Color.parseColor("#2021CA49"),
34                 Color.parseColor("#4021CA49"),
35                 Color.parseColor("#8021CA49"));
36
37             helper.setTextColor(R.id.tv_aqi_capsule_name, activity.getResources().ge
38             tColor(R.color.color_quality_1));
39         }
40         else if(quality.equals("良")){
41             waveView.setWaveColor(Color.parseColor("#20FEDD00"),
42             Color.parseColor("#40FEDD00"),
43             Color.parseColor("#80FEDD00"));
44
45             helper.setTextColor(R.id.tv_aqi_capsule_name, activity.getResources().ge
46             tColor(R.color.color_quality_2));
47         }
48         else if(quality.equals("轻度")){
49             waveView.setWaveColor(Color.parseColor("#20FFA200"),
50             Color.parseColor("#40FFA200"),
```



```
1          Color.parseColor("#80FFA200"));
2
3      helper.setTextColor(R.id.tv_aqi_capsule_name, activity.getResources().ge
4      tColor(R.color.color_quality_3));
5      }
6      else if (quality.equals("中度")) {
7          waveView.setWaveColor(Color.parseColor("#20FE1E1E"),
8          Color.parseColor("#40FE1E1E"),
9          Color.parseColor("#80FE1E1E"));
10
11      helper.setTextColor(R.id.tv_aqi_capsule_name, activity.getResources().ge
12      tColor(R.color.color_quality_4));
13      }
14      else if (quality.equals("重度")) {
15          waveView.setWaveColor(Color.parseColor("#20A20050"),
16          Color.parseColor("#40A20050"),
17          Color.parseColor("#80A20050"));
18
19      helper.setTextColor(R.id.tv_aqi_capsule_name, activity.getResources().ge
20      tColor(R.color.color_quality_5));
21      }
22      else if (quality.equals("严重")) {
23          waveView.setWaveColor(Color.parseColor("#2081022B"),
24          Color.parseColor("#4081022B"),
25          Color.parseColor("#8081022B"));
26
27      helper.setTextColor(R.id.tv_aqi_capsule_name, activity.getResources().ge
28      tColor(R.color.color_quality_6));
29      }
30      WaveHelper waveHelper = new WaveHelper(waveView);
31      waveHelper.start();
32
33      setAqiItem(helper.getView(R.id.item_pm25), "PM2.5(细颗粒
34      物)", baseAqi.getPm25());
35      setAqiItem(helper.getView(R.id.item_pm10), "PM10(可吸入颗粒
36      物)", baseAqi.getPm10());
37      setAqiItem(helper.getView(R.id.item_no2), "NO2(二氧化
38      氮)", baseAqi.getNo2());
39      setAqiItem(helper.getView(R.id.item_so2), "SO2(二氧化
40      硫)", baseAqi.getSo2());
41      setAqiItem(helper.getView(R.id.item_co), "CO(一氧化
42      碳)", baseAqi.getCo());
43      setAqiItem(helper.getView(R.id.item_o3), "O3(臭氧指
44      数)", baseAqi.getO3());
45
46      aqiParent.setOnClickListener(new View.OnClickListener() {
47          @Override
48          public void onClick(View v) {
49              Intent intent = new Intent(activity,
50              AqiDetailsActivity.class);
```

```

1          activity.startActivity(intent);
2          Map<String,String> params = new HashMap<>();
3          params.put("position","查看更多");
4          TrackPlugin.track(activity,AQI_CLICK,params);
5      }
6      });
7  }
8  else{
9      waveView.setWaveColor(Color.parseColor("#20b4b4b4"),
10      Color.parseColor("#40b4b4b4"),
11      Color.parseColor("#80b4b4b4"));
12      helper.setVisible(R.id.btn_more,false);
13      helper.setText(R.id.tv_aqi_capsule_value,"0");
14      helper.setText(R.id.tv_aqi_capsule_quality,"无");
15
16      helper.setTextColor(R.id.tv_aqi_capsule_name,Color.parseColor("#ffffff"
17      ));
18      setAqiItem(helper.getView(R.id.item_pm25),"PM2.5(细颗粒
19      物)","--");
20      setAqiItem(helper.getView(R.id.item_pm10),"PM10(可吸入颗粒
21      物)","--");
22      setAqiItem(helper.getView(R.id.item_no2),"NO2(二氧化氮)","-
23      -");
24      setAqiItem(helper.getView(R.id.item_so2),"SO2(二氧化硫)","-
25      -");
26      setAqiItem(helper.getView(R.id.item_co),"CO(一氧化碳)","--
27      ");
28      setAqiItem(helper.getView(R.id.item_o3),"O3(臭氧指数)","--
29      ");
30  }
31  forceUpdate.put(item.getItemType(),false);
32  }
33
34  private void setAqiItem(View itemView,String name,String value){
35      TextView tvName = itemView.findViewById(R.id.item_name);
36      tvName.setText(name);
37      TextView tvValue = itemView.findViewById(R.id.item_value);
38      tvValue.setText(value);
39  }
40
41  private void updateWanYearView(BaseViewHolder helper,
42  MultiItemEntity item){
43      helper.addOnClickListener(R.id.simple_fate_root);
44      Calendar calendar = Calendar.getInstance();
45      calendar.setTime(new Date());
46      int year = calendar.get(Calendar.YEAR);
47      int month = calendar.get(Calendar.MONTH)+1;
48      int day = calendar.get(Calendar.DAY_OF_MONTH);
49      WanYearItem wanYearItem = (WanYearItem)item;
50      BaseYear baseYear = wanYearItem.getBaseWanYear().getBaseYear();

```

```

1         BaseMonth baseMonth =
2         wanYearItem.getBaseWanYear().getBaseMonth();
3         BaseDay baseDay = wanYearItem.getBaseWanYear().getBaseDay();
4         if(baseMonth!= null && baseDay != null){
5             helper.setText(R.id.tv_time_1,"农历
6             "+baseMonth.getNongliyue()+baseDay.getNongliri());
7         }
8         if(baseYear != null && baseDay!=null && baseMonth!=null){
9
10        helper.setText(R.id.tv_time_3,baseYear.getNongli()+"(" +baseYear.getShuxiang()+")"+"年 "+baseMonth.getSizhu_yuezhuh()+"月
11        "+baseDay.getSizhu_rizhu()+"日");
12        }
13        if(baseDay != null){
14
15        helper.setText(R.id.tv_yi,baseDay.getRifen_yi().replaceAll(",",""));
16
17        helper.setText(R.id.tv_ji,baseDay.getRifen_ji().replaceAll(",",""));
18        }
19        int weekCount =
20        CalendarUtil.getWeekCountBetweenBothCalendar(year, 1, 1, year, month, day, 1)
21        ;
22        int week = SupportUtils.getWeek(year, month, day);
23        helper.setText(R.id.tv_time_2,"第"+weekCount+"周
24        "+SupportUtils.weekNameTransform(week));
25        }
26
27
28        private void updateLiveIndex(BaseViewHolder helper, MultiItemEntity
29        item){
30            boolean force = forceUpdate.get(item.getItemType());
31            if(!force)return;
32            LiveIndexItem liveIndexItem = (LiveIndexItem) item;
33            if(liveIndexItem.getYunLiveIndex() != null){
34                RecyclerView recyclerView =
35                helper.getView(R.id.live_index_recycler_view);
36                LiveIndexAdapter liveIndexAdapter =
37                liveIndexAdapterArray.get(helper.getLayoutPosition());
38                if(liveIndexAdapter == null){
39                    liveIndexAdapter = new
40                    LiveIndexAdapter(activity, LiveIndexAdapter.FRAGMENT, R.layout.item_live_
41                    index_1);
42                    GridLayoutManager gridLayoutManager = new
43                    GridLayoutManager(activity, 4);
44                    recyclerView.setAdapter(liveIndexAdapter);
45                    recyclerView.setLayoutManager(gridLayoutManager);
46
47                    liveIndexAdapterArray.put(helper.getLayoutPosition(), liveIndexAdapter);
48                }
49                List<AdLiveItem> adLiveItemList = new LinkedList<>();
50                YunLiveIndex yunLiveIndex =

```

```
1      liveIndexItem.getYunLiveIndex();
2          if (canShowLiveIndexItem(yunLiveIndex.getXiche())) {
3
4      adLiveItemList.add(createAdLiveItem(AdLiveItem.TYPE_LIVE,
5      R.drawable.ic_wash_car, yunLiveIndex.getXiche().getLevel(), "洗车"));
6          }
7          if (canShowLiveIndexItem(yunLiveIndex.getChuanyi())) {
8
9      adLiveItemList.add(createAdLiveItem(AdLiveItem.TYPE_LIVE,
10     R.drawable.ic_wear, yunLiveIndex.getChuanyi().getLevel(), "穿衣"));
11         }
12         if (canShowLiveIndexItem(yunLiveIndex.getZiwaixian())) {
13
14     adLiveItemList.add(createAdLiveItem(AdLiveItem.TYPE_LIVE,
15     R.drawable.ic_sun_light, yunLiveIndex.getZiwaixian().getLevel(), "紫外
16     线"));
17         }
18         if (canShowLiveIndexItem(yunLiveIndex.getGanmao())) {
19
20     adLiveItemList.add(createAdLiveItem(AdLiveItem.TYPE_LIVE,
21     R.drawable.ic_cold, yunLiveIndex.getGanmao().getLevel(), "感冒"));
22         }
23         if (canShowLiveIndexItem(yunLiveIndex.getChenlian())) {
24
25     adLiveItemList.add(createAdLiveItem(AdLiveItem.TYPE_LIVE,
26     R.drawable.ic_sport, yunLiveIndex.getChenlian().getLevel(), "运动"));
27         }
28         if (Constant.SHOW_AD && !Constant.isEmulator && !isVip()) {
29         }
30         liveIndexAdapter.setNewData(adLiveItemList);
31     }
32     forceUpdate.put(item.getItemType(), false);
33 }
34
35     private AdLiveItem createAdLiveItem(int type, int iconId, String
36     content, String label) {
37         return createAdLiveItem(type, iconId, content, label, "");
38     }
39
40     private AdLiveItem createAdLiveItem(int type, int iconId, String
41     content, String label, String contentColor) {
42         AdLiveItem adLiveItem = new AdLiveItem();
43         adLiveItem.setType(type);
44         adLiveItem.setIconId(iconId);
45         adLiveItem.setContent(content);
46         adLiveItem.setLabel(label);
47         adLiveItem.setContentColor(StringUtils.isEmpty(contentColor)?
48     Color.parseColor("#333333"):Color.parseColor(contentColor));
49         return adLiveItem;
50     }
```

```
1
2     private boolean canShowLiveIndexItem(YunLiveIndexItem indexItem) {
3         if(indexItem == null) return false;
4
5     return !StringUtils.isEmpty(indexItem.getLevel()) && !StringUtils.isEmpty
6         (indexItem.getTips());
7     }
8
9     private void updateBanner1(BaseViewHolder helper) {
10         boolean force = forceUpdate.get(helper.getItemViewType());
11         if(!force) return;
12         FrameLayout frameLayout =
13     helper.getView(R.id.banner_container);
14         showNativeBanner1(frameLayout, helper.getLayoutPosition());
15         forceUpdate.put(helper.getItemViewType(), false);
16     }
17
18     private void updateBanner2(BaseViewHolder helper) {
19         boolean force = forceUpdate.get(helper.getItemViewType());
20         if(!force) return;
21         FrameLayout frameLayout =
22     helper.getView(R.id.banner_container);
23         showNativeBanner2(frameLayout, helper.getLayoutPosition());
24         forceUpdate.put(helper.getItemViewType(), false);
25     }
26
27     private void updateNative(BaseViewHolder helper) {
28         boolean force = forceUpdate.get(helper.getItemViewType());
29         if(!force) return;
30         FrameLayout frameLayout = helper.getView(R.id.item_root);
31         showNative(frameLayout, helper.getLayoutPosition());
32         forceUpdate.put(helper.getItemViewType(), false);
33     }
34
35     private void updateNews(BaseViewHolder helper) {
36         LinearLayout linearLayout = helper.getView(R.id.item_root);
37         if(newsView == null) {
38             newsView = new NewsView();
39             newsView.initView(linearLayout);
40             newsView.setViewPosition("首页");
41         }
42         newsView.initData(activity, newsTypeManager.getNewsItems());
43     }
44
45     @Override
46     public void logInfo(String msg) {
47
48     }
49
50     @Override
```

```
1      public void onResume() {
2          super.onResume();
3          resumeAd(TYPE_BANNER_1);
4          resumeAd(TYPE_BANNER_2);
5          resumeAd(TYPE_NATIVE);
6          resumeAd(TYPE_CONDITION);
7          if(newsTypeManager.isNeedUpdate()){
8              notifyDataSetChanged();
9              newsTypeManager.setNeedUpdate(false);
10         }
11     }
12
13     @Override
14     public void onPause() {
15         super.onPause();
16         pauseAd(TYPE_BANNER_1);
17         pauseAd(TYPE_BANNER_2);
18         pauseAd(TYPE_NATIVE);
19         pauseAd(TYPE_CONDITION);
20     }
21
22     private void resumeAd(int itemType){
23         forceUpdate.put(itemType, true);
24     }
25
26     private void pauseAd(int itemType){
27         forceUpdate.put(itemType, false);
28     }
29
30     private void setCurrentVoiceView(VoiceWaveView
31     voiceWaveView, ImageView voiceClose){
32         if(speakHolder == null){
33             speakHolder = new SpeakHolder();
34             speakHolder.waveView = voiceWaveView;
35             speakHolder.closeView = voiceClose;
36             setVoiceWaveData(voiceWaveView);
37             speakStatus(false);
38         }
39     }
40
41     public void speakStatus(boolean isSpeak){
42         if(speakHolder == null){
43             logInfo("speakHolder null");
44             return;
45         }
46         logInfo("speak status , isSpeak : "+isSpeak);
47         if(speakHolder.waveView != null){
48             if(isSpeak){
49                 speakHolder.waveView.setVisibility(View.VISIBLE);
50                 speakHolder.waveView.start();
```

```
1         }else{
2             speakHolder.waveView.stop();
3             speakHolder.waveView.setVisibility(View.INVISIBLE);
4         }
5     }
6     if(speakHolder.closeView != null){
7
8         speakHolder.closeView.setVisibility(isSpeak?View.INVISIBLE:View.VISIBLE
9     );
10    }
11 }
12
13 public void shakeAnimation() {
14     if(speakHolder == null)return;
15     ViewAnimator.animate(speakHolder.closeView)
16         .scale(1f, 1.05f, 1.15f, 1.05f, 1f)
17         .duration(1500)
18         .startDelay(1000)
19         .repeatCount(5)
20         .repeatMode(ViewAnimator.REVERSE)
21         .start();
22 }
23
24 private void setVoiceWaveData(VoiceWaveView voiceWaveView) {
25     voiceWaveView.setDuration(200);
26     voiceWaveView.addHeader(2);
27     voiceWaveView.addBody(8);
28     voiceWaveView.addBody(17);
29     voiceWaveView.addBody(38);
30     voiceWaveView.addBody(24);
31     voiceWaveView.addBody(8);
32     voiceWaveView.addBody(38);
33     voiceWaveView.addBody(14);
34     voiceWaveView.addBody(27);
35     voiceWaveView.addBody(8);
36     voiceWaveView.addFooter(2);
37     voiceWaveView.setWaveMode(WaveMode.LEFT_RIGHT);
38     voiceWaveView.setLineWidth(5f);
39     voiceWaveView.setLineSpace(2f);
40 }
41
42 private static class SpeakHolder{
43     public VoiceWaveView waveView;
44     public ImageView closeView;
45 }
46
47 protected boolean isVip() {
48     return userManager.isLogin() && userManager.isVip();
49 }
50
```

```
1         @Override
2         public void onAttachedToRecyclerView(RecyclerView recyclerView) {
3             super.onAttachedToRecyclerView(recyclerView);
4             FullSpanUtil.onAttachedToRecyclerView(recyclerView, this,
5 TYPE_NEWS);
6         }
7
8         @Override
9         public void onViewAttachedToWindow(BaseViewHolder holder) {
10             super.onViewAttachedToWindow(holder);
11             FullSpanUtil.onViewAttachedToWindow(holder, this, TYPE_NEWS);
12         }
13
14         private int getItemWidth() {
15             int screenW = getWidthHeight()[0] -
16 PixelUtils.dip2px(activity, 20);
17             return (screenW);
18         }
19
20         private int[] getWidthHeight() {
21             Resources resources = activity.getResources();
22             DisplayMetrics dm = resources.getDisplayMetrics();
23             int width = dm.widthPixels;
24             int height = dm.heightPixels;
25             return new int[] {width, height};
26         }
27     }
28     package com.frostowner.magic.weather.activity;
29
30     import android.Manifest;
31     import android.app.AlertDialog;
32     import android.content.ActivityNotFoundException;
33     import android.content.Intent;
34     import android.net.Uri;
35     import android.os.Build;
36     import com.tencent.bugly.beta.Beta;
37     import com.zhy.android.percent.support.PercentRelativeLayout;
38
39     import org.greenrobot.eventbus.Subscribe;
40     import org.greenrobot.eventbus.ThreadMode;
41
42     import java.util.Date;
43     import java.util.HashMap;
44     import java.util.List;
45     import java.util.Map;
46     import static com.frostowner.magic.weather.Constant.SHOW_AD;
47
48     public class HomeTabActivity extends AtInsertAdActivity {
49
50         private final String[] location_permissions = {
```



```
1         Manifest.permission.ACCESS_COARSE_LOCATION,
2         Manifest.permission.ACCESS_FINE_LOCATION
3     };
4
5     private final static String NAVIGATION_POWER_ANIMA =
6     "tt_weather_navigation_power_animation";
7
8     private final long BACK_PRESS_DELAYED = 2000;
9
10    private int[][] navigationRes = {
11        {R.drawable.ic_condition_2, R.drawable.ic_condition_1},
12    };
13
14    private String[] navigationNames = {
15        "实况天气"
16    };
17
18    @BindView(R.id.view_pager)
19    NoSwipeViewPager homeViewPager;
20
21    @BindView(R.id.radio_line)
22    View radioLine;
23
24    @BindView(R.id.group_alert)
25    View groupAlert;
26    @BindView(R.id.tv_task_alert_msg)
27    TextView tvTaskAlertMsg;
28
29    @BindView(R.id.bottom_menu)
30    PercentRelativeLayout bottomMenu;
31
32
33    @BindViews({R.id.tab_condition, R.id.tab_forecast, R.id.tab_power, R.id.ta
34    b_gift, R.id.tab_mine})
35    List<View> navigationViews;
36
37    private HomeAdapter homeAdapter;
38    private LocalStorage localStorage;
39    private long backPressTime;
40    private AdSourceRequestManager adSourceRequestManager;
41    private DeviceManager deviceManager;
42    private UserManager userManager;
43    private UserActionPatternManager userActionPatternManager;
44    private NewGiftDialog newGiftDialog;
45    private ViewAnimator animator;
46
47    private long stayTime;
48
49    @Override
50    protected void onCreate(@Nullable Bundle savedInstanceState) {
```

```
1         super.onCreate(savedInstanceState);
2         setContentView(R.layout.activity_home_tab);
3         StatusBarUtil.setTranslucentForImageViewInFragment(this,
4         0, null);
5         localStorage = BeanFactory.getBean(LocalStorage.class);
6         adSourceRequestManager =
7         BeanFactory.getBean(AdSourceRequestManager.class);
8         userManager = BeanFactory.getBean(UserManager.class);
9         deviceManager = BeanFactory.getBean(DeviceManager.class);
10        userActionPatternManager =
11        BeanFactory.getBean(UserActionPatternManager.class);
12        deviceManager.init();
13        if(deviceManager.isFirstInstall()){
14            deviceManager.setInstall();
15        }else{
16            if(deviceManager.getFirstInstallTime() == 0){
17                deviceManager.setInstall();
18            }
19            if(deviceManager.getInstalledDay() > 365*2){
20                deviceManager.setInstall();
21            }
22        }
23        boolean isNeedRequestAdFlag =
24        localStorage.get(NEED_REQUEST_AD_FLAG, true);
25        if(!isNeedRequestAdFlag&&!SHOW_AD){
26            SHOW_AD = true;
27            Intent intent = new Intent(this,
28            WidgetHandleService.class);
29            if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
30                startForegroundService(intent);
31            }else{
32                startService(intent);
33            }
34        }
35        initView();
36        handleAction();
37        checkLocationSet();
38        showApproval();
39        // showImportAlertDialog();
40        showAutoRemoveAdDialog();
41        showInsertAd();
42        // checkNotificationSet();
43        Constant.HOME_HAS_DEAD = false;
44        if(WeatherConfig.isNotificationSignAlertKey()){
45            SignNotificationJob.schedule();
46        }
47
48        if(ApkResources.isDebug(this)||Constant.TEST_SERVER_ENVIRONMENT){
49            String builder = "APP 环境: debug = " +
50            ApkResources.isDebug(this) +
```

```
1          "\n 基础服务器环境: " +
2      (BeanFactory.getBean(AdSourceRequestManager.class).isUseTestServer() ?
3      "测试服" : "正式服");
4      //          showLongNotice(builder);
5      }
6      //          relaxFlagTest();
7      Map<String,String> params = new HashMap<>();
8
9      params.put("pattern",userActionPatternManager.getUserLevelName());
10         TrackPlugin.track(this,USER_PATTERN,params);
11         checkUpgrade();
12     }
13
14     private void showApproval() {
15         boolean isClick = localStorage.get(APPROVAL_SHOW,false);
16         if(isClick){
17             return;
18         }
19         SimpleDateManager simpleDateManager =
20     BeanFactory.getBean(SimpleDateManager.class);
21         String day =
22     simpleDateManager.transformTime(System.currentTimeMillis(),"yyyy-MM-
23     dd");
24         boolean oneTime =
25     localStorage.get(APPROVAL_DAY_TIME+day+"_1",false);
26         boolean secondTime =
27     localStorage.get(APPROVAL_DAY_TIME+day+"_2",false);
28         if(!oneTime){
29             localStorage.put(APPROVAL_DAY_TIME+day+"_1",true);
30             return;
31         }
32         if(secondTime){
33             return;
34         }
35         showApprovalDialog("开发不易,好评鼓励我们一下吧~");
36         localStorage.put(APPROVAL_DAY_TIME+day+"_2",true);
37     }
38
39     private void showImportAlertDialog() {
40         if(SHOW_AD && !Constant.isEmulator
41     && !deviceManager.isShowImportantAlert()){
42             ImportantAlertDialog dialog = new
43     ImportantAlertDialog(this);
44             dialog.show();
45             deviceManager.setImportantShow();
46         }
47     }
48
49     private void showAutoRemoveAdDialog() {
50         if(!Constant.SHOW_AD&&!Constant.isEmulator)return;
```

```
1         boolean isShowApproval = localStorage.get(APPROVAL_SHOW, false);
2         if(!isShowApproval)return;
3         long time = localStorage.get(AUTO_REMOVE_AD_SHOW, 0L);
4         if(time != 0 && System.currentTimeMillis() - time <
5 1000*60*60*24*3){
6             return;
7         }
8         SimpleDateManager simpleDateManager =
9 BeanFactory.getBean(SimpleDateManager.class);
10        String day =
11 simpleDateManager.transformTime(System.currentTimeMillis(), "yyyy-MM-
12 dd");
13        boolean oneTime =
14 localStorage.get(AUTO_REMOVE_AD_DAY_TIME+day+"_1", false);
15        boolean secondTime =
16 localStorage.get(AUTO_REMOVE_AD_DAY_TIME+day+"_2", false);
17        if(!oneTime){
18            localStorage.put(AUTO_REMOVE_AD_DAY_TIME+day+"_1", true);
19            return;
20        }
21        if(secondTime){
22            return;
23        }
24        fire(new ActionEvent(START_SHOW_REMOVE_AD_DIALOG));
25        localStorage.put(AUTO_REMOVE_AD_DAY_TIME+day+"_2", true);
26    }
27
28    private void showFakeAdDialog(){
29        if(!Constant.SHOW_AD&&!Constant.isEmulator)return;
30        if(Build.VERSION.SDK_INT >= 23
31 && !checkPermissionGranted(location_permissions))return;
32        SimpleDateManager simpleDateManager =
33 BeanFactory.getBean(SimpleDateManager.class);
34        String time =
35 simpleDateManager.transformTime(System.currentTimeMillis(), "yyyy-MM-
36 dd");
37        boolean isFakeAdShow =
38 localStorage.get(FAKE_AD_WATCH_VIDEO+time, false);
39        if(isFakeAdShow)return;
40        addSubscription(SubscribeUtils.doOnUIThreadDelayed(new
41 Action0(){
42            @Override
43            public void call() {
44                fire(new
45 ActionEvent(EventType.START_SHOW_FAKE_AD_DIALOG));
46            }
47        }, 8000));
48        localStorage.put(FAKE_AD_WATCH_VIDEO+time, true);
49    }
50
```

```
1      @Override
2      protected void onNewIntent(Intent intent) {
3          super.onNewIntent(intent);
4          setIntent(intent);
5          handleAction();
6          logInfo("onNewIntent");
7          Constant.HOME_HAS_DEAD = false;
8      }
9
10     private void handleAction() {
11         if(getIntent() == null)return;
12         String action = getIntent().getStringExtra("action");
13         logInfo("home action : "+action);
14         if(StringUtils.isEmpty(action)) {
15             return;
16         }
17         if(action.equals(ACTION_CONDITION)) {
18             Intent intent = new Intent();
19             intent.setClass(this, ConditionDetailsActivity.class);
20             startActivity(intent);
21             TrackPlugin.track(this, NOTIFICATION_CLICK, null);
22         }
23         else if(action.equals(ACTION_15FORECAST)) {
24             Intent intent = new Intent();
25             intent.setClass(this, ForecastDetailsActivity.class);
26             startActivity(intent);
27         }
28         else if(action.equals(ACTION_HOME)) {
29             TrackPlugin.track(this, DESKTOP_WIDGET_CLICK, null);
30         }
31     }
32
33     private void initView() {
34         homeViewPager.setReMeasure(false);
35         homeAdapter = new HomeAdapter(getSupportFragmentManager());
36         homeViewPager.setAdapter(homeAdapter);
37         homeViewPager.setOffscreenPageLimit(homeAdapter.getCount());
38         homeViewPager.setScrollFlag(false);
39
40         bottomMenu.setVisibility(View.GONE);//SHOW_AD&&!Constant.isEmulator?View
41         w.VISIBLE:View.GONE);
42
43         radioLine.setVisibility(View.GONE);//SHOW_AD&&!Constant.isEmulator?View
44         .VISIBLE:View.GONE);
45     }
46
47     private void initNavigation() {
48         for(int i=0;i<navigationViews.size();i++) {
49             View view = navigationViews.get(i);
50             view.setOnClickListener(navigationClick);
```

```
1      }
2      setNavigationVisible(0);
3      View view = navigationViews.get(2);
4      TextView itemName = view.findViewById(R.id.item_name);
5      String key = NAVIGATION_POWER_ANIMA+dateFormat.format(new
6      Date());
7      boolean isShow = localStorage.get(key, false);
8      if(userManager.isLogin() && userManager.isVip()){
9          isShow = true;
10     }
11     if(isShow){
12         itemName.setVisibility(View.INVISIBLE);
13         return;
14     }
15     animator = ViewAnimator.animate(itemName)
16         .duration(900)
17         .startDelay(1000)
18         .scale(1f, 1.03f, 1.06f, 1.1f, 1.06f, 1.03f)
19         .repeatCount(ViewAnimator.INFINITE)
20         .repeatMode(ViewAnimator.REVERSE)
21         .start();
22     }
23
24     private void setNavigationVisible(int position){
25         for(int i=0;i<navigationViews.size();i++){
26             setNavigationItemVisible(i, i==position);
27         }
28     }
29
30     private void setNavigationItemVisible(int position, boolean
31     selected){
32         View view = navigationViews.get(position);
33         ImageView itemIcon = view.findViewById(R.id.item_icon);
34
35         itemIcon.setImageDrawable(getResources().getDrawable(selected?navigatio
36         nRes[position][1]:navigationRes[position][0]));
37         TextView itemName = view.findViewById(R.id.item_name);
38         if(position == 2){
39             itemName.setTextColor(0xffffffff);
40         }else{
41             itemName.setTextColor(selected?0xff209CF4:0xffA49F9F);
42         }
43         itemName.setText(navigationNames[position]);
44     }
45
46     private View.OnClickListener navigationClick = new
47     View.OnClickListener(){
48         @Override
49         public void onClick(View v) {
50             Map<String, String> params = new HashMap<>();
```

```
1         switch (v.getId()) {
2             case R.id.tab_condition:
3                 homeViewPager.setCurrentItem(0, false);
4                 setNavigationVisible(0);
5                 params.clear();
6                 params.put("click", "实况天气");
7
8             TrackPlugin.track(getContext(), HOME_NAVIGATION_CLICK, params);
9             // startShowTaskAlert();
10            break;
11            case R.id.tab_forecast:
12                homeViewPager.setCurrentItem(1, false);
13                setNavigationVisible(1);
14                params.clear();
15                params.put("click", "万年历");
16
17            TrackPlugin.track(getContext(), HOME_NAVIGATION_CLICK, params);
18            showInsertAd();
19            break;
20            case R.id.tab_power:
21                homeViewPager.setCurrentItem(2, false);
22                setNavigationVisible(2);
23                if(animator != null) {
24                    animator.cancel();
25                    animator = null;
26                    View view = navigationViews.get(2);
27
28                view.findViewById(R.id.item_name).setVisibility(View.INVISIBLE);
29                String key =
30                NAVIGATION_POWER_ANIMA+dateFormat.format(new Date());
31                localStorage.put(key, true);
32            }
33            params.clear();
34            params.put("click", "活动");
35
36            TrackPlugin.track(getContext(), HOME_NAVIGATION_CLICK, params);
37            break;
38            case R.id.tab_gift:
39                if(deviceManager.isFirstInstall()) {
40                } else {
41                    homeViewPager.setCurrentItem(3, false);
42                    setNavigationVisible(3);
43                }
44                params.clear();
45                params.put("click", "福利");
46
47            TrackPlugin.track(getContext(), HOME_NAVIGATION_CLICK, params);
48            break;
49            case R.id.tab_mine:
50                homeViewPager.setCurrentItem(4, false);
```

```
1         setNavigationVisible(4);
2         params.clear();
3         params.put("click", "我的");
4
5         TrackPlugin.track(getContext(), HOME_NAVIGATION_CLICK, params);
6         params.clear();
7         params.put("position", "底部导航");
8
9         TrackPlugin.track(getContext(), MINE_NAVIGATION_CLICK, params);
10        break;
11    }
12    }
13    };
14
15    private void checkLocationSet() {
16        if((Build.VERSION.SDK_INT < 23 ||
17        checkPermissionGranted(location_permissions)) &&
18        LocationUtils.isLocServiceEnable(this))return;
19        boolean locationAlert =
20        isProperty("tt_weather_location_guide_alert");
21        if(locationAlert)return;
22        LocationGuideDialog dialog = new LocationGuideDialog(this, new
23        View.OnClickListener() {
24            @Override
25            public void onClick(View v) {
26                addPermission(location_permissions, 102, new
27        PermissionCallback() {
28            @Override
29            public void result(boolean isGranted, List<String>
30        permission) {
31                if(!isGranted){
32                    showNotice("申请失败");
33                }
34            }
35        });
36        checkPermissions();
37    }
38    });
39    dialog.show();
40    setProperty("tt_weather_location_guide_alert", "true");
41    }
42
43    private void checkNotificationSet() {
44        if(Build.VERSION.SDK_INT < Build.VERSION_CODES.KITKAT)return;
45        if(NotificationUtils.isNotificationEnable(this))return;
46        boolean isShow =
47        localStorage.get("tt_weather_notification_set_alert", false);
48        if(isShow)return;
49        AlertDialog.Builder builder = new AlertDialog.Builder(this);
50        builder.setTitle("温馨提示");
```



```
1         builder.setMessage("检测到您没有开启通知服务,可能会影响您的天气  
2 功能使用,是否跳转至设置页面");  
3         builder.setNegativeButton("取消",null);  
4         builder.setPositiveButton("立即前往", (dialog, which) -> {  
5             if (android.os.Build.VERSION.SDK_INT >=  
6 Build.VERSION_CODES.LOLLIPOP) {  
7                 Intent intent = new Intent();  
8  
9                 intent.setAction("android.settings.APP_NOTIFICATION_SETTINGS");  
10                    intent.putExtra("app_package", getPackageName());  
11                    intent.putExtra("app_uid", getApplicationInfo().uid);  
12                    startActivity(intent);  
13                } else if (android.os.Build.VERSION.SDK_INT ==  
14 Build.VERSION_CODES.KITKAT) {  
15                    Intent intent = new Intent();  
16  
17                    intent.setAction(Settings.ACTION_APPLICATION_DETAILS_SETTINGS);  
18                    intent.addCategory(Intent.CATEGORY_DEFAULT);  
19                    intent.setData(Uri.parse("package:" +  
20 getPackageName()));  
21                    startActivity(intent);  
22                }  
23            });  
24            builder.show();  
25            localStorage.put("tt_weather_notification_set_alert",true);  
26        }  
27  
28        private void showApprovalDialog(String content){  
29            ApprovalDialog approvalDialog = new ApprovalDialog(this,  
30 content, new View.OnClickListener() {  
31                @Override  
32                public void onClick(View v) {  
33                    try {  
34                        Uri uri = Uri.parse("market://details?id=" +  
35 HomeTabActivity.this.getApplication().getPackageName());  
36                        Intent intent1 = new Intent(Intent.ACTION_VIEW,  
37 uri);  
38                        intent1.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);  
39                        startActivity(intent1);  
40                    } catch (ActivityNotFoundException e) {  
41                        showNotice("无法打开应用市场");  
42                    }  
43                    localStorage.put(APPROVAL_SHOW,true);  
44                }  
45            }, new View.OnClickListener() {  
46                @Override  
47                public void onClick(View v) {  
48  
49 BeanFactory.getBean(PluginManager.class).showFeedBack(getContext(),user  
50 Manager.getSoc());
```

```
1         }
2     }, null);
3     approvalDialog.show();
4 }
5
6     @Subscribe(threadMode = ThreadMode.MAIN)
7     public void onSceneChangeEvent(ActionEvent event) {
8         if(event.getType() == CHANGE_SECEN_GAME) {
9             String position = event.getAttrStr("position");
10            Map<String, String> params = null;
11            if(!StringUtils.isEmpty(position)) {
12                params = new HashMap<>();
13                params.put("position", position);
14            }
15
16            TrackPlugin.track(getContext(), GAME_NAVIGATION_CLICK, params);
17            startActivity(new Intent(getContext(), GameActivity.class));
18        }
19        else if(event.getType() == CHANGE_SECEN_FORECAST) {
20            String time = event.getAttrStr("time");
21        else{
22            Intent intent = new
23            Intent(this, ForecastDetailsActivity.class);
24            intent.putExtra("time", time);
25            startActivity(intent);
26        }
27        else if(event.getType() == CHANGE_SECEN_CONDITION) {
28            homeViewPager.setCurrentItem(0, false);
29            setNavigationVisible(0);
30        }
31        else if(event.getType() == CHANGE_SECEN_POWER) {
32            homeViewPager.setCurrentItem(2, false);
33            setNavigationVisible(2);
34        }
35        else if(event.getType() == CHECK_CALENDAR_PERMISSION) {
36            final boolean isChecked = event.getAttrBoolean("checked");
37            addPermission(CALENDAR_PERMISSION, 102, (isGranted,
38 permissions) -> {
39                if(isGranted) {
40
41            TrackPlugin.track(getContext(), GIFT_PERMISSION_GRANTED, null);
42                }else{
43
44            TrackPlugin.track(getContext(), GIFT_PERMISSION_REFUSE, null);
45                }
46
47            ActionEvent permissionEvent = new
48            ActionEvent(GRANTED_CALENDAR_PERMISSION);
49            permissionEvent.setAttr("checked", isChecked);
50            permissionEvent.setAttr("isGranted", (isGranted));
51            fire(permissionEvent);
```

```
1         });
2         checkPermissions();
3     }
4     else if(event.getType() == PLATFORM_LOGIN_SUCCESS) {
5         if(newGiftDialog != null) {
6             newGiftDialog.dismiss();
7             newGiftDialog = null;
8         }
9         int position = event.getAttrInt("position");
10        if(position == 1) {
11            homeViewPager.setCurrentItem(3, false);
12            setNavigationVisible(3);
13        }
14        if(userManager.isLogin() && userManager.isVip()) {
15            fire(new ActionEvent(VIP_STATUS_CHANGE));
16        }
17    }
18    else if(event.getType() == SHOW_LIVE_INDEX_DETAILS) {
19        int id = event.getAttrInt("position");
20        int type = event.getAttrInt("type");
21        if(type != LiveIndexAdapter.FRAGMENT) return;
22        CityManager cityManager =
23        BeanFactory.getBean(CityManager.class);
24        if(cityManager.getCurrentCityWeather() == null) return;
25        LiveIndexDetailsDialog dialog = new
26        LiveIndexDetailsDialog(this, cityManager.getCurrentCityWeather(), id);
27        dialog.show();
28    }
29    else if(event.getType() == START_SHOW_APPROVAL_DIALOG) {
30        String content = event.getAttrStr("content");
31        showApprovalDialog(content);
32    }
33    else if(event.getType() == EventType.WEATHER_REQUEST_SUCCESS) {
34        showFakeAdDialog();
35    }
36    else if(event.getType() == EventType.CHECK_UPGRADE) {
37        checkUpgrade();
38    }
39    }
40
41    @Override
42    protected void onResume() {
43        super.onResume();
44        stayTime = System.currentTimeMillis();
45    }
46
47    @Override
48    protected void onPause() {
49        super.onPause();
50        long time = System.currentTimeMillis() - stayTime;
```

```
1         if(stayTime == 0)time = 0;
2         if(time<= 1000)return;
3         Map<String,String> params = new HashMap<>();
4         params.put("params",
5 SupportUtils.transformTime((int)time/1000));
6         TrackPlugin.track(this,APP_STAY_TIME,params);
7     }
8
9     private boolean isChecked = false;
10
11     private void checkUpgrade() {
12         if(isChecked)return;
13         addSubscription(SubscribeUtils.doOnUIThreadDelayed(new
14 Action0() {
15             @Override
16             public void call() {
17                 logInfo("检测自动更新");
18                 Beta.checkUpgrade(true, false);
19                 isChecked = false;
20             }
21         }, 5000));
22         isChecked = true;
23     }
24
25     @Override
26     protected void onDestroy() {
27         super.onDestroy();
28         Constant.HOME_HAS_DEAD = true;
29         BeanFactory.getBean(WeatherItemExposure.class).release();
30     }
31
32     @Override
33     public void onBackPressed() {
34         if(Constant.CONDITION_NEWS_STICKY) {
35             fire(new
36 ActionEvent(EventType.CONDITION_NEWS_STICKY_RELEASE));
37             Constant.CONDITION_NEWS_STICKY = false;
38             return;
39         }
40         if(System.currentTimeMillis() - backPressTime >
41 BACK_PRESS_DELAYED) {
42             showNotice("请再按一次确认退出");
43             backPressTime = System.currentTimeMillis();
44         }else{
45             super.onBackPressed();
46         }
47     }
```