

第 15 章 移动电子相册：捕捉精彩瞬间

移动电子相册是移动开心网中最吸引人的功能之一。将电子相册放在手机上，可以随时随地捕捉精彩瞬间。在移动电子相册中使用了大量的 Android 组件以外，还涉及网络与拍照功能。本章将详细介绍这些技术的使用方法。

本章的主要知识点如下：

- Button 组件
- EditText 组件
- Spinner 组件
- TabHost 组件
- HttpURLConnection 类
- TextView 组件
- ImageView 组件
- Gallery 组件
- HttpGet 类和 HttpPost 类

15.1 先睹为快：移动电子相册

移动电子相册是移动开心网中的重要组成部分。通过移动电子相册，可以随时随地拍摄照片，而且在有信号的情况下就可以上传到网络上与他人共享。在本节先看一下移动电子相册的效果图。本章后面的部分将详细介绍移动电子相册中涉及的组件及网络技术，并介绍如何利用这些技术实现移动电子相册。图 15.1 和图 15.2 分别显示了在手机上浏览照片和上传照片的主界面。



图 15.1 浏览相册中的照片



图 15.2 上传照片



15.2 移动电子相册中使用的组件

在移动电子相册中主要使用了 Button、EditText、TextView、ImageView、Spinner、Gallery 和 TabHost 等组件，在本节将详细介绍这些组件的用法。

15.2.1 Button 组件

Button 是 Android 中最常用的组件之一。Button 组件在 XML 布局文件中需要使用<Button>标签表示，下面是一段标准的 Button 组件定义代码。

```
<Button android:id="@+id/button1" android:layout_width="wrap_content"  
        android:layout_height="wrap_content" android:text="我的按钮 1" />
```

最常用的按钮事件是单击事件，可以通过 Button 类的 setOnClickListener 方法设置处理单击事件的对象实例，如果当前的类实现了 android.view.View.OnClickListener 接口，那么可以直接由 this 传入 setOnClickListener 方法，代码如下：

```
Button button1 = (Button) findViewById(R.id.button1);  
button1.setOnClickListener(this);
```

下面的代码中包含了两个按钮，并在单击事件中通过 value 变量来控制按钮放大或缩小（value=1 表示放大，value=-1 表示缩小），代码如下：

```
private int value = 1;  
@Override  
public void onClick(View view)  
{  
    Button button = (Button) view;  
    // 如果按钮宽度等于屏幕宽度，按钮开始缩小  
    if (value == 1 && button.getWidth() == getWindowManager().getDefaultDisplay().  
    getWidth())  
        value = -1;  
    // 如果按钮宽度小于 100，按钮开始放大  
    else if (value == -1 && button.getWidth() < 100)  
        value = 1;  
    // 以按钮宽度和高度的 10% 放大或缩小按钮  
    button.setWidth(button.getWidth() + (int) (button.getWidth() * 0.1) * value);  
    button.setHeight(button.getHeight() + (int) (button.getHeight() * 0.1) * value);  
}
```

15.2.2 TextView 组件

在前面介绍的移动开心网的登录界面已经使用过 TextView 组件，但只涉及了 TextView 组件非常初级的用法。TextView 组件的功能远不止显示文本这么简单，在本节将更进一步介绍 TextView 组件的功能。

TextView 组件的基本用法在前面已经多次接触到了，下面再来看看一下。TextView 组件



人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS



使用<TextView>标签定义，下面的代码是 TextView 组件最基本的用法。



```
<TextView android:id="@+id/textview1" android:layout_width="fill_parent"  
         android:layout_height="wrap_content" android:text="可以在这里设置  
TextView 组件的文本" />
```



上面的代码表示 TextView 的宽度应尽可能充满 TextView 组件所在的容器。将高度设为 wrap_content，表示 TextView 组件的高度需要根据组件中文本的行数、字体大小等因素决定。



当然，还可以对 TextView 组件进行更复杂的设置，例如，设置 TextView 组件的文字字体大小、文字颜色、背景颜色、文本距 TextView 组件边缘的距离、TextView 组件距其他组件的距离等。下面的代码包含了 3 个<TextView>标签，这 3 个标签设置了上述的 TextView 组件的相应属性。



```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
              android:orientation="vertical" android:layout_width="fill_parent"  
              android:layout_height="fill_parent">  
    <TextView android:id="@+id/textview1" android:layout_width="fill_parent"  
             android:layout_height="wrap_content" android:textColor="#0000FF"  
             android:background="#FFFFFF" android:text="可以在这里设置 TextView 组件的文本"  
    />  
    <TextView android:id="@+id/textview2" android:layout_width="fill_parent"  
             android:layout_height="wrap_content" android:text="更复杂的设置"  
             android:textSize="20dp" android:textColor="#FF00FF"  
             android:background="#FFFFFF" android:padding="30dp" android:layout_margin  
             ="30dp"/>  
    <TextView android:id="@+id/textview3" android:layout_width="fill_parent"  
             android:layout_height="wrap_content" android:textColor="#FF0000"  
             android:background="#FFFFFF" android:text="可以在这里设置 TextView 组件的文本"  
    />  
</LinearLayout>
```



上面代码中大多数属性的含义从字面上就可以猜出来，但要注意两个属性： android:padding 和 android:layout_margin，其中 android:padding 属性用于设置文字距 TextView 组件边缘的距离， android:layout_margin 属性用于设置 TextView 组件距离相邻其他组件的距离。这两个属性设置的都是四个方向的距离，也就是上、下、左、右的距离。如果要单独设置这四个方向的距离，可以使用其他的属性，这些属性名字的规则是在这两个属性后面添加 Left、Right、Top 和 Bottom，例如，设置 TextView 组件距离左侧的组件的距离，可以使用 android:layout_marginLeft 属性。

要注意的是，由于第 2 个<TextView>标签的 android:layout_width 属性值是 fill_parent，因此，文字距 TextView 组件右侧的距离并不是 android:padding 属性的值。系统会优先使用 android:layout_width 属性的值来设置 TextView 组件到右侧组件（这里是屏幕的右边缘）的距离。

除了可以在 XML 布局文件中设置 TextView 组件的属性外，还可以在代码中设置 TextView 组件的属性（实际上，所有的组件都可以采用这两种方式设置它们的属性）。例如，下面的代码设置了文本的颜色。





```
TextView textView = (TextView) findViewById(R.id.textview4);
textView.setTextColor(android.graphics.Color.RED); // 使用实际的颜色值设置字体颜色
```

设置 TextView 组件背景色的方法有 3 个，这些方法如下。

- setBackgroundResource：通过颜色资源 ID 设置背景色。
- setBackgroundColor：通过颜色值设置背景色。
- setBackgroundDrawable：通过 Drawable 对象设置背景色。

下面的代码分别演示了如何用这 3 个方法来设置 TextView 组件的背景色。

使用 setBackgroundResource 方法设置背景色：

```
textView.setBackgroundResource(R.color.background);
```

使用 setBackgroundColor 方法设置背景色：

```
textView.setBackgroundColor(android.graphics.Color.RED);
```

使用 setBackgroundDrawable 方法设置背景色：

```
Resources resources=getBaseContext().getResources();
Drawable drawable=resources.getDrawable(R.color.background);
textView.setBackgroundDrawable(drawable);
```



15.2.3 EditText 组件

EditText 是 TextView 的子类，因此，EditText 组件具有 TextView 组件的一切 XML 属性及方法。EditText 与 TextView 的区别是 EditText 组件可以输入文本，而 TextView 只能显示文本。

注意

虽然 TextView 通过设置某些属性也可以输入文本，但 TextView 组件的文本输入功能并不完善，需要进行扩展（例如，EditText 就是 TextView 的扩展组件）才可以正常输入文本。

在登录界面已经使用了 EditText 组件，读者也已经了解了 EditText 组件的基本使用方法，现在让我们再回顾一下 EditText 组件在 XML 布局文本中的使用方法，代码如下：

```
<EditText android:layout_width="wrap_content"
          android:layout_height="wrap_content" android:text="输入文本的组件"
          android:textColor="#000000" android:background="#FFFFFF"
          android:padding="20dp" android:layout_margin="10dp" />
```

从上面的代码可以看出，EditText 和 TextView 组件的使用方法完全一样，只需要将 <TextView> 标签换成 <EditText> 标签即可，几乎不需要做任何修改。



人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS



15.2.4 ImageView 组件



ImageView 组件可用于显示 Android 系统支持的图像（例如，GIF、JPG、PNG、BMP 等）。

在 XML 布局文件中使用<ImageView>标签来定义一个 ImageView 组件，代码如下：



```
<ImageView android:id="@+id/imageview" android:layout_width="wrap_content"  
          android:background="#F00" android:layout_height="wrap_content"  
          android:src="@drawable/icon" android:scaleType="center" />
```



在上面的代码中通过 android:src 属性指定了一个 drawable 资源的 ID，并使用 android:scaleType 属性指定 ImageView 组件显示图像的方式。例如，center 表示将图像以不缩放的方式显示在 ImageView 组件的中心。如果将 android:scaleType 属性设为 fitCenter，表示将图像按比例缩放至合适的位置，并显示在 ImageView 组件的中心。通常在设计相框时将 android:scaleType 属性设为 fitCenter，这样可以使照片按比例显示在相框的中心。



```
<ImageView android:layout_width="200dp" android:layout_height="100dp"  
          android:background="#F00" android:src="@drawable/background"  
          android:scaleType="fitCenter" android:padding="10dp" />
```



上面的代码直接设置了 ImageView 组件的宽度和高度，也可以在代码中设置和获取 ImageView 组件的宽度和高度。



```
ImageView imageView = (ImageView) findViewById(R.id.imageview);  
// 设置 ImageView 组件的宽度和高度  
imageView.setLayoutParams(new LinearLayout.LayoutParams(200, 100));  
// 获取 ImageView 组件的宽度和高度，并将获取的值显示在 Activity 的标题栏上  
setTitle("height:" + imageView.getLayoutParams().width + " height:" +  
        imageView.getLayoutParams().height);
```

执行上面的代码后，将显示如图 15.3 所示的效果。

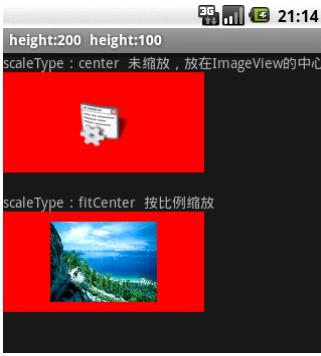


图 15.3 ImageView 组件的显示效果

15.2.5 Spinner 组件

Spinner 组件用于显示一个下拉列表。在装载数据时也需要创建一个 Adapter 对象，并在创建 Adapter 对象的过程中指定要装载的数据（数组或 List 对象）。例如，下面的代码分别使



用 ArrayAdapter 和 SimpleAdapter 对象向两个 Spinner 组件添加数据。

```
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    // 处理第 1 个 Spinner 组件
    Spinner spinner1 = (Spinner) findViewById(R.id.spinner1);
    String[] applicationNames = new String[]
    { "多功能日历", "乐博 Android 客户端", "白社会", "程序终结者" };
    ArrayAdapter<String> aaAdapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_spinner_item, applicationNames);
    // 将 ArrayAdapter 对象与第 1 个 Spinner 组件绑定
    spinner1.setAdapter(aaAdapter);
    // 处理第 2 个 Spinner 组件
    Spinner spinner2 = (Spinner) findViewById(R.id.spinner2);
    final List<Map<String, Object>> items = new ArrayList<Map<String, Object>>();
    Map<String, Object> item1 = new HashMap<String, Object>();
    item1.put("ivLogo", R.drawable.calendar);
    item1.put("tvApplicationName", "多功能日历");
    Map<String, Object> item2 = new HashMap<String, Object>();
    item2.put("ivLogo", R.drawable.eoemarket);
    item2.put("tvApplicationName", "eoemarket 客户端");
    items.add(item1);
    items.add(item2);
    SimpleAdapter simpleAdapter = new SimpleAdapter(this, items,
        R.layout.item, new String[]
        { "ivLogo", "tvApplicationName" }, new int[]
        { R.id.ivLogo, R.id.tvApplicationName });
    // 将 SimpleAdapter 对象与第 2 个 Spinner 组件绑定
    spinner2.setAdapter(simpleAdapter);
    // 为第 2 个 Spinner 组件设置 ItemSelected 事件
    spinner2.setOnItemSelectedListener(new OnItemSelectedListener()
    {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view,
            int position, long id)
        {
            // 当选中某一个列表项时，弹出一个对话框，并显示相应的 Logo 图像和应用程序名
            new AlertDialog.Builder(view.getContext()).setTitle(
                items.get(position).get("tvApplicationName").toString()).setIcon(
                Integer.parseInt(items.get(position).get("ivLogo").toString())).show();
        }
        @Override
        public void onNothingSelected(AdapterView<?> parent)
        {
        }
    });
}
```



人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS



运行本节的例子后，单击第 1 个和第 2 个 Spinner 组件右侧的下拉按钮，将显示如图 15.4 和图 15.5 所示的效果。





图 15.4 只显示文本的下拉列表框



图 15.5 带文本和图像的下拉列表框



15.2.6 使用 Gallery 组件循环显示图像

Gallery 组件一般用于显示图像列表，因此，也可称为相册组件。单击、选中或拖动 Gallery 中的图像，Gallery 中的图像列表会根据用户的操作向左或向右移动，直到显示到最后一个图像为止。

Gallery 本身并不支持循环显示图像，也就是说，当显示到最后一个图像时，图像列表就不再向左移动了。而我们要达到的循环显示的效果是当显示到最后一个图像时，下一个图像是图像列表中的第 1 个图像。实现这个效果也并不困难，只需要“欺骗”一下 ImageView 对象和 Adapter 对象即可。

从前面章节的内容可以知道，BaseAdapter 类中的 getView 方法的调用与 getCount 方法返回的值有关。如果 getCount 方法返回 n，那么 getView 方法中的 position 参数值是绝不会大于 n-1 的。因此，我们可以使 getCount 方法返回一个很大的数，例如，Integer.MAX_VALUE。这样系统就会认为 ImageAdapter 对象中有非常多的 Integer.MAX_VALUE 的值超过 20 亿，可以认为是接近无穷大）的 View 对象。

这样做还会带来另外一个问题。如果 getCount 方法返回了一个很大的数，那么 position 参数的值也会很大，在这种情况下，我们如何根据这个 position 参数值获得相应的图像 ID 资源呢？不会有人去创建 Integer.MAX_VALUE 大小的数组吧？当然，解决的方法也很简单。假设有一个 resIds 数组（长度为 15）保存了 15 个图像资源 ID。现在要使 Gallery 循环显示这 15 个图像。如果 position 的值超过了 14。可以使用取余的方法来循环取这个数组的值，代码如下：

```
| int imageResId = resIds[position % resIds.length];
```

下面还有一件重要事情要做，就是设置 Gallery 中每个图像的显示风格。首先需要获得图像背景的资源 ID。在 ImageAdapter 类的构造方法中编写如下的代码：

```
| TypedArray typedArray = obtainStyledAttributes(R.styleable.Gallery);  
| mGalleryItemBackground = typedArray.getResourceId(  
|     R.styleable.Gallery_android_galleryItemBackgrounds, 0);
```



人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS



其中 R.styleable.Gallery 是 res\values\attrs.xml 文件中一个属性的资源 ID，代码如下：

```
<declare-styleable name="Gallery">
    <attr name="android:galleryItemBackground" />
</declare-styleable>
```



在 getView 方法中需要设置 ImageView 组件的显示风格和图像资源，代码如下：



```
public View getView(int position, View convertView, ViewGroup parent)
{
    ImageView imageView = new ImageView(mContext);
    // 通过取余的方式获得图像的资源 ID
    imageView.setImageResource(resIds[position % resIds.length]);
    imageView.setScaleType(ImageView.ScaleType.FIT_XY);
    imageView.setLayoutParams(new Gallery.LayoutParams(136, 88));
    imageView.setBackgroundDrawable(mGalleryItemBackground);
    return imageView;
}
```



运行本节的例子后，会看到如图 15.6 所示的效果。



图 15.6 循环显示图像的 Gallery

15.2.7 TabHost 组件

如果屏幕上需要放置很多组件，可能一屏放不下，除了使用滚动视图的方式外，还可以使用标签组件对屏幕进行分页。也许很多读者在其他的编程语言中见过标签组件，当单击标签组件的不同标签时，会显示当前标签的内容。在 Android 系统中每一个标签可以显示一个 View 或一个 Activity。

TabHost 是标签组件的核心类，也是标签的集合。每一个标签是一个 TabHost.TabSpec 类的对象实例。通过 TabHost 类的 addTab 方法可以添加多个 TabHost.TabSpec 类的对象实例（多个标签）。如果从 XML 布局文件中添加 View，首先需要建立一个布局文件，并且根结点要使用<FrameLayout>或<TabHost>标签。

在本例中建立了 3 个标签。在第 1 个标签中显示了一个 View，在 View 中有两个组件：Button 和 ImageView。另两个标签分别显示其他的 Activity。XML 布局文件的内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent">
    <!-- 定义在第 1 个标签中显示的视图 -->
    <LinearLayout android:id="@+id/tab1" android:orientation="vertical"
        android:layout_width="fill_parent" android:layout_height="fill_parent">
        <Button android:id="@+id/button" android:layout_width="fill_parent"
```



```

        android:layout_height="wrap_content" android:text="切换到第 3 个标签" />
    <ImageView android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/background" android:layout_marginTop="30dp"/>
    </LinearLayout>
</TabHost>

```

在创建 TabHost 对象时一般使用从 TabActivity 继承的类，在该类的 onCreate 方法中添加 3 个标签，代码如下：

```

// 通过 TabActivity 类的 getTabHost 方法获取 TabHost 对象
TabHost tabHost = getTabHost();
// 装载 main.xml 布局文件，也就是上面给出的 XML 布局文件
LayoutInflater.from(this).inflate(R.layout.main,tabHost.getTabContentView(),true);
// 添加第 1 个标签，显示视图（按钮和 ImageView）
tabHost.addTab(tabHost.newTabSpec("tab1").setIndicator("切换标签").setContent(
(R.id.tab1)));
// 添加第 2 个标签，在标签页上显示一个图像，并在该页中显示 GalleryActivity
tabHost.addTab(tabHost.newTabSpec("tab2").setIndicator("相册",
        getResources().getDrawable(R.drawable.icon1))
        .setContent(new Intent(this, GalleryActivity.class)));
// 添加第 3 个标签，在该标签中显示 RatingListView
tabHost.addTab(tabHost.newTabSpec("tab3").setIndicator("评分")
        .setContent(new Intent(this, RatingListView.class)));

```

在上面的代码中通过 TabHost 类的 newTabSpec 方法创建了 TabSpec 对象。newTabSpec 方法的参数表示标签的字符串标识。也就是说，通过该标识可以获得相应的标签。在单击第 1 个标签中的按钮后，可以切换到第 3 个标签。要完成这个功能可以使用标签的索引，也可以使用通过 newTabSpec 方法设置的标识。切换到第 3 个标签的代码如下：

```

// 标签索引从 0 开始
getTabHost().setCurrentTab(2);
// 或采用如下的代码
// getTabHost().setCurrentTabByTag("tab3");

```

运行本节的例子后，第 2 个标签和第 3 个标签的效果如图 15.7 和图 15.8 所示。



图 15.7 第 2 个标签页的显示效果



图 15.8 第 3 个标签页的显示效果



人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS



15.3 访问网络



虽然在登录系统中使用了 Web Service 与服务端进行交互。但是在传递大量的数据时，Web Service 显得有些笨拙。在本节将介绍移动电子相册中使用的另外一种与数据库交互的方法。直接发送 HTTP GET 或 POST 请求。这就要用到 `HttpGet`、`HttpPost` 以及 `HttpURLConnection` 这些类。



15.3.1 `HttpGet` 类和 `HttpPost` 类



本节将介绍 Android SDK 集成的 Apache HttpClient 模块。要注意的是，这里的 Apache HttpClient 模块是 HttpClient 4.0 (`org.apache.http.*`)，而不是 Jakarta Commons HttpClient 3.x (`org.apache.commons.httpclient.*`)。



在 HttpClient 模块中用到了两个重要的类：`HttpGet` 和 `HttpPost`。这两个类分别用来提交 HTTP GET 和 HTTP POST 请求。为了测试本节的例子，需要先编写一个 Servlet 程序，用来接收 HTTP GET 和 HTTP POST 请求。读者也可以使用其他服务端的资源来测试本节的例子。



假设 192.168.17.81 是本机的 IP，客户端可以通过如下的 URL 来访问服务端的资源：

`http://192.168.17.81:8080/querybooks/QueryServlet?bookname=开发`



在这里 `bookname` 是 `QueryServlet` 的请求参数，表示图书名，通过该参数来查询图书信息。

现在我们要通过 `HttpGet` 和 `HttpPost` 类向 `QueryServlet` 提交请求信息，并将返回结果显示在 `TextView` 组件中。无论是使用 `HttpGet`，还是使用 `HttpPost`，都必须通过如下 3 步来访问 HTTP 资源。

- ① 创建 `HttpGet` 或 `HttpPost` 对象，将要请求的 URL 通过构造方法传入 `HttpGet` 或 `HttpPost` 对象。
- ② 使用 `DefaultHttpClient` 类的 `execute` 方法发送 HTTP GET 或 HTTP POST 请求，并返回 `HttpResponse` 对象。
- ③ 通过 `HttpResponse` 接口的 `getEntity` 方法返回响应信息，并进行相应的处理。

如果使用 `HttpPost` 方法提交 HTTP POST 请求，还需要使用 `HttpPost` 类的 `setEntity` 方法设置请求参数。

本例使用了两个按钮来分别提交 HTTP GET 和 HTTP POST 请求，并从 `EditText` 组件中获得请求参数（`bookname`）值，最后将返回结果显示在 `TextView` 组件中。两个按钮共用一个 `onClick` 事件方法，代码如下：

```
public void onClick(View view)
{
    // 读者需要将本例中的 IP 换成自己机器的 IP
    String url = "http://192.168.17.81:8080/querybooks/QueryServlet";
    TextView tvQueryResult = (TextView) findViewById(R.id.tvQueryResult);
    EditText etBookName = (EditText) findViewById(R.id.etBookName);
    HttpResponse httpResponse = null;
```





```
try
{
    switch (view.getId())
    {
        // 提交 HTTP GET 请求
        case R.id.btnGetQuery:
            // 向 url 添加请求参数
            url += "?bookname=" + etBookName.getText().toString();
            // 第 1 步：创建 HttpGet 对象
            HttpGet httpGet = new HttpGet(url);
            // 第 2 步：使用 execute 方法发送 HTTP GET 请求，并返回 HttpResponse 对象
            httpResponse = new DefaultHttpClient().execute(httpGet);
            // 判断请求响应状态码，状态码为 200 表示服务端成功响应了客户端的请求
            if (httpResponse.getStatusLine().getStatusCode() == 200)
            {
                // 第 3 步：使用 getEntity 方法获得返回结果
                String result = EntityUtils.toString(httpResponse.getEntity());
                // 去掉返回结果中的 “\r” 字符，否则会在结果字符串后面显示一个小方格
                tvQueryResult.setText(result.replaceAll("\r", ""));
            }
            break;
        // 提交 HTTP POST 请求
        case R.id.btnPostQuery:
            // 第 1 步：创建HttpPost 对象
            HttpPost httpPost = new HttpPost(url);
            // 设置 HTTP POST 请求参数必须用 NameValuePair 对象
            List<NameValuePair> params = new ArrayList<NameValuePair>();
            params.add(new BasicNameValuePair("bookname", etBookName.getText().toString()));
            // 设置 HTTP POST 请求参数
            httpPost.setEntity(new UrlEncodedFormEntity(params, HTTP.UTF_8));
            // 第 2 步：使用 execute 方法发送 HTTP POST 请求，并返回 HttpResponse 对象
            httpResponse = new DefaultHttpClient().execute(httpPost);
            if (httpResponse.getStatusLine().getStatusCode() == 200)
            {
                // 第 3 步：使用 getEntity 方法获得返回结果
                String result = EntityUtils.toString(httpResponse.getEntity());
                // 去掉返回结果中的 “\r” 字符，否则会在结果字符串后面显示一个小方格
                tvQueryResult.setText(result.replaceAll("\r", ""));
            }
            break;
    }
}
catch (Exception e)
{
    tvQueryResult.setText(e.getMessage());
}
```



15.3.2 HttpURLConnection 类

在移动电子相册上传图像时使用了 HttpURLConnection 类。java.net.HttpURLConnection



人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS

类是另外一种访问 HTTP 资源的方式。HttpURLConnection 类具有完全的访问能力，可以取代HttpGet 和HttpPost 类。使用 HttpURLConnection 访问 HTTP 资源可以使用如下几步。

① 使用 java.net.URL 封装 HTTP 资源的 URL，并使用 openConnection 方法获得 HttpURLConnection 对象，代码如下：

```
URL url = new  
URL("http://www.blogjava.net/nokiaguy/archive/2009/12/14/305890.html");  
HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
```

② 设置请求方法，例如，GET、POST 等，代码如下：

```
httpURLConnection.setRequestMethod("POST");
```

要注意的是，setRequestMethod 方法的参数值必须大写，例如，GET、POST 等。

③ 设置输入、输出及其他权限。如果要下载 HTTP 资源或向服务端上传数据，需要使用如下的代码进行设置：

```
// 下载 HTTP 资源，需要将 setDoInput 方法的参数值设为 true  
httpURLConnection.setDoInput(true);  
// 上传数据，需要将 setDoOutput 方法的参数值设为 true  
httpURLConnection.setDoOutput(true);
```

HttpURLConnection 类还包含了更多的选项，例如，使用下面的代码可以禁止 HttpURLConnection 使用缓存。

```
httpURLConnection.setUseCaches(false);
```

④ 设置 HTTP 请求头。在很多情况下，要根据实际情况设置一些 HTTP 请求头，例如，下面的代码设置了 Charset 请求头的值为 UTF-8。

```
httpURLConnection.setRequestProperty("Charset", "UTF-8");
```

⑤ 输入和输出数据。这一步是对 HTTP 资源的读写操作。也就是通过 InputStream 和 OutputStream 读取和写入数据。下面的代码获得了 InputStream 对象和 OutputStream 对象。

```
InputStream is = httpURLConnection.getInputStream();  
OutputStream os = httpURLConnection.getOutputStream();
```

至于是先读取还是先写入数据，需要根据具体情况而定。

⑥ 关闭输入/输出流。虽然关闭输入/输出流并不是必须的，在应用程序结束后，输入/输出流会自动关闭。但显式关闭输入/输出流是一个好习惯。关闭输入/输出流的代码如下：

```
is.close();  
os.close();
```

15.4 项目实战：浏览照片

浏览照片的基本功能包括显示当前用户的所有相册，当用户选择某个相册后，会显示当前相册中所有照片的缩略图。如果相册中的照片很多，会分页显示。当长按某个照片的缩略图时



会放大显示该照片的图像。

15.4.1 编写 CommonServlet 类

从本节开始，所有的服务端程序都使用 Servlet 实现。开心网的所有服务端功能除了登录之外，在访问之前都需要进行身份验证，也就是验证用户名和密码。除此之外，所有的 Servlet 都需要连接数据库，因此，这就需要编写一个通用的 CommonServlet 类，该类是 HttpServlet 的子类，负责打开数据库，并进行身份验证。CommonServlet 类的代码如下：

```
package service.servlet;

import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Random;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.sun.image.codec.jpeg.JPEGCodec;
import com.sun.image.codec.jpeg.JPEGImageEncoder;

public abstract class CommonServlet extends HttpServlet
{
    protected Connection mConnection;
    protected String mUsername;
    protected String mPassword;
    protected String mPhotoRootPath;
    protected abstract void execute(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException;

    protected void service(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException
    {
        try
        {
            mPhotoRootPath =
                getServletContext().getInitParameter("photoRootPath");
            Class.forName("com.mysql.jdbc.Driver");
            // 打开数据库，并获取 Connection 对象
        }
    }
}
```



人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS

```
mConnection = DriverManager.getConnection(  
        "jdbc:mysql://localhost/kxw?characterEncoding=UTF-8",  
        "root", "1234");  
// 定义查询 t_kx_users 表的 SQL 语句  
String sql = "select * from t_kx_users where email=? and password=?";  
PreparedStatement pstmt = mConnection.prepareStatement(sql);  
mUsername = request.getParameter("username");  
mPassword = request.getParameter("password");  
pstmt.setString(1, mUsername);  
pstmt.setString(2, mPassword);  
// 进行用户验证  
ResultSet rs = pstmt.executeQuery();  
if (rs.next())  
{  
    // 如果验证通过，执行 execute 方法。  
    execute(request, response);  
}  
else  
{  
    response.getWriter().println("Unauthorized Access.");  
}  
}  
catch (Exception e)  
{  
    response.getWriter().println(e.getMessage());  
}  
}  
}  
}
```

CommonServlet 类覆盖了 service 方法，并在该方法中打开数据库，并进行用户验证。如果验证通过，则调用抽象方法 execute。因此，在 CommonServlet 的子类中实现 execute 方法即可。如果 execute 方法被调用，则说明已通过了用户验证。

由于手机内存要比 PC 的内存小得多，因此，在手机客户端浏览图像时不能显示太大的图像，否则会抛出内容溢出的错误。因此，服务端在发送给客户端图像时应将当前图像按一定比例缩小，为此，可以在 CommonServlet 类中再加一个用于缩小图像的方法，代码如下：

```
public void scaleImage(InputStream imgInputStream,  
                      OutputStream imgOutputStream, int scale)  
{  
    try  
    {  
        Image src = javax.imageio.ImageIO.read(imgInputStream);  
        int width = (int) (src.getWidth(null) * scale / 100.0);  
        int height = (int) (src.getHeight(null) * scale / 100.0);  
        BufferedImage bufferedImage = new BufferedImage(width, height,  
                                                       BufferedImage.TYPE_INT_RGB);  
        bufferedImage.getGraphics().drawImage(  
            src.getScaledInstance(width, height, Image.SCALE_SMOOTH),  
            0, 0, null);  
        JPEGImageEncoder encoder = JPEGCodec
```



```
    .createJPEGEncoder(imgOutputStream);
    encoder.encode(bufferedImage);

}
catch (IOException e)
{
}
}
```

其中 scaleImage 方法的 imgInputStream 参数表示源图像的输入流； imgOutputStream 参数表示图像缩小后的目标图像的输入流； scale 参数表示缩放因子，在 1 至 100 之间，如果为 100，表示不缩放。

服务端和客户端的交互采用了直接传输二进制数据的形式，也就是服务端和客户端传递可序列化的对象。因此，在 CommonServlet 类中加一个 sendObject 方法，以便将服务端的对象发送到客户端。

```
protected void sendObject(Object obj, HttpServletResponse response)
{
    try
    {
        OutputStream os = response.getOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(os);
        oos.writeObject(obj);
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
```



15.4.2 获取当前用户的所有相册

用 AlbumServlet 类可以获取指定用户的所有相册名称、ID 等信息。这些信息通过 List<Album> 对象进行封装，Album 类封装了一个相册的信息，代码如下：

```
package service;
import java.io.Serializable;
public class Album implements Serializable
{
    private int id;
    private String name;
    private String email;
    private String description;
    // 此处省略了属性的 getter 和 setter 方法
}
```

AlbumServlet 类的代码如下：

```
package service.servlet;
import java.io.IOException;
```



人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import service.Album;
public class AlbumServlet extends CommonServlet
{
    @Override
    protected void execute(HttpServletRequest request,
                           HttpServletResponse response) throws ServletException, IOException
    {
        try
        {
            String sql = "select * from t_kx_albums where email=?";
            PreparedStatement preparedStatement = mConnection
                .prepareStatement(sql);
            preparedStatement.setString(1, mUsername);
            List<Album> albums = new ArrayList<Album>();
            ResultSet rs = preparedStatement.executeQuery();
            // 开始为每一个相册创建单独的 Album 对象，并将该对象添加到 List 对象中
            while (rs.next())
            {
                Album album = new Album();
                album.setId(rs.getInt("id"));
                album.setName(rs.getString("album_name"));
                album.setEmail(rs.getString("email"));
                album.setDescription(rs.getString("description"));
                albums.add(album);
            }
            // 将 Album 对象发送到客户端
            sendObject(albums, response);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

AlbumServlet 类的配置代码如下：

```
<servlet>
    <servlet-name>AlbumServlet</servlet-name>
    <servlet-class>service.servlet.AlbumServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>AlbumServlet</servlet-name>
    <url-pattern>/AlbumServlet</url-pattern>
</servlet-mapping>
```



15.4.3 设置移动电子相册的主界面

移动电子相册的主界面由一个 TabHost 组件构成。这个 TabHost 组件包含了【浏览照片】、【建立相册】和【上传图像】三个页面。一般带 TabHost 的 Activity 类需要从 TabActivity 类继承。在 onCreate 方法中加载 TabHost 中的页面，代码如下：

```
TabHost tabHost = getTabHost();
mTabView1 = LayoutInflater.from(this).inflate(R.layout.album_tab1,
    tabHost.getTabContentView(), true);
// 添加第一个页面
tabHost.addTab(tabHost.newTabSpec("tab1").setIndicator("浏览照片",
    getResources().getDrawable(R.drawable.browseimage))
    .setContent(R.id.albumTab1));
mTabView2 = LayoutInflater.from(this).inflate(R.layout.album_tab2,
    tabHost.getTabContentView(), true);
// 添加第二个页面
tabHost.addTab(tabHost.newTabSpec("tab2").setIndicator("建立相册",
    getResources().getDrawable(R.drawable.createalbum)).setContent(R.id.album
Tab2));
mTabView3 = LayoutInflater.from(this).inflate(R.layout.album_tab3,
    tabHost.getTabContentView(), true);
// 添加第三个页面
tabHost.addTab(tabHost.newTabSpec("tab3").setIndicator("上传图像",
    getResources().getDrawable(R.drawable.uploadimage)).setContent(R.id.album
Tab3));
```

15.4.4 在客户端显示相册列表

当单击如图 15.1 所示的 Spinner 组件，会显示如图 15.9 所示的相册列表。



图 15.9 显示当前用户的相册列表

Spinner 类需要一个 Adapter 类来提供对象，代码如下：

```
class AlbumAdapter extends BaseAdapter
{
    private LayoutInflater mLayoutInflater;
```



人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS



```
public AlbumAdapter(Context context)
{
    mLayoutInflater = (LayoutInflater) context
        .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
}

@Override
public int getCount()
{
    return mAlbums.size();
}

@Override
public Object getItem(int position)
{
    return mAlbums.get(position);
}

@Override
public long getItemId(int position)
{
    return 0;
}

@Override
public View getView(int position, View convertView, ViewGroup parent)
{
    View view = mLayoutInflater.inflate(R.layout.album_item, null);
    TextView tvAlbumItem = (TextView) view
        .findViewById(R.id.tvAlbumItem);
    tvAlbumItem.setText(mAlbums.get(position).getName());
    return view;
}
}
```

其中 mAlbums 是一个 List<Album>类型的变量，用于保存从服务端获得的当前用户相册列表。所以访问服务端资源的代码都在 CallService 类中。CallService 类有一个通用的 getObject 方法，该方法向服务端发送请求，并返回一个 Object 对象。getObject 方法的代码如下：

```
private static Object getObject(String url, String[] paramNames,
                               String[] paramValues, boolean original)
{
    try
    {
        HttpPost httpPost = new HttpPost(url);aa
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        params.add(new BasicNameValuePair("username", username));
        params.add(new BasicNameValuePair("password", passwordMD5));
        // 不管是什么样的请求，都需要用户名和密码两个参数，因此，在这里首先要发送这两个参数
        for (int i = 0; i < paramNames.length; i++)
        {
            params.add(new BasicNameValuePair(paramNames[i], paramValues[i]));
        }
        httpPost.setEntity(new UrlEncodedFormEntity(params, HTTP.UTF_8));
    }
}
```



```
// 向服务端发送请求
HttpResponse httpResponse = new DefaultHttpClient()
    .execute(httpPost);
// 判断请求响应状态码，状态码为 200 表示服务端成功响应了客户端的请求
if (httpResponse.getStatusLine().getStatusCode() == 200)
{
    InputStream is = httpResponse.getEntity().getContent();
    if (original)
    {
        // 直接返回服务端的 InputStream 对象
        return is;
    }
    else
    {
        // 将服务端返回的数据转换成对象返回
        ObjectInputStream ois = new ObjectInputStream(is);
        Object obj = ois.readObject();
        return obj;
    }
}
catch (Exception e)
{
}
return null;
}
```

其中 url 参数表示访问服务端的 URL; paramNames 参数表示请求参数名; paramValues 参数表示请求参数值; original 参数表示返回的对象类型, 如果该参数值为 true, 直接返回服务端的 InputStream 对象, 以便更灵活地获得服务端的数据, 如果该参数值为 false, 表示将服务端返回的数据转换成 Object 对象后再返回。

如果某个请求除了用户名和密码外, 不向服务端发送其他的请求参数, 在这种情况下就不需要 paramNames 和 paramValues 参数了, 因此, 我们来重载一次 getObject 方法, 代码如下:

```
private static Object getObject(String url, boolean original)
{
    return getObject(url, new String[]
    {}, new String[]
    {}, original);
}
```

下面来看一下获得当前用户相册列表的 getAlbums 方法, 代码如下:

```
public static List<Album> getAlbums()
{
    String url = "http://192.168.17.81:8080/kxw_service/AlbumServlet";
    Object obj = getObject(url, false);
    return (List<Album>) obj;
}
```



人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS



现在就可以使用如下的代码获得当前用户的相册列表了。

```
| mAlbums = CallService.getAlbums();
```

15.4.5 获取指定相册的照片路径

在显示指定相册中的照片之前，需要获取这些照片的在服务端的路径（在这里是随机生成的文件名）。PhotoUrlServlet 类通过 List<String> 对象返回指定相册的路径，代码如下：

```
package service.servlet;
import java.io.IOException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class PhotoUrlServlet extends CommonServlet
{
    @Override
    protected void execute(HttpServletRequest request,
                           HttpServletResponse response) throws ServletException, IOException
    {
        try
        {
            // 获取显示的页码，从 1 开始，每页显示 10 个图像缩略图
            int page = Integer.parseInt(request.getParameter("page"));
            // 获取指定的相册 ID
            int albumId = Integer.parseInt(request.getParameter("albumId"));
            // 计算开始的记录数
            int start = (page - 1) * 10;
            String sql = "select * from t_kx_photos where album_id = ? limit "
                         + start + ",10";
            PreparedStatement preparedStatement = mConnection
                .prepareStatement(sql);
            preparedStatement.setInt(1, albumId);
            List<String> filenames = new ArrayList<String>();
            // 开始查询当前页的图像信息
            ResultSet rs = preparedStatement.executeQuery();
            while (rs.next())
            {
                filenames.add(rs.getString("photo_filename"));
            }
            // 将查询到的图像信息发送到客户端
            sendObject(filenames, response);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

```
    }
}
```

PhotoUrlServlet 通过页码和相册 ID 获得相应的图像路径列表。页码从 1 开始，每次返回 10 个图像路径，如果不是 10 个图像路径，则返回全部的图像路径列表。

PhotoUrlServlet 类的配置代码如下：

```
<servlet>
    <servlet-name>PhotoUrlServlet</servlet-name>
    <servlet-class>service.servlet.PhotoUrlServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>PhotoUrlServlet</servlet-name>
    <url-pattern>/PhotoUrlServlet</url-pattern>
</servlet-mapping>
```

15.4.6 获取指定的照片数据

每一个照片图像都需要从 Servlet 中获取。为了在手机中显示图像，需要将图像按一定比例缩小，否则大多数手机的内存无法显示尺寸很大的图像。在本例中将图像按 1/5 缩小，读者也可以根据图像的实际大小缩放图像。获取照片图像数据由 PhotoServlet 类完成，代码如下：

```
package service.servlet;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.OutputStream;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class PhotoServlet extends CommonServlet
{
    @Override
    protected void execute(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException
    {
        try
        {
            // 获取指定图像的文件名
            String filename = request.getParameter("filename");
            // 获取指定的相册 ID
            int albumId = Integer.parseInt(request.getParameter("albumId"));
            // 如果 orginal 为 true，表示图像不缩小，如果为 false，表示按原图的 1/5 缩图
            boolean original = Boolean.parseBoolean(request.getParameter("original"));
            // 生成图像的本地路径
            String photoPath = mPhotoRootPath + mUsername.hashCode()
                + File.separator + albumId + File.separator + filename;
            OutputStream os = response.getOutputStream();
            FileInputStream fis = new FileInputStream(photoPath);
```





人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS



```
if (original)
{
    // 按源图像实际大小返回图像数据
    byte[] buffer = new byte[8192];
    int n = 0;
    while ((n = fis.read(buffer)) > 0)
    {
        os.write(buffer, 0, n);
    }

}
else
{
    // 按 1/5 比例缩小图像
    scaleImage(fis, os, 20);
}
fis.close();

}
catch (Exception e)
{
    e.printStackTrace();
}
}
```

PhotoServlet 会根据指定的图像文件名和相册的 ID 返回缩小或原始的图像数据(由 original 请求参数值确定)。

PhotoServlet 类的配置代码如下：

```
<servlet>
    <servlet-name>PhotoServlet</servlet-name>
    <servlet-class>service.servlet.PhotoServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>PhotoServlet</servlet-name>
    <url-pattern>/PhotoServlet</url-pattern>
</servlet-mapping>
```

15.4.7 在客户端分页显示照片缩略图

要从服务端获取当前页的图像路径，需要在 CallService 类中加一个 getAlbums 方法，代码如下：

```
public static List<String> getPhotoUrl(int page, int albumId)
{
    String url = "http://192.168.17.81:8080/kxw_service/PhotoUrlServlet";
    // 向服务端发送 page 和 albumId 请求参数
    Object obj = getObject(url, new String[]
    { "page", "albumId" }, new String[]
```



```
|     { String.valueOf(page), String.valueOf(albumId) }, false);
|     return (List<String>) obj;
| }
```

当选择某个相册时，会创建一个 Adapter 对象，并设置 Gallery 组件的值，代码如下：

```
public void onItemSelected(AdapterView<?> adapterView, View view,
    int position, long id)
{
    mCurrentAlbumPosition = position;
    mCurrentPage = 1;
    mPhotoUrls = CallService.getPhotoUrl(mCurrentPage, mAlbums
        .get(position).getId());
    mGallery.setAdapter(new GalleryAdapter(this));
}
```

其中 GalleryAdapter 类封装了相册缩略图数据，代码如下：

```
class GalleryAdapter extends BaseAdapter
{
    private LayoutInflater mLayoutInflater;
    private int mResourceId;
    // 保存显示图像的 View 对象
    private Map<String, View> mViewMap = new HashMap<String, View>();
    public GalleryAdapter(Context context)
    {
        mLayoutInflater = (LayoutInflater) context
            .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        TypedArray typedArray = obtainStyledAttributes(R.styleable.Gallery);
        mResourceId = typedArray.getResourceId(R.styleable.Gallery_android_
        galleryItemBackground, 0);
    }
    @Override
    public int getCount()
    {
        return mPhotoUrls.size();
    }
    @Override
    public Object getItem(int position)
    {
        return null;
    }
    @Override
    public long getItemId(int position)
    {
        return 0;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent)
    {
        View view = null;
        try
```



人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS



```
{  
    view = mViewMap.get(mPhotoUrls.get(position));  
    if (view == null)  
    {  
        view = mLayoutInflater.inflate(R.layout.gallery_item, null);  
        ImageView ivPhoto = (ImageView) view . findViewById(R.id.ivPhoto);  
        // 从服务端获取图像的缩略图  
        Bitmap bitmap = CallService.getPhoto(mAlbums.get(  
            mCurrentAlbumPosition).getId(), mPhotoUrls .get(position), false);  
        // 设置图像的缩略图  
        ivPhoto.setImageBitmap(bitmap);  
        ivPhoto.setBackgroundResource(mResourceId);  
        // 将显示图像的 View 对象保存在 Map 对象中  
        mViewMap.put(mPhotoUrls.get(position), view);  
    }  
}  
catch (Exception e)  
{  
}  
}  
return view;  
}  
}
```

其中 `getPhoto` 方法用于获取图像数据，代码如下：

```
public static Bitmap getPhoto(int albumId, String filename, boolean original)  
{  
  
    String url = "http://192.168.17.81:8080/kxw_service/PhotoServlet";  
    // 直接获得 InputStream 对象，并从 InputStream 对象中获取图像装饰，最后返回表示图像的  
    // Bitmap 对象  
    InputStream is = (InputStream) getObject(url, new String[]  
    { "albumId", "filename", "original" }, new String[]  
    { String.valueOf(albumId), filename, String.valueOf(original) }, true);  
    Bitmap bitmap = BitmapFactory.decodeStream(is);  
    return bitmap;  
}
```

选择一个相册后，缩略图的显示效果如图 15.1 所示。

15.4.8 显示照片大图

当长按图像的缩略图后，会显示该图像的大图，代码如下：

```
public boolean onItemLongClick(AdapterView<?> adapterView, View view,  
        int position, long id)  
{  
    Intent intent = new Intent(this, BigPicture.class);  
    intent.putExtra("albumId", mAlbums.get(mCurrentAlbumPosition).getId());  
    intent.putExtra("filename", mPhotoUrls.get(position));  
}
```



```
    startActivity(intent);
    return false;
}
```

其中 BigPicture 是显示大图的 Activity，需要两个参数：albumId 和 filename。在 BigPicture 类中需要通过如下的代码来显示图像的大图：

```
ImageView ivBigPicture = (ImageView) findViewById(R.id.ivBigPicture);
String filename = getIntent().getExtras().getString("filename");
int albumId = getIntent().getExtras().getInt("albumId");
Bitmap bitmap = CallService.getPhoto(albumId, filename, true);
ivBigPicture.setImageBitmap(bitmap);
```

显示大图的效果如图 15.10 所示。



图 15.10 显示大图

15.5 项目实战：建立相册

移动电子相册还需要一个建立相册的功能。用户在手机客户端只需输入相册名称和相册描述就可以创建一个相册，其他信息（如用户名、E-mail 等）在登录时已经确定。本节将详细介绍如何实现建立相册的功能。

15.5.1 编写建立相册的服务

相册需要 E-mail（也就是用户账号）、相册名称和相册描述这三个信息。通过 CreateAlbum Servlet 类可以从客户端获得相应的请求参数值，然后将这些值保存到数据库中，代码如下：

```
package service.servlet;
import java.io.IOException;
import java.sql.PreparedStatement;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
```



人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS

```
import javax.servlet.http.HttpServlet;
public class CreateAlbumServlet extends CommonServlet
{
    @Override
    protected void execute(HttpServletRequest request,
                           HttpServletResponse response) throws ServletException, IOException
    {
        try
        {
            String sql = "insert into t_kx_albums(email, album_name, description)
values(?, ?, ?)";
            PreparedStatement preparedStatement = mConnection
                .prepareStatement(sql);
            String albumName = request.getParameter("albumName");
            String description = request.getParameter("description");
            // 设置 SQL 的参数值
            preparedStatement.setString(1, mUsername);
            preparedStatement.setString(2, albumName);
            preparedStatement.setString(3, description);
            // 执行 insert 语句将数据插入到数据库中
            preparedStatement.execute();
            // 如果插入成功（成功建立相册），返回 ok
            response.getWriter().println("ok");
        }
        catch (Exception e)
        {
            // 插入失败，返回错误信息
            response.getWriter().println(e.getMessage());
        }
    }
}
```

CreateAlbumServlet 类的配置代码如下：

```
<servlet>
    <servlet-name>CreateAlbumServlet</servlet-name>
    <servlet-class>service.servlet.CreateAlbumServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>CreateAlbumServlet</servlet-name>
    <url-pattern>/CreateAlbumServlet</url-pattern>
</servlet-mapping>
```

15.5.2 在客户端提交相册信息

手机客户端建立相册的界面如图 15.11 所示。在输入完图 15.12 所示的内容后，单击【建立相册】按钮，会将相册信息保存到服务端。然后在【浏览照片】页面就可以查看相册信息了。



图 15.11 建立相册



图 15.12 输入相册信息

createAlbum 方法可以向服务端发送相册信息，代码如下：

```
public static String createAlbum(String albumName, String description)
{
    String url = "http://192.168.17.81:8080/kxw_service/CreateAlbumServlet";
    InputStream is = (InputStream) getObject(url, new String[]
    { "albumName", "description" }, new String[]
    { albumName, description }, true);
    InputStreamReader isr = new InputStreamReader(is);
    BufferedReader br = new BufferedReader(isr);
    String result = "";
    try
    {
        result = br.readLine();
    }
    catch (Exception e)
    {
    }
    return result;
}
```

单击【建立相册】按钮后，会执行下面的代码来建立相册。

```
String albumName = metAlbumName.getText().toString();
if ("".equals(albumName))
    return;
// 建立相册
result = CallService.createAlbum(albumName, metDescription.getText().toString());
if ("ok".equals(result))
    Message.showMsg(this, "成功建立相册.");
else
    Message.showMsg(this, result);
```

15.6 项目实战：上传图像

上传图像是移动电子相册中最吸引人的功能，因为 Android 手机都会至少带一个摄像头，



人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS

这样可以随时随地将拍摄的照片发到开心网上与好友分享。除了直接将拍摄的照片发到网上，还可以选择手机上已经存在的图像文件，并将其发到网上。

15.6.1 服务端接收客户端上传的图像

上传到服务端的图像需要如下的信息：

- username（也就是 E-mail）
- passwordMD5（用 MD5 加密的密码）
- userId（用户 ID）
- albumId（相册 ID）
- photoData（照片的数据，以 byte[] 形式保存）

客户端在上传这些数据时，将它们封装在了 UploadPhotoData 对象中。UploadPhotoData 类的代码如下：

```
package service;
import java.io.Serializable;
public class UploadPhotoData implements Serializable
{
    private String username;
    private String passwordMD5;
    private int userId;
    private int albumId;
    private byte[] photoData;
    // 此处省略了属性的 getter 和 setter 方法
    ...
}
```

UploadPhotoServlet 类用于接收客户端发送的图像数据。在该类中从 HTTP 请求获取 InputStream 对象，并将数据流转换成 UploadPhotoData 对象，代码如下：

```
package service.servlet;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import service.UploadPhotoData;
public class UploadPhotoServlet extends HttpServlet
{
    protected void service(HttpServletRequest request,
```



```
    HttpServletRequest response) throws ServletException, IOException
{
    try
    {
        String photoRootPath = getServletContext().getInitParameter(
            "photoRootPath");
        Class.forName("com.mysql.jdbc.Driver");
        // 获取 Connection 对象
        Connection connection = DriverManager.getConnection(
            "jdbc:mysql://localhost/kxw?characterEncoding=UTF-8",
            "root", "1234");
        InputStream is = request.getInputStream();
        ObjectInputStream ois = new ObjectInputStream(is);
        Object obj = ois.readObject();
        // 从客户端获取 UploadPhotoData 对象
        UploadPhotoData uploadPhotoData = (UploadPhotoData) obj;
        String sql = "select * from t_kx_users where email=? and password=?";

        PreparedStatement preparedStatement = connection
            .prepareStatement(sql);
        preparedStatement.setString(1, uploadPhotoData.getUsername());
        preparedStatement.setString(2, uploadPhotoData.getPasswordMD5());
        ResultSet rs = preparedStatement.executeQuery();
        // 进行用户身份验证
        if (rs.next())
        {
            sql = "insert into t_kx_photos(user_id,album_id,photo_filename,
                content_type) values(?, ?, ?, ?)";
            preparedStatement = connection.prepareStatement(sql);
            String filename = CommonServlet.getRandomFileName();

            String photoPath = photoRootPath
                + uploadPhotoData.getUsername().hashCode()
                + File.separator + uploadPhotoData.getAlbumId();

            File file = new File(photoPath);
            if (!file.exists())
            {
                // 建立保存上传照片的目录
                file.mkdirs();
            }
            photoPath += File.separator + filename;
            FileOutputStream fos = new FileOutputStream(photoPath);
            // 在服务端保存图像文件
            fos.write(uploadPhotoData.getPhotoData());
            fos.close();
            preparedStatement.setInt(1, uploadPhotoData.getUserId());
            preparedStatement.setInt(2, uploadPhotoData.getAlbumId());
            preparedStatement.setString(3, filename);
            preparedStatement.setString(4, "image/jpeg");
            // 向数据库中插入照片信息记录
        }
    }
}
```



人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS

```
        preparedStatement.execute();
        response.getWriter().println("ok");
    }
}
catch (Exception e)
{
    e.printStackTrace();
    response.getWriter().println(e.getMessage());
}
}
```

由于 UploadPhotoServlet 类需要直接使用 InputStream 对象，因此，该类需要直接从 HttpServlet 类继承（不能同时使用 InputStream 和 getParameter 方法从客户端获得数据）。

UploadPhotoServlet 类的配置代码如下：

```
<servlet>
    <servlet-name>UploadPhotoServlet</servlet-name>
    <servlet-class>service.servlet.UploadPhotoServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>UploadPhotoServlet</servlet-name>
    <url-pattern>/UploadPhotoServlet</url-pattern>
</servlet-mapping>
```

15.6.2 拍摄照片

移动电子相册可以直接调用 Android 系统本身的拍照功能，代码如下：

```
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(intent, 1);
```

当成功拍照后，系统会调用 onActivityResult 方法来接收图像数据，代码如下：

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == Activity.RESULT_OK)
    {
        switch (requestCode)
        {
            case 1:
                mBitmap = (Bitmap) data.getExtras().get("data");
                try
                {
                    // 将拍摄的照片显示在 ImageView 组件中
                    mivPhoto.setImageBitmap(mBitmap);
                }
                catch (Exception e)
                {
                }
        }
    }
}
```



```
        break;
    }
}
```

单击【拍摄照片】按钮开始拍摄照片，拍摄效果如图 15.2 所示。

15.6.3 从本地选择图像

通过如下的代码可以直接浏览手机中所有包含图像的目录：

```
Intent intent = new Intent();
intent.setType("image/*");
intent.setAction(Intent.ACTION_GET_CONTENT);
startActivityForResult(intent, 2);
```

下面的代码在 `onActivityResult` 方法中接收用户选择的图像数据，并将该数据显示在 `ImageView` 组件中。

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == Activity.RESULT_OK)
    {
        switch (requestCode)
        {
        case 2:
            Uri uri = data.getData();
            try
            {
                Cursor cursor = getContentResolver().query(uri, null,
                    null, null, null);
                cursor.moveToFirst();
                String imagePath = cursor.getString(1);
                cursor.close();
                Options options = new Options();
                // 根据图像实际的大小获取缩放比例
                options.inSampleSize = getSampleSize(imagePath);
                if (options.inSampleSize > 0)
                    mBitmap = BitmapFactory.decodeFile(imagePath,
                        options);
                else
                    mBitmap = BitmapFactory.decodeFile(imagePath);
                mivPhoto.setImageBitmap(mBitmap);
            }
            catch (Exception e)
            {
            }
            break;
        }
    }
}
```





人人都玩开心网：Ext JS + Android + SSH 整合开发 Web 与移动 SNS



在编写上面的代码时要注意，不要显示完全尺寸的图像，否则如果图像过大，系统会抛出内存溢出错误。为了使图像可以正常显示，需要通过 decodeFile 方法的第二个参数进行设置。其中 Options 类有一个 inSampleSize 字段，表示图像缩放比例。如果该字段为 n ，则 decodeFile 会按原图的 $1/n$ 创建 Bitmap 对象。



15.6.4 向服务端上传图像



由于上传图像需要直接向服务端发送 UploadPhotoData 对象，因此，需要使用 HttpURLConnection 直接操作 OutputStream，代码如下：

```
public static String uploadImage(int albumId, byte[] data)
{
    String result = "";
    try
    {
        URL url = new URL(
            "http://192.168.17.81:8080/kxw_service/UploadPhotoServlet");
        HttpURLConnection httpURLConnection = (HttpURLConnection) url
            .openConnection();
        httpURLConnection.setRequestMethod("POST");
        httpURLConnection.setDoInput(true);
        httpURLConnection.setDoOutput(true);
        httpURLConnection.setUseCaches(false);
        httpURLConnection.setRequestProperty("Charset", "UTF-8");
        // 获取向服务端发送数据的 OutputStream 对象
        OutputStream os = httpURLConnection.getOutputStream();
        // 创建 UploadPhotoData 对象
        UploadPhotoData uploadPhotoData = new UploadPhotoData();
        // 开始为 UploadPhotoData 对象的字段赋值
        uploadPhotoData.setUserId(Login.mUser.getId());
        uploadPhotoData.setUsername(username);
        uploadPhotoData.setPasswordMD5(passwordMD5);
        uploadPhotoData.setAlbumId(albumId);
        uploadPhotoData.setPhotoData(data);
        ObjectOutputStream oos = new ObjectOutputStream(os);
        // 向服务端发送 UploadPhotoData 对象
        oos.writeObject(uploadPhotoData);
        oos.close();
        InputStream is = httpURLConnection.getInputStream();
        InputStreamReader isr = new InputStreamReader(is);
        BufferedReader br = new BufferedReader(isr);
        // 从服务端获取返回结果
        result = br.readLine();
        is.close();
        httpURLConnection.disconnect();
        return result;
    }
    catch (Exception e)
    {
    }
}
```



单击【上传图像】按钮，会调用如下的代码上传 ImageView 组件中显示的图像。

```
if (mBitmap == null)
    return;
ByteArrayOutputStream baos = new ByteArrayOutputStream();
mBitmap.compress(CompressFormat.JPEG, 100, baos);
// 调用 uploadImage 方法上传图像
result = CallService.uploadImage(mAlbums.get(
    mCurrentAlbumPosition).getId(), baos.toByteArray());
if (!"ok".equals(result))
{
    Message.showMsg(this, result);
}
else
{
    Message.showMsg(this, "成功上传图像");
}
```



15.7 本章小结

本章主要介绍了移动电子相册中用到的 Android 组件和网络功能。其中的组件包括 Button、TextView、EditText、ImageView、Spinner、Gallery 和 TabHost。为了演示 Android 访问网络的能力，在移动电子相册中同时使用了 HttpClient 和 HttpURLConnection 来访问网络，在本章的最后详细介绍了移动电子相册的实现过程。