package com.kingsignal.elf1.ui.home;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

import android.os.Handler;

import android.os.Message;

import androidx.fragment.app.Fragment;

import androidx.fragment.app.FragmentTransaction;

import com.kingsignal.common.base.BasicActivity;

import com.kingsignal.common.base.MyApplication;

import com.kingsignal.common.http.event.ClassEvent;

import com.kingsignal.common.utils.AppManager;

import com.kingsignal.common.utils.toast.ToastUtils;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.databinding.ActivityMainBinding;

import com.kingsignal.elf1.ui.SplashActivity;

import com.kingsignal.elf1.ui.guide.NetworkDetectionActivity;

import com.kingsignal.elf1.ui.guide.NoConnectActivity;

import com.kingsignal.elf1.ui.home.fragment.HomeFragment;

import com.kingsignal.elf1.ui.home.fragment.SettingFragment0;

import com.kingsignal.elf1.ui.login.LoginActivity;

import com.kingsignal.elf1.utils.DoubleClickHelper;

import org.greenrobot.eventbus.EventBus;

import org.greenrobot.eventbus.Subscribe;

import org.greenrobot.eventbus.ThreadMode;

import java.lang.ref.WeakReference;

import static com.kingsignal.common.http.NetworkConstant.DETECTION\_FAILURE;

/\*\*  
 \* @author zwy  
 \* create at 2020-10-30  
 \*/

public class MainActivity extends BasicActivity<ActivityMainBinding> {

private HomeFragment page1 = new HomeFragment();

private SettingFragment0 page2 = new SettingFragment0();

MyHandler handler;

public static void start(Context context) {

Intent intent = new Intent(context, MainActivity.class);

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_main;

}

@Override

public void initData() {

}

@Override

public void initView() {

if (!EventBus.getDefault().isRegistered(this)) {

EventBus.getDefault().register(this);

}

handler = new MyHandler(this);

handlePage(page1);

initListener();

}

public void initListener() {

bindingView.bar.rgPass.setOnCheckedChangeListener((group, checkedId) -> {

if (checkedId == R.id.rb\_one) {

handlePage(page1);

} else {

handlePage(page2);

}

});

}

private void handlePage(Fragment page) {

if (page.isAdded()) {

hideAll().show(page).commit();

} else {

hideAll().add(R.id.fl\_content, page).commit();

}

}

private FragmentTransaction hideAll() {

FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();

if (page1.isAdded()) transaction.hide(page1);

if (page2.isAdded()) transaction.hide(page2);

return transaction;

}

@Subscribe(threadMode = ThreadMode.MAIN)

public void onClassEvent(ClassEvent event) {

// if (event.mclass.equals("NoConnectActivity")) {

// if (handler != null) {

// handler.sendEmptyMessageDelayed(1, 2000);

// }

// } else

// if (!DoubleClickHelper.isOnDoubleClick()) {

if (!MyApplication.getInstance().failureJump) {

EventBus.getDefault().cancelEventDelivery(event);

} else {

if (event.mclass.equals("NetworkDetectionActivity")) {

if (AppManager.getAppManager().currentActivity().getClass() != LoginActivity.class

|| AppManager.getAppManager().currentActivity().getClass() != NetworkDetectionActivity.class

|| AppManager.getAppManager().currentActivity().getClass() != NoConnectActivity.class

) {

AppManager.getAppManager().finishAllActivity\_1();

NetworkDetectionActivity.start(this, DETECTION\_FAILURE);

EventBus.getDefault().cancelEventDelivery(event);

}

} else if (event.mclass.equals("LoginActivity")) {

ToastUtils.show(this, getString(R.string.login\_failure));

AppManager.getAppManager().finishAllActivity\_1();

LoginActivity.start(this);

}

}

// }

}

public void run() {

// if (AppManager.getAppManager().currentActivity().getClass() != NetworkDetectionActivity.class) {

// NetworkDetectionActivity.start(this, "");

// }

}

static class MyHandler extends Handler {

// SoftReference<Activity> 也可以使用软应用 只有在内存不足的时候才会被回收

private final WeakReference<MainActivity> mActivity;

private MyHandler(MainActivity activity) {

mActivity = new WeakReference<MainActivity>(activity);

}

@Override

public void handleMessage(Message msg) {

Activity activity = mActivity.get();

if (activity != null) {

mActivity.get().run();

}

super.handleMessage(msg);

}

}

}

package com.kingsignal.elf1.ui.home.fragment;

import android.os.Build;

import android.text.TextUtils;

import android.view.animation.DecelerateInterpolator;

import androidx.annotation.RequiresApi;

import com.kingsignal.common.base.PresenterFragment;

import com.kingsignal.common.http.NetworkConstant;

import com.kingsignal.common.utils.Dp2PxUtils;

import com.kingsignal.common.utils.LogUtil;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.databinding.FragmentHomeBinding;

import com.kingsignal.elf1.entity.HomeInfoBean;

import com.kingsignal.elf1.entity.TopologyBean;

import com.kingsignal.elf1.presenter.home.HomePresenter;

import com.kingsignal.elf1.ui.guide.SubRouteOneActivity;

import com.kingsignal.elf1.ui.setting.ConnectedEquipmentActivity;

import com.kingsignal.elf1.ui.setting.LinkDeviceActivity;

import com.kingsignal.elf1.ui.setting.NotSettingsActivity;

import com.kingsignal.elf1.view.TopologyView;

import java.util.ArrayList;

import java.util.List;

import java.util.Timer;

import java.util.TimerTask;

/\*\*  
 \* @author zwy  
 \* create at 2020-11-05  
 \*/

public class HomeFragment extends PresenterFragment<HomePresenter, FragmentHomeBinding> {

private TimerTask mTimerTask = null;

private Timer mTimer = null;

private Timer mSecondTimer = null;

private TimerTask mSecondTimerTask = null;

private String mac;

boolean isVisible = true;

TopologyBean.TopologyInfoBean infoBean;

List<TopologyBean.TopologyInfoBean.ChildDevicesBean> childList = new ArrayList<>();

@Override

public int getLayoutId() {

return R.layout.fragment\_home;

}

@Override

public void initData() {

}

@RequiresApi(api = Build.VERSION\_CODES.M)

@Override

public void initView() {

basePresenter.getHomeInfo();

basePresenter.getTopology();

initListener();

}

public void initListener() {

bindingView.tvConnect.setOnClickListener(view -> {

LinkDeviceActivity.start(getContext(), "", true);

});

bindingView.ivIcon.setOnClickListener(view -> {

NotSettingsActivity.start(getContext(), mac);

});

bindingView.ivSubRouter.setOnClickListener(view -> {

SubRouteOneActivity.start(getContext(), NetworkConstant.HOME\_SUB\_ROUTER);

});

bindingView.topologyView.setListener(position -> {

if (infoBean == null) {

return;

}

if (1001 == position) {

mac = infoBean.getMac\_addr();

} else {

if (childList != null && childList.size() > 0) {

mac = childList.get(position).getMac\_addr();

}

}

NotSettingsActivity.start(getContext(), mac);

});

}

public void setHomeFailure() {

}

public void setHomeBean(HomeInfoBean homeBean) {

String[] txRate = homeBean.getWan\_info().getTx\_rate().split(" ");

String[] rxRate = homeBean.getWan\_info().getRx\_rate().split(" ");

String ssid = "";

if (!TextUtils.isEmpty(homeBean.getWifi\_info().get\_$2g\_ssid())) {

ssid = homeBean.getWifi\_info().get\_$2g\_ssid();

}

if (!TextUtils.isEmpty(homeBean.getWifi\_info().get\_$5g\_ssid())) {

ssid = homeBean.getWifi\_info().get\_$5g\_ssid();

}

mac = homeBean.getWan\_info().getMac\_addr();

bindingView.tvWifiName.setText(ssid);

bindingView.tvUpload.setText(txRate[0]);

bindingView.tvDownload.setText(rxRate[0]);

bindingView.tvConnect.setText(getString(R.string.device\_connect\_number,

homeBean.getDevice\_info().getClient\_cnt() + ""));

bindingView.tvUploadHint.setText(getString(R.string.upload) + " " + txRate[1]);

bindingView.tvDownloadHint.setText(getString(R.string.download) + " " + rxRate[1]);

bindingView.tvDeviceName.setText(homeBean.getDevice\_info().getDevice\_name());

}

public void setTopologyBean(TopologyBean.TopologyInfoBean infoBean) {

this.infoBean = infoBean;

childList = infoBean.getChild\_devices();

int topologyType = -1;

if (infoBean.getChild\_devices() != null) {

if (infoBean.getChild\_devices().size() == 1) {

topologyType = 1;

} else if (infoBean.getChild\_devices().size() == 2) {

topologyType = 2;

} else if (infoBean.getChild\_devices().size() == 3) {

topologyType = 3;

} else {

topologyType = 0;

}

}

bindingView.topologyView.initData(infoBean, topologyType);

}

@Override

public void onResume() {

super.onResume();

if (!isHidden()) {

startTimer();

}

}

@Override

public void onHiddenChanged(boolean hidden) {

super.onHiddenChanged(hidden);

isVisible = hidden;

if (hidden) {

stopTimer();

} else {

startTimer();

}

}

@Override

public void onPause() {

super.onPause();

stopTimer();

}

private void startTimer() {

stopTimer();

bindingView.topologyView.startAnimator();

if (mTimer == null) {

mTimer = new Timer();

}

if (mTimerTask == null) {

mTimerTask = new TimerTask() {

@Override

public void run() {

basePresenter.getHomeInfo();

}

};

}

if (mTimer != null && mTimerTask != null) {

mTimer.schedule(mTimerTask, 0, 5000);

}

if (mSecondTimer == null) {

mSecondTimer = new Timer();

}

if (mSecondTimerTask == null) {

mSecondTimerTask = new TimerTask() {

@Override

public void run() {

basePresenter.getTopology();

}

};

}

if (mSecondTimer != null && mSecondTimerTask != null) {

mSecondTimer.schedule(mSecondTimerTask, 0, 10000);

}

}

private void stopTimer() {

if (basePresenter.disposable != null && !basePresenter.disposable.isDisposed()) {

basePresenter.disposable.dispose();

}

if (basePresenter.mTopDisposable != null && !basePresenter.mTopDisposable.isDisposed()) {

basePresenter.mTopDisposable.dispose();

}

bindingView.topologyView.stopAnimator();

if (mTimerTask != null) {

mTimerTask.cancel();

mTimerTask = null;

}

if (mTimer != null) {

mTimer.cancel();

mTimer = null;

}

if (mSecondTimerTask != null) {

mSecondTimerTask.cancel();

mSecondTimerTask = null;

}

if (mSecondTimer != null) {

mSecondTimer.cancel();

mSecondTimer = null;

}

}

}

package com.kingsignal.elf1.ui.home.fragment;

import android.app.Activity;

import android.app.Dialog;

import android.content.Intent;

import android.content.pm.PackageInfo;

import android.content.pm.PackageManager;

import android.content.pm.ResolveInfo;

import android.net.Uri;

import android.os.Handler;

import android.os.Message;

import android.os.Parcelable;

import android.view.View;

import android.widget.Toast;

import androidx.databinding.ObservableField;

import androidx.recyclerview.widget.GridLayoutManager;

import com.chad.library.adapter.base.BaseQuickAdapter;

import com.google.android.material.appbar.AppBarLayout;

import com.hjq.toast.ToastUtils;

import com.kingsignal.common.base.PresenterFragment;

import com.kingsignal.common.utils.LogUtil;

import com.kingsignal.common.utils.SP;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.adapter.SettingAdapter;

import com.kingsignal.elf1.databinding.FragmentSetting0Binding;

import com.kingsignal.elf1.dialog.MessageDialog;

import com.kingsignal.elf1.entity.SettingBean;

import com.kingsignal.elf1.entity.VerInfoBean;

import com.kingsignal.elf1.presenter.home.SettingPresenter;

import com.kingsignal.elf1.ui.setting.GustModeActivity;

import com.kingsignal.elf1.ui.setting.online.MainInternetSettingActivity;

import com.kingsignal.elf1.ui.setting.online.WirelessSettingActivity;

import com.kingsignal.elf1.ui.setting.parentalcontrol.ParentalControlActivity;

import com.kingsignal.elf1.ui.setting.system.SettingsActivity;

import com.kingsignal.elf1.ui.setting.system.SoftwareUpgradeActivity;

import com.kingsignal.elf1.ui.setting.wifi.CheckWifiSignalActivity;

import com.kingsignal.elf1.view.AppBarStateChangeListener;

import java.util.ArrayList;

import java.util.List;

/\*\*  
 \* @author zwy  
 \* create at 2020-11-05  
 \*/

public class SettingFragment0 extends PresenterFragment<SettingPresenter, FragmentSetting0Binding> {

SettingAdapter mAdapter;

List<SettingBean> mList = new ArrayList<>();

boolean isVisible = true;

ObservableField<VerInfoBean> verInfoBean = new ObservableField<VerInfoBean>();

int[] icons = {R.mipmap.lvzhou\_wifi, R.mipmap.adduser

, R.mipmap.exportservices, R.mipmap.team

, R.mipmap.signaldetection, R.mipmap.assessed\_badge

, R.mipmap.download, R.mipmap.set\_feedback

};

@Override

public int getLayoutId() {

return R.layout.fragment\_setting0;

}

@Override

public void initData() {

}

@Override

public void initView() {

bindingView.viewHeader.ivBack.setVisibility(View.GONE);

bindingView.viewHeader.tvTitle.setText(getString(R.string.setting));

initAdapter();

initListener();

}

public void initAdapter() {

String[] d = getResources().getStringArray(R.array.settings\_list0);

SettingBean bean = null;

for (int i = 0; i < d.length; i++) {

bean = new SettingBean();

bean.setName(d[i]);

bean.setImgIcon(icons[i]);

mList.add(bean);

}

mAdapter = new SettingAdapter(mList);

//布局管理器所需参数，上下文

GridLayoutManager gridLayoutManager = new GridLayoutManager(getActivity(), 2);

bindingView.recyclerView.setLayoutManager(gridLayoutManager);

bindingView.recyclerView.setAdapter(mAdapter);

}

public void initListener() {

bindingView.appBarLayout.addOnOffsetChangedListener(new AppBarStateChangeListener() {

@Override

public void onStateChanged(AppBarLayout appBarLayout, State state) {

if (state == State.COLLAPSED) {

bindingView.viewHeader.clTitleView.setVisibility(View.VISIBLE);

} else {

bindingView.viewHeader.clTitleView.setVisibility(View.GONE);

}

}

@Override

public void onOffsetChanged(AppBarLayout appBarLayout) {

if (appBarLayout.getBottom() < 200) {

bindingView.viewHeader.clTitleView.setVisibility(View.VISIBLE);

} else {

bindingView.viewHeader.clTitleView.setVisibility(View.GONE);

}

}

});

mAdapter.setOnItemClickListener((adapter, view, position) -> {

switch (position) {

case 0:

WirelessSettingActivity.start(getContext());

break;

case 1:

GustModeActivity.start(getContext());

break;

case 2:

MainInternetSettingActivity.start(getContext());

break;

case 3:

ParentalControlActivity.start(getContext());

break;

case 4:

CheckWifiSignalActivity.start(getContext());

break;

case 5:

SettingsActivity.start(getActivity());

break;

case 6:

SoftwareUpgradeActivity.start(getContext(), verInfoBean.get());

break;

case 7:

openEmail();

break;

default:

break;

}

});

}

@Override

public void onResume() {

super.onResume();

if (!isHidden()) {

basePresenter.getRemoteVersion();

}

}

@Override

public void onPause() {

super.onPause();

if(basePresenter.disposable!=null&& !basePresenter.disposable.isDisposed()) {

basePresenter.disposable.dispose();

}

}

@Override

public void onHiddenChanged(boolean hidden) {

super.onHiddenChanged(hidden);

isVisible = hidden;

if (!hidden) {

basePresenter.getRemoteVersion();

}

}

/\*\*

\* 获取远程服务

\*/

public void onDataSuccess(VerInfoBean infoBean) {

verInfoBean.set(infoBean);

mAdapter.setVerInfoBean(infoBean);

mAdapter.notifyDataSetChanged();

}

/\*\*

\* 打开邮箱客户端

\*/

private void openEmail() {

Uri uri = Uri.parse("mailto:" + "rd\_support@kingsignal.com");

List<ResolveInfo> packageInfos = getActivity().getPackageManager().queryIntentActivities(new Intent(Intent.ACTION\_SENDTO, uri), 0);

List<String> tempPkgNameList = new ArrayList<>();

List<Intent> emailIntents = new ArrayList<>();

for (ResolveInfo info : packageInfos) {

String pkgName = info.activityInfo.packageName;

if (!tempPkgNameList.contains(pkgName)) {

tempPkgNameList.add(pkgName);

Intent intent = getActivity().getPackageManager().getLaunchIntentForPackage(pkgName);

emailIntents.add(intent);

}

}

if (!emailIntents.isEmpty()) {

Intent chooserIntent = Intent.createChooser(emailIntents.remove(0), getString(R.string.mail\_choose));

if (chooserIntent != null) {

chooserIntent = new Intent(Intent.ACTION\_SENDTO, uri);

/\*设置邮件默认地址，多个收件人，String数组\*/

// chooserIntent.putExtra(android.content.Intent.EXTRA\_EMAIL, new String[]{"752790234@qq.com"});

/\*多个抄送人，String数组\*/

//email.putExtra(android.content.Intent.EXTRA\_CC, new String[]{"752790234@qq.com"});

/\*邮件标题\*/

chooserIntent.putExtra(android.content.Intent.EXTRA\_SUBJECT, getString(R.string.mail\_subject));

/\*邮件正文\*/

chooserIntent.putExtra(android.content.Intent.EXTRA\_TEXT, getString(R.string.mail\_text));

chooserIntent.putExtra(Intent.EXTRA\_INITIAL\_INTENTS, emailIntents.toArray(new Parcelable[]{}));

startActivity(chooserIntent);

} else {

emailDialog();

}

} else {

emailDialog();

}

}

/\*\*

\* 未安装邮箱提示

\*/

public void emailDialog() {

new MessageDialog.Builder(getActivity())

.setTitle(getResources().getString(R.string.prompt\_hint)) // 标题可以不用填写

.setMessage(getResources().getString(R.string.email\_hint))

.setConfirm(getResources().getString(R.string.ok))

.setCancel("") // 设置 null 表示不显示取消按钮

//.setAutoDismiss(false) // 设置点击按钮后不关闭对话框

.setListener(new MessageDialog.OnListener() {

@Override

public void onConfirm(Dialog dialog) {

}

@Override

public void onCancel(Dialog dialog) {

}

})

.show();

}

}

package com.kingsignal.elf1.view;

import android.animation.ObjectAnimator;

import android.animation.TimeInterpolator;

import android.animation.ValueAnimator;

import android.content.Context;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.drawable.Drawable;

import android.os.Build;

import android.text.TextUtils;

import android.util.AttributeSet;

import android.util.Log;

import android.view.MotionEvent;

import android.view.View;

import androidx.annotation.Nullable;

import com.kingsignal.common.utils.Dp2PxUtils;

import com.kingsignal.common.utils.LogUtil;

import com.kingsignal.common.utils.toast.ToastUtils;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.entity.TopologyBean;

import java.util.ArrayList;

import java.util.List;

/\*\*  
 \* @author zwy  
 \* create at 2020-12-05  
 \*/

public class TopologyView extends View {

private Bitmap parentBitmap;

//白色圆弧画笔

private Paint parentBgPaint;

//白色圆弧画笔

private Paint linePaint;

//灰色的画笔

private Paint grayLinePaint;

private int textSize = 25;

private int topologyType;

int mTopologyX;

int mTopologyY;

int mMainTopologyX;

int mMainTopologyY;

int imgHeight;

int imgWidth;

private int test = 0;

float mCurAnimValue = 0;

ValueAnimator animator;

List<TopologyModel> list = new ArrayList<>();

int[] screen = new int[2];

private MyOnClickListener listener;

public interface MyOnClickListener {

public void onMyClickListener(int position);

}

public void setListener(MyOnClickListener listener) {

this.listener = listener;

}

private TopologyBean.TopologyInfoBean topologyBean;

public TopologyView(Context context) {

super(context);

initView();

}

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

super(context, attrs);

initView();

}

public void initView() {

screen = Dp2PxUtils.getDisplayMetric(getContext());

parentBitmap = getBitmap(getContext(), R.drawable.home\_icon);

imgHeight = parentBitmap.getHeight();

imgWidth = parentBitmap.getWidth();

parentBgPaint = new Paint();

parentBgPaint.setAntiAlias(true);

parentBgPaint.setColor(Color.parseColor("#303849"));

parentBgPaint.setTextSize(textSize);

linePaint = new Paint();

linePaint.setAntiAlias(true);

linePaint.setStrokeWidth(2);

linePaint.setColor(Color.parseColor("#303849"));

grayLinePaint = new Paint();

grayLinePaint.setAntiAlias(true);

grayLinePaint.setStrokeWidth(2);

grayLinePaint.setColor(Color.parseColor("#20303849"));

animator = ValueAnimator.ofFloat(0, 1);

animator.addUpdateListener(valueAnimator -> {

mCurAnimValue = (Float) valueAnimator.getAnimatedValue();

invalidate();

});

animator.setDuration(1300);

animator.setRepeatCount(ValueAnimator.INFINITE);

}

public void startAnimator() {

if (animator != null) {

animator.start();

}

}

public void stopAnimator() {

if (animator != null) {

animator.end();

}

}

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

drawParent(canvas);

if (topologyBean != null && topologyBean.getChild\_devices() != null && topologyBean.getChild\_devices().size() > 0) {

drawLine(canvas);

}

if (topologyBean != null) {

drawText(canvas);

}

}

private void drawParent(Canvas canvas) {

canvas.save();

if (topologyBean != null) {

list.clear();

mMainTopologyX = (getWidth() - parentBitmap.getWidth()) / 2;

mMainTopologyY = 50;

canvas.drawBitmap(parentBitmap, (getWidth() - parentBitmap.getWidth()) / 2, 70, parentBgPaint);

mTopologyX = (getWidth() - parentBitmap.getWidth()) / 2;

mTopologyY = getWidth() \* 3 / 5;

if (screen[0] >= 1440) {

mTopologyY = getWidth() \* 1 / 2;

}

if (topologyType == 1) {

if (topologyBean.getChild\_devices().get(0) != null) {

list.add(new TopologyModel((getWidth() - imgWidth) / 2, mTopologyY));

canvas.drawBitmap(parentBitmap, (getWidth() - imgWidth) / 2, mTopologyY, parentBgPaint);

}

} else if (topologyType == 2) {

for (int i = 0; i < topologyBean.getChild\_devices().size(); i++) {

if (i == 0) {

list.add(new TopologyModel(mTopologyX - getWidth() / 3, mTopologyY - imgHeight \* 2 / 3));

canvas.drawBitmap(parentBitmap, mTopologyX - getWidth() / 3, mTopologyY - imgHeight \* 2 / 3, parentBgPaint);

}

if (i == 1) {

list.add(new TopologyModel(mTopologyX + getWidth() / 3, mTopologyY - imgHeight \* 2 / 3));

canvas.drawBitmap(parentBitmap, mTopologyX + getWidth() / 3, mTopologyY - imgHeight \* 2 / 3, parentBgPaint);

}

}

} else if (topologyType == 3) {

for (int i = 0; i < topologyBean.getChild\_devices().size(); i++) {

if(i==0){

list.add(new TopologyModel(mTopologyX, mTopologyY));

canvas.drawBitmap(parentBitmap, mTopologyX, mTopologyY, parentBgPaint);

}

if (i == 1) {

list.add(new TopologyModel(mTopologyX - getWidth() / 3, mTopologyY - imgHeight \* 2 / 3));

canvas.drawBitmap(parentBitmap, mTopologyX - getWidth() / 3, mTopologyY - imgHeight \* 2 / 3, parentBgPaint);

}

if (i == 2) {

list.add(new TopologyModel(mTopologyX + getWidth() / 3, mTopologyY - imgHeight \* 2 / 3));

canvas.drawBitmap(parentBitmap, mTopologyX + getWidth() / 3, mTopologyY - imgHeight \* 2 / 3, parentBgPaint);

}

}

} else {

for (int i = 0; i < topologyBean.getChild\_devices().size(); i++) {

if (i == 0) {

list.add(0, new TopologyModel(mTopologyX, mTopologyY));

canvas.drawBitmap(parentBitmap, mTopologyX, mTopologyY, parentBgPaint);

}

if (i == 1) {

list.add(1, new TopologyModel(mTopologyX - getWidth() / 3, mTopologyY - imgHeight \* 2 / 3));

canvas.drawBitmap(parentBitmap, mTopologyX - getWidth() / 3, mTopologyY - imgHeight \* 2 / 3, parentBgPaint);

}

if (i == 2) {

list.add(2, new TopologyModel(mTopologyX + getWidth() / 3, mTopologyY - imgHeight \* 2 / 3));

canvas.drawBitmap(parentBitmap, mTopologyX + getWidth() / 3, mTopologyY - imgHeight \* 2 / 3, parentBgPaint);

}

if (i == 3) {

list.add(3, new TopologyModel(mTopologyX - getWidth() / 3, mMainTopologyY + imgHeight + imgHeight / 2));

canvas.drawBitmap(parentBitmap, mTopologyX - getWidth() / 3, mMainTopologyY + imgHeight + imgHeight / 2, parentBgPaint);

}

if (i == 4) {

list.add(4, new TopologyModel(mTopologyX + getWidth() / 3, mMainTopologyY + imgHeight + imgHeight / 2));

canvas.drawBitmap(parentBitmap, mTopologyX + getWidth() / 3, mMainTopologyY + imgHeight + imgHeight / 2, parentBgPaint);

}

}

}

}

}

private void drawLine(Canvas canvas) {

float startX = (getWidth() - imgWidth) / 2 + imgWidth / 2;

float startY = 70 + imgHeight;

float entX = (getWidth() - imgWidth) / 2 + imgWidth / 2;

float entY = mTopologyY;

for (int i = 0; i < topologyBean.getChild\_devices().size(); i++) {

if(i>4){

return;

}

if (topologyBean.getChild\_devices().get(i) != null) {

if (topologyType == 1) {

if (topologyBean.getChild\_devices().get(0) != null

&& !TextUtils.isEmpty(topologyBean.getChild\_devices().get(0).getDevice\_name())) {

entX = startX;

entY = list.get(0).getLocationY();

}

} else if (topologyType == 2) {

if (i == 0) {

entX = list.get(0).getLocationX() + imgWidth - 10;

entY = list.get(0).getLocationY() + 10;

}

if (i == 1) {

entX = list.get(1).getLocationX() + 10;

entY = list.get(1).getLocationY() + 10;

}

} else {

if (i == 0) {

entY = list.get(0).getLocationY();

} else {

if (i == 1 || i == 3) {

entX = list.get(i).getLocationX() + imgWidth - 10;

} else if (i == 2 || i == 4) {

entX = list.get(i).getLocationX() + 10;

}

entY = list.get(i).getLocationY() + 10;

}

}

if (mCurAnimValue >= 0.5) {

canvas.drawLine(startX, startY, entX, entY, linePaint);

} else {

canvas.drawLine(startX, startY, entX, entY, grayLinePaint);

}

}

}

}

private void drawText(Canvas canvas) {

String deviceName = "";

float textX = 0;

float textY = 0;

float textWidth = 0;

float mainTextHeight = getTextViewLength(parentBgPaint, topologyBean.getDevice\_name());

int mMainTextX;

if (mainTextHeight > imgWidth) {

mMainTextX = (int) (mMainTopologyX - (mainTextHeight - imgWidth) / 2);

} else {

mMainTextX = (int) (mMainTopologyX + (imgWidth - mainTextHeight) / 2);

}

canvas.drawText(topologyBean.getDevice\_name(), mMainTextX, 50, parentBgPaint);

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

if (topologyBean.getChild\_devices().get(i) != null

&& !TextUtils.isEmpty(topologyBean.getChild\_devices().get(i).getDevice\_name())) {

deviceName = topologyBean.getChild\_devices().get(i).getDevice\_name();

textWidth = getTextViewLength(parentBgPaint, deviceName);

if (textWidth < imgWidth) {

textX = (imgWidth - textWidth) / 2;

}else {

textX=-(textWidth-imgWidth)/2;

}

textY = imgHeight + 30;

canvas.drawText(deviceName, list.get(i).getLocationX()+textX, list.get(i).getLocationY() + textY, parentBgPaint);

}

}

}

private static float getTextViewLength(Paint paint, String text) {

float textLength = paint.measureText(text);

return textLength;

}

private static Bitmap getBitmap(Context context, int vectorDrawableId) {

Bitmap bitmap = null;

if (Build.VERSION.SDK\_INT > Build.VERSION\_CODES.LOLLIPOP) {

Drawable vectorDrawable = context.getDrawable(vectorDrawableId);

bitmap = Bitmap.createBitmap(vectorDrawable.getIntrinsicWidth(),

vectorDrawable.getIntrinsicHeight(), Bitmap.Config.ARGB\_8888);

Canvas canvas = new Canvas(bitmap);

vectorDrawable.setBounds(0, 0, canvas.getWidth(), canvas.getHeight());

vectorDrawable.draw(canvas);

} else {

bitmap = BitmapFactory.decodeResource(context.getResources(), vectorDrawableId);

}

return bitmap;

}

/\*\*

\* 设置数据

\*

\* @param

\*/

public void initData(TopologyBean.TopologyInfoBean topologyBean, int topologyType) {

if (topologyBean.getChild\_devices() == null) {

topologyBean.setChild\_devices(new ArrayList<>());

}

String childName=getResources().getString(R.string.child\_router);

if(TextUtils.isEmpty(topologyBean.getDevice\_name())){

topologyBean.setDevice\_name(getResources().getString(R.string.main\_router));

}

for (int i=0;i<topologyBean.getChild\_devices().size();i++){

if(TextUtils.isEmpty(topologyBean.getChild\_devices().get(i).getDevice\_name())){

topologyBean.getChild\_devices().get(i).setDevice\_name(childName);

}

}

this.topologyBean = topologyBean;

this.topologyType = topologyType;

list.clear();

invalidate();

}

@Override

public boolean onTouchEvent(MotionEvent event) {

float x;

float y;

switch (event.getAction()) {

case MotionEvent.ACTION\_DOWN:

break;

case MotionEvent.ACTION\_UP:

x = event.getX();

y = event.getY();

onClickListener(x, y);

break;

}

return true;

}

public void onClickListener(float x, float y) {

if (topologyBean == null) {

return;

}

float limitX = mMainTopologyX + parentBitmap.getWidth();

float limitY = mMainTopologyX + parentBitmap.getHeight();

float originX = mTopologyX;

float originY = mTopologyY;

if (x >= mMainTopologyX && x <= limitX && y >= mMainTopologyY && y <= limitY) {

listener.onMyClickListener(1001);

}

if (topologyBean.getChild\_devices() == null) {

return;

}

for (int i = 0; i < topologyBean.getChild\_devices().size(); i++) {

originX =list.get(i).getLocationX();

originY = list.get(i).getLocationY();

limitX = originX + parentBitmap.getWidth();

limitY = originY + parentBitmap.getHeight();

if (x >= originX && x <= limitX && y >= originY && y <= limitY) {

if (listener != null) {

listener.onMyClickListener(i);

}

}

}

}

}

package com.kingsignal.common.base;

import android.content.Context;

import android.content.res.Configuration;

import android.content.res.Resources;

import android.graphics.Typeface;

import android.os.Bundle;

import android.util.DisplayMetrics;

import android.widget.Toast;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AppCompatActivity;

import androidx.databinding.DataBindingUtil;

import androidx.databinding.ViewDataBinding;

import com.bigkoo.pickerview.builder.OptionsPickerBuilder;

import com.bigkoo.pickerview.listener.OnOptionsSelectListener;

import com.gyf.barlibrary.ImmersionBar;

import com.kaopiz.kprogresshud.KProgressHUD;

import com.kingsignal.common.R;

import com.kingsignal.common.base.BaseView;

import com.kingsignal.common.base.BasicPresenter;

import com.kingsignal.common.http.event.ClassEvent;

import com.kingsignal.common.utils.AppManager;

import com.kingsignal.common.utils.LanguageUtils;

import com.kingsignal.common.utils.SP;

import com.kingsignal.common.utils.toast.ToastUtils;

import java.lang.reflect.ParameterizedType;

import java.lang.reflect.Type;

import java.util.Locale;

/\*\*

\* PresenterActivity

\*

\* @author:zwy

\* @date:2020-10-8

\*/

public abstract class PresenterActivity<T extends BasicPresenter, SV extends ViewDataBinding> extends AppCompatActivity implements BaseView {

protected T basePresenter;

// 布局view

public SV bindingView;

//沉浸式

private ImmersionBar mImmersionBar;

KProgressHUD kProgressHUD;

@Override

protected void onCreate(@Nullable Bundle savedInstanceState) {

setLanguage(SP.getLanguage());

initPresenter();

super.onCreate(savedInstanceState);

mImmersionBar = ImmersionBar.with(this).statusBarDarkFont(true).keyboardEnable(true);

mImmersionBar.init();

AppManager.getAppManager().addActivity(this);

initData();

//创建一个新的的布局绑定

bindingView = DataBindingUtil.setContentView(this, getLayoutId());

initView();

}

protected void showProgress(String s) {

kProgressHUD = KProgressHUD.create(this)

.setStyle(KProgressHUD.Style.SPIN\_INDETERMINATE)

.setLabel(s)

.setCancellable(true)

.setAnimationSpeed(2)

.setDimAmount(0.5f)

.show();

}

protected void hideProgress() {

if (kProgressHUD != null)

kProgressHUD.dismiss();

}

@Override

protected void onDestroy() {

super.onDestroy();

hideProgress();

if (basePresenter != null) {

basePresenter.detachView();

}

if (mImmersionBar != null)

mImmersionBar.destroy();

if (bindingView != null) {

bindingView.unbind();

bindingView = null;

}

AppManager.getAppManager().finishActivity(this);

}

@Override

public void refresh() {

}

@Override

public void showLoading() {

showProgress(getString(R.string.network\_progress));

}

@Override

public void hideLoading() {

hideProgress();

}

@Override

public void showToast(int msg) {

ToastUtils.show(this, getString(msg));

}

@Override

public void showToast(String msg) {

ToastUtils.show(this, msg);

}

private void initPresenter() {

try {

Type type = getClass().getGenericSuperclass();

Class<T> superClass = null;

if (type instanceof ParameterizedType) {

superClass = (Class<T>) ((ParameterizedType) type).getActualTypeArguments()[0];

}

if (superClass != null) {

basePresenter = superClass.newInstance();

basePresenter.attachView(this);

}

} catch (IllegalAccessException e) {

e.printStackTrace();

} catch (InstantiationException e) {

e.printStackTrace();

}

}

/\*\*

\* 加载布局

\*/

public abstract int getLayoutId();

/\*\*

\* 初始化数据

\*/

public abstract void initData();

/\*\*

\* 初始化 View

\*/

public abstract void initView();

/\*\*

\* 文字语言

\*

\* @param languageType 0,中文 1,英文

\*/

public void setLanguage(int languageType) {

String languageToLoad = "zh";

Locale locale = new Locale(languageToLoad);

Locale.setDefault(locale);

Resources resources = getResources();//获得res资源对象

Configuration config = resources.getConfiguration();//获得设置对象

DisplayMetrics dm = resources.getDisplayMetrics();//获得屏幕参数：主要是分辨率，像素等。

switch (languageType) {

case 0:

config.locale = Locale.CHINA; //系统默认语言

SP.setLanguage(0);

break;

case 1:

config.locale = Locale.ENGLISH; //英文

SP.setLanguage(1);

break;

default:

config.locale = Locale.SIMPLIFIED\_CHINESE;

SP.setLanguage(0);

break;

}

resources.updateConfiguration(config, dm);

}

public OptionsPickerBuilder genPickerBuilder(Context context, OnOptionsSelectListener listener) {

return new OptionsPickerBuilder(context, listener)

.setSubmitText(getString(R.string.sure\_tips))//确定按钮文字

.setCancelText(getString(R.string.cancel\_tips))//取消按钮文字

.setSubCalSize(16)//确定和取消文字大小

.setSubmitColor(getResources().getColor(R.color.black))//确定按钮文字颜色

.setCancelColor(getResources().getColor(R.color.black))//取消按钮文字颜色

.setTitleBgColor(getResources().getColor(R.color.white))//标题背景颜色 Night mode

.setTitleSize(50)

.setContentTextSize(14)//滚轮文字大小

.setTextColorCenter(getResources().getColor(R.color.black))

.setTextColorOut(getResources().getColor(R.color.view\_gray))

.setDividerColor(getResources().getColor(R.color.view\_gray))

.setCyclic(false, false, false)//循环与否

.setSelectOptions(0, 0, 0) //设置默认选中项

.setOutSideCancelable(true)//点击外部dismiss default true

.isDialog(false)//是否显示为对话框样式

.isRestoreItem(true)//切换时是否还原，设置默认选中第一项。

.setTypeface(Typeface.DEFAULT)

.setLineSpacingMultiplier(3f);

}

}

package com.kingsignal.common.base;

public interface BaseView {

public void refresh();

public void showLoading();

public void hideLoading();

public void showToast(int msg);

public void showToast(String msg);

}

package com.kingsignal.elf1.ui.setting;

import android.app.Dialog;

import android.content.Context;

import android.content.Intent;

import android.view.View;

import androidx.recyclerview.widget.LinearLayoutManager;

import com.kingsignal.common.base.PresenterActivity;

import com.kingsignal.common.http.response.MacDeviceBean;

import com.kingsignal.common.utils.SP;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.adapter.ConnectedEquipmentAdapter;

import com.kingsignal.elf1.databinding.ActivityLinkDeviceBinding;

import com.kingsignal.elf1.dialog.MessageDialog;

import com.kingsignal.elf1.presenter.settings.LinkDevicePresenter;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Timer;

import java.util.TimerTask;

/\*\*  
 \* @author zwy  
 \* create at 2020-12-06  
 \*/

public class LinkDeviceActivity extends PresenterActivity<LinkDevicePresenter, ActivityLinkDeviceBinding> {

ConnectedEquipmentAdapter mAdapter;

ConnectedEquipmentAdapter mOffAdapter;

private List<MacDeviceBean> mOnList = new ArrayList<>();

private List<MacDeviceBean> mOffList = new ArrayList<>();

private List<String> mOnMacList = new ArrayList<>();

private List<String> mOffMacList = new ArrayList<>();

boolean isFlag = true;

boolean isOnlineOpen = true;

boolean isOfflineOpen = false;

String mac = "";

HashMap<String, MacDeviceBean> hashMap = new HashMap<>();

private TimerTask mTimerTask = null;

private Timer mTimer = null;

public static void start(Context context, String mac, boolean flag) {

Intent intent = new Intent(context, LinkDeviceActivity.class);

intent.putExtra("flag", flag);

intent.putExtra("mac", mac);

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_link\_device;

}

@Override

public void initData() {

if (getIntent() != null) {

isFlag = getIntent().getBooleanExtra("flag", true);

mac = getIntent().getStringExtra("mac");

}

}

@Override

public void initView() {

upOnLine();

upOffLine();

if (isFlag == false) {

bindingView.cvOnlineDevice.setVisibility(View.GONE);

bindingView.cvOfflineDevice.setVisibility(View.GONE);

bindingView.rvOffline.setVisibility(View.GONE);

bindingView.rvOnline.setVisibility(View.VISIBLE);

}

initAdapter();

initListener();

// basePresenter.getStationInfo(mac);

}

private void initListener() {

bindingView.cvOnlineDevice.setOnClickListener(view -> {

if (isOnlineOpen) {

isOnlineOpen = false;

} else {

isOnlineOpen = true;

}

upOnLine();

});

bindingView.cvOfflineDevice.setOnClickListener(view -> {

if (isOfflineOpen) {

isOfflineOpen = false;

} else {

isOfflineOpen = true;

}

upOffLine();

});

bindingView.ivBack.setOnClickListener(view -> {

this.finish();

});

mAdapter.setOnItemClickListener((adapter, view, position) ->

ConnectionDeviceActivity.start(LinkDeviceActivity.this, mOnList.get(position), mOnMacList.get(position))

);

mOffAdapter.setOnItemClickListener((adapter, view, position) -> {

ConnectionDeviceActivity.start(LinkDeviceActivity.this, mOffList.get(position), mOffMacList.get(position));

});

mOffAdapter.setOnItemLongClickListener((adapter, view, position) -> {

offDeviceDialog(position);

return true;

});

}

/\*\*

\* 确定删除此离线设备吗

\*/

public void offDeviceDialog(int position) {

new MessageDialog.Builder(this)

.setTitle(getResources().getString(R.string.prompt\_hint)) // 标题可以不用填写

.setMessage(getResources().getString(R.string.off\_device\_tips))

//.setConfirm("确认")

//.setCancel("取消") // 设置 null 表示不显示取消按钮

//.setAutoDismiss(false) // 设置点击按钮后不关闭对话框

.setListener(new MessageDialog.OnListener() {

@Override

public void onConfirm(Dialog dialog) {

mOffList.remove(position);

mOffAdapter.notifyDataSetChanged();

String mac = mOffMacList.get(position);

hashMap.put(mac, null);

SP.setDeviceMap(hashMap);

}

@Override

public void onCancel(Dialog dialog) {

}

})

.show();

}

@Override

protected void onResume() {

super.onResume();

basePresenter.getStationInfo(mac);

// startTimer();

}

@Override

protected void onPause() {

super.onPause();

// stopTimer();

}

private void stopTimer() {

if (basePresenter.disposable != null && !basePresenter.disposable.isDisposed()) {

basePresenter.disposable.dispose();

basePresenter.disposable = null;

}

if (mTimerTask != null) {

mTimerTask.cancel();

mTimerTask = null;

}

if (mTimer != null) {

mTimer.cancel();

mTimer = null;

}

}

private void startTimer() {

if (mTimer == null) {

mTimer = new Timer();

}

if (mTimerTask == null) {

mTimerTask = new TimerTask() {

@Override

public void run() {

basePresenter.getStationInfo(mac);

}

};

}

if (mTimer != null && mTimerTask != null) {

mTimer.schedule(mTimerTask, 0, 10000);

}

}

private void upOnLine() {

if (isOnlineOpen) {

bindingView.rvOnline.setVisibility(View.VISIBLE);

bindingView.ivOnline.setBackgroundResource(R.mipmap.arrow\_down);

} else {

bindingView.rvOnline.setVisibility(View.GONE);

bindingView.ivOnline.setBackgroundResource(R.mipmap.arrow\_right);

}

}

private void upOffLine() {

if (isOfflineOpen) {

bindingView.rvOffline.setVisibility(View.VISIBLE);

bindingView.ivOffline.setBackgroundResource(R.mipmap.arrow\_down);

} else {

bindingView.rvOffline.setVisibility(View.GONE);

bindingView.ivOffline.setBackgroundResource(R.mipmap.arrow\_right);

}

}

private void initAdapter() {

mAdapter = new ConnectedEquipmentAdapter(mOnList);

//布局管理器所需参数，上下文

LinearLayoutManager linearLayoutManager = new LinearLayoutManager(this);

bindingView.rvOnline.setLayoutManager(linearLayoutManager);

mAdapter.setOffLine(false);

bindingView.rvOnline.setAdapter(mAdapter);

//布局管理器所需参数，上下文

LinearLayoutManager linearLayoutManager01 = new LinearLayoutManager(this);

mOffAdapter = new ConnectedEquipmentAdapter(mOffList);

//布局管理器所需参数，上下文

bindingView.rvOffline.setLayoutManager(linearLayoutManager01);

mOffAdapter.setOffLine(true);

bindingView.rvOffline.setAdapter(mOffAdapter);

}

public void onDataSuccess(List<MacDeviceBean> onList, List<MacDeviceBean> offList

, List<String> onMacList, List<String> offMacList) {

this.mOnList = onList;

this.mOffList = offList;

this.mOnMacList = onMacList;

this.mOffMacList = offMacList;

bindingView.tvOnlineDevice.setText(getString(R.string.online\_unit, mOnList.size() + ""));

bindingView.tvOffDevice.setText(getString(R.string.device\_offline, mOffList.size() + ""));

mAdapter.setNewData(mOnList);

mAdapter.notifyDataSetChanged();

mOffAdapter.setNewData(mOffList);

mOffAdapter.notifyDataSetChanged();

}

}

package com.kingsignal.elf1.adapter;

import android.graphics.Bitmap;

import android.graphics.drawable.BitmapDrawable;

import android.graphics.drawable.Drawable;

import android.text.TextUtils;

import android.view.View;

import android.widget.ImageView;

import android.widget.TextView;

import androidx.annotation.Nullable;

import androidx.core.content.ContextCompat;

import com.chad.library.adapter.base.BaseQuickAdapter;

import com.chad.library.adapter.base.BaseViewHolder;

import com.kingsignal.common.http.response.MacDeviceBean;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.utils.StringUtils;

import com.kingsignal.elf1.utils.image.BitmapUtils;

import java.math.BigDecimal;

import java.util.HashMap;

import java.util.List;

/\*\*  
 \* @author zwy  
 \* create at 2020-12-10  
 \*/

public class ConnectedEquipmentAdapter extends BaseQuickAdapter<MacDeviceBean, BaseViewHolder> {

int[] a = {R.mipmap.terminal};

HashMap<String, Integer> hashMap = new HashMap<String, Integer>() {

};

private boolean isOffLine = false;

public boolean isOffLine() {

return isOffLine;

}

public void setOffLine(boolean offLine) {

isOffLine = offLine;

}

public ConnectedEquipmentAdapter(@Nullable List<MacDeviceBean> data) {

super(R.layout.item\_connect\_equipment, data);

hashMap.put("acer", R.mipmap.acer);

hashMap.put("apple", R.mipmap.apple);

hashMap.put("blackberry", R.mipmap.blackberry);

hashMap.put("dell", R.mipmap.dell);

hashMap.put("fujitsu", R.mipmap.fujitsu);

hashMap.put("google", R.mipmap.google);

hashMap.put("haier", R.mipmap.haier);

hashMap.put("huawei", R.mipmap.huawei);

hashMap.put("lenovo", R.mipmap.lenovo);

hashMap.put("htc", R.mipmap.htc);

hashMap.put("lg", R.mipmap.lg);

hashMap.put("motorola", R.mipmap.motorola);

hashMap.put("nokia", R.mipmap.nokia);

hashMap.put("oneplus", R.mipmap.oneplus);

hashMap.put("oppo", R.mipmap.oppo);

hashMap.put("philips", R.mipmap.philips);

hashMap.put("realtek", R.mipmap.realtek);

hashMap.put("samsung", R.mipmap.samsung);

hashMap.put("sharp", R.mipmap.sharp);

hashMap.put("smartisan", R.mipmap.smartisan);

hashMap.put("sony", R.mipmap.sony);

hashMap.put("tcl", R.mipmap.tcl);

hashMap.put("tenda", R.mipmap.tenda);

hashMap.put("terminal", R.mipmap.terminal);

hashMap.put("toshiba", R.mipmap.toshiba);

hashMap.put("tplink", R.mipmap.tplink);

hashMap.put("vivo", R.mipmap.vivo);

hashMap.put("vivotek", R.mipmap.vivotek);

hashMap.put("xiaomi", R.mipmap.xiaomi);

hashMap.put("zte", R.mipmap.zte);

}

@Override

protected void convert(BaseViewHolder helper, MacDeviceBean item) {

ImageView ivIcon = helper.itemView.findViewById(R.id.ivIcon);

TextView tvDeviceName = helper.itemView.findViewById(R.id.tvDeviceName);

TextView tvDeviceData = helper.itemView.findViewById(R.id.tvDeviceData);

ImageView ivSignalLevel = helper.itemView.findViewById(R.id.ivSignalLevel);

TextView tvSignalNumber = helper.itemView.findViewById(R.id.tvSignalNumber);

ImageView mIvLimit = helper.itemView.findViewById(R.id.ivLimit);

if (!TextUtils.isEmpty(item.getBrand())) {

Drawable drawable = mContext.getResources().getDrawable(hashMap.get(item.getBrand().toLowerCase()));

ivIcon.setBackground(drawable);

} else {

ivIcon.setBackgroundResource(R.mipmap.terminal);

}

if ("off".equals(item.getLimit\_network())) {

mIvLimit.setVisibility(View.GONE);

} else {

mIvLimit.setVisibility(View.VISIBLE);

if (item.getSpeed\_down() > 0 || item.getSpeed\_up() > 0) {

mIvLimit.setBackgroundResource(R.drawable.limit);

} else {

mIvLimit.setBackgroundResource(R.drawable.disenble);

}

}

if (isOffLine()) {

tvDeviceName.setTextColor(ContextCompat.getColor(mContext, R.color.text\_gray\_second));

tvDeviceData.setTextColor(ContextCompat.getColor(mContext, R.color.text\_gray\_second));

ivSignalLevel.setBackgroundResource(R.mipmap.signal0);

helper.setGone(R.id.ivDownload, false);

helper.setGone(R.id.tvSignalNumber, false);

if (!TextUtils.isEmpty(item.getRename())) {

tvDeviceName.setText(item.getRename());

} else {

tvDeviceName.setText(item.getHostname());

}

} else {

tvDeviceName.setTextColor(ContextCompat.getColor(mContext, R.color.black));

tvDeviceData.setTextColor(ContextCompat.getColor(mContext, R.color.black));

helper.setGone(R.id.ivDownload, true);

helper.setGone(R.id.tvSignalNumber, true);

if (!TextUtils.isEmpty(item.getRename())) {

tvDeviceName.setText(item.getRename() + " (" + item.getType() + ")");

} else {

tvDeviceName.setText(item.getHostname() + " (" + item.getType() + ")");

}

if(item.getSignal()>=-100&&item.getSignal()<=-81){

ivSignalLevel.setBackgroundResource(R.mipmap.signal1);

}else if(item.getSignal()>-81&&item.getSignal()<=-61){

ivSignalLevel.setBackgroundResource(R.mipmap.signal2);

}else if(item.getSignal()>-61&&item.getSignal()<=-41){

ivSignalLevel.setBackgroundResource(R.mipmap.signal3);

}else if(item.getSignal()>-41&&item.getSignal()<=-21){

ivSignalLevel.setBackgroundResource(R.mipmap.signal4);

}else {

ivSignalLevel.setBackgroundResource(R.mipmap.signal5);

}

tvDeviceData.setText(item.getLinktime());

tvSignalNumber.setText(StringUtils.rateChange(item.getRx\_rate()));

}

}

}

package com.kingsignal.elf1.presenter.settings;

import android.text.TextUtils;

import com.kingsignal.common.base.BasicPresenter;

import com.kingsignal.common.http.HttpCallBack;

import com.kingsignal.common.http.HttpConstant;

import com.kingsignal.common.http.HttpRequest;

import com.kingsignal.common.http.impl.HttpBaseCallBack;

import com.kingsignal.common.http.impl.HttpLoadingCallBack;

import com.kingsignal.common.http.response.MacDeviceBean;

import com.kingsignal.common.utils.LogUtil;

import com.kingsignal.common.utils.SP;

import com.kingsignal.elf1.entity.GuestWifiInfoBean;

import com.kingsignal.elf1.entity.LinkDeviceBean;

import com.kingsignal.elf1.ui.setting.LinkDeviceActivity;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.Set;

import io.reactivex.disposables.Disposable;

/\*\*  
 \* @author zwy  
 \* create at 2020-12-06  
 \*/

public class LinkDevicePresenter extends BasicPresenter<LinkDeviceActivity> {

private List<MacDeviceBean> mOnList = new ArrayList<>();

private List<MacDeviceBean> mOffList = new ArrayList<>();

private List<String> mOnMacList = new ArrayList<>();

private List<String> mOffMacList = new ArrayList<>();

public Disposable disposable;

public void getStationInfo(String mac) {

Map<String, String> map = new HashMap<>();

if (!TextUtils.isEmpty(mac)) {

map.put("mac", mac);

}

String url = HttpConstant.BASE\_PRE\_URL + getToken() + HttpConstant.GET\_STATION\_INFO;

disposable = HttpRequest.onGet(url, map, new HttpLoadingCallBack<LinkDeviceBean>(getBaseView()) {

@Override

public void onResponse(LinkDeviceBean response) {

super.onResponse(response);

SP.setDeviceMap(response.stations);

mOnList = new ArrayList<>();

mOffList = new ArrayList<>();

mOnMacList = new ArrayList<>();

mOffMacList = new ArrayList<>();

HashMap<String, MacDeviceBean> offMap = SP.getDeviceMap();

Iterator<Map.Entry<String, MacDeviceBean>> iterator = response.stations.entrySet().iterator();

while (iterator.hasNext()) {

Map.Entry<String, MacDeviceBean> entry = iterator.next();

mOnList.add(entry.getValue());

mOnMacList.add(entry.getKey());

if (offMap.containsKey(entry.getKey())) {

offMap.remove(entry.getKey());

}

}

Iterator<Map.Entry<String, MacDeviceBean>> itOff = offMap.entrySet().iterator();

while (itOff.hasNext()) {

Map.Entry<String, MacDeviceBean> entry = itOff.next();

mOffList.add(entry.getValue());

mOffMacList.add(entry.getKey());

}

if (checkAttach()) {

getBaseView().onDataSuccess(mOnList, mOffList, mOnMacList, mOffMacList);

}

}

@Override

public void onFailure(int code, String msg) {

super.onFailure(code, msg);

}

});

}

}

package com.kingsignal.elf1.ui.setting;

import android.content.Context;

import android.content.Intent;

import android.text.TextUtils;

import android.view.View;

import com.bigkoo.pickerview.view.OptionsPickerView;

import com.kingsignal.common.base.PresenterActivity;

import com.kingsignal.common.http.response.MacDeviceBean;

import com.kingsignal.common.utils.SP;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.databinding.ActivityConectionDeviceBinding;

import com.kingsignal.elf1.presenter.settings.ConnectionDevicePresenter;

import com.kingsignal.elf1.utils.KeyBoardUtils;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

/\*\*  
 \* @author zwy  
 \* create at 2020-11-07  
 \*/

public class ConnectionDeviceActivity extends PresenterActivity<ConnectionDevicePresenter, ActivityConectionDeviceBinding> {

private OptionsPickerView<String> internetPickerView;

private List<String> internetList = new ArrayList<>();

MacDeviceBean bean;

String mac;

String limitNetwork = "";

private HashMap<String, MacDeviceBean> paramMap = new HashMap<>();

public static void start(Context context, MacDeviceBean bean, String mac) {

Intent intent = new Intent(context, ConnectionDeviceActivity.class);

intent.putExtra("bean", bean);

intent.putExtra("mac", mac);

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_conection\_device;

}

@Override

public void initData() {

if (getIntent() != null) {

bean = getIntent().getParcelableExtra("bean");

mac = getIntent().getStringExtra("mac");

}

}

@Override

public void initView() {

bindingView.viewHeader.tvSave.setVisibility(View.GONE);

bindingView.viewHeader.tvTitle.setText(getString(R.string.connection\_device\_manager\_title));

initPickerView();

if (bean != null) {

bindingView.edtDeviceName.setText(bean.getRename());

if ("off".equals(bean.getLimit\_network())) {

bindingView.llRate.setVisibility(View.GONE);

bindingView.tvInternet.setText(internetList.get(0));

} else {

if (bean.getSpeed\_down() > 0 || bean.getSpeed\_up() > 0) {

bindingView.tvInternet.setText(internetList.get(2));

bindingView.llRate.setVisibility(View.VISIBLE);

bindingView.edtSpendUp.setText(bean.getSpeed\_up() + "");

bindingView.edtSpendDown.setText(bean.getSpeed\_down() + "");

} else {

bindingView.edtSpendUp.setText("0");

bindingView.edtSpendDown.setText("0");

bindingView.llRate.setVisibility(View.GONE);

bindingView.tvInternet.setText(internetList.get(1));

}

}

}

initListener();

}

public void initListener() {

bindingView.viewHeader.ivBack.setOnClickListener(view ->

ConnectionDeviceActivity.this.finish()

);

bindingView.tvNext.setOnClickListener(view -> {

KeyBoardUtils.hideInputForce(this);

if (internetList.get(0).equals(bindingView.tvInternet.getText().toString().trim())) {

limitNetwork = "off";

} else {

limitNetwork = "on";

}

String spendUp = bindingView.edtSpendUp.getText().toString().trim();

String spendDown = bindingView.edtSpendDown.getText().toString().trim();

String rename = bindingView.edtDeviceName.getText().toString().trim();

basePresenter.setStationInfo(mac, bean.getIp(), limitNetwork, spendUp, spendDown, rename);

});

bindingView.tvInternet.setOnClickListener(view -> {

KeyBoardUtils.hideInputForce(this);

if (internetPickerView != null) {

internetPickerView.show();

}

});

}

private void initPickerView() {

internetList.add(getString(R.string.device\_speed\_limit));

internetList.add(getString(R.string.device\_network\_access));

internetList.add(getString(R.string.device\_custom));

internetPickerView = genPickerBuilder(this, (options1, options2, options3, v) -> {

if (options1 == 2) {

bindingView.llRate.setVisibility(View.VISIBLE);

} else if (options1 == 1) {

bindingView.llRate.setVisibility(View.GONE);

bindingView.edtSpendUp.setText("0");

bindingView.edtSpendDown.setText("0");

} else {

bindingView.llRate.setVisibility(View.GONE);

}

bindingView.tvInternet.setText(internetList.get(options1));

}).build();

internetPickerView.setPicker(internetList);

}

public void onDataSuccess() {

String spendUp = bindingView.edtSpendUp.getText().toString().trim();

String spendDown = bindingView.edtSpendDown.getText().toString().trim();

String rename = bindingView.edtDeviceName.getText().toString().trim();

bean.setSpeed\_up(TextUtils.isEmpty(spendUp)?0:Integer.parseInt(spendUp));

bean.setSpeed\_down(TextUtils.isEmpty(spendDown)?0:Integer.parseInt(spendDown));

bean.setRename(rename);

bean.setLimit\_network(limitNetwork);

paramMap.put(mac, bean);

SP.setDeviceMap(paramMap);

showToast(getString(R.string.device\_update\_success));

this.finish();

}

}

package com.kingsignal.elf1.presenter.settings;

import android.text.TextUtils;

import com.kingsignal.common.base.BasicPresenter;

import com.kingsignal.common.http.HttpConstant;

import com.kingsignal.common.http.HttpRequest;

import com.kingsignal.common.http.impl.HttpLoadingCallBack;

import com.kingsignal.common.http.response.BasicResponse;

import com.kingsignal.common.http.response.MacDeviceBean;

import com.kingsignal.common.utils.SP;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.entity.LinkDeviceBean;

import com.kingsignal.elf1.ui.setting.ConnectedEquipmentActivity;

import com.kingsignal.elf1.ui.setting.ConnectionDeviceActivity;

import com.kingsignal.elf1.ui.setting.LinkDeviceActivity;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

/\*\*  
 \* @author zwy  
 \* create at 2020-11-07  
 \*/

public class ConnectionDevicePresenter extends BasicPresenter<ConnectionDeviceActivity> {

private List<MacDeviceBean> mOnList = new ArrayList<>();

private List<MacDeviceBean> mOffList = new ArrayList<>();

public void setStationInfo(String mac, String ip, String limit\_network

, String speed\_down, String speed\_up, String rename) {

Map<String, String> map = new HashMap<>();

if (TextUtils.isEmpty(rename)) {

getBaseView().showToast(getBaseView().getString(R.string.device\_name\_input));

return;

}

if (!TextUtils.isEmpty(mac)) {

map.put("mac", mac);

}

if (!TextUtils.isEmpty(ip)) {

map.put("ip", ip);

}

if (!TextUtils.isEmpty(limit\_network)) {

map.put("limit\_network", limit\_network);

}

if (!TextUtils.isEmpty(rename)) {

map.put("rename", rename);

}

map.put("speed\_down", speed\_down);

map.put("speed\_up", speed\_up);

String url = HttpConstant.BASE\_PRE\_URL + getToken() + HttpConstant.POST\_SET\_STATION\_INFO;

HttpRequest.onPost(url, map, new HttpLoadingCallBack<BasicResponse>(getBaseView()) {

@Override

public void onResponse(BasicResponse response) {

super.onResponse(response);

if (checkAttach()) {

getBaseView().onDataSuccess();

}

}

@Override

public void onFailure(int code, String msg) {

super.onFailure(code, msg);

}

});

}

}

package com.kingsignal.elf1.ui.setting.online;

import android.app.Dialog;

import android.content.Context;

import android.content.Intent;

import android.text.Editable;

import android.text.TextUtils;

import android.text.TextWatcher;

import android.view.View;

import android.view.WindowManager;

import com.kingsignal.common.base.PresenterActivity;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.databinding.ActivityWirelessSettingBinding;

import com.kingsignal.elf1.dialog.MessageDialog;

import com.kingsignal.elf1.entity.req.WifiInfoParam;

import com.kingsignal.elf1.presenter.settings.online.WirelessSettingPresenter;

import com.kingsignal.elf1.ui.setting.help.HelpContentActivity;

/\*\*  
 \* @author zwy  
 \* create at 2020-11-12  
 \*/

public class WirelessSettingActivity extends PresenterActivity<WirelessSettingPresenter, ActivityWirelessSettingBinding> {

private String ledSwitch;

private String disabled;

private String brand24 = "0";

private String brand5 = "0";

private String defaultBrand = "0";

private String hintValue;

WifiInfoParam infoBean;

private String brand24Name = "";

private String brand5Name = "";

public static void start(Context context) {

Intent intent = new Intent(context, WirelessSettingActivity.class);

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_wireless\_setting;

}

@Override

public void initData() {

}

@Override

public void initView() {

basePresenter.getWifiInfo();

bindingView.viewHeader.tvTitle.setText(R.string.wireless\_control);

bindingView.viewHeader.ivHelp.setVisibility(View.VISIBLE);

initListener();

}

public void initListener() {

bindingView.viewHeader.ivBack.setOnClickListener(view -> WirelessSettingActivity.this.finish());

bindingView.viewHeader.ivHelp.setOnClickListener(view -> {

HelpContentActivity.start(this,getString(R.string.answer\_function\_wifi));

});

bindingView.tvNext.setOnClickListener(view -> {

if ("1".equals(brand24) || "1".equals(brand5) || "1".equals(defaultBrand)) {

operateParamDialog();

} else {

setWifiNetWork();

}

});

bindingView.ivOpen.setOnClickListener(view -> {

if ("0".equals(ledSwitch)) {

ledSwitch = "1";

} else {

ledSwitch = "0";

}

brand24 = "0";

brand5 = "0";

defaultBrand = "0";

openSwitch(true);

bindingView.cv.setOnClickListener(view -> {

});

bindingView.edtName24.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

}

@Override

public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {

if (!TextUtils.isEmpty(charSequence)) {

brand24Name = charSequence.toString();

}

}

@Override

public void afterTextChanged(Editable editable) {

}

});

bindingView.edtName.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

}

@Override

public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {

if (!TextUtils.isEmpty(charSequence)) {

brand24Name = charSequence.toString();

}

}

@Override

public void afterTextChanged(Editable editable) {

}

});

}

public void setWifiNetWork() {

basePresenter.setWifi(ledSwitch, brand24, brand5, defaultBrand, bindingView.edtName.getText().toString()

, bindingView.edtPwd.getText().toString().trim(), bindingView.edtName24.getText().toString().trim()

, bindingView.edtPwd24.getText().toString().trim(), bindingView.edtName5.getText().toString().trim()

, bindingView.edtPwd5.getText().toString().trim());

}

/\*\*

\* 弹框提示

\*/

public void operateParamDialog() {

new MessageDialog.Builder(this)

.setTitle(getResources().getString(R.string.prompt\_hint)) // 标题可以不用填写

.setMessage(getResources().getString(R.string.dialog\_tips))

.setListener(new MessageDialog.OnListener() {

@Override

public void onConfirm(Dialog dialog) {

setWifiNetWork();

}

@Override

public void onCancel(Dialog dialog) {

}

})

.show();

}

public void openSwitch(boolean isType) {

brand24Name = bindingView.edtName24.getText().toString().trim();

if ("1".equals(ledSwitch)) {

bindingView.ivOpen.setBackgroundResource(R.mipmap.swithch\_on);

bindingView.cvHint.setVisibility(View.VISIBLE);

bindingView.llIntegration.setVisibility(View.GONE);

bindingView.llName.setVisibility(View.VISIBLE);

bindingView.llPwd.setVisibility(View.VISIBLE);

} else {

bindingView.ivOpen.setBackgroundResource(R.mipmap.switch\_off);

bindingView.cvHint.setVisibility(View.GONE);

bindingView.llIntegration.setVisibility(View.VISIBLE);

bindingView.llName.setVisibility(View.GONE);

bindingView.llPwd.setVisibility(View.GONE);

}

if (isType) {

bindingView.edtName.setText(brand24Name);

bindingView.edtName24.setText(brand24Name);

bindingView.edtName5.setText(brand24Name + "\_5G");

}

}

public void openDefaultSwitch() {

if ("0".equals(defaultBrand)) {

// bindingView.ivBrandDefault.setBackgroundResource(R.mipmap.swithch\_on);

bindingView.llName.setVisibility(View.VISIBLE);

bindingView.llPwd.setVisibility(View.VISIBLE);

} else {

// bindingView.ivBrandDefault.setBackgroundResource(R.mipmap.switch\_off);

bindingView.llName.setVisibility(View.GONE);

bindingView.llPwd.setVisibility(View.GONE);

}

}

public void openBrand2Switch() {

if ("0".equals(brand24)) {

// bindingView.ivBrand2.setBackgroundResource(R.mipmap.swithch\_on);

bindingView.viewName24.setVisibility(View.VISIBLE);

bindingView.viewPwd24.setVisibility(View.VISIBLE);

} else {

// bindingView.ivBrand2.setBackgroundResource(R.mipmap.switch\_off);

bindingView.viewName24.setVisibility(View.GONE);

bindingView.viewPwd24.setVisibility(View.GONE);

}

}

public void openBrand5Switch() {

if ("0".equals(brand5)) {

// bindingView.ivBrand5.setBackgroundResource(R.mipmap.swithch\_on);

bindingView.viewName5.setVisibility(View.VISIBLE);

bindingView.viewPwd5.setVisibility(View.VISIBLE);

} else {

// bindingView.ivBrand5.setBackgroundResource(R.mipmap.switch\_off);

bindingView.viewName5.setVisibility(View.GONE);

bindingView.viewPwd5.setVisibility(View.GONE);

}

}

public void onDataSuccess(WifiInfoParam wifiInfoBean) {

infoBean = wifiInfoBean;

disabled = wifiInfoBean.get\_$2g\_disabled();

ledSwitch = wifiInfoBean.getBand\_steering();

brand24 = wifiInfoBean.get\_$2g\_disabled();

brand5 = wifiInfoBean.get\_$5g\_disabled();

defaultBrand = wifiInfoBean.get\_$2g\_disabled();

openSwitch(false);

openBrand2Switch();

openBrand5Switch();

openDefaultSwitch();

brand24Name = wifiInfoBean.get\_$2g\_ssid();

bindingView.edtName24.setText(wifiInfoBean.get\_$2g\_ssid());

bindingView.edtPwd24.setText(wifiInfoBean.get\_$2g\_key());

bindingView.edtName5.setText(wifiInfoBean.get\_$5g\_ssid());

bindingView.edtPwd5.setText(wifiInfoBean.get\_$5g\_key());

bindingView.edtName.setText(wifiInfoBean.get\_$2g\_ssid());

bindingView.edtPwd.setText(wifiInfoBean.get\_$2g\_key());

}

@Override

protected void onPause() {

super.onPause();

if (basePresenter.disposable != null && !basePresenter.disposable.isDisposed()) {

basePresenter.disposable.dispose();

}

}

}

package com.kingsignal.elf1.ui.setting;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

import android.os.Handler;

import android.os.Message;

import android.view.View;

import com.bigkoo.pickerview.view.OptionsPickerView;

import com.kingsignal.common.base.PresenterActivity;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.databinding.ActivityGustModeBinding;

import com.kingsignal.elf1.entity.GuestWifiInfoBean;

import com.kingsignal.elf1.presenter.settings.GustModePresenter;

import com.kingsignal.elf1.ui.guide.NetworkDetectionActivity;

import com.kingsignal.elf1.ui.setting.help.HelpContentActivity;

import java.lang.ref.WeakReference;

import java.util.ArrayList;

import java.util.List;

/\*\*  
 \* @author zwy  
 \* create at 2020-12-16  
 \*/

public class GustModeActivity extends PresenterActivity<GustModePresenter, ActivityGustModeBinding> {

private OptionsPickerView<String> internetPickerView;

private List<String> internetList = new ArrayList<>();

private String gustWifiTimeLong = "";//开启时长

boolean isOpen;

boolean isNetworkOpen;

MyHandler handler;

public static void start(Context context) {

Intent intent = new Intent(context, GustModeActivity.class);

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_gust\_mode;

}

@Override

public void initData() {

}

@Override

public void initView() {

bindingView.viewHeader.tvTitle.setText(getString(R.string.guest\_mode));

bindingView.viewHeader.ivHelp.setVisibility(View.VISIBLE);

internetList.add(getResources().getString(R.string.hour, "4"));

internetList.add(getResources().getString(R.string.hour, "24"));

internetList.add(getResources().getString(R.string.all\_the\_time));

basePresenter.getGustWifiInfo();

initPickerView();

initListener();

}

public void initListener() {

bindingView.viewHeader.ivBack.setOnClickListener(view -> {

GustModeActivity.this.finish();

});

bindingView.viewHeader.ivHelp.setOnClickListener(view -> {

HelpContentActivity.start(this, getString(R.string.answer\_function\_gust));

});

bindingView.clUpdate.setOnClickListener(view -> {

if (internetPickerView != null) {

internetPickerView.show();

}

});

bindingView.ivOpen.setOnClickListener(view -> {

if (isOpen) {

bindingView.ivOpen.setImageResource(R.mipmap.switch\_off);

} else {

bindingView.ivOpen.setImageResource(R.mipmap.swithch\_on);

}

isOpen = !isOpen;

setVisibleView(isOpen);

});

bindingView.tvNext.setOnClickListener(view -> {

String disable = isOpen ? "0" : "1";

String name = bindingView.edtName.getText().toString().trim();

String pwd = bindingView.edtPwd.getText().toString().trim();

basePresenter.setGuestWifi(disable, name, pwd, gustWifiTimeLong, isNetworkOpen);

});

bindingView.ivNetworkOpen.setOnClickListener(view -> {

if (isNetworkOpen) {

bindingView.ivNetworkOpen.setImageResource(R.mipmap.switch\_off);

} else {

bindingView.ivNetworkOpen.setImageResource(R.mipmap.swithch\_on);

}

isNetworkOpen = !isNetworkOpen;

setNetworkVisibleView(isNetworkOpen);

});

}

private void initPickerView() {

internetPickerView = genPickerBuilder(this, (options1, options2, options3, v) -> {

bindingView.tvValidTime.setText(internetList.get(options1));

if (options1 == 0) {

gustWifiTimeLong = "4h";

} else if (options1 == 1) {

gustWifiTimeLong = "24h";

} else {

gustWifiTimeLong = "unlimit";

}

}).build();

internetPickerView.setPicker(internetList);

}

public void onDataSuccess(GuestWifiInfoBean.WifiInfoBean wifiInfoBean) {

if ("0".equals(wifiInfoBean.getDisabled())) {//开启

isOpen = true;

setVisibleView(true);

bindingView.ivOpen.setBackgroundResource(R.mipmap.swithch\_on);

} else {

setVisibleView(false);

isOpen = false;

bindingView.ivOpen.setBackgroundResource(R.mipmap.switch\_off);

}

if ("1".equals(wifiInfoBean.getEncrypt\_switch())) {//开启

isNetworkOpen = true;

bindingView.ivNetworkOpen.setBackgroundResource(R.mipmap.swithch\_on);

} else {

isNetworkOpen = false;

bindingView.ivNetworkOpen.setBackgroundResource(R.mipmap.switch\_off);

}

setNetworkVisibleView(isNetworkOpen);

bindingView.edtName.setText(wifiInfoBean.getSsid());

bindingView.edtPwd.setText(wifiInfoBean.getKey());

if ("4h".equals(wifiInfoBean.getTimelong())) {

gustWifiTimeLong = "4h";

bindingView.tvValidTime.setText(internetList.get(0));

} else if ("24h".equals(wifiInfoBean.getTimelong())) {

gustWifiTimeLong = "24h";

bindingView.tvValidTime.setText(internetList.get(1));

} else {

gustWifiTimeLong = "unlimit";

bindingView.tvValidTime.setText(internetList.get(2));

}

}

private void setVisibleView(boolean isFlag) {

if (isFlag) {

bindingView.llPwd.setVisibility(View.VISIBLE);

} else {

bindingView.llPwd.setVisibility(View.GONE);

}

}

private void setNetworkVisibleView(boolean isFlag) {

if (isFlag) {

bindingView.view03.setVisibility(View.VISIBLE);

} else {

bindingView.view03.setVisibility(View.GONE);

}

}

public void run() {

NetworkDetectionActivity.start(this, "");

this.finish();

}

public void onSetWifiInfoSuccess() {

if (isOpen) {

showToast(R.string.gust\_mode\_tip);

} else {

showToast(R.string.gust\_mode\_close\_tip);

}

handler = new MyHandler(this);

handler.sendEmptyMessageDelayed(1, 2000);

}

static class MyHandler extends Handler {

// SoftReference<Activity> 也可以使用软应用 只有在内存不足的时候才会被回收

private final WeakReference<GustModeActivity> mActivity;

private MyHandler(GustModeActivity activity) {

mActivity = new WeakReference<>(activity);

}

@Override

public void handleMessage(Message msg) {

Activity activity = mActivity.get();

if (activity != null) {

mActivity.get().run();

}

super.handleMessage(msg);

}

}

@Override

protected void onPause() {

super.onPause();

if (basePresenter.disposable != null && !basePresenter.disposable.isDisposed()) {

basePresenter.disposable.dispose();

}

}

}

package com.kingsignal.elf1.ui.setting.parentalcontrol;

import android.app.Dialog;

import android.content.Context;

import android.content.Intent;

import android.view.LayoutInflater;

import android.view.View;

import android.widget.TextView;

import androidx.recyclerview.widget.LinearLayoutManager;

import com.kingsignal.common.base.PresenterActivity;

import com.kingsignal.common.utils.SP;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.adapter.ParentalControlAdapter;

import com.kingsignal.elf1.databinding.ActivityParentalControlBinding;

import com.kingsignal.elf1.dialog.MessageDialog;

import com.kingsignal.elf1.entity.ControlBean;

import com.kingsignal.elf1.presenter.parentalcontrol.ParentalControlPresenter;

import com.kingsignal.elf1.presenter.parentalcontrol.ParentalControlView;

import com.kingsignal.elf1.ui.guide.NetworkDetectionActivity;

import com.kingsignal.elf1.ui.setting.help.HelpContentActivity;

import java.util.ArrayList;

import java.util.List;

/\*\*  
 \* @author zwy  
 \* create at 2020-12-16

\*/

public class ParentalControlActivity extends PresenterActivity<ParentalControlPresenter, ActivityParentalControlBinding>

implements ParentalControlView, ParentalControlAdapter.DelClickListener {

private ParentalControlAdapter mAdapter;

private List<ControlBean> mList = new ArrayList<>();

public static void start(Context context) {

Intent intent = new Intent(context, ParentalControlActivity.class);

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_parental\_control;

}

@Override

public void initData() {

basePresenter.setParentalSwitch();

}

@Override

public void initView() {

bindingView.viewHeader.tvTitle.setText(getString(R.string.parental\_control));

bindingView.viewHeader.ivHelp.setVisibility(View.VISIBLE);

initAdapter();

iniListener();

}

private void initAdapter() {

mAdapter = new ParentalControlAdapter(mList);

mAdapter.setListener(this);

View view = LayoutInflater.from(this).inflate(R.layout.include\_empty, null);

TextView tvEmptyName = view.findViewById(R.id.tvEmptyName);

tvEmptyName.setText(getString(R.string.no\_data));

mAdapter.setEmptyView(view);

bindingView.recyclerView01.setLayoutManager(new LinearLayoutManager(this));

bindingView.recyclerView01.setAdapter(mAdapter);

}

public void iniListener() {

bindingView.viewHeader.ivBack.setOnClickListener(view -> {

this.finish();

});

bindingView.tvNext.setOnClickListener(view -> {

ParentalControlNewGroupActivity.start(this);

});

mAdapter.setOnItemClickListener((adapter, view, position) -> {

ControlBean bean = mList.get(position);

ParentalControlNewGroupActivity.start(ParentalControlActivity.this, bean);

});

bindingView.viewHeader.ivHelp.setOnClickListener(view -> {

HelpContentActivity.start(this, getString(R.string.answer\_function\_parental));

});

}

@Override

protected void onResume() {

super.onResume();

basePresenter.getParentalCtrl();

}

@Override

public void onDataSuccess(List<ControlBean> list) {

mList = list;

mAdapter.setNewData(mList);

mAdapter.notifyDataSetChanged();

}

@Override

public void onDataFail() {

}

public void run() {

NetworkDetectionActivity.start(this, "");

this.finish();

}

@Override

public void onSetSuccess() {

showToast(getString(R.string.delete\_success));

basePresenter.getParentalCtrl();

}

public void onSetParentalSuccess() {

}

@Override

public void delClick(int position) {

deleteConfigureDialog(position);

}

public void deleteConfigureDialog(int position) {

new MessageDialog.Builder(this)

.setTitle(getResources().getString(R.string.prompt\_hint))

.setMessage(getResources().getString(R.string.delete\_configure))

//.setConfirm("确认")

//.setCancel("取消") // 设置 null 表示不显示取消按钮

//.setAutoDismiss(false) // 设置点击按钮后不关闭对话框

.setListener(new MessageDialog.OnListener() {

@Override

public void onConfirm(Dialog dialog) {

ControlBean bean = mList.get(position);

basePresenter.setParentalCtrl(bean, "del");

bindingView.recyclerView01.closeMenu();

}

@Override

public void onCancel(Dialog dialog) {

}

})

.show();

}

@Override

protected void onPause() {

super.onPause();

if (basePresenter.disposable != null && !basePresenter.disposable.isDisposed()) {

basePresenter.disposable.dispose();

}

}

}

package com.kingsignal.elf1.adapter;

import androidx.annotation.Nullable;

import com.chad.library.adapter.base.BaseQuickAdapter;

import com.chad.library.adapter.base.BaseViewHolder;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.entity.ControlBean;

import java.util.List;

/\*\*  
 \* @author zwy  
 \* create at 2020-12-16

\*/

public class ParentalControlAdapter extends BaseQuickAdapter<ControlBean, BaseViewHolder> {

private DelClickListener listener;

public interface DelClickListener {

void delClick(int position);

}

public void setListener(DelClickListener listener) {

this.listener = listener;

}

public ParentalControlAdapter(@Nullable List<ControlBean> data) {

super(R.layout.item\_parental\_control, data);

}

@Override

protected void convert(BaseViewHolder helper, ControlBean item) {

String startTime = item.getStart\_time().substring(0, item.getStart\_time().lastIndexOf(":"));

String entTime = item.getStop\_time().substring(0, item.getStop\_time().lastIndexOf(":"));

helper.setText(R.id.tvControlTime, startTime + " - " + entTime);

helper.setText(R.id.tvDeviceName, item.getName());

String[] macDevice = item.getClient\_macs().split(";");

helper.setText(R.id.tvControlEquipment, mContext.getResources().getString(R.string.control\_device, macDevice.length + ""));

char[] weekDays = item.getWeekdays().toCharArray();

StringBuilder weekStr = new StringBuilder();

String[] weekList = mContext.getResources().getStringArray(R.array.week\_value\_list);

int count = 0;

for (int i = 0; i < weekDays.length; i++) {

if ('1' == weekDays[i]) {

count++;

weekStr.append(weekList[i] + ",");

}

}

if (weekStr.length() > 1) {

weekStr.deleteCharAt(weekStr.length() - 1);

}

if (count == 6) {

helper.setText(R.id.tvControlWeek, mContext.getString(R.string.every\_day));

} else {

helper.setText(R.id.tvControlWeek, weekStr.toString());

}

helper.itemView.findViewById(R.id.ivDelete).setOnClickListener(view -> {

if (listener != null) {

listener.delClick(helper.getAdapterPosition());

}

});

helper.itemView.findViewById(R.id.tvDelete).setOnClickListener(view -> {

if (listener != null) {

listener.delClick(helper.getAdapterPosition());

}

});

}

}

package com.kingsignal.elf1.ui.setting.parentalcontrol;

import android.content.Context;

import android.content.Intent;

import android.graphics.Color;

import android.text.TextUtils;

import androidx.annotation.Nullable;

import com.bigkoo.pickerview.builder.TimePickerBuilder;

import com.bigkoo.pickerview.view.TimePickerView;

import com.google.gson.Gson;

import com.google.gson.reflect.TypeToken;

import com.kingsignal.common.base.PresenterActivity;

import com.kingsignal.common.http.response.MacDeviceBean;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.databinding.ActivityParentalControlNewGroupBinding;

import com.kingsignal.elf1.entity.ControlBean;

import com.kingsignal.elf1.presenter.parentalcontrol.ParentalControlPresenter;

import com.kingsignal.elf1.presenter.parentalcontrol.ParentalControlView;

import com.kingsignal.elf1.utils.KeyBoardUtils;

import com.kingsignal.elf1.utils.TimeZoneUtils;

import com.kingsignal.elf1.utils.UIUtil;

import java.util.ArrayList;

import java.util.List;

/\*\*  
 \* @author zwy  
 \* create at 2020-12-16

\*/

public class ParentalControlNewGroupActivity extends PresenterActivity<ParentalControlPresenter, ActivityParentalControlNewGroupBinding>

implements ParentalControlView {

TimePickerView tpvTime;

int selectTime = 0;//0,起始时间 1，结束时间

long startTime = 0;

long endTime = 0;

String weekDay;

List<MacDeviceBean> macList = new ArrayList<>();

private ControlBean controlBean;

String operate = "add";

public static void start(Context context) {

Intent intent = new Intent(context, ParentalControlNewGroupActivity.class);

context.startActivity(intent);

}

public static void start(Context context, ControlBean bean) {

Intent intent = new Intent(context, ParentalControlNewGroupActivity.class);

intent.putExtra("controlBean", bean);

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_parental\_control\_new\_group;

}

@Override

public void initData() {

if (getIntent() != null) {

controlBean = getIntent().getParcelableExtra("controlBean");

}

}

@Override

public void initView() {

initPop();

bindingView.viewHeader.tvTitle.setText(getString(R.string.new\_group));

setViewData();

initListener();

}

public void setViewData() {

if (controlBean != null) {

operate = "mod";

bindingView.edtGroupName.setText(controlBean.getName());

bindingView.tvStartTime.setText(controlBean.getStart\_time());

bindingView.tvEndTime.setText(controlBean.getStop\_time());

char[] weekList = controlBean.getWeekdays().toCharArray();

String[] weekValueList = getResources().getStringArray(R.array.week\_value\_list);

StringBuilder weekStr = new StringBuilder();

for (int i = 0; i < weekList.length; i++) {

if ('1' == weekList[i]) {

weekStr.append(weekValueList[i] + ",");

}

}

weekStr.deleteCharAt(weekStr.length() - 1);

bindingView.tvRepeat.setText(weekStr);

String[] macs = controlBean.getClient\_macs().split(";");

if (macs.length == 0 && !TextUtils.isEmpty(controlBean.getClient\_macs())) {

macs[0] = controlBean.getClient\_macs();

}

bindingView.tvAccessDevice.setText(getString(R.string.control\_device, macs.length + ""));

MacDeviceBean macDeviceBean = null;

for (int i = 0; i < macs.length; i++) {

macDeviceBean = new MacDeviceBean();

macDeviceBean.setMac(macs[i]);

macList.add(macDeviceBean);

}

}

}

public void initListener() {

bindingView.viewHeader.ivBack.setOnClickListener(view -> {

this.finish();

});

bindingView.tvStartTime.setOnClickListener(view -> {

KeyBoardUtils.hideInputForce(this);

selectTime = 0;

tpvTime.show();

});

bindingView.tvEndTime.setOnClickListener(view -> {

KeyBoardUtils.hideInputForce(this);

selectTime = 1;

tpvTime.show();

});

bindingView.clRepeat.setOnClickListener(view -> {

KeyBoardUtils.hideInputForce(this);

String weekStr = bindingView.tvRepeat.getText().toString().trim();

ParentalWeekActivity.start(this, weekStr);

});

bindingView.clAssessDevice.setOnClickListener(view -> {

KeyBoardUtils.hideInputForce(this);

String json = new Gson().toJson(macList);

ParentalModeAccessActivity.start(this, json);

});

bindingView.tvNext.setOnClickListener(view -> {

ControlBean bean = new ControlBean();

String name = bindingView.edtGroupName.getText().toString().trim();

String startTime = bindingView.tvStartTime.getText().toString().trim();

String endTime = bindingView.tvEndTime.getText().toString().trim();

String repeat = bindingView.tvRepeat.getText().toString().trim();

if ("mod".equals(operate)) {

bean = controlBean;

} else {

bean = new ControlBean();

bean.setNetwork\_enable("0");

}

bean.setName(name);

bean.setStart\_time(TimeZoneUtils.getTimeToSecond(startTime));

bean.setStop\_time(TimeZoneUtils.getTimeToSecond(endTime));

if (!TextUtils.isEmpty(repeat)) {

bean.setIsweekly("1");

}

String[] week = new String[]{};

if (!TextUtils.isEmpty(repeat)) {

week = repeat.split(",");

}

String[] weekValueList = getResources().getStringArray(R.array.week\_value\_list);

StringBuilder weekValue = new StringBuilder();

boolean isType;

for (int i = 0; i < weekValueList.length; i++) {

isType = false;

for (int j = 0; j < week.length; j++) {

if (weekValueList[i].equals(week[j])) {

isType = true;

}

}

if (isType) {

weekValue.append("1");

} else {

weekValue.append("0");

}

}

bean.setWeekdays(weekValue.toString());

StringBuilder deviceStr = new StringBuilder();

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

deviceStr.append(macList.get(i).getMac() + ";");

}

if (deviceStr.length() > 1) {

deviceStr.deleteCharAt(deviceStr.length() - 1);

}

bean.setClient\_macs(deviceStr.toString());

basePresenter.setParentalCtrl(bean, operate);

});

}

/\*\*

\* 初始化弹框

\*/

private void initPop() {

tpvTime = new TimePickerBuilder(this, (date, v) -> {//选中事件回调

if (selectTime == 0) {

if (endTime != 0 && endTime < date.getTime()) {

showToast(getResources().getString(R.string.time\_error));

return;

}

startTime = date.getTime();

bindingView.tvStartTime.setText(UIUtil.getTime(date));

} else if (selectTime == 1) {

if (startTime != 0 && startTime > date.getTime()) {

showToast(getResources().getString(R.string.time\_error));

return;

}

endTime = date.getTime();

bindingView.tvEndTime.setText(UIUtil.getTime(date));

}

})

.setType(new boolean[]{false, false, false, true, true, false})// 默认全部显示

.setCancelText(getResources().getString(R.string.cancel))//取消按钮文字

.setSubmitText(getResources().getString(R.string.ok))//确认按钮文字

.setSubCalSize(16)//滚轮文字大小

.setTitleSize(16)//标题文字大小

.setContentTextSize(14)

.setTitleText(getResources().getString(R.string.select\_time))//标题文字

.setOutSideCancelable(false)//点击屏幕，点在控件外部范围时，是否取消显示

.isCyclic(true)//是否循环滚动

.setTitleColor(getResources().getColor(R.color.text\_gray\_second))//标题文字颜色

.setSubmitColor(getResources().getColor(com.kingsignal.common.R.color.black))//确定按钮文字颜色

.setCancelColor(getResources().getColor(com.kingsignal.common.R.color.black))//取消按钮文字颜色

.setLabel("", "", "", "", "", "")//默认设置为年月日时分秒

.isCenterLabel(false) //是否只显示中间选中项的label文字，false则每项item全部都带有label。

.build();

}

@Override

public void onDataSuccess(List<ControlBean> controlBeanList) {

}

@Override

public void onDataFail() {

}

@Override

public void onSetSuccess() {

if ("add".equals(operate)) {

showToast(getString(R.string.add\_success));

} else {

showToast(getString(R.string.device\_update\_success));

}

ParentalControlNewGroupActivity.this.finish();

}

@Override

protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (resultCode == 200) {

if (data != null) {

weekDay = data.getStringExtra("weekDay");

bindingView.tvRepeat.setText(weekDay);

}

} else if (resultCode == 300) {

if (data != null) {

String device = data.getStringExtra("device");

Gson gson = new Gson();

macList = gson.fromJson(device, new TypeToken<List<MacDeviceBean>>() {

}.getType());

int count = 0;

if (macList != null) {

count = macList.size();

}

bindingView.tvAccessDevice.setText(getString(R.string.control\_device, count + ""));

}

}

}

}

package com.kingsignal.elf1.ui.setting.parentalcontrol;

import android.app.Activity;

import android.content.Intent;

import android.text.TextUtils;

import android.view.KeyEvent;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.LinearLayout;

import android.widget.TextView;

import androidx.constraintlayout.widget.ConstraintLayout;

import androidx.recyclerview.widget.LinearLayoutManager;

import com.google.gson.Gson;

import com.google.gson.reflect.TypeToken;

import com.kingsignal.common.base.PresenterActivity;

import com.kingsignal.common.http.response.MacDeviceBean;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.adapter.ParentalModeAccessAdapter;

import com.kingsignal.elf1.databinding.ActivityParentalModeAssessBinding;

import com.kingsignal.elf1.presenter.parentalcontrol.ParentalModeAccessPresenter;

import java.util.ArrayList;

import java.util.List;

/\*\*  
 \* @author zwy  
 \* create at 2020-12-18

\*/

public class ParentalModeAccessActivity extends PresenterActivity<ParentalModeAccessPresenter, ActivityParentalModeAssessBinding>

implements ParentalModeAccessAdapter.OperateClickListener {

private ParentalModeAccessAdapter mAdapter;

private ParentalModeAccessAdapter mNotAdapter;

private List<MacDeviceBean> mList = new ArrayList<>();

private List<MacDeviceBean> mNotList = new ArrayList<>();

List<MacDeviceBean> macList;

public static void start(Activity context, String controlDevice) {

Intent intent = new Intent(context, ParentalModeAccessActivity.class);

intent.putExtra("controlDevice", controlDevice);

context.startActivityForResult(intent, 100);

}

@Override

public int getLayoutId() {

return R.layout.activity\_parental\_mode\_assess;

}

@Override

public void initData() {

if (getIntent() != null) {

String controlDevice = getIntent().getStringExtra("controlDevice");

if (controlDevice != null) {

macList = new Gson().fromJson(controlDevice, new TypeToken<List<MacDeviceBean>>() {

}.getType());

}

}

}

@Override

public void initView() {

bindingView.viewHeader.tvTitle.setText(getString(R.string.parental\_device\_manager));

basePresenter.getStationInfo();

initAdapter();

initListener();

}

private void initAdapter() {

View view = LayoutInflater.from(this).inflate(R.layout.include\_parental\_empty, null);

TextView tvEmptyName = view.findViewById(R.id.tvEmptyName);

tvEmptyName.setText(getString(R.string.not\_data\_device));

View view01 = LayoutInflater.from(this).inflate(R.layout.include\_parental\_empty, null);

TextView tvEmptyName01 = view01.findViewById(R.id.tvEmptyName);

tvEmptyName01.setText(getString(R.string.no\_data));

mAdapter = new ParentalModeAccessAdapter(mList);

mAdapter.setOperate(0);

mAdapter.setEmptyView(view);

bindingView.rvAddDevice.setLayoutManager(new LinearLayoutManager(this));

bindingView.rvAddDevice.setAdapter(mAdapter);

mNotAdapter = new ParentalModeAccessAdapter(mNotList);

mNotAdapter.setOperate(1);

mNotAdapter.setEmptyView(view01);

bindingView.rvNotAddDevice.setLayoutManager(new LinearLayoutManager(this));

bindingView.rvNotAddDevice.setAdapter(mNotAdapter);

}

public void initListener() {

bindingView.viewHeader.ivBack.setOnClickListener(view ->

backValueActivity()

);

mAdapter.setListener(this);

mNotAdapter.setListener(this);

}

public void onDataSuccess(List<MacDeviceBean> onList) {

mNotList = onList;

int count;

for (int j=0;j<macList.size();j++){

count=0;

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

if (onList.get(i).getMac().toLowerCase().equals(macList.get(j).getMac().toLowerCase())) {

mList.add(onList.get(i));

mNotList.remove(i);

break;

}else{

count++;

}

}

if(count==onList.size()){

mList.add(macList.get(j));

}

}

mNotAdapter.setNewData(mNotList);

mNotAdapter.notifyDataSetChanged();

mAdapter.setNewData(mList);

mAdapter.notifyDataSetChanged();

}

@Override

public void operateClick(int position, int op) {

if (op == 1) {

mList.add(mNotList.get(position));

mNotList.remove(mNotList.get(position));

} else {

mNotList.add(mList.get(position));

mList.remove(mList.get(position));

}

mAdapter.setNewData(mList);

mNotAdapter.setNewData(mNotList);

mAdapter.notifyDataSetChanged();

mNotAdapter.notifyDataSetChanged();

}

public void backValueActivity() {

String json = new Gson().toJson(mList);

getIntent().putExtra("device", json);

this.setResult(300, getIntent());

this.finish();

}

@Override

public boolean onKeyDown(int keyCode, KeyEvent event) {

if (event.getKeyCode() == KeyEvent.KEYCODE\_BACK) {

backValueActivity();

return true;

} else {

return super.dispatchKeyEvent(event);

}

}

}

package com.kingsignal.elf1.ui.setting.online;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

import android.os.Handler;

import android.os.Message;

import android.view.View;

import com.bigkoo.pickerview.view.OptionsPickerView;

import com.kingsignal.common.base.PresenterActivity;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.bean.WanConfigBean;

import com.kingsignal.elf1.bean.WanInfoBean;

import com.kingsignal.elf1.databinding.ActivityInternetSettingsBinding;

import com.kingsignal.elf1.presenter.settings.online.MainInternetSettingPresenter;

import com.kingsignal.elf1.ui.setting.help.HelpContentActivity;

import com.kingsignal.elf1.utils.KeyBoardUtils;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* 网络设置

\* @author zwy

\* create at 2020-12-18

\*/

public class MainInternetSettingActivity extends PresenterActivity<MainInternetSettingPresenter, ActivityInternetSettingsBinding> {

private OptionsPickerView<String> internetPickerView;

private List<String> internetList = new ArrayList<>();

WanConfigBean wanInfoParam;

int selectIndex = 0;

WanConfigBean wanInfoBean;

public static void start(Context context) {

Intent intent = new Intent(context, MainInternetSettingActivity.class);

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_internet\_settings;

}

@Override

public void initData() {

}

@Override

public void initView() {

bindingView.viewHeader.tvTitle.setText(getString(R.string.internet\_title));

bindingView.tvNext.setText(getString(R.string.save));

bindingView.viewHeader.ivHelp.setVisibility(View.VISIBLE);

initPickerView();

basePresenter.getWanInfo();

initListener();

}

public void initListener() {

bindingView.viewHeader.ivBack.setOnClickListener(view -> {

this.finish();

});

bindingView.viewHeader.ivHelp.setOnClickListener(view -> {

HelpContentActivity.start(this, getString(R.string.answer\_function\_network));

});

bindingView.tvInternet.setOnClickListener(view -> {

KeyBoardUtils.hideInputForce(this);

if (!internetPickerView.isShowing()) internetPickerView.show();

});

bindingView.tvNext.setOnClickListener(view -> {

wanInfoParam = new WanConfigBean();

if (selectIndex == 1) {

wanInfoParam.setIpaddr(bindingView.edtIpAddress.getText().toString().trim());

wanInfoParam.setNetmask(bindingView.edtSubnetMask.getText().toString().trim());

wanInfoParam.setGateway(bindingView.edtGetWay.getText().toString().trim());

wanInfoParam.setDns1(bindingView.edtDns1.getText().toString().trim());

wanInfoParam.setDns2(bindingView.edtDns2.getText().toString().trim());

wanInfoParam.setPeerdns("0");

wanInfoParam.setProto("static");

} else if (selectIndex == 2) {

wanInfoParam.setProto("pppoe");

wanInfoParam.setUsername(bindingView.edtBroadbandNo.getText().toString().trim());

wanInfoParam.setPassword(bindingView.edtBroadbandPwd.getText().toString().trim());

wanInfoParam.setDns1(bindingView.edtDns1.getText().toString().trim());

wanInfoParam.setDns2(bindingView.edtDns2.getText().toString().trim());

} else {

wanInfoParam.setProto("dhcp");

wanInfoParam.setDns1(bindingView.edtDns1.getText().toString().trim());

wanInfoParam.setDns2(bindingView.edtDns2.getText().toString().trim());

}

wanInfoParam.setPeerdns(wanInfoBean.getPeerdns());

if (isModify()) {

showToast(R.string.not\_changes);

return;

}

basePresenter.setWanInfo(wanInfoParam);

});

}

public boolean isModify() {

boolean flag = false;

if (selectIndex == 1) {

if (wanInfoParam.getIpaddr().equals(wanInfoBean.getIpaddr())

&& wanInfoParam.getNetmask().equals(wanInfoBean.getNetmask())

&& wanInfoParam.getGateway().equals(wanInfoBean.getGateway())

&& wanInfoParam.getDns1().equals(wanInfoBean.getDns1())

&& wanInfoParam.getDns2().equals(wanInfoBean.getDns2())) {

flag = true;

}

} else if (selectIndex == 2) {

if (wanInfoParam.getUsername().equals(wanInfoBean.getUsername())

&& wanInfoParam.getPassword().equals(wanInfoBean.getPassword())) {

flag = true;

}

} else {

if ((wanInfoParam.getProto().equals(wanInfoBean.getProto()))) {

flag = true;

}

}

return flag;

}

private void initPickerView() {

internetList.add(getString(R.string.dynamic\_ip));

internetList.add(getString(R.string.static\_ip));

internetList.add(getString(R.string.pppoe\_account));

internetPickerView = genPickerBuilder(this, (options1, options2, options3, v) -> {

bindingView.tvInternet.setText(internetList.get(options1));

selectIndex = options1;

if (0 == options1) {

bindingView.llStaticIp.setVisibility(View.GONE);

bindingView.llPoe.setVisibility(View.GONE);

} else if (1 == options1) {

bindingView.llStaticIp.setVisibility(View.VISIBLE);

bindingView.llPoe.setVisibility(View.GONE);

} else if (2 == options1) {

bindingView.llPoe.setVisibility(View.VISIBLE);

bindingView.llStaticIp.setVisibility(View.GONE);

}

}).build();

internetPickerView.setPicker(internetList);

}

public void onDataSuccess(WanInfoBean bean) {

wanInfoBean = bean.getWan\_config();

if ("dhcp".equals(bean.getWan\_config().getProto())) {//动态Ip

bindingView.llStaticIp.setVisibility(View.GONE);

bindingView.llPoe.setVisibility(View.GONE);

bindingView.tvInternet.setText(internetList.get(0));

bindingView.edtDns1.setText(bean.getWan\_config().getDns1());

bindingView.edtDns2.setText(bean.getWan\_config().getDns2());

} else if ("static".equals(bean.getWan\_config().getProto())) {//静态Ip

bindingView.llPoe.setVisibility(View.GONE);

bindingView.llStaticIp.setVisibility(View.VISIBLE);

bindingView.tvInternet.setText(internetList.get(1));

bindingView.edtIpAddress.setText(bean.getWan\_config().getIpaddr());

bindingView.edtSubnetMask.setText(bean.getWan\_config().getNetmask());

bindingView.edtGetWay.setText(bean.getWan\_config().getGateway());

bindingView.edtDns1.setText(bean.getWan\_config().getDns1());

bindingView.edtDns2.setText(bean.getWan\_config().getDns2());

} else if ("pppoe".equals(bean.getWan\_config().getProto())) {//pppoe

bindingView.llStaticIp.setVisibility(View.GONE);

bindingView.llPoe.setVisibility(View.VISIBLE);

bindingView.edtBroadbandNo.setText(bean.getWan\_config().getUsername());

bindingView.edtBroadbandPwd.setText(bean.getWan\_config().getPassword());

bindingView.edtDns1.setText(bean.getWan\_config().getDns1());

bindingView.edtDns2.setText(bean.getWan\_config().getDns2());

// bindingView.edtMtu.setText(bean.getWan\_mtu());

bindingView.tvInternet.setText(internetList.get(2));

} else {

bindingView.llStaticIp.setVisibility(View.GONE);

bindingView.llPoe.setVisibility(View.GONE);

bindingView.edtIpAddress.setText(bean.getWan\_config().getIpaddr());

bindingView.edtSubnetMask.setText(bean.getWan\_config().getNetmask());

bindingView.edtGetWay.setText(bean.getWan\_config().getGateway());

bindingView.edtDns1.setText(bean.getWan\_config().getDns1());

bindingView.edtDns2.setText(bean.getWan\_config().getDns2());

}

}

@Override

protected void onPause() {

super.onPause();

KeyBoardUtils.hideInputForce(this);

if (basePresenter.disposable != null && !basePresenter.disposable.isDisposed()) {

basePresenter.disposable.dispose();

}

}

public void onWanFail() {

}

public void onSetWanSuccess() {

if (selectIndex == 0) {

wanInfoBean.setProto("dhcp");

} else if (selectIndex == 1) {

wanInfoBean.setProto("static");

wanInfoBean.setIpaddr(bindingView.edtIpAddress.getText().toString());

wanInfoBean.setNetmask(bindingView.edtSubnetMask.getText().toString());

wanInfoBean.setGateway(bindingView.edtGetWay.getText().toString());

wanInfoBean.setDns1(bindingView.edtDns1.getText().toString());

wanInfoBean.setDns2(bindingView.edtDns2.getText().toString());

} else if (selectIndex == 2) {

wanInfoBean.setProto("pppoe");

wanInfoBean.setUsername(bindingView.edtBroadbandNo.getText().toString().trim());

wanInfoBean.setPassword(bindingView.edtBroadbandPwd.getText().toString().trim());

}

showToast(R.string.save\_success);

}

}

package com.kingsignal.elf1.ui.setting.system;

import android.app.Activity;

import android.app.Dialog;

import android.content.Context;

import android.content.Intent;

import android.os.Handler;

import android.os.Message;

import android.view.View;

import androidx.recyclerview.widget.LinearLayoutManager;

import com.kingsignal.common.base.PresenterActivity;

import com.kingsignal.common.http.response.VersionBean;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.adapter.SoftwareUpgradeAdapter;

import com.kingsignal.elf1.databinding.ActivitySoftwareUpgradeBinding;

import com.kingsignal.elf1.dialog.MessageDialog;

import com.kingsignal.elf1.dialog.ProgressDialog;

import com.kingsignal.elf1.dialog.VersionUpdateDialog;

import com.kingsignal.elf1.entity.CheckFirmWareBean;

import com.kingsignal.elf1.entity.DownloadProgressBean;

import com.kingsignal.elf1.entity.UpdateDeviceBean;

import com.kingsignal.elf1.entity.VerInfoBean;

import com.kingsignal.elf1.presenter.settings.system.SoftwareUpgradePresenter;

import com.kingsignal.elf1.ui.guide.NetworkDetectionActivity;

import com.kingsignal.elf1.ui.setting.help.HelpContentActivity;

import java.lang.ref.WeakReference;

import java.util.ArrayList;

import java.util.List;

import java.util.Timer;

import java.util.TimerTask;

/\*\*

\* 软件升级

\* @author zwy

\* create at 2020-12-18

\*/

public class SoftwareUpgradeActivity extends PresenterActivity<SoftwareUpgradePresenter, ActivitySoftwareUpgradeBinding> {

SoftwareUpgradeAdapter mAdapter;

private List<UpdateDeviceBean.DevListBean> mList = new ArrayList<>();

VerInfoBean verInfoBean;

private TimerTask mTimerTask = null;

private Timer mTimer = null;

MyHandler handler;

private String updateDev;

private String deviceVer;

private ProgressDialog mProgressDialog;

public static void start(Context context, VerInfoBean infoBean) {

Intent intent = new Intent(context, SoftwareUpgradeActivity.class);

intent.putExtra("infoBean", infoBean);

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_software\_upgrade;

}

@Override

public void initData() {

if (getIntent() != null) {

verInfoBean = getIntent().getParcelableExtra("infoBean");

}

}

@Override

public void initView() {

bindingView.viewHeader.tvTitle.setText(getString(R.string.setting\_software\_upgrade));

bindingView.viewHeader.ivHelp.setVisibility(View.VISIBLE);

basePresenter.getDeviceList();

initAdapter();

initListener();

}

private void initAdapter() {

mAdapter = new SoftwareUpgradeAdapter(mList);

mAdapter.setVerInfoBean(verInfoBean);

//布局管理器所需参数，上下文

LinearLayoutManager linearLayoutManager = new LinearLayoutManager(this);

bindingView.recyclerView.setLayoutManager(linearLayoutManager);

bindingView.recyclerView.setAdapter(mAdapter);

}

public void initListener() {

bindingView.viewHeader.ivBack.setOnClickListener(view -> {

SoftwareUpgradeActivity.this.finish();

});

bindingView.viewHeader.ivHelp.setOnClickListener(view -> {

HelpContentActivity.start(this, getString(R.string.answer\_function\_software));

});

mAdapter.setListener(position -> {

if ("1".equals(mList.get(position).getIs\_master())) {

updateDev = "";

} else {

updateDev = mList.get(position).getMac();

}

deviceVer = mList.get(position).getVer();

basePresenter.downloadFirmWare();

showProgressLoading();

});

}

public void showProgressLoading() {

mProgressDialog = ProgressDialog.getInstance().builder(this);

mProgressDialog.setListener(() -> {

if (mProgressDialog != null) {

if (basePresenter.disposable != null) {

basePresenter.disposable.dispose();

basePresenter.disposable = null;

stopTimer();

}

mProgressDialog.dismiss();

}

});

mProgressDialog.show();

}

@Override

protected void onPause() {

super.onPause();

stopTimer();

if (basePresenter.disposable != null && !basePresenter.disposable.isDisposed()) {

basePresenter.disposable.dispose();

}

}

public void hideProgressLoading() {

if (mProgressDialog != null) {

mProgressDialog.dismiss();

}

}

public void onDataSuccess(List<UpdateDeviceBean.DevListBean> list) {

if (verInfoBean == null) {

verInfoBean = new VerInfoBean();

}

this.mList = list;

mAdapter.setVerInfoBean(verInfoBean);

mAdapter.setNewData(list);

mAdapter.notifyDataSetChanged();

}

public void onProgressSuccess(DownloadProgressBean response) {

if (mProgressDialog != null) {

mProgressDialog.setDownloadData(response.getProgress().getPercent());

}

if (response.getProgress().isComplete() && response.getProgress().getPercent() == 100) {

stopTimer();

hideProgressLoading();

basePresenter.checkFirmware(deviceVer);

}

}

public void onProgressFailure() {

stopTimer();

hideProgressLoading();

}

//检验版本通过

public void onCheckFirmwareSuccess(CheckFirmWareBean bean) {

if (0 == bean.getAuth\_info().getCodeX()) {

checkSuccessDialog();

} else {

checkFailureDialog();

}

}

public void onCheckFirmwareFailure() {

}

/\*\*

\* 升级成功

\* @author zwy

\* create at 2020-12-19

\*/

public void onUpdateFirmwareSuccess() {

showToast(getString(R.string.not\_get\_new\_version));

handler = new MyHandler(this);

handler.sendEmptyMessageDelayed(1, 2000);

}

private void startTimer() {

if (mTimer == null) {

mTimer = new Timer();

}

if (mTimerTask == null) {

mTimerTask = new TimerTask() {

@Override

public void run() {

basePresenter.downloadProgress();

}

};

}

if (mTimer != null && mTimerTask != null)

mTimer.schedule(mTimerTask, 0, 1000);

}

private void stopTimer() {

if (mTimerTask != null) {

mTimerTask.cancel();

mTimerTask = null;

}

if (mTimer != null) {

mTimer.cancel();

mTimer = null;

}

}

/\*\*

\* 版本是否可以用

\*/

public void checkSuccessDialog() {

new MessageDialog.Builder(this)

.setTitle("") // 标题可以不用填写

.setMessage(getResources().getString(R.string.update\_firmware))

//.setConfirm("确认")

//.setCancel("取消") // 设置 null 表示不显示取消按钮

//.setAutoDismiss(false) // 设置点击按钮后不关闭对话框

.setListener(new MessageDialog.OnListener() {

@Override

public void onConfirm(Dialog dialog) {

basePresenter.updateFirmware(updateDev);

}

@Override

public void onCancel(Dialog dialog) {

}

})

.show();

}

/\*\*

\* 版本是否不可以用

\*/

public void checkFailureDialog() {

new MessageDialog.Builder(this)

.setTitle("") // 标题可以不用填写

.setMessage(getResources().getString(R.string.not\_support))

//.setConfirm("确认")

.setCancel(null) // 设置 null 表示不显示取消按钮

//.setAutoDismiss(false) // 设置点击按钮后不关闭对话框

.setListener(new MessageDialog.OnListener() {

@Override

public void onConfirm(Dialog dialog) {

}

@Override

public void onCancel(Dialog dialog) {

}

})

.show();

}

public void downloadFirmWareSuccess() {

startTimer();

}

public void run() {

NetworkDetectionActivity.start(this, "");

this.finish();

}

static class MyHandler extends Handler {

// SoftReference<Activity> 也可以使用软应用 只有在内存不足的时候才会被回收

private final WeakReference<SoftwareUpgradeActivity> mActivity;

private MyHandler(SoftwareUpgradeActivity activity) {

mActivity = new WeakReference<>(activity);

}

@Override

public void handleMessage(Message msg) {

Activity activity = mActivity.get();

if (activity != null) {

mActivity.get().run();

}

super.handleMessage(msg);

}

}

}

package com.kingsignal.elf1.ui.setting.system;

import android.annotation.SuppressLint;

import android.app.Activity;

import android.app.Dialog;

import android.content.Context;

import android.content.Intent;

import android.graphics.Color;

import android.os.Handler;

import android.os.Message;

import android.view.View;

import com.bigkoo.pickerview.builder.TimePickerBuilder;

import com.bigkoo.pickerview.view.TimePickerView;

import com.kingsignal.common.base.PresenterActivity;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.databinding.ActivitySystemResetBinding;

import com.kingsignal.elf1.dialog.MessageDialog;

import com.kingsignal.elf1.entity.RebootInfoBean;

import com.kingsignal.elf1.presenter.settings.system.SystemResetPresenter;

import com.kingsignal.elf1.ui.guide.NetworkDetectionActivity;

import com.kingsignal.elf1.ui.setting.help.HelpContentActivity;

import com.kingsignal.elf1.utils.UIUtil;

import java.lang.ref.WeakReference;

/\*\*

\* 系统重启

\*

\*/

public class SystemResetActivity extends PresenterActivity<SystemResetPresenter, ActivitySystemResetBinding> {

boolean isOpen = false;

TimePickerView tpvTime;

MyHandler handler;

String type = "";

public static void start(Context context) {

Intent intent = new Intent(context, SystemResetActivity.class);

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_system\_reset;

}

@Override

public void initData() {

}

@Override

public void initView() {

bindingView.viewHeader.tvTitle.setText(getString(R.string.restart\_router));

bindingView.viewHeader.ivHelp.setVisibility(View.VISIBLE);

initPop();

basePresenter.getTimeRebootInfo();

initListener();

}

public void initListener() {

bindingView.viewHeader.ivBack.setOnClickListener(view -> SystemResetActivity.this.finish());

bindingView.viewHeader.ivHelp.setOnClickListener(view -> {

HelpContentActivity.start(this, getString(R.string.answer\_reset));

});

bindingView.tvResetImmediately.setOnClickListener(view -> {

restartWifi();

});

bindingView.ivOpen.setOnClickListener(view -> {

if (isOpen) {

isOpen = false;

} else {

isOpen = true;

}

switchStatus();

});

bindingView.clTime.setOnClickListener(view -> {

tpvTime.show();

});

bindingView.tvNext.setOnClickListener(view -> {

if (isOpen) {

type = "1";

basePresenter.setTimeRebootInfo("1", bindingView.tvTime.getText().toString().trim());

} else {

tipDialog();

}

});

}

private void switchStatus() {

if (isOpen) {

bindingView.ll.setVisibility(View.VISIBLE);

bindingView.ivOpen.setBackgroundResource(R.mipmap.swithch\_on);

} else {

bindingView.ll.setVisibility(View.GONE);

bindingView.ivOpen.setBackgroundResource(R.mipmap.switch\_off);

}

}

/\*\*

\* 立即重启

\*/

public void tipDialog() {

new MessageDialog.Builder(this)

.setTitle("") // 标题可以不用填写

.setMessage(getResources().getString(R.string.close\_router\_time\_tip))

//.setConfirm("确认")

//.setCancel("取消") // 设置 null 表示不显示取消按钮

//.setAutoDismiss(false) // 设置点击按钮后不关闭对话框

.setListener(new MessageDialog.OnListener() {

@Override

public void onConfirm(Dialog dialog) {

type = "0";

basePresenter.setTimeRebootInfo("0", "");

}

@Override

public void onCancel(Dialog dialog) {

}

})

.show();

}

/\*\*

\* 立即重启

\*/

public void restartWifi() {

new MessageDialog.Builder(this)

.setTitle("") // 标题可以不用填写

.setMessage(getResources().getString(R.string.restart\_wifi))

//.setConfirm("确认")

//.setCancel("取消") // 设置 null 表示不显示取消按钮

//.setAutoDismiss(false) // 设置点击按钮后不关闭对话框

.setListener(new MessageDialog.OnListener() {

@Override

public void onConfirm(Dialog dialog) {

basePresenter.setRebootInfo("1");

}

@Override

public void onCancel(Dialog dialog) {

}

})

.show();

}

/\*\*

\* 初始化弹框

\*/

private void initPop() {

tpvTime = new TimePickerBuilder(this, (date, v) -> {//选中事件回调

bindingView.tvTime.setText(UIUtil.getTime(date));

})

.setType(new boolean[]{false, false, false, true, true, false})// 默认全部显示

.setCancelText(getResources().getString(R.string.cancel))//取消按钮文字

.setSubmitText(getResources().getString(R.string.ok))//确认按钮文字

.setSubCalSize(16)//滚轮文字大小

.setTitleSize(16)//标题文字大小

.setContentTextSize(20)

.setTitleText(getResources().getString(R.string.select\_time))//标题文字

.setOutSideCancelable(false)//点击屏幕，点在控件外部范围时，是否取消显示

.isCyclic(true)//是否循环滚动

.setTitleColor(Color.BLACK)//标题文字颜色

.setSubmitColor(getResources().getColor(com.kingsignal.common.R.color.black))//确定按钮文字颜色

.setCancelColor(getResources().getColor(com.kingsignal.common.R.color.black))//取消按钮文字颜色

.setLabel("", "", "", "", "", "")//默认设置为年月日时分秒

.isCenterLabel(false) //是否只显示中间选中项的label文字，false则每项item全部都带有label。

.build();

}

public void onDataSuccess(RebootInfoBean bean) {

if ("1".equals(bean.getTimed\_reboot\_info().getTimed\_reboot\_switch())) {

isOpen = true;

} else {

isOpen = false;

}

bindingView.tvTime.setText(bean.getTimed\_reboot\_info().getReboot\_time());

switchStatus();

}

public void onRebootInfoSuccess() {

showToast(getResources().getString(R.string.system\_restart\_success));

handler = new MyHandler(this);

handler.sendEmptyMessageDelayed(1, 2000);

}

public void onTimeInfoSuccess() {

String time = bindingView.tvTime.getText().toString().trim();

if (isOpen) {

showToast(getResources().getString(R.string.router\_configure\_tip, time));

} else {

showToast(getResources().getString(R.string.setting\_success));

}

}

public void run() {

NetworkDetectionActivity.start(this, "");

}

static class MyHandler extends Handler {

// SoftReference<Activity> 也可以使用软应用 只有在内存不足的时候才会被回收

private final WeakReference<SystemResetActivity> mActivity;

private MyHandler(SystemResetActivity activity) {

mActivity = new WeakReference<>(activity);

}

@Override

public void handleMessage(Message msg) {

Activity activity = mActivity.get();

if (activity != null) {

mActivity.get().run();

}

super.handleMessage(msg);

}

}

}

package com.kingsignal.elf1.ui.setting.wifi;

import android.content.Context;

import android.content.Intent;

import android.view.View;

import androidx.recyclerview.widget.LinearLayoutManager;

import com.kingsignal.common.base.PresenterActivity;

import com.kingsignal.common.utils.toast.ToastUtils;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.adapter.TimeZoneAdapter;

import com.kingsignal.elf1.adapter.TransmitRegionAdapter;

import com.kingsignal.elf1.bean.LoginSetting;

import com.kingsignal.elf1.databinding.ActivityTransmitRegionBinding;

import com.kingsignal.elf1.entity.CheckBean;

import com.kingsignal.elf1.entity.TransmitRegionBean;

import com.kingsignal.elf1.presenter.wifi.TransmitRegionPresenter;

import com.kingsignal.elf1.ui.guide.NetworkDetectionActivity;

import com.kingsignal.elf1.ui.setting.help.HelpContentActivity;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* 传输地区

\*/

public class TransmitRegionActivity extends PresenterActivity<TransmitRegionPresenter, ActivityTransmitRegionBinding> {

private TransmitRegionAdapter mAdapter;

List<CheckBean> mSearchList;

int selectPosition = 0;

String[] regionCode;

public static void start(Context context) {

Intent intent = new Intent(context, TransmitRegionActivity.class);

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_transmit\_region;

}

@Override

public void initData() {

}

@Override

public void initView() {

bindingView.viewHeader.tvTitle.setText(getString(R.string.setting\_trans));

bindingView.viewHeader.ivHelp.setVisibility(View.VISIBLE);

initAdapter();

initListener();

getData();

}

private void initAdapter() {

String[] regionList = getResources().getStringArray(R.array.transmit\_region);

regionCode = getResources().getStringArray(R.array.transmit\_region\_code);

mSearchList = new ArrayList<>();

for (int i = 0; i < regionList.length; i++) {

mSearchList.add(new CheckBean(regionList[i], false));

}

mAdapter = new TransmitRegionAdapter(mSearchList);

bindingView.recyclerView.setLayoutManager(new LinearLayoutManager(this));

bindingView.recyclerView.setAdapter(mAdapter);

}

public void initListener() {

bindingView.viewHeader.ivBack.setOnClickListener(view -> {

this.finish();

});

bindingView.viewHeader.ivHelp.setOnClickListener(view -> {

HelpContentActivity.start(this, getString(R.string.answer\_function\_transmit));

});

mAdapter.setOnItemClickListener((adapter, view, position) -> {

if (mSearchList.get(position).isCheck()) {

return;

}

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

mSearchList.get(i).setCheck(false);

}

selectPosition = position;

mSearchList.get(position).setCheck(true);

mAdapter.notifyDataSetChanged();

});

bindingView.tvNext.setOnClickListener(view -> {

String regionStr = regionCode[selectPosition];

basePresenter.setCountryCode(regionStr);

});

}

public void getData() {

basePresenter.getCountryCode();

}

public void onDataSuccess(TransmitRegionBean response) {

int position = -1;

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

if (regionCode[i].equals(response.getWifi\_countrycode\_info().getCountrycode())) {

mSearchList.get(i).setCheck(true);

position = i;

}

}

if (position != -1) {

// mAdapter.setNewData(mSearchList);

mAdapter.notifyItemChanged(position);

}

}

public void onSetDataSuccess() {

ToastUtils.show(TransmitRegionActivity.this, getResources().getString(R.string.modify\_transmit));

NetworkDetectionActivity.start(this, "");

this.finish();

}

@Override

protected void onPause() {

super.onPause();

if (basePresenter.disposable != null && !basePresenter.disposable.isDisposed()) {

basePresenter.disposable.dispose();

}

}

}

public void loginOutDialog() {  
 new MessageDialog.Builder(this)  
 .setTitle("") // 标题可以不用填写  
 .setMessage(getResources().getString(R.string.how\_log\_out))  
 //.setConfirm("确认")  
 //.setCancel("取消") // 设置 null 表示不显示取消按钮  
 //.setAutoDismiss(false) // 设置点击按钮后不关闭对话框  
 .setListener(new MessageDialog.OnListener() {  
 @Override  
 public void onConfirm(Dialog dialog) {  
 basePresenter.setLoginOut();  
 }  
  
 @Override  
 public void onCancel(Dialog dialog) {  
  
 }  
 })  
 .show();  
}

package com.kingsignal.elf1.adapter;

import android.widget.ImageView;

import android.widget.TextView;

import androidx.annotation.Nullable;

import com.chad.library.adapter.base.BaseQuickAdapter;

import com.chad.library.adapter.base.BaseViewHolder;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.entity.CheckBean;

import java.util.List;

public class TransmitRegionAdapter extends BaseQuickAdapter<CheckBean, BaseViewHolder> {

public TransmitRegionAdapter(@Nullable List<CheckBean> data) {

super(R.layout.item\_choose\_zone, data);

}

@Override

protected void convert(BaseViewHolder helper, CheckBean item) {

ImageView ivImg = helper.itemView.findViewById(R.id.imageView);

TextView tvTitle = helper.itemView.findViewById(R.id.tv\_title);

if (item.isCheck()) {

ivImg.setBackgroundResource(R.mipmap.timezone\_s);

tvTitle.setTextColor(mContext.getResources().getColor(R.color.blue));

} else {

ivImg.setBackgroundResource(R.mipmap.timezone\_n);

tvTitle.setTextColor(mContext.getResources().getColor(R.color.black));

}

tvTitle.setText(item.getName());

}

}

package com.kingsignal.elf1.ui.guide;

import android.animation.Animator;

import android.animation.AnimatorSet;

import android.animation.ObjectAnimator;

import android.app.Activity;

import android.app.Dialog;

import android.content.Context;

import android.content.Intent;

import android.os.CountDownTimer;

import android.os.Handler;

import android.os.Message;

import android.view.View;

import android.view.animation.AccelerateDecelerateInterpolator;

import android.widget.ImageView;

import com.kingsignal.common.base.MyApplication;

import com.kingsignal.common.base.PresenterActivity;

import com.kingsignal.common.http.NetworkConstant;

import com.kingsignal.common.utils.WifiUtils;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.base.dialog.BaseDialog;

import com.kingsignal.elf1.databinding.ActivityNetworkDetectionBinding;

import com.kingsignal.elf1.dialog.MessageDialog;

import com.kingsignal.elf1.entity.GuideBean;

import com.kingsignal.elf1.presenter.guide.GuidePresenter;

import com.kingsignal.elf1.presenter.guide.GuideView;

import com.kingsignal.elf1.ui.SplashActivity;

import com.kingsignal.elf1.ui.login.LoginActivity;

import java.lang.ref.WeakReference;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Timer;

import java.util.TimerTask;

/\*\*  
 \* @author zwy  
 \* create at 2020-11-22  
 \*/

public class NetworkDetectionActivity extends PresenterActivity<GuidePresenter, ActivityNetworkDetectionBinding> implements GuideView {

private TimerTask mTimerTask = null;

private Timer mTimer = null;

MyHandler handler;

private AnimatorSet animatorSet;

private boolean animationRunning = false;

private int rippleDuration = 2500;

private int rippleAmount = 3;

private float rippleScale = 2;

private String type;

private ArrayList<ImageView> rippleViewList = new ArrayList<>();

boolean isShowPop = true;

boolean isShowTipFailure = true;

int duration = 120;

public static void start(Context context, String type) {

Intent intent = new Intent(context, NetworkDetectionActivity.class);

intent.putExtra("type", type);

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_network\_detection;

}

@Override

public void initData() {

if (getIntent() != null) {

type = getIntent().getStringExtra("type");

}

isShowPop = true;

}

@Override

public void initView() {

initAnimation();

}

@Override

protected void onResume() {

super.onResume();

duration = 120;

MyApplication.getInstance().failureJump = false;

NetworkConstant.REQUEST\_URL = WifiUtils.getWifiIpAddress();

handler = new MyHandler(this);

handler.sendEmptyMessageDelayed(1, 5000);

startRippleAnimation();

timer.start();

// startTimer();

}

@Override

protected void onPause() {

super.onPause();

timer.cancel();

stopTimer();

stopRippleAnimation();

if (handler != null) {

handler.removeCallbacksAndMessages(0);

}

}

@Override

protected void onDestroy() {

super.onDestroy();

MyApplication.getInstance().failureJump = true;

}

@Override

public void onGuideSuccess(GuideBean bean) {

isShowTipFailure = true;

if (NetworkConstant.DETECTION\_GUIDE.equals(type)) {

SubRouteOneActivity.start(this, "");

} else {

if (bean.getFirst\_login().equals("1")) {

//WiFi已连接进入引导页

NewGuideStartActivity.start(NetworkDetectionActivity.this, bean);

} else {

LoginActivity.start(this);

}

}

NetworkDetectionActivity.this.finish();

}

private void initAnimation() {

int rippleDelay = rippleDuration / rippleAmount;

rippleViewList.add(bindingView.ivRect1);

rippleViewList.add(bindingView.ivRect2);

rippleViewList.add(bindingView.ivRect3);

//分析该动画后将其拆分为缩放、渐变

ArrayList<Animator> animatorList = new ArrayList<>();

for (int i = 0; i < 3; i++) {

//ScaleX缩放

final ObjectAnimator scaleXAnimator = ObjectAnimator.ofFloat(rippleViewList.get(i), "ScaleX", 1.0f, 1.5F);

scaleXAnimator.setRepeatCount(ObjectAnimator.INFINITE);//无限重复

scaleXAnimator.setRepeatMode(ObjectAnimator.RESTART);

scaleXAnimator.setStartDelay(i \* rippleDelay);

scaleXAnimator.setDuration(rippleDuration);

animatorList.add(scaleXAnimator);

//ScaleY缩放

final ObjectAnimator scaleYAnimator = ObjectAnimator.ofFloat(rippleViewList.get(i), "ScaleY", 1.0f, 1.5f);

scaleYAnimator.setRepeatCount(ObjectAnimator.INFINITE);//无限重复

scaleYAnimator.setRepeatMode(ObjectAnimator.RESTART);

scaleYAnimator.setStartDelay(i \* rippleDelay);

scaleYAnimator.setDuration(rippleDuration);

animatorList.add(scaleYAnimator);

//Alpha渐变

final ObjectAnimator alphaAnimator = ObjectAnimator.ofFloat(rippleViewList.get(i), "Alpha", 1.0f, 0f);

alphaAnimator.setRepeatCount(ObjectAnimator.INFINITE);//无限重复

alphaAnimator.setRepeatMode(ObjectAnimator.RESTART);

alphaAnimator.setStartDelay(i \* rippleDelay);

alphaAnimator.setDuration(rippleDuration);

animatorList.add(alphaAnimator);

}

animatorSet = new AnimatorSet();

animatorSet.setInterpolator(new AccelerateDecelerateInterpolator());

animatorSet.playTogether(animatorList);

}

/\*\*

\* 是否正在执行

\*

\* @return boolean isRippleRunning

\*/

public boolean isRippleRunning() {

return animationRunning;

}

/\*\*

\* 开始动画

\*/

public void startRippleAnimation() {

if (!isRippleRunning()) {

for (ImageView rippleView : rippleViewList) {

rippleView.setVisibility(View.VISIBLE);

}

animatorSet.start();

animationRunning = true;

}

}

/\*\*

\* 停止动画

\*/

public void stopRippleAnimation() {

if (isRippleRunning()) {

Collections.reverse(rippleViewList);

for (ImageView rippleView : rippleViewList) {

rippleView.setVisibility(View.INVISIBLE);

}

animatorSet.end();

animationRunning = false;

}

}

@Override

public void onGuideFail() {

if (NetworkConstant.DETECTION\_FIRST.equals(type)) {

if (isShowPop) {

detectionDialog();

}

} else {

isShowTipFailure = false;

}

}

CountDownTimer timer = new CountDownTimer(120 \* 1000, 1000) {

@Override

public void onTick(long millisUntilFinished) {

}

@Override

public void onFinish() {

if (isShowTipFailure) {

detectionFailureDialog();

}

}

};

/\*\*

\* 提示用户需要连上产品WiFi

\*/

public void detectionFailureDialog() {

isShowPop = false;

new MessageDialog.Builder(this)

.setTitle("") // 标题可以不用填写

.setMessage(getResources().getString(R.string.detection\_product\_wifi))

.setConfirm("确认")

.setCancel(null) // 设置 null 表示不显示取消按钮

//.setAutoDismiss(false) // 设置点击按钮后不关闭对话框

.setListener(new MessageDialog.OnListener() {

@Override

public void onConfirm(Dialog dialog) {

}

@Override

public void onCancel(Dialog dialog) {

}

})

.show();

}

/\*\*

\* 检测是否第一次连接

\*/

public void detectionDialog() {

isShowPop = false;

new MessageDialog.Builder(this)

.setTitle("") // 标题可以不用填写

.setMessage(getResources().getString(R.string.detection\_not\_connected))

//.setConfirm("确认")

//.setCancel("取消") // 设置 null 表示不显示取消按钮

//.setAutoDismiss(false) // 设置点击按钮后不关闭对话框

.setListener(new MessageDialog.OnListener() {

@Override

public void onConfirm(Dialog dialog) {

//WiFi已连接进入引导页

NewGuideStartActivity.start(NetworkDetectionActivity.this, new GuideBean());

}

@Override

public void onCancel(Dialog dialog) {

showToast(getString(R.string.detection\_product\_wifi));

}

})

.show();

}

@Override

public void onSetSuccess() {

}

private void startTimer() {

stopTimer();

if (mTimer == null) {

mTimer = new Timer();

}

if (mTimerTask == null) {

mTimerTask = new TimerTask() {

@Override

public void run() {

basePresenter.getGuide();

}

};

}

if (mTimer != null && mTimerTask != null) {

mTimer.schedule(mTimerTask, 0, 5000);

}

}

private void stopTimer() {

if (mTimerTask != null) {

mTimerTask.cancel();

mTimerTask = null;

}

if (mTimer != null) {

mTimer.cancel();

mTimer = null;

}

}

public void run() {

startTimer();

}

static class MyHandler extends Handler {

// SoftReference<Activity> 也可以使用软应用 只有在内存不足的时候才会被回收

private final WeakReference<NetworkDetectionActivity> mActivity;

private MyHandler(NetworkDetectionActivity activity) {

mActivity = new WeakReference<>(activity);

}

@Override

public void handleMessage(Message msg) {

Activity activity = mActivity.get();

if (activity != null) {

mActivity.get().run();

}

super.handleMessage(msg);

}

}

}

package com.kingsignal.elf1.ui.login;

import android.content.Context;

import android.content.Intent;

import android.text.method.HideReturnsTransformationMethod;

import android.text.method.PasswordTransformationMethod;

import android.view.View;

import androidx.databinding.ObservableField;

import androidx.recyclerview.widget.LinearLayoutManager;

import com.kingsignal.common.base.PresenterActivity;

import com.kingsignal.common.utils.SP;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.adapter.LanguageAdapter;

import com.kingsignal.elf1.databinding.ActivityLoginBinding;

import com.kingsignal.elf1.entity.CheckBean;

import com.kingsignal.elf1.presenter.login.LoginPresenter;

import com.kingsignal.elf1.ui.webView.WebViewActivity;

import java.util.ArrayList;

import java.util.List;

/\*\*  
 \* @author zwy  
 \* create at 2020-11-20  
 \*/

public class LoginActivity extends PresenterActivity<LoginPresenter, ActivityLoginBinding> {

List<CheckBean> mList;

LanguageAdapter adapter;

String language = "cn";

boolean isHide;

boolean isOpen = true;

public static final ObservableField<String> password = new ObservableField<>();

/\*\*

\* @param

\*/

public static void start(Context context) {

Intent intent = new Intent(context, LoginActivity.class);

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

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_login;

}

@Override

public void initData() {

}

@Override

public void initView() {

dealData();

initAdapter();

initListener();

}

private void dealData() {

if (SP.getRemember()) {

password.set(SP.getPskValue());

bindingView.ivOpen.setBackgroundResource(R.mipmap.swithch\_on);

} else {

password.set("");

bindingView.ivOpen.setBackgroundResource(R.mipmap.switch\_off);

}

}

public void initAdapter() {

mList = new ArrayList<>();

mList.add(new CheckBean(getResources().getString(R.string.chinese), false));

mList.add(new CheckBean(getResources().getString(R.string.english), false));

if (SP.getLanguage() == 0) {

mList.get(0).setCheck(true);

bindingView.tvLanguage.setText(getResources().getString(R.string.chinese));

bindingView.jumpContext.setVisibility(View.VISIBLE);

bindingView.jumpLogin.setVisibility(View.VISIBLE);

bindingView.serviceLogin.setVisibility(View.VISIBLE);

} else {

mList.get(1).setCheck(true);

bindingView.tvLanguage.setText(getResources().getString(R.string.english));

bindingView.jumpContext.setVisibility(View.GONE);

bindingView.jumpLogin.setVisibility(View.GONE);

bindingView.serviceLogin.setVisibility(View.GONE);

}

adapter = new LanguageAdapter(mList);

bindingView.rvLanguage.setLayoutManager(new LinearLayoutManager(this));

bindingView.rvLanguage.setAdapter(adapter);

}

public void initListener() {

bindingView.jumpLogin.setOnClickListener(v -> {

// toast("隐私协议");//这是点击事件生效了,你可以去掉这一行代码

WebViewActivity.start(LoginActivity.this,"隐私政策","file:///android\_asset/privacy\_policy.html");

});

bindingView.serviceLogin.setOnClickListener(v -> {

// toast("隐私协议");//这是点击事件生效了,你可以去掉这一行代码

WebViewActivity.start(LoginActivity.this,"用户协议","file:///android\_asset/uesr\_agreement.html");

});

bindingView.tvLogin.setOnClickListener(view -> {

saveRememberPwd();

String sid = bindingView.etUserName.getText().toString().trim();

String pwd = bindingView.etPassword.getText().toString().trim();

basePresenter.login(sid, pwd);

});

bindingView.ivOpen.setOnClickListener(view -> {

if (isOpen) {

bindingView.ivOpen.setBackgroundResource(R.mipmap.switch\_off);

isOpen = false;

} else {

isOpen = true;

bindingView.ivOpen.setBackgroundResource(R.mipmap.swithch\_on);

}

saveRememberPwd();

});

bindingView.tvLanguage.setOnClickListener(v -> {//语言切换框隐藏

if (bindingView.rvLanguage.getVisibility() == View.GONE) {

bindingView.ivLanguage.setVisibility(View.VISIBLE);

bindingView.rvLanguage.setVisibility(View.VISIBLE);

} else {

bindingView.ivLanguage.setVisibility(View.GONE);

bindingView.rvLanguage.setVisibility(View.GONE);

}

});

bindingView.clContent.setOnClickListener(v -> {//语言切换框隐藏点击事件

if (bindingView.rvLanguage.getVisibility() == View.VISIBLE) {

bindingView.ivLanguage.setVisibility(View.GONE);

bindingView.rvLanguage.setVisibility(View.GONE);

}

});

bindingView.ivHide.setOnClickListener(v -> {//密码明文暗文切换点击事件

if (!isHide) {

bindingView.etPassword.setTransformationMethod(HideReturnsTransformationMethod.getInstance());

// bindingView.ivHide.setImageResource(R.mipmap.login\_icon\_unhide);

} else {

bindingView.etPassword.setTransformationMethod(PasswordTransformationMethod.getInstance());

// bindingView.ivHide.setImageResource(R.mipmap.login\_icon\_hide);

}

bindingView.etPassword.setSelection(bindingView.etPassword.getText().toString().length());

isHide = !isHide;

});

adapter.setOnItemClickListener((adapter, view, position) -> {

if (mList.get(position).isCheck()) {

return;

}

if (position == 0) {

language = "cn";

SP.setLanguage(0);

} else {

language = "en";

SP.setLanguage(1);

}

for (CheckBean c : mList) {

c.setCheck(false);

}

mList.get(position).setCheck(true);

adapter.notifyDataSetChanged();

//重启MainActivity

Intent intent = new Intent(LoginActivity.this, LoginActivity.class);

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

// intent.putExtra("pskValue", pskValue);

startActivity(intent);

overridePendingTransition(R.anim.screen\_zoom\_in, R.anim.screen\_zoom\_out);

finish();

});

}

private void saveRememberPwd() {

String pwdText = bindingView.etPassword.getText().toString().trim();

if (isOpen) {

SP.setPskValue(pwdText);

SP.setRemember(true);

password.set(SP.getPskValue());

} else {

SP.setPskValue("");

SP.setRemember(false);

password.set("");

}

}

}

package com.kingsignal.ks\_link\_wm126.ui.activity.newguide;

import android.content.Context;

import android.content.Intent;

import com.kingsignal.ks\_link\_wm126.R;

import com.kingsignal.ks\_link\_wm126.base.BaseActivity;

import com.kingsignal.ks\_link\_wm126.bean.LoginSetting;

import com.kingsignal.ks\_link\_wm126.databinding.ActivityNewGuideBinding;

import com.kingsignal.ks\_link\_wm126.injection.component.ApplicationComponent;

import com.kingsignal.ks\_link\_wm126.injection.component.DaggerHttpComponent;

import com.kingsignal.ks\_link\_wm126.ui.activity.guide.GuideContract;

import com.kingsignal.ks\_link\_wm126.ui.activity.guide.GuidePresenter;

/\*\*  
 \* @author zwy  
 \* create at 2020-11-22  
 \*/

public class NewGuideStartActivity extends BaseActivity<GuidePresenter, ActivityNewGuideBinding> implements GuideContract.View {

LoginSetting bean;

@Override

public int setContentLayout() {

return R.layout.activity\_new\_guide;

}

@Override

public void initInjector(ApplicationComponent appComponent) {

DaggerHttpComponent.builder().applicationComponent(appComponent).build().inject(this);

}

@Override

public void initView() {

bindingView.tvStart.setOnClickListener(v->{

//跳转到下一步

MainRouterGuideActivity.start(this,bean);

finish();

});

}

@Override

public void initIntent() {

if(null != getIntent()){

bean = (LoginSetting) getIntent().getSerializableExtra(GuideConstant.PSK\_VALUE);

}

}

public static void start(Context context, LoginSetting bean) {

Intent intent = new Intent(context, NewGuideStartActivity.class);

if (null != bean){

intent.putExtra(GuideConstant.PSK\_VALUE, bean);

}

context.startActivity(intent);

}

}

package com.kingsignal.elf1.ui.guide;

import android.content.Context;

import android.content.Intent;

import android.view.KeyEvent;

import android.view.View;

import com.kingsignal.common.base.BasicActivity;

import com.kingsignal.common.utils.GuideConstant;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.bean.LoginSetting;

import com.kingsignal.elf1.databinding.ActivityMianRouterGuideBinding;

import com.kingsignal.elf1.entity.GuideBean;

import com.kingsignal.elf1.utils.DoubleClickHelper;

import java.util.Timer;

import java.util.TimerTask;

/\*\*  
 \* @author zwy  
 \* create at 2020-11-22  
 \*/

public class MainRouterGuideActivity extends BasicActivity<ActivityMianRouterGuideBinding> {

private TimerTask mTimerTask = null;

private Timer mTimer = null;

int count = 3;

GuideBean bean;

public static void start(Context context, GuideBean bean) {

Intent intent = new Intent(context, MainRouterGuideActivity.class);

intent.putExtra(GuideConstant.PSK\_VALUE, bean);

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_mian\_router\_guide;

}

@Override

public void initData() {

if (null != getIntent()) {

bean = (GuideBean) getIntent().getSerializableExtra(GuideConstant.PSK\_VALUE);

}

}

@Override

public void initView() {

bindingView.viewHeader.ivBack.setVisibility(View.GONE);

bindingView.viewHeader.tvSave.setVisibility(View.GONE);

bindingView.viewHeader.tvTitle.setText(getString(R.string.configure\_route\_title));

initListener();

bindingView.tvNext.setEnabled(false);

}

public void initListener() {

bindingView.tvNext.setOnClickListener(view -> {

if (!DoubleClickHelper.isOnDoubleClick()) {

MainRouterBandGuideActivity.start(MainRouterGuideActivity.this, bean);

this.finish();

}

});

}

private void startTimer() {

if (mTimer == null) {

mTimer = new Timer();

}

if (mTimerTask == null) {

mTimerTask = new TimerTask() {

@Override

public void run() {

runOnUiThread(() -> {

if(count>0){

count--;

}

if (count <= 0) {

stopTimer();

bindingView.tvNext.setEnabled(true);

bindingView.tvDown.setVisibility(View.GONE);

} else {

bindingView.tvDown.setVisibility(View.VISIBLE);

bindingView.tvDown.setText(count + "");

bindingView.tvNext.setEnabled(false);

}

});

}

};

}

if (mTimer != null && mTimerTask != null)

mTimer.schedule(mTimerTask, 1000, 1000);

}

private void stopTimer() {

if (mTimer != null) {

mTimer.cancel();

mTimer = null;

}

if (mTimerTask != null) {

mTimerTask.cancel();

mTimerTask = null;

}

}

@Override

protected void onPause() {

super.onPause();

stopTimer();

}

@Override

protected void onResume() {

super.onResume();

startTimer();

}

@Override

public boolean dispatchKeyEvent(KeyEvent event) {

if (event.getKeyCode() == KeyEvent.KEYCODE\_BACK) {

//do something.

return true;

} else {

return super.dispatchKeyEvent(event);

}

}

}

package com.kingsignal.elf1.ui.guide;

import android.content.Context;

import android.content.Intent;

import android.view.KeyEvent;

import android.view.View;

import com.kingsignal.common.base.BasicActivity;

import com.kingsignal.common.utils.GuideConstant;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.databinding.ActivityMianRouterBandGuideBinding;

import com.kingsignal.elf1.databinding.ActivityMianRouterGuideBinding;

import com.kingsignal.elf1.entity.GuideBean;

import com.kingsignal.elf1.utils.DoubleClickHelper;

import java.util.Timer;

import java.util.TimerTask;

/\*\*  
 \* @author zwy  
 \* create at 2020-11-22  
 \*/

public class MainRouterBandGuideActivity extends BasicActivity<ActivityMianRouterBandGuideBinding> {

private TimerTask mTimerTask = null;

private Timer mTimer = null;

int count = 3;

GuideBean bean;

public static void start(Context context, GuideBean bean) {

Intent intent = new Intent(context, MainRouterBandGuideActivity.class);

intent.putExtra(GuideConstant.PSK\_VALUE, bean);

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_mian\_router\_band\_guide;

}

@Override

public void initData() {

if (null != getIntent()) {

bean = (GuideBean) getIntent().getSerializableExtra(GuideConstant.PSK\_VALUE);

}

}

@Override

public void initView() {

bindingView.viewHeader.ivBack.setVisibility(View.GONE);

bindingView.viewHeader.tvSave.setVisibility(View.GONE);

bindingView.viewHeader.tvTitle.setText(getString(R.string.configure\_route\_title));

initListener();

bindingView.tvNext.setEnabled(false);

}

public void initListener() {

bindingView.tvNext.setOnClickListener(view -> {

if (!DoubleClickHelper.isOnDoubleClick()) {

NoConnectActivity.start(MainRouterBandGuideActivity.this);

this.finish();

}

});

}

private void startTimer() {

if (mTimer == null) {

mTimer = new Timer();

}

if (mTimerTask == null) {

mTimerTask = new TimerTask() {

@Override

public void run() {

runOnUiThread(() -> {

if(count>0){

count--;

}

if (count <= 0) {

stopTimer();

bindingView.tvNext.setEnabled(true);

bindingView.tvDown.setVisibility(View.GONE);

} else {

bindingView.tvDown.setVisibility(View.VISIBLE);

bindingView.tvDown.setText(count + "");

bindingView.tvNext.setEnabled(false);

}

});

}

};

}

if (mTimer != null && mTimerTask != null)

mTimer.schedule(mTimerTask, 1000, 1000);

}

private void stopTimer() {

if (mTimer != null) {

mTimer.cancel();

mTimer = null;

}

if (mTimerTask != null) {

mTimerTask.cancel();

mTimerTask = null;

}

}

@Override

protected void onPause() {

super.onPause();

stopTimer();

}

@Override

protected void onResume() {

super.onResume();

startTimer();

}

@Override

public boolean dispatchKeyEvent(KeyEvent event) {

if (event.getKeyCode() == KeyEvent.KEYCODE\_BACK) {

//do something.

return true;

} else {

return super.dispatchKeyEvent(event);

}

}

}

package com.kingsignal.elf1.ui.guide;

import android.app.Dialog;

import android.content.Context;

import android.content.Intent;

import android.os.Build;

import android.text.TextUtils;

import android.view.KeyEvent;

import android.view.View;

import androidx.annotation.RequiresApi;

import com.kingsignal.common.base.MyApplication;

import com.kingsignal.common.base.PresenterActivity;

import com.kingsignal.common.http.NetworkConstant;

import com.kingsignal.common.utils.LogUtil;

import com.kingsignal.common.utils.SP;

import com.kingsignal.common.utils.WifiUtils;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.databinding.ActivityNoConnectBinding;

import com.kingsignal.elf1.dialog.MessageDialog;

import com.kingsignal.elf1.entity.GuideBean;

import com.kingsignal.elf1.presenter.guide.GuidePresenter;

import com.kingsignal.elf1.presenter.guide.GuideView;

import com.kingsignal.elf1.ui.SplashActivity;

import com.kingsignal.elf1.ui.home.MainActivity;

import com.kingsignal.elf1.ui.login.LoginActivity;

import java.util.Timer;

import java.util.TimerTask;

/\*\*  
 \* @author zwy  
 \* create at 2020-11-27  
 \*/

public class NoConnectActivity extends PresenterActivity<GuidePresenter, ActivityNoConnectBinding> implements GuideView {

private TimerTask mTimerTask = null;

private Timer mTimer = null;

boolean isShowPop = true;

/\*\*

\* 无连接跳转到该界面

\*/

public static void start(Context context) {

Intent intent = new Intent(context, NoConnectActivity.class);

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_no\_connect;

}

@Override

public void initData() {

}

@Override

public void initView() {

MyApplication.getInstance().failureJump = false;

bindingView.viewHeader.ivBack.setVisibility(View.GONE);

bindingView.viewHeader.tvSave.setVisibility(View.GONE);

bindingView.viewHeader.tvTitle.setText(getString(R.string.no\_connect\_title));

}

@RequiresApi(api = Build.VERSION\_CODES.M)

@Override

protected void onResume() {

super.onResume();

NetworkConstant.REQUEST\_URL = WifiUtils.getGateWayAddress();

basePresenter.getGuide();

startTimer();

}

@Override

public void onGuideSuccess(GuideBean bean) {

if (bean.getFirst\_login().equals("1")) {

if (!TextUtils.isEmpty(bean.getWan\_proto()) && "dhcp".equals(bean.getWan\_proto())) {

SetPasswordGuideActivity.start(NoConnectActivity.this, bean);

} else {

InternetSettingActivity.start(NoConnectActivity.this, bean);

NoConnectActivity.this.finish();

}

} else {

LoginActivity.start(NoConnectActivity.this);

}

}

@Override

public void onGuideFail() {

if (isShowPop) {

detectionDialog();

}

}

/\*\*

\* 提示用户需要连上产品WiFi

\*/

public void detectionDialog() {

isShowPop = false;

new MessageDialog.Builder(this)

.setTitle("") // 标题可以不用填写

.setMessage(getResources().getString(R.string.detection\_product\_wifi))

.setConfirm("确认")

.setCancel(null) // 设置 null 表示不显示取消按钮

//.setAutoDismiss(false) // 设置点击按钮后不关闭对话框

.setListener(new MessageDialog.OnListener() {

@Override

public void onConfirm(Dialog dialog) {

}

@Override

public void onCancel(Dialog dialog) {

}

})

.show();

}

@Override

public void onSetSuccess() {

}

@Override

protected void onPause() {

super.onPause();

stopTimer();

if (basePresenter.disposable != null && !basePresenter.disposable.isDisposed()) {

basePresenter.disposable.dispose();//取消订阅

}

}

private void startTimer() {

stopTimer();

if (mTimer == null) {

mTimer = new Timer();

}

if (mTimerTask == null) {

mTimerTask = new TimerTask() {

@Override

public void run() {

basePresenter.getGuide();

}

};

}

if (mTimer != null && mTimerTask != null) {

mTimer.schedule(mTimerTask, 0, 5000);

}

}

private void stopTimer() {

if (mTimerTask != null) {

mTimerTask.cancel();

mTimerTask = null;

}

if (mTimer != null) {

mTimer.cancel();

mTimer = null;

}

}

@Override

protected void onDestroy() {

super.onDestroy();

MyApplication.getInstance().failureJump = true;

}

@Override

public boolean dispatchKeyEvent(KeyEvent event) {

if (event.getKeyCode() == KeyEvent.KEYCODE\_BACK) {

//do something.

return true;

} else {

return super.dispatchKeyEvent(event);

}

}

}

package com.kingsignal.elf1.ui.guide;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

import android.os.Handler;

import android.os.Message;

import android.view.KeyEvent;

import android.view.View;

import com.kingsignal.common.base.PresenterActivity;

import com.kingsignal.common.http.NetworkConstant;

import com.kingsignal.common.utils.GuideConstant;

import com.kingsignal.elf1.R;

import com.kingsignal.elf1.databinding.ActivityPasswordGuideBinding;

import com.kingsignal.elf1.entity.GuideBean;

import com.kingsignal.elf1.presenter.guide.SetPasswordPresenter;

import com.kingsignal.elf1.ui.login.LoginActivity;

import com.kingsignal.elf1.ui.setting.online.MainInternetSettingActivity;

import com.kingsignal.elf1.utils.DoubleClickHelper;

import com.kingsignal.elf1.utils.KeyBoardUtils;

import java.lang.ref.WeakReference;

/\*\*  
 \* @author zwy  
 \* create at 2020-11-26  
 \*/

public class SetPasswordGuideActivity extends PresenterActivity<SetPasswordPresenter, ActivityPasswordGuideBinding> {

GuideBean bean;

boolean bool = true;

MyHandler handler;

public static void start(Context context, GuideBean bean) {

Intent intent = new Intent(context, SetPasswordGuideActivity.class);

if (null != bean) {

intent.putExtra(GuideConstant.PSK\_VALUE, bean);

}

context.startActivity(intent);

}

@Override

public int getLayoutId() {

return R.layout.activity\_password\_guide;

}

@Override

public void initData() {

if (null != getIntent()) {

bean = (GuideBean) getIntent().getSerializableExtra(GuideConstant.PSK\_VALUE);

}

}

@Override

public void initView() {

bindingView.viewHeader.ivBack.setVisibility(View.GONE);

bindingView.viewHeader.tvSave.setVisibility(View.GONE);

bindingView.viewHeader.tvTitle.setText(getString(R.string.guide\_pwd\_title));

bindingView.edtName.setText(bean.getSsid\_5g());

initListener();

}

public void initListener() {

bindingView.checkbox.setOnCheckedChangeListener((compoundButton, b) -> {

bool = b;

setVisibleManager(b);

});

bindingView.tvNext.setOnClickListener(view -> {

KeyBoardUtils.hideInputForce(this);

if (!DoubleClickHelper.isOnDoubleClick()) {

basePresenter.setGuide(bean, bindingView.edtName.getText().toString()

, bindingView.edtPwd.getText().toString()

, bindingView.edtManagerPwd.getText().toString()

, bindingView.edtConManagerPwd.getText().toString()

, bool

);

}

});

}

public void setVisibleManager(Boolean b) {

if (!b) {

// bindingView.tvManagerTips.setVisibility(View.VISIBLE);

bindingView.cvManager.setVisibility(View.VISIBLE);

} else {

// bindingView.tvManagerTips.setVisibility(View.GONE);

bindingView.cvManager.setVisibility(View.GONE);

}

}

public void onDataSuccess() {

showToast(R.string.reset\_router);

handler = new SetPasswordGuideActivity.MyHandler(this);

handler.sendEmptyMessageDelayed(1, 2000);

}

public void onDataFailure() {

// showToast(R.string.reset\_router);

// handler = new SetPasswordGuideActivity.MyHandler(this);

// handler.sendEmptyMessageDelayed(1, 2000);

}

public void run() {

NetworkDetectionActivity.start(this, NetworkConstant.DETECTION\_GUIDE);

this.finish();

}

static class MyHandler extends Handler {

// SoftReference<Activity> 也可以使用软应用 只有在内存不足的时候才会被回收

private final WeakReference<SetPasswordGuideActivity> mActivity;

private MyHandler(SetPasswordGuideActivity activity) {

mActivity = new WeakReference<>(activity);

}

@Override

public void handleMessage(Message msg) {

Activity activity = mActivity.get();

if (activity != null) {

mActivity.get().run();

}

super.handleMessage(msg);

}

}

@Override

public boolean dispatchKeyEvent(KeyEvent event) {

if (event.getKeyCode() == KeyEvent.KEYCODE\_BACK) {

//do something.

return true;

} else {

return super.dispatchKeyEvent(event);

}

}

}