package com.example.administrator.timenote.Manager.DremindManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/20.  
 \*/*public class ChangeAhead {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "UpdateAhead";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//UpdateAhead/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(int userid, int eventid, int aheadtime) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", userid);  
 rpc.addProperty("eventid",eventid);  
 rpc.addProperty("aheadtime",aheadtime);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse() != null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.DremindManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/21.  
 \*/*public class ChangeDRTime {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "UpdateDRTTime";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//UpdateDRTTime/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(int userid, int eventid, String defaulttime) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", userid);  
 rpc.addProperty("eventid",eventid);  
 rpc.addProperty("defaulttime",defaulttime);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse() != null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
  
 public static void main(String[] args){  
 ChangeDRTime changeDRTime = new ChangeDRTime();  
 changeDRTime.getRemoteInfo(9,-99,"0");  
 }  
}

package com.example.administrator.timenote.Manager.DremindManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/20.  
 \*/*public class ChangeLeavesTime {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "UpdateLeavesTime";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//UpdateLeavesTime/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(int userid, int eventid, int leavestime) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", userid);  
 rpc.addProperty("eventid",eventid);  
 rpc.addProperty("leavestime",leavestime);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse() != null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.DremindManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/21.  
 \*/*public class ChangeRepeat {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "UpdateRepeat";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//UpdateRepeat/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(int userid, int eventid, int dremindrepeat) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", userid);  
 rpc.addProperty("eventid",eventid);  
 rpc.addProperty("dremindrepeat",dremindrepeat);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse() != null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
  
 public static void main(String[] args){  
 ChangeRepeat changeDRTime = new ChangeRepeat();  
 changeDRTime.getRemoteInfo(9,-99,0);  
 }  
}

package com.example.administrator.timenote.Manager.DremindManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/21.  
 \*/*public class ChangeRing {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "UpdateRing";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//UpdateRing/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(int userid, int eventid, int dremindring) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", userid);  
 rpc.addProperty("eventid",eventid);  
 rpc.addProperty("dremindring",dremindring);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse() != null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
  
 public static void main(String[] args){  
 ChangeRing changeDRTime = new ChangeRing();  
 changeDRTime.getRemoteInfo(9,-99,0);  
 }  
}

package com.example.administrator.timenote.Manager.DremindManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/20.  
 \*/*public class ChangeVib {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "UpdateVib";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//UpdateVib/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(int userid, int eventid, int dremindvib) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", userid);  
 rpc.addProperty("eventid",eventid);  
 rpc.addProperty("dremindvib",dremindvib);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse() != null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.DremindManager;  
  
import com.example.administrator.timenote.Manager.TaskManager.LoadAllEvent;  
import com.example.administrator.timenote.Model.BeanDRemindInformation;  
import com.example.administrator.timenote.Model.BeanEventInformation;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/20.  
 \*/*public class LoadDremind {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "LoadDefaultSet";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//LoadDefaultSet/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint,60000\*5);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public BeanDRemindInformation getRemoteInfo(int userid,int eventid) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", userid);  
 rpc.addProperty("eventid", eventid);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 // 等价于envelope.bodyOut = rpc; envelope.setOutputSoapObject(rpc);  
 transport.debug = false;  
 BeanDRemindInformation dr = new BeanDRemindInformation();  
 try {  
 transport.call(soapAction, envelope);  
 SoapObject result = (SoapObject) envelope.getResponse();  
 if (result == null)  
 return null;  
 dr.setUserid(Integer.*parseInt*(result.getProperty("Userid").toString()));  
 dr.setEventid(Integer.*parseInt*(result.getProperty("Eventid").toString()));  
 dr.setDremindid(Integer.*parseInt*(result.getProperty("Dremindid").toString()));  
 dr.setDremindring(Integer.*parseInt*(result.getProperty("Dremindring").toString()));  
 dr.setDremindrepeat(Integer.*parseInt*(result.getProperty("Dremindrepeat").toString()));  
 dr.setAheadtime(Integer.*parseInt*(result.getProperty("Aheadtime").toString()));  
 dr.setPid(Integer.*parseInt*(result.getProperty("Pid").toString()));  
 dr.setLeavestime(Integer.*parseInt*(result.getProperty("Leavestime").toString()));  
 dr.setDefaulttime(result.getProperty("Defaulttime").toString());  
 dr.setDremindvib(result.getProperty("Dremindvib").toString());  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return dr;  
 }  
  
 public static void main(String[] args){  
 LoadDremind loadDremind = new LoadDremind();  
 loadDremind.getRemoteInfo(9,200);  
 }  
}

package com.example.administrator.timenote.Manager.DremindManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/22.  
 \*/*public class NewDRemind {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "NewDRemind";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//NewDRemind/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(int userid, int eventid) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", userid);  
 rpc.addProperty("eventid",eventid);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse() != null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.LeavesManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/22.  
 \*/*public class LeavesAdd {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "LeavesAdd";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//LeavesAdd/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(int userid) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", userid);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse() != null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.LeavesManager;  
  
import com.example.administrator.timenote.Model.BeanLeavesStatistics;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/22.  
 \*/*public class LoadLeaves {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "LoadLeaves";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//LoadLeaves/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint,60000\*2);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public Boolean getRemoteInfo(int userid) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", userid);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 // 等价于envelope.bodyOut = rpc; envelope.setOutputSoapObject(rpc);  
 transport.debug = false;  
 try {  
 transport.call(soapAction, envelope);  
 SoapObject result = (SoapObject) envelope.getResponse();  
 BeanLeavesStatistics ls = new BeanLeavesStatistics();  
 ls.setUserid(Integer.*parseInt*(result.getProperty("Userid").toString()));  
 ls.setLeavesamount(Integer.*parseInt*(result.getProperty("Leavesamount").toString()));  
 ls.setFocustime(Integer.*parseInt*(result.getProperty("Focustime").toString()));  
 ls.setLeavesdate(*stringToDate*(result.getProperty("Leavesdate").toString().replaceAll("T"," "), "yyyy-MM-dd HH:mm:ss"));  
 BeanLeavesStatistics.*leavesStatistics* = ls;  
 } catch (Exception e) {  
 e.printStackTrace();  
 return false;  
 }  
 return true;  
 }  
  
 public static Date stringToDate(String strTime, String formatType) {  
 SimpleDateFormat formatter = new SimpleDateFormat(formatType);  
 Date date = null;  
 try {  
 date = formatter.parse(strTime);  
 } catch (ParseException e) {  
 // *TODO Auto-generated catch block* e.printStackTrace();  
 }  
 return date;  
 }  
  
 public static void main(String[] args){  
 LoadLeaves loadLeaves = new LoadLeaves();  
 loadLeaves.getRemoteInfo(9);  
 }  
}

package com.example.administrator.timenote.Manager.ListManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/18.  
 \*/*public class DeleteList {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "DeleteList";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//DeleteList/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(int listid) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("listid",listid);  
  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse()!=null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.ListManager;  
  
import com.example.administrator.timenote.Manager.TaskManager.LoadAllEvent;  
import com.example.administrator.timenote.Model.BeanEventInformation;  
import com.example.administrator.timenote.Model.BeanListInformation;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
import static com.example.administrator.timenote.Model.BeanListInformation.*allList*;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/17.  
 \*/*public class LoadAllList {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "AllList";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//AllList/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint,60000\*5);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public Boolean getRemoteInfo(int userid) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", userid);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 // 等价于envelope.bodyOut = rpc; envelope.setOutputSoapObject(rpc);  
 transport.debug = false;  
 List<BeanListInformation> allListLoad = new ArrayList<>();  
 try {  
 transport.call(soapAction, envelope);  
 SoapObject result = (SoapObject) envelope.bodyIn;  
 SoapObject detail = (SoapObject) result.getProperty(methodName+"Result");  
 for (int j = 0; j < detail.getPropertyCount(); j++) {  
 BeanListInformation l = new BeanListInformation();  
 SoapObject mstr = (SoapObject) detail.getProperty(j);  
 l.setUserid(Integer.*parseInt*(mstr.getProperty("Userid").toString()));  
 l.setListid(Integer.*parseInt*(mstr.getProperty("Listid").toString()));  
 l.setListname(mstr.getProperty("Listname").toString());  
 l.setListpriority(Integer.*parseInt*(mstr.getProperty("Listpriority").toString()));  
 l.setCreatedate(*stringToDate*(mstr.getProperty("Createdate").toString().replaceAll("T"," "), "yyyy-MM-dd HH:mm:ss"));  
 allListLoad.add(l);  
 }  
 *allList* = allListLoad;  
 } catch (Exception e) {  
 e.printStackTrace();  
 return false;  
 }  
 return true;  
 }  
  
 public static Date stringToDate(String strTime, String formatType) {  
 SimpleDateFormat formatter = new SimpleDateFormat(formatType);  
 Date date = null;  
 try {  
 date = formatter.parse(strTime);  
 } catch (ParseException e) {  
 // *TODO Auto-generated catch block* e.printStackTrace();  
 }  
 return date;  
 }  
  
 public static void main(String[] args){  
 LoadAllList loadAllList = new LoadAllList();  
 loadAllList.getRemoteInfo(9);  
 }  
}

package com.example.administrator.timenote.Manager.ListManager;  
  
import com.example.administrator.timenote.Model.BeanUserInformation;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/12.  
 \*/*public class NewList {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "NewList";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//NewList/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(String listname) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", BeanUserInformation.*currentLoginUser*.getUserid());  
 rpc.addProperty("listname", listname);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse() != null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.ListManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/18.  
 \*/*public class UpdateList {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "UpdateList";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//UpdateList/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(String listname, int listid) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("listname", listname);  
 rpc.addProperty("listid",listid);  
  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse()!=null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.TaskManager;  
  
import com.example.administrator.timenote.Model.BeanUserInformation;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/17.  
 \*/*public class Complete {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "EventComplete";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//EventComplete/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(int eventid) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("eventid", eventid);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse() != null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.TaskManager;  
  
import com.example.administrator.timenote.Model.BeanUserInformation;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/18.  
 \*/*public class DeleteEvent {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "DeleteEvent";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//DeleteEvent/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(int eventid) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("eventid", eventid);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse() != null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.TaskManager;  
  
import com.example.administrator.timenote.Model.BeanEventInformation;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/10.  
 \*/*public class LoadAllEvent {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "AllEvent";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//AllEvent/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint,60000\*5);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public Boolean getRemoteInfo(int userid) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", userid);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 // 等价于envelope.bodyOut = rpc; envelope.setOutputSoapObject(rpc);  
 transport.debug = false;  
 List<BeanEventInformation> allEventListByDate = new ArrayList<>();  
 try {  
 transport.call(soapAction, envelope);  
// String a = transport.requestDump;  
// System.out.println(a);  
// String b = transport.responseDump;  
// System.out.println(b);  
 SoapObject result = (SoapObject) envelope.bodyIn;  
 SoapObject detail = (SoapObject) result.getProperty(methodName+"Result");  
 for (int j = 0; j < detail.getPropertyCount(); j++) {  
 BeanEventInformation e = new BeanEventInformation();  
 SoapObject mstr = (SoapObject) detail.getProperty(j);  
 e.setUserid(Integer.*parseInt*(mstr.getProperty("Userid").toString()));  
 e.setEventid(Integer.*parseInt*(mstr.getProperty("Eventid").toString()));  
 e.setDremindid(Integer.*parseInt*(mstr.getProperty("Dremindid").toString()));  
 e.setListid(Integer.*parseInt*(mstr.getProperty("Listid").toString()));  
 e.setEventname(mstr.getProperty("Eventname").toString());  
 e.setEventpriority(Integer.*parseInt*(mstr.getProperty("Eventpriority").toString()));  
 e.setEventdate(*stringToDate*(mstr.getProperty("Eventdate").toString().replaceAll("T"," "), "yyyy-MM-dd HH:mm:ss"));  
 e.setDreminddate(*stringToDate*(mstr.getProperty("Dreminddate").toString().replaceAll("T"," "), "yyyy-MM-dd HH:mm:ss"));  
 e.setLeaveseventsign(mstr.getProperty("Leaveseventsign").toString());  
 if(!mstr.getProperty("Eventnote").toString().equals("anyType{}")) e.setEventnote(mstr.getProperty("Eventnote").toString());  
 e.setCheckBox(Integer.*parseInt*(mstr.getProperty("Complete").toString()));  
 e.setPid(Integer.*parseInt*(mstr.getProperty("Pid").toString()));  
 allEventListByDate.add(e);  
 }  
 BeanEventInformation.*allEventList* = allEventListByDate;  
 } catch (Exception e) {  
 e.printStackTrace();  
 return false;  
 }  
 return true;  
 }  
  
 public static Date stringToDate(String strTime, String formatType) {  
 SimpleDateFormat formatter = new SimpleDateFormat(formatType);  
 Date date = null;  
 try {  
 date = formatter.parse(strTime);  
 } catch (ParseException e) {  
 // *TODO Auto-generated catch block* e.printStackTrace();  
 }  
 return date;  
 }  
  
 public static void main(String[] args){  
 LoadAllEvent loadAllEvent = new LoadAllEvent();  
 loadAllEvent.getRemoteInfo(9);  
 }  
}

package com.example.administrator.timenote.Manager.TaskManager;  
  
import com.example.administrator.timenote.Model.BeanEventInformation;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/17.  
 \*/*public class LoadAllEventByPro {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "AllEventByPro";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//AllEventByPro/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint,60000\*5);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public Boolean getRemoteInfo(int userid) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", userid);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 // 等价于envelope.bodyOut = rpc; envelope.setOutputSoapObject(rpc);  
 transport.debug = false;  
 //List<BeanEventInformation> eventInformationList = new ArrayList<>();  
 List<BeanEventInformation> allEventListByPro = new ArrayList<>();  
 try {  
 transport.call(soapAction, envelope);  
 SoapObject result = (SoapObject) envelope.bodyIn;  
 SoapObject detail = (SoapObject) result.getProperty(methodName+"Result");  
  
 for (int j = 0; j < detail.getPropertyCount(); j++) {  
 BeanEventInformation e = new BeanEventInformation();  
 SoapObject mstr = (SoapObject) detail.getProperty(j);  
 e.setUserid(Integer.*parseInt*(mstr.getProperty("Userid").toString()));  
 e.setEventid(Integer.*parseInt*(mstr.getProperty("Eventid").toString()));  
 e.setDremindid(Integer.*parseInt*(mstr.getProperty("Dremindid").toString()));  
 e.setListid(Integer.*parseInt*(mstr.getProperty("Listid").toString()));  
 e.setEventname(mstr.getProperty("Eventname").toString());  
 e.setEventpriority(Integer.*parseInt*(mstr.getProperty("Eventpriority").toString()));  
 e.setEventdate(*stringToDate*(mstr.getProperty("Eventdate").toString().replaceAll("T"," "), "yyyy-MM-dd HH:mm:ss"));  
 e.setDreminddate(*stringToDate*(mstr.getProperty("Dreminddate").toString().replaceAll("T"," "), "yyyy-MM-dd HH:mm:ss"));  
 e.setLeaveseventsign(mstr.getProperty("Leaveseventsign").toString());  
 if(!mstr.getProperty("Eventnote").toString().equals("anyType{}")) e.setEventnote(mstr.getProperty("Eventnote").toString());  
 e.setCheckBox(Integer.*parseInt*(mstr.getProperty("Complete").toString()));  
 allEventListByPro.add(e);  
 }  
 BeanEventInformation.*allEventList* = allEventListByPro;  
 } catch (Exception e) {  
 e.printStackTrace();  
 return false;  
 }  
 return true;  
 }  
  
 public static Date stringToDate(String strTime, String formatType) {  
 SimpleDateFormat formatter = new SimpleDateFormat(formatType);  
 Date date = null;  
 try {  
 date = formatter.parse(strTime);  
 } catch (ParseException e) {  
 // *TODO Auto-generated catch block* e.printStackTrace();  
 }  
 return date;  
 }  
  
 public static void main(String[] args){  
 LoadAllEvent loadAllEvent = new LoadAllEvent();  
 loadAllEvent.getRemoteInfo(9);  
 }  
}

package com.example.administrator.timenote.Manager.TaskManager;  
  
import com.example.administrator.timenote.Model.BeanUserInformation;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/9.  
 \*/*public class NewEvent {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "NewEvent";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//NewEvent/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(String eventname,int eventpriority, int listid, String eventdate) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", BeanUserInformation.*currentLoginUser*.getUserid());  
 rpc.addProperty("eventname", eventname);  
 rpc.addProperty("eventpriority",eventpriority);  
 rpc.addProperty("listid",listid);  
 rpc.addProperty("eventdate",eventdate);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse()!=null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
  
 public static void main(String[] args){  
 NewEvent newEvent = new NewEvent();  
 newEvent.getRemoteInfo("旅游",1,0,"2018-09-19 09:30");  
 }  
}

package com.example.administrator.timenote.Manager.TaskManager;  
  
import com.example.administrator.timenote.Model.BeanUserInformation;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/17.  
 \*/*public class UnComplete {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "EventUnComplete";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//EventUnComplete/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(int eventid) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("eventid", eventid);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse() != null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.TaskManager;  
  
import com.example.administrator.timenote.Model.BeanUserInformation;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/18.  
 \*/*public class UpdateEvent {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "UpdateEvent";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//UpdateEvent/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(int eventid, String eventname,int eventpriority, int listid, String eventdate, int leaveseventsign, String eventnote) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("eventid", eventid);  
 rpc.addProperty("eventname", eventname);  
 rpc.addProperty("eventpriority",eventpriority);  
 rpc.addProperty("listid",listid);  
 rpc.addProperty("eventdate",eventdate);  
 rpc.addProperty("leaveseventsign",leaveseventsign);  
 rpc.addProperty("eventnote",eventnote);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse()!=null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.TaskManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/26.  
 \*/*public class UpdatePid {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "UpdatePid";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//UpdatePid/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(int eventid, int pid) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("eventid", eventid);  
 rpc.addProperty("pid", pid);  
  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse()!=null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
  
 public static void main(String[] args){  
 UpdatePid updatePid = new UpdatePid();  
 updatePid.getRemoteInfo(146,534665);  
 }  
}

package com.example.administrator.timenote.Manager.TaskManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/18.  
 \*/*public class UpdatePriority {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "UpdateEventPriority";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//UpdateEventPriority/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(int eventid, int eventpriority) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("eventid", eventid);  
 rpc.addProperty("eventpriority",eventpriority);  
  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse()!=null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.UserManager;  
  
import com.example.administrator.timenote.Model.BeanUserInformation;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/20.  
 \*/*public class ChangeEmail {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "UpdateEmail";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//UpdateEmail/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint,60000\*5);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(String useremail) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", BeanUserInformation.*tryLoginUser*.getUserid());  
 rpc.addProperty("useremail",useremail);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 // 等价于envelope.bodyOut = rpc; envelope.setOutputSoapObject(rpc);  
 transport.debug = false;  
 String result = "false";  
 try {  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse() != null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.UserManager;  
  
import com.example.administrator.timenote.Model.BeanUserInformation;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/20.  
 \*/*public class ChangeName {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "UpdateName";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//UpdateName/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint,60000\*5);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(String username) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", BeanUserInformation.*tryLoginUser*.getUserid());  
 rpc.addProperty("username", username);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 // 等价于envelope.bodyOut = rpc; envelope.setOutputSoapObject(rpc);  
 transport.debug = false;  
 String result = "false";  
 try {  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse() != null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.UserManager;  
  
import android.util.Base64;  
  
import com.example.administrator.timenote.Model.BeanUserInformation;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/19.  
 \*/*public class ChangePassword {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "UpdateUserpassward";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//UpdateUserpassward/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint,60000\*5);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(String userpassword) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid", BeanUserInformation.*tryLoginUser*.getUserid());  
 rpc.addProperty("userpassword",userpassword);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 // 等价于envelope.bodyOut = rpc; envelope.setOutputSoapObject(rpc);  
 transport.debug = false;  
 String result = "false";  
 try {  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse() != null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.UserManager;  
  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class Email\_ck {  
 // 判断格式是否为email  
 public static boolean isEmail(String email) {  
 String str = "^([a-zA-Z0-9\_\\-\\.]+)@((\\[[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.)|(([a-zA-Z0-9\\-]+\\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\\]?)$";  
 Pattern p = Pattern.*compile*(str);  
 Matcher m = p.matcher(email);  
 return m.matches();  
 }  
  
}

package com.example.administrator.timenote.Manager.UserManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by Sprou on 2018/6/6.  
 \*/*public class EmailResend {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "SendAuthCode";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//SendAuthCode/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(String useremail) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("useremail", useremail);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 // 等价于envelope.bodyOut = rpc; envelope.setOutputSoapObject(rpc);  
 transport.debug = false;  
  
 String result = "false";  
 try {  
 transport.call(soapAction, envelope);  
 result = envelope.getResponse().toString();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.UserManager;  
  
import android.util.Base64;  
  
import com.example.administrator.timenote.Model.BeanUserInformation;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/13.  
 \*/*public class GetHead {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "GetHead";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//GetHead/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public void getRemoteInfo() {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("userid",9);  
 //rpc.addProperty("head", head);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 // 等价于envelope.bodyOut = rpc; envelope.setOutputSoapObject(rpc);  
 transport.debug = false;  
 try {  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse()!=null) {  
 //result = envelope.getResponse().toString().getBytes();  
 byte[] a = Base64.*decode*(envelope.getResponse().toString(), Base64.*DEFAULT*);  
 BeanUserInformation.*currentLoginUser*.setIcon(a);  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static void main(String[] args){  
 GetHead getHead = new GetHead();  
 getHead.getRemoteInfo();  
 }  
}

package com.example.administrator.timenote.Manager.UserManager;  
  
import android.content.Context;  
  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.InputStreamReader;  
import java.util.HashMap;  
import java.util.Map;  
  
public class loginServe {  
 public static boolean saveUserInfo(Context context,String username,String password){  
 try {  
 File file = new File(context.getFilesDir(),"info.txt");  
 FileOutputStream fos = new FileOutputStream(file);  
 fos.write((username+"##"+password).getBytes());  
 fos.close();  
 return true;  
 } catch (Exception e) {  
 // *TODO 自动生成 catch 块* e.printStackTrace();  
 return false;  
 }  
 }  
  
 //获取保存的数据  
 public static Map<String, String> getSaveUserInfo(Context context){  
 File file = new File(context.getFilesDir(),"info.txt");  
 try {  
 FileInputStream fis = new FileInputStream(file);  
 BufferedReader br = new BufferedReader(new InputStreamReader(fis));  
 String str = br.readLine();  
 String[] infos = str.split("##");  
 Map<String, String> map = new HashMap<String,String>();  
 map.put("useremail", infos[0]);  
 map.put("password", infos[1]);  
 return map;  
 } catch (Exception e) {  
 // *TODO 自动生成 catch 块* e.printStackTrace();  
 return null;  
 }  
 }  
}

package com.example.administrator.timenote.Manager.UserManager;  
  
import android.util.Base64;  
  
import com.example.administrator.timenote.Model.BeanUserInformation;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/13.  
 \*/*public class SaveHead {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "SaveHead";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//SaveHead/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint,60000\*5);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public String getRemoteInfo(byte[] head) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("head", Base64.*encodeToString*(head, Base64.*DEFAULT*));  
 rpc.addProperty("userid",BeanUserInformation.*currentLoginUser*.getUserid());  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 // 等价于envelope.bodyOut = rpc; envelope.setOutputSoapObject(rpc);  
 transport.debug = false;  
 String result = "false";  
 try {  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse()!=null) {  
 result = envelope.getResponse().toString();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
}

package com.example.administrator.timenote.Manager.UserManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by Sprou on 2018/6/6.  
 \*/*public class UserActivity {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "DeleteStopDate";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//DeleteStopDate/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public void getRemoteInfo(String useremail) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("useremail", useremail);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 // 等价于envelope.bodyOut = rpc; envelope.setOutputSoapObject(rpc);  
 transport.debug = false;  
 //Message msg = MainActivity.myHandler.obtainMessage();  
 try {  
 transport.call(soapAction, envelope);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

package com.example.administrator.timenote.Manager.UserManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by Sprou on 2018/5/28.  
 \*/*public class UserCreate {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "CreateUser";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//CreateUser/";  
 HttpTransportSE transport = new HttpTransportSE(endPoint,60000\*5);  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
 public String getRemoteInfo(String useremail, String userpwd, String username) {  
  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("useremail", useremail);  
 rpc.addProperty("userpwd", userpwd);  
 rpc.addProperty("username", username);  
  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 // 等价于envelope.bodyOut = rpc; envelope.setOutputSoapObject(rpc);  
  
 transport.debug = false;  
 String result = "false";  
 try {  
 // 调用WebService  
 transport.call(soapAction, envelope);  
 if (envelope.getResponse()!=null) {  
 result = envelope.getResponse().toString();  
 System.*out*.println(result);  
 }  
 } catch (Exception e) {  
 e.getMessage();  
 }  
 return result;  
 }  
 public static void main(String[] args){  
 UserCreate userCreate = new UserCreate();  
 String sign = userCreate.getRemoteInfo("3469109048@qq.com","12345678","sproutgod");  
 System.*out*.println(sign);  
 }  
}

package com.example.administrator.timenote.Manager.UserManager;  
  
  
import android.os.Message;  
import android.util.Base64;  
import android.util.Base64DataException;  
import android.util.Log;  
import android.widget.Toast;  
  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.Ui.MainActivity;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.SoapFault;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
import org.xmlpull.v1.XmlPullParserException;  
  
import java.io.IOException;  
import java.lang.annotation.Target;  
import java.util.Date;  
import java.util.List;  
  
import static com.example.administrator.timenote.Model.BeanUserInformation.*tryLoginUser*;  
  
*/\*\*  
 \* Created by Sprou on 2018/5/26.  
 \*/*public class UserLogin {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "LoadUser";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//LoadUser/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public void getRemoteInfo(String useremail) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("useremail", useremail);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 // 等价于envelope.bodyOut = rpc; envelope.setOutputSoapObject(rpc);  
 transport.debug = false;  
 //Message msg = MainActivity.myHandler.obtainMessage();  
 BeanUserInformation u = new BeanUserInformation();  
 try {  
 transport.call(soapAction, envelope);  
// String a = transport.requestDump;  
// System.out.println(a);  
// String b = transport.responseDump;  
// System.out.println(b);  
 if (envelope.getResponse() == null) throw new Exception("没有该用户");  
 SoapObject result = (SoapObject) envelope.getResponse();  
 u.setUserid(Integer.*parseInt*(result.getProperty("Userid").toString()));  
 if(!result.getProperty("Username").toString().equals("anyType{}")) u.setUsername(result.getProperty("Username").toString());  
 if(!result.getProperty("Userpassword").toString().equals("anyType{}")) u.setUserpassword(result.getProperty("Userpassword").toString());  
 u.setUseremail(result.getProperty("Useremail").toString());  
 u.setCreatdate(result.getProperty("Creatdate").toString());  
 u.setStopdate(result.getProperty("Stopdate").toString());  
 u.setUsercalendar(result.getProperty("Usercalendar").toString());  
 u.setUsertypeface(result.getProperty("Usertypeface").toString());  
 u.setAchievement(Integer.*parseInt*(result.getProperty("Achievement").toString()));  
 u.setLeavesid(Integer.*parseInt*(result.getProperty("Leavesid").toString()));  
 u.setAuthcode(result.getProperty("Authcode").toString());  
 u.setIcon(Base64.*decode*(result.getProperty("Userhead").toString(), Base64.*DEFAULT*));  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 BeanUserInformation.*tryLoginUser*=u;  
 }  
 public static void main(String[] args){  
 UserLogin userLogin = new UserLogin();  
 userLogin.getRemoteInfo("1191583108@qq.com");  
 System.*out*.println(*tryLoginUser*.getUseremail());  
 System.*out*.println(*tryLoginUser*.getUserpassword());  
 }  
}

package com.example.administrator.timenote.Manager.UserManager;  
  
import org.ksoap2.SoapEnvelope;  
import org.ksoap2.serialization.MarshalBase64;  
import org.ksoap2.serialization.SoapObject;  
import org.ksoap2.serialization.SoapSerializationEnvelope;  
import org.ksoap2.transport.HttpTransportSE;  
  
*/\*\*  
 \* Created by Sprou on 2018/6/6.  
 \*/*public class UserRollBack {  
 // 命名空间  
 String nameSpace = "http://tempuri.org/";  
 // 调用的方法名称  
 String methodName = "UserRollBack";  
 // EndPoint  
 String endPoint = "http://39.108.124.121:5818/WebService1.asmx";  
 // SOAP Action  
 String soapAction = "http://tempuri.org//UserRollBack/";  
  
 // 生成调用WebService方法的SOAP请求信息,并指定SOAP的版本  
 SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.*VER12*);  
  
 HttpTransportSE transport = new HttpTransportSE(endPoint);  
  
 //创建子线程并引用webservice层的LoadUser方法  
 public void getRemoteInfo(String useremail) {  
 // 指定WebService的命名空间和调用的方法名  
 SoapObject rpc = new SoapObject(nameSpace, methodName);  
 // 设置需调用WebService接口需要传入的两个参数mobileCode、userId  
 rpc.addProperty("useremail", useremail);  
 envelope.bodyOut = rpc;  
 // 设置是否调用的是dotNet开发的WebService  
 envelope.dotNet = true;  
 (new MarshalBase64()).register(envelope);  
 // 等价于envelope.bodyOut = rpc; envelope.setOutputSoapObject(rpc);  
 transport.debug = false;  
 //Message msg = MainActivity.myHandler.obtainMessage();  
 try {  
 transport.call(soapAction, envelope);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

package com.example.administrator.timenote.Manager;  
  
*/\*\*  
 \* Created by Sprou on 2018/6/6.  
 \*/*import java.io.IOException;  
  
import org.ksoap2.transport.HttpTransportSE;  
import org.ksoap2.transport.ServiceConnection;  
  
*/\*\*  
 \* 继承了HttpTransportSE ,添加请求超时的操作  
 \*  
 \** ***@author*** *wsx  
 \*  
 \*/*public class MyAndroidHttpTransport extends HttpTransportSE {  
  
 private int timeout = 30000; // 默认超时时间为30s  
  
 public MyAndroidHttpTransport(String url) {  
 super(url);  
 }  
  
 public MyAndroidHttpTransport(String url, int timeout) {  
 super(url);  
 this.timeout = timeout;  
 }  
  
 @Override  
 public ServiceConnection getServiceConnection() throws IOException {  
 ServiceConnectionSE serviceConnection = new ServiceConnectionSE(url);  
 serviceConnection.setConnectionTimeOut(timeout);  
  
 return serviceConnection;  
 }  
}

package com.example.administrator.timenote.Manager;  
  
*/\*\*  
 \* Created by Sprou on 2018/6/6.  
 \*/*import java.io.IOException;  
import java.io.InputStream;  
import java.io.OutputStream;  
import java.net.HttpURLConnection;  
import java.net.URL;  
import java.util.List;  
  
import org.ksoap2.transport.ServiceConnection;  
  
*/\*\*  
 \* 继承了HttpTransportSE ,添加请求超时的操作  
 \*  
 \** ***@author*** *wsx  
 \*  
 \*/*public class ServiceConnectionSE implements ServiceConnection {  
 private HttpURLConnection connection;  
  
 public ServiceConnectionSE(String url) throws IOException {  
 this.connection = ((HttpURLConnection) new URL(url).openConnection());  
 this.connection.setUseCaches(false);  
 this.connection.setDoOutput(true);  
 this.connection.setDoInput(true);  
 }  
  
 public void connect() throws IOException {  
 this.connection.connect();  
 }  
  
 public void disconnect() {  
 this.connection.disconnect();  
 }  
  
 @Override  
 public List getResponseProperties() throws IOException {  
 return (List) this.connection.getRequestProperties();  
 }  
  
 @Override  
 public int getResponseCode() throws IOException {  
 return this.connection.getResponseCode();  
 }  
  
 public void setRequestProperty(String string, String soapAction) {  
 this.connection.setRequestProperty(string, soapAction);  
 }  
  
 public void setRequestMethod(String requestMethod) throws IOException {  
 this.connection.setRequestMethod(requestMethod);  
 }  
  
 @Override  
 public void setFixedLengthStreamingMode(int i) {  
  
 }  
  
 @Override  
 public void setChunkedStreamingMode() {  
  
 }  
  
 public OutputStream openOutputStream() throws IOException {  
 return this.connection.getOutputStream();  
 }  
  
 public InputStream openInputStream() throws IOException {  
 return this.connection.getInputStream();  
 }  
  
 public InputStream getErrorStream() {  
 return this.connection.getErrorStream();  
 }  
  
 @Override  
 public String getHost() {  
 return null;  
 }  
  
 @Override  
 public int getPort() {  
 return 0;  
 }  
  
 @Override  
 public String getPath() {  
 return null;  
 }  
  
 // 设置连接服务器的超时时间,毫秒级,此为自己添加的方法  
 public void setConnectionTimeOut(int timeout) {  
 this.connection.setConnectTimeout(timeout);  
 }  
}

package com.example.administrator.timenote.Model;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/21.  
 \*/*public class BeanDRemindInformation {  
 public static BeanDRemindInformation *defaultset*;  
 public static BeanDRemindInformation *tsset*;  
 private int dremindid;  
 private int userid;  
 private int dremindring;  
 private int dremindrepeat;  
 private int aheadtime;  
 private int eventid;  
 private int leavestime;  
 private int pid;  
 private String defaulttime;  
 private String dremindvib;// 数据库中为Boolean  
  
 public int getDremindid() {  
 return dremindid;  
 }  
  
 public void setDremindid(int dremindid) {  
 this.dremindid = dremindid;  
 }  
  
 public int getUserid() {  
 return userid;  
 }  
  
 public void setUserid(int userid) {  
 this.userid = userid;  
 }  
  
 public int getDremindring() {  
 return dremindring;  
 }  
  
 public void setDremindring(int dremindring) {  
 this.dremindring = dremindring;  
 }  
  
 public int getDremindrepeat() {  
 return dremindrepeat;  
 }  
  
 public void setDremindrepeat(int dremindrepeat) {  
 this.dremindrepeat = dremindrepeat;  
 }  
  
 public int getAheadtime() {  
 return aheadtime;  
 }  
  
 public void setAheadtime(int aheadtime) {  
 this.aheadtime = aheadtime;  
 }  
  
 public int getEventid() {  
 return eventid;  
 }  
  
 public void setEventid(int eventid) {  
 this.eventid = eventid;  
 }  
  
 public int getLeavestime() {  
 return leavestime;  
 }  
  
 public void setLeavestime(int leavestime) {  
 this.leavestime = leavestime;  
 }  
  
 public int getPid() {  
 return pid;  
 }  
  
 public void setPid(int pid) {  
 this.pid = pid;  
 }  
  
 public String getDefaulttime() {  
 return defaulttime;  
 }  
  
 public void setDefaulttime(String defaulttime) {  
 this.defaulttime = defaulttime;  
 }  
  
 public String getDremindvib() {  
 return dremindvib;  
 }  
  
 public void setDremindvib(String dremindvib) {  
 this.dremindvib = dremindvib;  
 }  
  
}

package com.example.administrator.timenote.Model;  
  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
*/\*\*  
 \* Created by Sprou on 2018/6/9.  
 \*/*public class BeanEventInformation {  
 public static List<BeanEventInformation> *allEventList* = new ArrayList<>();  
 private int userid;  
 private int eventid;  
 private int dremindid;  
 private int listid;  
 private String eventname;  
 private String eventnote;  
 private int eventpriority;  
 private Date eventdate;  
 private Date dreminddate;  
 private String leaveseventsign;// 数据库存储形式为Boolean  
 private int checkBox;  
 private String datetype; // 标题  
 private Date date\_1; // 标题  
 private int pid;  
  
 public Date getDate\_1() {  
 return date\_1;  
 }  
  
 public void setDate\_1(Date date\_1) {  
 this.date\_1 = date\_1;  
 }  
  
 public int getPid() {  
 return pid;  
 }  
  
 public void setPid(int pid) {  
 this.pid = pid;  
 }  
  
 public String getDatetype() {  
 return datetype;  
 }  
  
 public void setDatetype(String datetype) {  
 this.datetype = datetype;  
 }  
  
 public int getCheckBox() {  
 return checkBox;  
 }  
  
 public void setCheckBox(int cheackBox) {  
 this.checkBox = cheackBox;  
 }  
  
 public int getUserid() {  
 return userid;  
 }  
  
 public void setUserid(int userid) {  
 this.userid = userid;  
 }  
  
 public int getEventid() {  
 return eventid;  
 }  
  
 public void setEventid(int eventid) {  
 this.eventid = eventid;  
 }  
  
 public int getDremindid() {  
 return dremindid;  
 }  
  
 public void setDremindid(int dreminid) {  
 this.dremindid = dreminid;  
 }  
  
 public int getListid() {  
 return listid;  
 }  
  
 public void setListid(int listid) {  
 this.listid = listid;  
 }  
  
 public String getEventname() {  
 return eventname;  
 }  
  
 public void setEventname(String eventname) {  
 this.eventname = eventname;  
 }  
  
 public String getEventnote() {  
 return eventnote;  
 }  
  
 public void setEventnote(String eventnote) {  
 this.eventnote = eventnote;  
 }  
  
 public int getEventpriority() {  
 return eventpriority;  
 }  
  
 public void setEventpriority(int eventpriority) {  
 this.eventpriority = eventpriority;  
 }  
  
 public Date getEventdate() {  
 return eventdate;  
 }  
  
 public void setEventdate(Date eventdate) {  
 this.eventdate = eventdate;  
 }  
  
 public Date getDreminddate() {  
 return dreminddate;  
 }  
  
 public void setDreminddate(Date dreminddate) {  
 this.dreminddate = dreminddate;  
 }  
  
 public String getLeaveseventsign() {  
 return leaveseventsign;  
 }  
  
 public void setLeaveseventsign(String leaveseventsign) {  
 this.leaveseventsign = leaveseventsign;  
 }  
  
 public BeanEventInformation(){  
  
 }  
 public BeanEventInformation(String datetype){  
 this.setDatetype(datetype);  
 }  
  
}

package com.example.administrator.timenote.Model;  
  
  
import java.util.Date;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/22.  
 \*/*public class BeanLeavesStatistics {  
 public static BeanLeavesStatistics *leavesStatistics*;  
 public static BeanLeavesStatistics *todayleaves*;  
 private Date leavesdate;  
 private int leavesamount;  
 private int focustime;  
 private int userid;  
  
 public Date getLeavesdate() {  
 return leavesdate;  
 }  
  
 public void setLeavesdate(Date leavesdate) {  
 this.leavesdate = leavesdate;  
 }  
  
 public int getLeavesamount() {  
 return leavesamount;  
 }  
  
 public void setLeavesamount(int leavesamount) {  
 this.leavesamount = leavesamount;  
 }  
  
 public int getFocustime() {  
 return focustime;  
 }  
  
 public void setFocustime(int focustime) {  
 this.focustime = focustime;  
 }  
  
 public int getUserid() {  
 return userid;  
 }  
  
 public void setUserid(int userid) {  
 this.userid = userid;  
 }  
  
}

package com.example.administrator.timenote.Model;  
  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/17.  
 \*/*public class BeanListInformation {  
 public static List<BeanListInformation> *allList* = new ArrayList<>();  
 private int userid;  
 private int listid;  
 private String listname;  
 private Date createdate;  
 private String backgroundcolor;  
 private int listpriority;  
  
 public int getUserid() {  
 return userid;  
 }  
  
 public void setUserid(int userid) {  
 this.userid = userid;  
 }  
  
 public int getListid() {  
 return listid;  
 }  
  
 public void setListid(int listid) {  
 this.listid = listid;  
 }  
  
 public String getListname() {  
 return listname;  
 }  
  
 public void setListname(String listname) {  
 this.listname = listname;  
 }  
  
 public Date getCreatedate() {  
 return createdate;  
 }  
  
 public void setCreatedate(Date createdate) {  
 this.createdate = createdate;  
 }  
  
 public String getBackgroundcolor() {  
 return backgroundcolor;  
 }  
  
 public void setBackgroundcolor(String backgroundcolor) {  
 this.backgroundcolor = backgroundcolor;  
 }  
  
 public int getListpriority() {  
 return listpriority;  
 }  
  
 public void setListpriority(int listpriority) {  
 this.listpriority = listpriority;  
 }  
  
  
}

package com.example.administrator.timenote.Model;  
  
import java.util.Date;  
  
*/\*\*  
 \* Created by Sprou on 2018/5/27.  
 \*/*public class BeanUserInformation {  
 public static BeanUserInformation *currentLoginUser* = null;  
 public static BeanUserInformation *tryLoginUser* = null;  
 private int userid;  
 private String username;  
 private String userpassword;  
 private String useremail;  
 private int leavesid;  
 private String creatdate;  
 private String stopdate;  
 private String usercalendar; // 数据库存储形式为Boolean  
 private String usertypeface;  
 private int achievement;  
 private String authcode;  
 private byte[] icon;// 头像  
  
 public byte[] getIcon() {  
 return icon;  
 }  
  
 public void setIcon(byte[] icon) {  
 this.icon = icon;  
 }  
  
  
  
 public int getUserid() {  
 return userid;  
 }  
  
 public void setUserid(int userid) {  
 this.userid = userid;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getUserpassword() {  
 return userpassword;  
 }  
  
 public void setUserpassword(String userpassword) {  
 this.userpassword = userpassword;  
 }  
  
 public String getUseremail() {  
 return useremail;  
 }  
  
 public void setUseremail(String useremail) {  
 this.useremail = useremail;  
 }  
  
 public int getLeavesid() {  
 return leavesid;  
 }  
  
 public void setLeavesid(int leavesid) {  
 this.leavesid = leavesid;  
 }  
  
 public String getCreatdate() {  
 return creatdate;  
 }  
  
 public void setCreatdate(String creatdate) {  
 this.creatdate = creatdate;  
 }  
  
 public String getStopdate() {  
 return stopdate;  
 }  
  
 public void setStopdate(String stopdate) {  
 this.stopdate = stopdate;  
 }  
  
 public String getUsercalendar() {  
 return usercalendar;  
 }  
  
 public void setUsercalendar(String usercalendar) {  
 this.usercalendar = usercalendar;  
 }  
  
 public String getUsertypeface() {  
 return usertypeface;  
 }  
  
 public void setUsertypeface(String usertypeface) {  
 this.usertypeface = usertypeface;  
 }  
  
 public int getAchievement() {  
 return achievement;  
 }  
  
 public void setAchievement(int achievement) {  
 this.achievement = achievement;  
 }  
  
 public String getAuthcode() {  
 return authcode;  
 }  
  
 public void setAuthcode(String authcode) {  
 this.authcode = authcode;  
 }  
}

package com.example.administrator.timenote.Model;  
  
import android.graphics.Bitmap;  
import android.widget.Button;  
import android.widget.ImageView;  
  
public class List\_menu {  
 private Bitmap imageView;  
 private String list\_menu\_name;  
 public List\_menu(){};  
  
 public List\_menu(String list\_menu\_name){  
 this.list\_menu\_name = list\_menu\_name;  
 }  
  
 public List\_menu(Bitmap imageView,String list\_menu\_name){  
 this.imageView = imageView;  
 this.list\_menu\_name = list\_menu\_name;  
 }  
 public Bitmap getImageView() {  
 return imageView;  
 }  
  
 public void setImageView(Bitmap imageView) {  
 this.imageView = imageView;  
 }  
  
 public String getList\_menu\_name() {  
 return list\_menu\_name;  
 }  
  
 public void setList\_menu\_name(String list\_menu\_name) {  
 this.list\_menu\_name = list\_menu\_name;  
 }  
  
  
  
}

package com.example.administrator.timenote.Model;  
  
import android.widget.CheckBox;  
  
import java.util.Date;  
  
public class task {  
 private Date date\_1;  
 private String datetype;  
 private String taskdes;  
 private String taskname;  
 private String taskp;  
 private int checkBox;  
 private int eventid;  
 private int imageId;  
 private int ViewId;  
 public int getEventid() {  
 return eventid;  
 }  
  
 public void setEventid(int eventid) {  
 this.eventid = eventid;  
 }  
 public int getImageId() {  
 return imageId;  
 }  
 public int getCheckBox() {  
 return checkBox;  
 }  
  
 public void setCheckBox(int checkBox) {  
 this.checkBox = checkBox;  
 }  
 public int getViewId() {  
 return ViewId;  
 }  
  
  
  
 public String getTaskdes() {  
 return taskdes;  
 }  
  
 public String getTaskp() {  
 return taskp;  
 }  
 public task(Date date\_1){  
 this.date\_1=date\_1;  
 }  
  
 public task(String datetype){  
 this.setDatetype(datetype);  
 }  
  
  
 public task(String taskname, Date date\_1,String taskdes,int imageId,int viewId){  
 this.date\_1=date\_1;  
 this.taskname=taskname;  
 this.taskdes=taskdes;  
 this.imageId=imageId;  
 this.ViewId=viewId;  
 }  
 public task(String taskname, Date date\_1,String taskdes,int imageId,int viewId,int checkBox,int evevtid){  
 this.date\_1=date\_1;  
 this.taskname=taskname;  
 this.taskdes=taskdes;  
 this.imageId=imageId;  
 this.ViewId=viewId;  
 this.checkBox = checkBox;  
 this.eventid = evevtid;  
 }  
 public String getTaskname() {  
 return taskname;  
 }  
  
  
 public Date getDate\_1() {  
 return date\_1;  
 }  
  
  
 public String getDatetype() {  
 return datetype;  
 }  
  
 public void setDatetype(String datetype) {  
 this.datetype = datetype;  
 }  
}

package com.example.administrator.timenote.Ring;  
  
 import android.content.Context;  
 import android.content.Intent;  
 import android.os.Bundle;  
 import android.widget.RemoteViews;  
 import android.widget.RemoteViewsService;  
  
 import com.example.administrator.timenote.Manager.TaskManager.LoadAllEvent;  
 import com.example.administrator.timenote.Manager.UserManager.UserLogin;  
 import com.example.administrator.timenote.Manager.UserManager.loginServe;  
 import com.example.administrator.timenote.Model.BeanEventInformation;  
 import com.example.administrator.timenote.Model.BeanUserInformation;  
 import com.example.administrator.timenote.R;  
 import com.example.administrator.timenote.Ui.MainActivity;  
 import com.example.administrator.timenote.Ui.taskAdapter;  
  
 import java.util.ArrayList;  
 import java.util.Date;  
 import java.util.List;  
 import java.util.Map;  
  
 import static com.example.administrator.timenote.Model.BeanEventInformation.*allEventList*;  
  
public class ListViewService extends RemoteViewsService {  
 public static final String *INITENT\_DATA* = "extra\_data";  
 List<String> todayEvent = new ArrayList<>();  
  
 @Override  
 public RemoteViewsFactory onGetViewFactory(Intent intent) {  
 return new ListRemoteViewsFactory(this.getApplicationContext(), intent);  
 }  
  
 private class ListRemoteViewsFactory implements RemoteViewsFactory {  
  
 private Context mContext;  
  
 private List<String> mList = new ArrayList<>();  
  
 public ListRemoteViewsFactory(Context context, Intent intent) {  
 mContext = context;  
 }  
  
 @Override  
 public void onCreate() {  
// mList.add("一");  
// mList.add("二");  
// mList.add("三");  
// mList.add("四");  
// mList.add("五");  
// mList.add("六");  
 final Map<String, String> map = new loginServe().*getSaveUserInfo*(ListViewService.this);  
 if (map != null){  
 Thread r = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UserLogin userLogin = new UserLogin();  
 userLogin.getRemoteInfo(map.get("useremail"));  
 }  
 });  
 r.start();  
 try {  
 r.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* LoadAllEvent loadAllEvent = new LoadAllEvent();  
 loadAllEvent.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid());  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 long millis;  
 for (int i = 0; i < *allEventList*.size(); i++){  
 millis = *allEventList*.get(i).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if ((millis >= 0) && (millis <= (1000 \* 60 \* 60 \* 24)) && (!*allEventList*.get(i).getEventdate().toString().equals("Sat Jan 01 00:00:00 GMT+08:00 1")))  
 todayEvent.add(*allEventList*.get(i).getEventname());  
 }  
 // System.out.print(1);  
 }  
 }  
  
 @Override  
 public void onDataSetChanged() {  
  
 }  
  
 @Override  
 public void onDestroy() {  
 todayEvent.clear();  
 }  
  
 @Override  
 public int getCount() {  
 return todayEvent.size();  
 }  
  
 @Override  
 public RemoteViews getViewAt(int position) {  
 RemoteViews views = new RemoteViews(mContext.getPackageName(), R.layout.*my\_widget\_layout\_item*);  
 if (todayEvent == null){  
 views.setTextViewText(R.id.*task\_1*, "今日无任务");  
 }  
 else {  
 views.setTextViewText(R.id.*task\_1*, todayEvent.get(position));  
 }  
 Bundle extras = new Bundle();  
 extras.putInt(ListViewService.*INITENT\_DATA*, position);  
 Intent changeIntent = new Intent();  
 changeIntent.setAction(MulAppWidgetProvider.*CHANGE\_IMAGE*);  
 changeIntent.putExtras(extras);  
 /\* android.R.layout.simple\_list\_item\_1 --- id --- text1  
 \* listview的item click：将 changeIntent 发送，  
 \* changeIntent 它默认的就有action 是provider中使用 setPendingIntentTemplate 设置的action\*/  
  
 views.setOnClickFillInIntent(R.id.*task\_1*, changeIntent);  
 return views;  
 }  
  
 /\* 在更新界面的时候如果耗时就会显示 正在加载... 的默认字样，但是你可以更改这个界面  
 \* 如果返回null 显示默认界面  
 \* 否则 加载自定义的，返回RemoteViews  
 \*/  
 @Override  
 public RemoteViews getLoadingView() {  
 return null;  
 }  
  
 @Override  
 public int getViewTypeCount() {  
 return 1;  
 }  
  
 @Override  
 public long getItemId(int position) {  
 return position;  
 }  
  
 @Override  
 public boolean hasStableIds() {  
 return false;  
 }  
 }  
  
}

package com.example.administrator.timenote.Ring;  
  
import android.app.PendingIntent;  
import android.appwidget.AppWidgetManager;  
import android.appwidget.AppWidgetProvider;  
import android.content.ComponentName;  
import android.content.Context;  
import android.content.Intent;  
import android.os.Bundle;  
import android.text.TextUtils;  
import android.widget.RemoteViews;  
  
import com.example.administrator.timenote.R;  
import com.example.administrator.timenote.Ui.MainActivity;  
  
public class MulAppWidgetProvider extends AppWidgetProvider {  
  
 public static final String *CHANGE\_IMAGE* = "com.example.joy.action.CHANGE\_IMAGE";  
  
 private RemoteViews mRemoteViews;  
 private ComponentName mComponentName;  
  
  
  
  
 @Override  
 public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {  
 mRemoteViews = new RemoteViews(context.getPackageName(), R.layout.*mul\_app\_widget\_provider*);  
  
 mRemoteViews.setTextViewText(R.id.*refresh*, "进入TimeNote查看");  
 Intent skipIntent = new Intent(context, MulAppWidgetProvider.class);  
 skipIntent.setAction("refresh");  
 PendingIntent pi = PendingIntent.*getActivity*(context, 200, skipIntent, PendingIntent.*FLAG\_CANCEL\_CURRENT*);  
 mRemoteViews.setOnClickPendingIntent(R.id.*refresh*, pi);  
  
 // 更新Wdiget  
  
 // 设置 ListView 的adapter。  
 // (01) intent: 对应启动 ListViewService(RemoteViewsService) 的intent  
 // (02) setRemoteAdapter: 设置 ListView 的适配器  
 // 通过setRemoteAdapter将 ListView 和ListViewService关联起来，  
 // 以达到通过 GridWidgetService 更新 gridview 的目的  
 Intent lvIntent = new Intent(context, ListViewService.class);  
 mRemoteViews.setRemoteAdapter(R.id.*lv\_test*, lvIntent);  
 mRemoteViews.setEmptyView(R.id.*lv\_test*,android.R.id.*empty*);  
  
 // 设置响应 ListView 的intent模板  
 // 说明：“集合控件(如GridView、ListView、StackView等)”中包含很多子元素，如GridView包含很多格子。  
 // 它们不能像普通的按钮一样通过 setOnClickPendingIntent 设置点击事件，必须先通过两步。  
 // (01) 通过 setPendingIntentTemplate 设置 “intent模板”，这是比不可少的！  
 // (02) 然后在处理该“集合控件”的RemoteViewsFactory类的getViewAt()接口中 通过 setOnClickFillInIntent 设置“集合控件的某一项的数据”  
  
 /\*  
 \* setPendingIntentTemplate 设置pendingIntent 模板  
 \* setOnClickFillInIntent 可以将fillInIntent 添加到pendingIntent中  
 \*/  
 Intent toIntent = new Intent(*CHANGE\_IMAGE*);  
 PendingIntent pendingIntent = PendingIntent.*getBroadcast*(context, 200, toIntent, PendingIntent.*FLAG\_UPDATE\_CURRENT*);  
 mRemoteViews.setPendingIntentTemplate(R.id.*lv\_test*, pendingIntent);  
  
  
  
 mComponentName = new ComponentName(context, MulAppWidgetProvider.class);  
 appWidgetManager.updateAppWidget(mComponentName, mRemoteViews);  
 }  
  
 @Override  
 public void onReceive(Context context, Intent intent) {  
 super.onReceive(context, intent);  
 if(TextUtils.*equals*(*CHANGE\_IMAGE*,intent.getAction())){  
 Bundle extras = intent.getExtras();  
 int position = extras.getInt(ListViewService.*INITENT\_DATA*);  
 Intent intent1 =new Intent();  
 intent1.setComponent(new ComponentName("com.example.administrator.timenote","com.example.administrator.timenote.Ui.MainActivity"));  
 intent1.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  
 context.startActivity(intent1);  
 mRemoteViews = new RemoteViews(context.getPackageName(), R.layout.*mul\_app\_widget\_provider*);  
 mComponentName = new ComponentName(context, MulAppWidgetProvider.class);  
 AppWidgetManager.*getInstance*(context).updateAppWidget(mComponentName, mRemoteViews);  
 }  
 }  
}

package com.example.administrator.timenote.Ring;  
  
import android.annotation.SuppressLint;  
import android.app.AlarmManager;  
import android.app.Notification;  
import android.app.NotificationManager;  
import android.app.PendingIntent;  
import android.app.Service;  
import android.content.Context;  
import android.content.Intent;  
import android.media.MediaPlayer;  
import android.media.Ringtone;  
import android.media.RingtoneManager;  
import android.net.Uri;  
import android.os.Build;  
import android.os.Bundle;  
import android.os.CountDownTimer;  
import android.os.PowerManager;  
import android.os.Vibrator;  
import android.support.annotation.RequiresApi;  
import android.support.v4.app.NotificationCompat;  
import android.support.v7.app.AppCompatActivity;  
import android.view.View;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.Button;  
import android.widget.TextView;  
  
import com.example.administrator.timenote.Manager.TaskManager.Complete;  
import com.example.administrator.timenote.Model.BeanDRemindInformation;  
import com.example.administrator.timenote.R;  
import com.example.administrator.timenote.Ui.MyCustomAdapter;  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.UUID;  
  
import static com.example.administrator.timenote.Model.BeanEventInformation.*allEventList*;  
  
public class RingActivity extends AppCompatActivity {  
  
 private MediaPlayer mediaPlayer;  
  
 private Vibrator mVibrator; //声明一个振动器对象  
  
 private Ringtone rt;  
 private TextView eventname\_text;  
 private String eventname;  
 private TextView eventdate\_text;  
 private String eventdate;  
 private Button complete;  
 private Button putoff;  
 private int eventid;  
 private int ahead;  
 private int repeat;  
 private Calendar calendar;  
 private Boolean vib;  
 private int ring;  
  
 PowerManager.WakeLock mWakelock;  
 @RequiresApi(api = Build.VERSION\_CODES.*KITKAT*)  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
  
 setContentView(R.layout.*activity\_ring*);  
  
 eventname\_text = findViewById(R.id.*task\_name*);  
 eventdate\_text = findViewById(R.id.*task\_time*);  
 complete = findViewById(R.id.*complete\_2*);  
 putoff = findViewById(R.id.*putoff*);  
  
 eventname = getIntent().getStringExtra("eventname");  
 eventname\_text.setText(eventname);  
  
 eventdate = getIntent().getStringExtra("eventdate");  
 eventdate\_text.setText(eventdate);  
  
 eventid = getIntent().getIntExtra("eventid", 0);  
 calendar = Calendar.*getInstance*();  
  
 ahead = getIntent().getIntExtra("ahead", 0);  
 repeat = getIntent().getIntExtra("repeat", 0);  
  
 Date date = null;  
 SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm");  
 try {  
 date = formatter.parse(eventdate);  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
  
 Calendar calendar2 = Calendar.*getInstance*();  
 calendar2.setTime(date);  
  
// long mils = 0;  
// switch (ahead){  
// case 0:break;  
// case 1:mils = 1000\*60\*5;break;  
// case 2:mils = 1000\*60\*30;break;  
// case 3:mils = 1000\*60\*60;break;  
// case 4:mils = 1000\*60\*60\*24;break;  
// }  
  
  
 if (repeat == 0){  
 init();  
 }else {  
 int id = UUID.*randomUUID*().hashCode();  
 Calendar calendar1 = Calendar.*getInstance*();  
 Intent intent = new Intent();  
 intent.setAction("com.liuqian.android\_27alarm.RING");  
 intent.putExtra("id", id);  
 intent.putExtra("ring", ring);  
 intent.putExtra("vib", vib);  
 intent.putExtra("ahead", ahead);  
 intent.putExtra("repeat", repeat);  
 intent.putExtra("eventid", eventid);  
 intent.putExtra("eventname", eventname);  
 intent.putExtra("eventdate", eventdate);  
 PendingIntent pendingIntent = PendingIntent.*getBroadcast*(RingActivity.this, id, intent, 0);  
 //闹钟管理者 参数： 唤醒屏幕，  
 AlarmManager alarmManager = (AlarmManager) getSystemService(Context.*ALARM\_SERVICE*);  
 alarmManager.setExact(AlarmManager.*RTC\_WAKEUP*, calendar2.getTimeInMillis() + 1000 \* 60 \* 60 \* 24, pendingIntent);  
  
 switch (repeat) {  
 case 1:  
 init();  
 break;  
 case 2:  
 if (calendar.get(Calendar.*DAY\_OF\_WEEK*) !=1 && calendar.get(Calendar.*DAY\_OF\_WEEK*) !=7)  
 init();  
 else  
 finish();  
 break;  
 case 3:  
 if (calendar.get(Calendar.*DAY\_OF\_WEEK*) ==1 || calendar.get(Calendar.*DAY\_OF\_WEEK*) ==7)  
 init();  
 else  
 finish();  
 break;  
 case 4:  
 if (calendar.get(Calendar.*DAY\_OF\_MONTH*) == calendar2.get(Calendar.*DAY\_OF\_MONTH*))  
 init();  
 else  
 finish();  
 break;  
 case 5:  
 if (calendar.get(Calendar.*DAY\_OF\_YEAR*) == calendar2.get(Calendar.*DAY\_OF\_YEAR*))  
 init();  
 else  
 finish();  
 break;  
 default:finish();  
 break;  
  
 }  
 }  
  
  
 }  
  
 private void init(){  
 //唤醒屏幕  
 final Window win = getWindow();  
 win.addFlags(WindowManager.LayoutParams.*FLAG\_SHOW\_WHEN\_LOCKED* | WindowManager.LayoutParams.*FLAG\_DISMISS\_KEYGUARD*);  
 win.addFlags(WindowManager.LayoutParams.*FLAG\_KEEP\_SCREEN\_ON* | WindowManager.LayoutParams.*FLAG\_TURN\_SCREEN\_ON*);  
  
 cdt.start();  
  
 // 铃声设置  
 ring = getIntent().getIntExtra("ring",0);  
 switch (ring){  
 case 0:break;  
 case 1:  
 mediaPlayer = MediaPlayer.*create*(this, R.raw.*one*);  
 mediaPlayer.setLooping(true);  
 mediaPlayer.start();  
 break;  
 case 2:  
 mediaPlayer = MediaPlayer.*create*(this, R.raw.*two*);  
 mediaPlayer.setLooping(true);  
 mediaPlayer.start();  
 break;  
 case 3:  
 mediaPlayer = MediaPlayer.*create*(this, R.raw.*three*);  
 mediaPlayer.setLooping(true);  
 mediaPlayer.start();  
 break;  
 case 4:  
 mediaPlayer = MediaPlayer.*create*(this, R.raw.*four*);  
 mediaPlayer.setLooping(true);  
 mediaPlayer.start();  
 break;  
 case 5:  
 mediaPlayer = MediaPlayer.*create*(this, R.raw.*five*);  
 mediaPlayer.setLooping(true);  
 mediaPlayer.start();  
 break;  
 case 6:  
 mediaPlayer = MediaPlayer.*create*(this, R.raw.*six*);  
 mediaPlayer.setLooping(true);  
 mediaPlayer.start();  
 break;  
 default:break;  
 }  
  
 // 振动设置  
 vib = getIntent().getBooleanExtra("vib",false);  
 mVibrator = (Vibrator) getApplication().getSystemService(Service.*VIBRATOR\_SERVICE*);  
 if (vib) {  
 mVibrator.vibrate(new long[]{1000, 1000, 1000, 1000}, 0);  
 }  
  
 //唤醒屏幕  
 PowerManager pm = (PowerManager)getSystemService(Context.*POWER\_SERVICE*);  
 mWakelock = pm.newWakeLock(PowerManager.*ACQUIRE\_CAUSES\_WAKEUP* | PowerManager.*SCREEN\_DIM\_WAKE\_LOCK*, "SimpleTimer");  
 mWakelock.acquire();  
  
 // 完成点击监听  
 complete.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* Complete complete = new Complete();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 complete.getRemoteInfo(eventid);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 stop(view);  
 }  
 });  
  
 // 推迟点击监听  
 putoff.setOnClickListener(new View.OnClickListener() {  
 @RequiresApi(api = Build.VERSION\_CODES.*KITKAT*)  
 @Override  
 public void onClick(View view) {  
 int id = UUID.*randomUUID*().hashCode();;  
 Calendar calendar1= Calendar.*getInstance*();  
 Intent intent=new Intent();  
 intent.setAction("com.liuqian.android\_27alarm.RING");  
 intent.putExtra("id",id);  
 intent.putExtra("ring", ring);  
 intent.putExtra("vib", vib);  
 intent.putExtra("ahead", ahead);  
 intent.putExtra("repeat", repeat);  
 intent.putExtra("eventid", eventid);  
 intent.putExtra("eventname", eventname);  
 intent.putExtra("eventdate", eventdate);  
 PendingIntent pendingIntent= PendingIntent.*getBroadcast*(RingActivity.this,id,intent,0);  
 //闹钟管理者 参数： 唤醒屏幕，  
 AlarmManager alarmManager =(AlarmManager) getSystemService(Context.*ALARM\_SERVICE*);  
 alarmManager.setExact(AlarmManager.*RTC\_WAKEUP*, calendar1.getTimeInMillis() + 1000 \* 60 \* 5, pendingIntent);  
 stop(view);  
 }  
 });  
 }  
  
  
 // 停止点击事件  
 @SuppressLint("NewApi")  
 public void stop(View view){  
  
 if (mediaPlayer!=null){  
 if (mediaPlayer.isPlaying()){  
 mediaPlayer.stop();  
 mediaPlayer.release();  
 }  
 }  
 if (mVibrator.hasVibrator())  
 {  
 mVibrator.cancel();  
 }  
 mWakelock.release();  
  
// //设置下一次提醒  
// int id = getIntent().getIntExtra("id",0);  
//  
// Calendar calendar1= Calendar.getInstance();  
// Intent intent=new Intent();  
// intent.setAction("com.liuqian.android\_27alarm.RING");  
// intent.putExtra("id",id);  
// PendingIntent pendingIntent= PendingIntent.getBroadcast(RingActivity.this,id,intent,0);  
// //闹钟管理者 参数： 唤醒屏幕，  
// AlarmManager alarmManager =(AlarmManager) getSystemService(Context.ALARM\_SERVICE);  
// alarmManager.setExact(AlarmManager.RTC\_WAKEUP,calendar1.getTimeInMillis(),pendingIntent);  
 cdt.cancel();  
 finish();  
  
 }  
  
 CountDownTimer cdt = new CountDownTimer(60000, 5000) {  
  
 @SuppressLint({"ResourceAsColor", "SetTextI18n"})  
 public void onTick(long millisUntilFinished) {  
 }  
 @SuppressLint("ResourceAsColor")  
 @Override  
 public void onFinish() {  
 //实例化通知管理器  
 NotificationManager notificationManager= (NotificationManager) getSystemService(Context.*NOTIFICATION\_SERVICE*);  
 //实例化通知  
 NotificationCompat.Builder builder=new NotificationCompat.Builder(RingActivity.this);  
 //设置值  
 builder.setContentTitle("错过闹钟");  
 builder.setContentText(eventname);  
 builder.setDefaults(NotificationCompat.*DEFAULT\_ALL*);  
 builder.setAutoCancel(true);  
 builder.setSmallIcon(android.R.drawable.*ic\_media\_rew*);  
  
 //发送通知  
 Notification notification=builder.build();  
 notificationManager.notify(0x101,notification);  
 if (mediaPlayer!=null){  
 if (mediaPlayer.isPlaying()){  
 mediaPlayer.stop();  
 mediaPlayer.release();  
 }  
 }  
 if (mVibrator.hasVibrator())  
 {  
 mVibrator.cancel();  
 }  
 mWakelock.release();  
 finish();  
 }  
 };  
  
}

package com.example.administrator.timenote.Ring;  
  
import android.content.BroadcastReceiver;  
import android.content.Context;  
import android.content.Intent;  
  
import java.util.Calendar;  
import java.util.Date;  
  
*/\*\*  
 \* Created by Administrator on 2017/02/14.  
 \*/*public class RingReceived extends BroadcastReceiver {  
  
  
 public void onReceive(Context context, Intent intent) {  
 int id = intent.getIntExtra("id", 1);  
 int ring = intent.getIntExtra("ring", 0);  
 Boolean vib = intent.getBooleanExtra("vib", false);  
 int ahead = intent.getIntExtra("ahead" , 0);  
 int repeat = intent.getIntExtra("repeat", 0);  
 int eventid = intent.getIntExtra("eventid", 0);  
 String eventname = intent.getStringExtra("eventname");  
 String eventdate = intent.getStringExtra("eventdate");  
  
 System.*out*.print(id);  
 //跳转到activity 播放闹钟  
 Intent intent1=new Intent(context,RingActivity.class);  
 //给INtentm设置一个标志位  
 intent1.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  
 intent1.putExtra("id", id);  
 intent1.putExtra("ring", ring);  
 intent1.putExtra("vib", vib);  
 intent1.putExtra("ahead", ahead);  
 intent1.putExtra("repeat", repeat);  
 intent1.putExtra("eventid", eventid);  
 intent1.putExtra("eventname", eventname);  
 intent1.putExtra("eventdate", eventdate);  
 context.startActivity(intent1);  
 Calendar calendar1= Calendar.*getInstance*();  
 }  
}

package com.example.administrator.timenote.Threadcontroal;  
  
import android.os.Handler;  
import android.os.Looper;  
  
*/\*\*  
 \* Created by Sprou on 2018/5/31.  
 \*/*public class MyHandler extends Handler {  
 public MyHandler() {  
 }  
  
 public MyHandler(Looper L) {  
 super(L);  
 }

package com.example.administrator.timenote.Ui.Calendar\_mode;  
  
import com.prolificinteractive.materialcalendarview.CalendarDay;  
import com.prolificinteractive.materialcalendarview.DayViewDecorator;  
import com.prolificinteractive.materialcalendarview.DayViewFacade;  
import com.prolificinteractive.materialcalendarview.spans.DotSpan;  
  
import java.util.Collection;  
import java.util.HashSet;  
  
*/\*\*  
 \* Decorate several days with a dot  
 \*/*public class EventDecorator implements DayViewDecorator {  
  
 private int color;  
 private HashSet<CalendarDay> dates;  
  
 public EventDecorator(int color, Collection<CalendarDay> dates) {  
 this.color = color;  
 this.dates = new HashSet<>(dates);  
 }  
  
 @Override  
 public boolean shouldDecorate(CalendarDay day) {  
 return dates.contains(day);  
 }  
  
 @Override  
 public void decorate(DayViewFacade view) {  
 view.addSpan(new DotSpan(5, color));  
 }  
}

package com.example.administrator.timenote.Ui.Calendar\_mode;  
  
import android.graphics.Color;  
import android.graphics.drawable.ColorDrawable;  
import android.graphics.drawable.Drawable;  
  
import com.prolificinteractive.materialcalendarview.CalendarDay;  
import com.prolificinteractive.materialcalendarview.DayViewDecorator;  
import com.prolificinteractive.materialcalendarview.DayViewFacade;  
  
import java.util.Calendar;  
  
*/\*\*  
 \* Highlight Saturdays and Sundays with a background  
 \*/*public class HighlightWeekendsDecorator implements DayViewDecorator {  
  
 private final Calendar calendar = Calendar.*getInstance*();  
 private final Drawable highlightDrawable;  
 private static final int *color* = Color.*parseColor*("#228BC34A");  
  
 public HighlightWeekendsDecorator() {  
 highlightDrawable = new ColorDrawable(*color*);  
 }  
  
 @Override  
 public boolean shouldDecorate(CalendarDay day) {  
 day.copyTo(calendar);  
 int weekDay = calendar.get(Calendar.*DAY\_OF\_WEEK*);  
 return weekDay == Calendar.*SATURDAY* || weekDay == Calendar.*SUNDAY*;  
 }  
  
 @Override  
 public void decorate(DayViewFacade view) {  
 view.setBackgroundDrawable(highlightDrawable);  
 }  
}

package com.example.administrator.timenote.Ui.Calendar\_mode;  
  
import android.app.Activity;  
import android.graphics.drawable.Drawable;  
  
import com.example.administrator.timenote.R;  
import com.prolificinteractive.materialcalendarview.CalendarDay;  
import com.prolificinteractive.materialcalendarview.DayViewDecorator;  
import com.prolificinteractive.materialcalendarview.DayViewFacade;  
  
*/\*\*  
 \* Use a custom selector  
 \*/*public class MySelectorDecorator implements DayViewDecorator {  
  
 private final Drawable drawable;  
  
 public MySelectorDecorator(Activity context) {  
 drawable = context.getResources().getDrawable(R.drawable.*my\_selector*);  
 }  
  
 @Override  
 public boolean shouldDecorate(CalendarDay day) {  
 return true;  
 }  
  
 @Override  
 public void decorate(DayViewFacade view) {  
 view.setSelectionDrawable(drawable);  
 }  
}

package com.example.administrator.timenote.Ui.Calendar\_mode;  
  
import android.graphics.Typeface;  
import android.text.style.RelativeSizeSpan;  
import android.text.style.StyleSpan;  
  
import com.prolificinteractive.materialcalendarview.CalendarDay;  
import com.prolificinteractive.materialcalendarview.DayViewDecorator;  
import com.prolificinteractive.materialcalendarview.DayViewFacade;  
import com.prolificinteractive.materialcalendarview.MaterialCalendarView;  
  
import java.util.Date;  
  
*/\*\*  
 \* Decorate a day by making the text big and bold  
 \*/*public class OneDayDecorator implements DayViewDecorator {  
  
 private CalendarDay date;  
  
 public OneDayDecorator() {  
 date = CalendarDay.*today*();  
 }  
  
 @Override  
 public boolean shouldDecorate(CalendarDay day) {  
 return date != null && day.equals(date);  
 }  
  
 @Override  
 public void decorate(DayViewFacade view) {  
 view.addSpan(new StyleSpan(Typeface.*BOLD*));  
 view.addSpan(new RelativeSizeSpan(1.4f));  
 }  
  
 */\*\*  
 \* We're changing the internals, so make sure to call {****@linkplain*** *MaterialCalendarView#invalidateDecorators()}  
 \*/* public void setDate(Date date) {  
 this.date = CalendarDay.*from*(date);  
 }  
}

package com.example.administrator.timenote.Ui;  
  
  
import android.app.AlarmManager;  
import android.content.Context;  
import android.content.pm.ActivityInfo;  
import android.graphics.Rect;  
import android.graphics.drawable.Drawable;  
import android.support.v4.app.Fragment;  
import android.support.v4.view.ViewPager;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.view.Window;  
import android.widget.Button;  
import android.widget.RadioButton;  
import android.widget.RadioGroup;  
  
import com.example.administrator.timenote.Model.task;  
import com.example.administrator.timenote.R;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Calendar\_View extends AppCompatActivity {  
  
 private View view1, view2, view3, view4;//四个主界面Pageer  
 private ViewPager viewPager; //对应的viewPager  
 private Button list2;  
 private List<View> viewList;//view数组  
 private static final String *TAG* = "LogDemo";  
 private Button menu;  
  
 private List<task> taskList = new ArrayList<task>();//清单列表  
 private Button task\_1;  
 private List<Fragment> list;  
 private RadioGroup radioGroup;  
  
 public static AlarmManager *alarmManager*;  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 supportRequestWindowFeature(Window.*FEATURE\_NO\_TITLE*);  
 setContentView(R.layout.*activity\_calendar\_view*);  
 {  
 setRequestedOrientation(ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT*);  
 }  
// initWindow();//影藏最顶层的系统菜单栏  
 *alarmManager* = (AlarmManager) getSystemService(Context.*ALARM\_SERVICE*);  
 radioGroup = findViewById(R.id.*rg\_group*);  
 RadioButton yezi = findViewById(R.id.*yezi\_1*);  
 RadioButton calendar = findViewById(R.id.*calendar\_1*);  
 RadioButton task = findViewById(R.id.*task\_1*);  
 RadioButton setup = findViewById(R.id.*setup\_1*);  
 //定义底部标签图片大小和位置  
 Drawable drawable\_news = getResources().getDrawable(R.drawable.*yezi*);  
 //当这个图片被绘制时，给他绑定一个矩形 ltrb规定这个矩形  
 drawable\_news.setBounds(0, 0, 95, 95);  
 //设置图片在文字的哪个方向  
 yezi.setCompoundDrawables(null, drawable\_news, null, null);  
  
 //定义底部标签图片大小和位置  
 Drawable drawable\_live = getResources().getDrawable(R.drawable.*calendar*);  
 //当这个图片被绘制时，给他绑定一个矩形 ltrb规定这个矩形  
 drawable\_live.setBounds(0, 0, 95, 95);  
 //设置图片在文字的哪个方向  
 calendar.setCompoundDrawables(null, drawable\_live, null, null);  
  
 //定义底部标签图片大小和位置  
 Drawable drawable\_tuijian = getResources().getDrawable(R.drawable.*task*);  
 //当这个图片被绘制时，给他绑定一个矩形 ltrb规定这个矩形  
 drawable\_tuijian.setBounds(0, 0, 100, 100);  
 //设置图片在文字的哪个方向  
 task.setCompoundDrawables(null, drawable\_tuijian, null, null);  
  
 //定义底部标签图片大小和位置  
 Drawable drawable\_me = getResources().getDrawable(R.drawable.*setup*);  
 //当这个图片被绘制时，给他绑定一个矩形 ltrb规定这个矩形  
 drawable\_me.setBounds(0, 0, 100, 100);  
 //设置图片在文字的哪个方向  
 setup.setCompoundDrawables(null, drawable\_me, null, null);  
 radioGroup.check(R.id.*task\_1*);  
  
 viewPager = (ViewPager) findViewById(R.id.*viewpager\_1*);  
  
 //viewlist实现viewPager  
// view1 = inflater.inflate(R.layout.page1, null, false);  
// view2 = inflater.inflate(R.layout.page2, null, false);  
// view3 = inflater.inflate(R.layout.page3, null, false);  
// view4 =inflater.inflate(R.layout.page4, null, false);  
// viewList = new ArrayList<View>();// 将要分页显示的View装入数组中  
// viewList.add(view1);  
// viewList.add(view2);  
// viewList.add(view3);  
// viewList.add(view4);  
// PagerAdapter pagerAdapter = new MyPagerAdapter((ArrayList<View>) viewList);  
// viewPager.setAdapter(pagerAdapter);  
// viewPager.setCurrentItem(0);  
 list = new ArrayList<Fragment>();  
 list.add(new page1());  
 list.add(new page2());  
 list.add(new page3());  
 list.add(new page4());  
  
 FragAdapter adapter = new FragAdapter(getSupportFragmentManager(), list);  
 viewPager.setAdapter(adapter);  
 viewPager.setCurrentItem(0);  
  
 radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {  
 @Override  
 public void onCheckedChanged(RadioGroup radioGroup, int i) {  
 switch (i){  
 case R.id.*task\_1*:  
 //点击不同的radioGroup刷新到不同的viewpager  
 viewPager.setCurrentItem(0, false);  
 if (page1.*drawerLayout*.isDrawerOpen(page1.*navigationView*)){  
 page1.*drawerLayout*.closeDrawer(page1.*navigationView*);  
 }  
 break;  
 case R.id.*calendar\_1*:  
 viewPager.setCurrentItem(1,false);  
 break;  
 case R.id.*yezi\_1*:  
 viewPager.setCurrentItem(2,false);  
 break;  
 case R.id.*setup\_1*:  
 viewPager.setCurrentItem(3,false);  
 break;  
  
 }  
 }  
 });  
  
 viewPager.setOnPageChangeListener(new ViewPager.OnPageChangeListener() {  
  
 @Override  
 public void onPageSelected(int position) {  
 // arg0是当前选中的页面的Position  
 switch(position)  
 {  
 case 0:  
 radioGroup.check(R.id.*task\_1*);  
 break;  
 case 1:  
 radioGroup.check(R.id.*calendar\_1*);  
 break;  
 case 2:  
 radioGroup.check(R.id.*yezi\_1*);  
 break;  
 case 3:  
 radioGroup.check(R.id.*setup\_1*);  
 break;  
 }  
 }  
  
 @Override  
 public void onPageScrolled(int arg0, float arg1, int arg2) {  
 // arg0 :当前页面，及你点击滑动的页面；arg1:当前页面偏移的百分比；arg2:当前页面偏移的像素位置  
  
 }  
  
 @Override  
 public void onPageScrollStateChanged(int arg0) {  
 //arg0 ==1的时表示正在滑动，arg0==2的时表示滑动完毕了，arg0==0的时表示什么都没做。  
  
 }  
 });  
 }  
// private void initWindow() {//初始化窗口属性，让状态栏和导航栏透明  
// if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.KITKAT) {  
// getWindow().addFlags(WindowManager.LayoutParams.FLAG\_TRANSLUCENT\_STATUS);  
// getWindow().addFlags(WindowManager.LayoutParams.FLAG\_TRANSLUCENT\_NAVIGATION);  
//  
// }  
// }  
}

package com.example.administrator.timenote.Ui;  
  
import android.content.pm.ActivityInfo;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.text.method.PasswordTransformationMethod;  
import android.view.KeyEvent;  
import android.view.View;  
import android.view.Window;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.example.administrator.timenote.Manager.UserManager.ChangePassword;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
  
import static com.example.administrator.timenote.Model.BeanUserInformation.*tryLoginUser*;  
  
public class Change\_pwd extends AppCompatActivity {  
 private EditText newpwd1,newpwd2;  
 private TextView pwd1\_error,pwd2\_error;  
 private Button sure;  
 private String spwd1;  
 private String result;  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 supportRequestWindowFeature(Window.*FEATURE\_NO\_TITLE*);  
 setContentView(R.layout.*change\_pwd\_1*);  
 TextView sign\_up\_title\_1=findViewById(R.id.*sign\_up\_title\_1*);  
 sign\_up\_title\_1.setText("修改密码");  
 setRequestedOrientation(ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT*);  
 Button back1=findViewById(R.id.*back\_1*);  
  
 back1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 finish();  
 }  
 });  
  
 newpwd1=findViewById(R.id.*new\_pwd\_1*);  
 newpwd2=findViewById(R.id.*new\_pwd\_2*);  
  
 pwd1\_error=findViewById(R.id.*pwd\_1\_error*);  
 pwd2\_error=findViewById(R.id.*pwd\_2\_error*);  
  
 // 密码隐藏  
 newpwd1.setTransformationMethod(PasswordTransformationMethod  
 .*getInstance*());  
 newpwd2.setTransformationMethod(PasswordTransformationMethod  
 .*getInstance*());  
  
 // 两个密码文本框监听回车  
 newpwd2.setOnEditorActionListener(new TextView.OnEditorActionListener() {  
 public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {  
 newpwd2.clearFocus();  
 return false;  
 }  
 });  
 newpwd1.setOnEditorActionListener(new TextView.OnEditorActionListener() {  
 @Override  
 public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {  
 newpwd2.requestFocus();  
 return false;  
 }  
 });  
  
 sure=findViewById(R.id.*sure\_pwd\_1*);  
  
 sure.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 spwd1=newpwd1.getText().toString();  
 String spwd2=newpwd2.getText().toString();  
 if (spwd1.equals(BeanUserInformation.*tryLoginUser*.getUserpassword())){  
 Toast.*makeText*(Change\_pwd.this,"请勿设置与原密码相同的密码",Toast.*LENGTH\_SHORT*).show();  
 }  
 else {  
 if (spwd1.length() >= 8 && (spwd1.equals(spwd2))) {  
 //修改密码  
 Thread f = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* ChangePassword changePassword = new ChangePassword();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 result = changePassword.getRemoteInfo(spwd1);  
 runOnUiThread(new Runnable() {  
  
 @Override  
 public void run() {  
 // 更新UI  
 if (result.equals("true"))  
 Toast.*makeText*(Change\_pwd.this, "密码修改成功", Toast.*LENGTH\_SHORT*).show();  
 else  
 Toast.*makeText*(Change\_pwd.this, "密码修改失败", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 });  
  
 }  
 });  
 f.start();  
 try {  
 f.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 finish();  
 } else if (spwd1.length() < 8) {  
 pwd1\_error.setVisibility(View.*VISIBLE*);  
 } else if (spwd1.equals(spwd2) != true) {  
 pwd2\_error.setVisibility(View.*VISIBLE*);  
 } else {  
 pwd1\_error.setVisibility(View.*VISIBLE*);  
 pwd2\_error.setVisibility(View.*VISIBLE*);  
 }  
 }  
 }  
 });  
  
 newpwd1.setOnFocusChangeListener(new View.OnFocusChangeListener() {  
 @Override  
 public void onFocusChange(View view, boolean b) {  
 if(b)  
 {  
 pwd1\_error.setVisibility(View.*INVISIBLE*);  
 }  
 else  
 {  
 String spwd1=newpwd1.getText().toString();  
 if(spwd1.length()<8|spwd1.length()>20)  
 {  
 pwd1\_error.setVisibility(View.*VISIBLE*);  
 }  
 }  
 }  
 });  
  
 newpwd2.setOnFocusChangeListener(new View.OnFocusChangeListener() {  
 @Override  
 public void onFocusChange(View view, boolean b) {  
 if(b)  
 {  
 pwd2\_error.setVisibility(View.*INVISIBLE*);  
 }  
 else  
 {  
 String spwd2=newpwd2.getText().toString();  
 String spwd1=newpwd1.getText().toString();  
 if(spwd2.equals(spwd1)!=true)  
 {  
 pwd1\_error.setVisibility(View.*VISIBLE*);  
 }  
 }  
 }  
 });  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.app.Dialog;  
import android.content.Context;  
import android.os.Bundle;  
import android.text.Editable;  
import android.text.TextUtils;  
import android.text.TextWatcher;  
import android.view.View;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.Button;  
import android.widget.EditText;  
  
import com.example.administrator.timenote.Manager.ListManager.NewList;  
import com.example.administrator.timenote.R;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/27.  
 \*/*public class Delete extends Dialog {  
 */\*\*  
 \* 上下文对象 \*  
 \*/* private Button back;  
 private Context context;  
 private boolean issue = false; // 是否验证成功  
 private Button sure;// 确认按钮  
  
 public boolean getIssue() {  
 return issue;  
 }  
  
 public Delete (Context context) {  
 super(context);  
 this.context = context;  
 }  
  
 public Delete (Context context, int theme) {  
 super(context, theme);  
 this.context = context;  
 }  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 // 指定布局  
 this.setContentView(R.layout.*delete*);  
  
 /\*  
 \* 获取验证码的窗口对象及参数对象以修改对话框的布局设置, 可以直接调用getWindow(),表示获得这个Activity的Window  
 \* 对象,这样这可以以同样的方式改变这个Activity的属性.  
 \*/  
 Window dialogWindow = this.getWindow();  
  
// WindowManager m = context.getWindowManager();  
// Display d = m.getDefaultDisplay(); // 获取屏幕宽、高用  
 WindowManager.LayoutParams p = dialogWindow.getAttributes(); // 获取对话框当前的参数值  
// p.height = (int) (d.getHeight() ); // 高度设置为屏幕  
// p.width = (int) (d.getWidth() ); // 宽度设置为屏幕  
 dialogWindow.setAttributes(p);  
  
 // 根据id在布局中找到控件对象  
 back = findViewById(R.id.*back\_20*);  
 sure = findViewById(R.id.*sure\_delete*);  
  
  
 // 为按钮绑定点击事件监听器  
 back.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View ciew) {  
 dismiss();  
 }  
 });  
  
 // 确认按钮事件  
 sure.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View view) {  
 issue = true;  
  
 dismiss();  
  
 }  
 });  
  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.annotation.SuppressLint;  
import android.app.Dialog;  
import android.content.Context;  
import android.os.Bundle;  
import android.os.CountDownTimer;  
import android.view.Display;  
import android.view.View;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
  
import com.example.administrator.timenote.Manager.UserManager.EmailResend;  
import com.example.administrator.timenote.Manager.UserManager.UserActivity;  
import com.example.administrator.timenote.Manager.UserManager.UserLogin;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
  
  
  
public class Email\_sure extends Dialog {  
 */\*\*  
 \* 上下文对象 \*  
 \*/* private Button back;  
 private Context context;  
 private boolean issue=false; // 是否验证成功  
 private Button resend;// 重新发送按钮  
 int recLen=60;// 倒计时  
 private Button sure;// 确认按钮  
 public EditText Verification\_code\_1;// 验证码输入框  
 private TextView Verification\_code\_error\_1;// 验证码错误信息  
  
 public boolean getIssue() {  
 return issue;  
 }  
  
 public Email\_sure(Context context) {  
 super(context);  
 this.context = context;  
 }  
  
 public Email\_sure(Context context, int theme) {  
 super(context, theme);  
 this.context = context;  
 }  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 // 指定布局  
 this.setContentView(R.layout.*email\_load\_1*);  
 Verification\_code\_1 = (EditText) findViewById(R.id.*Verification\_code\_1*);  
 Verification\_code\_error\_1=findViewById(R.id.*Verification\_code\_error\_1*);  
 /\*  
 \* 获取验证码的窗口对象及参数对象以修改对话框的布局设置, 可以直接调用getWindow(),表示获得这个Activity的Window  
 \* 对象,这样这可以以同样的方式改变这个Activity的属性.  
 \*/  
 Window dialogWindow = this.getWindow();  
  
// WindowManager m = context.getWindowManager();  
// Display d = m.getDefaultDisplay(); // 获取屏幕宽、高用  
 WindowManager.LayoutParams p = dialogWindow.getAttributes(); // 获取对话框当前的参数值  
// p.height = (int) (d.getHeight() ); // 高度设置为屏幕  
// p.width = (int) (d.getWidth() ); // 宽度设置为屏幕  
 dialogWindow.setAttributes(p);  
  
 // 根据id在布局中找到控件对象  
 back =findViewById(R.id.*back\_2*);  
 resend=findViewById(R.id.*resent\_Verification\_code\_2*);  
 sure =findViewById(R.id.*sure\_Verification\_code\_1*);  
  
 // 倒计时开始  
 cdt.start();  
  
 // 为按钮绑定点击事件监听器  
 back.setOnClickListener(new View.OnClickListener(){  
 public void onClick(View view)  
 {  
 dismiss();  
 }  
 });  
  
 // 确认按钮事件  
 sure.setOnClickListener(new View.OnClickListener(){  
 public void onClick(View view)  
 {  
 String scode = Verification\_code\_1.getText().toString();  
  
 // 查验证码  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UserLogin userLogin = new UserLogin();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 userLogin.getRemoteInfo(Sign\_UpActivity.*semail*);  
  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 if(scode.equals(BeanUserInformation.*tryLoginUser*.getAuthcode()))//验证码验证  
 {  
 issue = true;  
 dismiss();  
 }  
 else  
 {  
 Verification\_code\_error\_1.setVisibility(view.*VISIBLE*);  
 }  
 }  
 });  
  
 // 重新发送按钮  
 resend.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 //重新发送验证码  
 Thread f = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* EmailResend emailResend = new EmailResend();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 emailResend.getRemoteInfo(Sign\_UpActivity.*semail*);  
  
 }  
 });  
 f.start();  
 try {  
 recLen=60;  
 cdt.start();  
 f.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
// resend.setVisibility(View.INVISIBLE);  
// sure.setVisibility(View.VISIBLE);  
  
 }  
 });  
  
 // 验证码输入框  
 Verification\_code\_1.setOnFocusChangeListener(new View.OnFocusChangeListener() {  
 @Override  
 public void onFocusChange(View view, boolean b) {  
 if (b)  
 {  
 Verification\_code\_error\_1.setVisibility(View.*INVISIBLE*);  
 }  
 }  
 });  
  
 this.setCanceledOnTouchOutside(false);  
 this.setCancelable(true);  
 }  
  
 // 倒计时计时60秒，每1秒响应一次  
 CountDownTimer cdt = new CountDownTimer(60000, 1000) {  
  
 @SuppressLint({"ResourceAsColor", "SetTextI18n"})  
 public void onTick(long millisUntilFinished) {  
 recLen--;  
 resend.setText("重新发送"+recLen+"s");  
 resend.setClickable(false);  
 resend.setTextColor(context.getResources().getColor(R.color.*color\_ok\_2*));  
 }  
 @SuppressLint("ResourceAsColor")  
 @Override  
 public void onFinish() {  
 resend.setText("重新发送");  
 resend.setClickable(true);  
 resend.setTextColor(context.getResources().getColor(R.color.*color\_ok\_1*));  
// sure.setVisibility(View.INVISIBLE);  
// resend.setVisibility(View.VISIBLE);  
 }  
 };  
}

package com.example.administrator.timenote.Ui;  
  
import android.annotation.SuppressLint;  
import android.app.Dialog;  
import android.content.Context;  
import android.os.Bundle;  
import android.os.CountDownTimer;  
import android.view.Display;  
import android.view.View;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
  
import com.example.administrator.timenote.Manager.UserManager.EmailResend;  
import com.example.administrator.timenote.Manager.UserManager.UserActivity;  
import com.example.administrator.timenote.Manager.UserManager.UserLogin;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
  
  
  
public class Email\_sure1 extends Dialog {  
 */\*\*  
 \* 上下文对象 \*  
 \*/* private Button back;  
 private Context context;  
 private boolean issue=false; // 是否验证成功  
 private Button resend;// 重新发送按钮  
 int recLen=60;// 倒计时  
 private Button sure;// 确认按钮  
 public EditText Verification\_code\_1;// 验证码输入框  
 private TextView Verification\_code\_error\_1;// 验证码错误信息  
  
 public boolean getIssue() {  
 return issue;  
 }  
  
 public Email\_sure1(Context context) {  
 super(context);  
 this.context = context;  
 }  
  
 public Email\_sure1(Context context, int theme) {  
 super(context, theme);  
 this.context = context;  
 }  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 // 指定布局  
 this.setContentView(R.layout.*email\_load\_1*);  
 Verification\_code\_1 = (EditText) findViewById(R.id.*Verification\_code\_1*);  
 Verification\_code\_error\_1=findViewById(R.id.*Verification\_code\_error\_1*);  
 /\*  
 \* 获取验证码的窗口对象及参数对象以修改对话框的布局设置, 可以直接调用getWindow(),表示获得这个Activity的Window  
 \* 对象,这样这可以以同样的方式改变这个Activity的属性.  
 \*/  
 Window dialogWindow = this.getWindow();  
  
// WindowManager m = context.getWindowManager();  
// Display d = m.getDefaultDisplay(); // 获取屏幕宽、高用  
 WindowManager.LayoutParams p = dialogWindow.getAttributes(); // 获取对话框当前的参数值  
// p.height = (int) (d.getHeight() ); // 高度设置为屏幕  
// p.width = (int) (d.getWidth() ); // 宽度设置为屏幕  
 dialogWindow.setAttributes(p);  
  
 // 根据id在布局中找到控件对象  
 back =findViewById(R.id.*back\_2*);  
 resend=findViewById(R.id.*resent\_Verification\_code\_2*);  
 sure =findViewById(R.id.*sure\_Verification\_code\_1*);  
  
 // 倒计时开始  
 //cdt.start();  
  
 // 设为蓝色  
 resend.setTextColor(context.getResources().getColor(R.color.*color\_ok\_1*));  
  
 // 为按钮绑定点击事件监听器  
 back.setOnClickListener(new View.OnClickListener(){  
 public void onClick(View ciew)  
 {  
 dismiss();  
 }  
 });  
  
 // 确认按钮事件  
 sure.setOnClickListener(new View.OnClickListener(){  
 public void onClick(View view)  
 {  
 String scode = Verification\_code\_1.getText().toString();  
  
 // 查验证码  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UserLogin userLogin = new UserLogin();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 userLogin.getRemoteInfo(MainActivity.*suesrid*);  
  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 if(scode.equals(BeanUserInformation.*tryLoginUser*.getAuthcode()))//验证码验证  
 {  
 issue = true;  
 dismiss();  
 }  
 else  
 {  
 Verification\_code\_error\_1.setVisibility(view.*VISIBLE*);  
 }  
 }  
 });  
  
 // 重新发送按钮  
 resend.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 //重新发送验证码  
 Thread f = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* EmailResend emailResend = new EmailResend();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 emailResend.getRemoteInfo(MainActivity.*suesrid*);  
  
 }  
 });  
 f.start();  
 try {  
 recLen=60;  
 cdt.start();  
 f.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
// resend.setVisibility(View.INVISIBLE);  
// sure.setVisibility(View.VISIBLE);  
  
 }  
 });  
  
 // 验证码输入框  
 Verification\_code\_1.setOnFocusChangeListener(new View.OnFocusChangeListener() {  
 @Override  
 public void onFocusChange(View view, boolean b) {  
 if (b)  
 {  
 Verification\_code\_error\_1.setVisibility(View.*INVISIBLE*);  
 }  
 }  
 });  
  
 this.setCanceledOnTouchOutside(false);  
 this.setCancelable(true);  
 }  
  
 // 倒计时计时60秒，每1秒响应一次  
 CountDownTimer cdt = new CountDownTimer(60000, 1000) {  
  
 @SuppressLint({"ResourceAsColor", "SetTextI18n"})  
 public void onTick(long millisUntilFinished) {  
 recLen--;  
 resend.setText("重新发送"+recLen+"s");  
 resend.setClickable(false);  
 resend.setTextColor(context.getResources().getColor(R.color.*color\_ok\_2*));  
 }  
 @SuppressLint("ResourceAsColor")  
 @Override  
 public void onFinish() {  
 resend.setText("重新发送");  
 resend.setClickable(true);  
 resend.setTextColor(context.getResources().getColor(R.color.*color\_ok\_1*));  
// sure.setVisibility(View.INVISIBLE);  
// resend.setVisibility(View.VISIBLE);  
 }  
 };  
}

package com.example.administrator.timenote.Ui;  
  
import android.content.Intent;  
import android.content.pm.ActivityInfo;  
import android.os.CountDownTimer;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.ContextMenu;  
import android.view.View;  
import android.view.Window;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
  
import com.example.administrator.timenote.Manager.UserManager.EmailResend;  
import com.example.administrator.timenote.Manager.UserManager.Email\_ck;  
import com.example.administrator.timenote.Manager.UserManager.UserLogin;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
  
import static com.example.administrator.timenote.Model.BeanUserInformation.*tryLoginUser*;  
  
public class Forgot\_pwd extends AppCompatActivity {  
 private int recLen=60;  
 private EditText email3,verification\_code\_3;  
 private Button back1,send,resend2,sure\_Verification;  
 private TextView email3\_error,Verification\_error;  
 public static String *semail3*;  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 supportRequestWindowFeature(Window.*FEATURE\_NO\_TITLE*);  
 setContentView(R.layout.*forgot\_pwd\_1*);  
 TextView sign\_up\_title\_1=findViewById(R.id.*sign\_up\_title\_1*);  
 sign\_up\_title\_1.setText("忘记密码");  
 {setRequestedOrientation(ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT*);}  
 back1=findViewById(R.id.*back\_1*);  
 send=findViewById(R.id.*send\_Verification\_code\_2*);  
 resend2=findViewById(R.id.*resent\_Verification\_code\_3*);  
 sure\_Verification=findViewById(R.id.*sure\_Verification\_code\_2*);  
  
 email3=findViewById(R.id.*email\_3*);  
 verification\_code\_3=findViewById(R.id.*Verification\_code\_3*);  
  
 email3\_error=findViewById(R.id.*email\_error\_2*);  
 Verification\_error=findViewById(R.id.*Verification\_code\_error\_2*);  
  
 if (MainActivity.*suesrid* != null){  
 email3.setText(MainActivity.*suesrid*);  
 verification\_code\_3.requestFocus();  
 }  
  
 back1.setOnClickListener(new View.OnClickListener(){  
 public void onClick(View view) {  
 finish();  
 //转到登录界面  
 }  
 });  
  
 send.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 *semail3*=email3.getText().toString();  
 if(*semail3*.indexOf("@")>0)  
 {  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UserLogin userLogin = new UserLogin();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 userLogin.getRemoteInfo(*semail3*);  
 }  
 });  
 t.start();  
 try {  
 t.join(3000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 if(BeanUserInformation.*tryLoginUser* != null)  
 {  
 send.setVisibility(View.*INVISIBLE*);  
 resend2.setVisibility(View.*VISIBLE*);  
 recLen = 60;  
 cdt.start();  
 Thread f = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* EmailResend emailResend = new EmailResend();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 emailResend.getRemoteInfo(*semail3*);  
 }  
 });  
 f.start();  
 try {  
 f.join(3000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 Thread r = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UserLogin userLogin = new UserLogin();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 userLogin.getRemoteInfo(*semail3*);  
 }  
 });  
 r.start();  
 try {  
 r.join(3000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 else {  
 email3\_error.setVisibility(View.*VISIBLE*);  
 email3\_error.setText("邮箱不存在，请输入正确的邮箱");  
 }  
 }  
 else  
 {  
 email3\_error.setVisibility(View.*VISIBLE*);  
 email3\_error.setText("邮箱格式错误，请输入正确的邮箱");  
 }  
 }  
 });  
  
 sure\_Verification.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 String scode=verification\_code\_3.getText().toString();  
 if(scode.equals(BeanUserInformation.*tryLoginUser*.getAuthcode()))  
 {  
 //修改密码  
 Intent intent=new Intent(Forgot\_pwd.this,Change\_pwd.class);  
 startActivity(intent);  
 finish();  
 }  
 else  
 {  
 Verification\_error.setVisibility(View.*VISIBLE*);  
 }  
 }  
 });  
  
 email3.setOnFocusChangeListener(new View.OnFocusChangeListener() {  
 @Override  
 public void onFocusChange(View view, boolean b) {  
 if(b)  
 {  
 email3\_error.setVisibility(View.*INVISIBLE*);  
 }  
 else{  
 if(!Email\_ck.*isEmail*(email3.getText().toString())){  
 email3\_error.setVisibility(View.*VISIBLE*);  
 email3\_error.setText("邮箱格式错误，请输入正确的邮箱");  
 }  
 }  
 }  
 });  
  
 verification\_code\_3.setOnFocusChangeListener(new View.OnFocusChangeListener() {  
 @Override  
 public void onFocusChange(View view, boolean b) {  
 if(b)  
 {  
 Verification\_error.setVisibility(View.*INVISIBLE*);  
 }  
 }  
 });  
 }  
 CountDownTimer cdt = new CountDownTimer(60000, 1000) {  
  
 public void onTick(long millisUntilFinished) {  
 recLen--;  
 resend2.setText("重新发送"+recLen+"s");  
 }  
 @Override  
 public void onFinish() {  
 resend2.setVisibility(View.*INVISIBLE*);  
 send.setVisibility(View.*VISIBLE*);  
 }  
 };  
}

package com.example.administrator.timenote.Ui;  
  
import android.support.v4.app.Fragment;  
import android.support.v4.app.FragmentManager;  
import android.support.v4.app.FragmentPagerAdapter;  
  
import java.util.List;  
  
public class FragAdapter extends FragmentPagerAdapter {  
  
 private List<Fragment> list;  
  
 public FragAdapter(FragmentManager fm) {  
 super(fm);  
 }  
  
 public FragAdapter(FragmentManager fm, List<Fragment> list) {  
 super(fm);  
 this.list = list;  
 }  
  
 @Override  
 public Fragment getItem(int postion) {  
 return list.get(postion);  
 }  
  
 @Override  
 public int getCount() {  
 return list.size();  
 }  
  
}

package com.example.administrator.timenote.Ui;  
  
import android.app.Dialog;  
import android.content.Context;  
import android.os.Bundle;  
import android.os.CountDownTimer;  
import android.view.View;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
  
import com.example.administrator.timenote.R;  
  
public class Level\_select extends Dialog {  
 */\*\*  
 \* 上下文对象 \*  
 \*/* private Button back;  
 private Context context;  
 private Button level0;  
 private Button level1;  
 private Button level2;  
 private Button level3;  
  
 public static int getLevel() {  
 return *level*;  
 }  
  
 public static void setLevel(int level) {  
 Level\_select.*level* = level;  
 }  
  
 private static int *level*=-1;  
  
 public Level\_select (Context context) {  
 super(context);  
 this.context = context;  
 }  
  
 public Level\_select (Context context, int theme) {  
 super(context, theme);  
 this.context = context;  
 }  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 // 指定布局  
 this.setContentView(R.layout.*level\_select*);  
 Window dialogWindow = this.getWindow();  
  
// WindowManager m = context.getWindowManager();  
// Display d = m.getDefaultDisplay(); // 获取屏幕宽、高用  
 WindowManager.LayoutParams p = dialogWindow.getAttributes(); // 获取对话框当前的参数值  
// p.height = (int) (d.getHeight() ); // 高度设置为屏幕  
// p.width = (int) (d.getWidth() ); // 宽度设置为屏幕  
 dialogWindow.setAttributes(p);  
  
 // 根据id在布局中找到控件对象  
 back =findViewById(R.id.*back\_5*);  
 level0 =findViewById(R.id.*level0\_button*);  
 level1 =findViewById(R.id.*level1\_button*);  
 level2 =findViewById(R.id.*level2\_button*);  
 level3 =findViewById(R.id.*level3\_button*);  
  
  
 // 为按钮绑定点击事件监听器  
 back.setOnClickListener(new View.OnClickListener(){  
 public void onClick(View view)  
 {  
 dismiss();  
 }  
 });  
  
 level0.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 *level*=0;  
 dismiss();  
 }  
 });  
  
 level1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 *level*=1;  
 dismiss();  
 }  
 });  
  
 level2.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 *level*=2;  
 dismiss();  
 }  
 });  
  
 level3.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 *level*=3;  
 dismiss();  
 }  
 });  
  
 this.setCancelable(true);  
 }  
  
}

package com.example.administrator.timenote.Ui;  
  
import android.content.Context;  
import android.content.DialogInterface;  
import android.support.v7.app.AlertDialog;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ArrayAdapter;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.example.administrator.timenote.Manager.ListManager.DeleteList;  
import com.example.administrator.timenote.Manager.ListManager.LoadAllList;  
import com.example.administrator.timenote.Model.BeanListInformation;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.Model.List\_menu;  
import com.example.administrator.timenote.Model.task;  
import com.example.administrator.timenote.R;  
  
import java.text.SimpleDateFormat;  
import java.util.List;  
  
public class List\_menu\_Adapter extends ArrayAdapter<List\_menu> implements View.OnClickListener, View.OnLongClickListener {  
 private int resourceId;  
 private ListAdapter.InnerItemOnclickListener mListener;  
 private int position;  
  
 public List\_menu\_Adapter(Context context, int resource, List<List\_menu> objects) {  
 super(context, resource, objects);  
 resourceId=resource;  
 }  
  
 public View getView(int position, View convertView, ViewGroup parent){  
 List\_menu l = getItem(position);  
 ViewHolder holder = new ViewHolder();  
 View view = LayoutInflater.*from*(getContext()).inflate(resourceId,null,false);  
 holder.imageView = view.findViewById(R.id.*imageView4*);  
 holder.button = view.findViewById(R.id.*list\_menu\_name*);  
 holder.imageView.setImageBitmap(l.getImageView());  
 holder.button.setText(l.getList\_menu\_name());  
  
 holder.imageView.setOnClickListener(this);  
 holder.imageView.setTag(position);  
 holder.imageView.setOnLongClickListener(this);  
 holder.button.setOnClickListener(this);  
 holder.button.setTag(position);  
 holder.button.setOnLongClickListener(this);  
  
 return view;  
 }  
  
 @Override  
 public boolean onLongClick(View v) {  
 position = (Integer) v.getTag();  
 if (!getItem(position).getList\_menu\_name().equals("收集箱")) {  
 Toast.*makeText*(getContext(), String.*valueOf*(position), Toast.*LENGTH\_SHORT*).show();  
 final AlertDialog.Builder alterDialog = new AlertDialog.Builder(getContext());  
 alterDialog.setTitle("提示！");  
 alterDialog.setMessage("您确定要删除这个清单吗？所属任务将会加入默认列表");  
 alterDialog.setPositiveButton("确认", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 // *TODO: 2018/6/26 清单删除* Thread r = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* DeleteList deleteList = new DeleteList();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 deleteList.getRemoteInfo(BeanListInformation.*allList*.get(position).getListid());  
  
 }  
 });  
 r.start();  
 try {  
 r.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* LoadAllList loadAllList = new LoadAllList();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 loadAllList.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid());  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 page1.*list\_update*.delete\_update();  
 }  
  
 }  
 );  
 alterDialog.setNegativeButton("取消", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 dialog.dismiss();  
 }  
 });  
 alterDialog.show();  
 }  
 return true;  
  
 }  
  
 interface InnerItemOnclickListener {  
 void itemClick(View v);  
 }  
  
 public void setOnInnerItemOnClickListener(ListAdapter.InnerItemOnclickListener listener){  
 this.mListener=listener;  
 }  
  
 @Override  
 public void onClick(View v) {  
 mListener.itemClick(v);  
 }  
 static class ViewHolder{  
 Button button;  
 ImageView imageView;  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.app.Dialog;  
import android.app.DialogFragment;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.os.Bundle;  
import android.os.CountDownTimer;  
import android.view.View;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.AdapterView;  
import android.widget.Button;  
import android.widget.ListView;  
import android.widget.Toast;  
  
import com.example.administrator.timenote.Manager.ListManager.LoadAllList;  
import com.example.administrator.timenote.Manager.ListManager.NewList;  
import com.example.administrator.timenote.Manager.TaskManager.Complete;  
import com.example.administrator.timenote.Manager.TaskManager.LoadAllEvent;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
import android.widget.AdapterView.OnItemClickListener;  
import java.util.ArrayList;  
  
import static com.example.administrator.timenote.Model.BeanListInformation.*allList*;  
  
public class List\_select extends Dialog implements OnItemClickListener, ListAdapter.InnerItemOnclickListener {  
 */\*\*  
 \* 上下文对象 \*  
 \*/* private Button back;// 取消按钮  
 private Context context;  
 private ListAdapter listAdapter;// 清单列表适配器  
 private ListView listView;// 清单列表  
 private ArrayList<String> listname;// 清单名称列表  
 private New\_List new\_list;// 新建清单;  
 private static int *nlistid*; // 获得清单id  
 private static String *nlistname*="所有";// 当前选择的清单名称  
 private Boolean issure = false;  
  
 public Boolean getIssure() {  
 return issure;  
 }  
  
  
// public static int getListid() {  
// return listid;  
// }  
//  
// private static int listid;  
  
 public static int getNlistid() {  
 return *nlistid*;  
 }  
  
  
  
 public static String getNlistname() {  
 return *nlistname*;  
 }  
  
  
  
 public List\_select(Context context) {  
 super(context);  
 this.context = context;  
 }  
  
 public List\_select(Context context, int theme) {  
 super(context, theme);  
 this.context = context;  
 }  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 // 指定布局  
 this.setContentView(R.layout.*list\_select\_1*);  
  
  
 /\*  
 \* 获取验证码的窗口对象及参数对象以修改对话框的布局设置, 可以直接调用getWindow(),表示获得这个Activity的Window  
 \* 对象,这样这可以以同样的方式改变这个Activity的属性.  
 \*/  
 Window dialogWindow = this.getWindow();  
  
// WindowManager m = context.getWindowManager();  
// Display d = m.getDefaultDisplay(); // 获取屏幕宽、高用  
 WindowManager.LayoutParams p = dialogWindow.getAttributes(); // 获取对话框当前的参数值  
// p.height = (int) (d.getHeight() ); // 高度设置为屏幕  
// p.width = (int) (d.getWidth() ); // 宽度设置为屏幕  
 dialogWindow.setAttributes(p);  
  
 // 根据id在布局中找到控件对象  
 back=findViewById(R.id.*back\_4*);  
 listView = findViewById(R.id.*listView3*);  
  
 initList();  
  
 // 为按钮绑定点击事件监听器  
 back.setOnClickListener(new View.OnClickListener(){  
 public void onClick(View view)  
 {  
 dismiss();  
 }  
 });  
  
  
  
 this.setCancelable(true);  
 }  
  
  
 // 清单选择后点击的事件回调  
 public void itemClick(View v) {  
 int position;  
 position = (Integer) v.getTag();  
 switch (v.getId()) {  
 case R.id.*list\_select\_button*:  
 *nlistname* = listname.get(position);  
 if(*nlistname*.equals("添加清单"))  
 {  
 new\_list = new New\_List(getContext(),R.style.*dialog*);  
 new\_list.show();  
 new\_list.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 public void onDismiss(DialogInterface dialog) {  
 if(new\_list.getIssue())  
 {  
 *nlistname* = new\_list.getList\_name();  
  
 // *TODO 2018.6.17 添加清单nlistname* Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* NewList newList = new NewList();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 newList.getRemoteInfo(*nlistname*);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 issure = true;  
 initList();  
 dismiss();  
 Toast.*makeText*(context,*nlistname*,Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 }  
 else{  
 *nlistid* = *allList*.get(position).getListid();  
 issure = true;  
 dismiss();  
 Toast.*makeText*(context,*nlistname*,Toast.*LENGTH\_SHORT*).show();  
 }  
  
 break;  
 default:  
 break;  
 }  
 }  
 @Override  
 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
// Toast.makeText(context,"列表 "+position+" 被点击了",Toast.LENGTH\_SHORT).show();  
 }  
  
 private void initList(){  
 listname=new ArrayList<String>();  
// listname.add("收集箱");  
// for(int i=0;i<10;i++)  
// {  
// listname.add("清单");  
// }  
  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* LoadAllList loadAllList = new LoadAllList();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 loadAllList.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid());  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 for(int i = 0; i < *allList*.size(); i++){  
 listname.add(*allList*.get(i).getListname());  
 }  
 listAdapter=new ListAdapter(context,R.layout.*list\_button*,listname);  
 listname.add("添加清单");  
 *nlistid* = *allList*.get(*allList*.size()-1).getListid();  
 listAdapter.setOnInnerItemOnClickListener(this);  
 listView.setAdapter(listAdapter);  
 listView.setOnItemClickListener(this);  
 }  
  
}  
// public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
// //通过view获取其内部的组件，进而进行操作  
// String text = (String) ((TextView)view.findViewById(R.id.text)).getText();  
// //大多数情况下，position和id相同，并且都从0开始  
// String showText = "点击第" + position + "项，文本内容为：" + text + "，ID为：" + id;  
// Toast.makeText(context, showText, Toast.LENGTH\_LONG).show();  
// }

package com.example.administrator.timenote.Ui;  
  
import android.app.Dialog;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.pm.ActivityInfo;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.view.ViewGroup;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.AdapterView;  
import android.widget.Button;  
import android.widget.ListView;  
  
import com.example.administrator.timenote.Manager.ListManager.UpdateList;  
import com.example.administrator.timenote.Model.List\_menu;  
import com.example.administrator.timenote.R;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import static com.example.administrator.timenote.Model.BeanListInformation.*allList*;  
  
public class List\_Update extends Dialog implements AdapterView.OnItemClickListener, ListAdapter.InnerItemOnclickListener{  
  
 private Button back;// 返回按钮  
 private Context context;  
 private ListView listView;// 清单列表  
 private List<List\_menu> list;  
 private List\_update\_dialog list\_update\_dialog;  
 private static int *position*;  
  
 public List\_Update(){  
 super(null);  
 }  
  
 public static int getPosition() {  
 return *position*;  
 }  
 public List\_Update(Context context) {  
 super(context);  
 this.context = context;  
 }  
  
 public List\_Update(Context context, int theme) {  
 super(context, theme);  
 this.context = context;  
 }  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 setContentView(R.layout.*activity\_list\_update*);  
 Window dialogWindow = this.getWindow();  
  
// WindowManager m = context.getWindowManager();  
// Display d = m.getDefaultDisplay(); // 获取屏幕宽、高用  
 WindowManager.LayoutParams p = dialogWindow.getAttributes(); // 获取对话框当前的参数值  
// p.height = (int) (d.getHeight() ); // 高度设置为屏幕  
// p.width = (int) (d.getWidth() ); // 宽度设置为屏幕  
 dialogWindow.setAttributes(p);  
 getWindow().setLayout(ViewGroup.LayoutParams.*MATCH\_PARENT*, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
  
 // 绑定按钮  
 back = findViewById(R.id.*back\_11*);  
 listView = findViewById(R.id.*list\_update\_listview*);  
  
 // 返回  
 back.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 dismiss();  
 }  
 });  
  
 // 列表点击  
 inindrawer();  
  
 }  
 public void inindrawer(){  
 list = new ArrayList<>();  
 Bitmap bmp = BitmapFactory.*decodeResource*(context.getResources(), R.drawable.*menu\_list*);  
 List\_menu list\_menu\_item ;  
 for(int i=0 ; i < *allList*.size(); i++){  
 list\_menu\_item = new List\_menu(bmp,*allList*.get(i).getListname());  
 list.add(list\_menu\_item);  
 }  
  
 List\_menu\_Adapter list\_menu\_adapter = new List\_menu\_Adapter(context,R.layout.*list\_menu\_button*,list);  
 listView.setAdapter(list\_menu\_adapter);  
 list\_menu\_adapter.setOnInnerItemOnClickListener(this);  
 listView.setOnItemClickListener(this);  
 }  
  
 @Override  
 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
  
 }  
 private String nameget;  
 @Override  
 public void itemClick(View v) {  
 *position* = (Integer) v.getTag();  
 if (!list.get(*position*).getList\_menu\_name().equals("收集箱")) {  
 switch (v.getId()) {  
 case R.id.*imageView4*:  
 case R.id.*list\_menu\_name*:  
 nameget = list.get(*position*).getList\_menu\_name();  
 list\_update\_dialog = new List\_update\_dialog(context, R.style.*dialog*);  
 list\_update\_dialog.show();  
 list\_update\_dialog.setOnDismissListener(new OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialog) {  
 if (list\_update\_dialog.getIssue()) {  
 nameget = list\_update\_dialog.getList\_name();  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UpdateList updateList = new UpdateList();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 updateList.getRemoteInfo(nameget, *allList*.get(*position*).getListid());  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 update();  
 }  
  
 }  
 });  
  
 break;  
 default:  
 break;  
 }  
 }  
 }  
 public void update(){  
 Bitmap bmp = BitmapFactory.*decodeResource*(context.getResources(), R.drawable.*menu\_list*);  
 List\_menu list\_menu\_item = new List\_menu(bmp,nameget);  
 list.set(*position*,list\_menu\_item);  
 List\_menu\_Adapter list\_menu\_adapter = new List\_menu\_Adapter(context,R.layout.*list\_menu\_button*,list);  
 listView.setAdapter(list\_menu\_adapter);  
 list\_menu\_adapter.setOnInnerItemOnClickListener(this);  
 listView.setOnItemClickListener(this);  
 }  
 public void delete\_update(){  
 inindrawer();  
 List\_menu\_Adapter list\_menu\_adapter = new List\_menu\_Adapter(context,R.layout.*list\_menu\_button*,list);  
 listView.setAdapter(list\_menu\_adapter);  
 list\_menu\_adapter.setOnInnerItemOnClickListener(this);  
 listView.setOnItemClickListener(this);  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.app.Dialog;  
import android.content.Context;  
import android.os.Bundle;  
import android.text.Editable;  
import android.text.TextUtils;  
import android.text.TextWatcher;  
import android.view.View;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.Button;  
import android.widget.EditText;  
  
import com.example.administrator.timenote.Model.BeanListInformation;  
import com.example.administrator.timenote.R;  
  
public class List\_update\_dialog extends Dialog {  
 */\*\*  
 \* 上下文对象 \*  
 \*/* private Button back;  
 private Context context;  
 private boolean issue = false; // 是否验证成功  
 private EditText list\_update\_name;// 清单名输入框  
 public String getList\_name() {  
 return list\_name;  
 }  
 private String list\_name;// 读取列表名字  
 private Button sure;// 确认按钮  
  
 public boolean getIssue() {  
 return issue;  
 }  
  
 public List\_update\_dialog(Context context) {  
 super(context);  
 this.context = context;  
 }  
  
 public List\_update\_dialog(Context context, int theme) {  
 super(context, theme);  
 this.context = context;  
 }  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 // 指定布局  
 this.setContentView(R.layout.*list\_update*);  
  
 /\*  
 \* 获取验证码的窗口对象及参数对象以修改对话框的布局设置, 可以直接调用getWindow(),表示获得这个Activity的Window  
 \* 对象,这样这可以以同样的方式改变这个Activity的属性.  
 \*/  
 Window dialogWindow = this.getWindow();  
  
// WindowManager m = context.getWindowManager();  
// Display d = m.getDefaultDisplay(); // 获取屏幕宽、高用  
 WindowManager.LayoutParams p = dialogWindow.getAttributes(); // 获取对话框当前的参数值  
// p.height = (int) (d.getHeight() ); // 高度设置为屏幕  
// p.width = (int) (d.getWidth() ); // 宽度设置为屏幕  
 dialogWindow.setAttributes(p);  
  
 // 根据id在布局中找到控件对象  
 back = findViewById(R.id.*back\_12*);  
 list\_update\_name = findViewById(R.id.*list\_update\_1*);  
 sure = findViewById(R.id.*sure\_list\_update*);  
  
 list\_update\_name.setText(BeanListInformation.*allList*.get(List\_Update.*getPosition*()).getListname());  
  
 // 为按钮绑定点击事件监听器  
 back.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View ciew) {  
 dismiss();  
 }  
 });  
  
 // 确认按钮事件  
 sure.setEnabled(Boolean.*FALSE*);  
 sure.setTextColor(context.getResources().getColor(R.color.*color\_ok\_2*));  
  
 sure.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View view) {  
 issue = true;  
 dismiss();  
  
 }  
 });  
  
 list\_update\_name.addTextChangedListener(new TextWatcher() {  
 @Override  
 public void beforeTextChanged(CharSequence s, int start, int count, int after) {  
 sure.setTextColor(context.getResources().getColor(R.color.*color\_ok\_2*));  
 sure.setEnabled(Boolean.*FALSE*);  
 }  
  
 // 如果没有输入则禁止确认  
 public void onTextChanged(CharSequence s, int start, int before, int count) {  
 list\_name = list\_update\_name.getText().toString();  
  
 if (TextUtils.*isEmpty*(list\_update\_name.getText()) | list\_name.equals("准备做什么？")) {  
 sure.setTextColor(context.getResources().getColor(R.color.*color\_ok\_2*));  
 sure.setEnabled(Boolean.*FALSE*);  
 } else {  
 sure.setTextColor(context.getResources().getColor(R.color.*color\_ok\_1*));  
 sure.setEnabled(Boolean.*TRUE*);  
 }  
 }  
 public void afterTextChanged(Editable s) {  
  
 }  
 });  
  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.annotation.SuppressLint;  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ArrayAdapter;  
import android.widget.Button;  
import android.view.View.OnClickListener;  
  
import com.example.administrator.timenote.R;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class ListAdapter extends ArrayAdapter<String> implements OnClickListener {  
 private int resourceId;  
 private InnerItemOnclickListener mListener;  
  
 public ListAdapter(Context context, int resource, ArrayList<String> objects) {  
 super(context, resource, objects);  
 resourceId=resource;  
 }  
  
 public View getView(int position, View convertView, ViewGroup parent){  
 ViewHolder holder=null;  
 @SuppressLint("ViewHolder")  
 View view = LayoutInflater.*from*(getContext()).inflate(resourceId,null,false);  
 holder=new ViewHolder();  
 holder.mButton= (Button) view.findViewById(R.id.*list\_select\_button*);  
 holder.mButton.setText(getItem(position));  
 holder.mButton.setOnClickListener(this);  
 holder.mButton.setTag(position);  
 return view;  
  
 }  
  
 interface InnerItemOnclickListener {  
 void itemClick(View v);  
 }  
  
 public void setOnInnerItemOnClickListener(InnerItemOnclickListener listener){  
 this.mListener=listener;  
 }  
  
 @Override  
 public void onClick(View v) {  
 mListener.itemClick(v);  
 }  
 static class ViewHolder{  
 Button mButton;  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.annotation.SuppressLint;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.content.pm.ActivityInfo;  
import android.os.Handler;  
import android.os.Looper;  
import android.os.Message;  
import android.os.StrictMode;  
import android.os.SystemClock;  
import android.support.v7.app.AlertDialog;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.text.method.PasswordTransformationMethod;  
import android.view.KeyEvent;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.example.administrator.timenote.Manager.UserManager.UserActivity;  
import com.example.administrator.timenote.Manager.UserManager.UserLogin;  
import com.example.administrator.timenote.Manager.UserManager.loginServe;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
import com.example.administrator.timenote.Threadcontroal.MyHandler;  
  
import java.io.UnsupportedEncodingException;  
import java.util.Map;  
  
import static com.example.administrator.timenote.Model.BeanUserInformation.*currentLoginUser*;  
import static com.example.administrator.timenote.Model.BeanUserInformation.*tryLoginUser*;  
import static java.security.AccessController.*getContext*;  
  
  
public class MainActivity extends AppCompatActivity {  
  
 private EditText pwd, uesrid;  
 public static String *suesrid*;  
 private Button button1;  
 public static Handler *myHandler*;  
 //Message msg = new Message();  
 //Message msg = myHandler.obtainMessage();  
 //Log.("msg",msg);  
 private String spwd;  
 @Override  
 //访问网络同时加入这个  
 @SuppressLint("NewApi")  
 protected void onCreate(Bundle savedInstanceState) {  
 //允许使用webervice同时启用网络访问  
 StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();  
 StrictMode.*setThreadPolicy*(policy);  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*loading\_layout*);  
 button1 = findViewById(R.id.*loading\_1*);  
 Button button2 = findViewById(R.id.*sign\_up\_1*);  
 Button button3 = findViewById(R.id.*pwd\_change\_1*);  
 pwd = (EditText) findViewById(R.id.*pwd\_1*);  
 uesrid = (EditText) findViewById(R.id.*userid\_1*);  
  
 //强制竖屏  
 {  
 setRequestedOrientation(ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT*);  
 }  
  
 // 隐藏密码  
 pwd.setTransformationMethod(PasswordTransformationMethod  
 .*getInstance*());  
 // 密码文本框监听回车  
 pwd.setOnEditorActionListener(new TextView.OnEditorActionListener() {  
 public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {  
 pwd.clearFocus();  
 return false;  
 }  
 });  
  
 // 用户名文本框监听回车  
 uesrid.setOnEditorActionListener(new TextView.OnEditorActionListener() {  
 @Override  
 public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {  
 pwd.requestFocus();  
 return false;  
 }  
 });  
 Map<String, String> map = new loginServe().*getSaveUserInfo*(MainActivity.this);  
 {  
 if(map!=null){  
 *suesrid* = map.get("useremail");  
 spwd = map.get("password");  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UserLogin userLogin = new UserLogin();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 userLogin.getRemoteInfo(*suesrid*);  
 //myHandler.obtainMessage(1,u\_rst).sendToTarget();  
 //msg.obj = u\_rst;  
 //msg.what = 1;  
 //myHandler.sendMessage(msg);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 //判断账号密码是否匹配，是则进入主界面，否则弹框提示  
 AlertDialog.Builder alterDialog = new AlertDialog.Builder(MainActivity.this);  
 if (((*suesrid*.equals(*tryLoginUser*.getUseremail())) && (spwd.equals(*tryLoginUser*.getUserpassword())) && *tryLoginUser*.getStopdate().equals("0001-01-01T00:00:00"))) {  
 Intent intent = new Intent(MainActivity.this, Calendar\_View.class);  
 startActivity(intent);  
 *currentLoginUser* = *tryLoginUser*;  
 boolean result = new loginServe().*saveUserInfo*(MainActivity.this,*suesrid*,spwd);  
 finish();  
 }  
 }  
 }  
 // 登录按钮  
 button1.setOnClickListener(new View.OnClickListener() {  
 @SuppressLint("HandlerLeak")  
 @Override  
 //获得文本框内容  
// public void onClick(View v) {  
// Toast.makeText(MainActivity.this, uesrid.getText().toString(), Toast.LENGTH\_SHORT).show();  
// }  
  
 public void onClick(View v) {  
 *suesrid*=uesrid.getText().toString();  
 spwd = pwd.getText().toString();  
  
// myHandler = new Handler(){  
// @Override  
// public void handleMessage(Message msg) {  
// super.handleMessage(msg);  
// switch(msg.what){  
// case 1:tryLoginUser = (BeanUserInformation) msg.obj;break;  
// case 0:Toast.makeText(MainActivity.this,"没有该用户",Toast.LENGTH\_SHORT).show();break;  
// case -1:Toast.makeText(MainActivity.this,"服务器错误",Toast.LENGTH\_SHORT).show();break;  
// }  
//  
//// myHandler=new MainActivity.MyHandler(Looper.myLooper());  
// }  
// };  
// 创建子线程并引用webservice层的LoadUser方法  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UserLogin userLogin = new UserLogin();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 userLogin.getRemoteInfo(*suesrid*);  
 //myHandler.obtainMessage(1,u\_rst).sendToTarget();  
 //msg.obj = u\_rst;  
 //msg.what = 1;  
 //myHandler.sendMessage(msg);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 //判断账号密码是否匹配，是则进入主界面，否则弹框提示  
 AlertDialog.Builder alterDialog = new AlertDialog.Builder(MainActivity.this);  
 if (((*suesrid*.equals(*tryLoginUser*.getUseremail())) && (spwd.equals(*tryLoginUser*.getUserpassword())) && *tryLoginUser*.getStopdate().equals("0001-01-01T00:00:00"))) {  
 Intent intent = new Intent(MainActivity.this, Calendar\_View.class);  
 startActivity(intent);  
 *currentLoginUser* = *tryLoginUser*;  
 boolean result = new loginServe().*saveUserInfo*(MainActivity.this,*suesrid*,spwd);  
 finish();  
 }  
 else if((*suesrid*.equals(""))){  
 alterDialog.setTitle("提示！");  
 alterDialog.setMessage("账号不能为空");  
 alterDialog.setCancelable(false);  
 alterDialog.setPositiveButton("确认", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(MainActivity.this, "确认", Toast.*LENGTH\_SHORT*).show();  
 uesrid.requestFocus();  
 }  
 });  
 alterDialog.show();  
 }  
 else if((spwd.equals(""))){  
 alterDialog.setTitle("提示！");  
 alterDialog.setMessage("密码不能为空");  
 alterDialog.setCancelable(false);  
 alterDialog.setPositiveButton("确认", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(MainActivity.this, "确认", Toast.*LENGTH\_SHORT*).show();  
 uesrid.requestFocus();  
 }  
 });  
 alterDialog.show();  
 }  
// else if(tryLoginUser.getUseremail()==null){  
// alterDialog.setTitle("提示！");  
// alterDialog.setMessage("用户不存在");  
// alterDialog.setCancelable(false);  
// alterDialog.setPositiveButton("确认", new DialogInterface.OnClickListener() {  
// @Override  
// public void onClick(DialogInterface dialog, int which) {  
// Toast.makeText(MainActivity.this, "确认", Toast.LENGTH\_SHORT).show();  
// uesrid.requestFocus();  
// }  
// });  
// alterDialog.show();  
// }  
 else if(*tryLoginUser*.getUseremail()!=null&&!*tryLoginUser*.getStopdate().equals("0001-01-01T00:00:00")){  
 alterDialog.setTitle("您的账号还未激活");  
 alterDialog.setMessage("是否现在激活");  
 alterDialog.setCancelable(false);  
 alterDialog.setPositiveButton("否", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(MainActivity.this, "尚未激活", Toast.*LENGTH\_SHORT*).show();  
 uesrid.requestFocus();  
 }  
 });  
 alterDialog.setNegativeButton("是", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 // *TODO: 2018/6/6 调用邮箱验证弹框* final Email\_sure1 email\_sure= new Email\_sure1(MainActivity.this,R.style.*dialog*);  
 email\_sure.show();  
 email\_sure.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialog) {  
 if(email\_sure.getIssue())  
 {  
 // 激活  
 Thread r = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UserActivity userActivity = new UserActivity();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 userActivity.getRemoteInfo(*suesrid*);  
  
 }  
 });  
 r.start();  
 try {  
 r.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 pwd.setText("");  
 Toast.*makeText*(MainActivity.this, "用户激活成功", Toast.*LENGTH\_SHORT*).show();  
 uesrid.requestFocus();  
 }  
 }  
 });  
 }  
 });  
  
 alterDialog.show();  
 }  
 else if((*tryLoginUser*.getUseremail()==null)||(!*suesrid*.equals(*tryLoginUser*.getUseremail())) || (!spwd.equals(*tryLoginUser*.getUserpassword())))  
 {  
 alterDialog.setTitle("糟糕！");  
 alterDialog.setMessage("账号或密码错误");  
 alterDialog.setCancelable(false);  
 alterDialog.setPositiveButton("确认", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(MainActivity.this, "确认", Toast.*LENGTH\_SHORT*).show();  
 pwd.setText("");  
 uesrid.requestFocus();  
 }  
 });  
 alterDialog.show();  
 }  
 }  
 });  
  
 //注册按钮  
 button2.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 pwd.requestFocus();  
 uesrid.requestFocus();  
 Intent intent = new Intent(MainActivity.this, Sign\_UpActivity.class);  
 startActivity(intent);  
 //转到注册界面  
  
 }  
 });  
  
 //忘记密码按钮  
 button3.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View view) {  
 pwd.requestFocus();  
 uesrid.requestFocus();  
 Intent intent = new Intent(MainActivity.this, Forgot\_pwd.class);  
 startActivity(intent);  
 //转到忘记密码界面  
  
 }  
 });  
 }  
  
  
  
  
// class MyThread implements Runnable {  
// public void run() {  
// //从消息池中取出一个message  
// Message msg = myHandler.obtainMessage();  
// //Bundle是message中的数据  
// Bundle b = new Bundle();  
// b.putString("color", "我的");  
// msg.setData(b);  
// //传递数据  
// myHandler.sendMessage(msg); // 向Handler发送消息,更新UI  
// }  
// }  
  
}

package com.example.administrator.timenote.Ui;  
  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
  
import com.example.administrator.timenote.R;  
  
public class Mainlist extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*main\_list\_1*);  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.BaseAdapter;  
import android.widget.CheckBox;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
import com.example.administrator.timenote.Model.BeanEventInformation;  
import com.example.administrator.timenote.Model.task;  
import com.example.administrator.timenote.R;  
  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.TreeSet;  
  
public class MyCustomAdapter extends BaseAdapter implements View.OnClickListener{  
  
 public static ArrayList<BeanEventInformation> getData() {  
 return *data*;  
 }  
  
 public static void setData(ArrayList<BeanEventInformation> data) {  
 MyCustomAdapter.*data* = data;  
 }  
  
 private static ArrayList<BeanEventInformation> *data* = new ArrayList<>();  
 private LayoutInflater inflater;  
 private ListAdapter.InnerItemOnclickListener mListener;  
 private TreeSet<Integer> set = new TreeSet<>();  
 private static final int *TYPE\_ITEM* = 0;  
 private static final int *TYPE\_SEPARATOR* = 1;  
 private static final int *TYPE\_MAX\_COUNT* = *TYPE\_SEPARATOR* + 1;  
 private int index = 0;  
  
  
 public MyCustomAdapter(Context context) {  
 *data*.clear();  
 inflater = LayoutInflater.*from*(context);  
 }  
  
 public void addItem(BeanEventInformation item) {  
 *data*.add(item);  
 }  
  
 public void addSeparatorItem(BeanEventInformation item) {  
 *data*.add(item);  
 set.add(*data*.size() - 1);  
 }  
  
 public int getItemViewType(int position) {  
 return set.contains(position) ? *TYPE\_SEPARATOR* : *TYPE\_ITEM*;  
 }  
  
 @Override  
 public int getViewTypeCount() {  
 return *TYPE\_MAX\_COUNT*;  
 }  
  
 @Override  
 public int getCount() {  
 return *data*.size();  
 }  
  
 @Override  
 public Object getItem(int position) {  
 return *data*.get(position);  
 }  
  
 @Override  
 public long getItemId(int position) {  
 return position;  
 }  
  
 @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
 ViewHolder holder = null;  
 int type = getItemViewType(position);  
 holder = new ViewHolder();  
 switch (type) {  
 case *TYPE\_SEPARATOR*:  
 convertView = inflater.inflate(R.layout.*item2*, null);  
 holder.date1 = convertView .findViewById(R.id.*item\_2*);  
 holder.t = *data*.get(position);  
// holder.format=new SimpleDateFormat("yyyy年MM月dd日 HH:mm:SS");  
// holder.date1.setText(holder.format.format(holder.t.getDate\_1()));  
 break;  
  
 case *TYPE\_ITEM*:  
 convertView = inflater.inflate(R.layout.*arrary\_button*, null);  
 holder.taskname = convertView .findViewById(R.id.*taskname*);  
 holder.date1 = convertView .findViewById(R.id.*data\_1*);  
 holder.view\_line = convertView .findViewById(R.id.*view\_color\_1*);  
 holder.taskdes = convertView .findViewById(R.id.*taskdes*);  
 holder.ImageId = convertView .findViewById(R.id.*image\_level*);  
 holder.checkBox = convertView.findViewById(R.id.*checkBox*);  
 break;  
  
 }  
 convertView.setTag(holder);  
  
 holder.t = *data*.get(position);  
 switch (type) {  
 case *TYPE\_SEPARATOR*:  
// holder.format = new SimpleDateFormat("yyyy年MM月dd日");  
 holder.date1.setText(holder.t.getDatetype());  
 break;  
 case *TYPE\_ITEM*:  
 holder.taskname.setText(holder.t.getEventname());  
 holder.taskname.setOnClickListener(this);  
 holder.taskname.setTag(position);  
  
 holder.format = new SimpleDateFormat("yyyy年MM月dd日 HH:mm:SS");  
 if(!holder.format.format(holder.t.getEventdate()).equals("0001年01月01日 00:00:00")){  
 holder.date1.setText(holder.format.format(holder.t.getEventdate()));  
 }  
 else {  
// holder.date1.setText("无");  
 }  
 holder.date1.setOnClickListener(this);  
 holder.date1.setTag(position);  
  
 holder.taskdes.setText(holder.t.getEventnote());  
 holder.taskdes.setOnClickListener(this);  
 holder.taskdes.setTag(position);  
  
 if(holder.t.getCheckBox()==1) {  
 holder.checkBox.setChecked(true);  
 }  
 else{  
 holder.checkBox.setChecked(false);  
 }  
  
 holder.checkBox.setOnClickListener(this);  
 holder.checkBox.setTag(position);  
  
 switch (holder.t.getEventpriority()){  
 case 0:  
 index = R.drawable.*level0*;break;  
 case 1:  
 index = R.drawable.*level1*;break;  
 case 2:  
 index = R.drawable.*level2*;break;  
 case 3:  
 index = R.drawable.*level3*;break;  
 }  
 holder.ImageId.setImageResource(index);  
 holder.ImageId.setOnClickListener(this);  
 holder.ImageId.setTag(position);  
  
 break;  
 }  
  
  
 return convertView;  
 }  
  
 interface InnerItemOnclickListener {  
 void itemClick(View v);  
 }  
  
 public void setOnInnerItemOnClickListener(ListAdapter.InnerItemOnclickListener listener){  
 this.mListener=listener;  
 }  
  
 @Override  
 public void onClick(View v) {  
 mListener.itemClick(v);  
 }  
  
 public static class ViewHolder {  
 public TextView taskname;  
 public TextView date1;  
 public View view\_line;  
 public TextView taskdes;  
 //public TextView taskp;  
 public ImageView ImageId;  
 private CheckBox checkBox;  
 public SimpleDateFormat format;  
 public BeanEventInformation t;  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.BaseAdapter;  
import android.widget.CheckBox;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
import com.example.administrator.timenote.Model.BeanEventInformation;  
import com.example.administrator.timenote.R;  
  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.TreeSet;  
  
public class MyCustomAdapter2 extends BaseAdapter implements View.OnClickListener{  
  
 public static ArrayList<BeanEventInformation> getData() {  
 return *data*;  
 }  
  
 public static void setData(ArrayList<BeanEventInformation> data) {  
 MyCustomAdapter2.*data* = data;  
 }  
  
 private static ArrayList<BeanEventInformation> *data* = new ArrayList<>();  
 private LayoutInflater inflater;  
 private ListAdapter.InnerItemOnclickListener mListener;  
 private TreeSet<Integer> set = new TreeSet<>();  
 private static final int *TYPE\_ITEM* = 0;  
 private static final int *TYPE\_SEPARATOR* = 1;  
 private static final int *TYPE\_MAX\_COUNT* = *TYPE\_SEPARATOR* + 1;  
 private int index = 0;  
  
  
 public MyCustomAdapter2(Context context) {  
 *data*.clear();  
 inflater = LayoutInflater.*from*(context);  
 }  
  
 public void addItem(BeanEventInformation item) {  
 *data*.add(item);  
 }  
  
 public void addSeparatorItem(BeanEventInformation item) {  
 *data*.add(item);  
 set.add(*data*.size() - 1);  
 }  
  
 public int getItemViewType(int position) {  
 return set.contains(position) ? *TYPE\_SEPARATOR* : *TYPE\_ITEM*;  
 }  
  
 @Override  
 public int getViewTypeCount() {  
 return *TYPE\_MAX\_COUNT*;  
 }  
  
 @Override  
 public int getCount() {  
 return *data*.size();  
 }  
  
 @Override  
 public Object getItem(int position) {  
 return *data*.get(position);  
 }  
  
 @Override  
 public long getItemId(int position) {  
 return position;  
 }  
  
 @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
 ViewHolder holder = null;  
 int type = getItemViewType(position);  
  
 holder = new ViewHolder();  
 switch (type) {  
 case *TYPE\_SEPARATOR*:  
 convertView = inflater.inflate(R.layout.*item2*, null);  
 holder.date1 = convertView.findViewById(R.id.*item\_2*);  
 holder.t = *data*.get(position);  
// holder.format=new SimpleDateFormat("yyyy年MM月dd日 HH:mm:SS");  
// holder.date1.setText(holder.format.format(holder.t.getDate\_1()));  
 break;  
  
 case *TYPE\_ITEM*:  
 convertView = inflater.inflate(R.layout.*arrary\_button*, null);  
 holder.taskname = convertView.findViewById(R.id.*taskname*);  
 holder.date1 = convertView.findViewById(R.id.*data\_1*);  
 holder.view\_line = convertView.findViewById(R.id.*view\_color\_1*);  
 holder.taskdes = convertView.findViewById(R.id.*taskdes*);  
 holder.ImageId = convertView.findViewById(R.id.*image\_level*);  
 holder.checkBox = convertView.findViewById(R.id.*checkBox*);  
 break;  
 }  
  
 convertView.setTag(holder);  
  
 holder.t = *data*.get(position);  
 switch (type) {  
 case *TYPE\_SEPARATOR*:  
// holder.format = new SimpleDateFormat("yyyy年MM月dd日");  
 holder.date1.setText(holder.t.getDatetype());  
 break;  
 case *TYPE\_ITEM*:  
 holder.taskname.setText(holder.t.getEventname());  
 holder.taskname.setOnClickListener(this);  
 holder.taskname.setTag(position);  
  
 holder.format = new SimpleDateFormat("yyyy年MM月dd日 HH:mm:SS");  
 if(!holder.format.format(holder.t.getEventdate()).equals("0001年01月01日 00:00:00")){  
 holder.date1.setText(holder.format.format(holder.t.getEventdate()));  
 }  
 holder.date1.setOnClickListener(this);  
 holder.date1.setTag(position);  
  
 holder.taskdes.setText(holder.t.getEventnote());  
 holder.taskdes.setOnClickListener(this);  
 holder.taskdes.setTag(position);  
  
 if(holder.t.getCheckBox()==1) {  
 holder.checkBox.setChecked(true);  
 }  
 else{  
 holder.checkBox.setChecked(false);  
 }  
  
 holder.checkBox.setOnClickListener(this);  
 holder.checkBox.setTag(position);  
  
 switch (holder.t.getEventpriority()){  
 case 0:  
 index = R.drawable.*level0*;break;  
 case 1:  
 index = R.drawable.*level1*;break;  
 case 2:  
 index = R.drawable.*level2*;break;  
 case 3:  
 index = R.drawable.*level3*;break;  
 }  
 holder.ImageId.setImageResource(index);  
 holder.ImageId.setOnClickListener(this);  
 holder.ImageId.setTag(position);  
  
 break;  
 }  
  
  
 return convertView;  
 }  
  
 interface InnerItemOnclickListener {  
 void itemClick(View v);  
 }  
  
 public void setOnInnerItemOnClickListener(ListAdapter.InnerItemOnclickListener listener){  
 this.mListener=listener;  
 }  
  
 @Override  
 public void onClick(View v) {  
 mListener.itemClick(v);  
 }  
  
 public static class ViewHolder {  
 public TextView taskname;  
 public TextView date1;  
 public View view\_line;  
 public TextView taskdes;  
 //public TextView taskp;  
 public ImageView ImageId;  
 private CheckBox checkBox;  
 public SimpleDateFormat format;  
 public BeanEventInformation t;  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.BaseAdapter;  
import android.widget.CheckBox;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
import com.example.administrator.timenote.Model.BeanEventInformation;  
import com.example.administrator.timenote.R;  
  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.TreeSet;  
  
public class MyCustomAdapter3 extends BaseAdapter implements View.OnClickListener{  
  
 public static ArrayList<BeanEventInformation> getData() {  
 return *data*;  
 }  
  
 public static void setData(ArrayList<BeanEventInformation> data) {  
 MyCustomAdapter3.*data* = data;  
 }  
  
 private static ArrayList<BeanEventInformation> *data* = new ArrayList<>();  
 private LayoutInflater inflater;  
 private ListAdapter.InnerItemOnclickListener mListener;  
 //private TreeSet<Integer> set = new TreeSet<>();  
 private static final int *TYPE\_ITEM* = 0;  
 private static final int *TYPE\_SEPARATOR* = 1;  
 private static final int *TYPE\_MAX\_COUNT* = *TYPE\_SEPARATOR* + 1;  
 private int index = 0;  
  
  
 public MyCustomAdapter3(Context context) {  
 *data*.clear();  
 inflater = LayoutInflater.*from*(context);  
 }  
  
 public void addItem(BeanEventInformation item) {  
 *data*.add(item);  
 }  
  
// public void addSeparatorItem(BeanEventInformation item) {  
// data.add(item);  
// set.add(data.size() - 1);  
// }  
//  
// public int getItemViewType(int position) {  
// return set.contains(position) ? TYPE\_SEPARATOR : TYPE\_ITEM;  
// }  
  
// @Override  
// public int getViewTypeCount() {  
// return TYPE\_MAX\_COUNT;  
// }  
  
 @Override  
 public int getCount() {  
 return *data*.size();  
 }  
  
 @Override  
 public Object getItem(int position) {  
 return *data*.get(position);  
 }  
  
 @Override  
 public long getItemId(int position) {  
 return position;  
 }  
  
 @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
 ViewHolder holder = null;  
  
  
 holder = new ViewHolder();  
 convertView = inflater.inflate(R.layout.*task\_secect\_button*, null);  
 holder.taskname = convertView .findViewById(R.id.*task\_select\_button\_1*);  
 holder.t = *data*.get(position);  
 holder.taskname.setText(holder.t.getEventname());  
 holder.taskname.setOnClickListener(this);  
 holder.taskname.setTag(position);  
  
  
 return convertView;  
 }  
  
 interface InnerItemOnclickListener {  
 void itemClick(View v);  
 }  
  
 public void setOnInnerItemOnClickListener(ListAdapter.InnerItemOnclickListener listener){  
 this.mListener=listener;  
 }  
  
 @Override  
 public void onClick(View v) {  
 mListener.itemClick(v);  
 }  
  
 public static class ViewHolder {  
 public TextView taskname;  
 public TextView date1;  
 public View view\_line;  
 public TextView taskdes;  
 //public TextView taskp;  
 public ImageView ImageId;  
 private CheckBox checkBox;  
 public SimpleDateFormat format;  
 public BeanEventInformation t;  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.support.v4.view.PagerAdapter;  
import android.view.View;  
import android.view.ViewGroup;  
  
import java.util.ArrayList;  
  
public class MyPagerAdapter extends PagerAdapter {  
 private ArrayList<View> viewLists;  
  
 public MyPagerAdapter() {  
 }  
  
 public MyPagerAdapter(ArrayList<View> viewLists) {  
 super();  
 this.viewLists = viewLists;  
 }  
  
 public int getCount() {  
 return viewLists.size();  
 }  
  
 public boolean isViewFromObject(View view, Object object) {  
 return view == object;  
 }  
  
 public Object instantiateItem(ViewGroup container, int position) {  
 container.addView(viewLists.get(position));  
 return viewLists.get(position);  
 }  
  
 public void destroyItem(ViewGroup container, int position, Object object) {  
 container.removeView(viewLists.get(position));  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.app.Dialog;  
import android.content.Context;  
import android.os.Bundle;  
import android.text.Editable;  
import android.text.TextUtils;  
import android.text.TextWatcher;  
import android.view.View;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.Button;  
import android.widget.EditText;  
  
import com.example.administrator.timenote.Manager.ListManager.NewList;  
import com.example.administrator.timenote.R;  
  
public class New\_List extends Dialog {  
 */\*\*  
 \* 上下文对象 \*  
 \*/* private Button back;  
 private Context context;  
 private boolean issue = false; // 是否验证成功  
 private EditText list\_new\_name;// 清单名输入框  
 public String getList\_name() {  
 return list\_name;  
 }  
 private String list\_name;// 读取列表名字  
 private Button sure;// 确认按钮  
  
 public boolean getIssue() {  
 return issue;  
 }  
  
 public New\_List(Context context) {  
 super(context);  
 this.context = context;  
 }  
  
 public New\_List(Context context, int theme) {  
 super(context, theme);  
 this.context = context;  
 }  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 // 指定布局  
 this.setContentView(R.layout.*list\_new*);  
  
 /\*  
 \* 获取验证码的窗口对象及参数对象以修改对话框的布局设置, 可以直接调用getWindow(),表示获得这个Activity的Window  
 \* 对象,这样这可以以同样的方式改变这个Activity的属性.  
 \*/  
 Window dialogWindow = this.getWindow();  
  
// WindowManager m = context.getWindowManager();  
// Display d = m.getDefaultDisplay(); // 获取屏幕宽、高用  
 WindowManager.LayoutParams p = dialogWindow.getAttributes(); // 获取对话框当前的参数值  
// p.height = (int) (d.getHeight() ); // 高度设置为屏幕  
// p.width = (int) (d.getWidth() ); // 宽度设置为屏幕  
 dialogWindow.setAttributes(p);  
  
 // 根据id在布局中找到控件对象  
 back = findViewById(R.id.*back\_10*);  
 list\_new\_name = findViewById(R.id.*list\_new\_1*);  
 sure = findViewById(R.id.*sure\_list\_new*);  
  
  
 // 为按钮绑定点击事件监听器  
 back.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View ciew) {  
 dismiss();  
 }  
 });  
  
 // 确认按钮事件  
 sure.setEnabled(Boolean.*FALSE*);  
 sure.setTextColor(context.getResources().getColor(R.color.*color\_ok\_2*));  
 sure.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View view) {  
 issue = true;  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* NewList newList = new NewList();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 newList.getRemoteInfo(list\_name);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 //list\_name;  
 dismiss();  
  
 }  
 });  
  
 list\_new\_name.addTextChangedListener(new TextWatcher() {  
 @Override  
 public void beforeTextChanged(CharSequence s, int start, int count, int after) {  
 sure.setTextColor(context.getResources().getColor(R.color.*color\_ok\_2*));  
 sure.setEnabled(Boolean.*FALSE*);  
 }  
  
 // 如果没有输入则禁止确认  
 public void onTextChanged(CharSequence s, int start, int before, int count) {  
 list\_name = list\_new\_name.getText().toString();  
  
 if (TextUtils.*isEmpty*(list\_new\_name.getText()) | list\_name.equals("准备做什么？")) {  
 sure.setTextColor(context.getResources().getColor(R.color.*color\_ok\_2*));  
 sure.setEnabled(Boolean.*FALSE*);  
 } else {  
 sure.setTextColor(context.getResources().getColor(R.color.*color\_ok\_1*));  
 sure.setEnabled(Boolean.*TRUE*);  
 }  
 }  
 public void afterTextChanged(Editable s) {  
  
 }  
 });  
  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.annotation.SuppressLint;  
import android.app.Dialog;  
import android.app.DialogFragment;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.os.Bundle;  
import android.text.Editable;  
import android.text.TextUtils;  
import android.text.TextWatcher;  
import android.util.Log;  
import android.view.Gravity;  
import android.view.View;  
import android.view.ViewGroup;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.FrameLayout;  
import android.widget.ImageView;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.bigkoo.pickerview.builder.TimePickerBuilder;  
import com.bigkoo.pickerview.listener.CustomListener;  
import com.bigkoo.pickerview.listener.OnTimeSelectChangeListener;  
import com.bigkoo.pickerview.listener.OnTimeSelectListener;  
import com.bigkoo.pickerview.view.TimePickerView;  
import com.example.administrator.timenote.Manager.TaskManager.LoadAllEvent;  
import com.example.administrator.timenote.Manager.TaskManager.NewEvent;  
import com.example.administrator.timenote.Model.BeanDRemindInformation;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
  
import org.w3c.dom.Text;  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.Date;  
  
public class New\_task extends Dialog {  
 */\*\*  
 \* 上下文对象 \*  
 \*/* private Button back; // 取消按钮  
 private Context context;  
 private Button dateselect; // 日期选择按钮  
 public String getdate; // 事务时间获取  
 private Button list; // 清单选择按钮  
 private Button level; // 优先级选择按钮  
 private Level\_select level\_select;// 优先级选择界面  
 public int listid = 0; // 清单id获取  
 private int priority = 0; // 清单优先级  
 private TimePickerView pvTime, pvCustomTime;// 时间选择界面  
 private Button sure; // 确定按钮  
 public String scode; // 事务名称获取  
 private Calendar selectedDate; // 系统时间获取  
 public EditText task; // 任务输入框  
 private boolean issure = false;  
 public boolean issure() {  
 return issure;  
 }  
  
 public New\_task(Context context) {  
 super(context);  
 this.context = context;  
 }  
  
 public New\_task(Context context, int theme) {  
 super(context, theme);  
 this.context = context;  
 selectedDate = Calendar.*getInstance*();  
 }  
 public New\_task(Context context, int theme, Calendar calendar) {  
 super(context, theme);  
 this.context = context;  
 this.selectedDate = calendar;  
 }  
  
 @SuppressLint("ResourceAsColor")  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 // 指定布局  
 this.setContentView(R.layout.*createa\_transaction*);  
 Window dialogWindow = this.getWindow();  
  
// WindowManager m = context.getWindowManager();  
// Display d = m.getDefaultDisplay(); // 获取屏幕宽、高用  
 WindowManager.LayoutParams p = dialogWindow.getAttributes(); // 获取对话框当前的参数值  
// p.height = (int) (d.getHeight() ); // 高度设置为屏幕  
// p.width = (int) (d.getWidth() ); // 宽度设置为屏幕  
 dialogWindow.setAttributes(p);  
  
// SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  
// getdate = sdf.format(selectedDate.getTime());  
 String defaulttime;  
 switch (BeanDRemindInformation.*defaultset*.getDefaulttime()){  
 case "0":defaulttime = "00:00";break;  
 case "1":defaulttime = "02:00";break;  
 case "2":defaulttime = "04:00";break;  
 case "3":defaulttime = "06:00";break;  
 case "4":defaulttime = "08:00";break;  
 case "5":defaulttime = "10:00";break;  
 case "6":defaulttime = "12:00";break;  
 case "7":defaulttime = "14:00";break;  
 case "8":defaulttime = "16:00";break;  
 case "9":defaulttime = "18:00";break;  
 case "10":defaulttime = "20:00";break;  
 case "11":defaulttime = "22:00";break;  
 default:defaulttime = BeanDRemindInformation.*defaultset*.getDefaulttime();break;  
 }  
 SimpleDateFormat sdf= new SimpleDateFormat("HH:mm");  
 Date date = null;  
 try {  
 date = sdf.parse(defaulttime);  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 Calendar calendar = Calendar.*getInstance*();  
 calendar.setTime(date);  
 selectedDate.set(Calendar.*HOUR\_OF\_DAY*,calendar.get(Calendar.*HOUR*));  
 selectedDate.set(Calendar.*MINUTE*,calendar.get(Calendar.*MINUTE*));  
 SimpleDateFormat sdf1 = new SimpleDateFormat("yyyy-MM-dd HH:mm");  
 getdate = sdf1.format(selectedDate.getTime());  
  
 // 根据id在布局中找到控件对象  
 back = findViewById(R.id.*back\_3*);  
 sure = findViewById(R.id.*sure\_task*);  
 dateselect = findViewById(R.id.*task\_date\_1*);  
 list = findViewById(R.id.*task\_list\_1*);  
 task = findViewById(R.id.*new\_task*);  
 level = findViewById(R.id.*task\_level\_1*);  
  
  
 // 取消按钮  
 back.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View ciew) {  
 dismiss();  
 }  
 });  
  
 //确认按钮事件  
 sure.setEnabled(Boolean.*FALSE*);  
 sure.setTextColor(context.getResources().getColor(R.color.*color\_ok\_2*));  
 sure.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View view) {  
 // 获取事务信息  
 scode = task.getText().toString();  
 // 新建插入  
 initsure();  
 // 重新获取  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* LoadAllEvent loadAllEvent = new LoadAllEvent();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 loadAllEvent.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid());  
  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 issure = true;  
 dismiss();  
 }  
 });  
  
 // 日期选择按钮  
 dateselect.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 initCustomTimePicker();  
 pvCustomTime.show();  
 }  
 });  
  
 // 清单选择按钮  
 list.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 final List\_select list\_select=new List\_select(context,R.style.*dialog*);  
 list\_select.show();  
 list\_select.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialog) {  
 if (list\_select.getIssure()){  
 listid = list\_select.*getNlistid*();  
 }  
 }  
 });  
 }  
 });  
  
 // 事务输入框  
 task.addTextChangedListener(new TextWatcher() {  
 // 如果没有输入则禁止确认  
 public void onTextChanged(CharSequence s, int start, int before, int count) {  
 if(TextUtils.*isEmpty*(task.getText())|task.getText().equals("准备做什么？"))  
 {  
 sure.setTextColor(context.getResources().getColor(R.color.*color\_ok\_2*));  
 sure.setEnabled(Boolean.*FALSE*);  
 }  
 else  
 {  
 sure.setTextColor(context.getResources().getColor(R.color.*color\_ok\_1*));  
 sure.setEnabled(Boolean.*TRUE*);  
 }  
 }  
  
 public void beforeTextChanged(CharSequence s, int start, int count, int after) {  
  
 }  
  
 public void afterTextChanged(Editable s) {  
 }  
  
 });  
  
 // 优先级选择  
 level.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 level\_select=new Level\_select(context,R.style.*dialog*);  
 level\_select.show();  
 // 监听弹框消失  
 level\_select.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialog) {  
 initlevel();  
 }  
 });  
 }  
 });  
  
 }  
  
 //  
 private void initlevel(){  
 switch(Level\_select.*getLevel*())  
 {  
 case 0:  
 level.setBackgroundResource(R.drawable.*level0*);  
 priority = 0;  
 break;  
 case 1:  
 level.setBackgroundResource(R.drawable.*level1*);  
 priority = 1;  
 break;  
 case 2:  
 level.setBackgroundResource(R.drawable.*level2*);  
 priority = 2;  
 break;  
 case 3:  
 level.setBackgroundResource(R.drawable.*level3*);  
 priority = 3;  
 break;  
 }  
 }  
 private void initCustomTimePicker() {//Dialog 模式下，在底部弹出  
 pvCustomTime = new TimePickerBuilder(context, new OnTimeSelectListener() {  
 @Override  
 public void onTimeSelect(Date date, View v) {  
 getdate = getTime(date);  
 selectedDate.setTime(date);  
 }  
 })  
 .setDate(selectedDate)  
 .setLayoutRes(R.layout.*pickerview\_custom\_time*, new CustomListener() {  
 @Override  
 public void customLayout(View v) {  
 final TextView tvSubmit = (TextView) v.findViewById(R.id.*tv\_finish*);  
 ImageView ivCancel = (ImageView) v.findViewById(R.id.*iv\_cancel*);  
 tvSubmit.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 pvCustomTime.returnData();  
 pvCustomTime.dismiss();  
 initCustomTimePicker2();  
 pvTime.show();  
 }  
 });  
 ivCancel.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 pvCustomTime.dismiss();  
 }  
 });  
 }  
 })  
 .setTimeSelectChangeListener(new OnTimeSelectChangeListener() {  
 @Override  
 public void onTimeSelectChanged(Date date) {  
 Log.*i*("pvTime", "onTimeSelectChanged");  
 }  
 })  
 .setType(new boolean[]{true, true, true, false, false, false})  
 .isDialog(true)  
 .build();  
 Dialog mDialog = pvCustomTime.getDialog();  
 if (mDialog != null) {  
  
 FrameLayout.LayoutParams params = new FrameLayout.LayoutParams(  
 ViewGroup.LayoutParams.*MATCH\_PARENT*,  
 ViewGroup.LayoutParams.*WRAP\_CONTENT*,  
 Gravity.*BOTTOM*);  
  
 params.leftMargin = 0;  
 params.rightMargin = 0;  
 pvCustomTime.getDialogContainerLayout().setLayoutParams(params);  
  
 Window dialogWindow = mDialog.getWindow();  
 if (dialogWindow != null) {  
 dialogWindow.setWindowAnimations(com.bigkoo.pickerview.R.style.*picker\_view\_slide\_anim*);//修改动画样式  
 dialogWindow.setGravity(Gravity.*CENTER*);//改成Bottom,底部显示  
 }  
 }  
 }  
 private void initCustomTimePicker2() {  
 pvTime = new TimePickerBuilder(context, new OnTimeSelectListener() {  
 @Override  
 public void onTimeSelect(Date date, View v) {  
 getdate = getTime(date);  
 }  
 })  
 .setDate(selectedDate)  
 .setLayoutRes(R.layout.*pickerview\_custom\_time*, new CustomListener() {  
 @Override  
 public void customLayout(View v) {  
 final TextView tvSubmit = (TextView) v.findViewById(R.id.*tv\_finish*);  
 ImageView ivCancel = (ImageView) v.findViewById(R.id.*iv\_cancel*);  
 tvSubmit.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 pvTime.returnData();  
 pvTime.dismiss();  
 }  
 });  
 ivCancel.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 pvTime.dismiss();  
 }  
 });  
 }  
 })  
 .setTimeSelectChangeListener(new OnTimeSelectChangeListener() {  
 @Override  
 public void onTimeSelectChanged(Date date) {  
 Log.*i*("pvTime", "onTimeSelectChanged");  
 }  
 })  
 .setType(new boolean[]{false, false, false, true, true, false})  
 .isDialog(true)  
 .build();  
 Dialog mDialog = pvTime.getDialog();  
 if (mDialog != null) {  
  
 FrameLayout.LayoutParams params = new FrameLayout.LayoutParams(  
 ViewGroup.LayoutParams.*MATCH\_PARENT*,  
 ViewGroup.LayoutParams.*WRAP\_CONTENT*,  
 Gravity.*BOTTOM*);  
  
 params.leftMargin = 0;  
 params.rightMargin = 0;  
 pvTime.getDialogContainerLayout().setLayoutParams(params);  
  
 Window dialogWindow = mDialog.getWindow();  
 if (dialogWindow != null) {  
 dialogWindow.setWindowAnimations(com.bigkoo.pickerview.R.style.*picker\_view\_slide\_anim*);// 修改动画样式  
 dialogWindow.setGravity(Gravity.*CENTER*); // 改成Bottom,底部显示  
 }  
 }  
 }  
 private String getTime(Date date) { // 可根据需要自行截取数据显示  
 SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm");  
 return format.format(date);  
 }  
  
 private void initsure(){  
 // *TODO: 2018/6/11 调用NewEvent类中的方法插入新事务* Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* NewEvent newEvent = new NewEvent();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 newEvent.getRemoteInfo(scode,priority,listid,getdate);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
}

package com.example.administrator.timenote.Ui;  
  
import android.annotation.SuppressLint;  
import android.app.AlarmManager;  
import android.app.PendingIntent;  
import android.content.ComponentName;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.content.res.Resources;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.os.Build;  
import android.os.Bundle;  
import android.os.Handler;  
import android.os.Looper;  
import android.support.annotation.RequiresApi;  
import android.support.design.widget.NavigationView;  
import android.support.v4.app.Fragment;  
import android.support.v4.widget.DrawerLayout;  
import android.util.Log;  
import android.view.KeyEvent;  
import android.view.LayoutInflater;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.SubMenu;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.AdapterView;  
import android.widget.ArrayAdapter;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.ListView;  
import android.widget.PopupMenu;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.example.administrator.timenote.Manager.DremindManager.LoadDremind;  
import com.example.administrator.timenote.Manager.ListManager.LoadAllList;  
import com.example.administrator.timenote.Manager.TaskManager.Complete;  
import com.example.administrator.timenote.Manager.TaskManager.LoadAllEvent;  
import com.example.administrator.timenote.Manager.TaskManager.LoadAllEventByPro;  
import com.example.administrator.timenote.Manager.TaskManager.UnComplete;  
import com.example.administrator.timenote.Manager.TaskManager.UpdatePid;  
import com.example.administrator.timenote.Manager.TaskManager.UpdatePriority;  
import com.example.administrator.timenote.Manager.UserManager.GetHead;  
import com.example.administrator.timenote.Manager.UserManager.UserLogin;  
import com.example.administrator.timenote.Model.BeanDRemindInformation;  
import com.example.administrator.timenote.Model.BeanEventInformation;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.Model.List\_menu;  
import com.example.administrator.timenote.Model.task;  
import com.example.administrator.timenote.R;  
  
import java.text.DateFormat;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.List;  
import java.util.UUID;  
  
import static com.example.administrator.timenote.Model.BeanEventInformation.*allEventList*;  
import static com.example.administrator.timenote.Model.BeanListInformation.*allList*;  
import static com.example.administrator.timenote.Model.BeanUserInformation.*tryLoginUser*;  
  
public class page1 extends Fragment implements AdapterView.OnItemClickListener, ListAdapter.InnerItemOnclickListener{  
  
 public static DrawerLayout *drawerLayout*;// 侧滑菜单  
 private MenuItem gMenuItem1, gMenuItem2;// list1的两个按钮  
 private LayoutInflater inflater;// 接口  
 private Button list1, taday\_1, all\_1, list2;// 任务列表（page1）的按钮从右到左  
 private TextView list\_name\_1;// 清单名称  
 private ListView listView;// 事务列表  
 private ListView list\_menu;// 清单列表  
 public static List\_Update *list\_update*;  
 private List<List\_menu> list;  
 private Level\_select level\_select;// 优先级选择界面  
 private New\_List new\_list;// 新建清单  
 public static NavigationView *navigationView*;  
 private Button new\_button;// 新建事务  
 public static ImageView *person*;// 侧滑菜单头像  
 private int position; // 点击位置  
 private TextView name;// 侧滑菜单名字  
 private static boolean *stase1* = true, *stase2* = false, *stase3* = true;// 列表显示状态  
 private int ringpid;  
 private Task\_Update task\_update;  
 public static Handler *handler1*;  
 private int eventidget;  
  
  
 public View onCreateView(final LayoutInflater inflater, final ViewGroup container, Bundle savedInstanceState) {  
 this.inflater = inflater;  
 View view = inflater.inflate(R.layout.*page1*, container, false);  
 *drawerLayout* = (DrawerLayout) view.findViewById(R.id.*page1*);  
 *navigationView* = (NavigationView)view.findViewById(R.id.*nav*);  
 list\_menu = view.findViewById(R.id.*list\_menu\_list*);  
 inindrawer();  
 *person* = view.findViewById(R.id.*person\_head*);  
 name = view.findViewById(R.id.*name*);  
// View headerView = navigationView .inflateHeaderView()getHeaderView(R.layout.head);// 获取头布局  
 new\_button=view.findViewById(R.id.*new\_task2*);  
  
 // 载入默认提醒设置  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* LoadDremind loadDremind = new LoadDremind();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 BeanDRemindInformation.*tsset* = loadDremind.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),-99);  
 BeanDRemindInformation.*defaultset* = BeanDRemindInformation.*tsset*;  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 // 侧滑菜单  
 *drawerLayout*.setDrawerLockMode(DrawerLayout.*LOCK\_MODE\_LOCKED\_CLOSED*); // 禁用手势侧滑操作  
 list2 = view.findViewById(R.id.*list\_1*);  
 list2.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View view) {  
 switch (view.getId()){  
 case R.id.*list\_1*:// 点击菜单，跳出侧滑菜单  
 if (*drawerLayout*.isDrawerOpen(*navigationView*)){  
 *drawerLayout*.closeDrawer(*navigationView*);  
 }else{  
 *drawerLayout*.openDrawer(*navigationView*);  
 }  
 break;  
 }  
 }  
 });  
 // 显示用户名  
 if(BeanUserInformation.*currentLoginUser*.getIcon()!=null)  
 {  
 Bitmap bm;  
 byte[] bitmapByte = BeanUserInformation.*currentLoginUser*.getIcon();  
 if (bitmapByte.length != 0) {  
 bm = BitmapFactory.*decodeByteArray*(bitmapByte, 0,bitmapByte.length);  
 *person*.setImageBitmap(bm);  
 } else {  
  
 }  
 }  
 else{  
 Thread r = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* GetHead getHead = new GetHead();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 getHead.getRemoteInfo();  
  
 }  
 });  
 r.start();  
 try {  
 r.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 if(BeanUserInformation.*currentLoginUser*.getIcon()!=null){  
 Bitmap bm;  
 byte[] bitmapByte = BeanUserInformation.*currentLoginUser*.getIcon();  
 if (bitmapByte.length != 0) {  
 bm = BitmapFactory.*decodeByteArray*(bitmapByte, 0,bitmapByte.length);  
 *person*.setImageBitmap(bm);  
 }  
 }  
 }  
 name.setText(BeanUserInformation.*currentLoginUser*.getUsername());  
  
 *navigationView*.setNavigationItemSelectedListener(new NavigationView.OnNavigationItemSelectedListener() {  
 public boolean onNavigationItemSelected(MenuItem item) {  
 //item.setChecked(true);  
 *drawerLayout*.closeDrawer(*navigationView*);  
 return true;  
 }  
 });  
 //page1按钮定义  
  
 //顶层菜单栏  
 list1 = view.findViewById(R.id.*list\_spot\_1*);  
 taday\_1 = view.findViewById(R.id.*taday\_1*);  
 all\_1 = view.findViewById(R.id.*all\_1*);  
  
 list\_name\_1 = view.findViewById(R.id.*list\_name\_1*);  
 //显示状态子菜单  
 list1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 showPopupMenu(list1);  
 }  
 });  
  
 //回到今天按钮  
 taday\_1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 //事务列表加载到今天  
 list\_name\_1.setText("今天");  
 *stase3* = false;  
 *initReflesh*();  
 inittoday();  
 }  
 });  
  
 //回到所有按钮  
 all\_1.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View view) {  
 //事务列表加载到所有  
 list\_name\_1.setText("所有");  
 *stase3* = true;  
 *initReflesh*();  
 initadapter();  
 }  
 });  
  
 //打开清单菜单  
  
 *initReflesh*(); // 重新从数据库读取  
  
 listView = (ListView) view.findViewById(R.id.*list\_view*);  
 initadapter();  
  
  
 //新建事务逻辑  
 new\_button.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View view) {  
 final New\_task new\_task= new New\_task(inflater.getContext(),R.style.*dialog*);  
 new\_task.show();  
 new\_task.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @RequiresApi(api = Build.VERSION\_CODES.*KITKAT*)  
 @Override  
 public void onDismiss(DialogInterface dialogInterface) {  
 if (new\_task.issure()) {  
 *initReflesh*();  
 if(*stase3*){  
 initadapter();  
 }  
 else {  
 inittoday();  
 }  
 if (new\_task.getdate != null) {  
 Date date = null;  
 SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm");  
 try {  
 date = formatter.parse(new\_task.getdate);  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 int eventid = 0;  
 for (int i = 0; i < BeanEventInformation.*allEventList*.size(); i++) {  
 if (eventid < BeanEventInformation.*allEventList*.get(i).getEventid())  
 eventid = BeanEventInformation.*allEventList*.get(i).getEventid();  
 }  
 setAlarm(date, BeanDRemindInformation.*defaultset*.getDremindring(),  
 Boolean.*valueOf*(BeanDRemindInformation.*defaultset*.getDremindvib()),  
 BeanDRemindInformation.*defaultset*.getAheadtime(),  
 BeanDRemindInformation.*defaultset*.getDremindrepeat(),  
 eventid, new\_task.scode, new\_task.getdate);  
  
 }  
 }  
 }  
 });  
 }  
 });  
 return view;  
  
 }  
  
 private void showPopupMenu(View view) {  
 // View当前PopupMenu显示的相对View的位置  
 PopupMenu popupMenu = new PopupMenu(this.inflater.getContext(), view);  
 // menu布局  
 popupMenu.getMenuInflater().inflate(R.menu.*showstyle*, popupMenu.getMenu());  
 gMenuItem1 = popupMenu.getMenu().findItem(R.id.*complete\_1*);  
 if (*stase1*) {  
 gMenuItem1.setTitle("显示已完成");  
 }  
 gMenuItem2 = popupMenu.getMenu().findItem(R.id.*priorit\_1*);  
 if (!*stase2*) {  
 gMenuItem2.setTitle("按优先级排序");  
 }  
 // menu的item点击事件  
 popupMenu.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {  
 @Override  
 public boolean onMenuItemClick(MenuItem item) {  
 switch (item.getItemId()) {  
 case R.id.*complete\_1*:  
 if (gMenuItem1.getTitle().equals("显示已完成")) {  
 *stase1* = false;  
 //修改状态重新加载事务列表  
 if(*stase3*){  
 initadapter();  
 }  
 else {  
 inittoday();  
 }  
 } else {  
 //修改状态重新加载事务列表  
 *stase1* = true;  
 if(*stase3*){  
 initadapter();  
 }  
 else {  
 inittoday();  
 }  
 }  
 break;  
 case R.id.*priorit\_1*:  
 if (gMenuItem2.getTitle().equals("按时间排序")) {  
 *stase2* = false;  
 //修改状态重新加载事务列表  
 initadapter();  
 } else {  
 //修改状态重新加载事务列表  
 *stase2* = true;  
 initadapter();  
 }  
 break;  
 default:  
 }  
 return true;  
 }  
 });  
 // PopupMenu关闭事件  
 popupMenu.setOnDismissListener(new PopupMenu.OnDismissListener() {  
 public void onDismiss(PopupMenu menu) {  
 }  
 });  
 popupMenu.show();  
 }  
  
 @Override  
 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
// Toast.makeText(getContext(),"列表 "+position+" 被点击了",Toast.LENGTH\_SHORT).show();  
 }  
  
 @Override  
 public void itemClick(View v) {  
 position = (Integer) v.getTag() ;  
 switch (v.getId()) {  
 case R.id.*taskdes*:  
 case R.id.*data\_1*:  
 case R.id.*taskname*:  
 task\_update = new Task\_Update(getContext(),R.style.*dialog1*,  
 MyCustomAdapter.*getData*().get(position).getEventid(),  
 MyCustomAdapter.*getData*().get(position).getEventname(),  
 MyCustomAdapter.*getData*().get(position).getEventnote(),  
 MyCustomAdapter.*getData*().get(position).getLeaveseventsign(),  
 MyCustomAdapter.*getData*().get(position).getEventdate());  
 task\_update.show();  
 task\_update.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @RequiresApi(api = Build.VERSION\_CODES.*KITKAT*)  
 @Override  
 public void onDismiss(DialogInterface dialogInterface) {  
 if (task\_update.issure())  
 {  
 int eventid = MyCustomAdapter.*getData*().get(position).getEventid();  
 *initReflesh*();  
 if(*stase3*){  
 initadapter();  
 }  
 else {  
 inittoday();  
 }  
  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* LoadDremind loadDremind = new LoadDremind();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 BeanDRemindInformation.*tsset* = loadDremind.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),MyCustomAdapter.*getData*().get(position).getEventid());  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 int pos = 0;  
 for (int i = 0; i < *allEventList*.size(); i++){  
 if (*allEventList*.get(i).getEventid() == eventid){  
 pos = i;  
 break;  
 }  
 }  
 if ( BeanDRemindInformation.*tsset* != null) {  
 updateAlarm(*allEventList*.get(pos).getEventdate(),  
 BeanDRemindInformation.*defaultset*.getDremindring(),  
 Boolean.*valueOf*(BeanDRemindInformation.*tsset*.getDremindvib()),  
 BeanDRemindInformation.*tsset*.getAheadtime(),  
 BeanDRemindInformation.*tsset*.getDremindrepeat(),  
 eventid,  
 *allEventList*.get(pos).getEventname(),  
 *allEventList*.get(pos).getPid(),  
 new SimpleDateFormat("yyyy-MM-dd hh:mm:ss").format(*allEventList*.get(pos).getEventdate()));  
  
 }  
 else {  
 updateAlarm(*allEventList*.get(pos).getEventdate(),  
 BeanDRemindInformation.*defaultset*.getDremindring(),  
 Boolean.*valueOf*(BeanDRemindInformation.*defaultset*.getDremindvib()),  
 BeanDRemindInformation.*defaultset*.getAheadtime(),  
 BeanDRemindInformation.*defaultset*.getDremindrepeat(),  
 eventid,  
 *allEventList*.get(pos).getEventname(),  
 *allEventList*.get(pos).getPid(),  
 new SimpleDateFormat("yyyy-MM-dd hh:mm:ss").format(*allEventList*.get(pos).getEventdate()));  
 }  
 }  
 }  
 });  
  
 break;  
 // 修改优先级  
 case R.id.*image\_level*:  
 level\_select=new Level\_select(getContext(),R.style.*dialog*);  
 level\_select.show();  
 // 监听弹框消失  
 level\_select.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialog) {  
 if ((Level\_select.*getLevel*() != -1) && (Level\_select.*getLevel*() != MyCustomAdapter.*getData*().get(position).getEventpriority())){  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UpdatePriority updatePriority = new UpdatePriority();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 updatePriority.getRemoteInfo(MyCustomAdapter.*getData*().get(position).getEventid(), Level\_select.*getLevel*());  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 *initReflesh*();  
 if(*stase3*){  
 initadapter();  
 }  
 else {  
 inittoday();  
 }  
 }  
 }  
 });  
 break;  
 // 修改完成状态  
 case R.id.*checkBox*:  
 // 设置已完成  
 if(MyCustomAdapter.*getData*().get(position).getCheckBox()==1)  
 {  
 // *TODO: 2018/6/17 修改为未完成* Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UnComplete unComplete = new UnComplete();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 unComplete.getRemoteInfo(MyCustomAdapter.*getData*().get(position).getEventid());  
 int i = 0;  
 for (; i < *allEventList*.size(); i++){  
 if (*allEventList*.get(i).getEventid() == MyCustomAdapter.*getData*().get(position).getEventid())  
 break;  
 }  
 *allEventList*.get(i).setCheckBox(0);  
  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 // 刷新  
 *initReflesh*();  
 if(*stase3*){  
 initadapter();  
 }  
 else {  
 inittoday();  
 }  
 }  
 else{  
 // *TODO: 2018/6/17 修改为已完成* Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* Complete complete = new Complete();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 complete.getRemoteInfo(MyCustomAdapter.*getData*().get(position).getEventid());  
 int i = 0;  
 for (; i < *allEventList*.size(); i++){  
 if (*allEventList*.get(i).getEventid() == MyCustomAdapter.*getData*().get(position).getEventid())  
 break;  
 }  
 *allEventList*.get(i).setCheckBox(1);  
  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 *initReflesh*();  
 if(*stase3*){  
 initadapter();  
 }  
 else {  
 inittoday();  
 }  
 }  
 break;  
  
 case R.id.*list\_menu\_name*:  
 case R.id.*imageView4*:  
 if (*drawerLayout*.isDrawerOpen(*navigationView*))  
 *drawerLayout*.closeDrawer(*navigationView*);  
 if (position == 0) {  
 list\_name\_1.setText("所有");  
 initadapter();  
 }  
 else if (position == 1)  
 inittoday();  
 else if (position == 2)  
 initrecentlythree();  
 else if ((position > 2) && (position < (*allList*.size() + 3)))  
 initlistinfo();  
  
 String name = list.get(position).getList\_menu\_name();  
 if(name.equals("新建清单"))  
 {  
 new\_list = new New\_List(getContext(),R.style.*dialog*);  
 new\_list.show();  
 new\_list.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 public void onDismiss(DialogInterface dialog) {  
 if (new\_list.getIssue()) {  
 inindrawer();  
 Toast.*makeText*(getContext(), new\_list.getList\_name(), Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
 }  
 else if(name.equals("设置清单")){  
 *list\_update* = new List\_Update(getContext(),R.style.*dialog1*);  
 *list\_update*.show();  
 *list\_update*.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialog) {  
 inindrawer();  
 }  
 });  
 }  
// else {  
// inindrawer();  
// if(stase3){  
// initadapter();  
// }  
// else {  
// inittoday();  
// }  
// }  
  
 break;  
 default:  
 break;  
 }  
 }  
 @RequiresApi(api = Build.VERSION\_CODES.*KITKAT*)  
 private void setAlarm(Date date, int ring, Boolean vib, int ahead, int repeat, int eventid, String eventname, String eventdate){  
 Calendar calendar = Calendar.*getInstance*();  
 if (calendar.getTimeInMillis() > date.getTime()){  
 return;  
 }  
 eventidget = eventid;  
 calendar.setTime(date);  
 long mils = 0;  
 switch (ahead){  
 case 0:break;  
 case 1:mils = 1000\*60\*5;break;  
 case 2:mils = 1000\*60\*30;break;  
 case 3:mils = 1000\*60\*60;break;  
 case 4:mils = 1000\*60\*60\*24;break;  
 }  
 //跳转  
 Intent intent=new Intent();  
 ringpid = UUID.*randomUUID*().hashCode();  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UpdatePid updatePid = new UpdatePid();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 updatePid.getRemoteInfo(eventidget, ringpid);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 intent.setAction("com.liuqian.android\_27alarm.RING");  
 intent.putExtra("id", ringpid);  
 intent.putExtra("ring", ring);  
 intent.putExtra("vib", vib);  
 intent.putExtra("ahead", ahead);  
 intent.putExtra("repeat", repeat);  
 intent.putExtra("eventid", eventid);  
 intent.putExtra("eventname", eventname);  
 intent.putExtra("eventdate", eventdate);  
 //PendingIntent  
 PendingIntent pendingIntent= PendingIntent.*getBroadcast*(getContext(),ringpid,intent,0);  
 //闹钟管理者 参数： 唤醒屏幕，  
 Calendar\_View.*alarmManager*.setExact(AlarmManager.*RTC\_WAKEUP*, calendar.getTimeInMillis() - mils, pendingIntent);  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.*KITKAT*)  
 private void updateAlarm(Date date, int ring, Boolean vib, int ahead, int repeat, int eventid, String eventname, int pid, String eventdate){  
 Calendar calendar = Calendar.*getInstance*();  
 if (calendar.getTimeInMillis() > date.getTime()){  
 return;  
 }  
 eventidget = eventid;  
 calendar.setTime(date);  
 long mils = 0;  
 switch (ahead){  
 case 0:break;  
 case 1:mils = 1000\*60\*5;break;  
 case 2:mils = 1000\*60\*30;break;  
 case 3:mils = 1000\*60\*60;break;  
 case 4:mils = 1000\*60\*60\*24;break;  
 }  
 //跳转  
 Intent intent=new Intent();  
 if (pid == -99){  
 ringpid = UUID.*randomUUID*().hashCode();  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UpdatePid updatePid = new UpdatePid();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 updatePid.getRemoteInfo(eventidget, ringpid);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 else  
 ringpid = pid;  
 intent.setAction("com.liuqian.android\_27alarm.RING");  
 intent.putExtra("id", ringpid);  
 intent.putExtra("ring", ring);  
 intent.putExtra("vib", vib);  
 intent.putExtra("ahead", ahead);  
 intent.putExtra("repeat", repeat);  
 intent.putExtra("eventid", eventid);  
 intent.putExtra("eventname", eventname);  
 intent.putExtra("eventdate", eventdate);  
 //PendingIntent  
 PendingIntent pendingIntent= PendingIntent.*getBroadcast*(getContext(),ringpid,intent,0);  
 //闹钟管理者 参数： 唤醒屏幕，  
 Calendar\_View.*alarmManager*.setExact(AlarmManager.*RTC\_WAKEUP*, calendar.getTimeInMillis() - mils, pendingIntent);  
 }  
  
 private void inindrawer(){  
 list = new ArrayList<>();  
 Bitmap bmp = BitmapFactory.*decodeResource*(getContext().getResources(), R.drawable.*menu\_list*);  
 List\_menu list\_menu\_item = new List\_menu(bmp,"所有");  
 list.add(list\_menu\_item);  
 list\_menu\_item = new List\_menu(bmp,"今天");  
 list.add(list\_menu\_item);  
 list\_menu\_item = new List\_menu(bmp,"近三天");  
 list.add(list\_menu\_item);  
  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* LoadAllList loadAllList = new LoadAllList();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 loadAllList.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid());  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 bmp = BitmapFactory.*decodeResource*(getContext().getResources(), R.drawable.*list\_1*);  
 for(int i=0 ; i < *allList*.size(); i++){  
 list\_menu\_item = new List\_menu(bmp,*allList*.get(i).getListname());  
 list.add(list\_menu\_item);  
 }  
  
 bmp = BitmapFactory.*decodeResource*(getContext().getResources(), R.drawable.*plus*);  
 list\_menu\_item = new List\_menu(bmp,"新建清单");  
 list.add(list\_menu\_item);  
 bmp = BitmapFactory.*decodeResource*(getContext().getResources(), R.drawable.*setup1*);  
 list\_menu\_item = new List\_menu(bmp,"设置清单");  
 list.add(list\_menu\_item);  
  
 List\_menu\_Adapter list\_menu\_adapter = new List\_menu\_Adapter(this.getContext(),R.layout.*list\_menu\_button*,list);  
 list\_menu.setAdapter(list\_menu\_adapter);  
 list\_menu\_adapter.setOnInnerItemOnClickListener(this);  
 list\_menu.setOnItemClickListener(this);  
 }  
  
 // 装载今天  
 private void inittoday(){  
 MyCustomAdapter adapter = new MyCustomAdapter(this.inflater.getContext());  
 listView.setAdapter(adapter);  
  
 int j = 0;  
 if(j<*allEventList*.size()) {  
 Calendar calendar1 = Calendar.*getInstance*();  
 Calendar calendar2 = Calendar.*getInstance*();  
 calendar2.setTime(*allEventList*.get(j).getEventdate());  
  
  
  
 //adapter.addSeparatorItem(new BeanEventInformation("今天"));  
 list\_name\_1.setText("今天");  
  
 if (*stase2*) {  
 List<BeanEventInformation> allEventBypro = new ArrayList<>();  
 for (int n = 0; n < *allEventList*.size(); n++) {  
 calendar2.setTime(*allEventList*.get(n).getEventdate());  
 if ((calendar1.get(Calendar.*YEAR*) == calendar1.get(Calendar.*YEAR*)) && (calendar1.get(Calendar.*MONTH*) == calendar2.get(Calendar.*MONTH*)) && (calendar1.get(Calendar.*DAY\_OF\_MONTH*) == calendar2.get(Calendar.*DAY\_OF\_MONTH*)))  
 allEventBypro.add(*allEventList*.get(n));  
 }  
 for (int l = 1; l < allEventBypro.size(); l++) {  
 BeanEventInformation eventchange;  
 int priority = allEventBypro.get(l - 1).getEventpriority();  
 int m = l - 1;  
 int k = l;  
 for (; k < allEventBypro.size(); k++) {  
 if (priority < allEventBypro.get(k).getEventpriority()) {  
 priority = allEventBypro.get(k).getEventpriority();  
 m = k;  
 }  
 }  
 eventchange = allEventBypro.get(l - 1);  
 allEventBypro.set(l - 1, allEventBypro.get(m));  
 allEventBypro.set(m, eventchange);  
 }  
 for (int k = 0; k < allEventBypro.size(); k++) {  
 if (*stase1* && allEventBypro.get(k).getCheckBox() == 1)  
 continue;  
 adapter.addItem(allEventBypro.get(k));  
 }  
 } else {  
 for (; j < *allEventList*.size(); j++) {  
 if (*stase1* && *allEventList*.get(j).getCheckBox() == 1)  
 continue;  
 calendar2.setTime(*allEventList*.get(j).getEventdate());  
 if ((calendar1.get(Calendar.*YEAR*) == calendar1.get(Calendar.*YEAR*)) && (calendar1.get(Calendar.*MONTH*) == calendar2.get(Calendar.*MONTH*)) && (calendar1.get(Calendar.*DAY\_OF\_MONTH*) == calendar2.get(Calendar.*DAY\_OF\_MONTH*)))  
 adapter.addItem(*allEventList*.get(j));  
 }  
 }  
 }  
  
 adapter.setOnInnerItemOnClickListener(this);  
 listView.setOnItemClickListener(this);  
 }  
  
 // 装载近三天  
 private void initrecentlythree(){  
 MyCustomAdapter adapter = new MyCustomAdapter(this.inflater.getContext());  
 listView.setAdapter(adapter);  
  
 //adapter.addSeparatorItem(new BeanEventInformation("最近三天"));  
 list\_name\_1.setText("最近三天");  
  
 long millis;  
 if (*stase2* == true){  
 List<BeanEventInformation> allEventBypro = new ArrayList<>();  
 for (int n = 0; n<*allEventList*.size(); n++){  
 millis = *allEventList*.get(n).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if ((millis >= 0) && (millis <= 3 \* (1000 \* 60 \* 60 \* 24)) && (!*allEventList*.get(n).getEventdate().toString().equals("Sat Jan 01 00:00:00 GMT+08:00 1")))  
 allEventBypro.add(*allEventList*.get(n));  
 }  
 for (int l = 1; l < allEventBypro.size(); l++) {  
 BeanEventInformation eventchange;  
 int priority = allEventBypro.get(l-1).getEventpriority();  
 int m = l - 1;  
 int k = l;  
 for (; k < allEventBypro.size(); k++) {  
 if (priority < allEventBypro.get(k).getEventpriority()) {  
 priority = allEventBypro.get(k).getEventpriority();  
 m = k;  
 }  
 }  
 eventchange = allEventBypro.get(l-1);  
 allEventBypro.set(l-1, allEventBypro.get(m));  
 allEventBypro.set(m, eventchange);  
 }  
 for(int k = 0; k < allEventBypro.size(); k++){  
 if(*stase1* == true && allEventBypro.get(k).getCheckBox() == 1)  
 continue;  
 adapter.addItem(allEventBypro.get(k));  
 }  
 }  
 else {  
 for (int i = 0; i < *allEventList*.size(); i++) {  
 millis = *allEventList*.get(i).getEventdate().getTime() - new Date(System.*currentTimeMillis*()).getTime();  
 if ((millis >= 0) && (millis <= 3 \* (1000 \* 60 \* 60 \* 24))) {  
 if (*stase1* == true && *allEventList*.get(i).getCheckBox() == 1) continue;  
 adapter.addItem(*allEventList*.get(i));  
 }  
  
 }  
 }  
  
 adapter.setOnInnerItemOnClickListener(this);  
 listView.setOnItemClickListener(this);  
 }  
  
 // 装载指定清单列表  
 private void initlistinfo(){  
 MyCustomAdapter adapter = new MyCustomAdapter(this.inflater.getContext());  
 listView.setAdapter(adapter);  
  
 // adapter.addSeparatorItem(new BeanEventInformation(allList.get(position - 3).getListname());  
 list\_name\_1.setText(*allList*.get(position - 3).getListname());  
  
 if (*stase2*){  
 List<BeanEventInformation> allEventBypro = new ArrayList<>();  
 for (int n = 0; n<*allEventList*.size(); n++){  
 if (position - 3 == *allEventList*.get(n).getListid())  
 allEventBypro.add(*allEventList*.get(n));  
 }  
 for (int l = 1; l < allEventBypro.size(); l++) {  
 BeanEventInformation eventchange;  
 int priority = allEventBypro.get(l-1).getEventpriority();  
 int m = l - 1;  
 int k = l;  
 for (; k < allEventBypro.size(); k++) {  
 if (priority < allEventBypro.get(k).getEventpriority()) {  
 priority = allEventBypro.get(k).getEventpriority();  
 m = k;  
 }  
 }  
 eventchange = allEventBypro.get(l-1);  
 allEventBypro.set(l-1, allEventBypro.get(m));  
 allEventBypro.set(m, eventchange);  
 }  
 for(int k = 0; k < allEventBypro.size(); k++){  
 if(*stase1* && allEventBypro.get(k).getCheckBox() == 1)  
 continue;  
 adapter.addItem(allEventBypro.get(k));  
 }  
 }  
 else {  
 for (int i = 0; i < *allEventList*.size(); i++) {  
 if ((*allList*.get(position - 3).getListid()) == *allEventList*.get(i).getListid()) {  
 if (*stase1* && *allEventList*.get(i).getCheckBox() == 1) continue;  
 adapter.addItem(*allEventList*.get(i));  
 }  
  
 }  
 }  
  
 adapter.setOnInnerItemOnClickListener(this);  
 listView.setOnItemClickListener(this);  
 }  
  
 // 装载所有  
 public void initadapter(){  
 MyCustomAdapter adapter = new MyCustomAdapter(this.inflater.getContext());  
 listView.setAdapter(adapter);  
  
  
 int i = 0;  
 int j = 0;  
 if(i < *allEventList*.size()){  
 if (*allEventList*.get(i).getEventdate().toString().equals("Sat Jan 01 00:00:00 GMT+08:00 1")) {  
 int index = 0;  
 for (; j< *allEventList*.size();j++) {  
 if (!*allEventList*.get(j).getEventdate().toString().equals("Sat Jan 01 00:00:00 GMT+08:00 1"))  
 break;  
 if(*stase1* == true && *allEventList*.get(j).getCheckBox() == 1)  
 continue;  
 index++;  
 }  
 if (index != 0){  
 adapter.addSeparatorItem(new BeanEventInformation("未设置时间"));  
 }  
  
 if (*stase2* == true){  
 List<BeanEventInformation> allEventBypro = new ArrayList<>();  
 for (int n = 0; n<*allEventList*.size(); n++){  
 if (*allEventList*.get(n).getEventdate().toString().equals("Sat Jan 01 00:00:00 GMT+08:00 1"))  
 allEventBypro.add(*allEventList*.get(n));  
 }  
 for (int l = 1; l < allEventBypro.size(); l++) {  
 BeanEventInformation eventchange;  
 int priority = allEventBypro.get(l-1).getEventpriority();  
 int m = l - 1;  
 int k = l;  
 for (; k < allEventBypro.size(); k++) {  
 if (priority < allEventBypro.get(k).getEventpriority()) {  
 priority = allEventBypro.get(k).getEventpriority();  
 m = k;  
 }  
 }  
 eventchange = allEventBypro.get(l-1);  
 allEventBypro.set(l-1, allEventBypro.get(m));  
 allEventBypro.set(m, eventchange);  
 }  
 for(int k = 0; k < allEventBypro.size(); k++){  
 if(*stase1* == true && allEventBypro.get(k).getCheckBox() == 1)  
 continue;  
 adapter.addItem(allEventBypro.get(k));  
 }  
 for (; i < *allEventList*.size(); i++) {  
 if (!*allEventList*.get(i).getEventdate().toString().equals("Sat Jan 01 00:00:00 GMT+08:00 1"))  
 break;  
 }  
 }  
 else{  
 for (; i < *allEventList*.size(); i++) {  
 if (!*allEventList*.get(i).getEventdate().toString().equals("Sat Jan 01 00:00:00 GMT+08:00 1"))  
 break;  
 if(*stase1* == true && *allEventList*.get(i).getCheckBox() == 1)  
 continue;  
 adapter.addItem(*allEventList*.get(i));  
 }  
 }  
  
 }  
 }  
  
 long millis;  
 if(i < *allEventList*.size()){  
 millis = *allEventList*.get(i).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if (millis < 0) {  
 int index = 0;  
 for (; j< *allEventList*.size();j++) {  
 millis = *allEventList*.get(j).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if (millis >= 0)  
 break;  
 if(*stase1* == true && *allEventList*.get(j).getCheckBox() == 1)  
 continue;  
 index++;  
 }  
 if (index != 0) {  
 adapter.addSeparatorItem(new BeanEventInformation("过去"));  
 }  
  
 if (*stase2* == true){  
 List<BeanEventInformation> allEventBypro = new ArrayList<>();  
 for (int n = 0; n<*allEventList*.size(); n++){  
 millis = *allEventList*.get(n).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if ((millis < 0) && (!*allEventList*.get(n).getEventdate().toString().equals("Sat Jan 01 00:00:00 GMT+08:00 1")))  
 allEventBypro.add(*allEventList*.get(n));  
 }  
 for (int l = 1; l < allEventBypro.size(); l++) {  
 BeanEventInformation eventchange;  
 int priority = allEventBypro.get(l-1).getEventpriority();  
 int m = l - 1;  
 int k = l;  
 for (; k < allEventBypro.size(); k++) {  
 if (priority < allEventBypro.get(k).getEventpriority()) {  
 priority = allEventBypro.get(k).getEventpriority();  
 m = k;  
 }  
 }  
 eventchange = allEventBypro.get(l-1);  
 allEventBypro.set(l-1, allEventBypro.get(m));  
 allEventBypro.set(m, eventchange);  
 }  
 for(int k = 0; k < allEventBypro.size(); k++){  
 if(*stase1* == true && allEventBypro.get(k).getCheckBox() == 1)  
 continue;  
 adapter.addItem(allEventBypro.get(k));  
 }  
 for (; i< *allEventList*.size();i++) {  
 millis = *allEventList*.get(i).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if (millis >= 0)  
 break;  
 }  
 }  
 else {  
 for (; i< *allEventList*.size();i++) {  
 millis = *allEventList*.get(i).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if (millis >= 0)  
 break;  
 if(*stase1* == true && *allEventList*.get(i).getCheckBox() == 1)  
 continue;  
 adapter.addItem(*allEventList*.get(i));  
 }  
 }  
  
 }  
 }  
  
 if(i<*allEventList*.size()){  
 millis = *allEventList*.get(i).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if ((millis >= 0) && (millis <= (1000 \* 60 \* 60 \* 24))) {  
 int index = 0;  
 for (; j< *allEventList*.size();j++) {  
 millis = *allEventList*.get(j).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if (millis > (1000 \* 60 \* 60 \* 24))  
 break;  
 if(*stase1* == true && *allEventList*.get(j).getCheckBox() == 1)  
 continue;  
 index++;  
 }  
 if (index != 0) {  
 adapter.addSeparatorItem(new BeanEventInformation("最近一天"));  
 }  
  
 if (*stase2* == true){  
 List<BeanEventInformation> allEventBypro = new ArrayList<>();  
 for (int n = 0; n<*allEventList*.size(); n++){  
 millis = *allEventList*.get(n).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if ((millis >= 0) && (millis <= (1000 \* 60 \* 60 \* 24)) && (!*allEventList*.get(n).getEventdate().toString().equals("Sat Jan 01 00:00:00 GMT+08:00 1")))  
 allEventBypro.add(*allEventList*.get(n));  
 }  
 for (int l = 1; l < allEventBypro.size(); l++) {  
 BeanEventInformation eventchange;  
 int priority = allEventBypro.get(l-1).getEventpriority();  
 int m = l - 1;  
 int k = l;  
 for (; k < allEventBypro.size(); k++) {  
 if (priority < allEventBypro.get(k).getEventpriority()) {  
 priority = allEventBypro.get(k).getEventpriority();  
 m = k;  
 }  
 }  
 eventchange = allEventBypro.get(l-1);  
 allEventBypro.set(l-1, allEventBypro.get(m));  
 allEventBypro.set(m, eventchange);  
 }  
 for(int k = 0; k < allEventBypro.size(); k++){  
 if(*stase1* == true && allEventBypro.get(k).getCheckBox() == 1)  
 continue;  
 adapter.addItem(allEventBypro.get(k));  
 }  
 for (; i < *allEventList*.size(); i++) {  
 millis = *allEventList*.get(i).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if (millis > (1000 \* 60 \* 60 \* 24))  
 break;  
 }  
 }  
 else {  
 for (; i < *allEventList*.size(); i++) {  
 millis = *allEventList*.get(i).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if (millis > (1000 \* 60 \* 60 \* 24))  
 break;  
 if (*stase1* == true && *allEventList*.get(i).getCheckBox() == 1)  
 continue;  
 adapter.addItem(*allEventList*.get(i));  
 }  
 }  
 }  
 }  
  
 if(i<*allEventList*.size()){  
 millis = *allEventList*.get(i).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if ((millis > (1000 \* 60 \* 60 \* 24)) && (millis <= 7 \* (1000 \* 60 \* 60 \* 24))) {  
 int index = 0;  
 for (; j< *allEventList*.size();j++) {  
 millis = *allEventList*.get(j).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if (millis > 7 \* (1000 \* 60 \* 60 \* 24))  
 break;  
 if(*stase1* == true && *allEventList*.get(j).getCheckBox() == 1)  
 continue;  
 index++;  
 }  
 if (index != 0) {  
 adapter.addSeparatorItem(new BeanEventInformation("最近七天"));  
 }  
  
 if (*stase2* == true){  
 List<BeanEventInformation> allEventBypro = new ArrayList<>();  
 for (int n = 0; n<*allEventList*.size(); n++){  
 millis = *allEventList*.get(n).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if ((millis > (1000 \* 60 \* 60 \* 24)) && (millis <= 7 \* (1000 \* 60 \* 60 \* 24)) && (!*allEventList*.get(n).getEventdate().toString().equals("Sat Jan 01 00:00:00 GMT+08:00 1")))  
 allEventBypro.add(*allEventList*.get(n));  
 }  
 for (int l = 1; l < allEventBypro.size(); l++) {  
 BeanEventInformation eventchange;  
 int priority = allEventBypro.get(l-1).getEventpriority();  
 int m = l - 1;  
 int k = l;  
 for (; k < allEventBypro.size(); k++) {  
 if (priority < allEventBypro.get(k).getEventpriority()) {  
 priority = allEventBypro.get(k).getEventpriority();  
 m = k;  
 }  
 }  
 eventchange = allEventBypro.get(l-1);  
 allEventBypro.set(l-1, allEventBypro.get(m));  
 allEventBypro.set(m, eventchange);  
 }  
 for(int k = 0; k < allEventBypro.size(); k++){  
 if(*stase1* == true && allEventBypro.get(k).getCheckBox() == 1)  
 continue;  
 adapter.addItem(allEventBypro.get(k));  
 }  
 for (; i < *allEventList*.size(); i++) {  
 millis = *allEventList*.get(i).getEventdate().getTime() - new Date(System.*currentTimeMillis*()).getTime();  
 if (millis > 7 \* (1000 \* 60 \* 60 \* 24)) break;  
 }  
 }  
 else {  
 for (; i < *allEventList*.size(); i++) {  
 millis = *allEventList*.get(i).getEventdate().getTime() - new Date(System.*currentTimeMillis*()).getTime();  
 if (millis > 7 \* (1000 \* 60 \* 60 \* 24)) break;  
 if (*stase1* == true && *allEventList*.get(i).getCheckBox() == 1) continue;  
 adapter.addItem(*allEventList*.get(i));  
 }  
 }  
 }  
 }  
  
 if(i<*allEventList*.size()){  
 millis = *allEventList*.get(i).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if (millis > 7 \* (1000 \* 60 \* 60 \* 24)){  
 int index = 0;  
 for (; j< *allEventList*.size();j++) {  
 if(*stase1* == true && *allEventList*.get(j).getCheckBox() == 1)  
 continue;  
 index++;  
 }  
 if (index != 0) {  
 adapter.addSeparatorItem(new BeanEventInformation("更远"));  
 }  
  
 if (*stase2* == true){  
 List<BeanEventInformation> allEventBypro = new ArrayList<>();  
 for (int n = 0; n<*allEventList*.size(); n++){  
 millis = *allEventList*.get(n).getEventdate().getTime()-new Date(System.*currentTimeMillis*()).getTime();  
 if ((millis > 7 \* (1000 \* 60 \* 60 \* 24)) && (!*allEventList*.get(n).getEventdate().toString().equals("Sat Jan 01 00:00:00 GMT+08:00 1")))  
 allEventBypro.add(*allEventList*.get(n));  
 }  
 for (int l = 1; l < allEventBypro.size(); l++) {  
 BeanEventInformation eventchange;  
 int priority = allEventBypro.get(l-1).getEventpriority();  
 int m = l - 1;  
 int k = l;  
 for (; k < allEventBypro.size(); k++) {  
 if (priority < allEventBypro.get(k).getEventpriority()) {  
 priority = allEventBypro.get(k).getEventpriority();  
 m = k;  
 }  
 }  
 eventchange = allEventBypro.get(l-1);  
 allEventBypro.set(l-1, allEventBypro.get(m));  
 allEventBypro.set(m, eventchange);  
 }  
 for(int k = 0; k < allEventBypro.size(); k++){  
 if(*stase1* == true && allEventBypro.get(k).getCheckBox() == 1)  
 continue;  
 adapter.addItem(allEventBypro.get(k));  
 }  
 }  
 else {  
 for (; i< *allEventList*.size();i++) {  
 if(*stase1* == true && *allEventList*.get(i).getCheckBox() == 1) continue;  
 adapter.addItem(*allEventList*.get(i));  
 }  
 }  
  
 }  
 }  
  
 adapter.setOnInnerItemOnClickListener(this);  
 listView.setOnItemClickListener(this);  
 }  
  
 public static void initReflesh() {  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* LoadAllEvent loadAllEvent = new LoadAllEvent();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 loadAllEvent.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid());  
  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.app.AlarmManager;  
import android.app.PendingIntent;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.os.Build;  
import android.os.Bundle;  
import android.support.annotation.NonNull;  
import android.support.annotation.RequiresApi;  
import android.support.v4.app.Fragment;  
import android.view.LayoutInflater;  
import android.view.MenuItem;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.AdapterView;  
import android.widget.Button;  
import android.widget.ListView;  
import android.widget.PopupMenu;  
import android.widget.Toast;  
  
import com.example.administrator.timenote.Manager.DremindManager.LoadDremind;  
import com.example.administrator.timenote.Manager.TaskManager.Complete;  
import com.example.administrator.timenote.Manager.TaskManager.LoadAllEvent;  
import com.example.administrator.timenote.Manager.TaskManager.LoadAllEventByPro;  
import com.example.administrator.timenote.Manager.TaskManager.UnComplete;  
import com.example.administrator.timenote.Manager.TaskManager.UpdatePid;  
import com.example.administrator.timenote.Manager.TaskManager.UpdatePriority;  
import com.example.administrator.timenote.Model.BeanDRemindInformation;  
import com.example.administrator.timenote.Model.BeanEventInformation;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.Model.task;  
import com.example.administrator.timenote.R;  
import com.prolificinteractive.materialcalendarview.CalendarDay;  
import com.prolificinteractive.materialcalendarview.CalendarMode;  
import com.prolificinteractive.materialcalendarview.MaterialCalendarView;  
import com.prolificinteractive.materialcalendarview.OnDateSelectedListener;  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.List;  
import java.util.UUID;  
  
import static com.example.administrator.timenote.Model.BeanEventInformation.*allEventList*;  
  
public class page2 extends Fragment implements OnDateSelectedListener, AdapterView.OnItemClickListener, ListAdapter.InnerItemOnclickListener{  
 private Button calendar\_mode; // 日历模式  
 private Calendar calendar;// 获取当前时间用  
 private LayoutInflater inflater1;  
 private ListView listView;// 事务列表  
 private Button list1; // 菜单按钮  
 private Level\_select level\_select;// 优先级选择界面  
 private MenuItem gMenuItem1, gMenuItem2;// list2的两个按钮  
 private Button new\_button;// 新建事务  
 private static boolean *stase1* = true, *stase2* = false;// 列表显示状态  
 private Button taday\_1;// 回到今天按钮  
 private MaterialCalendarView widget;// 日历视图接口  
 private Task\_Update task\_update;// 修改事务  
 private int position;  
 private int ringpid;  
 private int eventidget;  
  
 public View onCreateView(final LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
 this.inflater1 = inflater;  
 View view = inflater.inflate(R.layout.*page2*, container, false);  
 widget=view.findViewById(R.id.*calendarView*);  
 calendar\_mode=view.findViewById(R.id.*calendar\_mode*);  
 new\_button=view.findViewById(R.id.*new\_task1*);  
 listView = (ListView) view.findViewById(R.id.*list\_view\_2*);  
  
 //日历设置为今天  
 calendar = Calendar.*getInstance*();  
 widget.setSelectedDate(calendar.getTime());  
  
  
 //初始化事务列表适配器  
  
 inittask(calendar.getTime());  
  
 //顶层菜单栏  
 //日历视图模式转换  
 calendar\_mode.setOnClickListener(new View.OnClickListener() {  
 @RequiresApi(api = Build.VERSION\_CODES.*JELLY\_BEAN\_MR2*)  
 @Override  
 public void onClick(View view) {  
 if(calendar\_mode.getText().equals("月视图")) {  
 widget.state().edit()  
// .setFirstDayOfWeek(Calendar.WEDNESDAY)//日历开始于周三  
// .setMinimumDate(CalendarDay.from(2016, 4, 3))//日历开始  
// .setMaximumDate(CalendarDay.from(2016, 5, 12))//日历结束  
 .setCalendarDisplayMode(CalendarMode.*WEEKS*)  
 .commit();  
 calendar\_mode.setText("周视图");  
 }  
 else{  
 widget.state().edit()  
 .setCalendarDisplayMode(CalendarMode.*MONTHS*)  
 .commit();  
 calendar\_mode.setText("月视图");  
  
 }  
  
 }  
 });  
  
 widget.setOnDateChangedListener(this);  
  
 //回到今天按钮  
 taday\_1=view.findViewById(R.id.*taday\_2*);  
 taday\_1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 calendar = Calendar.*getInstance*();  
 widget.setSelectedDate(calendar.getTime());  
 if(calendar\_mode.getText().equals("周视图")) {  
 widget.state().edit()  
 .setCalendarDisplayMode(CalendarMode.*WEEKS*)  
 .commit();  
 calendar\_mode.setText("周视图");  
 inittask(calendar.getTime());  
 }  
 else{  
 widget.state().edit()  
 .setCalendarDisplayMode(CalendarMode.*MONTHS*)  
 .commit();  
 calendar\_mode.setText("月视图");  
 inittask(calendar.getTime());  
 }  
 //更新事务列表  
 }  
 });  
  
 //模式菜单按钮  
 list1 = view.findViewById(R.id.*list\_spot\_2*);  
 list1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 showPopupMenu(list1);  
 }  
 });  
  
 //新建事务逻辑  
 new\_button.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View view) {  
 final New\_task new\_task= new New\_task(inflater.getContext(), R.style.*dialog*, calendar);  
 new\_task.show();  
 new\_task.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @RequiresApi(api = Build.VERSION\_CODES.*KITKAT*)  
 @Override  
 public void onDismiss(DialogInterface dialogInterface) {  
 if (new\_task.issure()) {  
 page1.*initReflesh*();  
 inittask(calendar.getTime());  
 if (new\_task.getdate != null) {  
 Date date = null;  
 SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm");  
 try {  
 date = formatter.parse(new\_task.getdate);  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 int eventid = 0;  
 for (int i = 0; i < BeanEventInformation.*allEventList*.size(); i++) {  
 if (eventid < BeanEventInformation.*allEventList*.get(i).getEventid())  
 eventid = BeanEventInformation.*allEventList*.get(i).getEventid();  
 }  
  
 setAlarm(date, BeanDRemindInformation.*defaultset*.getDremindring(),  
 Boolean.*valueOf*(BeanDRemindInformation.*defaultset*.getDremindvib()),  
 BeanDRemindInformation.*defaultset*.getAheadtime(),  
 BeanDRemindInformation.*defaultset*.getDremindrepeat(),  
 eventid, new\_task.scode, new\_task.getdate);  
 }  
 }  
  
 }  
 });  
 }  
 });  
  
 return view;  
 }  
  
 @Override  
 public void onDateSelected(@NonNull MaterialCalendarView widget, @NonNull CalendarDay date, boolean selected) {  
 if(date == null) {  
 //默认跳到今天  
 calendar = Calendar.*getInstance*();  
 widget.setSelectedDate(calendar.getTime());  
 //更新任务列表  
 page1.*initReflesh*();  
 inittask(calendar.getTime());  
 }  
 else {  
 //更新任务列表  
 page1.*initReflesh*();  
 calendar.setTime(date.getDate());  
 inittask(date.getDate());  
 }  
 }  
 private void showPopupMenu(View view) {  
 // View当前PopupMenu显示的相对View的位置  
 PopupMenu popupMenu = new PopupMenu(this.inflater1.getContext(), view);  
 // menu布局  
 popupMenu.getMenuInflater().inflate(R.menu.*showstyle*, popupMenu.getMenu());  
 gMenuItem1 = popupMenu.getMenu().findItem(R.id.*complete\_1*);  
 if (*stase1* == true) {  
 gMenuItem1.setTitle("显示已完成");  
 }  
 gMenuItem2 = popupMenu.getMenu().findItem(R.id.*priorit\_1*);  
 if (*stase2* == false) {  
 gMenuItem2.setTitle("按优先级排序");  
 }  
 // menu的item点击事件  
 popupMenu.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {  
 @Override  
 public boolean onMenuItemClick(MenuItem item) {  
 switch (item.getItemId()) {  
 case R.id.*complete\_1*:  
 if (gMenuItem1.getTitle().equals("显示已完成")) {  
 *stase1* = false;  
 //修改状态重新加载事务列表  
 inittask(calendar.getTime());  
 } else {  
 *stase1* = true;  
 //修改状态重新加载事务列表  
 inittask(calendar.getTime());  
 }  
 break;  
 case R.id.*priorit\_1*:  
 if (gMenuItem2.getTitle().equals("按时间排序")) {  
 *stase2* = false;  
 //修改状态重新加载事务列表  
  
 inittask(calendar.getTime());  
 } else {  
 *stase2* = true;  
 //修改状态重新加载事务列表  
  
 inittask(calendar.getTime());  
 }  
 break;  
 default:  
 }  
 return true;  
 }  
 });  
 // PopupMenu关闭事件  
 popupMenu.setOnDismissListener(new PopupMenu.OnDismissListener() {  
 public void onDismiss(PopupMenu menu) {  
 }  
 });  
 popupMenu.show();  
 }  
  
 @Override  
 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
// Toast.makeText(getContext(),"列表 "+position+" 被点击了",Toast.LENGTH\_SHORT).show();  
 }  
  
 @Override  
 public void itemClick(View v) {  
 position = (Integer) v.getTag();  
 switch (v.getId()) {  
 case R.id.*taskdes*:  
 case R.id.*data\_1*:  
 case R.id.*taskname*:  
  
 task\_update = new Task\_Update(getContext(),R.style.*dialog1*,  
 MyCustomAdapter.*getData*().get(position).getEventid(),  
 MyCustomAdapter.*getData*().get(position).getEventname(),  
 MyCustomAdapter.*getData*().get(position).getEventnote(),  
 MyCustomAdapter.*getData*().get(position).getLeaveseventsign(),  
 MyCustomAdapter.*getData*().get(position).getEventdate());  
 task\_update.show();  
 task\_update.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @RequiresApi(api = Build.VERSION\_CODES.*KITKAT*)  
 @Override  
 public void onDismiss(DialogInterface dialogInterface) {  
 if (task\_update.issure())  
 {  
 int eventid = MyCustomAdapter.*getData*().get(position).getEventid();  
 page1.*initReflesh*();  
 inittask(calendar.getTime());  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* LoadDremind loadDremind = new LoadDremind();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 BeanDRemindInformation.*tsset* = loadDremind.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),MyCustomAdapter.*getData*().get(position).getEventid());  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 int pos = 0;  
 for (int i = 0; i < *allEventList*.size(); i++){  
 if (*allEventList*.get(i).getEventid() == eventid){  
 pos = i;  
 break;  
 }  
 }  
 if ( BeanDRemindInformation.*tsset* != null) {  
 updateAlarm(*allEventList*.get(pos).getEventdate(),  
 BeanDRemindInformation.*defaultset*.getDremindring(),  
 Boolean.*valueOf*(BeanDRemindInformation.*tsset*.getDremindvib()),  
 BeanDRemindInformation.*tsset*.getAheadtime(),  
 BeanDRemindInformation.*tsset*.getDremindrepeat(),  
 eventid,  
 *allEventList*.get(pos).getEventname(),  
 *allEventList*.get(pos).getPid(),  
 new SimpleDateFormat("yyyy-MM-dd hh:mm:ss").format(*allEventList*.get(pos).getEventdate()));  
  
 }  
 else {  
 updateAlarm(*allEventList*.get(pos).getEventdate(),  
 BeanDRemindInformation.*defaultset*.getDremindring(),  
 Boolean.*valueOf*(BeanDRemindInformation.*defaultset*.getDremindvib()),  
 BeanDRemindInformation.*defaultset*.getAheadtime(),  
 BeanDRemindInformation.*defaultset*.getDremindrepeat(),  
 eventid,  
 *allEventList*.get(pos).getEventname(),  
 *allEventList*.get(pos).getPid(),  
 new SimpleDateFormat("yyyy-MM-dd hh:mm:ss").format(*allEventList*.get(pos).getEventdate()));  
 }  
 }  
 }  
 });  
 //Toast.makeText(getContext(), "an", Toast.LENGTH\_SHORT).show();  
 break;  
 // 修改优先级  
 case R.id.*image\_level*:  
 level\_select=new Level\_select(getContext(),R.style.*dialog*);  
 level\_select.show();  
 // 监听弹框消失  
 level\_select.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialog) {  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UpdatePriority updatePriority = new UpdatePriority();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 updatePriority.getRemoteInfo(MyCustomAdapter2.*getData*().get(position).getEventid(), Level\_select.*getLevel*());  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 // 刷新  
 inittask(calendar.getTime());  
 }  
 });  
 break;  
 case R.id.*checkBox*:  
 // 设置已完成  
 if(MyCustomAdapter.*getData*().get(position).getCheckBox()==1)  
 {  
 // *TODO: 2018/6/17 修改为未完成* Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UnComplete unComplete = new UnComplete();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 unComplete.getRemoteInfo(MyCustomAdapter.*getData*().get(position).getEventid());  
 int i = 0;  
 for (; i < *allEventList*.size(); i++){  
 if (*allEventList*.get(i).getEventid() == MyCustomAdapter.*getData*().get(position).getEventid())  
 break;  
 }  
 *allEventList*.get(i).setCheckBox(0);  
  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 // 刷新  
 page1.*initReflesh*();  
 inittask(calendar.getTime());  
 }  
 else{  
 // *TODO: 2018/6/17 修改为已完成* Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* Complete complete = new Complete();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 complete.getRemoteInfo(MyCustomAdapter.*getData*().get(position).getEventid());  
 int i = 0;  
 for (; i < *allEventList*.size(); i++){  
 if (*allEventList*.get(i).getEventid() == MyCustomAdapter.*getData*().get(position).getEventid())  
 break;  
 }  
 *allEventList*.get(i).setCheckBox(1);  
  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 page1.*initReflesh*();  
 inittask(calendar.getTime());  
 }  
 default:  
 break;  
 }  
 }  
  
 // 列表装配  
 private void inittask(Date date)  
 {  
 MyCustomAdapter2 adapter = new MyCustomAdapter2(this.inflater1.getContext());  
  
 java.text.SimpleDateFormat formatter = new SimpleDateFormat( "yyyy-MM-dd HH:mm:ss");  
 String datetype = formatter.format(date);//格式化数据  
  
 int j = 0;  
 if(j<*allEventList*.size()) {  
 Calendar calendar1 = Calendar.*getInstance*();  
 Calendar calendar2 = Calendar.*getInstance*();  
 calendar1.setTime(date);  
 calendar2.setTime(*allEventList*.get(j).getEventdate());  
  
 listView.setAdapter(adapter);  
 int index = 0;  
 for (int i = 0; i < *allEventList*.size(); i++) {  
 calendar2.setTime(*allEventList*.get(i).getEventdate());  
 if (!((calendar1.get(Calendar.*YEAR*) == calendar1.get(Calendar.*YEAR*)) && (calendar1.get(Calendar.*MONTH*) == calendar2.get(Calendar.*MONTH*)) && (calendar1.get(Calendar.*DAY\_OF\_MONTH*) == calendar2.get(Calendar.*DAY\_OF\_MONTH*))))  
 continue;  
 if (*stase1* && *allEventList*.get(i).getCheckBox() == 1)  
 continue;  
 index++;  
 }  
 if (index != 0) {  
 adapter.addSeparatorItem(new BeanEventInformation(datetype));  
 }  
  
 if (*stase2*) {  
 List<BeanEventInformation> allEventBypro = new ArrayList<>();  
 for (int n = 0; n < *allEventList*.size(); n++) {  
 calendar2.setTime(*allEventList*.get(n).getEventdate());  
 if ((calendar1.get(Calendar.*YEAR*) == calendar1.get(Calendar.*YEAR*)) && (calendar1.get(Calendar.*MONTH*) == calendar2.get(Calendar.*MONTH*)) && (calendar1.get(Calendar.*DAY\_OF\_MONTH*) == calendar2.get(Calendar.*DAY\_OF\_MONTH*)))  
 allEventBypro.add(*allEventList*.get(n));  
 }  
 for (int l = 1; l < allEventBypro.size(); l++) {  
 BeanEventInformation eventchange;  
 int priority = allEventBypro.get(l - 1).getEventpriority();  
 int m = l - 1;  
 int k = l;  
 for (; k < allEventBypro.size(); k++) {  
 if (priority < allEventBypro.get(k).getEventpriority()) {  
 priority = allEventBypro.get(k).getEventpriority();  
 m = k;  
 }  
 }  
 eventchange = allEventBypro.get(l - 1);  
 allEventBypro.set(l - 1, allEventBypro.get(m));  
 allEventBypro.set(m, eventchange);  
 }  
 for (int k = 0; k < allEventBypro.size(); k++) {  
 if (*stase1* && allEventBypro.get(k).getCheckBox() == 1)  
 continue;  
 adapter.addItem(allEventBypro.get(k));  
 }  
 } else {  
 for (; j < *allEventList*.size(); j++) {  
 if (*stase1* && *allEventList*.get(j).getCheckBox() == 1)  
 continue;  
 calendar2.setTime(*allEventList*.get(j).getEventdate());  
 if ((calendar1.get(Calendar.*YEAR*) == calendar1.get(Calendar.*YEAR*)) && (calendar1.get(Calendar.*MONTH*) == calendar2.get(Calendar.*MONTH*)) && (calendar1.get(Calendar.*DAY\_OF\_MONTH*) == calendar2.get(Calendar.*DAY\_OF\_MONTH*)))  
 adapter.addItem(*allEventList*.get(j));  
 }  
 }  
 }  
  
 adapter.setOnInnerItemOnClickListener(this);  
 listView.setOnItemClickListener(this);  
 }  
 @RequiresApi(api = Build.VERSION\_CODES.*KITKAT*)  
 private void setAlarm(Date date, int ring, Boolean vib, long ahead, int repeat, int eventid, String eventname, String eventdate){  
 Calendar calendar = Calendar.*getInstance*();  
 if (calendar.getTimeInMillis() > date.getTime()){  
 return;  
 }  
 eventidget = eventid;  
 calendar.setTime(date);  
 long mils = 0;  
 switch (BeanDRemindInformation.*tsset*.getAheadtime()) {  
 case 0:  
 break;  
 case 1:  
 mils = 1000 \* 60 \* 5;  
 break;  
 case 2:  
 mils = 1000 \* 60 \* 30;  
 break;  
 case 3:  
 mils = 1000 \* 60 \* 60;  
 break;  
 case 4:  
 mils = 1000 \* 60 \* 60 \* 24;  
 break;  
 }  
 //跳转  
 Intent intent=new Intent();  
 ringpid = UUID.*randomUUID*().hashCode();  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UpdatePid updatePid = new UpdatePid();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 updatePid.getRemoteInfo(eventidget, ringpid);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 intent.setAction("com.liuqian.android\_27alarm.RING");  
 intent.putExtra("id", ringpid);  
 intent.putExtra("ring", ring);  
 intent.putExtra("vib", vib);  
 intent.putExtra("repeat", repeat);  
 intent.putExtra("eventid", eventid);  
 intent.putExtra("eventname", eventname);  
 intent.putExtra("eventdate", eventdate);  
 //PendingIntent  
 PendingIntent pendingIntent= PendingIntent.*getBroadcast*(getContext(),ringpid,intent,0);  
 //闹钟管理者 参数： 唤醒屏幕，  
 Calendar\_View.*alarmManager*.setExact(AlarmManager.*RTC\_WAKEUP*, calendar.getTimeInMillis() - mils, pendingIntent);  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.*KITKAT*)  
 private void updateAlarm(Date date, int ring, Boolean vib, int ahead, int repeat, int eventid, String eventname, int pid, String eventdate){  
 Calendar calendar = Calendar.*getInstance*();  
 if (calendar.getTimeInMillis() > date.getTime()){  
 return;  
 }  
 eventidget = eventid;  
 calendar.setTime(date);  
 long mils = 0;  
 switch (ahead){  
 case 0:break;  
 case 1:mils = 1000\*60\*5;break;  
 case 2:mils = 1000\*60\*30;break;  
 case 3:mils = 1000\*60\*60;break;  
 case 4:mils = 1000\*60\*60\*24;break;  
 }  
 //跳转  
 Intent intent=new Intent();  
 if (pid == -99){  
 ringpid = UUID.*randomUUID*().hashCode();  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UpdatePid updatePid = new UpdatePid();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 updatePid.getRemoteInfo(eventidget, ringpid);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 else  
 ringpid = pid;  
 intent.setAction("com.liuqian.android\_27alarm.RING");  
 intent.putExtra("id", ringpid);  
 intent.putExtra("ring", ring);  
 intent.putExtra("vib", vib);  
 intent.putExtra("ahead", ahead);  
 intent.putExtra("repeat", repeat);  
 intent.putExtra("eventid", eventid);  
 intent.putExtra("eventname", eventname);  
 intent.putExtra("eventdate", eventdate);  
 //PendingIntent  
 PendingIntent pendingIntent= PendingIntent.*getBroadcast*(getContext(),ringpid,intent,0);  
 //闹钟管理者 参数： 唤醒屏幕，  
 Calendar\_View.*alarmManager*.setExact(AlarmManager.*RTC\_WAKEUP*, calendar.getTimeInMillis() - mils, pendingIntent);  
 }  
  
}

package com.example.administrator.timenote.Ui;  
  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.os.Bundle;  
import android.support.v4.app.Fragment;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.example.administrator.timenote.Manager.LeavesManager.LoadLeaves;  
import com.example.administrator.timenote.Model.BeanLeavesStatistics;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
  
import java.util.Calendar;  
  
public class page3 extends Fragment {  
  
 private Button tasklist;//事务选择列表  
 private Button start;// 开始专注按钮  
 private TextView tasktext1;// 提示文本大字  
 private TextView tasktext2;// 提示文本小字  
 private Task\_select\_list task\_select\_list;// 事务选择  
 private String tishi;  
  
 public View onCreateView(final LayoutInflater inflater, final ViewGroup container, Bundle savedInstanceState) {  
  
 View view = inflater.inflate(R.layout.*page3*, container, false);  
 tasklist=view.findViewById(R.id.*task\_list\_yezi*);  
 start=view.findViewById(R.id.*start\_task*);  
 tasktext1=view.findViewById(R.id.*tasktext1*);  
 tasktext2=view.findViewById(R.id.*tasktext2*);  
  
 // 初始化  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* LoadLeaves loadLeaves = new LoadLeaves();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 loadLeaves.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid());  
 BeanLeavesStatistics.*todayleaves* = BeanLeavesStatistics.*leavesStatistics*;  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 Calendar calendar = Calendar.*getInstance*();  
 if (calendar.get(Calendar.*HOUR\_OF\_DAY*) < 4 || calendar.get(Calendar.*HOUR\_OF\_DAY*) >= 22)  
 tishi = "夜深了，早点休息";  
 else if (calendar.get(Calendar.*HOUR\_OF\_DAY*) < 7)  
 tishi = "早上好";  
 else if (calendar.get(Calendar.*HOUR\_OF\_DAY*) < 11)  
 tishi = "上午好";  
 else if (calendar.get(Calendar.*HOUR\_OF\_DAY*) < 13)  
 tishi = "中午了，休息一下";  
 else if (calendar.get(Calendar.*HOUR\_OF\_DAY*) < 6)  
 tishi = "下午也要加油";  
 else if (calendar.get(Calendar.*HOUR\_OF\_DAY*) < 7)  
 tishi = "是时候放松一下了";  
 else if (calendar.get(Calendar.*HOUR\_OF\_DAY*) < 22)  
 tishi = "晚上好";  
 tasktext1.setText(tishi);  
 if (BeanLeavesStatistics.*todayleaves*.getLeavesamount() != 0)  
 tasktext2.setText("今天您已收获"+String.*valueOf*(BeanLeavesStatistics.*todayleaves*.getLeavesamount())+"叶子");  
  
 //事务选择列表按钮监听  
 tasklist.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 task\_select\_list=new Task\_select\_list(inflater.getContext(),R.style.*dialog*);  
 task\_select\_list.show();  
 task\_select\_list.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialogInterface) {  
 if(task\_select\_list.isIssure())  
 {  
 Intent intent = new Intent(getContext(),Yezi\_start.class);  
 intent.putExtra("p",task\_select\_list.getNposition());  
 startActivity(intent);  
 }  
  
 }  
 });  
 }  
 });  
  
 //开始按钮监听  
 start.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Intent intent = new Intent(getContext(),Yezi\_start.class);  
 startActivity(intent);  
 }  
 });  
  
 return view;  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.annotation.SuppressLint;  
import android.app.Dialog;  
import android.app.Notification;  
import android.app.NotificationManager;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.media.Image;  
import android.media.MediaPlayer;  
import android.os.Bundle;  
import android.os.CountDownTimer;  
import android.support.v4.app.Fragment;  
import android.support.v4.app.NotificationCompat;  
import android.util.Log;  
import android.view.Gravity;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.view.Window;  
import android.widget.AdapterView;  
import android.widget.ArrayAdapter;  
import android.widget.Button;  
import android.widget.CompoundButton;  
import android.widget.FrameLayout;  
import android.widget.ImageView;  
import android.widget.Spinner;  
import android.widget.Switch;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.bigkoo.pickerview.builder.TimePickerBuilder;  
import com.bigkoo.pickerview.listener.CustomListener;  
import com.bigkoo.pickerview.listener.OnDismissListener;  
import com.bigkoo.pickerview.listener.OnTimeSelectChangeListener;  
import com.bigkoo.pickerview.listener.OnTimeSelectListener;  
import com.bigkoo.pickerview.view.TimePickerView;  
import com.example.administrator.timenote.Manager.DremindManager.ChangeDRTime;  
import com.example.administrator.timenote.Manager.DremindManager.ChangeLeavesTime;  
import com.example.administrator.timenote.Manager.DremindManager.ChangeRepeat;  
import com.example.administrator.timenote.Manager.DremindManager.ChangeRing;  
import com.example.administrator.timenote.Manager.DremindManager.ChangeVib;  
import com.example.administrator.timenote.Manager.DremindManager.LoadDremind;  
import com.example.administrator.timenote.Model.BeanDRemindInformation;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
import com.example.administrator.timenote.Ring.RingActivity;  
  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.Date;  
  
public class page4 extends Fragment {  
  
 private ArrayAdapter adapter\_time = null;// 时间下拉列表配置器  
 private ArrayAdapter adapter\_even = null;// 频率下拉列表配置器  
 private ArrayAdapter adapter\_ring =null;// 铃声下拉列表配置器  
 public static ImageView *imageButton1*; // 头像按钮  
 private TimePickerView pvTime;// 时间自定义界面  
 private Switch switch1;// 振动按钮  
 private Spinner time\_select\_1; // 提醒时间下拉列表  
 private Spinner time\_ring\_1;// 提醒闹铃下拉列表  
 private Spinner time\_even\_1;// 时间  
 private TextView time\_text;// 自定义时间提醒框  
 private TextView time\_even\_text;// 自定义频率  
 private static TextView *user\_name\_page4*;// 用户名显示  
 private Button user\_psonal\_button;// 个人中心  
 private TextView yezi\_time;// 叶子时长  
 private TextView yezi\_2;// 设置叶子时长  
 private yezi\_time\_select yezi\_time\_select1;//设置叶子时长界面  
 private int time\_p;// 定义时间位置  
 private String time\_get;// 获取时间  
 private MediaPlayer mediaPlayer;  
  
 public static void setUser\_name\_page4(String user\_name\_page4) {  
 page4.*user\_name\_page4*.setText(user\_name\_page4);  
 }  
  
 public View onCreateView(LayoutInflater inflater, final ViewGroup container, Bundle savedInstanceState) {  
  
 View view = inflater.inflate(R.layout.*page4*, container, false);  
  
 // 对应控件  
 *imageButton1* = view.findViewById(R.id.*imageButton*); // 头像按钮  
 switch1 = view.findViewById(R.id.*switch1*);// 振动开关  
 time\_select\_1 = view.findViewById(R.id.*time\_select\_1*);// 时间选择  
 time\_ring\_1 = view.findViewById(R.id.*time\_ring\_1*);// 闹铃选择  
 time\_even\_1 = view.findViewById(R.id.*time\_even\_1*);// 重复选择  
 time\_text = view.findViewById(R.id.*time\_text\_1*);// 自定义时间显示  
 //time\_even\_text = view.findViewById(R.id.retime\_text\_1);// 重复选择存储文本  
 *user\_name\_page4* = view.findViewById(R.id.*user\_name\_page4*);// 用户名文本  
 user\_psonal\_button = view.findViewById(R.id.*user\_psonal\_button*);//  
 yezi\_time = view.findViewById(R.id.*yezi\_time*);// 叶子时间选择  
 yezi\_2 = view.findViewById(R.id.*yezi\_2*);// 叶子时间显示  
  
 initloaddefault();  
  
 // 初始叶子时长  
 yezi\_time.setText(String.*valueOf*(BeanDRemindInformation.*defaultset*.getLeavestime()) + "分钟");  
  
 // 初始默认提醒时间  
 switch (BeanDRemindInformation.*defaultset*.getDefaulttime()){  
 case "0":time\_p = 0;break;  
 case "1":time\_p = 1;break;  
 case "2":time\_p = 2;break;  
 case "3":time\_p = 3;break;  
 case "4":time\_p = 4;break;  
 case "5":time\_p = 5;break;  
 case "6":time\_p = 6;break;  
 case "7":time\_p = 7;break;  
 case "8":time\_p = 8;break;  
 case "9":time\_p = 9;break;  
 case "10":time\_p = 10;break;  
 case "11":time\_p = 11;break;  
 default:time\_p = 12;break;  
 }  
  
 // 初始震动状态  
 if (BeanDRemindInformation.*defaultset*.getDremindvib().equals("true"))  
 switch1.setChecked(true);  
 else  
 switch1.setChecked(false);  
  
 // 显示用户名  
 if(BeanUserInformation.*currentLoginUser*.getIcon()!=null)  
 {  
 Bitmap bm;  
 byte[] bitmapByte = BeanUserInformation.*currentLoginUser*.getIcon();  
 if (bitmapByte.length != 0) {  
 bm = BitmapFactory.*decodeByteArray*(bitmapByte, 0,bitmapByte.length);  
 *imageButton1*.setImageBitmap(bm);  
 } else {  
  
 }  
 }  
 *user\_name\_page4*.setText(BeanUserInformation.*currentLoginUser*.getUsername());  
  
 // 头像按钮  
 *imageButton1*.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent1 = new Intent(getContext(),User\_information.class);  
 startActivity(intent1);  
 }  
 });  
  
 //振动按钮  
 switch1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {  
 @Override  
 public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {  
 if (isChecked){  
 Toast.*makeText*(getContext(),"开启振动",Toast.*LENGTH\_SHORT*).show();  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* ChangeVib changeVib = new ChangeVib();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 changeVib.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),-99,1);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 else {  
 Toast.*makeText*(getContext(),"关闭振动",Toast.*LENGTH\_SHORT*).show();  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* ChangeVib changeVib = new ChangeVib();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 changeVib.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),-99,0);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 });  
  
 //叶子时长选择  
 yezi\_2.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 yezi\_time\_select1 =new yezi\_time\_select(getContext(),R.style.*dialog*);  
 yezi\_time\_select1.show();  
 yezi\_time\_select1.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @SuppressLint("SetTextI18n")  
 @Override  
 public void onDismiss(DialogInterface dialog) {  
 if(yezi\_time\_select1.getYezi\_time()!=null) {  
 yezi\_time.setText(yezi\_time\_select1.getYezi\_time() + "分钟");  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* ChangeLeavesTime changeLeavesTime = new ChangeLeavesTime();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 changeLeavesTime.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),-99,Integer.*parseInt*(yezi\_time\_select1.getYezi\_time()));  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 initloaddefault();  
 Yezi\_start.*setM*(Integer.*parseInt*(yezi\_time\_select1.getYezi\_time()));  
 }  
 }  
 });  
 }  
 });  
 yezi\_time.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 yezi\_time\_select1 =new yezi\_time\_select(getContext(),R.style.*dialog*);  
 yezi\_time\_select1.show();  
 yezi\_time\_select1.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @SuppressLint("SetTextI18n")  
 @Override  
 public void onDismiss(DialogInterface dialog) {  
 if(yezi\_time\_select1.getYezi\_time()!=null) {  
 yezi\_time.setText(yezi\_time\_select1.getYezi\_time() + "分钟");  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* ChangeLeavesTime changeLeavesTime = new ChangeLeavesTime();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 changeLeavesTime.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(), -99, Integer.*parseInt*(yezi\_time\_select1.getYezi\_time()));  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 initloaddefault();  
 Yezi\_start.*setM*(Integer.*parseInt*(yezi\_time\_select1.getYezi\_time()));  
 }  
 }  
 });  
 }  
 });  
  
 // 进入个人中心  
 user\_psonal\_button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(getContext(),Pesonal.class);  
 startActivity(intent);  
 }  
 });  
  
 // 重复选择列表  
 adapter\_even = ArrayAdapter.*createFromResource*(getContext(),R.array.*time\_even*,android.R.layout.*simple\_spinner\_item*);  
 adapter\_even.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);  
 time\_even\_1.setAdapter(adapter\_even);  
 // 初始重复  
 time\_even\_1.setSelection(BeanDRemindInformation.*defaultset*.getDremindrepeat(),true);  
 time\_even\_1.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
 @Override  
 public void onItemSelected(AdapterView<?> parent, View view, final int position, long id) {  
  
 // 更新设置  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* ChangeRepeat changeRepeat = new ChangeRepeat();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 changeRepeat.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),-99, position);  
  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 initloaddefault();  
 }  
  
 @Override  
 public void onNothingSelected(AdapterView<?> parent) {  
  
 }  
 });  
  
 // 铃声选择列表  
 adapter\_ring = ArrayAdapter.*createFromResource*(getContext(),R.array.*time\_ring*,android.R.layout.*simple\_spinner\_item*);  
 adapter\_ring.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);  
 time\_ring\_1.setAdapter(adapter\_ring);  
 // 初始铃声  
 time\_ring\_1.setSelection(BeanDRemindInformation.*defaultset*.getDremindring(),true);  
 time\_ring\_1.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
 @Override  
 public void onItemSelected(AdapterView<?> parent, View view, final int position, long id) {  
 // 试听  
 if (BeanDRemindInformation.*defaultset*.getDremindring() != position) {  
 switch (position) {  
 case 0:  
 if (mediaPlayer != null) {  
 if (mediaPlayer.isPlaying()) {  
 mediaPlayer.stop();  
 mediaPlayer.release();  
 }  
 cdt.cancel();  
 }  
 mediaPlayer = new MediaPlayer();  
 break;  
 case 1:  
 if (mediaPlayer != null) {  
 if (mediaPlayer.isPlaying()) {  
 mediaPlayer.stop();  
 mediaPlayer.release();  
 }  
 cdt.cancel();  
 }  
 mediaPlayer = MediaPlayer.*create*(getContext(), R.raw.*one*);  
 mediaPlayer.setLooping(true);  
 mediaPlayer.start();  
 break;  
 case 2:  
 if (mediaPlayer != null) {  
 if (mediaPlayer.isPlaying()) {  
 mediaPlayer.stop();  
 mediaPlayer.release();  
 }  
 cdt.cancel();  
 }  
 mediaPlayer = MediaPlayer.*create*(getContext(), R.raw.*two*);  
 mediaPlayer.setLooping(true);  
 mediaPlayer.start();  
 break;  
 case 3:  
 if (mediaPlayer != null) {  
 if (mediaPlayer.isPlaying()) {  
 mediaPlayer.stop();  
 mediaPlayer.release();  
 }  
 cdt.cancel();  
 }  
 mediaPlayer = MediaPlayer.*create*(getContext(), R.raw.*three*);  
 mediaPlayer.setLooping(true);  
 mediaPlayer.start();  
 break;  
 case 4:  
 if (mediaPlayer != null) {  
 if (mediaPlayer.isPlaying()) {  
 mediaPlayer.stop();  
 mediaPlayer.release();  
 }  
 cdt.cancel();  
 }  
 mediaPlayer = MediaPlayer.*create*(getContext(), R.raw.*four*);  
 mediaPlayer.setLooping(true);  
 mediaPlayer.start();  
 break;  
 case 5:  
 if (mediaPlayer != null) {  
 if (mediaPlayer.isPlaying()) {  
 mediaPlayer.stop();  
 mediaPlayer.release();  
 }  
 cdt.cancel();  
 }  
 mediaPlayer = MediaPlayer.*create*(getContext(), R.raw.*five*);  
 mediaPlayer.setLooping(true);  
 mediaPlayer.start();  
 break;  
 case 6:  
 if (mediaPlayer != null) {  
 if (mediaPlayer.isPlaying()) {  
 mediaPlayer.stop();  
 mediaPlayer.release();  
 }  
 cdt.cancel();  
 }  
 mediaPlayer = MediaPlayer.*create*(getContext(), R.raw.*six*);  
 mediaPlayer.setLooping(true);  
 mediaPlayer.start();  
 break;  
 default:  
 break;  
 }  
 cdt.start();  
 }  
  
 // 更新设置  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* ChangeRing changeRing = new ChangeRing();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 changeRing.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),-99, position);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 initloaddefault();  
 }  
  
 @Override  
 public void onNothingSelected(AdapterView<?> parent) {  
  
 }  
 });  
  
 // 时间选择列表  
 adapter\_time = ArrayAdapter.*createFromResource*(getContext(),R.array.*time\_select*,android.R.layout.*simple\_spinner\_item*);  
 adapter\_time.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);  
 time\_select\_1.setAdapter(adapter\_time);  
 // 初始默认提醒时间  
 time\_select\_1.setSelection(time\_p,true);  
 if (time\_p == 12){  
 time\_text.setText(BeanDRemindInformation.*defaultset*.getDefaulttime());  
 }  
 time\_select\_1.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
 @Override  
 public void onItemSelected(AdapterView<?> parent, View view, final int position, long id) {  
 final String cityName = (String) adapter\_time.getItem(position);  
 if(cityName.equals("自定义"))  
 {  
 initCustomTimePicker2();  
 pvTime.show();  
 pvTime.setOnDismissListener(new OnDismissListener() {  
 @SuppressLint("WrongConstant")  
 @Override  
 public void onDismiss(Object o) {  
 if(time\_text.getVisibility()==0)  
 {  
 // 更新设置  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* ChangeDRTime loadDremind = new ChangeDRTime();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 loadDremind.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),-99, time\_get);  
  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 initloaddefault();  
 }  
 }  
 });  
 }  
 else  
 {  
 time\_text.setVisibility(View.*INVISIBLE*);  
 // 更新设置  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* ChangeDRTime loadDremind = new ChangeDRTime();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 loadDremind.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),-99, String.*valueOf*(position));  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 initloaddefault();  
 }  
 }  
  
 @Override  
 public void onNothingSelected(AdapterView<?> parent) {  
  
 }  
 });  
  
 return view;  
 }  
 private void initCustomTimePicker2() {  
  
 Calendar selectedDate = Calendar.*getInstance*();  
 pvTime = new TimePickerBuilder(getContext(), new OnTimeSelectListener() {  
 @Override  
 public void onTimeSelect(Date date, View v) {  
 time\_get = getTime(date);  
 time\_text.setText(getTime(date));  
 time\_text.setVisibility(View.*VISIBLE*);  
 }  
 })  
 .setDate(selectedDate)  
 .setLayoutRes(R.layout.*pickerview\_custom\_time*, new CustomListener() {  
 @Override  
 public void customLayout(View v) {  
 final TextView tvSubmit = (TextView) v.findViewById(R.id.*tv\_finish*);  
 ImageView ivCancel = (ImageView) v.findViewById(R.id.*iv\_cancel*);  
 tvSubmit.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 pvTime.returnData();  
 pvTime.dismiss();  
 }  
 });  
 ivCancel.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 pvTime.dismiss();  
 }  
 });  
 }  
 })  
 .setTimeSelectChangeListener(new OnTimeSelectChangeListener() {  
 @Override  
 public void onTimeSelectChanged(Date date) {  
 Log.*i*("pvTime", "onTimeSelectChanged");  
 }  
 })  
 .setType(new boolean[]{false, false, false, true, true, false})  
 .isDialog(true)  
 .build();  
 Dialog mDialog = pvTime.getDialog();  
 if (mDialog != null) {  
  
 FrameLayout.LayoutParams params = new FrameLayout.LayoutParams(  
 ViewGroup.LayoutParams.*MATCH\_PARENT*,  
 ViewGroup.LayoutParams.*WRAP\_CONTENT*,  
 Gravity.*BOTTOM*);  
  
 params.leftMargin = 0;  
 params.rightMargin = 0;  
 pvTime.getDialogContainerLayout().setLayoutParams(params);  
  
 Window dialogWindow = mDialog.getWindow();  
 if (dialogWindow != null) {  
 dialogWindow.setWindowAnimations(com.bigkoo.pickerview.R.style.*picker\_view\_slide\_anim*);//修改动画样式  
 dialogWindow.setGravity(Gravity.*CENTER*);//改成Bottom,底部显示  
 }  
 }  
 }  
 private String getTime(Date date) {//可根据需要自行截取数据显示  
 SimpleDateFormat format = new SimpleDateFormat("HH:mm");  
 return format.format(date);  
 }  
  
 private void initloaddefault(){  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* LoadDremind loadDremind = new LoadDremind();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 BeanDRemindInformation.*tsset* = loadDremind.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),-99);  
 BeanDRemindInformation.*defaultset* = BeanDRemindInformation.*tsset*;  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 CountDownTimer cdt = new CountDownTimer(10000, 5000) {  
  
 @SuppressLint({"ResourceAsColor", "SetTextI18n"})  
 public void onTick(long millisUntilFinished) {  
 }  
 @SuppressLint("ResourceAsColor")  
 @Override  
 public void onFinish() {  
  
 if (mediaPlayer != null) {  
 if (mediaPlayer.isPlaying()) {  
 mediaPlayer.stop();  
 mediaPlayer.release();  
 }  
 }  
 }  
  
 };  
  
}

package com.example.administrator.timenote.Ui;  
  
import android.content.Intent;  
import android.content.pm.ActivityInfo;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.view.Window;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
  
public class Pesonal extends AppCompatActivity {  
  
 private Button back;// 返回按钮  
 private ImageView imageView3;// 头像按钮  
 private TextView textView20;// 用户名  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 supportRequestWindowFeature(Window.*FEATURE\_NO\_TITLE*);  
 setContentView(R.layout.*activity\_pesonal*);  
 {  
 setRequestedOrientation(ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT*);  
 }  
  
 // 绑定按钮  
 back = findViewById(R.id.*back\_9*);  
 imageView3 = findViewById(R.id.*imageView3*);  
 textView20 = findViewById(R.id.*textView20*);  
  
 if(BeanUserInformation.*currentLoginUser*.getIcon()!=null) {  
 Bitmap bm;  
 byte[] bitmapByte1 = BeanUserInformation.*currentLoginUser*.getIcon();  
 if (bitmapByte1.length != 0) {  
 bm = BitmapFactory.*decodeByteArray*(bitmapByte1, 0, bitmapByte1.length);  
 imageView3.setImageBitmap(bm);  
 }  
 else {  
  
 }  
 }  
 textView20.setText(BeanUserInformation.*currentLoginUser*.getUsername());  
  
 // 返回按钮  
 back.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 finish();  
 }  
 });  
  
 // 头像按钮  
 imageView3.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(Pesonal.this,User\_information.class);  
 startActivity(intent);  
 finish();  
 }  
 });  
 }  
}

package com.example.administrator.timenote.Ui;  
  
  
import com.contrarywind.interfaces.IPickerViewData;  
  
*/\*\*  
 \* Created by Sai on 15/11/22.  
 \*/*public class ProvinceBean implements IPickerViewData {  
 private long id;  
 private String name;  
 private String description;  
 private String others;  
  
 public ProvinceBean(long id, String name, String description, String others){  
 this.id = id;  
 this.name = name;  
 this.description = description;  
 this.others = others;  
 }  
  
 public long getId() {  
 return id;  
 }  
  
 public void setId(long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 public String getOthers() {  
 return others;  
 }  
  
 public void setOthers(String others) {  
 this.others = others;  
 }  
  
 //这个用来显示在PickerView上面的字符串,PickerView会通过getPickerViewText方法获取字符串显示出来。  
 @Override  
 public String getPickerViewText() {  
 return name;  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.app.Dialog;  
import android.content.Context;  
import android.os.Bundle;  
  
import android.view.View;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.AdapterView;  
import android.widget.Button;  
import android.widget.ListView;  
import android.widget.Toast;  
  
import com.example.administrator.timenote.Manager.DremindManager.ChangeRepeat;  
import com.example.administrator.timenote.Manager.DremindManager.LoadDremind;  
import com.example.administrator.timenote.Manager.DremindManager.NewDRemind;  
import com.example.administrator.timenote.Model.BeanDRemindInformation;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
  
import java.util.ArrayList;  
import java.util.List;  
  
  
public class Rering\_setup extends Dialog implements AdapterView.OnItemClickListener, ListAdapter.InnerItemOnclickListener{  
 */\*\*  
 \* 上下文对象 \*  
 \*/* private Button back;  
 private Context context;  
 private ListView listView;// 选择列表  
 private ArrayList<String> list = new ArrayList<>();  
 private int position;  
 private String show;  
// private Boolean redx = true;  
  
 public Rering\_setup(Context context) {  
 super(context);  
 this.context = context;  
 }  
  
 public Rering\_setup(Context context, int theme) {  
 super(context, theme);  
 this.context = context;  
 }  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 // 指定布局  
 this.setContentView(R.layout.*rering\_set*);  
  
 /\*  
 \* 获取验证码的窗口对象及参数对象以修改对话框的布局设置, 可以直接调用getWindow(),表示获得这个Activity的Window  
 \* 对象,这样这可以以同样的方式改变这个Activity的属性.  
 \*/  
 Window dialogWindow = this.getWindow();  
  
// WindowManager m = context.getWindowManager();  
// Display d = m.getDefaultDisplay(); // 获取屏幕宽、高用  
 WindowManager.LayoutParams p = dialogWindow.getAttributes(); // 获取对话框当前的参数值  
// p.height = (int) (d.getHeight() ); // 高度设置为屏幕  
// p.width = (int) (d.getWidth() ); // 宽度设置为屏幕  
 dialogWindow.setAttributes(p);  
  
 // 根据id在布局中找到控件对象  
 back = findViewById(R.id.*back\_14*);  
 listView = findViewById(R.id.*rering\_setup\_list*);  
  
 // 为按钮绑定点击事件监听器  
 back.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View ciew) {  
 dismiss();  
 }  
 });  
  
 list.add("从不");  
 list.add("每天");  
 list.add("每周工作日(周一至周五)");  
 list.add("每周休息日");  
 list.add("每月(当天)");  
 list.add("每年(当天)");  
 ListAdapter listAdapter = new ListAdapter(context,R.layout.*list\_button*,list);  
 listView.setAdapter(listAdapter);  
 listAdapter.setOnInnerItemOnClickListener(this);  
 listView.setOnItemClickListener(this);  
  
  
 this.setCancelable(true);  
 }  
  
  
 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
  
 }  
  
  
 public void itemClick(View v) {  
 position = (Integer) v.getTag();  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* LoadDremind loadDremind = new LoadDremind();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 BeanDRemindInformation.*tsset* = loadDremind.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),Task\_Update.*getEventid*());  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 if (BeanDRemindInformation.*tsset* == null){  
 Thread s = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* NewDRemind newDRemind = new NewDRemind();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 newDRemind.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),Task\_Update.*getEventid*());  
 }  
 });  
 s.start();  
 try {  
 s.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 Thread r = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* ChangeRepeat changeRepeat = new ChangeRepeat();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 changeRepeat.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),Task\_Update.*getEventid*(),position);  
 }  
 });  
 r.start();  
 try {  
 r.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 else {  
 Thread r = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* ChangeRepeat changeRepeat = new ChangeRepeat();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 changeRepeat.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),Task\_Update.*getEventid*(),position);  
 }  
 });  
 r.start();  
 try {  
 r.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 switch (position){  
 case 0:show = "从不";break;  
 case 1:show = "每天";break;  
 case 2:show = "每周工作日(周一至周五)";break;  
 case 3:show = "每周休息日";break;  
 case 4:show = "每月(当天)";break;  
 case 5:show = "每年(当天)";break;  
 }  
 Toast.*makeText*(getContext(),show,Toast.*LENGTH\_SHORT*).show();  
 dismiss();  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.app.Dialog;  
import android.content.Context;  
import android.os.Bundle;  
import android.view.View;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.AdapterView;  
import android.widget.Button;  
import android.widget.ListView;  
import android.widget.Toast;  
  
import com.example.administrator.timenote.Manager.DremindManager.ChangeAhead;  
import com.example.administrator.timenote.Manager.DremindManager.LoadDremind;  
import com.example.administrator.timenote.Manager.DremindManager.NewDRemind;  
import com.example.administrator.timenote.Model.BeanDRemindInformation;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
  
import java.util.ArrayList;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/18.  
 \*/*public class Ringtime\_setup extends Dialog implements AdapterView.OnItemClickListener, ListAdapter.InnerItemOnclickListener{  
 */\*\*  
 \* 上下文对象 \*  
 \*/* private Button back;  
 private Context context;  
 private ListView listView;// 选择列表  
 private ArrayList<String> list = new ArrayList<>();  
 private int position;  
 private String show;  
// private Boolean aheaddx = true;  
  
 public Ringtime\_setup(Context context) {  
 super(context);  
 this.context = context;  
 }  
  
 public Ringtime\_setup(Context context, int theme) {  
 super(context, theme);  
 this.context = context;  
 }  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 // 指定布局  
 this.setContentView(R.layout.*ringtime\_setup*);  
  
 /\*  
 \* 获取验证码的窗口对象及参数对象以修改对话框的布局设置, 可以直接调用getWindow(),表示获得这个Activity的Window  
 \* 对象,这样这可以以同样的方式改变这个Activity的属性.  
 \*/  
 Window dialogWindow = this.getWindow();  
  
// WindowManager m = context.getWindowManager();  
// Display d = m.getDefaultDisplay(); // 获取屏幕宽、高用  
 WindowManager.LayoutParams p = dialogWindow.getAttributes(); // 获取对话框当前的参数值  
// p.height = (int) (d.getHeight() ); // 高度设置为屏幕  
// p.width = (int) (d.getWidth() ); // 宽度设置为屏幕  
 dialogWindow.setAttributes(p);  
  
 // 根据id在布局中找到控件对象  
 back = findViewById(R.id.*back\_18*);  
 listView = findViewById(R.id.*ringtime\_setup\_list*);  
  
 // 为按钮绑定点击事件监听器  
 back.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View ciew) {  
 dismiss();  
 }  
 });  
  
 list.add("准时");  
 list.add("提前 5 分钟");  
 list.add("提前 30 分钟");  
 list.add("提前 1 小时");  
 list.add("提前 1 天");  
 ListAdapter listAdapter = new ListAdapter(context,R.layout.*list\_button*,list);  
 listView.setAdapter(listAdapter);  
 listAdapter.setOnInnerItemOnClickListener(this);  
 listView.setOnItemClickListener(this);  
  
  
 this.setCancelable(true);  
 }  
  
  
 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
  
 }  
  
  
 public void itemClick(View v) {  
 position = (Integer) v.getTag();  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* LoadDremind loadDremind = new LoadDremind();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 BeanDRemindInformation.*tsset* = loadDremind.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),Task\_Update.*getEventid*());  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 if (BeanDRemindInformation.*tsset* == null){  
 Thread s = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* NewDRemind newDRemind = new NewDRemind();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 newDRemind.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),Task\_Update.*getEventid*());  
 }  
 });  
 s.start();  
 try {  
 s.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 Thread r = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* ChangeAhead changeAhead = new ChangeAhead();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 changeAhead.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),Task\_Update.*getEventid*(),position);  
 }  
 });  
 r.start();  
 try {  
 r.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 else {  
 Thread r = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* ChangeAhead changeAhead = new ChangeAhead();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 changeAhead.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid(),Task\_Update.*getEventid*(),position);  
 }  
 });  
 r.start();  
 try {  
 r.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 switch (position){  
 case 0:show = "准时";break;  
 case 1:show = "提前 5 分钟";break;  
 case 2:show = "提前 30 分钟";break;  
 case 3:show = "提前 1 小时";break;  
 case 4:show = "提前 1 天";break;  
 }  
 Toast.*makeText*(getContext(),show,Toast.*LENGTH\_SHORT*).show();  
 dismiss();  
 }  
}

/\*  
 \* Copyright 2016 Eric Liu  
 \*  
 \* Licensed under the Apache License, Version 2.0 (the "License");  
 \* you may not use this file except in compliance with the License.  
 \* You may obtain a copy of the License at  
 \*  
 \* http://www.apache.org/licenses/LICENSE-2.0  
 \*  
 \* Unless required by applicable law or agreed to in writing, software  
 \* distributed under the License is distributed on an "AS IS" BASIS,  
 \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 \* See the License for the specific language governing permissions and  
 \* limitations under the License.  
 \*/  
  
package com.example.administrator.timenote.Ui;  
  
import android.content.Context;  
import android.support.v7.widget.RecyclerView;  
import android.view.LayoutInflater;  
import android.view.ViewGroup;  
import android.widget.ImageView;  
  
import com.bumptech.glide.Glide;  
import com.example.administrator.timenote.R;  
import com.liuguangqiang.ipicker.adapters.BaseAdapter;  
  
import java.util.List;  
  
*/\*\*  
 \* Created by Eric on 16/9/12.  
 \*/*public class SelectedAdapter extends BaseAdapter<String, SelectedAdapter.ViewHolder> {  
 public static ImageView *ivPhoto*;  
 public SelectedAdapter(Context context, List<String> data) {  
 super(context, data);  
 }  
  
 @Override  
 public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
 return new ViewHolder(layoutInflater, parent, false);  
 }  
  
 @Override  
 public void onBindViewHolder(ViewHolder holder, int position) {  
 super.onBindViewHolder(holder, position);  
 holder.bindData(data.get(position));  
 }  
  
 public static class ViewHolder extends RecyclerView.ViewHolder {  
  
  
  
 public ViewHolder(LayoutInflater layoutInflater, ViewGroup parent, boolean attachToRoot) {  
 super(layoutInflater.inflate(R.layout.*item\_selected*, parent, attachToRoot));  
 *ivPhoto* = (ImageView) itemView.findViewById(R.id.*iv\_photo*);  
 }  
  
 public void bindData(String path) {  
 Glide.*with*(itemView.getContext()).load(path).into(*ivPhoto*);  
 }  
  
 }  
  
}

package com.example.administrator.timenote.Ui;  
  
import android.annotation.SuppressLint;  
import android.app.ProgressDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.content.pm.ActivityInfo;  
import android.support.v7.app.AlertDialog;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.text.method.PasswordTransformationMethod;  
import android.view.KeyEvent;  
import android.view.View;  
import android.view.Window;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ProgressBar;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.example.administrator.timenote.Manager.UserManager.UserActivity;  
import com.example.administrator.timenote.Manager.UserManager.UserCreate;  
import com.example.administrator.timenote.Manager.UserManager.UserLogin;  
import com.example.administrator.timenote.R;  
  
import static com.example.administrator.timenote.Model.BeanUserInformation.*tryLoginUser*;  
  
  
public class Sign\_UpActivity extends AppCompatActivity {  
 private EditText email,name,pwd,pwd2;  
 private TextView email\_error\_1,email\_remind\_1,email\_error\_2,pwd\_error\_1,pwd\_error\_2;  
 private ProgressDialog logDialog;// 等待注册的加载框  
 public static String *semail*;// 邮箱提取  
 private String sname,spwd;// 昵称和密码提取  
 private String spwd2;// 确认密码提取  
 private static String *judge*;  
  
 @SuppressLint("ResourceType")  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 supportRequestWindowFeature(Window.*FEATURE\_NO\_TITLE*);  
 setContentView(R.layout.*sign\_up\_1*);  
  
 {setRequestedOrientation(ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT*);}  
 Button button1= findViewById(R.id.*sign\_up\_2*);//确认注册按钮  
 Button button2= findViewById(R.id.*sign\_in\_1*);//已有账号按钮  
 Button back1=findViewById(R.id.*back\_1*);  
 email=findViewById(R.id.e\_mail\_1);  
 name=findViewById(R.id.name\_1);  
 pwd=findViewById(R.id.pwd\_1);  
 pwd2=findViewById(R.id.pwd\_2);  
 email\_error\_1=findViewById(R.id.emil\_error\_1);  
 email\_error\_2=findViewById(R.id.emil\_error\_2);  
 email\_remind\_1=findViewById(R.id.email\_remind\_1);  
 pwd\_error\_1=findViewById(R.id.pwd\_error\_1);  
 pwd\_error\_2=findViewById(R.id.pwd\_error\_2);  
  
 if (MainActivity.suesrid != null){  
 email.setText(MainActivity.suesrid);  
 name.requestFocus();  
 }  
  
 // 密码隐藏  
 pwd.setTransformationMethod(PasswordTransformationMethod  
 .getInstance());  
 pwd2.setTransformationMethod(PasswordTransformationMethod  
 .getInstance());  
  
 back1.setOnClickListener(new View.OnClickListener(){  
 public void onClick(View view) {  
 finish();  
 // 转到登录界面  
 }  
 });  
  
 // 确认注册按钮  
 button1.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View v) {  
 semail=email.getText().toString();  
 sname=name.getText().toString();  
 spwd=pwd.getText().toString();  
 spwd2=pwd2.getText().toString();  
 showProgressDialog();  
  
  
  
 }  
 });  
  
 // 已有账号按钮  
 button2.setOnClickListener(new View.OnClickListener(){  
 public void onClick(View view) {  
 Intent intent =new Intent(Sign\_UpActivity.this,MainActivity.class);  
 startActivity(intent);  
 // 转到登录界面  
 }  
 });  
  
  
 // 邮箱焦点获得与失去  
 email.setOnFocusChangeListener(new android.view.View.  
 OnFocusChangeListener() {  
 @Override  
 public void onFocusChange(View v, boolean hasFocus) {  
 if (hasFocus) {  
 // 此处为得到焦点时的处理内容  
 email\_error\_1.setVisibility(View.INVISIBLE);  
 email\_error\_2.setVisibility(View.INVISIBLE);  
 email\_remind\_1.setVisibility(View.INVISIBLE);  
 }  
 else {  
 // 此处为失去焦点时的处理内容  
 semail=email.getText().toString();  
  
 // 开启子线程进行邮箱判重  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // TODO Auto-generated method stub  
 UserLogin userLogin = new UserLogin();  
 try {  
 Thread.sleep(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 userLogin.getRemoteInfo(semail);  
 runOnUiThread(new Runnable(){  
 @Override  
 public void run() {  
 // 更新UI  
 // 提示邮箱是否已经注册  
 if(semail.indexOf("@")<0)  
 {  
 email\_error\_1.setVisibility(View.VISIBLE);  
 }  
 else if(tryLoginUser.getUseremail()!=null&&tryLoginUser.getStopdate().equals("0001-01-01T00:00:00"))  
 email\_error\_2.setVisibility(View.VISIBLE);  
 else  
 email\_remind\_1.setVisibility(View.VISIBLE);  
 }  
  
 });  
 }  
 });  
 t.start();  
 try {  
 t.join(3000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 }  
 });  
  
 // 密码焦点获得与失去  
 pwd.setOnFocusChangeListener(new android.view.View.  
 OnFocusChangeListener() {  
 @Override  
 public void onFocusChange(View v, boolean hasFocus) {  
 if (hasFocus) {  
 // 此处为得到焦点时的处理内容  
 pwd\_error\_1.setVisibility(View.INVISIBLE);  
 }  
 else {  
 // 此处为失去焦点时的处理内容  
 String spwd=pwd.getText().toString();  
 if(spwd.length()<8|spwd.length()>20)  
 {  
 pwd\_error\_1.setVisibility(View.VISIBLE);  
 }  
 }  
 }  
 });  
  
 pwd2.setOnFocusChangeListener(new android.view.View.  
 OnFocusChangeListener() {  
 @Override  
 public void onFocusChange(View v, boolean hasFocus) {  
 if (hasFocus) {  
 // 此处为得到焦点时的处理内容  
 pwd\_error\_2.setVisibility(View.INVISIBLE);  
 }  
 else {  
 // 此处为失去焦点时的处理内容  
 String spwd=pwd.getText().toString();  
 String spwd2=pwd2.getText().toString();  
 if(spwd.equals(spwd2)!=true)  
 {  
 pwd\_error\_2.setVisibility(View.VISIBLE);  
 }  
 }  
 }  
 });  
  
 //四个文本框监听回车  
 email.setOnEditorActionListener(new TextView.OnEditorActionListener() {  
 @Override  
 public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {  
 name.requestFocus();  
 return false;  
 }  
 });  
  
 name.setOnEditorActionListener(new TextView.OnEditorActionListener() {  
 @Override  
 public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {  
 pwd.requestFocus();  
 return false;  
 }  
 });  
  
 pwd.setOnEditorActionListener(new TextView.OnEditorActionListener() {  
 @Override  
 public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {  
 pwd2.requestFocus();  
 return false;  
 }  
 });  
  
 pwd2.setOnEditorActionListener(new TextView.OnEditorActionListener() {  
 @Override  
 public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {  
 pwd2.clearFocus();  
 return false;  
 }  
 });  
 }  
 private void showProgressDialog(){  
 ProgressDialog dialog = new ProgressDialog(this);  
 dialog.setProgressStyle(ProgressDialog.STYLE\_SPINNER);// 设置进度条的形式为圆形转动的进度条  
 dialog.setCancelable(true);// 设置是否可以通过点击Back键取消  
 dialog.setCanceledOnTouchOutside(false);// 设置在点击Dialog外是否取消Dialog进度条  
 dialog.setIcon(R.mipmap.ic\_launcher);//  
 // 设置提示的title的图标，默认是没有的，需注意的是如果没有设置title的话只设置Icon是不会显示图标的  
 dialog.setTitle("Notice");  
 // dismiss监听  
 dialog.setOnDismissListener(new DialogInterface.OnDismissListener() {  
  
 @Override  
 public void onDismiss(DialogInterface dialog) {  
 }  
 });  
 // 监听Key事件被传递给dialog  
 dialog.setOnKeyListener(new DialogInterface.OnKeyListener() {  
  
 @Override  
 public boolean onKey(DialogInterface dialog, int keyCode,  
 KeyEvent event) {  
 return false;  
 }  
 });  
 // 监听cancel事件  
 dialog.setOnCancelListener(new DialogInterface.OnCancelListener() {  
  
 @Override  
 public void onCancel(DialogInterface dialog) {  
 }  
 });  
 //设置可点击的按钮，最多有三个(默认情况下)  
 dialog.setButton(DialogInterface.BUTTON\_POSITIVE, "Confirm",  
 new DialogInterface.OnClickListener() {  
  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 }  
 });  
 dialog.setButton(DialogInterface.BUTTON\_NEGATIVE, "Cancel",  
 new DialogInterface.OnClickListener() {  
  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 }  
 });  
 dialog.setButton(DialogInterface.BUTTON\_NEUTRAL, "Natrue",  
 new DialogInterface.OnClickListener() {  
  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 }  
 });  
 dialog.setMessage("Loging in ...");  
 dialog.show();  
 if(semail.indexOf("@")>0&&spwd.length()>=8&& spwd.equals(spwd2)) {  
 Thread j = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // TODO Auto-generated method stub  
 UserCreate userCreate = new UserCreate();  
 try {  
 Thread.sleep(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 judge = userCreate.getRemoteInfo(semail, spwd, sname);  
 }  
 });  
 j.start();  
 try {  
 j.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 // 判断预插入成功与否  
 if (judge.equals("false")) {  
 // 注册失败提示  
 dialog.dismiss();  
 email.requestFocus();  
 pwd.requestFocus();  
 pwd2.requestFocus();  
 name.requestFocus();  
 Toast.makeText(Sign\_UpActivity.this, "注册失败,请重试", Toast.LENGTH\_SHORT).show();  
  
 // Intent intent =new Intent(Sign\_UpActivity.this,MainActivity.class);  
 // startActivity(intent);  
 // finish();  
 }  
 else {  
 dialog.dismiss();  
 final Email\_sure email\_sure = new Email\_sure(Sign\_UpActivity.this, R.style.dialog);  
 email\_sure.show();  
 email\_sure.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @Override  
 public void onDismiss(final DialogInterface dialog) {  
 if (email\_sure.getIssue()) {  
 // 激活用户  
 Thread r = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // TODO Auto-generated method stub  
 UserActivity userActivity = new UserActivity();  
 try {  
 Thread.sleep(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 userActivity.getRemoteInfo(Sign\_UpActivity.semail);  
  
 }  
 });  
 r.start();  
// try {  
// r.join(30000);  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
 Toast.makeText(Sign\_UpActivity.this, "注册成功", Toast.LENGTH\_SHORT).show();  
 finish();  
 }  
 }  
 });  
 }  
 }  
 else  
 {  
 email.requestFocus();  
 pwd.requestFocus();  
 pwd2.requestFocus();  
 name.requestFocus();  
 dialog.dismiss();  
 }  
// if(semail.indexOf("@")>0&&spwd.length()>=8&& spwd.equals(spwd2)) {  
// Thread j = new Thread(new Runnable() {  
// @Override  
// public void run() {  
// // TODO Auto-generated method stub  
// UserCreate userCreate = new UserCreate();  
// try {  
// Thread.sleep(300);  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
// judge = userCreate.getRemoteInfo(semail, spwd, sname);  
// }  
// });  
// j.start();  
// try {  
// j.join(3000);  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
// // 判断预插入成功与否  
// if (judge.equals("false")) {  
// // 注册失败提示  
// Toast.makeText(Sign\_UpActivity.this, "注册失败,请重试", Toast.LENGTH\_SHORT).show();  
//  
//  
// // Intent intent =new Intent(Sign\_UpActivity.this,MainActivity.class);  
// // startActivity(intent);  
// // finish();  
// }  
// else {  
// final Email\_sure email\_sure = new Email\_sure(Sign\_UpActivity.this, R.style.dialog);  
// // TODO: 2018/6/6 换成登录提示错误的那种弹框  
// email\_sure.show();  
// email\_sure.setOnDismissListener(new DialogInterface.OnDismissListener() {  
// @Override  
// public void onDismiss(DialogInterface dialog) {  
// if (email\_sure.getIssue()) {  
// // 激活用户  
// Thread r = new Thread(new Runnable() {  
// @Override  
// public void run() {  
// // TODO Auto-generated method stub  
// UserActivity userActivity = new UserActivity();  
// try {  
// Thread.sleep(300);  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
// userActivity.getRemoteInfo(Sign\_UpActivity.semail);  
//  
// }  
// });  
// r.start();  
// try {  
// r.join(30000);  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
// finish();  
// Toast.makeText(Sign\_UpActivity.this, "注册成功", Toast.LENGTH\_SHORT).show();  
// }  
// }  
// });  
//  
// }  
  
// }  
// else  
// {  
// email.requestFocus();  
// pwd.requestFocus();  
// pwd2.requestFocus();  
// name.requestFocus();  
// dialog.dismiss();  
// }  
 }  
// public void initSignUp(){  
// //判断是否是注册未激活用户  
// if(tryLoginUser==null) {  
// Thread t = new Thread(new Runnable() {  
// @Override  
// public void run() {  
// // TODO Auto-generated method stub  
// UserLogin userLogin = new UserLogin();  
// try {  
// Thread.sleep(300);  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
// userLogin.getRemoteInfo(semail);  
// }  
// });  
// t.start();  
// try {  
// t.join(30000);  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
// }  
// else{  
// if(tryLoginUser.getUseremail()!=null&&tryLoginUser.getStopdate()!=null){  
// Thread j = new Thread(new Runnable() {  
// @Override  
// public void run() {  
// // TODO Auto-generated method stub  
// EmailResend emailResend = new EmailResend();  
// try {  
// Thread.sleep(300);  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
// emailResend.getRemoteInfo(semail);  
// }  
// });  
// j.start();  
// try {  
// j.join();  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
// }  
// else {  
// // 注册账号，返回注册结果  
// // 开启子线程进行用户信息预插入  
// Thread j = new Thread(new Runnable() {  
// @Override  
// public void run() {  
// // TODO Auto-generated method stub  
// UserCreate userCreate = new UserCreate();  
// try {  
// Thread.sleep(300);  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
// judge = userCreate.getRemoteInfo(semail,spwd,sname);  
// }  
// });  
// j.start();  
// try {  
// j.join();  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
// // 判断预插入成功与否  
// if(judge.equals("false")){  
// // 注册失败提示  
// Toast.makeText(Sign\_UpActivity.this, "注册失败,请重试", Toast.LENGTH\_SHORT).show();  
// Thread s = new Thread(new Runnable() {  
// @Override  
// public void run() {  
// // TODO Auto-generated method stub  
// UserRollBack userRollBack = new UserRollBack();  
// try {  
// Thread.sleep(300);  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
// userRollBack.getRemoteInfo(Sign\_UpActivity.semail);  
//  
// }  
// });  
// s.start();  
// try {  
// s.join(30000);  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
// finish();  
// }  
// }  
// }  
// }  
}

package com.example.administrator.timenote.Ui;  
  
import android.annotation.SuppressLint;  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ArrayAdapter;  
import android.widget.Button;  
  
import com.example.administrator.timenote.Model.BeanEventInformation;  
import com.example.administrator.timenote.R;  
  
import java.util.ArrayList;  
  
public class task\_select\_Adapter extends ArrayAdapter<BeanEventInformation> implements View.OnClickListener {  
 private int resourceId;  
 private ListAdapter.InnerItemOnclickListener mListener;  
  
 public task\_select\_Adapter (Context context, int resource, ArrayList<BeanEventInformation> objects) {  
 super(context, resource, objects);  
 resourceId=resource;  
 }  
  
 public View getView(int position, View convertView, ViewGroup parent){  
 ViewHolder holder=null;  
 @SuppressLint("ViewHolder")  
 View view = LayoutInflater.from(getContext()).inflate(resourceId,null,false);  
 holder=new ViewHolder();  
 holder.mButton= (Button) view.findViewById(R.id.task\_select\_button\_1);  
 holder.mButton.setText(getItem(position).getEventname());  
 holder.mButton.setOnClickListener(this);  
 holder.mButton.setTag(position);  
 return view;  
  
 }  
  
 interface InnerItemOnclickListener {  
 void itemClick(View v);  
 }  
  
 public void setOnInnerItemOnClickListener(ListAdapter.InnerItemOnclickListener listener){  
 this.mListener=listener;  
 }  
  
 @Override  
 public void onClick(View v) {  
 mListener.itemClick(v);  
 }  
 static class ViewHolder{  
 Button mButton;  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.app.Dialog;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.AdapterView;  
import android.widget.Button;  
import android.widget.ListView;  
import android.widget.Toast;  
  
import com.example.administrator.timenote.Manager.ListManager.LoadAllList;  
import com.example.administrator.timenote.Manager.TaskManager.LoadAllEvent;  
import com.example.administrator.timenote.Model.BeanEventInformation;  
import com.example.administrator.timenote.Model.BeanListInformation;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.Model.List\_menu;  
import com.example.administrator.timenote.R;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import static com.example.administrator.timenote.Model.BeanEventInformation.*allEventList*;  
  
public class Task\_select\_list extends Dialog implements AdapterView.OnItemClickListener, ListAdapter.InnerItemOnclickListener {  
 */\*\*  
 \* 上下文对象 \*  
 \*/* private Button back;// 取消按钮  
 private Context context;  
 private MyCustomAdapter3 listAdapter;// 事务列表适配器  
 private ListView listView;// 事务列表  
 private ArrayList<BeanEventInformation> listname;// 事务名称列表  
 private Button list\_select\_1;// 清单选择按钮  
 private List\_select list\_select;// 清单选择  
 private int listid;  
 private boolean issure;  
 private int nposition;  
  
 public int getNposition() {  
 return nposition;  
 }  
  
  
  
 public boolean isIssure() {  
 return issure;  
 }  
  
  
  
  
 public Task\_select\_list(Context context) {  
 super(context);  
 this.context = context;  
 }  
  
 public Task\_select\_list(Context context, int theme) {  
 super(context, theme);  
 this.context = context;  
 }  
  
 protected void onCreate(Bundle savedInstanceState) {  
  
 listid = BeanListInformation.*allList*.get(0).getListid();  
  
 super.onCreate(savedInstanceState);  
 // 指定布局  
 this.setContentView(R.layout.*task\_select\_list\_1*);  
  
  
 /\*  
 \* 获取验证码的窗口对象及参数对象以修改对话框的布局设置, 可以直接调用getWindow(),表示获得这个Activity的Window  
 \* 对象,这样这可以以同样的方式改变这个Activity的属性.  
 \*/  
 Window dialogWindow = this.getWindow();  
  
// WindowManager m = context.getWindowManager();  
// Display d = m.getDefaultDisplay(); // 获取屏幕宽、高用  
 WindowManager.LayoutParams p = dialogWindow.getAttributes(); // 获取对话框当前的参数值  
// p.height = (int) (d.getHeight() ); // 高度设置为屏幕  
// p.width = (int) (d.getWidth() ); // 宽度设置为屏幕  
 dialogWindow.setAttributes(p);  
  
 // 根据id在布局中找到控件对象  
 // 取消按钮  
 back=findViewById(R.id.*back\_6*);  
  
 // 事务列表添加  
 listView =findViewById(R.id.*listView4*);  
 initEventList();  
  
 // 任务清单选择按钮  
 list\_select\_1=findViewById(R.id.*list\_select\_2*);  
  
 // 为按钮绑定点击事件监听器  
 back.setOnClickListener(new View.OnClickListener(){  
 public void onClick(View ciew)  
 {  
 dismiss();  
 }  
 });  
  
// listView.setOnItemClickListener(this);  
  
 list\_select\_1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 list\_select=new List\_select(context,R.style.*dialog*);  
 list\_select.show();  
 list\_select.setOnDismissListener(new OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialogInterface) {  
 list\_select\_1.setText(List\_select.*getNlistname*());  
 listid = List\_select.*getNlistid*();  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* LoadAllEvent loadAllEvent = new LoadAllEvent();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 loadAllEvent.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid());  
  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 initEventList();  
 }  
 });  
 }  
 });  
  
 this.setCancelable(true);  
 }  
  
  
 @Override  
 public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {  
// Toast.makeText(context,"列表 "+adapterView+" 被点击了",Toast.LENGTH\_SHORT).show();  
 }  
  
 @Override  
 public void itemClick(View v) {  
 int position;  
 position = (Integer) v.getTag();  
 switch (v.getId()) {  
 case R.id.*task\_select\_button\_1*:  
 Toast.*makeText*(context,"点击"+MyCustomAdapter3.*getData*().get(position).getEventname(),Toast.*LENGTH\_SHORT*).show();  
 issure = true;  
 nposition = position;  
 dismiss();  
 break;  
 default:  
 break;  
 }  
 }  
  
 private void initEventList(){  
  
 listAdapter = new MyCustomAdapter3(getLayoutInflater().getContext());  
  
 for(int i = 0; i < *allEventList*.size(); i++){  
 if ((*allEventList*.get(i).getListid() == listid) && (*allEventList*.get(i).getCheckBox() == 0))  
 listAdapter.addItem(*allEventList*.get(i));  
 }  
  
 listAdapter.setOnInnerItemOnClickListener(this);  
 listView.setAdapter(listAdapter);  
 listView.setOnItemClickListener(this);  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.app.Dialog;  
import android.app.TimePickerDialog;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.pm.ActivityInfo;  
import android.graphics.Color;  
import android.os.Message;  
import android.os.SystemClock;  
import android.support.v7.app.AlertDialog;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.Display;  
import android.view.Gravity;  
import android.view.MenuItem;  
import android.view.View;  
import android.view.ViewGroup;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.Button;  
import android.widget.CompoundButton;  
import android.widget.EditText;  
import android.widget.FrameLayout;  
import android.widget.ImageView;  
import android.widget.PopupMenu;  
import android.widget.Switch;  
import android.widget.TextView;  
import android.widget.TimePicker;  
import android.widget.Toast;  
  
import com.bigkoo.pickerview.builder.TimePickerBuilder;  
import com.bigkoo.pickerview.listener.CustomListener;  
import com.bigkoo.pickerview.listener.OnTimeSelectChangeListener;  
import com.bigkoo.pickerview.listener.OnTimeSelectListener;  
import com.bigkoo.pickerview.view.TimePickerView;  
import com.example.administrator.timenote.Manager.TaskManager.DeleteEvent;  
import com.example.administrator.timenote.Manager.TaskManager.LoadAllEvent;  
import com.example.administrator.timenote.Manager.TaskManager.UpdateEvent;  
import com.example.administrator.timenote.Model.BeanEventInformation;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
  
import java.sql.ResultSet;  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.Date;  
  
public class Task\_Update extends Dialog {  
  
 private Button back;// 返回按钮  
 private Context context;  
 private MenuItem gMenuItem1, gMenuItem2;// list1的两个按钮  
 private String getdate = null;// 获得的时间  
  
 public boolean issure() {  
 return issure;  
 }  
  
 private boolean issure = false;  
 private Button level\_update;// 优先级修改  
 private Button list\_spot\_3;// 菜单按钮  
 private TextView list\_name\_task;// 修改清单按钮  
 private TimePickerView pvCustomTime,pvTime;// 时间选择  
 private Button rering\_setup;// 重复设置  
 private Button ring\_setup;// 提醒时间设置  
 private Switch switch2;// 开启叶子计时按钮  
 private Button save;// 保存按钮  
 private Button time\_setup;// 事务时间修改  
 private EditText task\_update\_name;// 事务名称修改  
 private EditText task\_update\_d;// 事务描述修改  
 private Calendar selectedDate;  
  
 //private int position = getIntent().getIntExtra("position",0);  
 private static int *eventid*;  
 private String oldname;  
 private String oldnote;  
 private String oldleaveseventsign;  
 private Date eventdate;  
  
 private int getpriority = -1;// 获得优先级  
 private int getlistid = -1;// 获得清单id  
 private int leaveseventsign = -1;// 获得叶子事务选择  
 private String geteventname = "";// 获得事务名称  
 private String geteventnote = "";// 获得事务备注  
  
 public static int getEventid() {  
 return *eventid*;  
 }  
  
 public Task\_Update(){  
 super(null);  
 }  
  
 public Task\_Update(Context context) {  
 super(context);  
 this.context = context;  
 }  
  
 public Task\_Update(Context context, int theme) {  
 super(context, theme);  
 this.context = context;  
 }  
 public Task\_Update(Context context, int theme,int eventid,String eventname,String eventnote,String leaveseventsign, Date eventdate) {  
 super(context, theme);  
 this.context = context;  
 this.*eventid* = eventid;  
 this.oldname = eventname;  
 this.oldnote = eventnote;  
 this.oldleaveseventsign = leaveseventsign;  
 this.eventdate = eventdate;  
 }  
  
// @Override  
// public void show() {  
// super.show();  
// /\*\*  
// \* 设置宽度全屏，要设置在show的后面  
// \*/  
// WindowManager.LayoutParams layoutParams = getWindow().getAttributes();  
// layoutParams.gravity = Gravity.BOTTOM;  
// layoutParams.width = ViewGroup.LayoutParams.MATCH\_PARENT;  
// layoutParams.height = ViewGroup.LayoutParams.WRAP\_CONTENT;  
//  
// getWindow().getDecorView().setPadding(0, 0, 0, 0);  
//  
// getWindow().setAttributes(layoutParams);  
// }  
  
 protected void onCreate(Bundle savedInstanceState) {  
  
 super.onCreate(savedInstanceState);  
 this.setContentView(R.layout.*activity\_task\_\_update*);  
  
 Window dialogWindow = this.getWindow();  
  
// WindowManager m = context.getWindowManager();  
// Display d = m.getDefaultDisplay(); // 获取屏幕宽、高用  
 WindowManager.LayoutParams p = dialogWindow.getAttributes(); // 获取对话框当前的参数值  
// p.height = (int) (d.getHeight() ); // 高度设置为屏幕  
// p.width = (int) (d.getWidth() ); // 宽度设置为屏幕  
 dialogWindow.setAttributes(p);  
 getWindow().setLayout(ViewGroup.LayoutParams.*MATCH\_PARENT*, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
  
 //绑定按钮  
 back =findViewById(R.id.*back\_8*);  
 level\_update = findViewById(R.id.*level\_update*);  
 list\_spot\_3 = findViewById(R.id.*list\_spot\_3*);  
 list\_name\_task = findViewById(R.id.*list\_name\_task*);  
 rering\_setup = findViewById(R.id.*rering\_setup*);  
 ring\_setup = findViewById(R.id.*ring\_setup*);  
 save = findViewById(R.id.*save\_update*);  
 switch2 = findViewById(R.id.*switch2*);  
 time\_setup = findViewById(R.id.*time\_setup*);  
 task\_update\_name = findViewById(R.id.*task\_upda\_name*);  
 task\_update\_d = findViewById(R.id.*task\_update\_d*);  
  
 task\_update\_name.setText(oldname);  
 if(oldnote != null){  
 task\_update\_d.setText(oldnote);  
 }  
  
 if (oldleaveseventsign.equals("true"))  
 switch2.setChecked(true);  
 else  
 switch2.setChecked(false);  
  
 // 绑定监听  
 back.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 dismiss();  
 }  
 });  
  
 // 是否允许设置提醒  
// if (getdate == null){  
// ring\_setup.setVisibility(View.INVISIBLE);  
// rering\_setup.setVisibility(View.INVISIBLE);  
// }  
  
 // 开启叶子计时  
 switch2.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {  
 @Override  
 public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {  
 if (isChecked){  
 Toast.*makeText*(getContext(),"开启叶子计时",Toast.*LENGTH\_SHORT*).show();  
 leaveseventsign = 1;  
 }else {  
 Toast.*makeText*(getContext(),"关闭叶子计时",Toast.*LENGTH\_SHORT*).show();  
 leaveseventsign = 0;  
 }  
 }  
 });  
  
 // 优先级选择  
 level\_update.setOnClickListener( new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Level\_select level\_select = new Level\_select(getContext(),R.style.*dialog*);  
 level\_select.show();  
 level\_select.setOnDismissListener(new OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialog) {  
 initlevelupdate();  
 }  
 });  
 }  
 });  
  
  
 // 显示状态子菜单  
 list\_spot\_3.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 showPopupMenu(list\_spot\_3);  
 }  
 });  
  
 list\_name\_task.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 List\_select list\_select = new List\_select(getContext(),R.style.*dialog*);  
 list\_select.show();  
 list\_select.setOnDismissListener(new OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialog) {  
  
 }  
 });  
 }  
 });  
  
 // 提醒重复设置  
 rering\_setup.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Rering\_setup rering\_setup = new Rering\_setup(getContext(),R.style.*dialog*);  
 rering\_setup.show();  
 rering\_setup.setOnDismissListener(new OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialog) {  
  
 }  
 });  
 }  
 });  
  
 // 提醒时间设置  
 ring\_setup.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Ringtime\_setup ringtime\_setup = new Ringtime\_setup(getContext(),R.style.*dialog*);  
 ringtime\_setup.show();  
 ringtime\_setup.setOnDismissListener(new OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialog) {  
  
 }  
 });  
 }  
 });  
  
  
 // 保存按钮监听  
 save.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 // 保存信息  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UpdateEvent updateEvent = new UpdateEvent();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 updateEvent.getRemoteInfo(*eventid*,geteventname,getpriority,getlistid,getdate,leaveseventsign,geteventnote);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 issure = true;  
 dismiss();  
 }  
 });  
  
 // 事务时间修改  
 time\_setup.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 initCustomTimePicker();  
 pvCustomTime.show();  
 Toast.*makeText*(getContext(),getdate,Toast.*LENGTH\_SHORT*).show();  
 // 是否允许设置提醒  
// if (getdate != null || !eventdate.toString().equals("Sat Jan 01 00:00:00 GMT+00:00 1")){  
// ring\_setup.setVisibility(View.VISIBLE);  
// rering\_setup.setVisibility(View.VISIBLE);  
// }  
 }  
 });  
  
 // 事务名称框监听  
 task\_update\_name.setOnFocusChangeListener(new View.OnFocusChangeListener() {  
 @Override  
 public void onFocusChange(View v, boolean hasFocus) {  
 if (hasFocus){  
 }  
 else{  
 geteventname = task\_update\_name.getText().toString();  
 }  
 }  
 });  
 // 事务描述框监听  
 task\_update\_d.setOnFocusChangeListener(new View.OnFocusChangeListener() {  
 @Override  
 public void onFocusChange(View v, boolean hasFocus) {  
 if (hasFocus){  
 }  
 else{  
 geteventnote = task\_update\_d.getText().toString();  
 }  
 }  
 });  
// this.setCanceledOnTouchOutside(false);  
// this.setCancelable(true);  
 }  
  
 private void showPopupMenu(View view) {  
 // View当前PopupMenu显示的相对View的位置  
 PopupMenu popupMenu = new PopupMenu(getContext(), view);  
 // menu布局  
 popupMenu.getMenuInflater().inflate(R.menu.*task\_up*, popupMenu.getMenu());  
 gMenuItem1 = popupMenu.getMenu().findItem(R.id.*save*);  
 gMenuItem2 = popupMenu.getMenu().findItem(R.id.*delete*);  
 // menu的item点击事件  
 popupMenu.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {  
 @Override  
 public boolean onMenuItemClick(MenuItem item) {  
 switch (item.getItemId()) {  
 case R.id.*save*:  
 Toast.*makeText*(getContext(),"保存",Toast.*LENGTH\_SHORT*).show();  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* UpdateEvent updateEvent = new UpdateEvent();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 updateEvent.getRemoteInfo(*eventid*,geteventname,getpriority,getlistid,getdate,leaveseventsign,geteventnote);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 issure = true;  
  
// Thread u = new Thread(new Runnable() {  
// @Override  
// public void run() {  
// // *TODO Auto-generated method stub*// for (int i = 0; i < 10; i++) {  
// Message msg = Message.obtain();  
// msg.what = 1;  
// msg.obj = System.currentTimeMillis() + "";  
// page1.handler1.sendMessage(msg);  
// Log.i("MThread", Thread.currentThread().getName() + "----发送了消息！" + msg.obj);  
// SystemClock.sleep(1000);  
// }  
// }  
// });  
// u.start();  
// try {  
// u.join(30000);  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
 dismiss();  
 break;  
 case R.id.*delete*:  
 Toast.*makeText*(getContext(),"删除",Toast.*LENGTH\_SHORT*).show();  
// final AlertDialog.Builder alterDialog = new AlertDialog.Builder(getContext());  
// alterDialog.setTitle("提示！");  
// alterDialog.setTitleColor(Color.parseColor("#00c8aa"));  
// alterDialog.setMessage("您确定要删除这个任务吗？");  
// alterDialog.setCancelable(false);  
 final Delete delete = new Delete(getContext(), R.style.*dialog*);  
 delete.show();  
 delete.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialogInterface) {  
 if (delete.getIssue()) {  
 // *TODO: 2018/6/26 事务删除* Thread r = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* DeleteEvent deleteEvent = new DeleteEvent();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 deleteEvent.getRemoteInfo(*eventid*);  
 }  
 });  
 r.start();  
 try {  
 r.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 issure = true;  
 dismiss();  
 }  
 }  
  
 });  
  
 break;  
 default:break;  
 }  
 return true;  
 }  
 });  
 // PopupMenu关闭事件  
 popupMenu.setOnDismissListener(new PopupMenu.OnDismissListener() {  
 public void onDismiss(PopupMenu menu) {  
 }  
 });  
 popupMenu.show();  
 }  
  
 private String getTime(Date date) { // 可根据需要自行截取数据显示  
 SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm");  
 return format.format(date);  
 }  
 private void initCustomTimePicker() {//Dialog 模式下，在底部弹出  
 selectedDate = Calendar.*getInstance*();  
 pvCustomTime = new TimePickerBuilder(getContext(), new OnTimeSelectListener() {  
 @Override  
 public void onTimeSelect(Date date, View v) {  
 getdate = getTime(date);  
 selectedDate.setTime(date);  
 }  
 })  
 .setDate(selectedDate)  
 .setLayoutRes(R.layout.*pickerview\_custom\_time*, new CustomListener() {  
 @Override  
 public void customLayout(View v) {  
 final TextView tvSubmit = (TextView) v.findViewById(R.id.*tv\_finish*);  
 ImageView ivCancel = (ImageView) v.findViewById(R.id.*iv\_cancel*);  
 tvSubmit.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 pvCustomTime.returnData();  
 pvCustomTime.dismiss();  
 initCustomTimePicker2();  
 pvTime.show();  
 }  
 });  
 ivCancel.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 pvCustomTime.dismiss();  
 }  
 });  
 }  
 })  
 .setTimeSelectChangeListener(new OnTimeSelectChangeListener() {  
 @Override  
 public void onTimeSelectChanged(Date date) {  
 Log.*i*("pvTime", "onTimeSelectChanged");  
 }  
 })  
 .setType(new boolean[]{true, true, true, false, false, false})  
 .isDialog(true)  
 .build();  
 Dialog mDialog = pvCustomTime.getDialog();  
 if (mDialog != null) {  
  
 FrameLayout.LayoutParams params = new FrameLayout.LayoutParams(  
 ViewGroup.LayoutParams.*MATCH\_PARENT*,  
 ViewGroup.LayoutParams.*WRAP\_CONTENT*,  
 Gravity.*BOTTOM*);  
  
 params.leftMargin = 0;  
 params.rightMargin = 0;  
 pvCustomTime.getDialogContainerLayout().setLayoutParams(params);  
  
 Window dialogWindow = mDialog.getWindow();  
 if (dialogWindow != null) {  
 dialogWindow.setWindowAnimations(com.bigkoo.pickerview.R.style.*picker\_view\_slide\_anim*);//修改动画样式  
 dialogWindow.setGravity(Gravity.*CENTER*);//改成Bottom,底部显示  
 }  
 }  
 }  
 private void initCustomTimePicker2() {  
 pvTime = new TimePickerBuilder(getContext(), new OnTimeSelectListener() {  
 @Override  
 public void onTimeSelect(Date date, View v) {  
 getdate = getTime(date);  
 }  
 })  
 .setDate(selectedDate)  
 .setLayoutRes(R.layout.*pickerview\_custom\_time*, new CustomListener() {  
 @Override  
 public void customLayout(View v) {  
 final TextView tvSubmit = (TextView) v.findViewById(R.id.*tv\_finish*);  
 ImageView ivCancel = (ImageView) v.findViewById(R.id.*iv\_cancel*);  
 tvSubmit.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 pvTime.returnData();  
 pvTime.dismiss();  
 }  
 });  
 ivCancel.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 pvTime.dismiss();  
 }  
 });  
 }  
 })  
 .setTimeSelectChangeListener(new OnTimeSelectChangeListener() {  
 @Override  
 public void onTimeSelectChanged(Date date) {  
 Log.*i*("pvTime", "onTimeSelectChanged");  
 }  
 })  
 .setType(new boolean[]{false, false, false, true, true, false})  
 .isDialog(true)  
 .build();  
 Dialog mDialog = pvTime.getDialog();  
 if (mDialog != null) {  
  
 FrameLayout.LayoutParams params = new FrameLayout.LayoutParams(  
 ViewGroup.LayoutParams.*MATCH\_PARENT*,  
 ViewGroup.LayoutParams.*WRAP\_CONTENT*,  
 Gravity.*BOTTOM*);  
  
 params.leftMargin = 0;  
 params.rightMargin = 0;  
 pvTime.getDialogContainerLayout().setLayoutParams(params);  
  
 Window dialogWindow = mDialog.getWindow();  
 if (dialogWindow != null) {  
 dialogWindow.setWindowAnimations(com.bigkoo.pickerview.R.style.*picker\_view\_slide\_anim*);// 修改动画样式  
 dialogWindow.setGravity(Gravity.*CENTER*); // 改成Bottom,底部显示  
 }  
 }  
 }  
  
 private void initlevelupdate() {  
 switch(Level\_select.*getLevel*())  
 {  
 case 0:  
 level\_update.setBackgroundResource(R.drawable.*level0*);  
 getpriority = 0;  
 break;  
 case 1:  
 level\_update.setBackgroundResource(R.drawable.*level1*);  
 getpriority = 1;  
 break;  
 case 2:  
 level\_update.setBackgroundResource(R.drawable.*level2*);  
 getpriority = 2;  
 break;  
 case 3:  
 level\_update.setBackgroundResource(R.drawable.*level3*);  
 getpriority = 3;  
 break;  
 }  
 }  
  
}

package com.example.administrator.timenote.Ui;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ArrayAdapter;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
import com.example.administrator.timenote.Model.task;  
import com.example.administrator.timenote.R;  
  
import java.text.SimpleDateFormat;  
import java.util.List;  
  
public class taskAdapter extends ArrayAdapter<task> {  
 private int resourceId;  
 public taskAdapter(Context context, int resource, List<task> objects) {  
 super(context, resource,objects);  
 resourceId=resource;  
 }  
  
 public View getView(int position, View convertView, ViewGroup parent){  
 task t=getItem(position);  
 View view = LayoutInflater.*from*(getContext()).inflate(resourceId,null,false);  
 View view\_line=convertView.findViewById(R.id.*view\_color\_1*);  
 TextView taskname=convertView.findViewById(R.id.*taskname*);  
 TextView date=convertView.findViewById(R.id.*data\_1*);  
 TextView taskdes=convertView.findViewById(R.id.*taskdes*);  
 TextView taskp=convertView.findViewById(R.id.*taskp*);  
 ImageView ImageId=convertView.findViewById(R.id.*image\_level*);  
  
 taskname.setText(t.getTaskname());  
 SimpleDateFormat format=new SimpleDateFormat("yyyy年MM月dd日 HH:mm:SS");  
 date.setText(format.format( t.getDate\_1()));  
 taskdes.setText(t.getTaskdes());  
 taskp.setText(t.getTaskp());  
 ImageId.setImageResource(t.getImageId());  
 return view;  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.app.Activity;  
import android.content.pm.ActivityInfo;  
import android.os.Bundle;  
import android.support.v7.app.AppCompatActivity;  
import android.view.MenuItem;  
import android.view.View;  
import android.view.Window;  
import android.widget.Button;  
import android.widget.PopupMenu;  
import android.widget.Toast;  
  
import com.example.administrator.timenote.R;  
  
public class Title\_base1\_ad extends AppCompatActivity {  
 private Button list1;  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 supportRequestWindowFeature(Window.FEATURE\_NO\_TITLE);  
 setContentView(R.layout.title\_base1);  
 {  
 setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_PORTRAIT);  
 }  
 list1=findViewById(R.id.taday\_1);  
  
 list1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 showPopupMenu(list1);  
 Toast.makeText(Title\_base1\_ad.this, "确认", Toast.LENGTH\_SHORT).show();  
 }  
 });  
 }  
 private void showPopupMenu(View view) {  
 // View当前PopupMenu显示的相对View的位置  
 PopupMenu popupMenu = new PopupMenu(this, view);  
 // menu布局  
 popupMenu.getMenuInflater().inflate(R.menu.showstyle, popupMenu.getMenu());  
 // menu的item点击事件  
 popupMenu.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {  
 @Override  
 public boolean onMenuItemClick(MenuItem item) {  
 Toast.makeText(Title\_base1\_ad.this, item.getTitle(), Toast.LENGTH\_SHORT).show();  
 return false;  
 }  
 });  
 // PopupMenu关闭事件  
 popupMenu.setOnDismissListener(new PopupMenu.OnDismissListener() {  
 @Override  
 public void onDismiss(PopupMenu menu) {  
 Toast.makeText(Title\_base1\_ad.this, "关闭PopupMenu", Toast.LENGTH\_SHORT).show();  
 }  
 });  
  
 popupMenu.show();  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.annotation.SuppressLint;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.content.pm.ActivityInfo;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.media.Image;  
import android.os.CountDownTimer;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.support.v7.widget.GridLayoutManager;  
import android.support.v7.widget.RecyclerView;  
import android.view.KeyEvent;  
import android.view.MenuItem;  
import android.view.View;  
import android.view.Window;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ImageView;  
import android.widget.PopupMenu;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.bumptech.glide.Glide;  
import com.example.administrator.timenote.Manager.UserManager.ChangeEmail;  
import com.example.administrator.timenote.Manager.UserManager.ChangeName;  
import com.example.administrator.timenote.Manager.UserManager.UserLogin;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
import com.liuguangqiang.ipicker.IPicker;  
import com.liuguangqiang.ipicker.events.IPickerEvent;  
  
import java.io.ByteArrayOutputStream;  
import java.io.File;  
import java.util.ArrayList;  
import java.util.List;  
  
public class User\_information extends AppCompatActivity {  
  
 private Button back9;// 返回按钮  
 private EditText email\_update;// 邮箱  
 private Button loadout;// 登出  
 private EditText name\_update;// 昵称  
 public static ImageView *uesr\_image*;// 用户头像  
 private String semail;  
 private String sname;  
  
  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 supportRequestWindowFeature(Window.*FEATURE\_NO\_TITLE*);  
 setContentView(R.layout.*activity\_user\_information*);  
 {  
 setRequestedOrientation(ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT*);  
 }  
  
 // 控件绑定  
 back9 = findViewById(R.id.*back\_9*);  
 email\_update = findViewById(R.id.*email\_update*);  
 email\_update.setText(BeanUserInformation.*currentLoginUser*.getUseremail());  
 loadout = findViewById(R.id.*load\_out*);  
 name\_update = findViewById(R.id.*name\_update*);  
 name\_update.setText(BeanUserInformation.*currentLoginUser*.getUsername());  
 *uesr\_image* = findViewById(R.id.*user\_image*);  
  
 // 监听回车  
 email\_update.setOnEditorActionListener(new TextView.OnEditorActionListener() {  
 @Override  
 public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {  
 email\_update.requestFocus();  
 return false;  
 }  
 });  
 name\_update.setOnEditorActionListener(new TextView.OnEditorActionListener() {  
 public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {  
 name\_update.clearFocus();  
 return false;  
 }  
 });  
  
 // 返回按钮  
 back9.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 name\_update.requestFocus();  
 email\_update.requestFocus();  
 name\_update.requestFocus();  
 finish();  
 }  
 });  
  
 // 邮箱修改框  
 email\_update.setOnFocusChangeListener(new View.OnFocusChangeListener() {  
 @Override  
 public void onFocusChange(View v, boolean hasFocus) {  
 semail = email\_update.getText().toString();  
 if(hasFocus)  
 {  
  
 }  
 else  
 {  
 if((semail.indexOf("@") > 0) && !semail.equals(BeanUserInformation.*currentLoginUser*.getUseremail()))  
 {  
 //发送验证码  
 final Email\_sure email\_sure= new Email\_sure(User\_information.this,R.style.*dialog*);  
 email\_sure.show();  
 email\_sure.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialog) {  
 if(email\_sure.getIssue())  
 {  
 //修改用户信息  
 Thread f = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // TODO Auto-generated method stub  
 ChangeEmail changeEmail = new ChangeEmail();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 changeEmail.getRemoteInfo(semail);  
  
 }  
 });  
 f.start();  
 try {  
 f.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // TODO Auto-generated method stub  
 UserLogin userLogin = new UserLogin();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 userLogin.getRemoteInfo(semail);  
  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 BeanUserInformation.*currentLoginUser* = BeanUserInformation.*tryLoginUser*;  
 }  
 }  
 });  
 }  
 }  
 }  
 });  
  
 // 登出  
 loadout.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 name\_update.requestFocus();  
 email\_update.requestFocus();  
 name\_update.requestFocus();  
 Intent intent = new Intent(User\_information.this,MainActivity.class);  
 startActivity(intent);  
 File file = new File(User\_information.this.getFilesDir(),"info.txt");  
 file.delete();  
 finish();  
 }  
 });  
  
 // 昵称修改  
 name\_update.setOnFocusChangeListener(new View.OnFocusChangeListener() {  
 @Override  
 public void onFocusChange(View v, boolean hasFocus) {  
 sname = name\_update.getText().toString();  
 if(hasFocus)  
 {  
  
 }  
 else  
 {  
 if (!sname.equals(BeanUserInformation.*currentLoginUser*.getUsername())) {  
 // 修改密码  
 Thread f = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // TODO Auto-generated method stub  
 ChangeName changeName = new ChangeName();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 changeName.getRemoteInfo(sname);  
  
 }  
 });  
 f.start();  
 try {  
 f.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // TODO Auto-generated method stub  
 UserLogin userLogin = new UserLogin();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 userLogin.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUseremail());  
  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 BeanUserInformation.currentLoginUser = BeanUserInformation.tryLoginUser;  
 page4.setUser\_name\_page4(sname);  
 }  
  
 }  
 }  
 });  
  
 // 头像按钮  
 if(BeanUserInformation.currentLoginUser.getIcon()!=null)  
 {  
 Bitmap bm;  
 byte[] bitmapByte = BeanUserInformation.currentLoginUser.getIcon();  
 if (bitmapByte.length != 0) {  
 bm = BitmapFactory.decodeByteArray(bitmapByte, 0,bitmapByte.length);  
 uesr\_image.setImageBitmap(bm);  
 } else {  
  
 }  
 }  
 uesr\_image.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(User\_information.this,UserHead.class);  
 startActivity(intent);  
 }  
 });  
  
 }  
  
 @Override  
 protected void onDestroy() {  
 super.onDestroy();  
  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.content.pm.ActivityInfo;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.support.v7.widget.RecyclerView;  
import android.view.MenuItem;  
import android.view.View;  
import android.view.Window;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.PopupMenu;  
  
import com.bumptech.glide.Glide;  
import com.example.administrator.timenote.Manager.UserManager.SaveHead;  
import com.example.administrator.timenote.Manager.UserManager.UserActivity;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
import com.liuguangqiang.ipicker.IPicker;  
  
import java.io.ByteArrayOutputStream;  
import java.util.ArrayList;  
import java.util.List;  
  
public class UserHead extends AppCompatActivity {  
  
 private Button back;// 返回按钮  
 private ImageView head;// 头像图片  
 private Button list\_spot\_4;// 菜单  
 private Button sure;// 确认按钮  
 private RecyclerView recyclerView;  
 private SelectedAdapter adapter;  
 private ArrayList<String> selectPictures = new ArrayList<>();  
 private byte[] bitmapByte;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 supportRequestWindowFeature(Window.*FEATURE\_NO\_TITLE*);  
 setContentView(R.layout.*activity\_user\_head*);  
 {  
 setRequestedOrientation(ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT*);  
 }  
  
 // 绑定按钮  
 back = findViewById(R.id.*back\_15*);  
 head = findViewById(R.id.*head*);  
 list\_spot\_4 = findViewById(R.id.*list\_spot\_4*);  
 sure = findViewById(R.id.*save\_head*);  
  
 if(BeanUserInformation.*currentLoginUser*.getIcon()!=null) {  
 Bitmap bm;  
 byte[] bitmapByte1 = BeanUserInformation.*currentLoginUser*.getIcon();  
 if (bitmapByte1.length != 0) {  
 bm = BitmapFactory.*decodeByteArray*(bitmapByte1, 0, bitmapByte1.length);  
 head.setImageBitmap(bm);  
 }  
 else {  
  
 }  
 }  
 head.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 showPopupMenu(list\_spot\_4);  
 }  
 });  
  
 // 返回按钮  
 back.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 finish();  
 }  
 });  
  
 // 菜单按钮  
 list\_spot\_4.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 showPopupMenu(list\_spot\_4);  
 }  
 });  
  
 // 保存按钮  
 sure.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 head.setDrawingCacheEnabled(true);  
 Bitmap bm = Bitmap.*createBitmap*(convertViewToBitmap(head));  
  
 ByteArrayOutputStream baos=new ByteArrayOutputStream();  
 bm.compress(Bitmap.CompressFormat.*PNG*, 100, baos);  
 bitmapByte = baos.toByteArray();  
 if (bitmapByte.length != 0) {  
 bm = BitmapFactory.*decodeByteArray*(bitmapByte, 0,bitmapByte.length);  
 } else {  
  
 }  
 if(BeanUserInformation.*currentLoginUser*!=null)  
 {  
 BeanUserInformation.*currentLoginUser*.setIcon(bitmapByte);  
  
 Thread r = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* SaveHead saveHead = new SaveHead();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 saveHead.getRemoteInfo(bitmapByte);  
  
 }  
 });  
 r.start();  
 try {  
 r.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 User\_information.*uesr\_image*.setImageBitmap(bm);  
 page4.*imageButton1*.setImageBitmap(bm);  
 page1.*person*.setImageBitmap(bm);  
 finish();  
 }  
 else{  
 finish();  
 }  
 list\_spot\_4.setDrawingCacheEnabled(false);  
 }  
 });  
 }  
 private void showPopupMenu(View view) {  
  
 IPicker.*setLimit*(1);  
 IPicker.*setCropEnable*(true);  
 IPicker.*setOnSelectedListener*(new IPicker.OnSelectedListener() {  
 @Override  
 public void onSelected(List<String> paths) {  
 selectPictures.clear();  
 selectPictures.addAll(paths);  
 adapter.notifyDataSetChanged();  
 Glide.*with*(UserHead.this).load(paths.get(0)).into(head);  
  
 }  
 });  
 adapter = new SelectedAdapter(getApplicationContext(), selectPictures);  
 // View当前PopupMenu显示的相对View的位置  
 PopupMenu popupMenu = new PopupMenu(this, view);  
 // menu布局  
 popupMenu.getMenuInflater().inflate(R.menu.*photo\_menu*, popupMenu.getMenu());  
 // menu的item点击事件  
 popupMenu.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {  
 @Override  
 public boolean onMenuItemClick(MenuItem item) {  
 switch (item.getItemId()) {  
  
 case R.id.*select*:  
 IPicker.*open*(getApplicationContext(), selectPictures);  
 break;  
 default:  
 }  
 return true;  
 }  
 });  
 // PopupMenu关闭事件  
 popupMenu.setOnDismissListener(new PopupMenu.OnDismissListener() {  
 public void onDismiss(PopupMenu menu) {  
  
 }  
 });  
 popupMenu.show();  
  
 }  
 public Bitmap convertViewToBitmap(View view){  
 view.measure(View.MeasureSpec.*makeMeasureSpec*(0, View.MeasureSpec.*UNSPECIFIED*), View.MeasureSpec.*makeMeasureSpec*(0, View.MeasureSpec.*UNSPECIFIED*));  
 view.layout(0, 0, view.getMeasuredWidth(), view.getMeasuredHeight());  
 Bitmap bitmap = view.getDrawingCache();  
 return bitmap;  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.annotation.SuppressLint;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.os.Bundle;  
import android.os.CountDownTimer;  
import android.support.v7.app.AlertDialog;  
import android.support.v7.app.AppCompatActivity;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.example.administrator.timenote.Manager.LeavesManager.LeavesAdd;  
import com.example.administrator.timenote.Manager.TaskManager.Complete;  
import com.example.administrator.timenote.Manager.TaskManager.UpdatePriority;  
import com.example.administrator.timenote.Model.BeanDRemindInformation;  
import com.example.administrator.timenote.Model.BeanUserInformation;  
import com.example.administrator.timenote.R;  
  
import static com.example.administrator.timenote.Model.BeanEventInformation.*allEventList*;  
  
*/\*\*  
 \* Created by XuanWem Chen on 2018/6/19.  
 \*/*public class Yezi\_start extends AppCompatActivity{  
 private Button back;// 退出按钮  
 private TextView task\_name;// 当前事物名称  
 private TextView time;// 时间  
 private static int *M* = 0;// 分钟  
 private int s = 10;// 秒  
 private int p;  
 private Task\_select\_list task\_select\_list;// 事务选择  
  
 public static void setM(int m) {  
 *M* = m;  
 }  
  
  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_yezi\_start*);  
 // 对应按钮  
 back = findViewById(R.id.*quit\_task*);  
 task\_name = findViewById(R.id.*task\_name\_yezi*);  
 p = getIntent().getIntExtra("p",-1);  
 *M* = BeanDRemindInformation.*defaultset*.getLeavestime();  
 if (p != -1) {  
 task\_name.setText(MyCustomAdapter3.*getData*().get(p).getEventname());  
 task\_name.setVisibility(View.*VISIBLE*);  
 }  
 else {  
 task\_name.setVisibility(View.*INVISIBLE*);  
 }  
 time = findViewById(R.id.*yezi\_jishi*);  
 cdt.start();  
  
 // 退出按钮监听  
 back.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 initquit();  
 }  
 });  
  
 }  
  
 CountDownTimer cdt = new CountDownTimer((*M*\*60+s)\*1000, 1000) {  
  
 @SuppressLint({"ResourceAsColor", "SetTextI18n"})  
 public void onTick(long millisUntilFinished) {  
 if (s <= 0) {  
 s = 60;  
 *M*--;  
 }  
 s--;  
 time.setText(*M* + ":" + s);  
  
 }  
  
  
 @SuppressLint("ResourceAsColor")  
 public void onFinish() {  
 if (*M* == 0 || s == 1){  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* LeavesAdd leavesAdd = new LeavesAdd();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 leavesAdd.getRemoteInfo(BeanUserInformation.*currentLoginUser*.getUserid());  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 // *TODO: 2018/6/19 退出逻辑* initquit();  
 }  
 };  
  
 private void initquit(){  
 AlertDialog.Builder alterDialog = new AlertDialog.Builder(Yezi\_start.this);  
 if (p != -1) {  
 alterDialog.setTitle("您完成当前任务了吗？");  
 alterDialog.setMessage(MyCustomAdapter3.*getData*().get(p).getEventname());  
 alterDialog.setCancelable(true);  
 alterDialog.setPositiveButton("完成", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(Yezi\_start.this, "完成", Toast.*LENGTH\_SHORT*).show();  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* Complete complete = new Complete();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 complete.getRemoteInfo(MyCustomAdapter3.*getData*().get(p).getEventid());  
 int i = 0;  
 for (; i < *allEventList*.size(); i++){  
 if (*allEventList*.get(i).getEventid() == MyCustomAdapter3.*getData*().get(p).getEventid())  
 break;  
 }  
 *allEventList*.get(i).setCheckBox(1);  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 finish();  
 }  
 });  
 alterDialog.setNegativeButton("未完成", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(Yezi\_start.this, "未完成", Toast.*LENGTH\_SHORT*).show();  
 finish();  
 }  
 });  
 alterDialog.setNeutralButton("取消", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(Yezi\_start.this, "取消", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 alterDialog.show();  
 }  
 else {  
 alterDialog.setTitle("您未选择任何任务");  
 alterDialog.setMessage("需要选择您刚才做的任务吗？");  
 alterDialog.setCancelable(true);  
 alterDialog.setPositiveButton("选择", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(Yezi\_start.this, "选择", Toast.*LENGTH\_SHORT*).show();  
 task\_select\_list=new Task\_select\_list(Yezi\_start.this,R.style.*dialog*);  
 task\_select\_list.show();  
 task\_select\_list.setOnDismissListener(new DialogInterface.OnDismissListener() {  
 @Override  
 public void onDismiss(DialogInterface dialogInterface) {  
 if(task\_select\_list.isIssure())  
 {  
 Thread t = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 // *TODO Auto-generated method stub* Complete complete = new Complete();  
 try {  
 Thread.*sleep*(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 complete.getRemoteInfo(MyCustomAdapter3.*getData*().get(task\_select\_list.getNposition()).getEventid());  
 int i = 0;  
 for (; i < *allEventList*.size(); i++){  
 if (*allEventList*.get(i).getEventid() == MyCustomAdapter3.*getData*().get(task\_select\_list.getNposition()).getEventid())  
 break;  
 }  
 *allEventList*.get(i).setCheckBox(1);  
  
 }  
 });  
 t.start();  
 try {  
 t.join(30000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 }  
 finish();  
  
 }  
 });  
 }  
 });  
 alterDialog.setNegativeButton("算了", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(Yezi\_start.this, "算了", Toast.*LENGTH\_SHORT*).show();  
 finish();  
 }  
 });  
 alterDialog.setNeutralButton("取消", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(Yezi\_start.this, "取消", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 alterDialog.show();  
 }  
 }  
}

package com.example.administrator.timenote.Ui;  
  
import android.annotation.SuppressLint;  
import android.app.Dialog;  
import android.content.Context;  
import android.net.sip.SipSession;  
import android.os.Bundle;  
import android.support.annotation.Nullable;  
import android.view.KeyboardShortcutGroup;  
import android.view.Menu;  
import android.view.View;  
import android.view.Window;  
import android.view.WindowManager;  
import android.widget.Button;  
import android.widget.SeekBar;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.example.administrator.timenote.Model.BeanDRemindInformation;  
import com.example.administrator.timenote.R;  
  
import java.util.List;  
  
public class yezi\_time\_select extends Dialog {  
 */\*\*  
 \* 上下文对象 \*  
 \*/* private Button back;// 取消按钮  
 private Context context;  
 private SeekBar seek\_yezi;// 进度条  
 private Button sure;// 确认按钮  
 private TextView yezi\_Time\_Text;// 叶子时长文本  
 private String yezi\_time;// 记录叶子时间  
  
 public String getYezi\_time() {  
 return yezi\_time;  
 }  
  
 public void setYezi\_time(String yezi\_time) {  
 this.yezi\_time = yezi\_time;  
 }  
  
  
  
  
  
 public yezi\_time\_select (Context context) {  
 super(context);  
 this.context = context;  
 }  
  
 public yezi\_time\_select (Context context, int theme) {  
 super(context, theme);  
 this.context = context;  
 }  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 // 指定布局  
 this.setContentView(R.layout.*yezi\_select*);  
  
  
 Window dialogWindow = this.getWindow();  
  
// WindowManager m = context.getWindowManager();  
// Display d = m.getDefaultDisplay(); // 获取屏幕宽、高用  
 WindowManager.LayoutParams p = dialogWindow.getAttributes(); // 获取对话框当前的参数值  
// p.height = (int) (d.getHeight() ); // 高度设置为屏幕  
// p.width = (int) (d.getWidth() ); // 宽度设置为屏幕  
 dialogWindow.setAttributes(p);  
  
 // 根据id在布局中找到控件对象  
 back = findViewById(R.id.*back\_7*);  
 seek\_yezi = findViewById(R.id.*seek\_yezi*);  
 sure = findViewById(R.id.*sure\_yezi*);  
 yezi\_Time\_Text = findViewById(R.id.*yezi\_time\_text*);  
  
 // 初始值  
 seek\_yezi.setProgress(BeanDRemindInformation.*defaultset*.getLeavestime());  
 yezi\_Time\_Text.setText(String.*valueOf*(BeanDRemindInformation.*defaultset*.getLeavestime()));  
  
 // 为按钮绑定点击事件监听器  
 back.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 dismiss();  
 }  
 });  
  
 // 确认按钮  
 sure.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View v) {  
 yezi\_time = yezi\_Time\_Text.getText().toString();  
 dismiss();  
 }  
 });  
  
 // 进度条监听  
 seek\_yezi.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
 */\*\*  
 \* 拖动条停止拖动的时候调用  
 \*/* @Override  
 public void onStopTrackingTouch(SeekBar seekBar) {  
 }  
 */\*\*  
 \* 拖动条开始拖动的时候调用  
 \*/* @Override  
 public void onStartTrackingTouch(SeekBar seekBar) {  
 }  
 */\*\*  
 \* 拖动条进度改变的时候调用  
 \*/* @Override  
 public void onProgressChanged(SeekBar seekBar, int progress,  
 boolean fromUser) {  
  
 yezi\_Time\_Text.setText(String.*valueOf*(progress));  
 }  
 });  
 this.setCancelable(true);  
 }  
  
  
}

package com.example.administrator.timenote  
  
import android.os.Bundle  
import android.support.design.widget.BottomNavigationView  
import android.support.v7.app.AppCompatActivity  
import kotlinx.android.synthetic.main.activity\_main\_file.\*  
  
class MainFile : AppCompatActivity() {  
  
 private val mOnNavigationItemSelectedListener = BottomNavigationView.OnNavigationItemSelectedListener **{** item **->** when (item.*itemId*) {  
 R.id.*priorit\_1* -> {  
 message.setText(R.string.*按优先级排序*)  
 return@OnNavigationItemSelectedListener true  
 }  
 R.id.complete\_1 -> {  
 message.setText(R.string.显示已完成)  
 return@OnNavigationItemSelectedListener true  
 }  
 }  
 false  
 **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_main)  
 navigation.setOnNavigationItemSelectedListener(mOnNavigationItemSelectedListener)  
 }  
}