第 20 章

高级阶段——开发公共交通线路查询系统

学习指引

随着“绿色出行，节能减排”的提出，同时为了缓解城市交通压力，许多城市都在城市发展中优先发展公共交通事业，人们也越来越习惯乘公共交通出行。由于人们出行的随机性比较大，因此怎么能随时随地、方便、快捷地获取出行路线，为自己的行程做合理安排，就显得非常有意义。本章介绍的是基于 Android 平台的公共交通查询系统。

重点导读

- 了解系统开发背景和功能概述。
- 熟悉系统数据库设计。
- 掌握界面相关类的方法。
- 熟悉辅助界面的相关类。
- 掌握数据库表的创建和操作方法。

20.1 系统开发背景及功能概述

1. 开发背景简介

公共交通查询系统通过信息技术的应用，为人们提供方便、快捷的查询功能。其主要包括以下功能。

（1）车次查询：查询本车次的停靠路线。

（2）站点查询：查询所有经过本站点的公共交通车辆。

（3）站站查询：查询经过两站点的公共交通车辆及中转情况。

（4）系统维护：对车次、站点进行维护与更新。

2. 功能概述

通过对人们出行的需求进行分析，我们对公共交通查询系统的功能有了深入了解。下面就可以确定系统实现的功能结构。

本系统包括车次查询、站点查询、站站查询、系统维护等功能，其结构如图 20-1 所示。

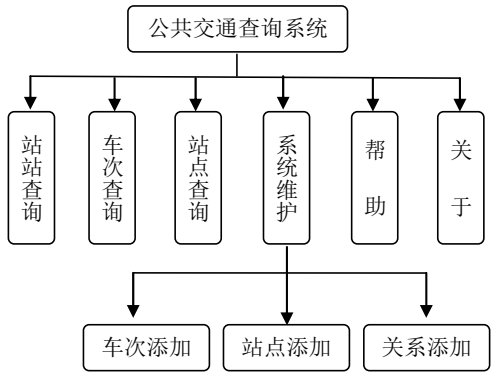

图 20-1　公共交通查询系统的结构

3. 开发环境和目标平台

开发该管理系统需要如下软件环境。

（1）JDK 7 及其以上版本。

（2）Android Studio 集成开发 IDE 工具。

（3）数据库 SQLite。

20.2　开发前的准备工作

数据库是信息查询系统的基础，设计时保证其合理性对提高信息检索效率和后期数据库维护都有很大的好处。一个设计良好的数据库系统同时也可以减少开发的难度和缩短开发周期。

本系统规模比较小，又是单机，故采用 SQLite 作为开发数据库。本系统共有 3 张数据表，分别是车次表（bus）、站点表（busstop）和关系表（relation）。各表间的关系如图 20-2 所示。

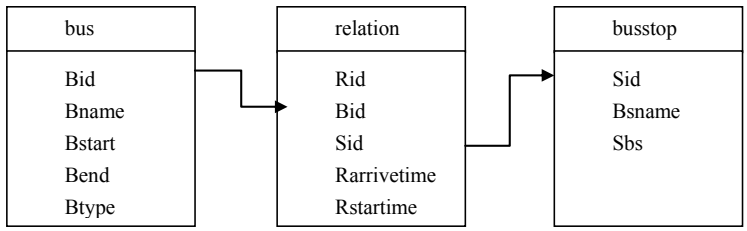

图 20-2　数据表间关系

下面将对图 20-2 的 3 张数据表逐一介绍。

（1）车次表：用于记录车次的基本信息，其具体字段设置情况如表 20-1 所示。

该表的 SQL 语句如下：

```
create table if not exists bus (
   Bid integer primary key,   //车次 ID
   Bname char(20),            //车次名称
   Bstart char(20),           //始发站
   Bend char(20),             //终点站
   Btype char(20));          //车次类型
```

表 20-1　车次表的字段设置情况

字 段 名 称	数 据 类 型	字 段 大 小	是否为主键	是 否 为 空	说　　明
Bid	数字	整型	是	否	车次 ID
Bname	文本	20	否	否	车次名称
Bstart	文本	20	否	否	始发站
Bend	文本	20	否	否	终点站
Btype	文本	20	否	否	车次类型

（2）站点表：用于记录站点的基本信息，其具体字段设置情况如表 20-2 所示。

表 20-2　站点表的字段设置情况

字 段 名 称	数 据 类 型	字 段 大 小	是否为主键	是 否 为 空	说　　明
Sid	数字	整型	是	否	站点 ID
Bsname	文本	20	否	否	站点名称
Sbs	文本	20	否	否	首字母拼音

该表的 SQL 语句如下：

```
create table if not exists  busstop(
    Sid integer primary key,    //站点 ID
    Bsname char(20),            //站点名称
    Sbs char(10));              //首字母拼音
```

（3）关系表：用于把站点表与车次表关联起来，其具体字段设置情况如表 20-3 所示。

表 20-3　关系表的字段设置情况

字 段 名 称	数 据 类 型	字 段 大 小	是否为主键	是 否 为 空	说　　明
Rid	数字	整型	是	否	关系表 ID
Bid	数字	整型	否	否	车次 ID
Sid	数字	整型	否	否	站点 ID
Rarrivetime	文本	20	否	否	到达时间
Rstarttime	文本	20	否	否	出发时间

该表 SQL 语句如下：

```
create table if not exists relation(
    Rid integer primary key,  //关系表 ID
    Bid integer,              //车次 ID
    Sid integer,              //站点 ID
    Rarrivetime char(20),     //到达时间
    Rstarttime char(20));     //出发时间
```

20.3　系统功能预览

本系统实现了公共交通车次查询及车次添加的功能，主要界面设计效果如下。

（1）欢迎界面：人机交互程序有一个好的欢迎界面往往能给用户留下特别的印象，这里的欢迎界面设计效果如图 20-3 所示。

（2）主菜单界面：欢迎界面进入后是主菜单界面，其设计效果如图 20-4 所示。

（3）站站查询界面：效果如图 20-5 所示。

图 20-3　欢迎界面

图 20-4　主菜单界面

图 20-5　站站查询界面

（4）车站查询界面及详细资料界面：其设计效果如图 20-6 和图 20-7 所示。

（5）关于界面效果如图 20-8 所示。

图 20-6　车站查询界面

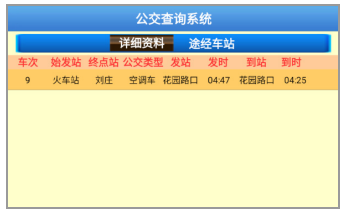

图 20-7　所选车站的详细资料界面

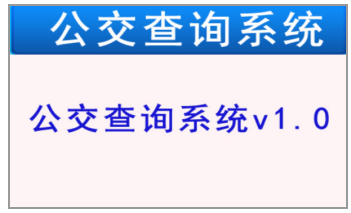

图 20-8　关于界面

（6）车次查询界面及详细资料界面：其设计效果如图 20-9 和图 20-10 所示。

（7）系统维护主界面（图 20-11）：其中设有 3 个子菜单，分别为“车次添加”“车站添加”和“关系添加”界面，设计效果分别如图 20-12～图 20-14 所示。

图 20-9　车次查询界面

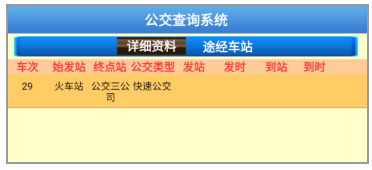

图 20-10　所选车次的详细资料界面

图 20-11　系统维护主界面

图 20-12　车次添加界面

图 20-13　车站添加界面

图 20-14　关系添加界面

20.4 界面主类 GJCXActivity

界面主类的作用是监听用户操作，做相应界面切换，并实现相应功能。实现过程的主要框架代码如下：

```
package com.gjcx;
import java.util.List;
⋮
//省略部分类引入代码
import static com.gjcx.DbUtil.*;
enum WhichView {MAIN_MENU,ZZCX_VIEW,CCCX_VIEW,CZCCCX_VIEW,LIST_VIEW,PASSbusstop_VIEW,
   CCTJ_VIEW,CZTJ_VIEW,GXTJ_VIEW,xtwh_VIEW,WELCOME_VIEW,ABOUT_VIEW,HELP_VIEW}
public class GJCXActivity extends Activity
{
   Welcome wv;                                          //声明欢迎界面变量
   WhichView curr;                                      //声明当前枚举变量
   static int flag;    //设置界面的标志位。其中，0代表站站查询；1代表车次查询；2代表车站（或站点查询）
   String[][]msgg=new String[][]{{""}};                 //存放 listview 中的数组
   String s1[];
   String s2[];
   String currcc;                                       //声明车次变量
   String currzd;                                       //声明站点变量
   Handler hd=new Handler()                             //声明消息处理器
   {
      @Override
      public void handleMessage(Message msg)            //重写方法
      {
         switch(msg.what)
         {
            case 0:                                     //进入欢迎界面
               goToWelcome();
               break;
            case 1:                                     //进入主菜单界面
               goToMainMenu();
               break;
            case 2:                                     //进入关于界面
               setContentView(R.layout.about);
               curr=WhichView.ABOUT_VIEW;
               break;
            case 3:                                     //进入帮助界面
               setContentView(R.layout.help);
               curr=WhichView.HELP_VIEW;
               break;
         }
      }
   };

   @Override
   public void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      requestWindowFeature(Window.FEATURE_NO_TITLE);//设置为全屏模式
      getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN                    ,
WindowManager.LayoutParams. FLAG_FULLSCREEN);             //设置为横屏模式
      setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
      display = getWindowManager().getDefaultDisplay();
```

```
        //通过 WindowManager 获取
        DisplayMetrics dm = new DisplayMetrics();
        getWindowManager().getDefaultDisplay().getMetrics(dm);
        density = dm.density;                          //屏幕密度（像素比例: 0.75/1.0/1.5/2.0）
        int densityDPI = dm.densityDpi;
                                                       //屏幕密度（每寸像素: 120/160/240/320）
        float xdpi = dm.xdpi;
        float ydpi = dm.ydpi;

        CreatTable.creattable();                       //建表
        iniTLisit();                                   //初始化数组
        this.hd.sendEmptyMessage(0);                   //发送消息，进入欢迎界面
    }
    public void goToWelcome()                          //进入欢迎界面的方法
    {
    ⋮   //代码省略
    }
    public void goToMainMenu()                         //进入主菜单界面的方法
    {
    ⋮   //代码省略
    }
    public void goTozzcxView()                         //进入站站查询界面
    {
    ⋮   //代码省略
    }
    public void goTocccxView()                         //进入车次查询界面
    {
    ⋮   //代码省略
    }

    public void initccSpinner()                        //初始化车次
    {
    ⋮   //代码省略
    }
    public void goTozdcccxView()                       //进入站点车次查询界面
    {
    ⋮   //代码省略
    }

    public void initzdSpinner()                        //初始化站点
    {
    ⋮   //代码省略
    }
    public void goToxtwhView()                         //进入系统维护界面
    {
    ⋮   //代码省略
    }
    public void goTocctjView()                         //进入车次添加界面
    {
    ⋮   //代码省略
    }
    public void goTocztjView()                         //进入车站添加界面
    {
    ⋮   //代码省略
    }
```

```
    public void goTogxtjView()                          //进入关系添加界面
    {
    ⋮   //代码省略
    }
    public void goToListView(String[][]mssg)            //进入查询结果界面
    {
    ⋮   //代码省略
    }
    public void goTogjxlView(String[][]mssg)            //某公交经过的所有站点，进入经过站点界面查询
    {
    ⋮   //代码省略
    }

    //查看在某个界面中单击“查询”按钮时，判断输入框是否为空
    public boolean isLegal()
    {
    ⋮   //代码省略
    }
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent e) //键盘监听
    {
    ⋮   //代码省略
    }

    public void iniTLisit()                             //初始化适配器中需要的数据函数
    {

    ⋮   //代码省略
    }

    public void iniTLisitarray(int id)                  //为对应 ID 的输入框添加适配器
    {
    ⋮   //代码省略
    }
}
```

上述的代码先进行了变量声明，并重写了 onCreate( )方法（执行该方法可以为系统初始化数据），接下来定义了切换到各个界面的方法，当需要切换到某个界面时，直接在消息处理器中调用对应方法即可。需要注意的是，调用站站查询方法时使用了适配器，调用车次查询和站点查询时使用了列表的初始化方法。此外，还定义了数据验证方法——当某个执行查询和某个输入框不能为空时，可以调用此方法进行校验。下面详细介绍上述框架中各个功能模块的具体实现方法。

20.4.1 goToWelcome( )方法

该方法用来将当前界面切换到欢迎界面。主要代码如下：

```
public void goToWelcome()
{
    if(wv==null)    //如果该对象没创建，则创建
    {
        wv=new Welcome(this);
    }
    setContentView(wv);
    curr=WhichView.WELCOME_VIEW;//标示当前所在界面
}
```

提示：要实现界面切换，在网页中使用超链接或重定位方法，在 Android 中则使用 setContentView()方法。

20.4.2　goToMainMenu()方法

该方法用来将当前界面切换到主菜单界面，并对各个按钮创建监听。主要代码如下：

```
public void goToMainMenu()
{
       setContentView(R.layout.main);       //切换界面
       curr=WhichView.MAIN_MENU;            //更改标示界面
       //获取主菜单界面中各个按钮的引用
       ImageButton ibzzcx=(ImageButton) findViewById(R.id.ibzzcx);
       ImageButton ibcccx=(ImageButton) findViewById(R.id.ibcccx);
       ImageButton ibczcccx=(ImageButton) findViewById(R.id.ibczcccx);
       ImageButton ibxtwh=(ImageButton) findViewById(R.id.ibxtwh);
       ImageButton ibabout=(ImageButton) findViewById(R.id.about_button);
       ImageButton ibhelp=(ImageButton) findViewById(R.id.help_button);
       ibabout.setOnClickListener           //关于按钮的监听
       (
          new OnClickListener()
          {
             public void onClick(View v)
             {
                hd.sendEmptyMessage(3);     //发消息进入关于界面
             }
          }
       );
       ibhelp.setOnClickListener            //帮助查询的监听
       (
          new OnClickListener()
          {
             public void onClick(View v)
             {
                hd.sendEmptyMessage(2);     //发消息进入帮助界面
             }
          }
       );
       ibzzcx.setOnClickListener            //站站查询按钮的监听
       (
          new OnClickListener()
          {
             public void onClick(View v)
             {
                goTozzcxView();             //进入站站查询模块
             }
          }
       );
       ibcccx.setOnClickListener            //车次查询按钮的监听
       (
          new OnClickListener()
          {
             public void onClick(View v)
             {
                goTocccxView();             //进入车次查询模块
             }
          }
       );
       ibczcccx.setOnClickListener          //站点所有车次查询的监听
       (
```

```
            new OnClickListener()
            {
                public void onClick(View v)
                {
                    goTozdcccxView();           //进入站点查询模块
                }
            }
        );
        ibxtwh.setOnClickListener               //系统维护按钮的监听
        (
            new OnClickListener()
            {
                public void onClick(View v)
                {
                    goToxtwhView();             //进入系统维护模块
                }
            }
        );
    }
```

提示：Handler( )方法是 Android 中的消息处理器，用于接收子线程发送的数据，并用此数据配合主线程更新界面。

20.4.3 goTozzcxView( )方法

该方法实现站站查询功能，并可选择是否设置中转站。主要代码如下：

```
    public void goTozzcxView()
    {
        setContentView(R.layout.zzcx);
        curr=WhichView.ZZCX_VIEW;
        flag=0;//标志位
        Button bcx=(Button) findViewById(R.id.zzcxbt);          //定义“查询”按钮
        Button bfh=(Button) findViewById(R.id.zzcxfhbt);        //定义“返回”按钮
        iniTLisitarray(R.id.EditText01);                        //为各个站点的输入框添加适配器
        iniTLisitarray(R.id.zzcxzzz);
        iniTLisitarray(R.id.zzcxzdz);
        final CheckBox zzzcx=(CheckBox)findViewById(R.id.zzcxzzzbt);   //中转站复选框的引用
        bcx.setOnClickListener          //为“查询”按钮添加监听
        (
            new OnClickListener()
            {
                @Override
                public void onClick(View v)
                {
                    if(!isLegal())
                    {
                        ⋮
                    }
                }
            }
        );
        bfh.setOnClickListener          //为“返回”按钮添加监听
        (
            new OnClickListener()
            {
                @Override
                public void onClick(View v)
                {
                    goToMainMenu();       //返回到主菜单界面
```

```
            }
        }
    );
    //建立适配器
}
```

提示：在上述代码中，为出发站、中转站、终点站都添加了一个适配器。这样，在输入拼音首字母时会出现一个下拉列表，供用户选择，从而方便了手机用户的输入。

20.4.4　goTocccxView()方法

该方法实现车次查询，选择所要查询的车次并确定后，返回该车次具体信息。主要代码如下：

```
public void goTocccxView()
{
    setContentView(R.layout.cccx);       //切换到车次查询界面
    curr=WhichView.CCCX_VIEW;            //标示界面
    flag=1;
    initccSpinner();                     //初始化下拉列表框
    List<String> linesList=DbUtil.searchccList();
    currcc=linesList.get(0);
    Button bcx=(Button) findViewById(R.id.cccx_cx);
    Button bfh=(Button) findViewById(R.id.cccx_fh);
    bcx.setOnClickListener
    (
        new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                if(!isLegal())           //如果各个输入框不满足规则，则返回
                {
                    return;
                }
                Vector<Vector<String>> temp= DbUtil.busSearch(currcc); //调用工具函数查询得到结果集
                currcc=null;
                if(temp.size()==0)       //如果结果向量长度为 0，说明没有查询结果，即无此车次相关信息
                    {
                        Toast.makeText(GJCXActivity.this, "没有相关信息!!!",
                        Toast.LENGTH_ SHORT).show();
                        return;
                    }                    //新建对应于向量的数组
                String[][] msgInfo=new String[temp.elementAt(0).size()][temp.size()];
                    for(int i=0;i<temp.size();i++)  //否则将向量中的数据导入对应的数组
                    {
                        for(int j=0;j<temp.elementAt(i).size();j++)
                        {
                            msgInfo[j][i]=(String)temp.get(i).get(j);
                        }
                    }
                    goToListView(msgInfo); //切换到结果显示界面（ListView 界面）
            }
        }
    );
    bfh.setOnClickListener               //为“返回”按钮添加监听
    (
        new OnClickListener()
        {
            @Override
```

```
            public void onClick(View v)
            {
                goToMainMenu();            //返回到菜单界面
            }
        }
    );

}
```

提示：在上述代码中也使用了方便输入的和有效输入的处理，即通过 initccSpinner()方法初始化下拉列表框数据。

20.4.5 goTozdcccxView()方法

该方法实现按站点查找经过本站点所有车次的功能。与 goTocccxView()方法的实现过程类似，并且也对下拉列表框进行了初始化，减少用户的输入。主要代码如下：

```
public void goTozdcccxView()
{
    setContentView(R.layout.zdcx);                    //切换到站点查询界面
    curr=WhichView.CZCCCX_VIEW;                       //标示界面
    flag=2;//标示所在界面为车次查询界面
    //为站点文本框添加适配器来完成文本输入的提示功能
    initzdSpinner();//初始化站点信息
    List<String> linesList=DbUtil.searchzdList();
    currzd=linesList.get(0);
    Button bcx=(Button) findViewById(R.id.zdcx_cx);   //获取“查询”按钮的引用
    Button bfh=(Button) findViewById(R.id.zdcx_fh);   //获取“返回”按钮的引用
    bcx.setOnClickListener                            //为“查询”按钮添加监听
    (
        new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                if(!isLegal())                        //如果某个文本框不合规则，则返回
                {
                    return;
                }
                ⋮
                goToListView(msgInfo);
            }
        }
    );
    bfh.setOnClickListener                            //为“返回”按钮添加监听
    (
        new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                goToMainMenu();                       //切换到主菜单界面
            }
        }
    );
}
```

20.4.6　goToListView()方法

该方法用于显示查询结果，双击某一查询结果，可以查看具体车次信息。主要代码如下：

```
public void goToListView(String[][]mssg)
{
    msgg=mssg;                                          //赋值给全局数组，用来实现“返回”按钮功能
    setContentView(R.layout.zzcxjg);                    //切换界面
    curr=WhichView.LIST_VIEW;                           //标示界面
    final String[][]msg=mssg;                           //新建数组，并赋值
    ListView lv_detail=(ListView)this.findViewById(R.id.ListView_detail);
    //获取 ListView 的引用
    BaseAdapter ba_detail=new BaseAdapter()             //新建适配器
    {
        @Override
        public int getCount()
        {
            return msg[0].length;                       //得到列表的长度
        }
        @Override
        public Object getItem(int arg0){return null;}
        @Override
        public long getItemId(int arg0){return 0;}
        @Override
        public View getView(int arg0, View arg1, ViewGroup arg2)
        //为每一项添加内容
        {
            LinearLayout ll_detail=new LinearLayout(GJCXActivity.this);
            ll_detail.setOrientation(LinearLayout.HORIZONTAL);
            //设置朝向
            ll_detail.setPadding(5,5,5,5);              //四周留白
            for(int i=0;i<msg.length;i++)               //为每一行设置显示的数据
            {
                TextView s= new TextView(GJCXActivity.this);
                s.setText(msg[i][arg0]);                //TextView 中显示的文字
                s.setTextSize(14);                      //字体大小
                s.setTextColor(getResources().getColor(R.color.black));  //字体颜色
                s.setPadding(1,2,2,1);                  //四周留白
                s.setWidth(60);//宽度
                s.setGravity(Gravity.CENTER);
                ll_detail.addView(s);                   //放入 LinearLayout
            }
            return ll_detail;                           //将此 LinearLayout 返回
        }
    };
    lv_detail.setAdapter(ba_detail);                    //将适配器添加进 ListView
    lv_detail.setOnItemClickListener                    //为列表添加监听
    (
        new OnItemClickListener()
        {
            @Override
            public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3)
            //arg2 为点击的第几项，当点击列表中的某一项时调用此函数
                {
                String cccx=msg[0][arg2];               //取出对应项中对应的车次信息
                Vector<Vector<String>> temp= DbUtil.getInfo(cccx);
                //查询该车次经过的所有站点
```

```
                if(temp.size()==0)                          //判断是否有查询结果
                {
                    Toast.makeText(GJCXActivity.this, "没有相关信息!!!", Toast.LENGTH_
                        SHORT).show();
                    return;
                }
                String[][] msgInfo=new String[temp.elementAt(0).size()][temp.size()];
                //如果有，则将结果放入对应的数组
                for(int i=0;i<temp.size();i++)
                {
                    for(int j=0;j<temp.elementAt(0).size();j++)
                    {
                        msgInfo[j][i]=(String)temp.get(i).get(j);
                    }
                }
                msgg=msg;
                goTogjxlView(msgInfo);                      //切换到车次具体情况显示界面
            }
        }
    );
}
```

20.4.7 goTogjxlView()方法

该方法实现把某一车次经过的站点显示出来。主要代码如下：

```
public void goTogjxlView(String[][]mssg)
{
    setContentView(R.layout.gjxl);                          //切换界面
    curr=WhichView.PASSbusstop_VIEW;                        //标示界面
    ListView lv_detail=(ListView)this.findViewById(R.id.ListView_passstation);
                                                            //得到 ListView 的引用
    final String[][]msg=mssg;
    BaseAdapter ba_detail=new BaseAdapter()                 //新建适配器
    {
        @Override
        public int getCount()
        {
            return msg[0].length;                           //得到列表的长度
        }
        @Override
        public Object getItem(int arg0){return null;}
        @Override
        public long getItemId(int arg0){return 0;}
        @Override
        public View getView(int arg0, View arg1, ViewGroup arg2)
        {
        ⋮
    };
    lv_detail.setAdapter(ba_detail);                        //将适配器添加进列表
}
```

20.4.8 goToxtwhView()方法

该方法实现切换到系统维护界面中，并建立相应功能按钮监听。主要代码如下：

```
public void goToxtwhView()
{
    setContentView(R.layout.xtwh);                              //切换到系统维护界面
```

```
        curr=WhichView.xtwh_VIEW;                                    //标示当前所在界面为系统维护界面
        ImageButton ibcctj=(ImageButton)findViewById(R.id.ibcctj);    //获取“车次添加”按钮引用
        ImageButton ibcztj=(ImageButton)findViewById(R.id.ibcztj);    //获取“车站添加”按钮引用
        ImageButton ibgxtj=(ImageButton)findViewById(R.id.ibgxtj);    //获取“关系添加”按钮的引用
        ibcctj.setOnClickListener                                    //“车次添加”按钮的监听
        (
           new OnClickListener()
           {
              public void onClick(View v)
              {
                 goTocctjView();                                     //进入车次添加界面
              }
           }
        );
        ibcztj.setOnClickListener                                    //“车站添加”按钮的监听
        (
           new OnClickListener()
           {
              public void onClick(View v)
              {
                 goTozdtjView();                                     //切换到车站添加界面
              }
           }
        );
        ibgxtj.setOnClickListener                                    //“关系添加”按钮的监听
        (
           new OnClickListener()
           {
              public void onClick(View v)
              {
                 goTogxtjView();
              }
           }
        );
    }
```

20.4.9　goTocctjView()方法

该方法实现车次添加。主要代码如下：

```
    public void goTocctjView()
    {
        setContentView(R.layout.cctj);                               //切换界面
        curr=WhichView.CCTJ_VIEW;                                    //标示界面
        Button bcctjtj=(Button)findViewById(R.id.cctj_tj);          //获取“添加”按钮的引用
        Button bcctjfh=(Button)findViewById(R.id.cctj_fh);          //获取“返回”按钮的引用
        iniTLisitarray(R.id.cctj_sfz);                               //为始发站文本框添加适配器
        iniTLisitarray(R.id.cctj_zdz);                               //为终点站文本框添加适配器
        final int Bid=DbUtil.getInserBid("cc","Bid")+1; //获取此时站点表中 Bid 列的最大 ID，然后加 1 得出
要插入此车次的 ID。
        bcctjtj.setOnClickListener                                   //为“添加”按钮添加监听
        (
           new OnClickListener()
           {
              @Override
              public void onClick(View v)
```

```
            {
                if(!isLegal())                              //判断输入框是否符合规则
                {
                    return;
                }
                ⋮
            }
        }
    );
    bcctjfh.setOnClickListener                              //为返回按钮添加监听
    (
        new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                goToxtwhView();                             //返回到系统维护界面
            }
        }
    );
}
```

20.4.10 goTozdtjView()方法

该方法实现站点添加。主要代码如下：

```
public void goTozdtjView()
{
    setContentView(R.layout.zdtj);//切换界面
    curr=WhichView.CZTJ_VIEW;//标示界面
    Button bcztjtj=(Button)findViewById(R.id.zdtj_tj); //获取“添加”按钮的引用
    Button bcztjfh=(Button)findViewById(R.id.zdtj_fh); //获取“返回”按钮的引用
    final int Sid=DbUtil.getInserBid("busstop","Sid")+1;//查出 Sid 列中最大的 ID，加 1 得到此时需要插入站点的 ID
    bcztjtj.setOnClickListener                              //为“添加”按钮添加监听
    (
        new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                if(!isLegal())
                {
                    return;
                }
                EditText cztjmc=(EditText)findViewById(R.id.et_cztj_czmc);//得到各输入框中的引用
                EditText cztjjc=(EditText)findViewById(R.id.et_cztj_czjc);
                String cnm=cztjmc.getText().toString().trim();//得到对应的文本
                String clx=cztjjc.getText().toString().trim();
                if(!clx.matches("[a-zA-Z]+"))//正则式匹配，查看简称输入框中的文本是否符合都是字母的规则
                {
                    //发不匹配消息
                    Toast.makeText(GJCXActivity.this, "对不起，简称只能为字母!!!",
                        Toast.LENGTH_SHORT).show();
                    return;
                }
                String sql="select * from busstop where Bsname='" +cnm+"'";
                Vector<Vector<String>> ss=query(sql);   //查看该站点是否已经存在
```

```
                if(ss.size()>0)                                    //如果结果向量的长度大于 0，说明已经有了该车
                {
                    Toast.makeText(GJCXActivity.this, "对不起，已经有了此站点!!!",
                        Toast. LENGTH_SHORT).show();
                    return;
                }
                sql="insert into busstop values(" +Sid +   ",'" +cnm +   "','" +   clx +"')";
                if(!insert(sql))                                   //进行插入操作，如果结果为“是”，则添加失败
                {
                    Toast.makeText(GJCXActivity.this, "对不起，添加失败!!!", Toast.LENGTH_SHORT). show();
                    return;
                }else{                                             //否则为添加成功
                    iniTLisit();
                    Toast.makeText(GJCXActivity.this, "恭喜你，添加成功!!!", Toast.LENGTH_SHORT). show();
                }
            }
        }
    );
    bcztjfh.setOnClickListener                                     //为“返回”按钮添加监听
    (
        new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
            goToxtwhView();                                        //返回到系统维护界面
            }
        }
    );
}
```

20.4.11　goTogxtjView( )方法

该方法实现在站点与车次上建立对应关系。主要代码如下：

```
public void goTogxtjView()
{
    setContentView(R.layout.gxtj);                                 //切换界面
    curr=WhichView.GXTJ_VIEW;                                      //标示界面
    Button bgxtjtj=(Button)findViewById(R.id.gxtj_tj);             //获取“添加”按钮的引用
    Button bgxtjfh=(Button)findViewById(R.id.gxtj_fh);             //获取“返回”按钮的引用
    iniTLisitarray(R.id.et_gxtj_zm);                               //为站点名称添加适配器

    bgxtjtj.setOnClickListener                                     //为“添加”按钮添加监听
    (
        new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                EditText gxtjcnm=(EditText)findViewById(R.id.et_gxtj_cm);
                //获取车名输入框的引用
                AutoCompleteTextView gxtjclx=(AutoCompleteTextView)findViewById(R.id.et_gxtj_zm);
                //获取站名输入框的引用
                EditText gxtjcsf=(EditText)findViewById(R.id.et_gxtj_dzsj);
                //拿到到站时间输入框的引用
                EditText gxtjczd=(EditText)findViewById(R.id.et_gxtj_kcsj);
                //拿到发车时间输入框的引用
                ⋮
```

```
            }
        }
    );
    bgxtjfh.setOnClickListener                          //为“返回”按钮添加监听
    (
        new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
            goToxtwhView();                             //返回到系统维护界面
            }
        }
    );
}
```

20.4.12 initccSpinner()方法

该方法实现把车次数据从数据库中取出，加载到车次选择列表中。主要代码如下：

```
public void initccSpinner()
{
    //初始化 Spinner
    Spinner sp=(Spinner)this.findViewById(R.id.cccxcc);
    final List<String> linesList=DbUtil.searchccList(); //查询车次表

    //为 Spinner 准备内容适配器
    BaseAdapter ba=new BaseAdapter()
    {
        @Override
        public int getCount() {
            return linesList.size();                    //总共 3 个选项
        }
        @Override
        public Object getItem(int arg0) { return null; }
        @Override
        public long getItemId(int arg0) { return 0; }

        @Override
        public View getView(int arg0, View arg1, ViewGroup arg2) {
            /*
             * 动态生成每个下拉项对应的 View，每个下拉项 View 由一个 TextView 构成
             */

            //初始化 TextView
            TextView tv=new TextView(GJCXActivity.this);
            tv.setText(linesList.get(arg0));            //设置内容
            tv.setTextSize(18);                         //设置字体大小
            tv.setTextColor(Color.BLACK);               //设置字体颜色
            return tv;
        }
    };
    sp.setAdapter(ba);                                  //为 Spinner 设置内容适配器
    //设置选项选中的监听器
    sp.setOnItemSelectedListener(
    new OnItemSelectedListener()
    {
        @Override
        public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2, long arg3)
        {                                               //重写选项被选中事件的处理方法
```

```
            TextView tvn=(TextView)arg1;                //获取其中的 TextView
            currcc=tvn.getText().toString();
        }
        @Override
        public void onNothingSelected(AdapterView<?> arg0) { }
        }
    );
}
```

20.4.13　initzdSpinner()方法

该方法实现从数据库中取出站点信息，加载到站点选择列表中。主要代码如下：

```
public void initzdSpinner()//初始化站点
{
    //初始化 Spinner
    Spinner sp=(Spinner)this.findViewById(R.id.czcxwb);
    final List<String> linesList=DbUtil.searchzdList();   //查询车次表
    //为 Spinner 准备内容适配器
    BaseAdapter ba=new BaseAdapter()
    {
        @Override
        public int getCount() {
            return linesList.size();                       //总共 3 个选项
        }
        @Override
        public Object getItem(int arg0) { return null; }
        @Override
        public long getItemId(int arg0) { return 0; }
        @Override
        public View getView(int arg0, View arg1, ViewGroup arg2) {
            /*
             * 动态生成每个下拉项对应的 View，每个下拉项 View 由一个 TextView 构成
             */

            //初始化 TextView
            TextView tv=new TextView(GJCXActivity.this);
            tv.setText(linesList.get(arg0));               //设置内容
            tv.setTextSize(18);                            //设置字体大小
            tv.setTextColor(Color.BLACK);                  //设置字体颜色
            return tv;
        }
    };
    sp.setAdapter(ba);//为 Spinner 设置内容适配器
    //设置选项选中的监听器
    sp.setOnItemSelectedListener(
    new OnItemSelectedListener()
    {
        @Override
        public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2, long arg3)
        {   //重写选项被选中事件的处理方法
            TextView tvn=(TextView)arg1;//获取其中的 TextView
            currzd=tvn.getText().toString();
        }
        @Override
        public void onNothingSelected(AdapterView<?> arg0) { }
    }
    );
}
```

20.4.14 isLegal()方法

该方法用于验证用户输入数据的有效性。主要代码如下：

```
    //查看在某个界面中单击“查询”按钮时，判断输入框是否为空
    public boolean isLegal()
    {
        if(curr==WhichView.ZZCX_VIEW)//如果当前为站站查询界面，对相应的文本框等进行合法验证
        {
            EditText etcfz=(EditText)findViewById(R.id.EditText01);     //出发站
            EditText etzzz=(EditText)findViewById(R.id.zzcxzzz);        //中转站
            EditText etzdz=(EditText)findViewById(R.id.zzcxzdz);        //终点站
            CheckBox cbzzz=(CheckBox)findViewById(R.id.zzcxzzzbt);      //中转站复选框
            if(etcfz.getText().toString().trim().equals(""))            //出发站为空
            {
                Toast.makeText(this, "出发站不能为空！！！",Toast.LENGTH_LONG).show();
                return false;
            }
            if(etzzz.getText().toString().trim().equals("")&&cbzzz.isChecked())//中转站为空
            {
                Toast.makeText(this, "中转站不能为空！！！",Toast.LENGTH_LONG).show();
                return false;
            }
            if(etzdz.getText().toString().trim().equals(""))            //终点站为空
            {
                Toast.makeText(this, "终点站不能为空！！！",Toast.LENGTH_LONG).show();
                return false;
            }
           if(etcfz.getText().toString().trim().contentEquals(etzdz.getText().toString(). trim()))
            //出发站和终点站相同
            {
                Toast.makeText(this, "出发站和终点站不能相同！！！",Toast.LENGTH_LONG).show();
                return false;
            }

if(cbzzz.isChecked()&&etcfz.getText().toString().trim().contentEquals(etzzz.getText().
            toString().trim()))                                         //出发站和中转站相同
            {
                Toast.makeText(this, "出发站和中转站不能相同！！！",Toast.LENGTH_LONG).show();
                return false;
            }
            if(cbzzz.isChecked()&&etzdz.getText().toString().trim().contentEquals
                (etzzz.getText().toString().trim()))
        //终点站和中转站相同
            {
                Toast.makeText(this, "终点站和中转站不能相同！！！",Toast.LENGTH_LONG).show();
                return false;
            }
        }
        if(curr==WhichView.CCTJ_VIEW)//如果当前为车次添加界面，对相应的文本框进行合法验证
        {
            EditText et_cm=(EditText)findViewById(R.id.cctj_cm);        //车名
            EditText et_lclx=(EditText)findViewById(R.id.cctj_lclx);    //公交类型
            EditText et_sfz=(EditText)findViewById(R.id.cctj_sfz);      //始发站
            EditText et_zdz=(EditText)findViewById(R.id.cctj_zdz);      //终点站
            if(et_cm.getText().toString().trim().contentEquals(""))
            {
                Toast.makeText(this, "车名不能为空！！！",Toast.LENGTH_SHORT).show();
                return false;
```

```
            }
            if(et_lclx.getText().toString().trim().contentEquals(""))
            {
                Toast.makeText(this, "公交类型不能为空！！！",Toast.LENGTH_SHORT).show();
                return false;
            }
            if(et_sfz.getText().toString().trim().contentEquals(""))
            {
                Toast.makeText(this, "始发站不能为空！！！",Toast.LENGTH_SHORT).show();
                return false;
            }
            if(et_zdz.getText().toString().trim().contentEquals(""))
            {
                Toast.makeText(this, "终点站不能为空！！！",Toast.LENGTH_SHORT).show();
                return false;
            }
        }
        if(curr==WhichView.CZTJ_VIEW)//如果当前在站点添加界面
        {
            EditText et_czmc=(EditText)findViewById(R.id.et_cztj_czmc);//站点名称
            EditText et_czjc=(EditText)findViewById(R.id.et_cztj_czjc);//站点简称
            if(et_czmc.getText().toString().trim().contentEquals(""))
            {
                Toast.makeText(this, "站点名称不能为空！！！",Toast.LENGTH_SHORT).show();
                return false;
            }
            if(et_czjc.getText().toString().trim().contentEquals(""))
            {
                Toast.makeText(this, "站点简称不能为空！！！",Toast.LENGTH_SHORT).show();
                return false;
            }
        }
        if(curr==WhichView.GXTJ_VIEW)//如果当前在关系添加界面
        {
            EditText et_cm=(EditText)findViewById(R.id.et_gxtj_cm);     //车名
            EditText et_zm=(EditText)findViewById(R.id.et_gxtj_zm);     //站名
            if(et_cm.getText().toString().trim().contentEquals(""))
            {
                Toast.makeText(this, "车名不能为空！！！",Toast.LENGTH_SHORT).show();
                return false;
            }
            if(et_zm.getText().toString().trim().contentEquals(""))
            {
                Toast.makeText(this, "站名不能为空！！！",Toast.LENGTH_SHORT).show();
                return false;
            }
        }
        return true;
    }
```

提示：数据有效性验证在程序中很有用，它是衡量一个程序健壮性和稳定性的标准之一，有些未经验证的数据可能会导致程序崩溃。isLegal()方法检验当前界面的文本框是否能为空，如果满足要求，返回 true，否则返回 false。

20.5　辅助界面的相关类

上节主要介绍了软件的界面主类，下面简单介绍与界面相关的其他类，主要有 WelcomeView 类、

GGview 类及 CityAdapter 类。

20.5.1 欢迎界面 WelcomeView 类

WelcomeView 类对应的是软件运行后出现的第一个界面——欢迎界面，主要作用是加载一个欢迎图片，给用户一种友好化的界面感觉。其主要代码如下：

```
public class WelcomeView extends View
{
    GJCXActivity activity;                                  //activity 的引用
    Paint paint;                                            //画笔
    int currentAlpha=0;                                     //当前的不透明值
    int screenWidth=480;                                    //屏幕宽度
    int screenHeight=320;                                   //屏幕高度
    int sleepSpan=50;                                       //动画的时延（ms）
    Bitmap[] logos=new Bitmap[1];//logo 图片数组
    Bitmap currentLogo;                                     //当前 logo 图片引用
    int currentX;                                           //图片位置
    int currentY;
    public Welcome(GJCXActivity activity)
    {
        super(activity);
        this.activity = activity;
        screenWidth=activity.display.getWidth();
        screenHeight=activity.display.getHeight();
        this.getHolder().addCallback(this);                 //设置生命周期回调接口的实现者
        paint = new Paint();                                //创建画笔
        paint.setAntiAlias(true);                           //打开抗锯齿
        //加载图片
        logos[0]=BitmapFactory.decodeResource(activity.getResources(), R.drawable.welcome);
    }
    public void onDraw(Canvas canvas)
    {
        //绘制黑色矩形
        paint.setColor(Color.BLACK);                        //设置画笔颜色
        paint.setAlpha(255);                                //设置不透明度为 255
        canvas.drawRect(0, 0, screenWidth, screenHeight, paint);
        //进行平面贴图
        if(currentLogo==null)return;
        paint.setAlpha(currentAlpha);
        canvas.drawBitmap(currentLogo, currentX, currentY, paint);
    }
    public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3)
    {
        ⋮
    }
    public void surfaceCreated(SurfaceHolder holder)  //创建时被调用
    {
        new Thread()
        {
            public void run()
            {
                ⋮
        }.start();
    }
    public void surfaceDestroyed(SurfaceHolder arg0)
    {    //销毁时被调用
```

```
        ⋮
    }
}
```

提示：在上述代码中我们仅为欢迎界面加载了一幅图片，如果要加载多张图片，则需要对 Bitmap[] logos=new Bitmap[1]的定义做相应更改。surfaceCreated()方法可以实现对加载图片进行渐变效果处理。

20.5.2　自定义控件 GGView 类

该类实现几张图片循环播放的幻灯效果，用于显示广告或人性化图片，增加软件的友好性。

```
public class GGView extends View {
    int COMPONENT_WIDTH;                                    //该控件宽度
    int COMPONENT_HEIGHT;                                   //该控件高度
    boolean initflag=false;                                 //是否要获取控件的高度和宽度标志
    static Bitmap[] bma;                                    //需要播放的图片的数组
    Paint paint;                                            //画笔
    int[] drawablesId;                                      //图片 ID 数组
    int currIndex=0;                                        //图片 ID 数组下标，根据此变量画图片
    boolean workFlag=true;                                  //播放图片线程标志位
    public GGView(Context father,AttributeSet as) {         //构造器
        super(father,as);
        drawablesId=new int[]{                              //初始化图片 ID 数组
            R.drawable.adv1,                                //将需要播放的图片 ID 放于此处
            R.drawable.adv2,
            R.drawable.adv3,
        };
        bma=new Bitmap[drawablesId.length];                 //创建存放图片的数组
        initBitmaps();                                      //调用初始化图片函数，初始化图片数组
        paint=new Paint();                                  //创建画笔
        paint.setFlags(Paint.ANTI_ALIAS_FLAG);              //消除锯齿
        new Thread(){                                       //创建播放图片线程
            public void run(){
                while(workFlag){
                    currIndex=(currIndex+1)%drawablesId.length;//改变 ID 数组下标值
                    GGView.this.postInvalidate();           //绘制
                    try {
                        Thread.sleep(3000);                 //休息 3s
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }}}}.start();                           //启动线程
    }
    public void initBitmaps(){                              //初始化图片函数
        Resources res=this.getResources();                  //获取 Resources 对象
        for(int i=0;i<drawablesId.length;i++){
            bma[i]=BitmapFactory.decodeResource(res, drawablesId[i]);
        }}
    public void onDraw(Canvas canvas){                      //绘制函数
        if(!initflag) {                                     //第一次绘制时需要获取宽度和高度
            COMPONENT_WIDTH=this.getWidth();                //获取 view 的宽度
            COMPONENT_HEIGHT=this.getHeight();              //获取 view 的高度
            initflag=true;
```

```
        }
        int picWidth=bma[currIndex].getWidth();        //获取当前绘制图片的宽度
        int picHeight=bma[currIndex].getHeight();      //获取当前绘制图片的高度
        int startX=(COMPONENT_WIDTH-picWidth)/2;       //得到绘制图片的左上角 x 轴坐标
        int startY=(COMPONENT_HEIGHT-picHeight)/2;     //得到绘制图片的左上角 y 轴坐标
        canvas.drawARGB(255, 200, 128, 128);           //设置背景色
        canvas.drawBitmap(bma[currIndex], startX,startY, paint);    //绘制图片
    }}
```

提示：这是一个自定义控件，我们可以用在任何需要的地方。

20.5.3 适配器 CityAdapter 类

该类用在站站查询时车站名称的输入，以增加软件操作的人性化和方便性。在用户输入框下出现一个下拉列表，显示用户可能的输入，供用户选择，这很贴合我们的记忆习惯（当我们记不准某个站点时，这样的功能就显得很重要），同时也提高了用户的输入效率。其主要代码如下：

```
public class CityAdapter<T> extends BaseAdapter implements Filterable
{
private List<T> mObjects;                              //车站名称 - 汉字数组
private List<T> mObjects2;                             //车站名称 - 拼音数组
private final Object mLock = new Object();
private int mResource;                                 //展示数组适配器内容的 View Id
private int mDropDownResource;                         //下拉框中内容的 Id
private int mFieldId = 0;                              //下拉框选项 ID
private boolean mNotifyOnChange = true;
private Context mContext;                              //当前上下文对象 - Activity
private ArrayList<T> mOriginalValues;                  //原始数组列表
private ArrayFilter mFilter;
private LayoutInflater mInflater;
public CityAdapter(Context context, int textViewResourceId, T[] objects,T[] objects2)
{
   init(context, textViewResourceId, 0, Arrays.asList(objects),Arrays.asList(objects2));
}
public void add(T object)
{
   ⋮
}
public void insert(T object, int index)
{
   ⋮
}
public void remove(T object)
{
   ⋮
}
public void clear()                                    //从列表中删除所有的信息
{
   ⋮
}
public void sort(Comparator<? super T> comparator)   //根据指定的比较器，对适配器中的内容进行排序
{
   Collections.sort(mObjects, comparator);
   if (mNotifyOnChange) notifyDataSetChanged();
```

```
    }
    @Override
    public void notifyDataSetChanged()
    {
        super.notifyDataSetChanged();
        mNotifyOnChange = true;
    }
    //设置自动修改
    public void setNotifyOnChange(boolean notifyOnChange)
    {
        mNotifyOnChange = notifyOnChange;
    }
    //构造器初始化所有信息
    private void init(Context context, int resource, int textViewResourceId, List<T> objects ,List<T>
objects2)
    {
        mContext = context;
        mInflater = (LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        mResource = mDropDownResource = resource;
        mObjects = objects;
        mObjects2 = objects2;
        mFieldId = textViewResourceId;
    }
    //返回与数组适配器相关联的上下文对象
    public Context getContext()
    {
        return mContext;
    }
    public int getCount()                                    //返回车站名称汉字列表的大小
    {
        return mObjects.size();
    }

    public T getItem(int position)                           //返回车站名称汉字列表中指定位置的字符串值
    {
        return mObjects.get(position);
    }
    public int getPosition(T item)                           //返回车站名称汉字列表中指定字符串值的索引
    {
        return mObjects.indexOf(item);
    }
    public long getItemId(int position)                      //将 int 型整数以长整型返回
    {
        return position;
    }
    public View getView(int position, View convertView, ViewGroup parent)//创建 View
    {
        return createViewFromResource(position, convertView, parent, mResource);
    }
    private View createViewFromResource(int position, View convertView, ViewGroup parent,int resource)
    {
        View view;
        TextView text;
        if (convertView == null)                             //如果当前为空
        {
            view = mInflater.inflate(resource, parent, false);
```

```
        }
        else                                                    //如果不为空
        {
            view = convertView;
        }

        try {
            if (mFieldId == 0)                                  //如果当前域为空，假定所有的资源就是一个 TextView
            {
                text = (TextView) view;
            }
            else                                                //否则,在界面中找到 TextView
            {
                text = (TextView) view.findViewById(mFieldId);
            }
        }
        catch (ClassCastException e)                            //异常处理
        {
            throw new IllegalStateException
            (
            "ArrayAdapter requires the resource ID to be a TextView", e
            );
        }
        text.setText(getItem(position).toString());             //为 Text 设值,返回当前车站名称汉字列表中选中的值
        return view;
    }
    public void setDropDownViewResource(int resource)    //创建下拉视图
    {
        this.mDropDownResource = resource;
    }
    @Override
    public View getDropDownView(int position, View convertView, ViewGroup parent)
    {
        return createViewFromResource(position, convertView, parent, mDropDownResource);
    }
    public static ArrayAdapter<CharSequence> createFromResource(Context context,int textArrayResId,
int textViewResId)
        //从外部资源中创建新的数组适配器
    {
        CharSequence[] strings = context.getResources().getTextArray(textArrayResId);
        //创建字符串序列
        return new ArrayAdapter<CharSequence>(context, textViewResId, strings);
        //返回数组适配器
    }
    public Filter getFilter()                                   //得到过滤器
    {
        if (mFilter == null)                                    //如果为空，创建数组过滤器
        {
            mFilter = new ArrayFilter();
        }
        return mFilter;
    }
    //数组过滤器限制数组适配器以指定的前缀开头，如果与提供的前缀不匹配则将其从中删除
    private class ArrayFilter extends Filter
    {
        @Override
```

```
protected FilterResults performFiltering(CharSequence prefix)//执行过滤
{
    FilterResults results = new FilterResults();//创建 FilterResults 对象
    if (mOriginalValues == null)                  //如果为空
    {
        synchronized(mLock)
        {
            mOriginalValues = new ArrayList<T>(mObjects);
        }
    }
    if (prefix == null || prefix.length() == 0)
    {
        synchronized (mLock)
        {
            ArrayList<T> list = new ArrayList<T>(mOriginalValues);
            results.values = list;
            results.count = list.size();
        }
    }
    else
    {
        String prefixString = prefix.toString().toLowerCase();  //转换成小写
        final ArrayList<T> values = mOriginalValues;
        final int count = values.size();
        final ArrayList<T> newValues = new ArrayList<T>(count);
        for (int i = 0; i < count; i++)
        {
            final T value = values.get(i);
            final String valueText = value.toString().toLowerCase();
            final T value2 = mObjects2.get(i);
            final String valueText2 = value2.toString().toLowerCase();
            //查找拼音
            if(valueText2.startsWith(prefixString))
            {
                newValues.add(value);
            }                                          //查找汉字
            else if(valueText.startsWith(prefixString))
            {
                newValues.add(value);
            }
            else
            {    //添加汉字关联
                final String[] words = valueText.split(" ");
                final int wordCount = words.length;
                for (int k = 0; k < wordCount; k++)
                {
                    if (words[k].startsWith(prefixString))
                    {
                        newValues.add(value);
                        break;
                    }
                }
                //添加拼音关联汉字
                final String[] words2 = valueText2.split(" ");
                final int wordCount2 = words2.length;
                for (int k = 0; k < wordCount2; k++) {
                    if (words2[k].startsWith(prefixString))
```

```
                        {
                            newValues.add(value);
                            break;
                        }
                    }
                }
            }
            results.values = newValues;
            results.count = newValues.size();
        }
        return results;
    }
    @SuppressWarnings("unchecked")
    protected void publishResults(CharSequence constraint, FilterResults results)
    {
        mObjects = (List<T>) results.values;
        if (results.count > 0)
        {
            notifyDataSetChanged();
        } else
        {
            notifyDataSetInvalidated();
        }
    }
}
}
```

20.6 数据库操作相关类

本软件主要涉及数据库创建类和数据库操作类。

20.6.1 数据库表的创建——CreatTable 类

该类用于本软件数据初始化，可以通过执行建立表和插入数据的 SQL 语句来实现建表和数据初始化功能。主要代码如下：

```
package com.gjcx;
public class CreatTable {
    public static void creattable(){
        try{
            String sqll[]=new String[]{
                   "create table if not exists bus " + //建立车次表
                   "(Bid integer primary key,Bname char(20)," +
                   "Bstart char(20),Bend char(20),Btype char(20))", "create  table if not exists
                   busstop(Sid integer primary key," +
                   "Bsname char(20),Sbs char(10))",    //建立站点表
                   "create table if not exists relation" +
                   "(Rid integer primary key,Bid integer,Sid integer,Rarrivetime " +
                   "char(20),Rstarttime char(20))",    //建立关系表
                   //插入一些初始化数据
                   "insert into bus values(10001,'9','火车站','刘庄','空调车')",
                   ⋮
```

```
                "insert into busstop values(1,'车次站','hcz')",
                ⋮
                "insert into relation values(1,10001,1,'','6:00')",
                ⋮
            };
            for(String o:sqll){ //循环所有 SQL 语句，进行建表和初始化数据的操作
                DbUtil.createTable(o);
            }
        }catch(Exception e){
            e.printStackTrace();
        }}}
```

20.6.2　数据库操作——LoadUtil 类

该类是一个功能类，实现对数据库的操作，可以在需要的地方调用。

```
public class LoadUtil {
    public static SQLiteDatabase createOrOpenDatabase()   //连接数据库
    {
        SQLiteDatabase sld=null;
        try{
            sld=SQLiteDatabase.openDatabase                //连接并创建数据库，如果不存在则创建
            (
                "/data/data/com.gjcx/mydb",
                null,
                SQLiteDatabase.OPEN_READWRITE|SQLiteDatabase.CREATE_IF_NECESSARY);
        }catch(Exception e)
        {
            e.printStackTrace();
        }
        return sld;//返回该连接
    }
    public static void createTable(String sql){           //创建表
        SQLiteDatabase sld=createOrOpenDatabase();         //连接数据库
        try{
            sld.execSQL(sql);//执行 SQL 语句
            sld.close();//关闭连接
        }catch(Exception e){
        }
    }
    public static boolean insert(String sql)               //插入数据
    {
        ⋮
    }
    /*====================================begin=========================================*/
    //获得线路名称列表
    public static List<String> searchccList()
    {
        ⋮
    }
    /*====================================begin=========================================*/
    //获得线路名称列表
```

```
        public static List<String> searchzdList()
        {
            ⋮
        }
        public  static Vector<Vector<String>> query(String sql)    //查询
        {
            ⋮
        }
        //查找某车经过的所有车站
        public static Vector<Vector<String>> getInfo(String Bname)
        {
            ⋮
        }
        //站站查询
        public static Vector<Vector<String>> getSameVector(String start,String end)
        {
            ⋮
        }
        //某车次的情况，初始站和末尾站还有车型等时间
        public static Vector<Vector<String>> busSearch(String Bname)
        {//车次查询
            ⋮
        }
        public static Vector<Vector<String>> busstopSearch(String busstop) //根据车站名字,查询经过车
站的所有车
        {   //车站查询，得到经过每一辆车的到站时间和出站时间
            //查询有关车次的信息
            ⋮
        }
        //查询其插入表项 ID 的最大值
        public static int getInsertBid(String name,String Bid)
        {
            ⋮
        }
    }
```

createOrOpenDatabase()方法用来连接数据库；createTable()方法用于创建表；insert()方法用来执行数据插入操作；query()方法用来执行查询操作；getInfo()方法实现检索车次详细情况；getSameVector()方法实现站站查询操作；busSearch()方法实现在数据库中检索车次；busstopSearch()方法实现通过传入站点名称，检索经过该站点的所有车次，再结合首班时间和末班时间等信息；getInsertBid()方法实现插入数据时初始化 ID。