```
package com.baoneng.bnmall.ui;
import android.content.Intent;
import android.os.Bundle;
import com.baoneng.bnmall.BuildConfig;
import com.baoneng.bnmall.utils.Store;
/**
 * 开始显示动画
 */
public class SplashActivity extends BaseActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        int appGuideVersion = Store.getAppGuideVersionCode(this);
        if (appGuideVersion < BuildConfig.VERSION_CODE) {
            startActivity(new Intent(this, AppGuideActivity.class));
        } else {
            startActivity(new Intent(this, MainActivity.class));
        }
        finish();
    }
    @Override
    public String getStatusBarStyle() {
        return STYLE_STATUS_BAR_OTHER;
    }
}
package com.baoneng.bnmall.ui;
import android.animation.AnimatorSet;
import android.animation.ObjectAnimator;
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.os.Handler;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentTransaction;
import android.support.v4.content.LocalBroadcastManager;
import android.text.TextUtils;
import android.view.View;
import android.view.animation.LinearInterpolator;
import android.widget.Button;
import android.widget.LinearLayout;
```

```
import android.widget.TextView;
import android.widget.Toast;
import com.baoneng.bnmall.ExitApplication;
import com.baoneng.bnmall.R;
import com.baoneng.bnmall.common.Constants;
import com.baoneng.bnmall.contract.mainscreen.MainActContract;
import com.baoneng.bnmall.model.shoppingcar.AddCartItemReq;
import com.baoneng.bnmall.presenter.mainscreen.MainActPresenter;
import com.baoneng.bnmall.ui.member.MemberCenterFragment;
import com.baoneng.bnmall.ui.mainscreen.homepage.HomePageFragment;
import com.baoneng.bnmall.ui.mainscreen.homepage.LocationHelper;
import com.baoneng.bnmall.ui.shelf.ShelfFragment;
import com.baoneng.bnmall.ui.shoppingcar.ShoppingCarFrag;
import com.baoneng.bnmall.utils.CartNumChangeNoticUtils;
import com.baoneng.bnmall.utils.ToastUtil;
import com.baoneng.bnmall.utils.anim.OnParabolaAnimation;
import com.baoneng.bnmall.utils.anim.ParabolaAnimationHelper;
import com.baoneng.bnmall.widget.CartPointView;
import com.baoneng.bnmall.widget.UpdateHelper;
import java.util.ArrayList;
import java.util.List;
import butterknife.BindView;
import butterknife.OnClick;
public class MainActivity extends BaseActivity<MainActContract.Presenter> implements
OnParabolaAnimation, MainActContract.View {
    // 需要显示的fragment数量， 暂时与TAB_NUM数量相当， 如一个tab有可能会显示多个fragment
时则不等
    private final static int FRAGMENT_COUNT = 4;
    private final static String TAG_FRAGMENT = "tab_fragment";
    public static final String EXTRA_ADD_CART_GOODS_SKU_NO = "EXTRA_ADD_CART_GOODS_SKU_NO";
    // 添加到购物车商品数量，可不填，默认为1
    public static final String EXTRA_ADD_CART_GOODS_NUM = "EXTRA_ADD_CART_GOODS_NUM";
    // 跳转到MainActivity，默认显示的Tab
    public static final String EXTRA_SELECT_TAB_INDEX = "EXTRA_SELECT_TAB_INDEX";
    // 跳转到MainActivity，跳转到分类页面，参数classNo
    public static final String EXTRA_CLASSIFY_CLASS_NO = "EXTRA_CLASSIFY_CLASS_NO";
    private final List<Fragment> mFragments = new ArrayList<>();
    private FragmentManager mFragmentManager;
    private LinearLayout mRadioGroup;
    private final int[] mRadioBtnIds = {R.id.radio_home, R.id.radio_manage_finance,
R.id.radio_cart, R.id.radio_me};
    @BindView(R.id.radio_cart)
    View cartView;
    @BindView(R.id.cartGoodsNum)
```

```
    TextView mCartGoodsNum;
    @BindView(R.id.cartview)
    CartPointView cartview;
    private int positionWithRedPoint = -1;
    private Button radioWithRedPoint;
    public static int mCartTotalNum;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // 开始定位
        LocationHelper.getInstance().startLocation(getActivity());
        mPresenter = new MainActPresenter(this);
        mFragmentManager = getSupportFragmentManager();
        initView();
        setDefaultTabShown(0);
        IntentFilter intentFilter = new IntentFilter();
//     intentFilter.addAction(CartNumChangeNoticUtils.CART_NUM_CHANGE_ACTION);
        intentFilter.addAction(Constants.ACTION_ADD_GOODS_TO_CART);
    LocalBroadcastManager.getInstance(this).registerReceiver(mBroadcastReceiver,
intentFilter);
//        mPresenter.loadData();
         // 检查更新
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
     UpdateHelper.getInstance().autoCheckUpdate(MainActivity.this);
            }
        }, 1000);
    }
    private void initView() {
        mRadioGroup = findViewById(R.id.bottom_bar);
        mFragments.clear();
        Fragment fragment1 = mFragmentManager.findFragmentByTag(TAG_FRAGMENT + "0");
        if (fragment1 == null) {
            fragment1 = new HomePageFragment();
        }
        mFragments.add(fragment1);
        Fragment fragment2 = mFragmentManager.findFragmentByTag(TAG_FRAGMENT + "1");
        if (fragment2 == null) {
            fragment2 = new ShelfFragment();
        }
        mFragments.add(fragment2);
        Fragment fragment3 = mFragmentManager.findFragmentByTag(TAG_FRAGMENT + "2");
```

```
        if (fragment3 == null) {
            fragment3 = new ShoppingCarFrag();
        }
        mFragments.add(fragment3);
        Fragment fragment4 = mFragmentManager.findFragmentByTag(TAG_FRAGMENT + "3");
        if (fragment4 == null) {
            fragment4 = new MemberCenterFragment();
        }
        mFragments.add(fragment4);
    }
    private void setDefaultTabShown(int defaultPosition) {
        if (defaultPosition > mRadioBtnIds.length || defaultPosition < 0) {
            defaultPosition = 0;
        }
        selectTab(defaultPosition);
    }
    private void showFragment(int position) {
        if (position < mFragments.size()) {
            hideAllFragment();
            FragmentTransaction transaction = mFragmentManager.beginTransaction();
            Fragment fragment = mFragments.get(position);
            if (mNeedShowFirstClass != null
                    && fragment instanceof ShelfFragment) {
                Bundle args = new Bundle();
args.putString(Constants.EXTRA_SHELF_GOODS_CLASS,mNeedShowFirstClass);
                fragment.setArguments(args);
                mNeedShowFirstClass = null;
            }
            if (fragment.isAdded()) {
                transaction.show(fragment);
            } else {
                transaction.add(R.id.tab_container, fragment, TAG_FRAGMENT +
position).show(fragment);
            }
            transaction.commitAllowingStateLoss();
        }
    }
    //需显示的分类
    private String mNeedShowFirstClass;
    /**
     * 显示货架号并带上一级分类号
     */
    public void showGoodsWithClass(String classNo) {
        ShelfFragment fragment2 = (ShelfFragment)
```

```
mFragmentManager.findFragmentByTag(TAG_FRAGMENT + 1);
        if (!TextUtils.isEmpty(classNo)){
            if (fragment2 != null) {
                Bundle args = new Bundle();
     args.putString(Constants.EXTRA_SHELF_GOODS_CLASS,mNeedShowFirstClass);
                fragment2.setArguments(args);
                fragment2.setCurrentFirstClass(classNo);
            }else{
                mNeedShowFirstClass = classNo;
            }
        }
        selectTab(1);
    }
    public void selectTab(int position) {
        View childView = mRadioGroup.getChildAt(position);
        if (childView instanceof Button) {
//            ((Button) childView).setChecked(true);
            childView.performClick();
        } else {
            if (positionWithRedPoint == -1) {
                positionWithRedPoint = position;
                radioWithRedPoint = this.findViewById(mRadioBtnIds[positionWithRedPoint]);
            }
            radioWithRedPoint.performClick();
        }
    }
    private void hideAllFragment() {
        List<Fragment> fragments = mFragmentManager.getFragments();
        FragmentTransaction transaction = mFragmentManager.beginTransaction();
        if (fragments != null && fragments.size() > 0) {
            // 如果 FragmentManager 池中缓存的 fragment 数量大于需要显示的 fragment 总数，说明有
重复，
            // 则将 FragmentManager 池中缓存的 fragment 全部清空
            if (fragments.size() > FRAGMENT_COUNT) {
                for (Fragment f : fragments) {
                    if (f != null) {
                        transaction.remove(f);
                    }
                }
            } else { // 否则执行正常 hide 操作
                for (Fragment f : fragments) {
                    if (f != null) {
                        transaction.hide(f);
                    }
```

```
            }
        }
        transaction.commitAllowingStateLoss();
    }
}
@OnClick({R.id.radio_home, R.id.radio_manage_finance, R.id.radio_cart, R.id.radio_me})
public void onClick(View view) {
    int checkId = view.getId();
    for (int i = 0; i < mRadioBtnIds.length; i++) {
        if (mRadioBtnIds[i] == checkId) {
            showFragment(i);
            View childView = mRadioGroup.getChildAt(i);
            if (childView instanceof Button) {
                childView.setSelected(true);
                startBottomBarAnimation(childView);
            } else {
                if (positionWithRedPoint == -1) {
                    positionWithRedPoint = i;
                    radioWithRedPoint = findViewById(mRadioBtnIds[positionWithRedPoint]);
                }
                radioWithRedPoint.setSelected(true);
                startBottomBarAnimation(mCartGoodsNum);
                startBottomBarAnimation(radioWithRedPoint);
            }
        } else {
            View childView = mRadioGroup.getChildAt(i);
            if (childView instanceof Button) {
                childView.setSelected(false);
            } else {
                if (positionWithRedPoint == -1) {
                    positionWithRedPoint = i;
                    radioWithRedPoint = findViewById(mRadioBtnIds[positionWithRedPoint]);
                }
                radioWithRedPoint.setSelected(false);
            }
        }
    }
}
private Toast mBackToast;
private long mExitTime = 0L;
@Override
public void onBackPressed() {
    if (!getSupportFragmentManager().popBackStackImmediate()) {
        back();
```

```
        }
    }
    /**
     * 点击后退按钮时处理
     */
    public void back() {
        if ((System.currentTimeMillis() - mExitTime) > 3000) {
            mBackToast = ToastUtil.showShortToast("再按一次退出应用");
            mExitTime = System.currentTimeMillis();
        } else {
            if (mBackToast != null) {
                mBackToast.cancel();
            }
            finish();
            ExitApplication.getInstance().appExit(getApplicationContext());
        }
    }
    private void startBottomBarAnimation(View obj) {
        AnimatorSet set = new AnimatorSet();
        ObjectAnimator scaleXAnim = ObjectAnimator.ofFloat(obj, "scaleX", 1f, 1.2f, 1f);
//        scaleXAnim.setRepeatCount(ValueAnimator.REVERSE);
        scaleXAnim.setDuration(500);
        ObjectAnimator scaleYAnim = ObjectAnimator.ofFloat(obj, "scaleY", 1f, 1.2f, 1f);
//        scaleYAnim.setRepeatCount(ValueAnimator.REVERSE);
        scaleYAnim.setDuration(500);
        set.playTogether(scaleXAnim, scaleYAnim);
        set.setInterpolator(new LinearInterpolator());
        set.start();
    }
    @Override
    public void onParabolaAnimation(View startView) {
        ParabolaAnimationHelper.startAnimation(this, startView, mCartGoodsNum);
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        LocationHelper.getInstance().destroyLocation();
LocalBroadcastManager.getInstance(this).unregisterReceiver(mBroadcastReceiver);
    }
    @Override
    protected void onNewIntent(Intent intent) {
        super.onNewIntent(intent);
        if (intent != null) {
            int index = intent.getIntExtra(EXTRA_SELECT_TAB_INDEX, -1);
```

```
            if (index == 1) {
                String classNo = intent.getStringExtra(EXTRA_CLASSIFY_CLASS_NO);
                showGoodsWithClass(classNo);
            } else if (index != -1) {
                selectTab(index);
            }
        }
    }
    public static void switchTab(Activity activity, int tabIndex) {
        if (activity != null) {
            Intent intent = new Intent(activity, MainActivity.class);
            intent.putExtra(MainActivity.EXTRA_SELECT_TAB_INDEX, tabIndex);
            activity.startActivity(intent);
        }
    }
    @Override
    public void setCartNum(int num) {
//        setCartGoodsTotalNum(num);
        CartNumChangeNoticUtils.storeTotalNum(num);
        cartview.setTotalNum(num);
    }
    private BroadcastReceiver mBroadcastReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            if (intent != null) {
                String action = intent.getAction();
                switch (action == null ? "" : action) {
                    case Constants.ACTION_ADD_GOODS_TO_CART: // 添加商品到购物车
                        String skuNo =
intent.getStringExtra(MainActivity.EXTRA_ADD_CART_GOODS_SKU_NO);
                        int num = intent.getIntExtra(MainActivity.EXTRA_ADD_CART_GOODS_NUM,
1);
                        AddCartItemReq addCartItemReq = new AddCartItemReq();
                        addCartItemReq.setNum(num);
                        addCartItemReq.setSalesType("0");
                        addCartItemReq.setSkuNo(skuNo);
                        mPresenter.addCartItem(addCartItemReq, false);
                        break;
                    default:
                        break;
                }
            }
        }
    };
```

```
}
package com.baoneng.bnmall.ui.mainscreen.homepage;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.content.LocalBroadcastManager;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.text.TextUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewStub;
import android.webkit.URLUtil;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.PopupWindow;
import android.widget.RelativeLayout;
import android.widget.TextView;
import com.baoneng.bnmall.R;
import com.baoneng.bnmall.common.Constants;
import com.baoneng.bnmall.contract.cart.BaseScanContract;
import com.baoneng.bnmall.contract.mainscreen.HomePageContract;
import com.baoneng.bnmall.model.mainscreen.LocationModel;
import com.baoneng.bnmall.model.mainscreen.RequestQueryStoreAndSwitch;
import com.baoneng.bnmall.model.mainscreen.ResponseHomePageStorageQry;
import com.baoneng.bnmall.model.mainscreen.ResponseQueryStoreAndSwitch;
import com.baoneng.bnmall.network.BNUrl;
import com.baoneng.bnmall.presenter.cart.BaseScanPresenter;
import com.baoneng.bnmall.presenter.mainscreen.HomePagePresenter;
import com.baoneng.bnmall.recyclerview.CommonRecyclerAdapter;
import com.baoneng.bnmall.recyclerview.MultiRecyclerAdapter;
import com.baoneng.bnmall.ui.BaseFragment;
import com.baoneng.bnmall.ui.QrCodeScannerActivity;
import com.baoneng.bnmall.ui.WebViewActivity;
import com.baoneng.bnmall.ui.goods.ScanCodePurchaseActivity;
import com.baoneng.bnmall.ui.mainscreen.homepage.delegate.ActivityGoodsAdapterDelegate;
```

```
import com.baoneng.bnmall.ui.mainscreen.homepage.delegate.AdvertFlexAdapterDelegate;
import com.baoneng.bnmall.ui.mainscreen.homepage.delegate.AdvertTwoRowAdapterDelegate;
import com.baoneng.bnmall.ui.mainscreen.homepage.delegate.BannerAdapterDelegate;
import com.baoneng.bnmall.ui.mainscreen.homepage.delegate.GoodsAdapterDelegate;
import com.baoneng.bnmall.ui.mainscreen.homepage.delegate.IconBannerAdapterDelegate;
import com.baoneng.bnmall.ui.mainscreen.homepage.delegate.PromGoodsAdapterDelegate;
import com.baoneng.bnmall.ui.mainscreen.homepage.delegate.ScrollableGoodsAdapterDelegate;
import com.baoneng.bnmall.ui.mainscreen.homepage.delegate.SearchBarAdapterDelegate;
import com.baoneng.bnmall.ui.mainscreen.homepage.delegate.SectionTitleAdapterDelegate;
import com.baoneng.bnmall.ui.mainscreen.homepage.model.HomePageModel;
import com.baoneng.bnmall.ui.mainscreen.homepage.viewholder.PopStoreSelectItemViewHolder;
import com.baoneng.bnmall.ui.mainscreen.homepage.viewholder.StoreSelectItemViewHolder;
import com.baoneng.bnmall.ui.search.SearchActivity;
import com.baoneng.bnmall.utils.AndroidUtils;
import com.baoneng.bnmall.utils.Log;
import com.baoneng.bnmall.utils.Store;
import com.baoneng.bnmall.utils.ToastUtil;
import com.baoneng.bnmall.widget.CommonExceptLayout;
import com.baoneng.bnmall.widget.dialog.NormalDialog;
import com.baoneng.zxing.integration.android.IntentIntegrator;
import com.baoneng.zxing.integration.android.IntentResult;
import com.orhanobut.logger.Logger;
import com.squareup.picasso.Picasso;
import java.util.List;
import butterknife.BindView;
import butterknife.OnClick;
import static com.baoneng.bnmall.ui.BaseActivity.STYLE_STATUS_BAR_WHITE;
import static com.baoneng.zxing.integration.android.IntentIntegrator.REQUEST_CODE;
public class HomePageFragment extends BaseFragment<HomePageContract.Presenter> implements
HomePageContract.View, BaseScanContract.View {
    public static final String EXTRA_STORE_INFO = "EXTRA_STORE_INFO";
    @BindView(R.id.llTitleParent)
    RelativeLayout llTitleParent;
    @BindView(R.id.tvAddress)
    TextView tvAddress;
    @BindView(R.id.ivLogo)
    ImageView ivLogo;
    @BindView(R.id.ivTriangle)
    ImageView ivTriangle;
    @BindView(R.id.recyclerView)
    RecyclerView recyclerView;
    @BindView(R.id.vsOutDelivery)
    ViewStub vsOutDelivery;
    @BindView(R.id.llHomePageContent)
```

```
LinearLayout llHomePageContent;
@BindView(R.id.searchTitleLayout)
View searchTitleLayout;
private LinearLayout llOutDelivery;
private NormalDialog permissionDialog;
private List<ResponseHomePageStorageQry.DatasBean> homePageDataList;
/**
 * 配送范围内的门店列表
 */
private List<ResponseQueryStoreAndSwitch.StoreBean> inDeliveryStoreList;
/**
 * 当前首页展示门店对象
 */
private ResponseQueryStoreAndSwitch.StoreBean currentStoreModel;
/**
 * 门店选择 pop 框对象
 */
private PopupWindow popupWindow;
private BaseScanPresenter mScanPresenter;
private String storeNo;
private String storeType;
private final int NO_SEARCH_LAYOUT_POSITION = -1;
// 搜索布局所在 RecyclerView 中的位置
private int searchLayoutPosition = NO_SEARCH_LAYOUT_POSITION;
@OnClick({R.id.ivLogo, R.id.ivTriangle})
public void logoClick() {
    showPopupWindow();
}
@OnClick(R.id.llLocation)
public void llLocationClick() {
    // 选择地址
    Intent intent = new Intent(mContext, WebViewActivity.class);
    intent.putExtra(WebViewActivity.EXTRA_URL, BNUrl.URL_ADDRESS_SELECT);
    mContext.startActivity(intent);
}
@OnClick(R.id.ivQRCode)
public void qrCodeClick() {
    startActivityStateForResult(new Intent(mContext, QrCodeScannerActivity.class)
            .putExtra("title", "扫描货架二维码")
            .putExtra("content", getString(R.string.shelf_content)), REQUEST_CODE);
}
/**
 * 吸顶搜索处理
 */
```

```
    @OnClick(R.id.myEditText)
    public void searchLayoutClick() {
        mContext.startActivity(new Intent(mContext, SearchActivity.class));
    }
    @Override
    public int getContentViewId() {
        return R.layout.fragment_home_page;
    }
    @Override
    protected void initUI() {
        super.initUI();
        if (mSwipeRefresh != null) {
            mSwipeRefresh.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
                @Override
                public void onRefresh() {
                    if (currentStoreModel == null) {
                        handleLocationResult();
                    } else {
                        mPresenter.switchStore(currentStoreModel.getStoreNo(),
currentStoreModel.getStoreType());
                    }
                }
            });
        }
    }
    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        mPresenter = new HomePagePresenter(this);
        mScanPresenter = new BaseScanPresenter(this);
        mExceptLayout = CommonExceptLayout.on(llHomePageContent);
        mExceptLayout.setOnRefreshListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (LocationHelper.locationState == LocationHelper.LOCATION_STATE_NO_NET) {
                    reLocation();
                } else if (currentStoreModel == null) {
                    handleLocationResult();
                } else {
                    mPresenter.switchStore(currentStoreModel.getStoreNo(),
currentStoreModel.getStoreType());
                }
            }
        });
```

```
        recyclerView.setLayoutManager(new LinearLayoutManager(mBaseActivity) {
            @Override
            public boolean canScrollHorizontally() {
                return false;
            }
        });
        recyclerView.addOnScrollListener(new RecyclerView.OnScrollListener() {
            @Override
            public void onScrolled(RecyclerView recyclerView, int dx, int dy) {
                if (recyclerView != null && recyclerView.getChildCount() > searchLayoutPosition
&& searchLayoutPosition != NO_SEARCH_LAYOUT_POSITION) {
                    LinearLayoutManager      layoutManager      =      (LinearLayoutManager)
recyclerView.getLayoutManager();
                    //屏幕中第一个可见子项的 position
                    int                      firstVisibleItemPosition                      =
layoutManager.findFirstVisibleItemPosition();
                    if     (firstVisibleItemPosition     !=     searchLayoutPosition     &&
firstVisibleItemPosition > searchLayoutPosition - 1) {
                        searchTitleLayout.setVisibility(View.VISIBLE);
                    } else {
                        searchTitleLayout.setVisibility(View.GONE);
                    }
                }
                if (!mSwipeRefresh.isRefreshing()) {
                    setSwipeRefreshEnabled();
                }
            }
            @Override
            public void onScrollStateChanged(RecyclerView recyclerView, int newState) {
                super.onScrollStateChanged(recyclerView, newState);
            }
        });
        handleLocationResult();
    }
    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        //监听定位
        IntentFilter filter = new IntentFilter();
        filter.addAction(Constants.ACTION_LOCATION_STATE_CHANGED);
        filter.addAction(Constants.ACTION_SELECT_ONE_STORE);
        filter.addAction(Constants.ACTION_SWITCH_ONE_STORE);
        LocalBroadcastManager.getInstance(context).registerReceiver(mReceiver, filter);
    }
```

```
    @Override
    public void onDetach() {
        super.onDetach();
        setCurrentStoreModel(null);
        LocalBroadcastManager.getInstance(mBaseActivity).unregisterReceiver(mReceiver);
    }
    @Override
    public void onResume() {
        super.onResume();
        if (currentStoreModel == null && permissionDialog != null
&& !permissionDialog.isShowing()) {
            reLocation();
        }
    }
    @Override
    public void onHiddenChanged(boolean hidden) {
        super.onHiddenChanged(hidden);
        if (!hidden) {
            reLocation();
            setSwipeRefreshEnabled();
        }
    }
    /**
     * 设置下拉刷新是否可用，解决 SwipeRefreshLayout 和 recyclerView 下滑的冲突
     */
    private void setSwipeRefreshEnabled() {
        LinearLayoutManager layoutManager = (LinearLayoutManager)
recyclerView.getLayoutManager();
        //屏幕中第一个可见子项的 position
        int firstVisibleItemPosition = layoutManager.findFirstVisibleItemPosition();
        int topRowVerticalPosition =
                (recyclerView == null || recyclerView.getChildCount() == 0) ? 0 :
recyclerView.getChildAt(0).getTop();
        if (firstVisibleItemPosition == 0 && topRowVerticalPosition >= 0) {
            mSwipeRefresh.setEnabled(true);
        } else {
            mSwipeRefresh.setRefreshing(false);
            mSwipeRefresh.setEnabled(false);
        }
    }
    @Override
    public void showLoadingProgressBar(boolean show, Object type) {
        if (recyclerView.getVisibility() == View.VISIBLE &&
mExceptLayout.getExceptionLayout().getVisibility() == View.GONE) {
```

```
            showLoadingByActivity(false, type);
            mSwipeRefresh.setRefreshing(show);
        } else {
            mSwipeRefresh.setRefreshing(false);
            showLoadingByActivity(show, type);
        }
    }
    /**
     * 重新定位
     */
    private void reLocation() {
        if (LocationHelper.locationState == LocationHelper.LOCATION_STATE_FAILED
                || LocationHelper.locationState == LocationHelper.LOCATION_STATE_NO_NET) {
            LocationHelper.getInstance().startLocation(getActivity());
            Logger.d("定位中");
            setTitleAddress(getString(R.string.location_processing));
            showLoadingProgressBar(true, null);
        }
    }
    private void handleLocationResult() {
        setTitleAddress();
        if (LocationHelper.locationState == LocationHelper.LOCATION_STATE_SUCCESS) {
            // 获取完经纬度，去查询配送范围内的门店列表
            RequestQueryStoreAndSwitch requestQueryStoreAndSwitch = new
RequestQueryStoreAndSwitch();
            LocationModel locationModel = LocationHelper.locationModel;
            Logger.d("定位成功，经度：" + locationModel.longitude + "，纬度：" +
locationModel.latitude);
            if (!TextUtils.isEmpty(locationModel.contactId)) {
                requestQueryStoreAndSwitch.contactId = locationModel.contactId;
            }
            requestQueryStoreAndSwitch.longitude = locationModel.longitude;
            requestQueryStoreAndSwitch.latitude = locationModel.latitude;
            mPresenter.queryStoreAndSwitch(requestQueryStoreAndSwitch);
        } else if (LocationHelper.locationState == LocationHelper.LOCATION_STATE_PROCESSING) {
            Logger.d("定位中");
            showLoadingProgressBar(true, null);
        } else if (LocationHelper.locationState == LocationHelper.LOCATION_STATE_FAILED) {
            Logger.d("定位没有权限或失败");
            permissionDialog = AndroidUtils.notifySettingPermission(mBaseActivity,
getString(R.string.open_location_service));
        } else if (LocationHelper.locationState == LocationHelper.LOCATION_STATE_NO_NET) {
            Logger.d("没联网");
            mExceptLayout.showNetworkLoadFailedView();
```

```
        }
    }
    // 定位响应
    private BroadcastReceiver mReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context arg0, Intent intent) {
            if (intent != null) {
                String action = intent.getAction();
                switch (action == null ? "" : action) {
                    case Constants.ACTION_LOCATION_STATE_CHANGED:// 定位成功
                        showLoadingProgressBar(false, null);
                        handleLocationResult();
                        break;
                    case Constants.ACTION_SELECT_ONE_STORE:// 选择了一个门店
                        setCurrentStoreModel((ResponseQueryStoreAndSwitch.StoreBean)
intent.getSerializableExtra(HomePageFragment.EXTRA_STORE_INFO));
                        mPresenter.switchStore(currentStoreModel.getStoreNo(),
currentStoreModel.getStoreType());
                        updateTitleLogo();
                        break;
                    case Constants.ACTION_SWITCH_ONE_STORE:// 切换了一个门店
                        if (popupWindow != null && popupWindow.isShowing()) {
                            popupWindow.dismiss();
                            popupWindow = null;
                        }
                        setCurrentStoreModel((ResponseQueryStoreAndSwitch.StoreBean)
intent.getSerializableExtra(HomePageFragment.EXTRA_STORE_INFO));
                        String switchStoreNo = currentStoreModel.getStoreNo();
                        for (int i = 0; i < inDeliveryStoreList.size(); i++) {
                            ResponseQueryStoreAndSwitch.StoreBean        storeBean        =
inDeliveryStoreList.get(i);
                            if (switchStoreNo.equals(storeBean.getStoreNo())) {
                                storeBean.setSelected(true);
                            } else {
                                storeBean.setSelected(false);
                            }
                        }
                        mPresenter.switchStore(switchStoreNo,
currentStoreModel.getStoreType());
                        updateTitleLogo();
                        break;
                    default:
                        break;
                }
```

```
            }
        }
    };
    @Override
    public void setDatas(List<ResponseHomePageStorageQry.DatasBean> datas) {
        // 设置门店选择页面不可见
        if (llOutDelivery != null) {
            llOutDelivery.setVisibility(View.GONE);
            recyclerView.setVisibility(View.VISIBLE);
        }
        // 把异常布局隐藏掉
        mExceptLayout.showContentView();
        MultiRecyclerAdapter<ResponseHomePageStorageQry.DatasBean> adapter;
        adapter = new MultiRecyclerAdapter<>(mBaseActivity, false);
//        // 为了拉到最下面有一些空白，Footer 必须放在第一个
//                  adapter.setFooterDelegate(new  BaseFooterAdapterDelegate(mBaseActivity,
FooterViewHolder.class));
        adapter.addDelegate(new BannerAdapterDelegate(mBaseActivity));
        adapter.addDelegate(new SearchBarAdapterDelegate(mBaseActivity));
        adapter.addDelegate(new IconBannerAdapterDelegate(mBaseActivity));
        adapter.addDelegate(new ScrollableGoodsAdapterDelegate(mBaseActivity));
        adapter.addDelegate(new AdvertFlexAdapterDelegate(mBaseActivity));
        adapter.addDelegate(new AdvertTwoRowAdapterDelegate(mBaseActivity));
        adapter.addDelegate(new SectionTitleAdapterDelegate(mBaseActivity));
        adapter.addDelegate(new GoodsAdapterDelegate(mBaseActivity));
        adapter.addDelegate(new PromGoodsAdapterDelegate(mBaseActivity));
        adapter.addDelegate(new ActivityGoodsAdapterDelegate(mBaseActivity));
        recyclerView.setAdapter(adapter);
        homePageDataList = datas;
        // 设置搜索布局的位置
        searchLayoutPosition = NO_SEARCH_LAYOUT_POSITION;
        for (int i = 0; i < homePageDataList.size(); i++) {
            if
(HomePageModel.VIEW_TYPE_SEARCH_BAR.equals(homePageDataList.get(i).getModuleType())) {
                searchLayoutPosition = i;
            }
        }
        adapter.setItems(homePageDataList);
    }
    @Override
    public void outDelivery(ResponseQueryStoreAndSwitch responseQueryStoreAndSwitch) {
        updateTitleLogo(null);
        if (llOutDelivery == null) {
            vsOutDelivery.inflate();
```

```
        }
        // 把异常布局隐藏掉
        mExceptLayout.showContentView();
        recyclerView.setVisibility(View.GONE);
        View root = getView();
        if (root == null) {
            return;
        }
        RecyclerView outRecyclerView = root.findViewById(R.id.recyclerView);
        llOutDelivery = root.findViewById(R.id.llOutDelivery);
        llOutDelivery.setVisibility(View.VISIBLE);
        CommonRecyclerAdapter<ResponseQueryStoreAndSwitch.StoreBean,
StoreSelectItemViewHolder> adapter =
                new    CommonRecyclerAdapter<>(mContext,    StoreSelectItemViewHolder.class,
false);
        outRecyclerView.setLayoutManager(new LinearLayoutManager(mContext));
        adapter.appendData(responseQueryStoreAndSwitch.getList(), 0);
        outRecyclerView.setAdapter(adapter);
    }
    private void updateTitleLogo() {
        updateTitleLogo(currentStoreModel);
    }
    @Override
    public void updateTitleLogo(ResponseQueryStoreAndSwitch.StoreBean storeBean) {
        setCurrentStoreModel(storeBean);
        if (currentStoreModel == null) {
            ivLogo.setImageResource(R.drawable.pic_home_logo);
            ivTriangle.setVisibility(View.GONE);
            return;
        }
        if (!TextUtils.isEmpty(storeBean.getStoreTypeUrL())) {
            Picasso.with(mContext).load(storeBean.getStoreTypeUrL()).into(ivLogo);
        }
        if (inDeliveryStoreList != null && inDeliveryStoreList.size() > 1) {
            ivTriangle.setVisibility(View.VISIBLE);
            ivTriangle.setImageResource(R.drawable.icon_index_down);
        } else {
            // 单门店展示
            ivTriangle.setVisibility(View.GONE);
        }
    }
    @Override
    public void showPopupWindow() {
        // 如果是单门店，则不弹出门店选择框
```

```
        if (inDeliveryStoreList == null || inDeliveryStoreList.size() < 2) {
            return;
        }
        ivTriangle.setImageResource(R.drawable.icon_index_up);
        // 多门店展示，默认弹出门店选择列表
        View                                    contentView                                    =
LayoutInflater.from(mContext).inflate(R.layout.pop_home_page_store_select, null);
        popupWindow = new PopupWindow(mContext);
        RecyclerView popRecyclerView = contentView.findViewById(R.id.recyclerView);
        TextView tvDeliveryNum = contentView.findViewById(R.id.tvDeliveryNum);
        tvDeliveryNum.setText(getString(R.string.delivery_no, inDeliveryStoreList.size()));
        popRecyclerView.setLayoutManager(new LinearLayoutManager(mContext));
        LinearLayout llPopupContent = contentView.findViewById(R.id.llPopupContent);
        llPopupContent.setClickable(true);
        View vOutPopupContent = contentView.findViewById(R.id.vOutPopupContent);
        vOutPopupContent.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                popupWindow.dismiss();
                popupWindow = null;
            }
        });
        popupWindow.setOnDismissListener(new PopupWindow.OnDismissListener() {
            @Override
            public void onDismiss() {
                ivTriangle.setImageResource(R.drawable.icon_index_down);
            }
        });
        CommonRecyclerAdapter<ResponseQueryStoreAndSwitch.StoreBean,
PopStoreSelectItemViewHolder> adapter =
                new   CommonRecyclerAdapter<>(mContext,   PopStoreSelectItemViewHolder.class,
false);
        adapter.appendData(inDeliveryStoreList, 0);
        popRecyclerView.setAdapter(adapter);
        popupWindow.setContentView(contentView);
        popupWindow.setWidth(ViewGroup.LayoutParams.MATCH_PARENT);
        popupWindow.setHeight(ViewGroup.LayoutParams.MATCH_PARENT);
        popupWindow.setOutsideTouchable(true);
        popupWindow.setFocusable(true);
        popupWindow.setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
        AndroidUtils.showAsDropDown(popupWindow, llTitleParent, 0, 0);
    }
    /**
     * 设置顶部位置
```

```
     */
    private void setTitleAddress() {
        setTitleAddress(null);
    }
    private void setTitleAddress(String address) {
        if (!TextUtils.isEmpty(address)) {
            tvAddress.setText(address);
            return;
        }
        if (LocationHelper.locationState == LocationHelper.LOCATION_STATE_SUCCESS) {
            tvAddress.setText(LocationHelper.locationModel.street);
        } else if (LocationHelper.locationState == LocationHelper.LOCATION_STATE_PROCESSING) {
            tvAddress.setText(R.string.location_processing);
        } else {
            tvAddress.setText(R.string.location_fail);
        }
    }
    @Override
    public void finishRefresh() {
//        showLoadingWithCustomeSwipe(false);
    }
    @Override
    public      void      setInDeliveryStoreList(List<ResponseQueryStoreAndSwitch.StoreBean>
inDeliveryStoreList) {
        this.inDeliveryStoreList = inDeliveryStoreList;
    }
    @Override
    public void selectSwitchStoreSuccess() {
        Logger.d("选择或者切换门店成功后发送广播");
        Intent intent = new Intent(Constants.ACTION_SELECT_SWITCH_STORE_SUCCESS);
        LocalBroadcastManager.getInstance(mContext).sendBroadcast(intent);
    }
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode != IntentIntegrator.REQUEST_CODE) {
            super.onActivityResult(requestCode, resultCode, data);
            return;
        }
        IntentResult result = IntentIntegrator.parseActivityResult(resultCode, data);
        String contents = result.getContents();
        if (!TextUtils.isEmpty(contents)) {
            if (URLUtil.isNetworkUrl(contents)) {
                Uri uri = Uri.parse(contents);
                //店铺
```

```
                storeNo = uri.getQueryParameter("storeNo");
                //货架
                storeType = uri.getQueryParameter("storeType");
                String storageNo = uri.getQueryParameter("storageNo");
                Constants.STORENO = storeNo;
                Constants.STORETYPE = storeType;
                mScanPresenter.scanStore(storeNo, storeType ,storageNo);
            } else {
                ToastUtil.showShortToast("请先扫货架二维码");
            }
        }
    }
    @Override
    public void showResult(String result) {
        if (TextUtils.equals(result, "failed")) {
            ToastUtil.showShortToast("获取店铺信息失败");
        } else {
            startActivity(new                                                    Intent(mContext,
ScanCodePurchaseActivity.class).putExtra("storeName", result)
           /* .putExtra("storeNo",storeNo)
            .putExtra("storeType",storeType)*/);
        }
    }
    @Override
    public String getStatusBarStyle() {
        return STYLE_STATUS_BAR_WHITE;
    }
    private void setCurrentStoreModel(ResponseQueryStoreAndSwitch.StoreBean storeModel) {
        this.currentStoreModel = storeModel;
        if (storeModel == null) {
            Store.setCurrentStoreNo(mContext, null);
        } else {
            Store.setCurrentStoreNo(mContext, storeModel.getStoreNo());
        }
    }
}
package com.baoneng.bnmall.ui.shelf;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.support.annotation.DrawableRes;
import android.support.annotation.Nullable;
```

```
import android.support.v4.content.LocalBroadcastManager;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.content.res.AppCompatResources;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.text.InputType;
import android.text.TextUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.TextView;
import com.baoneng.bnmall.R;
import com.baoneng.bnmall.common.Constants;
import com.baoneng.bnmall.model.goods.RspGoodsClass;
import com.baoneng.bnmall.ui.BaseFragment;
import com.baoneng.bnmall.ui.MainActivity;
import com.baoneng.bnmall.ui.search.SearchActivity;
import com.baoneng.bnmall.ui.shelf.goodslist.ShelfGoodsListFragment;
import com.baoneng.bnmall.ui.shelf.utils.GroupItemsLayout;
import com.baoneng.bnmall.ui.shelf.utils.ShelfSortBy;
import com.baoneng.bnmall.utils.anim.OnParabolaAnimation;
import com.yqritc.recyclerviewflexibledivider.HorizontalDividerItemDecoration;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;
import static com.baoneng.bnmall.ui.BaseActivity.STYLE_STATUS_BAR_WHITE;
/**
 * 分类货架
 */
public class ShelfFragment extends BaseFragment<ShelfPresenter> implements ShelfView,
ShelfGoodsListFragment.Listener {
    private OnParabolaAnimation mCallBack;
    @BindView(R.id.et_search_edit)
    EditText mSearchEdit;
    @BindView(R.id.sort)
    View sortTitle;
    @BindView(R.id.sortRG)
    View sortRG;
    @BindView(R.id.sortTV)
```

```
TextView sortTV;
@BindView(R.id.sort_divide_line)
View sort_divide_line;
@BindView(R.id.sort_price)
RadioButton sortPrice;
@BindView(R.id.group)
GroupItemsLayout group;
@BindView(R.id.menu)
RecyclerView shelfMenu;//分类
@BindView(R.id.loadingSwiperefresh)
SwipeRefreshLayout mSwipeRefresh;
@BindView(R.id.error_notice)
View errorView;
LinearLayoutManager lm;
ShelfGoodsListFragment goodsListFragment;
private ShelfMenuAdapter mShelfMenuAdapter = new ShelfMenuAdapter();
//排序方式
private ShelfSortBy sortBy = ShelfSortBy.Sales;
//外部传入一级分类号
private String mNeedShowFirstClassNo;
@Override
public int getContentViewId() {
    return R.layout.fragment_shelf;
}
@Override
public void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mPresenter = new ShelfPresenter(this);
    IntentFilter filter = new IntentFilter();
    filter.addAction(Constants.ACTION_SELECT_SWITCH_STORE_SUCCESS);
    LocalBroadcastManager.getInstance(getContext()).registerReceiver(mReceiver, filter);
    if (getArguments() != null) {
        String firstClassNo = getArguments().getString(Constants.EXTRA_SHELF_GOODS_CLASS);
        setCurrentFirstClass(firstClassNo);
    }
}
@Override
public void onAttach(Context context) {
    super.onAttach(context);
    if (context instanceof OnParabolaAnimation) {
        mCallBack = (OnParabolaAnimation) context;
    }
}
@Override
```

```
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        View v = super.onCreateView(inflater, container, savedInstanceState);
        initSearchBar();
        //左方分类菜单
        lm = new LinearLayoutManager(getContext(), LinearLayoutManager.VERTICAL, false);
        shelfMenu.setLayoutManager(lm);
        shelfMenu.setAdapter(mShelfMenuAdapter);
        shelfMenu.addItemDecoration(new HorizontalDividerItemDecoration.Builder(getContext())
                .colorResId(R.color.gray_line)
                .sizeResId(R.dimen.shelf_divide_line_width).build());
        group.setPivotY(0);
        group.setListener(groupItemsChangedListener);
        group.addOnLayoutChangeListener(mOnGroupLayoutChanged);
        mSwipeRefresh.setColorSchemeResources(R.color.progress);
        mSwipeRefresh.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
            @Override
            public void onRefresh() {
                mPresenter.loadData(true);
            }
        });
        goodsListFragment = (ShelfGoodsListFragment) getChildFragmentManager().findFragmentById(R.id.goodsList);
        return v;
    }
    @Override
    public void onDestroy() {
        LocalBroadcastManager.getInstance(getContext()).unregisterReceiver(mReceiver);
        super.onDestroy();
    }
    private void initSearchBar() {
        mSearchEdit.setCursorVisible(false);
        mSearchEdit.setFocusable(false);
        mSearchEdit.setInputType(InputType.TYPE_NULL);
    }
    @Override
    public void showLoadingProgressBar(boolean show, Object type) {
        mSwipeRefresh.setRefreshing(show);
    }
    /**
     * 设置当前分类
     */
    @Override
    public void setCurrentFirstClass(String classNo) {
```

```
        mNeedShowFirstClassNo = classNo;
        //如果当前有数据，尝试直接显示
        boolean find = trySetSelectedFirstClass();
        if (!find) {
            //如果没有 started，会在 start 时 loadData，无需手动调用 loadData
            if (mPresenter.isStarted()){
                mPresenter.loadData(true);
            }
        }
    }
    @Override
    public void showNoStore(boolean noStore) {
        errorView.setVisibility(noStore ? View.VISIBLE : View.GONE);
    }
    @Override
    public void showFirstClassList(List<RspGoodsClass.ShelfCLassItem> menuItems) {
        if (menuItems == null) {
            menuItems = new ArrayList<>();
        }
        mShelfMenuAdapter.items = menuItems;
        mShelfMenuAdapter.notifyDataSetChanged();
        trySetSelectedFirstClass();
        goodsListFragment.reloadData();
        if (mShelfMenuAdapter.selectedIndex < 0 || mShelfMenuAdapter.selectedIndex >= menuItems.size()) {
            mShelfMenuAdapter.setSelected(0);
        }
    }
    //根据外部传入的 classNO，尝试设置为 selected
    private    boolean trySetSelectedFirstClass(){
        if (mNeedShowFirstClassNo == null) {
            return false;
        }
        if (mShelfMenuAdapter.items != null && mShelfMenuAdapter.items.size() > 0) {
            for (int i = mShelfMenuAdapter.items.size() - 1; i >= 0; i--) {
                RspGoodsClass.ShelfCLassItem item = mShelfMenuAdapter.items.get(i);
                if (Objects.equals(item.classNo, mNeedShowFirstClassNo)) {
                    mShelfMenuAdapter.setSelected(i);
                    mNeedShowFirstClassNo = null;
                    return true;
                }
            }
        }
        return false;
    }
```

```
    @Override
    public void showSecondClassList(List<RspGoodsClass.ShelfCLassItem> menuItems) {
        if (menuItems == null) {
            menuItems = new ArrayList<>();
        }
        //添加一个“全部”
        List<RspGoodsClass.ShelfCLassItem> list = new ArrayList<>();
        RspGoodsClass.ShelfCLassItem allItem = new RspGoodsClass.ShelfCLassItem();
        allItem.className = "全部";
        list.add(allItem);
        list.addAll(menuItems);
        group.setItems(list);
        group.setSelectedIndex(0);
    }
    //监听 group 高度变化
    private View.OnLayoutChangeListener mOnGroupLayoutChanged = new View.OnLayoutChangeListener() {
        @Override
        public void onLayoutChange(View v, int left, int top, int right, int bottom, int oldLeft, int oldTop, int
oldRight, int oldBottom) {
            final int oldH = oldBottom - oldTop;
            final int h = bottom - top;
            if (h != oldH) {
                mHandler.post(new Runnable() {
                    @Override
                    public void run() {
                        goodsListFragment.resetGoodsListHeader(h);
                    }
                });
            }
        }
    };
//
//    public void setCurrentFirstClass(RspGoodsClass.ShelfCLassItem position) {
//        mShelfMenuAdapter.setSelected(position);
//    }
    @OnClick(R.id.go_shopping)
    void onCLickGoShopping(){
        MainActivity.switchTab(getActivity(), 0);
    }
    //---------------------------------- 事件接收 ----------------------------------
    private BroadcastReceiver mReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            //门店切换
```

```
                if (TextUtils.equals(Constants.ACTION_SELECT_SWITCH_STORE_SUCCESS, intent.getAction())) {
                    mPresenter.loadData(true);
                }
            }
        };
        //---------------------------------- 列表上下滑处理 ----------------------------------
//      private int sortAnimState = 0;
        private GroupItemsLayout.GroupItemsChangedListener groupItemsChangedListener = new
GroupItemsLayout.GroupItemsChangedListener() {
            @Override
            public void itemChanged(RspGoodsClass.ShelfCLassItem newItem) {
                sortTV.setText(sortText());
                goodsListFragment.setSecondClass(newItem);
                goodsListFragment.reloadData();
            }
        };
        // scrollPosition 仅当 SORT_STATE_ANIMATING 有效
        @Override
        public void setSortState(int sortState, float animPercent) {
            switch (sortState) {
                case ShelfConstants.SORT_STATE_ANIMATING: {
                    sortRG.setVisibility(View.VISIBLE);
                    group.setVisibility(View.VISIBLE);
                    sortTV.setVisibility(View.VISIBLE);
                    sortRG.setEnabled(false);
                    group.setEnabled(false);
                    sortTV.setEnabled(false);
//                      float animPercent = (scrollPosition > groupOriHeight ? groupOriHeight : scrollPosition) /
groupOriHeight;
                    float groupY = sort_divide_line.getBottom() - sortTitle.getHeight() * animPercent;
                    group.setY((int) groupY);
                    group.setScaleY(((sortTitle.getHeight() - group.getHeight()) / group.getHeight()) *
animPercent + 1);
                    group.setAlpha(1 - animPercent);
                    sortRG.setAlpha((1 - animPercent) * 0.3f);
                    sortTV.setAlpha(0.4f + animPercent * 0.6f);
                    break;
                }
                case ShelfConstants.SORT_STATE_EXPAND: {
                    sortRG.setVisibility(View.VISIBLE);
                    group.setVisibility(View.VISIBLE);
                    sortTV.setVisibility(View.INVISIBLE);
                    sortRG.setEnabled(false);
                    group.setEnabled(false);
```

```
                sortTV.setEnabled(false);
                group.setY(sort_divide_line.getBottom());
                group.setScaleY(1);
                group.setAlpha(1);
                sortRG.setAlpha(1);
                break;
            }
            case ShelfConstants.SORT_STATE_PACKED: {
                sortRG.setVisibility(View.INVISIBLE);
                group.setVisibility(View.INVISIBLE);
                sortTV.setVisibility(View.VISIBLE);
                sortRG.setEnabled(false);
                group.setEnabled(false);
                sortTV.setEnabled(true);
                break;
            }
            default:
                break;
        }
    }
    private CharSequence sortText() {
        RspGoodsClass.ShelfCLassItem groupItem = group.getSelectedItem();
        RspGoodsClass.ShelfCLassItem currentItem;
        String text;
        if (groupItem != null) {
            text = sortBy.getText() + " - " + groupItem.className;
        } else if ((currentItem = mShelfMenuAdapter.currentFirstClass()) != null) {
            text = sortBy.getText() + " - " + currentItem.className;
        } else {
            text = sortBy.getText();
        }
        return text;
    }
    @OnClick(R.id.sortTV)
    void onClickSortResult() {
        setSortState(ShelfConstants.SORT_STATE_EXPAND, 0);
    }
    @OnClick(R.id.et_search_edit)
    void onClickSearchView() {
        startActivity(new Intent(this.getActivity(), SearchActivity.class));
    }
    @OnClick(R.id.sortTV)
    void onExpand() {
        setSortState(ShelfConstants.SORT_STATE_EXPAND, 0);
```

```
    }
    @OnClick({R.id.sort_price, R.id.sort_sales})
    void onClickSort(View v) {
        @DrawableRes int priceSortTypeDrawable = 0;
        switch (v.getId()) {
            case R.id.sort_price:
                if (sortBy == ShelfSortBy.Price_ACE) {
                    sortBy = ShelfSortBy.Price_DESC;
                    priceSortTypeDrawable = R.drawable.search_icon_sort_down;
                } else {
                    sortBy = ShelfSortBy.Price_ACE;
                    priceSortTypeDrawable = R.drawable.search_icon_sort_up;
                }
                break;
            case R.id.sort_sales:
                sortBy = ShelfSortBy.Sales;
                priceSortTypeDrawable = R.drawable.search_icon_sort_def;
                break;
        }
        sortTV.setText(sortText());
//          priceSort.setImageResource(priceSortTypeDrawable);
        sortPrice.setCompoundDrawablesWithIntrinsicBounds(null,null,
AppCompatResources.getDrawable(getContext(),priceSortTypeDrawable),null);
        goodsListFragment.setSortBy(sortBy);
        goodsListFragment.reloadData();
    }
    //---------------------------------- 一级分类菜单 ----------------------------------
    private class ShelfMenuAdapter extends RecyclerView.Adapter<ShelfMenuHolder> {
        List<RspGoodsClass.ShelfCLassItem> items;
        int selectedIndex = -1;
        private RspGoodsClass.ShelfCLassItem currentFirstClass() {
            if (items == null || items.size() == 0) {
                return null;
            }
            return items.get(selectedIndex);
        }
        @Override
        public ShelfMenuHolder onCreateViewHolder(ViewGroup parent, int viewType) {
            View v = LayoutInflater.from(parent.getContext())
                    .inflate(R.layout.shelf_menu_item, parent, false);
            return new ShelfMenuHolder(v);
        }
        @Override
        public void onBindViewHolder(ShelfMenuHolder holder, int position) {
```

```
            RspGoodsClass.ShelfCLassItem item = items.get(position);
            holder.textView.setText(item.className);
            holder.textView.setSelected(position == selectedIndex);
            holder.item = item;
        }
        @Override
        public int getItemCount() {
            return items != null ? items.size() : 0;
        }
        void setSelected(RspGoodsClass.ShelfCLassItem item) {
            if (items == null) {
                return;
            }
            int pos = items.indexOf(item);
            setSelected(pos);
        }
        void setSelected(int position) {
            RspGoodsClass.ShelfCLassItem item = items.get(position);
            int oldIdx = selectedIndex;
            selectedIndex = position;
            goodsListFragment.setFirstClass(item);
            mPresenter.setFirstClass(item);
            notifyItemChanged(oldIdx);
            notifyItemChanged(selectedIndex);
            shelfMenu.scrollToPosition(position);
            if (position < lm.findFirstCompletelyVisibleItemPosition() || position > lm.findLastCompletelyVisibleItemPosition()) {
                lm.scrollToPosition(position);
            }
        }
    }
    class ShelfMenuHolder extends RecyclerView.ViewHolder {
        @BindView(R.id.text)
        TextView textView;
        RspGoodsClass.ShelfCLassItem item;
        ShelfMenuHolder(View itemView) {
            super(itemView);
            ButterKnife.bind(this, itemView);
        }
        @OnClick(R.id.text)
        void onClickItem() {
            mShelfMenuAdapter.setSelected(getAdapterPosition());
        }
    }
```

```
    //---------------------------------- 一级分类菜单 ----------------------------------
    @Override
    public String getStatusBarStyle() {
        return STYLE_STATUS_BAR_WHITE;
    }
}
package com.baoneng.bnmall.ui.shoppingcar;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.LocalBroadcastManager;
import android.support.v7.widget.DefaultItemAnimator;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.text.SpannableString;
import android.text.TextUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewParent;
import android.widget.CheckBox;
import android.widget.TextView;
import com.baoneng.bnmall.R;
import com.baoneng.bnmall.XApplication;
import com.baoneng.bnmall.common.Constants;
import com.baoneng.bnmall.model.UserLoginInfo;
import com.baoneng.bnmall.model.mainscreen.LocationModel;
import com.baoneng.bnmall.model.shoppingcar.ShoppingCarModalResponse;
import com.baoneng.bnmall.network.BNUrl;
import com.baoneng.bnmall.ui.BaseFragment;
import com.baoneng.bnmall.ui.MainActivity;
import com.baoneng.bnmall.ui.WebViewActivity;
import com.baoneng.bnmall.ui.authentication.AuthenticationActivity;
import com.baoneng.bnmall.ui.mainscreen.homepage.LocationHelper;
import com.baoneng.bnmall.utils.DensityUtil;
import com.baoneng.bnmall.utils.NetworkUtils;
import com.baoneng.bnmall.utils.SpannableUtil;
import com.baoneng.bnmall.utils.ToastUtil;
import com.baoneng.bnmall.utils.UrlParseUtil;
import com.baoneng.bnmall.utils.Utils;
import com.baoneng.bnmall.utils.keyboard.KeyboardStateHelper;
import com.baoneng.bnmall.utils.keyboard.KeyboardStateHelper.SoftKeyboardStateListener;
```

```
import com.baoneng.bnmall.widget.SwipeMenuLayout;
import com.baoneng.bnmall.widget.dialog.NormalDialog;
import com.chad.library.adapter.base.BaseQuickAdapter;
import com.chad.library.adapter.base.entity.MultiItemEntity;
import com.oushangfeng.pinnedsectionitemdecoration.PinnedHeaderItemDecoration;
import com.oushangfeng.pinnedsectionitemdecoration.callback.OnHeaderClickListener;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import butterknife.BindView;
import butterknife.OnClick;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.ADDBUY_GOODS_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.BUY_SEND_GOODS_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.EMPTY_NETWORK_ERROR_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.EMPTY_OTHER_ERROR_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.GIFT_GOODS_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.GROUP_GOODS_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.NOMAL_GOODS_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.OUT_DELIVERY_GOODS_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.OUT_DELIVERY_HEADER_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.PACKAGE_GOODS_HEADER_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.PACKAGE_GOODS_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.PROMOTION_ADDBUY_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.REACH_DISCOUNT_GOODS_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.SALESOUT_GOODS_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.SHOP_BANLANCE_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.SHOP_ENDER_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.SHOP_HEADER_ITEM_VIEW_TYPE;
import static com.baoneng.bnmall.ui.shoppingcar.ShoppingCarListAdapter.Y;
/**
 * 作者:huangcs
 * 时间:2018/3/8
 * 描述: 购物车 frag
 */
public class ShoppingCarFrag extends BaseFragment<ShoppingCarContract.Presenter> implements
ShoppingCarContract.View {
```

```
    @BindView(R.id.edit)
    TextView mEdit;
    @BindView(R.id.shoppingcar_list)
    RecyclerView mShoppingCarRecyclerView;
    @BindView(R.id.all_count)
    TextView mAllCount;
    @BindView(R.id.count_des)
    TextView mCountDes;
    @BindView(R.id.choose_all)
    CheckBox mChooseAll;
    @BindView(R.id.balance)
    TextView mBalance;
    @BindView(R.id.divide_bottom)
    View mDivideBottom;
    @BindView(R.id.countContainer)
    View mCountContainer;
    @BindView(R.id.title)
    TextView mTitle;
    @BindView(R.id.root)
    View mRoot;
//      @BindView(R.id.balance_line)
//      View mBalanceLine;
    //      @BindView(R.id.smartRefreshLayout)
//      SmartRefreshLayout mSmartRefreshLayout;
    //      @BindView(R.id.balance_layout)
//      View mBanlanceLayout;
    @BindView(R.id.bottom_view)
    View mBottomView;
    private boolean isSelectedAll;
    public static boolean mIsEditStatue;
    private List<MultiItemEntity> mCarModalList = new ArrayList<>();
    private ShoppingCarListAdapter mShoppingCarListAdapter;
    private TextView mAdress;
    private LinearLayoutManager mLayoutManager;
    private PinnedHeaderItemDecoration mHeaderItemDecoration;
    private boolean isALlSelected;
    private boolean isDelEnable;
    private int startHeaderPos;
    private int endHeaderPos;
    private NormalDialog mDelTipDialog;
    private int lastBalancePosition = NOT_FOUND;
    private int mOutDeliveryPosition = NOT_FOUND;
    private int locationViewHeight;
    private static final int HEADER_COUNT = 1;
```

```
public static final int NOT_FOUND = -1;
private List<ShoppingCarModalResponse.CartInfoBean> mShopList;
private int changeItemsNum;
private String mAddressStr;
private TextView mNameTv;
private int mBalancePosition = NOT_FOUND;
private boolean mRefreshDelay;
private String mContactId;
private View mTopView;
private View mHeaderBottomView;
private boolean isInFragment;
private KeyboardStateHelper mKeyboardStateHelper;
private SoftKeyboardStateListener mKeyboardListener;
private int mLastChangeNum;
private int mLastChangePostion;
private boolean mIsInputFinish;
boolean isRecyclerViewInitFinish = false;
private Handler handler = new Handler();
private Map<String, Boolean> mEditSelectedMap = new HashMap<>();
private int mDelPosition;
@Override
public int getContentViewId() {
    return R.layout.layout_shoppingcarfrag;
}
@Override
public void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mPresenter = new ShoppingCarPresenter(this);
    isInFragment = true;
}
@Override
public void bindView(View view) {
    super.bindView(view);
    setBalanceViewVisiable(View.GONE);//先隐藏掉
}
@Override
protected void initUI() {
    super.initUI();
    //定位的情况
    addKeyboardListener();
    locationViewHeight = DensityUtil.dip2px(mContext, 50);
    mLayoutManager = new LinearLayoutManager(mContext, LinearLayoutManager.VERTICAL, false);
    mShoppingCarRecyclerView.setLayoutManager(mLayoutManager);
    mHeaderItemDecoration = new
```

```
PinnedHeaderItemDecoration.Builder(ShoppingCarListAdapter.SHOP_HEADER_ITEM_VIEW_TYPE).enableDivider(
false)
                    .setClickIds(R.id.fare_des,
R.id.radio_all).disableHeaderClick(false).setHeaderClickListener(clickAdapter).create();
        mShoppingCarRecyclerView.addItemDecoration(mHeaderItemDecoration);
        mShoppingCarListAdapter = new ShoppingCarListAdapter(mCarModalList);
        ((DefaultItemAnimator)
mShoppingCarRecyclerView.getItemAnimator()).setSupportsChangeAnimations(false);//去除默认动画，避免图
片闪烁的假象
        mShoppingCarListAdapter.setHasStableIds(true);//
        mShoppingCarRecyclerView.setAdapter(mShoppingCarListAdapter);
        View    headerView    =    LayoutInflater.from(mContext).inflate(R.layout.item_shoppingcar_header,
mShoppingCarRecyclerView, false);
        mAdress = headerView.findViewById(R.id.address);
        mNameTv = headerView.findViewById(R.id.name);
        mTopView = headerView.findViewById(R.id.topView);
        mHeaderBottomView = headerView.findViewById(R.id.bottomView);
        setHeaderViewClickListener(headerView);
        mShoppingCarListAdapter.addHeaderView(headerView);
        // 因为添加了 1 个头部，他是不在 clickAdapter.getData 这个数据里面的，所以这里要设置数据的
偏移值告知 ItemDecoration 真正的数据索引
        mHeaderItemDecoration.setDataPositionOffset(mShoppingCarListAdapter.getHeaderLayoutCount());
        loadData();
        setNumberChangerListener();
        addOnScrollListener();
        setOnItemChildClickListener();
//          mSmartRefreshLayout.setEnableLoadmore(false);
//          mSmartRefreshLayout.setEnableOverScrollDrag(false);
//          mSmartRefreshLayout.setLayerType(View.LAYER_TYPE_SOFTWARE, null);//禁用硬件加速
//          mSmartRefreshLayout.setOnRefreshListener(new OnRefreshListener() {
//              @Override
//              public void onRefresh(RefreshLayout refreshlayout) {
//                  loadData();
//              }
//          });
    }
    @Override
    public void onHiddenChanged(boolean hidden) {
        super.onHiddenChanged(hidden);
        if (mInitFinished && !hidden) {//可见
            if (mKeyboardStateHelper != null) {
                mKeyboardStateHelper.addSoftKeyboardStateListener(mKeyboardListener);
                mKeyboardStateHelper.setPauseListener(false);
            }
```

```
            isInFragment = true;
            if (mIsEditStatue) {//如果是编辑状态切换回来后变成正常状态
                mIsEditStatue = false;
                setEditView();
                switchToNormolStatue();
            }
            loadData();
        } else {
            isInFragment = false;
            if (mKeyboardStateHelper != null) {
                mKeyboardStateHelper.setPauseListener(true);
                mKeyboardStateHelper.removeSoftKeyboardStateListener(mKeyboardListener);
            }
        }
    }
    @Override
    public void onResume() {
        if (mInitFinished && isInFragment) {//初始化完成且不是编辑状态且且在这个页面
            if (mRefreshDelay) {
                handler.postDelayed(new Runnable() {
                    @Override
                    public void run() {
                        loadData();
                    }
                }, 300);
            } else {
                loadData();
            }
        }
        super.onResume();
    }
    @Override
    public void onDetach() {
        handler.removeCallbacksAndMessages(null);
        mKeyboardStateHelper.removeGlobalLayoutListener();
        super.onDetach();
    }
    private void startToSetBalanceView() {
        if (isNotEmpty()) {
            if (!isRecyclerViewInitFinish && !mIsEditStatue) {
                mShoppingCarRecyclerView.post(new Runnable() {
                    @Override
                    public void run() {
                        isRecyclerViewInitFinish = true;
```

```
                        findLastVisiableItemPosition();
                    }
                });
            } else {
                setBalanceViewDifferentStatue();
            }
        } else {
            setBalanceViewVisiable(View.GONE);
            mBottomView.setVisibility(View.GONE);
            mChooseAll.setChecked(false);
            mChooseAll.setEnabled(false);
            mAllCount.setText(getString(R.string.all_count));
            mCountDes.setText(getString(R.string.all_discount));
            mBalance.setText(getString(R.string.go_balance, "0"));
        }
    }
    private void setBalanceViewDifferentStatue() {
        if (mIsEditStatue) {
            if (isNotEmpty() && mOutDeliveryPosition != 2) {
                setBalanceViewVisiable(View.VISIBLE);
            } else {
                setBalanceViewVisiable(View.GONE);
                mBottomView.setVisibility(View.GONE);
            }
        } else {
            mChooseAll.setEnabled(true);
            findLastVisiableItemPosition();
        }
    }
    //查找最后一个可见 item 的位置，用来设置结算按钮的显示和隐藏
    private void findLastVisiableItemPosition() {
        int lastCompletelyVisibleItemPosition = mLayoutManager.findLastCompletelyVisibleItemPosition() -
HEADER_COUNT;//减去头部
        int findLastVisibleItemPosition = mLayoutManager.findLastVisibleItemPosition() - HEADER_COUNT;//减
去头部
        if (lastCompletelyVisibleItemPosition <= NOT_FOUND || findLastVisibleItemPosition >=
mCarModalList.size()) {
            return;
        }
        switch (mCarModalList.get(findLastVisibleItemPosition).getItemType()) {
            case SHOP_BANLANCE_ITEM_VIEW_TYPE://店铺结算 item
                if (lastCompletelyVisibleItemPosition == findLastVisibleItemPosition) {//说明尾部完全可见
                    setBalanceViewVisiable(View.GONE);
                } else {
```

```
                    findNSetBalanceView(findLastVisibleItemPosition);
                }
                break;
            case SHOP_HEADER_ITEM_VIEW_TYPE://店铺头 item
                if (lastCompletelyVisibleItemPosition == findLastVisibleItemPosition) {//说明头部完全可见
                    findNSetBalanceView(findLastVisibleItemPosition);
                } else {
                    setBalanceViewVisiable(View.GONE);
                }
                break;
            case SHOP_ENDER_ITEM_VIEW_TYPE://店铺尾 item
                setBalanceViewVisiable(View.GONE);
                break;
            case OUT_DELIVERY_HEADER_ITEM_VIEW_TYPE://超出配送头
                mOutDeliveryPosition = findLastVisibleItemPosition;
                setBalanceViewVisiable(View.GONE);
                break;
            default://其他情况
                if (mOutDeliveryPosition != NOT_FOUND && findLastVisibleItemPosition > mOutDeliveryPosition) {
                    setBalanceViewVisiable(View.GONE);//超出配送范围的隐藏掉
                } else {
                    findNSetBalanceView(findLastVisibleItemPosition);
                }
                break;
        }
    }
    private void findNSetBalanceView(int findLastVisibleItemPosition) {
        int balancePosition = findBalancePositionFromPositionToEnd(findLastVisibleItemPosition);
        if (balancePosition != NOT_FOUND) {
            setBalanceViewVisiable(View.VISIBLE);
            if (balancePosition != lastBalancePosition) {//上一个和这一个相同的时候不需要重复设置
                lastBalancePosition = balancePosition;
                setBalanceViewText(balancePosition);
            }
        }
    }
    private void setBalanceViewText(int balancePosition) {
        mBalancePosition = balancePosition;
        final ShoppingCarModalResponse.CartInfoBean itemEntity = (ShoppingCarModalResponse.CartInfoBean) mCarModalList.get(balancePosition);
        isSelectedAll = (Y.equals(itemEntity.getAllSelected()));
        String selectedTotalRealAmt = itemEntity.getSelectedTotalRealAmt();
        String allCountStr = getString(R.string.all_count) + " ¥" + selectedTotalRealAmt;
```

```
            SpannableString span = new SpannableString(allCountStr);
            span = SpannableUtil.setStrColor(mContext.getString(R.string.all_count), span, R.color.gray_666666);
            span = SpannableUtil.setTargetTextSize(mContext.getString(R.string.all_count), span, 12);
            mAllCount.setText(span);
            //需要一定延迟处理，否则可能显示会出现展示问题
            handler.postDelayed(new Runnable() {
                @Override
                public void run() {
                    String fare = itemEntity.getFare();//运费
                    if (Utils.parseDouble(fare) == 0) {
                        fare = getString(R.string.free_fare);
                    } else {
                        fare = getString(R.string.fare_rmb) + fare;
                    }
                    mCountDes.setText(getString(R.string.all_discount) + itemEntity.getSelectedTotalDiscount() +
fare);
                    if (mIsEditStatue) {
                        mBalance.setText(R.string.del);
                    } else {
                        mChooseAll.setChecked(isSelectedAll);
                        mBalance.setEnabled(true);
                        mBalance.setText(getString(R.string.go_balance, itemEntity.getSelectedTotalNum()));
                    }
                }
            }, 50);
        }
        private void setBalanceViewVisiable(int visiable) {
            mDivideBottom.setVisibility(visiable);
            mBalance.setVisibility(visiable);
            mCountContainer.setVisibility(visiable);
//              mBalanceLine.setVisibility(visiable);
            if (mIsEditStatue) {
                mChooseAll.setVisibility(visiable);
                mAllCount.setVisibility(View.GONE);
                mCountDes.setVisibility(View.GONE);
            } else {
                mChooseAll.setVisibility(View.GONE);
                mAllCount.setVisibility(visiable);
                mCountDes.setVisibility(visiable);
            }
        }
        private int findBalancePositionFromPositionToEnd(int lastCompletelyVisibleItemPosition) {
            if (lastCompletelyVisibleItemPosition <= NOT_FOUND) {
                return NOT_FOUND;
```

```
            }
            for (int i = lastCompletelyVisibleItemPosition; i < mCarModalList.size(); i++) {
                if (mCarModalList.get(i).getItemType() == ShoppingCarListAdapter.SHOP_BANLANCE_ITEM_VIEW_TYPE) {
                    return i;
                }
            }
            return NOT_FOUND;
        }
        @OnClick({R.id.balance, R.id.choose_all, R.id.edit})
        public void onClick(View view) {
            switch (view.getId()) {
                case R.id.balance://结算按钮
                    if (mIsEditStatue) {
                        creatDelDialog(null, 0, true, false);
                    } else {
                        if (UserLoginInfo.getInstance().isLogin()) {
                            String selectedTotalNum = "0";
                            if (mBalancePosition != NOT_FOUND && mBalancePosition != 0 && mBalancePosition < mCarModalList.size()) {
                                selectedTotalNum = ((ShoppingCarModalResponse.CartInfoBean) mCarModalList.get(mBalancePosition)).getSelectedTotalNum();
                            }
                            if (Utils.parseInt(selectedTotalNum) > 0) {
                                startHeaderPos = 0;
                                findStartHeaderPosition(mBalancePosition);
                                mPresenter.preOrder(startHeaderPos, mBalancePosition, mCarModalList);
                            } else {
                                ToastUtil.showShortToast(R.string.no_goods_to_balance);
                            }
                        } else {
                            toAuthenticationAct();
                        }
                    }
                    break;
                case R.id.choose_all://全选
                    if (!isNotEmpty()) {
                        return;
                    }
                    isSelectedAll = mChooseAll.isChecked();
                    mChooseAll.setChecked(mChooseAll.isChecked());
                    clickChoseAll(isSelectedAll);
                    break;
                case R.id.edit://编辑
```

```
                    mIsEditStatue = !mIsEditStatue;
                    setEditView();
                    if (mIsEditStatue) {
                        mChooseAll.setChecked(false);
                        mBalance.setBackgroundResource(R.drawable.balance_del_bg);
                        mBalance.setEnabled(isDelEnable);
                        if (isNotEmpty() && mOutDeliveryPosition != 2) {//空白的时候是 1,不全是超出配送的
时候
                            mBottomView.setVisibility(View.VISIBLE);
                            setBalanceViewVisiable(View.VISIBLE);
                        } else {
                            mBottomView.setVisibility(View.GONE);
                        }
                        if (isDelEnable) {
                            mBalance.setTextColor(ActivityCompat.getColor(XApplication.getInstance(),
R.color.red_ff5050));
                        } else {
                            mBalance.setTextColor(ActivityCompat.getColor(XApplication.getInstance(),
R.color.gray_999999));
                        }
                        mBalance.setText(R.string.del);
                        mShoppingCarListAdapter.notifyDataSetChanged();
                    } else {
                        //恢复成原始状态
                        isDelEnable = false;
                        mChooseAll.setChecked(false);
                        clickChoseAll(false);
                        switchToNormolStatue();
                    }
                    break;
            }
        }
        private void switchToNormolStatue() {
            mEditSelectedMap.clear();
            lastBalancePosition = NOT_FOUND;
            mBottomView.setVisibility(View.GONE);
            mBalance.setTextColor(ActivityCompat.getColor(XApplication.getInstance(), R.color.white));
            mBalance.setBackgroundResource(R.drawable.balance_bg);
            setBalanceViewDifferentStatue();
            mShoppingCarListAdapter.notifyDataSetChanged();
        }
        private void clickChoseAll(boolean isSelectedAll) {
            for (MultiItemEntity entity : mCarModalList) {
                int itemType = entity.getItemType();
```

```
                switch (itemType) {
                    case SHOP_HEADER_ITEM_VIEW_TYPE:
                        if (entity instanceof ShoppingCarModalResponse.CartInfoBean) {
                            ((ShoppingCarModalResponse.CartInfoBean) entity).setSelected(isSelectedAll);
                            mEditSelectedMap.put(((ShoppingCarModalResponse.CartInfoBean)
entity).getShopIdSkuNo(), isSelectedAll);
                        }
                        break;
                    case NOMAL_GOODS_ITEM_VIEW_TYPE:
                    case ADDBUY_GOODS_ITEM_VIEW_TYPE:
                    case REACH_DISCOUNT_GOODS_ITEM_VIEW_TYPE:
                    case BUY_SEND_GOODS_ITEM_VIEW_TYPE:
                    case PACKAGE_GOODS_ITEM_VIEW_TYPE:
                    case GROUP_GOODS_ITEM_VIEW_TYPE:
                    case SALESOUT_GOODS_ITEM_VIEW_TYPE:
                        if (entity instanceof
ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean) {
                            ((ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean)
entity).setSeleced(isSelectedAll);

mEditSelectedMap.put(((ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean)
entity).getShopIdSkuNo(), isSelectedAll);
                        }
                        break;
                }
            }
            mShoppingCarListAdapter.notifyDataSetChanged();
            setBalanceViewCheckedAndDelEnable(isSelectedAll, isSelectedAll, true);
        }
        //点击头部跳去选择地址
        private void setHeaderViewClickListener(View headerView) {
            headerView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    if (UserLoginInfo.getInstance().isLogin()) {
                        mRefreshDelay = true;
                        Intent intent = new Intent(mContext, WebViewActivity.class);
                        intent.putExtra(WebViewActivity.EXTRA_URL, BNUrl.ADDRESS_MNG);
                        HashMap<String, Object> map = new HashMap<>();
                        map.put("addressMngEdit", "1");
                        if (!TextUtils.isEmpty(mContactId)) {
                            map.put("contactId", mContactId);
                        }
                        map.put("orderChooseAdress", "shopcart");
```

```
                    intent.putExtra(WebViewActivity.EXTRA_INIT_H5_STORE, map);
                    mBaseActivity.startActivity(intent);
                } else {
                    toAuthenticationAct();
                }
            }
        });
    }
    private void setNumberChangerListener() {
        mShoppingCarListAdapter.setNumberChangedListener(new
ShoppingCarListAdapter.NumberChangedListener() {
            @Override
            public void onNumberChangedListener(int number, final int position, int oriNum, boolean
changeNumByEdiText) {
                if (changeNumByEdiText) {//是否是 editText 输入而变化的
                    mIsInputFinish = (oriNum == 0);//判断是否跟原来的一致
                    mLastChangeNum = number;
                    mLastChangePostion = position;
                } else {
                    if (number > 0) {
                        mPresenter.onNumberChangedListener(mCarModalList, number, position);
                    } else {
                        creatDelDialog(null, position, false, false);
                    }
                }
            }
        });
    }
    private void setOnItemChildClickListener() {
        mShoppingCarListAdapter.setOnItemChildClickListener(new
BaseQuickAdapter.OnItemChildClickListener() {
            @Override
            public void onItemChildClick(BaseQuickAdapter adapter, View view, final int position) {
                switch (view.getId()) {
                    case R.id.delItem:
                        creatDelDialog(view, position, false, false);
                        break;
                    case R.id.radio_isSelected:
                        boolean checked = ((CheckBox) view).isChecked();
                        if (mIsEditStatue) {
                            ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean
goodsBean                =                (ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean)
mCarModalList.get(position);
                            goodsBean.setSeleced(checked);
```

```
                    initParam();
                    findEndHeaderPosition(position);
                    findStartHeaderPosition(position);
                    mShoppingCarListAdapter.setData(position, mCarModalList.get(position));
                    mEditSelectedMap.put(goodsBean.getShopIdSkuNo(), checked);
                    setListItemSelected(startHeaderPos, isALlSelected, isDelEnable);
                    setListItemSelected(endHeaderPos, isALlSelected, isDelEnable);
                    checkDelEnable();
                    setBalanceViewCheckedAndDelEnable(isALlSelected, isDelEnable, false);
                } else {
                    mPresenter.modifyCartItem(mCarModalList, position, checked);
                }
                break;
            case R.id.radio_all://header
                boolean isCheck = ((CheckBox) view).isChecked();
                if (mIsEditStatue) {
                    isDelEnable = isCheck;
                    changeItemsNum = 0;
                    changeSelectedFromStartToEnd(position, isCheck, true);
                    checkDelEnable();
                    setBalanceViewCheckedAndDelEnable(isCheck, isDelEnable, false);
                    mShoppingCarListAdapter.notifyItemRangeChanged(position,
changeItemsNum);
//                      mShoppingCarListAdapter.notifyDataSetChanged();
                } else {
                    if (!NetworkUtils.isConnected()) {
                        ToastUtil.showShortToast(R.string.netword_not_available);
                        return;
                    }
                    endHeaderPos = 0;
                    findEndHeaderPosition(position);
                    mPresenter.selectedAll(isCheck, mCarModalList, position, endHeaderPos);
                }
                break;
            case R.id.guideTip: {
                if (mOutDeliveryPosition != NOT_FOUND && position >= mOutDeliveryPosition) {
                    //超出配送范围
                    ToastUtil.showShortToast(R.string.out_delivery_tip);
                    return;
                }
                ShoppingCarModalResponse.CartInfoBean.PromotionBean promotionBean =
(ShoppingCarModalResponse.CartInfoBean.PromotionBean) mCarModalList.get(position);
                Intent intent = new Intent(mBaseActivity, PromotionGoodsActivity.class);
                if (PROMOTION_ADDBUY_TYPE.equals(promotionBean.getPromotionType())) {
```

```
                                PromotionGoodsActivity.setGoodsList(promotionBean.getGoodsList());
                            }
                            intent.putExtra(PromotionGoodsActivity.PROMOTION_RULE,
promotionBean.getPromotionRule());
                            intent.putExtra(PromotionGoodsActivity.PROMOTION_TYPE,
promotionBean.getPromotionType());
                            intent.putExtra(PromotionGoodsActivity.SHOP_ID, promotionBean.getShopId());
                            intent.putExtra(PromotionGoodsActivity.SHOP_TYPE,
promotionBean.getShopType());
                            mBaseActivity.startActivity(intent);
                            mRefreshDelay = false;
                        }
                        break;
                    case R.id.choose_all://endheader
                        boolean isChooseAll = ((CheckBox) view).isChecked();
                        clickEndHeaderCheckBox(position, isChooseAll);
                        break;
                    case R.id.balance:
                        if (mIsEditStatue) {
                            creatDelDialog(null, 0, true, false);
                        } else {
                            if (UserLoginInfo.getInstance().isLogin()) {
                                ShoppingCarModalResponse.CartInfoBean cartInfoBean =
(ShoppingCarModalResponse.CartInfoBean) mCarModalList.get(position);
                                if (Utils.parseInt(cartInfoBean.getSelectedTotalNum()) > 0) {
                                    startHeaderPos = 0;
                                    findStartHeaderPosition(position);
                                    mPresenter.preOrder(startHeaderPos, position, mCarModalList);
                                } else {
                                    ToastUtil.showShortToast(R.string.no_goods_to_balance);
                                }
                            } else {
                                toAuthenticationAct();
                            }
                        }
                        break;
                    case R.id.container:
                        if (!mIsEditStatue) {
                            ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean
goodsBean = (ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean)
mCarModalList.get(position);
                            if(goodsBean.getItemType()!=OUT_DELIVERY_GOODS_ITEM_VIEW_TYPE){// 超
出配送不让点击跳转
                                UrlParseUtil.openUrl(mBaseActivity, goodsBean.getDetailUrl());
```

```
                }
            }
            break;
        case R.id.go_shopping:
            if (getString(R.string.refresh_btn).equals(((TextView) view).getText().toString())) {
                loadData();
            } else {
                MainActivity.switchTab(mBaseActivity, 0);
            }
            break;
        case R.id.fare_des:
            if (getString(R.string.del_all).equals(((TextView) view).getText().toString())) {
                creatDelDialog(view, position, false, true);
            }
            break;
        case R.id.arrow: {
            if (mOutDeliveryPosition != NOT_FOUND && position >= mOutDeliveryPosition) {
                //超出配送范围
                ToastUtil.showShortToast(R.string.out_delivery_tip);
                return;
            }
            ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean goodsBean =
(ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean) mCarModalList.get(position);
            int buySendPositon = position;
            for (int i = position; i >= 0; i--) {
                int itemType = mCarModalList.get(i).getItemType();
                if (itemType == ShoppingCarListAdapter.BUY_SEND_GOODS_ITEM_VIEW_TYPE)
{//买赠的商品
                    buySendPositon = i;
                    break;
                }
            }
            Intent intent = new Intent(mBaseActivity, PromotionGoodsActivity.class);
            List<ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean>
giftGoodsList =
                    ((ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean)
mCarModalList.get(buySendPositon)).getGiftGoodsList();
            PromotionGoodsActivity.setGoodsList(giftGoodsList);
//                                    intent.putExtra(PromotionGoodsActivity.PROMOTION_RULE,
promotionBean.getPromotionRule());
            intent.putExtra(PromotionGoodsActivity.PROMOTION_TYPE,
ShoppingCarListAdapter.PROMOTION_BUY_SEND_TYPE);
            intent.putExtra(PromotionGoodsActivity.SHOP_ID, goodsBean.getShopId());
            intent.putExtra(PromotionGoodsActivity.SHOP_TYPE, goodsBean.getShopType());
```

```
                    mBaseActivity.startActivity(intent);
                }
                break;
            }
        }
    });
}
private void initParam() {
    isDelEnable = false;
    isALlSelected = true;
    startHeaderPos = 0;
    endHeaderPos = 0;
}
private void addOnScrollListener() {
    mShoppingCarRecyclerView.addOnScrollListener(new RecyclerView.OnScrollListener() {
        int allDy = 0;
        @Override
        public void onScrolled(RecyclerView recyclerView, int dx, int dy) {
            super.onScrolled(recyclerView, dx, dy);
            handlerTitle(dy);
            if (isRecyclerViewInitFinish && !mIsEditStatue && isNotEmpty()) {
                findLastVisiableItemPosition();
            }
        }
        @Override
        public void onScrollStateChanged(RecyclerView recyclerView, int newState) {
            super.onScrollStateChanged(recyclerView, newState);
            if (newState == RecyclerView.SCROLL_STATE_IDLE) {//停止的时候
                int firstCompletelyVisibleItemPosition =
mLayoutManager.findFirstCompletelyVisibleItemPosition();
                if (firstCompletelyVisibleItemPosition == 0) {
                    allDy = 0;
                    mTitle.setText(getString(R.string.shoppingcar));
                }
            }
        }
        private void handlerTitle(int dy) {
            allDy = allDy + dy;
            if (allDy < 0) {
                allDy = 0;
            }
            if (allDy < 400) {//比 locationViewHeight 多一点才处理
                if (allDy > locationViewHeight) {//头部不可见
                    mTitle.setText(mAddressStr);
```

```
                    } else {//头部可见
                        mTitle.setText(getString(R.string.shoppingcar));
                    }
                }
            }
        });
    }
    private void addKeyboardListener() {
        mKeyboardStateHelper = new KeyboardStateHelper(mRoot);
        mKeyboardListener = new SoftKeyboardStateListener() {
            @Override
            public void onSoftKeyboardOpened(int keyboardHeightInPx) {
            }
            @Override
            public void onSoftKeyboardClosed() {
                if (mIsInputFinish) {
                    mRoot.clearFocus();
                    if (mLastChangePostion < 0 || mLastChangePostion >= mCarModalList.size()) {
                        return;
                    }
                    ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean item =
                            (ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean)
mCarModalList.get(mLastChangePostion);
                    if (Utils.parseInt(item.getNum()) != mLastChangeNum) {//不相等
                        mPresenter.onNumberChangedListener(mCarModalList,        mLastChangeNum,
mLastChangePostion);
                    }
                    mIsInputFinish = false;
                } else {
                    mRoot.clearFocus();
                }
            }
        };
        mKeyboardStateHelper.addSoftKeyboardStateListener(mKeyboardListener);
    }
    private void toAuthenticationAct() {
        Intent intent = new Intent(mBaseActivity, AuthenticationActivity.class);
        mBaseActivity.startActivity(intent);
    }
    //多店的情况需要
    private void checkDelEnable() {
        if (!isDelEnable) {
            for (ShoppingCarModalResponse.CartInfoBean cartInfoBean : mShopList) {//查看所有的店头是否
选中
```

```
                if (cartInfoBean.isDelEnable()) {
                    isDelEnable = true;
                    break;
                }
            }
        }
    }
    //弹出删除的 dialog
    private void creatDelDialog(final View delView, final int position, final boolean isBalanceDelItems, final boolean delOutDelivery) {
        if (mDelTipDialog == null) {
            mDelTipDialog = new NormalDialog(mContext);
            mDelTipDialog.show();
            mDelTipDialog.setCancelColor(R.color.gray_666666)
                    .setConfirmText(R.string.del)
                    .setNegativeListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            if (mDelTipDialog != null && mDelTipDialog.isShowing()) {
                                mDelTipDialog.dismiss();
                            }
                            closeDelView(delView);
                        }
                    });
        }
        if (!mDelTipDialog.isShowing()) {
            mDelTipDialog.show();
        }
        String tips=getString(R.string.confirm_to_del_goods);
        if(delOutDelivery){
            tips=getString(R.string.confirm_to_del_all_goods);
        }
        mDelTipDialog.setContent(tips);
        mDelTipDialog.setPositiveListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (mDelTipDialog != null && mDelTipDialog.isShowing()) {
                    mDelTipDialog.dismiss();
                }
                closeDelView(delView);
                if (isBalanceDelItems) {
                    mPresenter.delItemsInEditStatue(mCarModalList);
                } else if (delOutDelivery) {
                    mDelPosition = position;
```

```
                        mPresenter.delOutDelivery(position, mCarModalList);
                    } else {
                        mDelPosition = position;
                        mPresenter.delCarItem(mCarModalList, position);
                    }
                }
            });
        }
        private void closeDelView(View delView) {
            if (delView != null) {
                ViewParent parent = delView.getParent();
                if (parent instanceof SwipeMenuLayout) {
                    ((SwipeMenuLayout) parent).quickClose();
                }
            }
        }
        private void clickEndHeaderCheckBox(int position, boolean isChooseAll) {
            if (mIsEditStatue) {
                changeSelectedFromEndToStart(position, isChooseAll, false);
                setBalanceViewCheckedAndDelEnable(isChooseAll, isChooseAll, false);
                mShoppingCarListAdapter.notifyDataSetChanged();
            } else {
                startHeaderPos = 0;
                findStartHeaderPosition(position);
                mPresenter.selectedAll(isChooseAll, mCarModalList, startHeaderPos, position);
            }
        }
        private void findStartHeaderPosition(int position) {
            for (int i = position - 1; i >= 0; i--) {
                if (findHeadersPosition(i, false)) break;//结束
            }
        }
        private void findEndHeaderPosition(int position) {
            for (int i = position; i < mCarModalList.size(); i++) {
                if (findHeadersPosition(i, true)) break;
            }
        }
        private void setBalanceViewCheckedAndDelEnable(boolean isALlSelected, boolean isDelEnable, boolean
isClickAllChoseCheckBox) {
            if (isClickAllChoseCheckBox) {//直接点击全选按钮
                mChooseAll.setChecked(isALlSelected);
            } else {
                boolean isAllChose = true;
                if (isALlSelected) {
```

```
                    if (isEmptyShop()) {
                        isAllChose = false;
                    } else {
                        for (ShoppingCarModalResponse.CartInfoBean cartInfoBean : mShopList) {//查看所有的店头是否选中
                            if (!cartInfoBean.isSelected()) {
                                isAllChose = false;
                                break;
                            }
                        }
                    }
                } else {
                    isAllChose = false;
                }
                mChooseAll.setChecked(isAllChose);
            }
            setBalanceDelAble(isDelEnable);
        }
        private void setBalanceDelAble(boolean isDelEnable) {
            if (isDelEnable) {
                mBalance.setTextColor(ActivityCompat.getColor(XApplication.getInstance(), R.color.red_ff5050));
            } else {
                mBalance.setTextColor(ActivityCompat.getColor(XApplication.getInstance(), R.color.gray_999999));
            }
            mBalance.setEnabled(isDelEnable);
        }
        //设置 item 选中的数据状态
        private void setListItemSelected(int position, boolean isSelected, boolean isDelEnable) {
            MultiItemEntity itemEntity = mCarModalList.get(position);
            if (itemEntity instanceof ShoppingCarModalResponse.CartInfoBean) {//头尾
                ((ShoppingCarModalResponse.CartInfoBean) itemEntity).setDelEnable(isDelEnable);
                ((ShoppingCarModalResponse.CartInfoBean) itemEntity).setSelected(isSelected);
                mEditSelectedMap.put(((ShoppingCarModalResponse.CartInfoBean) itemEntity).getShopIdSkuNo(), isSelected);
            } else if (itemEntity instanceof ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean) {
                ((ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean) itemEntity).setSeleced(isSelected);
                mEditSelectedMap.put(((ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean) itemEntity).getShopIdSkuNo(), isSelected);
            }
            mShoppingCarListAdapter.setData(position, itemEntity);
        }
        /**
         * 寻找头部 item 的位置所在
```

```
     *
     * @param i                目前所处的位置
     * @param endHeaderBreak  遇到尾部 item 退出
     * @return
     */
    private boolean findHeadersPosition(int i, boolean endHeaderBreak) {
        MultiItemEntity itemEntity = mCarModalList.get(i);
        int itemType = itemEntity.getItemType();
        if (itemEntity instanceof ShoppingCarModalResponse.CartInfoBean) {//头尾
            if (itemType == ShoppingCarListAdapter.SHOP_HEADER_ITEM_VIEW_TYPE) {
                startHeaderPos = i;
                if (!endHeaderBreak) {
                    return true;
                }
            } else if (itemType == ShoppingCarListAdapter.SHOP_BANLANCE_ITEM_VIEW_TYPE) {
                endHeaderPos = i;
                if (endHeaderBreak) {
                    return true;
                }
            }
        } else if (itemEntity instanceof ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean) {
            if (itemType != GIFT_GOODS_ITEM_VIEW_TYPE && itemType != PACKAGE_GOODS_ITEM_VIEW_TYPE
                    && itemType != PACKAGE_GOODS_HEADER_ITEM_VIEW_TYPE) {
                boolean seleced = ((ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean) itemEntity).isSeleced();
                if (seleced) {
                    isDelEnable = true;
                } else {
                    isALlSelected = false;
                }
            } else if (itemType == PACKAGE_GOODS_ITEM_VIEW_TYPE
                    && ((ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean) itemEntity).isPromotionFirstItem()) {
                boolean seleced = ((ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean) itemEntity).isSeleced();
                if (seleced) {
                    isDelEnable = true;
                } else {
                    isALlSelected = false;
                }
            }
        }
        return false;
```

```
    }
    /**
     * 从头到尾更改 item 的选中状态
     *
     * @param position          位置
     * @param isChooseAll       是否全选
     * @param endHeaderBreak  遇到尾部退出
     */
    private void changeSelectedFromStartToEnd(int position, boolean isChooseAll,
                                              boolean endHeaderBreak) {
        for (int i = position; i < mCarModalList.size(); i++) {
            changeItemsNum++;
            if (setOneByOne(isChooseAll, endHeaderBreak, i)) break;//结束
        }
    }
    /**
     * 从尾到头更改 item 的选中状态
     *
     * @param position          位置
     * @param isChooseAll       是否全选
     * @param endHeaderBreak  遇到尾部退出
     */
    private void changeSelectedFromEndToStart(int position, boolean isChooseAll,
                                              boolean endHeaderBreak) {
        for (int i = position; i >= 0; i--) {
            if (setOneByOne(isChooseAll, endHeaderBreak, i)) break;//结束
        }
    }
    private boolean setOneByOne(boolean isChooseAll, boolean endHeaderBreak, int i) {
        MultiItemEntity itemEntity = mCarModalList.get(i);
        if (itemEntity instanceof ShoppingCarModalResponse.CartInfoBean) {//头尾
            if (itemEntity.getItemType() == ShoppingCarListAdapter.SHOP_HEADER_ITEM_VIEW_TYPE) {
                ((ShoppingCarModalResponse.CartInfoBean) itemEntity).setSelected(isChooseAll);
                ((ShoppingCarModalResponse.CartInfoBean) itemEntity).setDelEnable(isChooseAll);
                mEditSelectedMap.put(((ShoppingCarModalResponse.CartInfoBean)
itemEntity).getShopIdSkuNo(), isChooseAll);
                return !endHeaderBreak;
            } else if (itemEntity.getItemType() == ShoppingCarListAdapter.SHOP_BANLANCE_ITEM_VIEW_TYPE)
{
                ((ShoppingCarModalResponse.CartInfoBean) itemEntity).setSelected(isChooseAll);
                ((ShoppingCarModalResponse.CartInfoBean) itemEntity).setDelEnable(isChooseAll);
                mEditSelectedMap.put(((ShoppingCarModalResponse.CartInfoBean)
itemEntity).getShopIdSkuNo(), isChooseAll);
                return endHeaderBreak;
```

```
                }
            } else if (itemEntity instanceof ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean) {
                ((ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean)
itemEntity).setSeleced(isChooseAll);
                mEditSelectedMap.put(((ShoppingCarModalResponse.CartInfoBean.PromotionBean.GoodsBean)
itemEntity).getShopIdSkuNo(), isChooseAll);
            }
            return false;
        }
        private void loadData() {
            mPresenter.loadData(true);
        }
        OnHeaderClickListener clickAdapter = new OnHeaderClickListener() {
            @Override
            public void onHeaderClick(View view, int id, int position) {
                switch (id) {
                    case R.id.radio_all:
                        view.performClick();
                        mShoppingCarRecyclerView.invalidateItemDecorations();//刷新头部
                        break;
                    case R.id.fare_des:
                        view.performClick();
                        mShoppingCarRecyclerView.invalidateItemDecorations();//刷新头部
                        break;
                }
            }
            @Override
            public void onHeaderLongClick(View view, int id, int position) {
            }
            @Override
            public void onHeaderDoubleClick(View view, int id, int position) {
            }
        };
        @Override
        public void loadDataFinish(List<MultiItemEntity> multiItemEntityList) {
            mCarModalList.clear();
//              if (mSmartRefreshLayout != null) {
//                  mSmartRefreshLayout.finishRefresh();
//              }
            mShoppingCarListAdapter.addData(multiItemEntityList);
            startToSetBalanceView();
            setEditView();
        }
        private void setEditView() {
```

```
            if (mIsEditStatue) {
                mEdit.setText(R.string.done);
                mEdit.setVisibility(View.VISIBLE);
            } else {
                if (isNotEmpty()) {
                    mEdit.setText(R.string.edit);
                    mEdit.setVisibility(View.VISIBLE);
                } else {
                    mEdit.setVisibility(View.GONE);
                }
            }
        }
        private boolean isNotEmpty() {//是否不为空
            return mCarModalList.size() > 1;//empty 时有 1，空布局
        }
        @Override
        public void allSelected(boolean allSelected) {
            isSelectedAll = allSelected;
        }
        @Override
        public void loadDataError(String message) {//加载数据错误
            if (!TextUtils.isEmpty(message)) {
                ToastUtil.showShortToast(message);
            }
//              if (mSmartRefreshLayout != null && mSmartRefreshLayout.isRefreshing()) {
//                  mSmartRefreshLayout.finishRefresh(false);
//              }
        }
        @Override
        public void setShopList(ArrayList<ShoppingCarModalResponse.CartInfoBean> shopList) {
            mShopList = shopList;
        }
        //设置 个人联系信息
        @Override
        public void setContactInfo(String address, String name, String phone, String contactId, String latitude, String
longitude, String adCode) {//设置联系信息
            mContactId = contactId;
            if (TextUtils.isEmpty(address)) {
                mHeaderBottomView.setVisibility(View.GONE);
                mTopView.setVisibility(View.GONE);
                if (LocationHelper.locationModel != null) {
                    mAddressStr = getString(R.string.delivery_to) + LocationHelper.locationModel.street;
                }
                mNameTv.setVisibility(View.GONE);
```

```
                mAdress.setText(mAddressStr);
            } else {
                mHeaderBottomView.setVisibility(View.VISIBLE);
                mTopView.setVisibility(View.VISIBLE);
                mAddressStr = getString(R.string.delivery_to) + address;
                mNameTv.setVisibility(View.VISIBLE);
                mNameTv.setText(name + " " + phone);
                mAdress.setText(address);
            }
            notifyChangeLocation(address, latitude, longitude, contactId, adCode);
        }
        //广播通知地理位置变化
        private void notifyChangeLocation(String address, String latitude, String longitude, String contactId, String adCode) {
            if (!TextUtils.isEmpty(latitude)) {
                // 1、定位没成功 2、选择的位置与之前的位置的经纬度不一致（位置发生变化），都需要发送位置变化广播
                if (LocationHelper.locationModel == null ||
                        (!latitude.equals(LocationHelper.locationModel.latitude)
                                || !longitude.equals(LocationHelper.locationModel.longitude))) {
                    LocationHelper.locationModel = new LocationModel();
                    LocationHelper.locationModel.adcode = adCode;
                    LocationHelper.locationModel.latitude = latitude;
                    LocationHelper.locationModel.longitude = longitude;
                    LocationHelper.locationModel.street = address;
                    LocationHelper.locationModel.contactId = contactId;
                    LocationHelper.locationState = LocationHelper.LOCATION_STATE_SUCCESS;
                    LocalBroadcastManager.getInstance(XApplication.getInstance().getApplicationContext())
                            .sendBroadcast(new Intent(Constants.ACTION_LOCATION_STATE_CHANGED));
                }
            }
        }
        @Override
        public void setOutDeliveryPosition(int outDeliveryPosition) {
            mOutDeliveryPosition = outDeliveryPosition;
            lastBalancePosition = NOT_FOUND;
        }
        @Override
        public void modifyCartItemError(int position) {//更改数据错误
            if (position < 0 || position >= mCarModalList.size()) {
                return;
            }
            mShoppingCarListAdapter.notifyItemChanged(position);//刷回原来的数据
        }
```

```
//编辑状态下  item 选中的 map
@Override
public Map<String, Boolean> getEditSelectedMap() {
    return mEditSelectedMap;
}
@Override
public void loadDataActionType(int actionType) {
    if (mIsEditStatue) {//编辑状态下删除商品后的状态处理
        switch (actionType) {
            case ShoppingCarPresenter.EDIT_MUTI_DEL_ACTIONTYPE://编辑状态下  多项删除之后
                isDelEnable = false;
                setBalanceDelAble(isDelEnable);
                break;
            case ShoppingCarPresenter.NORMAL_DEL_ACTIONTYPE://正常删除
                initParam();
                //检查越界问题
                if (mCarModalList.size() > mDelPosition) {
                    findEndHeaderPosition(mDelPosition);
                    findStartHeaderPosition(mDelPosition);
                }
                if (mCarModalList.size() > startHeaderPos) {
                    setListItemSelected(startHeaderPos, isALlSelected, isDelEnable);
                    setListItemSelected(endHeaderPos, isALlSelected, isDelEnable);
                }
                checkDelEnable();
                setBalanceViewCheckedAndDelEnable(isALlSelected, isDelEnable, false);
                break;
        }
    }
}
//是否展示网络错误相关的 view
@Override
public boolean isShowNetworkErrorView() {
    int size = mCarModalList.size();
    if (mCarModalList.size() == 1) {
        MultiItemEntity itemEntity = mCarModalList.get(0);
        if (itemEntity != null) {
            int itemType = itemEntity.getItemType();
            if (itemType == EMPTY_NETWORK_ERROR_ITEM_VIEW_TYPE
                    || itemType == EMPTY_OTHER_ERROR_ITEM_VIEW_TYPE) {
                size = 0;
            }
        }
    }
```

```
            return size == 0;
        }
        @Override
        public void loadDataErrorView(List<MultiItemEntity> entityList) {
            mCarModalList.clear();
            mShoppingCarListAdapter.addData(entityList);
        }
        private boolean isEmptyShop() {//是否没有店铺
            return mShopList == null || mShopList.size() == 0;
        }
}
package com.baoneng.bnmall.ui.member;
import android.content.Intent;
import android.graphics.Typeface;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.text.TextUtils;
import android.view.View;
import android.view.ViewGroup;
import android.widget.RelativeLayout;
import android.widget.TextView;
import com.baoneng.bnmall.R;
import com.baoneng.bnmall.common.Constants;
import com.baoneng.bnmall.contract.member.MemberContract;
import com.baoneng.bnmall.model.UserLoginInfo;
import com.baoneng.bnmall.model.order.RespOrderCountModel;
import com.baoneng.bnmall.model.shoppingcar.CouponModel;
import com.baoneng.bnmall.network.BNUrl;
import com.baoneng.bnmall.presenter.member.MemberPresenter;
import com.baoneng.bnmall.ui.BaseFragment;
import com.baoneng.bnmall.ui.WebViewActivity;
import com.baoneng.bnmall.utils.ImageLoaderUtils;
import com.baoneng.bnmall.utils.RxBus;
import com.baoneng.bnmall.utils.SpannableUtil;
import com.baoneng.bnmall.utils.Utils;
import com.baoneng.bnmall.utils.ViewUtils;
import com.baoneng.bnmall.widget.CommonCornerView;
import com.baoneng.bnmall.widget.SettingItemView;
import java.util.Arrays;
import java.util.List;
import butterknife.BindView;
import butterknife.OnClick;
import de.hdodenhof.circleimageview.CircleImageView;
import io.reactivex.functions.Consumer;
```

```
import static com.baoneng.bnmall.ui.WebViewActivity.EXTRA_TITLE;
import static com.baoneng.bnmall.ui.WebViewActivity.EXTRA_URL;
/**
 * Created by lenovo on 2018/2/5.
 * 订单相关，优惠券，地址管理都是 H5
 * 门店列表可以复用
 * 设置 基本完了
 * 如果用户信息更改用 rxbus 通知
 * <p>
 */
public class MemberCenterFragment extends BaseFragment<MemberContract.Presenter> implements MemberContract.View {
    //优惠券列表
    @BindView(R.id.sivUsrCoupon)
    SettingItemView sivUsrCoupon;
    //待付款
    @BindView(R.id.tvInPendingPayment)
    CommonCornerView tvInPendingPayment;
    //待配送
    @BindView(R.id.tvInPendingDelivery)
    CommonCornerView tvInPendingDelivery;
    //待收货
    @BindView(R.id.tvInDispatched)
    CommonCornerView tvDispatched;
    //待自提
    @BindView(R.id.tvInPendingTakeDelivery)
    CommonCornerView tvInPendingTakeDelivery;
    //消费记录
    @BindView(R.id.sivPurchaseHistory)
    SettingItemView sivPurchaseHistory;
    //用户昵称
    @BindView(R.id.user_name)
    TextView userName;
    //菁选会员昵称
    @BindView(R.id.premiumName)
    TextView premiumName;
    //去开通菁选会员
    @BindView(R.id.premium_member)
    TextView mPremiumMember;
    // 马上登陆
    @BindView(R.id.user_login)
    TextView userLogin;
    @BindView(R.id.membership_tip)
    TextView mTip;
```

```
//普通会员头像
@BindView(R.id.ivUserPhoto)
CircleImageView ivUserPhoto;
//菁选会员头像
@BindView(R.id.ivPremiumHead)
CircleImageView ivPremiumHead;
//普通会员布局
@BindView(R.id.normal_member_layout)
RelativeLayout normalLayout;
//菁选会员布局
@BindView(R.id.premium_member_layout)
ViewGroup premiumLayout;
//账户余额
@BindView(R.id.user_blance)
TextView premiumBlance;
//菁选会员优惠券
@BindView(R.id.premium_coupon)
TextView premiumCouponNum;
private boolean change = true;
@Override
public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
    mPresenter = new MemberPresenter(this);
    if (UserLoginInfo.getInstance().isLogin()) {
        mPresenter.queryUserInfo();
    } else {
        showUserinfo(false);
    }
}
@Override
public void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    RxBus rxBus = RxBus.getIntanceBus();
    registerRxBus(NotifyUserInfo.class, new Consumer<NotifyUserInfo>() {
        @Override
        public void accept(NotifyUserInfo notifyUserInfo) throws Exception {
            if (notifyUserInfo.needUpdate) {
                if (mPresenter != null) {
                    mPresenter.queryUserInfo();
                }
            } else {
                showUserinfo(UserLoginInfo.getInstance().isLogin());
            }
        }
    }, rxBus);
```

```
}
@Override
public void onHiddenChanged(boolean hidden) {
    super.onHiddenChanged(hidden);
    if (mInitFinished && !hidden) {
        if (UserLoginInfo.getInstance().isLogin()) {
            mPresenter.getCouponDetail("3");
            mPresenter.getOrderDetail();
        }
        change = true;
    } else {
        change = false;
    }
}
@Override
public void onResume() {
    super.onResume();
    if (mInitFinished && change) {
        if (UserLoginInfo.getInstance().isLogin()) {
            mPresenter.getCouponDetail("3");
            mPresenter.getOrderDetail();
        }
    }
}
@Override
public void showCouponInfo(List<CouponModel> coupons) {
    if (coupons.isEmpty()) {
        if (premiumLayout.getVisibility() == View.VISIBLE) {
            premiumCouponNum.setText("0");
        } else {
            sivUsrCoupon.setRightText("");
        }
    } else {
        String str = getString(R.string.coupon_count, coupons.size());
        if (premiumLayout.getVisibility() == View.VISIBLE) {
            premiumCouponNum.setText(str);
        } else {
            sivUsrCoupon.setRightText(str);
        }
    }
}
@Override
public void updateUserInfo() {
    showUserinfo(true);
```

```
    }
    @OnClick({R.id.ivUserPhoto, R.id.user_login, R.id.check_all_order
            , R.id.sivUsrCoupon, R.id.sivStoreList, R.id.sivAddressManager, R.id.sivSetting,
R.id.tvInPendingPayment
            , R.id.tvInPendingDelivery, R.id.tvInDispatched, R.id.tvInPendingTakeDelivery,
            R.id.sivMemberBenefits, R.id.sivPurchaseHistory, R.id.premium_code, R.id.premium_member,
R.id.premiumName,
            R.id.ivPremiumHead, R.id.user_blance, R.id.user_blance_title, R.id.premium_coupon,
R.id.premium_coupon_title})
    public final void clickEvent(View view) {
        boolean isLogin = UserLoginInfo.getInstance().isLogin();
        Intent intent = null;
        switch (view.getId()) {
            case R.id.ivUserPhoto:
            case R.id.ivPremiumHead:
            case R.id.premiumName:
            case R.id.user_login:
                if (isLogin) {
                    intent = new Intent(mContext, AccountSettingActivity.class).putExtra("action",
Constants.USER_SETTING);
                } else {
                    intent = getLoginIntent();
                }
                break;
            case R.id.user_blance:
            case R.id.user_blance_title:    // 余额
                break;
            case R.id.check_all_order: //查看全部订单
                toWebViewAct(mContext, BNUrl.ALL_ORDER);
                break;
            case R.id.sivPurchaseHistory: // 消费记录
                break;
            case R.id.sivUsrCoupon:   //优惠券
            case R.id.premium_coupon:
            case R.id.premium_coupon_title:
                toWebViewAct(mContext, BNUrl.COUPON_URL);
                break;
            case R.id.sivStoreList: //门店列表 是否需要登录
                intent = new Intent(mContext, UserRelativeActivity.class).putExtra("action",
Constants.SHOP_LIST);
                break;
            case R.id.sivAddressManager:   // 地址管理
                toWebViewAct(mContext, BNUrl.ADDRESS_MNG);
                break;
```

```
                case R.id.sivSetting: //设置
                    intent = new Intent(mContext, UserRelativeActivity.class).putExtra("action",
Constants.SETTING);
                    break;
                case R.id.premium_member: //成为会员
                    break;
                case R.id.tvInPendingPayment: //待支付
                    toWebViewAct(mContext, BNUrl.WAIT_PAY);
                    break;
                case R.id.tvInPendingDelivery:  //待配送
                    toWebViewAct(mContext, BNUrl.WAITING_DELIVERY);
                    break;
                case R.id.tvInDispatched:  //待收货
                    toWebViewAct(mContext, BNUrl.WAIT_RECEIVE);
                    break;
                case R.id.tvInPendingTakeDelivery:  //待自提
                    toWebViewAct(mContext, BNUrl.WAIT_PICK);
                    break;
                case R.id.sivMemberBenefits:
                    intent = new Intent(mContext, WebViewActivity.class).putExtra(EXTRA_URL,
BNUrl.MEMBER_URL).putExtra(EXTRA_TITLE, "会员权益");
                    break;
                case R.id.premium_code:  //菁选会员 二维码
                    // TODO: 2018/3/24
                    break;
            }
            if (intent != null) {
                startActivity(intent);
            }
        }
        /**
         * 显示用户信息
         *
         * @param login
         */
        private void showUserinfo(boolean login) {
            UserLoginInfo loginInfo = UserLoginInfo.getInstance();
            String name = TextUtils.isEmpty(loginInfo.getCustName()) ? "菁选用户" : loginInfo.getCustName();
            boolean vip = loginInfo.isVip();
            ViewUtils.setVisable(normalLayout, !vip);
            ViewUtils.setVisable(premiumLayout, vip);
            ImageLoaderUtils.display(mContext, ivPremiumHead, loginInfo.getImageUrl(), R.drawable.pic_head,
R.drawable.pic_head);
            ImageLoaderUtils.display(mContext, ivUserPhoto, loginInfo.getImageUrl(), R.drawable.pic_head,
```

```
R.drawable.pic_head);
            premiumName.setText(name);
            userName.setText(name);
            if (vip) {    //vip
                ViewUtils.setVisable(sivPurchaseHistory, true);
                ViewUtils.setVisable(sivUsrCoupon, false);
                SpannableUtil.setTargetTextBoldSize(premiumBlance, "元", "0.00 元", 10, Typeface.NORMAL);
            } else {
                ViewUtils.setVisable(Arrays.asList(userName, mTip, mPremiumMember), login);
                ViewUtils.setVisable(userLogin, !login);
                if (!login) {
                    ivUserPhoto.setImageResource(R.drawable.pic_head);
                    sivUsrCoupon.setRightText("");
                    tvInPendingPayment.setCount(0);
                    tvInPendingTakeDelivery.setCount(0);
                    tvDispatched.setCount(0);
                    tvInPendingDelivery.setCount(0);
                }
            }
        }
        @Override
        public void showOrderInfo(List<RespOrderCountModel.OrderItem> items) {
            for (RespOrderCountModel.OrderItem item : items) {
                switch (item.getOrderStatus()) {
                    case "WAIT_PAY":
                        tvInPendingPayment.setCount(Utils.parseInt(item.getCount()));
                        break;
                    case "WAIT_DELIVERED":
                        tvInPendingDelivery.setCount(Utils.parseInt(item.getCount()));
                        break;
                    case "WAIT_SELF_RECEIPT":
                        tvInPendingTakeDelivery.setCount(Utils.parseInt(item.getCount()));
                        break;
                    case "WAIT_RECEIPT":
                        tvDispatched.setCount(Utils.parseInt(item.getCount()));
                        break;
                }
            }
        }
        @Override
        public int getContentViewId() {
            return R.layout.fragment_member_center;
        }
}
```

```
package com.baoneng.bnmall.ui.member;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.text.TextUtils;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import com.baoneng.barcodescanner.Util;
import com.baoneng.bnmall.R;
import com.baoneng.bnmall.common.Constants;
import com.baoneng.bnmall.model.UserLoginInfo;
import com.baoneng.bnmall.network.BNUrl;
import com.baoneng.bnmall.ui.BaseFragment;
import com.baoneng.bnmall.ui.WebViewActivity;
import com.baoneng.bnmall.utils.ImageLoaderUtils;
import com.baoneng.bnmall.utils.RxBus;
import com.baoneng.bnmall.utils.Utils;
import butterknife.BindView;
import butterknife.OnClick;
import de.hdodenhof.circleimageview.CircleImageView;
import io.reactivex.functions.Consumer;
import static com.baoneng.bnmall.ui.BaseActivity.STYLE_STATUS_BAR_WHITE;
import static com.baoneng.bnmall.ui.WebViewActivity.EXTRA_URL;
/**
 *  账户设置
 *
 */
//TODO 修改头像之后需要一个通知来更新
public class UserSettingFragment extends BaseFragment{
    @BindView(R.id.img_update_info)
    ImageView imgUpdateInfo ;
    @BindView(R.id.ivUserPhoto)
    CircleImageView ivUserPhoto ;
    @BindView(R.id.img_user_club)
    TextView imgUserClub;
    @BindView(R.id.user_login)
    TextView userName ;
    @BindView(R.id.user_moblie)
    TextView userMobile ;
    private long lastPressedTime = 0;
    @Override
    public int getContentViewId() {
        return R.layout.fragment_user_setting;
```

```
    }
    public static UserSettingFragment newInstance() {
        UserSettingFragment fragment = new UserSettingFragment();
        return fragment;
    }
    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        RxBus rxBus = RxBus.getIntanceBus();
        registerRxBus(NotifyUserInfo.class, new Consumer<NotifyUserInfo>() {
            @Override
            public void accept(NotifyUserInfo notifyUserInfo) throws Exception {
                showUserinfo();
            }
        },rxBus);
    }
    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        showUserinfo();
    }
    private void showUserinfo() {
        imgUpdateInfo.setVisibility(View.VISIBLE);
        imgUserClub.setVisibility(View.GONE);
        String name = TextUtils.isEmpty(UserLoginInfo.getInstance().getCustName()) ? "菁选用户" :
UserLoginInfo.getInstance().getCustName();
        userName.setText(name);
        userMobile.setText(UserLoginInfo.getInstance().getMobilePhone());
        ImageLoaderUtils.display(mContext,ivUserPhoto,
UserLoginInfo.getInstance().getImageUrl(),R.drawable.pic_head,R.drawable.pic_head);
    }
    @OnClick({R.id.user_root,R.id.siv_add_manager,R.id.siv_account_safe,R.id.setting})
    public void clickEvent(View v)
    {
        if (Utils.isFastClick()){
            return;
        }
        switch (v.getId()){
            case R.id.user_root:
                listener.replace(Constants.UPDATE_USER_INFO,Constants.USER_SETTING);
                break;
            case R.id.siv_add_manager:
                startActivity(new Intent(mContext, WebViewActivity.class).putExtra(EXTRA_URL,
BNUrl.ADDRESS_MNG));
```

```
                    break;
                case R.id.siv_account_safe:
                    listener.replace(Constants.ACCOUNT_SAFE,Constants.USER_SETTING);
                    break;
                case R.id.setting:
                    startActivity(new Intent(mContext,UserRelativeActivity.class).putExtra("action",
Constants.SETTING));
                        break;
            }
        }
        @Override
        public String getStatusBarStyle() {
            return STYLE_STATUS_BAR_WHITE;
        }
}
package com.baoneng.bnmall.ui.member;
import android.content.Intent;
import android.graphics.Color;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.text.TextUtils;
import android.view.View;
import com.baoneng.bnmall.R;
import com.baoneng.bnmall.common.Constants;
import com.baoneng.bnmall.contract.member.UpdateUserInfoContract;
import com.baoneng.bnmall.model.UserLoginInfo;
import com.baoneng.bnmall.presenter.member.UpdateUserInfoPresenter;
import com.baoneng.bnmall.ui.BaseFragment;
import com.baoneng.bnmall.utils.ImageLoaderUtils;
import com.baoneng.bnmall.utils.RxBus;
import com.baoneng.bnmall.utils.picture.PhotoPicker;
import com.baoneng.bnmall.widget.SettingItemView;
import com.bigkoo.pickerview.OptionsPickerView;
import com.bigkoo.pickerview.TimePickerView;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import butterknife.BindView;
import butterknife.OnClick;
```

```
import de.hdodenhof.circleimageview.CircleImageView;
import io.reactivex.functions.Consumer;
import static com.baoneng.bnmall.ui.BaseActivity.STYLE_STATUS_BAR_WHITE;
/**
 * 修改用户资料
 */
public class UpdateUserInfoFragment extends BaseFragment<UpdateUserInfoPresenter> implements
UpdateUserInfoContract.View, PhotoPicker.Listener {
    private PhotoPicker mPhotoPicker;
    private TimePickerView pvTime;
    @BindView(R.id.ivUserPhoto)
    CircleImageView ivUserPhoto;
    @BindView(R.id.siv_update_nickname)
    SettingItemView siv_nickname;
    @BindView(R.id.siv_update_gender)
    SettingItemView siv_gender ;
    @BindView(R.id.siv_update_birthday)
    SettingItemView siv_birthday ;
    private List<String> sexItem = Arrays.asList("女","男");
    private OptionsPickerView pvNoLinkOptions;
    @Override
    public int getContentViewId() {
        return R.layout.fragment_update_userinfo;
    }
    public static UpdateUserInfoFragment newInstance() {
        UpdateUserInfoFragment fragment = new UpdateUserInfoFragment();
        return fragment;
    }
    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        showUserinfo();
        mPresenter = new UpdateUserInfoPresenter(this);
        mPhotoPicker = new PhotoPicker(this, savedInstanceState, this)
                .setMaxSize(Constants.PhotoSize.PORTRAIT, Constants.PhotoSize.PORTRAIT)
                .setCrop(PhotoPicker.CropType.CROP_SQUARE);
        initTimePicker();
        initNoLinkOptionsPicker();
    }
    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        RxBus rxBus = RxBus.getIntanceBus();
        registerRxBus(NotifyUserInfo.class, new Consumer<NotifyUserInfo>() {
```

```
            @Override
            public void accept(NotifyUserInfo notifyUserInfo) throws Exception {
                showUserinfo();
            }
        },rxBus);
    }
    private void showUserinfo() {
        UserLoginInfo loginInfo = UserLoginInfo.getInstance();
        siv_nickname.setRightText(loginInfo.getCustName());
        siv_gender.setRightText(loginInfo.getSex());
        siv_birthday.setRightText(loginInfo.getBirthDay());
        ImageLoaderUtils.display(mContext,ivUserPhoto, UserLoginInfo.getInstance().getImageUrl());
    }
    //如果设置了就是设置到-当前，如果没有就是默认当前
    private void initTimePicker() {
        //因为系统 Calendar 的月份是从 0-11 的,所以如果是调用 Calendar 的 set 方法来设置时间,月份的范围也要是从 0-11
        Calendar selectedDate = Calendar.getInstance();
        String birthday = UserLoginInfo.getInstance().getBirthDay();
        if (!TextUtils.isEmpty(birthday)){
          SimpleDateFormat sdf = new SimpleDateFormat(Constants.TRANS_DATE_BIRTHDAY_FORMAT);
            try {
                Date date = sdf.parse(birthday);
                selectedDate.setTime(date);
            } catch (ParseException e) {
                e.printStackTrace();
            }
        }
        Calendar endDate = Calendar.getInstance();
        Calendar startDate = Calendar.getInstance();
        startDate.set(1900, 0, 1);
        //时间选择器
        pvTime = new TimePickerView.Builder(mContext, new TimePickerView.OnTimeSelectListener() {
            @Override
            public void onTimeSelect(Date date, View v) {//选中事件回调
                // 这里回调过来的 v,就是 show()方法里面所添加的 View 参数，如果 show 的时候没有添加参数，v 则为 null
//                ToastUtil.showToast();

mPresenter.updateInfo(UserLoginInfo.getInstance().getCustName(),UserLoginInfo.getInstance().getRealSex()
                ,getTime(date));
            }
        })
                //年月日时分秒 的显示与否，不设置则默认全部显示
```

```
                .setType(new boolean[]{true, true, true, false, false, false})
                .setLabel("", "", "", "", "", "")
                .isCenterLabel(false)
                .setDividerColor(Color.DKGRAY)
                .setContentSize(21)
                .setDate(selectedDate)
                .setRangDate(startDate, endDate)
//                .setBackgroundId(Color.BLUE) //设置外部遮罩颜色
                .setDecorView(null)
                .build();
    }
    private void initNoLinkOptionsPicker() {// 不联动的多级选项
        pvNoLinkOptions = new OptionsPickerView.Builder(mContext, new
OptionsPickerView.OnOptionsSelectListener() {
            @Override
            public void onOptionsSelect(int options1, int options2, int options3, View v) {
                String sex = TextUtils.equals(sexItem.get(options2),"女") ? "0" : "1";
                mPresenter.updateInfo(UserLoginInfo.getInstance().getCustName(),sex
                        ,UserLoginInfo.getInstance().getBirthDay());
            }
        }).build();
        pvNoLinkOptions.setNPicker(new ArrayList(), sexItem, new ArrayList());
    }
    @OnClick({R.id.siv_update_nickname, R.id.siv_update_birthday, R.id.siv_update_gender, R.id.ivUserPhoto})
    public void updateUserInfo(View view) {
        switch (view.getId()) {
            case R.id.siv_update_nickname:
                startActivity(new Intent(mContext,
UpdateUserNameActivity.class).putExtra("name",UserLoginInfo.getInstance().getCustName()));
                break;
            case R.id.siv_update_gender:
                if (pvNoLinkOptions != null)
                    pvNoLinkOptions.show();
                break;
            case R.id.siv_update_birthday:
                 pvTime.show();
                break;
            case R.id.ivUserPhoto:
                mPhotoPicker.getPhoto();
                break;
        }
    }
    private String getTime(Date date) {//可根据需要自行截取数据显示
        SimpleDateFormat format = new SimpleDateFormat(Constants.TRANS_DATE_BIRTHDAY_FORMAT);
```

```
            return format.format(date);
        }
        @Override
        public void showSuccess(int type) {
            UserLoginInfo loginInfo = UserLoginInfo.getInstance();
            if (type == 0)
            {
               siv_nickname.setRightText(loginInfo.getCustName());
               siv_gender.setRightText(loginInfo.getSex());
               siv_birthday.setRightText(loginInfo.getBirthDay());
            }else{
                 ImageLoaderUtils.display(mContext,ivUserPhoto,loginInfo.getImageUrl());
            }
            RxBus.getIntanceBus().post(new NotifyUserInfo());
        }
        @Override
        public void onSaveInstanceState(Bundle outState) {
            super.onSaveInstanceState(outState);
            mPhotoPicker.onSaveInstanceState(outState);
        }
        @Override
        public void onDestroy() {
            if (mPhotoPicker != null){
                 mPhotoPicker = null ;
            }
            super.onDestroy();
        }
        @Override
        public void onActivityResult(int requestCode, int resultCode, Intent data) {
            super.onActivityResult(requestCode, resultCode, data);
            mPhotoPicker.onActivityResult(requestCode, resultCode, data);
        }
        @Override
        public void onPickPhoto(Uri uri) {
            mPresenter.updateUserHeader(uri);
        }
        @Override
        public String getStatusBarStyle() {
            return STYLE_STATUS_BAR_WHITE;
        }
}
package com.baoneng.bnmall.ui.member;
import android.content.ActivityNotFoundException;
import android.content.Intent;
```

```
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.text.TextUtils;
import android.view.View;
import android.widget.TextView;
import com.baoneng.bnmall.R;
import com.baoneng.bnmall.common.Config;
import com.baoneng.bnmall.common.Constants;
import com.baoneng.bnmall.contract.authentication.AccountLoginContract;
import com.baoneng.bnmall.model.UserLoginInfo;
import com.baoneng.bnmall.network.JsonCache;
import com.baoneng.bnmall.network.Network;
import com.baoneng.bnmall.presenter.authentication.AccountLoginPresenter;
import com.baoneng.bnmall.ui.BaseFragment;
import com.baoneng.bnmall.ui.ClientSettingActivity;
import com.baoneng.bnmall.ui.EnvSettingActivity;
import com.baoneng.bnmall.utils.CacheUtils;
import com.baoneng.bnmall.utils.LoginUtil;
import com.baoneng.bnmall.utils.RxBus;
import com.baoneng.bnmall.utils.ToastUtil;
import com.baoneng.bnmall.utils.Utils;
import com.baoneng.bnmall.widget.LoadingWrap;
import com.baoneng.bnmall.widget.SettingItemView;
import com.baoneng.bnmall.widget.UpdateHelper;
import com.jakewharton.rxbinding2.view.RxView;
import java.util.concurrent.TimeUnit;
import butterknife.BindView;
import butterknife.OnClick;
import io.reactivex.Observable;
import io.reactivex.ObservableEmitter;
import io.reactivex.ObservableOnSubscribe;
import io.reactivex.android.schedulers.AndroidSchedulers;
import io.reactivex.functions.Consumer;
import io.reactivex.functions.Function;
import io.reactivex.schedulers.Schedulers;
import static com.baoneng.bnmall.ui.BaseActivity.STYLE_STATUS_BAR_WHITE;
import static com.baoneng.bnmall.utils.AndroidUtils.getPackageName;
/**
 * app 设置界面
 * <p>
 *  先拿缓存路径的缓存
 */
public class SettingFragment extends BaseFragment<AccountLoginPresenter> implements
```

```
AccountLoginContract.View {
    @BindView(R.id.tv_cache_size)
    TextView mTvCacheSize;
    @BindView(R.id.tv_clean_cache)
    LoadingWrap mLwClearCache;
    @BindView(R.id.go_rating)
    SettingItemView goRating;
    @BindView(R.id.env_setting)
    SettingItemView envSetting;
    //客户端信息设置（测试环境使用）
    @BindView(R.id.client_setting)
    SettingItemView clientSetting;
    @BindView(R.id.tv_logout)
    TextView logout;
    public static SettingFragment newInstance() {
        SettingFragment fragment = new SettingFragment();
        return fragment;
    }
    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        mPresenter = new AccountLoginPresenter(this);
        logout.setVisibility(UserLoginInfo.getInstance().isLogin() ? View.VISIBLE : View.GONE);
        getAndShowCacheSize();
        RxView.clicks(goRating).throttleFirst(1500, TimeUnit.MILLISECONDS)
                .subscribe(new Consumer<Object>() {
                    @Override
                    public void accept(Object object) throws Exception {
                        gotoRate();
                    }
                });
        if (Config.NOT_PRODUCT) {//测试环境暴露
            envSetting.setVisibility(View.VISIBLE);
            clientSetting.setVisibility(View.VISIBLE);
        }
    }
    //获取并显示 cache 大小
    private void getAndShowCacheSize() {
        mLwClearCache.showLoading();
        Observable
                .create(new ObservableOnSubscribe<Long>() {
                    @Override
                    public void subscribe(ObservableEmitter<Long> emitter) throws Exception {
                        JsonCache jsonCache = JsonCache.getInstance(mContext);
```

```
                        emitter.onNext(Network.cacheSize(
                                mContext)
                                        + jsonCache.cacheSize()
                                        + CacheUtils.getTotalCacheSize(mContext)
                                /*+ CustomCacheHelper.getInstance(mContext).cacheSize()*/);
                    }
                })
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(new Consumer<Long>() {
                    @Override
                    public void accept(Long aLong) throws Exception {
                        mTvCacheSize.setText(Utils.getShowByteSize(aLong));
                        mLwClearCache.showContent();
                    }
                });
    }
    void cleanCache() {
        Utils.confirmDialog(mContext, getString(R.string.confirm_clear_cache))
                .map(new Function<Integer, Long>() {
                    @Override
                    public Long apply(Integer integer) throws Exception {
                        mLwClearCache.showLoading();
                        return 0l;
                    }
                })
                .subscribeOn(AndroidSchedulers.mainThread())
                .map(new Function<Long, Long>() {
                    @Override
                    public Long apply(Long l) throws Exception {
                        JsonCache jsonCache = JsonCache.getInstance(mContext);
                        Network.clearCache(mContext);
                        jsonCache.clearCache();
                        CacheUtils.clearAllCache(mContext);
//                          CustomCacheHelper.getInstance(SettingActivity.this).clearDatabase();
                        return Network.cacheSize(mContext)
                                        + jsonCache.cacheSize() + CacheUtils.getTotalCacheSize(mContext)
                                        /* + CustomCacheHelper.getInstance(SettingActivity.this).cacheSize()*/;
                    }
                })
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(new Consumer<Long>() {
                    @Override
```

```
                    public void accept(Long aLong) throws Exception {
                        mTvCacheSize.setText(Utils.getShowByteSize(aLong));
                        mLwClearCache.showContent();
                    }
                });
    }
    private void gotoRate() {
        Uri uri = Uri.parse("market://details?id=" + getPackageName());
        Intent goToMarket = new Intent(Intent.ACTION_VIEW, uri);
        try {
            startActivity(goToMarket);
        } catch (ActivityNotFoundException e) {
            ToastUtil.showShortToast("尚未安装应用市场，无法评分");
        }
    }
    @OnClick({R.id.clean_layout, R.id.about_us, R.id.check_update, R.id.env_setting,
            R.id.tv_logout, R.id.client_setting})
    public void clickBtn(View view) {
        if (Utils.isFastClick()){
            return;
        }
        switch (view.getId()) {
            case R.id.clean_layout:
                String text = mTvCacheSize.getText().toString().replaceAll(" ","");
                if (TextUtils.equals(text,"0KB")){
                    ToastUtil.showShortToast(R.string.no_cache);
                }else {
                    cleanCache();
                }
                break;
            case R.id.about_us:
                listener.replace(Constants.ABOUT_US, Constants.SETTING);
                break;
            case R.id.check_update:
                UpdateHelper.getInstance().checkUpdate(getActivity());
                break;
            case R.id.env_setting:
                startActivity(new Intent(mContext, EnvSettingActivity.class));
                break;
            case R.id.client_setting:
                startActivity(new Intent(mContext, ClientSettingActivity.class));
                break;
            case R.id.tv_logout:
                mPresenter.logout();
```

```
                    break;
            }
        }
        @Override
        public int getContentViewId() {
            return R.layout.fragment_setting;
        }
        @Override
        public String getStatusBarStyle() {
            return STYLE_STATUS_BAR_WHITE;
        }
        @Override
        public void showPasswordError(String message) {
        }
        @Override
        public void noticeRegister(String message) {
        }
        @Override
        public void showLoginSucceed(String msg) {
        }
        @Override
        public void showRegisterWechatUI() {
        }
        @Override
        public void showFailedByWechat(String message) {
        }
        @Override
        public void onUnionLoginSuccess(boolean isFirst) {
        }
        @Override
        public void logoutSuccess() {
            ToastUtil.showShortToast("退出登录成功");
            LoginUtil.logoutSucceed(mContext);
            RxBus.getIntanceBus().post(new NotifyUserInfo());
            getActivity().finish();
        }
        @Override
        public void setNeedImgCode(boolean needImgCode) {
        }
        @Override
        public String getInputImageCode() {
            return null;
        }
        @Override
```

```
    public void refreshImageCode() {
    }
}
package com.baoneng.bnmall.ui.search;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentTransaction;
import android.text.TextUtils;
import android.view.KeyEvent;
import android.view.View;
import android.view.Window;
import android.view.inputmethod.EditorInfo;
import android.view.inputmethod.InputMethodManager;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import com.baoneng.bnmall.R;
import com.baoneng.bnmall.contract.search.SearchContract;
import com.baoneng.bnmall.presenter.search.SearchPresenter;
import com.baoneng.bnmall.ui.BaseActivity;
import java.util.ArrayList;
import java.util.List;
import butterknife.BindView;
import butterknife.OnClick;
import butterknife.OnEditorAction;
import butterknife.OnTextChanged;
import static com.baoneng.bnmall.ui.search.SearchFragment.SEARCH_ASSOCIATIVE_WORDS_TAG;
/**
 * Created by liuxu on 2017/9/14.
 */
public class SearchActivity extends BaseActivity<SearchContract.Presenter> implements SearchContract.View {
    @BindView(R.id.et_search_edit)
    EditText mSearchEdit;
    @BindView(R.id.right_btn)
    View mRightBtn;
    @BindView(R.id.tv_search_btn)
    TextView mSearchBtn;
    @BindView(R.id.tv_cancel_btn)
    TextView mCancelBtn;
    @BindView(R.id.iv_search_clear)
```

```
    ImageView mSearchClear;
    private FragmentManager mFragmentManager;
    private List<SearchFragment> mFragments = new ArrayList<>();
    private String mSearchWord;
    private String mCurrentFragmentTag;
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.search_main_layout);
        initFragment();
        mSearchWord = getIntent().getStringExtra("search_word");
        if(!TextUtils.isEmpty(mSearchWord)) {
            mSearchEdit.setText(mSearchWord);
            mSearchEdit.setSelection(mSearchWord.length());
        }
        mPresenter = new SearchPresenter(this);
    }
    private void initFragment() {
        mFragments.clear();
        mRightBtn.setVisibility(View.VISIBLE);
        mFragmentManager = getSupportFragmentManager();
        FragmentTransaction transaction = mFragmentManager.beginTransaction();
        SearchHistoryFragment searchHistoryFragment = new SearchHistoryFragment();
        AssociativeWordsFragment associativeWordsFragment = new AssociativeWordsFragment();
        mFragments.add(searchHistoryFragment);
        mFragments.add(associativeWordsFragment);
        transaction.add(R.id.fragment_content,                                    searchHistoryFragment,
SearchFragment.SEARCH_HISTORY_TAG);
        transaction.add(R.id.fragment_content,                                  associativeWordsFragment,
SEARCH_ASSOCIATIVE_WORDS_TAG);
        transaction.commit();
        mCurrentFragmentTag = SearchFragment.SEARCH_HISTORY_TAG;
        showFragmentByTag(mCurrentFragmentTag);
    }
    private void showFragmentByTag(String tag) {
        FragmentTransaction transaction = mFragmentManager.beginTransaction();
        for (Fragment f : mFragments) {
            if (tag.equals(f.getTag())) {
                transaction.show(f);
            } else {
                transaction.hide(f);
            }
        }
```

```
        transaction.commit();
    }
    @OnTextChanged(R.id.et_search_edit)
    void searchEditChanaged(CharSequence s, int start, int before, int count) {
        if (TextUtils.isEmpty(s)) {
            mSearchClear.setVisibility(View.GONE);
            mCancelBtn.setVisibility(View.VISIBLE);
            mSearchBtn.setVisibility(View.GONE);
            // 如果输入框中无内容，只展示搜索历史页
            mCurrentFragmentTag = SearchFragment.SEARCH_HISTORY_TAG;
        } else {
            mCancelBtn.setVisibility(View.GONE);
            mSearchBtn.setVisibility(View.VISIBLE);
            mSearchClear.setVisibility(View.VISIBLE);
            //当输入框中有搜索词后，自动进行搜索关联词
            mCurrentFragmentTag = SearchFragment.SEARCH_ASSOCIATIVE_WORDS_TAG;
        }
        mSearchWord = s.toString();
        query(mCurrentFragmentTag, mSearchWord);
    }
    @OnEditorAction(R.id.et_search_edit)
    boolean onEditorActionChanged(TextView v, int actionId, KeyEvent event) {
        boolean handled = false;
        if (actionId == EditorInfo.IME_ACTION_SEARCH) {
            handled = true;
            doSearch(mSearchWord);
                    /*隐藏软键盘*/
            InputMethodManager inputMethodManager = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
            if (inputMethodManager.isActive()) {
                inputMethodManager.hideSoftInputFromWindow(this.getCurrentFocus().getWindowToken(),
0);
            }
        }
        return handled;
    }
    @OnClick({R.id.iv_search_clear, R.id.tv_search_btn, R.id.tv_cancel_btn})
    void onClick(View v) {
        switch (v.getId()) {
            case R.id.iv_search_clear:
                mSearchEdit.getText().clear();
                break;
            case R.id.tv_search_btn:
                doSearch(mSearchWord);
```

```
                    break;
                case R.id.tv_cancel_btn:
                    finish();
                    break;
            }
        }
        public void doSearch(String searchWord) {
            if(!TextUtils.isEmpty(searchWord)) {
                // 保存历史搜索记录
                mPresenter.addToSearchHistory(searchWord);
                Intent intent = new Intent(this, SearchResultActivity.class);
                intent.putExtra("search_word", searchWord);
                startActivity(intent);
                this.finish();
            }
        }
        /**
         * 调用对应 fragment 进行查询操作
         * @param whichFragment
         * @param searchWord
         */
        private void query(String whichFragment, String searchWord) {
            for (SearchFragment f : mFragments) {
                if (whichFragment.equals(f.getTag())) {
                    f.doSearch(searchWord);
                    break;
                }
            }
            showFragmentByTag(whichFragment);
        }
        @Override
        public String getStatusBarStyle() {
            return STYLE_STATUS_BAR_WHITE;
        }
}
```