第 25 章

网络编程及 Internet 应用

智能手机的一个主要功能就是可以访问互联网，大多数 App 都需要通过互联网执行某种网络通信，因此网络支持对于手机 App 来说是尤为重要的。Android 平台在网络编程和 Internet 应用上也是非常优秀的。本章将对 Android 中的网络编程和 Internet 应用的相关知识进行详细介绍。

25.1 通过 HTTP 访问网络

在 Android 中也可以使用 HTTP 协议访问网络。例如，在使用应用宝 App 下载游戏时（如图 25.1 所示），或者刷新朋友圈时（如图 25.2 所示），都需要通过 HTTP 协议访问网络。

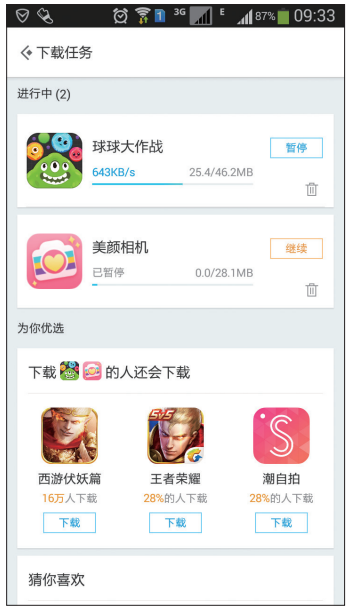

图 25.1　下载游戏

图 25.2　刷新朋友圈

在 Android 中提供了两个用于 HTTP 通信的 API，即 HttpURLConnection 和 Apache 的 HttpClient。由于 Android 6.0 版本已经基本将 HttpClient 从 SDK 中移除了。所以这里主要介绍 HttpURLConnection。

HttpURLConnection 类位于 java.net 包中，用于发送 HTTP 请求和获取 HTTP 响应。由于该类是抽象类，不能直接实例化对象，则需要使用 URL 的 openConnection() 方法来获得。例如，要创建一个“http://www.mingribook.com”网站对应的 HttpURLConnection 对象，可以使用下面的代码：

```
01  URL url = new URL("http://www.mingribook.com/");
02  HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
```

HttpURLConnection 是 URLConnection 的一个子类，它在 URLConnection 的基础上提供了如表 25.1 所示的方法，从而方便发送和响应 HTTP 请求。

表 25.1　HttpURLConnection 常用的方法

方　法	描　述
int getResponseCode()	获取服务器的响应代码
String getResponseMessage()	获取服务器的响应消息
String getRequestMethod()	获取发送请求的方法
void setRequestMethod(String method)	设置发送请求的方法

创建了 HttpURLConnection 对象后，就可以使用该对象发送 HTTP 请求了。

25.1.1 发送 GET 请求

使用 HttpURLConnection 对象发送请求时，默认发送的就是 GET 请求。因此，发送 GET 请求比较简单，只需要在指定连接地址时，先将要传递的参数通过“?参数名=参数值”的形式进行传递（多个参数间使用英文半角的“&”符号分隔。例如，要传递用户名和 E-mail 地址这两个参数，可以使用 ?user=wgh&email= wgh717@sohu.com 实现），然后获取流中的数据，并关闭连接即可。

说明 使用HTTP协议访问网络就是客户端与服务器的通信，所以运行本章实例不仅需要创建客户端App实例，还需要创建简单的后台服务器。

注意 （1）永远不要在主线程上执行网络调用。（2）在Service而不是Activity中执行网络操作。

下面通过一个实例来说明如何使用 HttpURLConnection 发送 GET 请求。

例 25.1 使用 GET 方式发表并显示微博信息

在 Android Studio 中创建一个 Module，名称为“GET Request”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为垂直线性布局管理器并为其设置背景图片，删除默认添加的 TextView 组件，然后添加一个编辑框（用于输入微博内容）以及一个“发表”按钮，再添加一个滚动视图，在该视图中添加一个文本框，用于显示从服务器上读取的微博内容。

（2）打开主活动 MainActivity，该类继承 Activity，定义所需的成员变量，具体代码如下：

```
01  private EditText content;                    //定义一个输入文本内容的编辑框对象
02  private Handler handler;                     //定义一个android.os.Handler对象
03  private String result = "";                  //定义一个代表显示内容的字符串
```

（3）创建 base64() 方法，对传递的参数进行 Base64 编码，用于解决乱码问题。具体代码如下：

```
01 public String base64(String content) {                    //对字符串进行Base64编码
02     try {
03         //对字符串进行Base64编码
04         content = Base64.encodeToString(content.getBytes("utf-8"), Base64.DEFAULT);
05         content = URLEncoder.encode(content, "utf-8");//对字符串进行URL编码
06     } catch (UnsupportedEncodingException e) {
07         e.printStackTrace();
08     }
09     return content;
10 }
```

说明 要解决应用GET方法传递中文参数时产生乱码的问题，也可以使用Java提供的URLEncoder类来实现。

（4）创建 send() 方法，用于建立一个 HTTP 连接，并将输入的内容发送到 Web 服务器，再读取服务器的处理结果，具体代码如下：

```
01 public void send() {
02     String target = "";
03     target = "http://192.168.1.198:8080/example/get.jsp?content="
04             + base64(content.getText().toString().trim()); //要访问的URL地址
05     URL url;
06     try {
07         url = new URL(target);
08         HttpURLConnection urlConn = (HttpURLConnection) url
09                 .openConnection();                          //创建一个HTTP连接
10         InputStreamReader in = new InputStreamReader(
11                 urlConn.getInputStream());                  //获得读取的内容
12         BufferedReader buffer = new BufferedReader(in);     //获取输入流对象
13         String inputLine = null;
14         //通过循环逐行读取输入流中的内容
15         while ((inputLine = buffer.readLine()) != null) {
16             result += inputLine + "\n";
17         }
18         in.close();                                         //关闭字符输入流对象
19         urlConn.disconnect();                               //断开连接
20     } catch (MalformedURLException e) {
21         e.printStackTrace();
22     } catch (IOException e) {
23         e.printStackTrace();
24     }
25 }
```

说明 根据当前计算机的IP和Tomcat服务器的端口号设置要访问的URL地址。上面代码中的192.168.1.198是当前计算机的IP地址，8080是Tomcat服务器的端口号。

（5）在 onCreate() 方法中，首先在发表按钮的单击事件中，判断输入内容是否为空，然后创

建 Handler 对象并重写 handleMessage() 方法，用于更新 UI 界面，最后创建新的线程，用于从服务器中获取相关数据，具体代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //获取输入文本内容的EditText组件
    content = (EditText) findViewById(R.id.content);
    //获取显示结果的TextView组件
    final TextView resultTV = (TextView) findViewById(R.id.result);
    Button button = (Button) findViewById(R.id.button);    //获取"发表"按钮组件
    //单击"发表"按钮，实现读取服务器微博信息
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //判断输入内容是否为空，为空时给出提示消息，否则访问服务器
            if ("".equals(content.getText().toString())) {
                Toast.makeText(MainActivity.this, "请输入要发表的内容！",
                        Toast.LENGTH_SHORT).show();       //显示消息提示
                return;
            }
            handler = new Handler() {                //将服务器中的数据，显示在UI界面中
                @Override
                public void handleMessage(Message msg) {
                    super.handleMessage(msg);
                }
            };
            new Thread(new Runnable() { //创建一个新线程，用于发送并读取微博信息
                public void run() {
                    send();              //调用send()方法，用于发送文本内容到Web服务器
                    Message m = handler.obtainMessage();  //获取一个Message
                    handler.sendMessage(m);               //发送消息
                }
            }).start();                                   //开启线程
        }
    });
}
```

（6）重写 Handler 对象中的 handleMessage() 方法，在该方法中实现将服务器中的数据，显示在 UI 界面中，关键代码如下：

```
if (result != null) {          //如果服务器返回结果不为空
    resultTV.setText(result);  //显示获得的结果
    content.setText("");       //清空文本框
}
```

（7）由于在本实例中需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网络资源的权限，具体代码如下：

```
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
```

（8）创建 Web 应用，用于接收 Android 客户端发送的请求，并做出响应。这里在 Tomcat 安装路径下的 webapps 目录中创建一个名称为 example 的文件夹，再在该文件夹中创建一个名称为 get.jsp 的文件，用于获取参数 content 指定的微博信息，并输出转码后的 content 变量的值，具体代码如下：

```
<%@page contentType="text/html; charset=utf-8" language="java" import="sun.misc.BASE64Decoder"%>
<%
String content=request.getParameter("content");                    //获取输入的微博信息
 if(content!=null){
    BASE64Decoder decoder=new BASE64Decoder();
    content=new String(decoder.decodeBuffer(content),"utf-8");     //进行base64解码
String date=new java.util.Date().toLocaleString();                 //获取系统时间
%>
<%="[马 云]于 "+date+" 发表一条微博，内容如下："%>
<%=content%>
<% }%>
```

说明 可以将云盘需要部署到Tomcat下的文件\example文件夹放到Tomcat安装路径下的webapps目录中，并启动Tomcat服务器，然后运行本实例。

（9）运行本实例，将显示如图 25.3 所示。

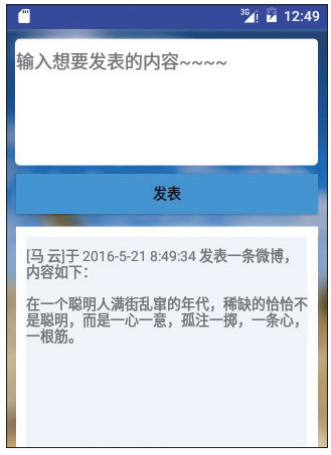

图 25.3 使用 GET 方式发表并显示微博信息

25.1.2 发送 POST 请求

在 Android 中，使用 HttpURLConnection 类发送请求时，默认采用的是 GET 请求，如果要发送 POST 请求，需要通过其 setRequestMethod() 方法进行指定。例如，创建一个 HTTP 连接，并为该连接指定请求的发送方式为 POST，可以使用下面的代码：

```
HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();//创建一个HTTP连接
urlConn.setRequestMethod("POST");                                  //指定请求方式为POST
```

发送 POST 请求要比发送 GET 请求复杂一些，它需要通过 HttpURLConnection 类及其父类 URLConnection 提供的方法设置相关内容，常用的方法如表 25.2 所示。

表 25.2　发送 POST 请求时常用的方法

方　法	描　述
setDoInput(boolean newValue)	用于设置是否向连接中写入数据，如果参数值为 true，表示写入数据；否则不写入数据
setDoOutput(boolean newValue)	用于设置是否从连接中读取数据，如果参数值为 true，表示读取数据；否则不读取数据
setUseCaches(boolean newValue)	用于设置是否缓存数据，如果参数值为 true，表示缓存数据；否则表示禁用缓存
setInstanceFollowRedirects(boolean followRedirects)	用于设置是否应该自动执行 HTTP 重定向，参数值为 true 时，表示自动执行；否则不自动执行
setRequestProperty(String field, String newValue)	用于设置一般请求属性，例如，要设置内容类型为表单数据，可以进行以下设置 setRequestProperty(“Content-Type”,” application/x-www-form-urlencoded”)

下面通过一个具体的实例来介绍如何使用 HttpURLConnection 类发送 POST 请求。

例 25.2　使用 POST 方式登录 QQ

在 Android Studio 中创建 Module，名称为“POST Request”。实现本实例的具体步骤如下：

（1）打开主活动 MainActivity，该类继承 Activity，定义所需的成员变量，具体代码如下：

```
01  private EditText edit_Username;          //定义一个输入用户名的编辑框组件
02  private EditText edit_Password;          //定义一个输入密码的编辑框组件
03  private Handler handler;                 //定义一个android.os.Handler对象
04  private String result = "";              //定义一个代表显示内容的字符串
```

（2）创建 send() 方法，用于建立一个 HTTP 连接，并将输入的内容发送到 Web 服务器，再读取服务器的处理结果，具体代码如下：

```
01  public void send() {
02      String target = "http://192.168.1.198:8080/example/post.jsp";//要提交的服务器地址
03      URL url;
04      try {
05          url = new URL(target);                                    //创建URL对象
06          //创建一个HTTP连接
07          HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
08          urlConn.setRequestMethod("POST");                         //指定使用POST请求方式
09          urlConn.setDoInput(true);                                 //向连接中写入数据
10          urlConn.setDoOutput(true);                                //从连接中读取数据
11          urlConn.setUseCaches(false);                              //禁止缓存
12          urlConn.setInstanceFollowRedirects(true);                 //自动执行HTTP重定向
13          urlConn.setRequestProperty("Content-Type",
14                  "application/x-www-form-urlencoded");             //设置内容类型
```

```
        DataOutputStream out = new DataOutputStream(
                urlConn.getOutputStream());                          //获取输出流
        //连接要提交的数据
        String param = "username="
                + URLEncoder.encode(edit_Username.getText().toString(), "utf-8")
                + "&password="
                + URLEncoder.encode(edit_Password.getText().toString(), "utf-8");
        out.writeBytes(param);          //将要传递的数据写入数据输出流
        out.flush();                    //输出缓存
        out.close();                    //关闭数据输出流
        if (urlConn.getResponseCode() == HttpURLConnection.HTTP_OK) {//判断是否响应成功
            InputStreamReader in = new InputStreamReader(
                    urlConn.getInputStream());                       //获得读取的内容
            BufferedReader buffer = new BufferedReader(in);          //获取输入流对象
            String inputLine = null;
            //通过循环逐行读取输入流中的内容
            while ((inputLine = buffer.readLine()) != null) {
                result += inputLine;
            }
            in.close();                                              //关闭字符输入流
        }
        urlConn.disconnect();                                        //断开连接
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

（3）在 onCreate() 方法中，首先在登录按钮的单击事件中，判断账号和密码是否为空，然后创建 Handler 对象并重写 handleMessage() 方法，用于更新 UI 界面，最后创建新的线程，用于从服务器中获取相关数据，关键代码如下：

```
edit_Username = (EditText) findViewById(R.id.username); //获取用于输入账号的编辑框组件
edit_Password = (EditText) findViewById(R.id.password); //获取用于输入密码的编辑框组件
ImageButton btn_Login = (ImageButton) findViewById(R.id.login);//获取用于登录的按钮控件
//单击"登录"按钮，发送信息与服务器交互
btn_Login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //当用户名、密码为空时给出相应提示
        if ("".equals(edit_Username.getText().toString())
                || "".equals(edit_Password.getText().toString())) {
            Toast.makeText(MainActivity.this, "请填写账号或密码！",
                    Toast.LENGTH_SHORT).show();
            return;
        }
```

```
        handler = new Handler() {
            @Override
            public void handleMessage(Message msg) {
                super.handleMessage(msg);
            }
        };
        new Thread(new Runnable() {//创建一个新线程，用于从网络上获取数据
            public void run() {
                send();             //调用send()方法，用于将账号和密码发送到Web服务器
                Message m = handler.obtainMessage();     //获取一个Message对象
                handler.sendMessage(m);                  //发送消息
            }
        }).start();                                      //开启线程
    }
});
```

（4）重写 Handler 对象中的 handleMessage() 方法，在该方法中实现通过服务器中返回的数据判断是否显示登录后界面，关键代码如下：

```
//如果服务器返回值为"ok"，则表示账号和密码输入正确
if ("ok".equals(result)) {
    //跳转登录后界面
    Intent in = new Intent(MainActivity.this, MessageActivity.class);
    startActivity(in);
}else {
    //账号、密码错误的提示信息
    Toast.makeText(MainActivity.this, "请填写正确的账号和密码！",
            Toast.LENGTH_SHORT).show();
}
```

（5）由于在本实例中需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网络资源的权限，具体代码如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

（6）创建 Web 应用，用于接收 Android 客户端发送的请求，并做出响应。这里编写一个名称为 post.jsp 的文件。在该文件中首先获取客户端填写的账号和密码，然后判断账号和密码是否正确，如果账号和密码正确，则向客户端传递通过指令“ok”。具体代码如下：

```
<%@ page contentType="text/html; charset=utf-8" language="java" %>
<%String password=request.getParameter("password");           //获取输入的密码
String username=request.getParameter("username");             //获取输入的用户名
if(password!=null && username!=null){
username=new String(username.getBytes("iso-8859-1"),"utf-8");  //对用户名进行转码
password=new String(password.getBytes("iso-8859-1"),"utf-8");  //对密码进行转码
if("mr".equals(username)&&"mrsoft".equals(password)){
%>
```

```
09  <%="ok"%>
10  <%}%>
11  <%}%>
```

说明 将post.jsp文件放到Tomcat安装路径下的“webapps\example”目录中，并启动Tomcat服务器，然后运行本实例。

（7）运行本实例，在显示的界面中输入账号“mr”与密码“mrsoft”，如图 25.4 所示，单击“登录”按钮，通过服务器判断账号和密码正确后显示登录后界面，如图 25.5 所示。

图 25.4 登录界面

图 25.5 登录后的界面

25.2 通过 OkHttp3 访问网络

25.2.1 OkHttp3 简介

除了可以使用 HttpURLConnection 进行网络访问以外，还可以使用一种更加方便、简单的方式来进行网络的访问，那就是 OkHttp3，它是由 Square 公司开发的开源网络通讯库。OkHttp3 的基本特点有：

- 支持 HTTP 协议 2.0，并允许连接到同一个主机地址的所有请求共享套接字。
- 如果 HTTP 协议 2.0 不可用，连接池也可以减少请求延迟。
- 支持 GZIP, 可以压缩下载体积。
- 响应缓存可以完全避免重复请求的网络。

在使用 OkHttp3 开源网络通讯库之前，需要在 gradle 文件的 dependencies 节点中添加依赖库代码。具体代码如下：

```
compile 'com.squareup.okhttp3:okhttp:插入新的版本'
```

在 AndroidManifest.xml 文件中添加网络权限的代码，具体代码如下：

```
01  <!--网络权限-->
02  <uses-permission android:name="android.permission.INTERNET"/>
```

25.2.2 OkHttp3 的基本用法

OkHttp3 的请求方式有以下两种：

◆ 使用 GET 方式进行网络请求可以使用以下代码：

```
//创建OkHttpClient对象
OkHttpClient okHttpClient = new OkHttpClient();
//创建请求对象
Request request = new Request.Builder()
        .url("请求地址")
        .build();
try {
    //创建请求响应
    Response response = okHttpClient.newCall(request).execute();
    response.body().string();     //获取响应资源
} catch (IOException e) {
    e.printStackTrace();
}
```

◆ 使用 POST 方式进行网络请求可以使用以下代码：

```
//创建OkHttpClient 对象
OkHttpClient okHttpClient = new OkHttpClient();
//Form表单格式的参数传递
FormBody formBody = new FormBody.Builder()
        .add("参数名","参数值").build();
//创建请求对象
Request request = new Request.Builder()
        .url("请求地址")
        .post(formBody)
        .build();
try {
    //创建请求响应
    Response response = okHttpClient.newCall(request).execute();
    response.body().string();     //获取响应资源
} catch (IOException e) {
    e.printStackTrace();
}
```

POST 网络请求与 GET 网络请求类似，只是多了 1 个 FormBody 用于添加请求的参数，然后传递给 Request。

下面通过一个具体的实例演示 OkHttp3 的具体应用。

例 25.3 使用 OkHttp3 下载网络图片

在 Android Studio 中创建 Module，名称为“OkHttp3”，实现本实例的具体步骤如下：

（1）打开 build.gradle（Module：OkHttp3）文件，在 dependencies 节点中添加 OkHttp3 的依赖

库代码，然后在 AndroidManifest.xml 文件中添加网络权限的代码。

（2）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后添加 1 个用于显示下载图片的 ImageView 组件，再添加 1 个 Button 组件用于下载按钮。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!--显示下载的图片-->
    <ImageView
        android:id="@+id/image1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:src="@mipmap/bbs_new" />
    <!--下载按钮-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="down"
        android:text="下载图片" />
</RelativeLayout>
```

（3）打开主活动 MainActivity.java 文件，定义所需要的全局变量并创建 1 个 Handler 对象用于显示下载的图片，然后在 onCreate() 方法中创建进度对话框用于显示下载图片时的等待进度，然后获取用于显示图片的 ImageView 组件。具体代码如下：

```
public class MainActivity extends AppCompatActivity {
    private ProgressDialog dialog;          //下载等待对话框
    private ImageView image;                //显示下载图片的组件
    private Handler handler=new Handler(){
        @Override
        public void handleMessage(Message msg) {
            dialog.dismiss();               //关闭弹窗
            //显示下载的图片
            image.setImageBitmap((Bitmap) msg.obj);
        }
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        dialog = new ProgressDialog(this);              //创建进度对话框
        dialog.setTitle("提示");                         //设置弹窗的标题
        dialog.setMessage("下载图片中,请稍等！");          //设置对话框提示内容
        image = (ImageView) findViewById(R.id.image1); //获取显示图片的组件
    }
}
```

（4）创建 1 个名称为 down 的方法，该方法是下载按钮的单击事件，在该方法中首先创建 OkHttpClient 对象，然后创建网络请求，再创建请求呼叫，最后在 onResponse() 方法中将响应数据转换为 Bitmap 位图数据并通过 Handler 发送该数据。具体代码如下：

```
public void down(View view) {
    dialog.show();//显示进度对话框
    //创建OkHttpClient对象
    OkHttpClient okHttpClient = new OkHttpClient();
    //创建网络请求
    Request request = new Request.Builder()
            .get()
            .url("http://test.mingrisoft.com//Public/" +
                    "uploads/book_pic/590af7401f562.png")
            .build();
    //创建请求呼叫
    Call call = okHttpClient.newCall(request);
    call.enqueue(new Callback() {
        //无法执行请求时调用
        @Override
        public void onFailure(Call call, IOException e) {
        }
        //返回HTTP响应时调用
        @Override
        public void onResponse(Call call, Response response) throws IOException {
            //将响应数据转换为输入流数据
            InputStream inputStream = response.body().byteStream();
            //将输入流数据转换为Bitmap位图数据
            Bitmap bitmap = BitmapFactory.decodeStream(inputStream);
            //创建Message消息并发送
            Message msg = Message.obtain();
            msg.obj = bitmap;
            handler.sendMessage(msg);
        }
    });
}
```

（5）运行本实例，将显示如图 25.6 所示。单击下载图片按钮将显示如图 25.7 所示的下载等待对话框。下载完成后将自动显示如图 25.8 所示下载完成的网络图片。

图 25.6 显示默认图片

图 25.7 显示下载等待对话框

图 25.8 显示下载完成的网络图片

25.3 解析 JSON 格式数据

25.3.1 JSON 简介

JSON（JavaScript Object Notation）是一种轻量级的数据交换格式，语法简洁，不仅易于阅读和编写，而且也易于机器的解析和生成。

JSON 通常由两种数据结构组成，一种是对象（“名称 / 值”形式的映射），另一种是数组（值的有序列表）。JSON 没有变量或其他控制，只用于数据传输。

（1）对象

在 JSON 中，可以使用下面的语法格式来定义对象。

```
{"属性1":属性值1,"属性2":属性值2……"属性n":属性值n}
```

◆ 属性 1~ 属性 n：用于指定对象拥有的属性名。

◆ 属性值 1~ 属性值 n：用于指定各属性对应的属性值，其值可以是字符串、数字、布尔值（true/false）、null、对象和数组。

例如，定义一个保存名人信息的对象，可以使用下面的代码：

```
01 {
02     "name":"扎克伯格",
03         "address":"United States New York",
04         "wellknownsaying":"当你有使命，它会让你更专注"
05 }
```

（2）数组

在 JSON 中，可以使用下面的语法格式来定义对象。

```
{"数组名":[
```

```
            对象1,对象2……,对象n
]}
```

◆ 数组名：用于指定当前数组名。

◆ 对象 1~ 对象 n：用于指定各数组元素，它的值为合法的 JSON 对象。

例如，定义一个保存名人信息的数组，可以使用下面的代码：

```
{"famousPerson":[
  {"name":"扎克伯格","address":" 美国 "," wellknownsaying ":"当你有使命，它会让你更专注"},
  {"name":"马云","address":"中国"," wellknownsaying ":"心中无敌者，无敌于天下"}
]}
```

25.3.2 解析 JSON 数据

在 Android 的官网中提供了解析 JSON 数据的 JSONObject 和 JSONArray 对象。其中，JSONObject 用于解析 JSON 对象；JSONArray 用于解析 JSON 数组。下面将通过一个具体的实例说明如何解析 JSON 数据。

例 25.4 获取 JSON 数据，显示计步器的个人信息

在 Android Studio 中创建 Module，名称为“Analysis Of JSON Data”。实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为垂直线性布局管理器，然后添加 8 个 TextView 组件，用于显示计步器的 8 个信息值。

（2）打开主活动 MainActivity，该类继承 Activity，定义所需的成员变量，关键代码如下：

```
private Handler handler;          //定义一个android.os.Handler对象
private String result = "";       //定义一个代表显示内容的字符串
```

（3）创建 send() 方法，实现发送请求并获取 JSON 数据，关键代码如下：

```
public void send() {                    //创建send()方法，实现发送请求并获取JSON数据
    //要发送请求的服务器地址
    String target = "http://192.168.1.198:8080/example/index.json";
    URL url;
    try {
        url = new URL(target);                              //创建URL对象
        //创建一个HTTP连接
        HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
        urlConn.setRequestMethod("POST");                   //指定使用POST请求方式
        urlConn.setDoOutput(true);                          //从连接中读取数据
        urlConn.setUseCaches(false);                        //禁止缓存
        urlConn.setInstanceFollowRedirects(true);           //自动执行HTTP重定向
        InputStreamReader in = new InputStreamReader(
                urlConn.getInputStream());                  //获得读取的内容
        BufferedReader buffer = new BufferedReader(in); //获取输入流对象
        String inputLine = null;
```

```
        //通过循环逐行读取输入流中的内容
        while ((inputLine = buffer.readLine()) != null) {
            result += inputLine;
        }
        in.close();                                          //关闭输入流
        urlConn.disconnect();                                //断开连接
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

（4）在 onCreate() 方法中，首先创建 Handler 对象并重写 handleMessage() 方法，用于更新 UI 界面，然后创建新的线程，用于从服务器中获取 JSON 数据，关键代码如下：

```
final TextView step = (TextView) findViewById(R.id.text1); //获取TextView显示单日步数
final TextView time = (TextView) findViewById(R.id.text2); //获取TextView显示单日时间
final TextView heat = (TextView) findViewById(R.id.text3); //获取TextView显示单日热量
final TextView km = (TextView) findViewById(R.id.text4);   //获取TextView显示单日公里数
final TextView step1 = (TextView) findViewById(R.id.text5);//获取TextView显示每周步数
final TextView time1 = (TextView) findViewById(R.id.text6);//获取TextView显示每周时间
final TextView heat1 = (TextView) findViewById(R.id.text7);//获取TextView显示每周热量
final TextView km1 = (TextView) findViewById(R.id.text8);  //获取TextView显示每周公里数
handler = new Handler() {                                  //解析返回的JSON串数据并显示
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
    }
};
new Thread(new Runnable() {                 //创建一个新线程，用于从服务器中获取JSON数据
    public void run() {
        send();                             //调用send()方法，用于发送请求并获取JSON数据
        Message m = handler.obtainMessage();            //获取一个Message对象
        handler.sendMessage(m);                         //发送消息
    }
}).start();                                             //开启线程
```

（5）重写 Handler 对象中的 handleMessage() 方法，在该方法中实现解析返回的 JSON 串数据并显示，关键代码如下：

```
//创建TextView二维数组
TextView[][] tv = {{step, time, heat, km}, {step1, time1, heat1, km1}};
try {
    JSONArray jsonArray = new JSONArray(result);   //将获取的数据保存在JSONArray数组中
    for (int i = 0; i < jsonArray.length(); i++) { //通过for循环遍历JSON数据
        JSONObject jsonObject = jsonArray.getJSONObject(i);  //解析JSON数据
```

```
            tv[i][0].setText(jsonObject.getString("step"));       //获取JSON中的步数值
            tv[i][1].setText(jsonObject.getString("time"));       //获取JSON中的时间值
            tv[i][2].setText(jsonObject.getString("heat"));       //获取JSON中的热量值
            tv[i][3].setText(jsonObject.getString("km"));         //获取JSON中的公里数
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
```

（6）由于在本实例中需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网络资源的权限，具体代码如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

（7）创建 Web 服务器，用于接收 Android 客户端发送的请求，并做出响应。这里编写一个名称为 index.json 的文件。在该文件中编写要返回的 JSON 数据，具体代码如下：

```
[{"step":"12,672","time":"1h 58m","heat":"306","km":"8.3"},
{"step":"73,885","time":"11h 41m","heat":"1,771","km":"48.7"}]
```

说明 将index.json文件放到Tomcat安装路径下的webapps\example目录中，并启动Tomcat服务器，然后运行本实例。

（8）运行本实例，将显示如图 25.9 所示。

图 25.9 显示计步器的个人信息

25.3.3 使用 GSON 解析数据

GSON 是 Google 提供的一个 Java 类库，用于将 Java 对象转换为 JSON 数据，也可以将一个 JSON 字符串转换为对应的 Java 对象。不过 GSON 并没有被 Android 官方添加在 API 当中，所以在使用 GSON 类库时需要在 gradle 文件的 dependencies 节点中添加 GSON 库的依赖代码。具体代码如下：

```
compile'com.google.code.gson:gson:插入新的版本'
```

也可以在 GSON 的官方网站中下载 GSON 的 jar 包，来使用 GSON 类库。

GSON 类库在解析 JSON 数据时更加的方便、简单，它可以将一段 JSON 格式的字符串自动映射成一个对象，所以再也不需要手动去编写代码进行解析了。例如，一段 JSON 格式的数据如下所示：

```
{
        "id":101,
        "name":"Summer",
        "age":28,
        "height":1.75
    }
```

解析单个对象首先需要创建 1 个 Information 类，在该类中定义 ID、名字、年龄、身高，并且为这些属性设置 get() 方法。具体代码如下：

```
public class Information {
    public int id;                  //定义ID
    public String name;             //定义名字
    public int age;                 //定义年龄
    public float height;            //定义身高
    public int getId() {            //获取ID
        return id;
    }
    public String getName() {       //获取姓名
        return name;
    }
    public int getAge() {           //获取年龄
        return age;
    }
    public float getHeight() {      //获取身高
        return height;
    }
}
```

然后创建 Gson 对象，再通过 fromJson() 方法进行 json 数据的解析，最后通过 Information 类中的 getId() 方法获取 json 数据中的 ID 值。具体代码如下：

```
Gson gson=new Gson();                                          //创建Gson对象
Information info=gson.fromJson(json数据,Information.class);    //解析json数据
info.getId();                                                  //获取json数据中的ID
```

通常情况下 JSON 数据包含多个对象，如果解析一段 JSON 数组就需要使用 TypeToken 类来帮忙，将需要解析的数据类型传入到 fromJson() 方法中。

下面通过一个具体的实例演示 GSON 解析数据的应用。

例 25.5 将 json 序列变为 list 对象

在 Android Studio 中创建 Module，名称为“GSON”，实现本实例的具体步骤如下（本实例在 25.3.2 小节中的实例基础上进行修改）：

（1）打开 build.gradle（Module：GSON）文件，在 dependencies 节点中添加 GSON 与 OkHttp3 的依赖库代码。然后在 AndroidManifest.xml 文件中添加网络权限的代码。

（2）在“java/com.mingrisoft”包中创建 Information.java 类，在该类中定义步数、时间、热量、公里数，，并且为这些属性设置 get() 方法。具体代码如下：

```
01  public class Information {
02      public String step;             //定义步数
03      public String time;             //定义时间
04      public String heat;             //定义热量
05      public String km;               //定义公里数
06      public String getStep() {       //获取步数
07          return step;
08      }
09      public String getTime() {       //获取时间
10          return time;
11      }
12      public String getHeat() {       //获取热量
13          return heat;
14      }
15      public String getKm() {         //获取公里数
16          return km;
17      }
18  }
```

（3）打开主活动 MainActivity.java 文件，修改 send() 方法中的代码，在该方法中通过 OkHttp3 实现向服务器发送获取 JSON 信息的网络请求。关键代码如下：

```
01  public void send() {
02      //要发送请求的服务器地址
03      String target = "http://192.168.1.198:8080/example/index.json";
04      //创建OkHttpClient对象
05      OkHttpClient okHttpClient = new OkHttpClient();
06      //创建网络请求
07      Request request = new Request.Builder()
08              .get()
09              .url(target)
10              .build();
11      try {
12          //创建请求响应
13          Response response = okHttpClient.newCall(request).execute();
14          result = response.body().string();   //获取json数据
15      } catch (IOException e) {
16          e.printStackTrace();
17      }
18  }
```

（4）修改 handleMessage() 方法中的代码，在该方法中首先创建 1 个用于解析 JSON 数据的类型，然后通过 fromJson() 方法进行 JSON 数据的解析，最后将解析的数据显示在 TextView 文本框组件当中。关键代码如下：

```
handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        //创建TextView二维数组
        TextView[][] tv = {{step, time, heat, km}, {step1, time1, heat1, km1}};
        //创建解析JSON的类型
        Type listType = new TypeToken<ArrayList<Information>>(){}.getType();
        //解析JSON数据
        ArrayList<Information> foos = new Gson().fromJson(result,listType);
        for (int i = 0; i < foos.size(); i++) {          //通过for循环遍历JSON数据
            tv[i][0].setText(foos.get(i).getStep());     //获取JSON中的步数值
            tv[i][1].setText(foos.get(i).getTime());     //获取JSON中的时间值
            tv[i][2].setText(foos.get(i).getHeat());     //获取JSON中的热量值
            tv[i][3].setText(foos.get(i).getKm());       //获取JSON中的公里数
        }
        super.handleMessage(msg);
    }
};
```

（5）运行本实例，如图 25.10 所示，显示解析后的 JSON 数据。

图 25.10　显示解析后的 JSON 数据

说明　由于该实例是在25.3.2小节中的实例基础上进行修改，所以同样需要启动Tomcat服务器，然后运行本实例。

25.4 使用 WebView 显示网页

Android 提供了内置的浏览器，该浏览器使用了开源的 WebKit 引擎。WebKit 不仅能够搜索网址、查看电子邮件，而且能够播放视频节目。在 Android 中，要使用内置的浏览器，需要通过 WebView 组件来实现。通过 WebView 组件可以轻松实现显示网页功能。例如，通过 WebView 来实现上网功能，通过 QQ 浏览器显示明日学院主页，效果如图 25.11 所示。

图 25.11　通过 QQ 浏览器显示网页

25.4.1　使用 WebView 组件浏览网页

WebView 组件是专门用来浏览网页的，其使用方法与其他组件一样，既可以在 XML 布局文件中使用 <WebView> 标记添加，又可以在 Java 文件中通过 new 关键字创建。推荐采用 <WebView> 标记在 XML 布局文件中添加。在 XML 布局文件中添加一个 WebView 组件可以使用下面的代码：

```
<WebView
    android:id="@+id/webView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

添加 WebView 组件后，就可以应用该组件提供的方法执行浏览器操作。WebView 组件提供的常用方法如表 25.3 所示。

表 25.3　WebView 组件提供的常用方法

方　法	描　述
loadUrl(String url)	用于加载指定 URL 对应的网页
loadData(String data, String mimeType, String encoding)	用于将指定的字符串数据加载到浏览器中
loadDataWithBaseURL(String baseUrl, String data, String mimeType, String encoding, String historyUrl)	用于基于 URL 加载指定的数据
capturePicture()	用于创建当前屏幕的快照
goBack()	执行后退操作，相当于浏览器上的后退按钮的功能
goForward()	执行前进操作，相当于浏览器上的前进按钮的功能

续表

方　法	描　述
stopLoading()	用于停止加载当前页面
reload()	用于刷新当前页面

下面通过一个实例来说明如何使用 WebView 组件浏览网页。

例 25.6 使用 WebView 组件浏览网页

在 Android Studio 中创建 Module，名称为“WebView”。实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并删除默认添加的 TextView 组件，然后添加一个 WebView 组件，关键代码如下：

```
<WebView
    android:id="@+id/webView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

（2）打开主活动 MainActivity，该类继承 Activity，在 onCreate() 方法中，首先获取布局管理器中添加的 WebView 组件，然后为 WebView 指定要加载网页的 URL 地址，最后设置加载内容自适应屏幕，具体代码如下：

```
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //设置全屏显示
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        //获取布局管理器中添加的WebView组件
        WebView webView = (WebView) findViewById(R.id.webView1);
        webView.loadUrl("http://www.mingrisoft.com/Bbs.html");//指定要加载的网页
        webView.getSettings().setUseWideViewPort(true);  //设置此属性，可任意比例缩放
        webView.getSettings().setLoadWithOverviewMode(true);//设置加载内容自适应屏幕
    }
}
```

（3）由于在本实例中需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网络资源的权限，具体代码如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

说明 如果想让WebView组件具有放大和缩小网页的功能，则要进行以下设置：

```
webView.getSettings().setSupportZoom(true);
webView.getSettings().setBuiltInZoomControls(true);
```

（4）运行本实例，将显示如图 25.12 所示。

图 25.12 使用 WebView 组件浏览网页

25.4.2 使用 WebView 加载 HTML 代码

在进行 Android 开发时，对于一些游戏的帮助信息，使用 HTML 代码进行显示比较实用，不仅可以让界面更加美观，而且可以让开发更加简单、快捷。WebView 组件提供了 loadData() 和 loadDataWithBaseURL() 方法来加载 HTML 代码。loadData() 方法一般很少使用，因为使用该方法加载带中文的 HTML 内容时会产生乱码，而使用 loadDataWithBaseURL() 方法就不会出现这种情况。loadDataWithBaseURL() 方法的基本语法格式如下：

```
loadDataWithBaseURL(String baseUrl, String data, String mimeType, String encoding,
            String historyUrl)
```

loadDataWithBaseURL() 方法各参数的说明如表 25.4 所示。

表 25.4 loadDataWithBaseURL() 方法的参数说明

参　数	描　述
baseUrl	用于指定当前页使用的基本 URL。如果为 null，则使用默认的 about:blank，即空白页
data	用于指定要显示的字符串数据
mimeType	用于指定要显示内容的 MIME 类型。如果为 null，则默认使用 text/html
encoding	用于指定数据的编码方式
historyUrl	用于指定当前页的历史 URL，也就是进入该页前显示页的 URL。如果为 null，则使用默认的 about:blank

下面通过一个具体的实例来说明如何使用 WebView 组件加载 HTML 代码。

例 25.7 实现使用 WebView 组件加载 HTML 游戏指南界面

在 Android Studio 中创建 Module，名称为“WebView And HTML”，实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器，然后将 TextView 组件删除，再添加一个 Button 组件，用于单击后跳转到游戏指南界面。

（2）创建一个名称为 HelpActivity 的 Activity，修改布局文件 activity_help.xml，将默认添加的布局管理器修改为相对布局管理器，然后添加一个 WebView 组件，用于加载 HTML 代码编写的游戏指南界面。

（3）打开 HelpActivity 类，该类继承 Activity，在 onCreate() 方法中，首先获取布局管理器中添加的 WebView 组件，然后创建一个字符串构建器，将要显示的 HTML 内容放置在该构建器中，最后通过 loadDataWithBaseURL() 方法加载数据，具体代码如下：

```
01  public class HelpActivity extends Activity {
02      @Override
03      protected void onCreate(Bundle savedInstanceState) {
04          super.onCreate(savedInstanceState);
05          setContentView(R.layout.activity_help);
06          //设置全屏显示
07          getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
08                  WindowManager.LayoutParams.FLAG_FULLSCREEN);
09          //获取布局管理器中添加的WebView组件
10          WebView webview = (WebView) findViewById(R.id. webView1);
11          webview.setBackgroundResource(R.drawable. bg_help);//设置WebView背景图片
12          webview.setBackgroundColor(0);                    //设置WebView背景色为透明
13          //创建一个字符串构建器，将要显示的HTML内容放置在该构建器中
14          StringBuilder sb = new StringBuilder();
15          sb.append("<br>");
16          sb.append("<br>");
17          sb.append("<span style=font-size:20px>" +
18                  "<div>疯狂动物来找茬操作指南：</div></span>");
19          sb.append("<ul>");
20          sb.append("<span style=font-size:20px><li>一共三关。</li></span>");
21          sb.append("<br>");
22          sb.append("<span style=font-size:20px>" +
23                  "<li>找出两张图片的5处不同点。</li></span>");
24          sb.append("<br>");
25          sb.append("<span style=font-size:20px>" +
26                  "<li>剩余时间越长，分数越高。</li></span>");
27          sb.append("<br>");
28          sb.append("<span style=font-size:20px>" +
29                  "<li>每过完一个关卡将询问是否进入</li></span>");
30          sb.append("<span style=font-size:20px>下一关。</span>");
31          sb.append("</ul>");
32          //加载数据
33          webview.loadDataWithBaseURL(null, sb.toString(), "text/html", "utf-8", null);
34      }
35  }
```

（4）打开 MainActivity 类，该类继承 Activity，在 onCreate() 方法中，实现单击“游戏玩法”按钮后跳转到游戏操作指南页面，具体代码如下：

```
01  public class MainActivity extends Activity {
02      @Override
03      public void onCreate(Bundle savedInstanceState) {
04          super.onCreate(savedInstanceState);
05          setContentView(R.layout.activity_main);
06          //设置全屏显示
07          getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
08                  WindowManager.LayoutParams.FLAG_FULLSCREEN);
09          Button btn= (Button) findViewById(R.id.btn_help);//获取布局文件中的游戏玩法按钮
10          //实现单击按钮跳转游戏指南页面
11          btn.setOnClickListener(new View.OnClickListener() {
12              @Override
13              public void onClick(View v) {
14                  //设置通过Intent跳转游戏指南页面
15                  Intent intent = new Intent(MainActivity.this, HelpActivity.class);
16                  startActivity(intent);
17              }
18          });
19      }
20  }
```

（5）在 AndroidManifest.xml 文件的 <activity> 标记中添加 screenOrientation 属性，分别设置 MainActivity 与 HelpActivity 横屏显示，关键代码如下：

```
android:screenOrientation="landscape"
```

（6）运行本实例，将显示如图 25.13 所示。单击“游戏玩法”按钮，将显示如图 25.14 所示的运行结果。

图 25.13　游戏初始界面

图 25.14　使用 WebView 加载 HTML 界面

25.4.3　让 WebView 支持 JavaScript

在默认的情况下，WebView 组件是不支持 JavaScript 的，但是在运行某些不得不使用 JavaScript 代码的网站时，需要让 WebView 支持 JavaScript。例如，在图 25.15 中显示的是明日图书网的登录页面，

如果在填写登录信息时只输入昵称，此时单击“登录”按钮，就会弹出如图 25.16 所示的提示框，该提示框就是网页中通过 JavaScript 代码实现的。

图 25.15 网站登录页面

图 25.16 弹出 JavaScript 提示框

实际上，让 WebView 组件支持 JavaScript 比较简单，只需要以下两个步骤就可以实现：

（1）使用 WebView 组件的 WebSettings 对象提供的 setJavaScriptEnabled() 方法让 JavaScript 可用。例如，存在一个名称为 webview 的 WebView 组件，要设置在该组件中允许使用 JavaScript，可以使用的代码如下：

```
webView.getSettings().setJavaScriptEnabled(true);          //设置JavaScript可用
```

（2）经过以上设置后，网页中的大部分 JavaScript 代码均可用。但是，对于通过 window.alert() 方法弹出的对话框并不可用。要想显示弹出的对话框，需要使用 WebView 组件的 setWebChromeClient() 方法来处理 JavaScript 的对话框，具体代码如下：

```
webView.setWebChromeClient(new WebChromeClient());        //设置处理JavaScript中的对话框
```

这样设置后，再使用 WebView 显示带弹出 JavaScript 对话框的网页时，网页中弹出的对话框将不会被屏蔽掉。下面通过一个具体的实例来说明如何让 WebView 支持 JavaScript。

例 25.8 WebView 加载 QQ 空间“写说说”界面

在 Android Studio 中创建 Module，名称为“WebView And JavaScript”。实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，首先将默认添加布局管理器修改为相对布局管理器，然后将 TextView 组件删除，再添加一个 WebView 组件，用于加载 JavaScript 页面。

（2）打开主活动 MainActivity，该类继承 Activity，在 onCreate() 方法中，首先获取布局管理器中添加的 WebView 组件，然后设置 JavaScript 可用，再设置处理 JavaScript 中的对话框，最后指定要加载的网页，具体代码如下：

```
01  public class MainActivity extends Activity {
02      @Override
03      protected void onCreate(Bundle savedInstanceState) {
04          super.onCreate(savedInstanceState);
05          setContentView(R.layout.activity_main);
06          //获取布局文件中的WebView组件
07          WebView webView = (WebView) findViewById(R.id.webView1);
08          //设置JavaScript可用
09          webView.getSettings().setJavaScriptEnabled(true);
10          //设置处理JavaScript中的对话框
11          webView.setWebChromeClient(new WebChromeClient());
12          //指定要加载的网页
13          webView.loadUrl("http://192.168.1.198:8080/example/javascript.jsp");
14      }
15  }
```

（3）由于在本实例中需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网络资源的权限，具体代码如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

说明 将javascript.jsp文件放到Tomcat安装路径下的“webapps\example”目录中，并启动Tomcat服务器，然后运行本实例。

（4）运行本实例，如果未输入任何内容，直接单击“发表”按钮后将弹出一个提示对话框，如图 25.17 所示。

图 25.17　让 WebView 支持 JavaScript