项目中有时候需要获取网络上的图片，并下载下来到手机客户端显示。怎么做呢？

实现思路是：

1：在UI线程中启动一个线程,让这个线程去下载图片。

2：图片完成下载后发送一个消息去通知UI线程

2：UI线程获取到消息后，更新UI。

这里的UI线程就是主线程。

这两个步骤涉及到一些知识点，即是：﻿﻿﻿ProgressDialog,Handler,Thread/Runnable,URL,HttpURLConnection等等一系列东东的使用。

现在让我们开始来实现这个功能吧！

第一步：新建项目。

第二步：设计好UI，如下所示

<?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"

>

<Button

android:id="@+id/btnFirst"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="异步下载方式一"

>

</Button>

<Button

android:id="@+id/btnSecond"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="异步下载方式二"

>

</Button>

<FrameLayout

android:layout\_width="fill\_parent"

android:layout\_height="match\_parent"

android:id="@+id/frameLayout"

>

<ImageView

android:id="@+id/image"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:scaleType="centerInside"

android:padding="2dp"

>

</ImageView>

<ProgressBar

android:id="@+id/progress"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center">

</ProgressBar>

</FrameLayout>

</LinearLayout>

第三步：获取UI相应View组件，并添加事件监听。

public class DownLoaderActivity extends Activity implements OnClickListener{

private static final String params="http://upload.wikimedia.org/wikipedia/commons/thumb/e/ea/Hukou\_Waterfall.jpg/800px-Hukou\_Waterfall.jpg";

private Button btnFirst,btnSecond;

private ProgressBar progress;

private FrameLayout frameLayout;

private Bitmap bitmap=null;

ProgressDialog dialog=null;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

btnFirst=(Button)this.findViewById(R.id.btnFirst);

btnSecond=(Button)this.findViewById(R.id.btnSecond);

progress=(ProgressBar)this.findViewById(R.id.progress);

progress.setVisibility(View.GONE);

frameLayout=(FrameLayout)this.findViewById(R.id.frameLayout);

btnFirst.setOnClickListener(this);

btnSecond.setOnClickListener(this);

}

第四步：在监听事件中处理我们的逻辑，即是下载服务器端图片数据。

这里我们需要讲解一下了。

通常的我们把一些耗时的工作用另外一个线程来操作，比如，下载上传图片，读取大批量XML数据，读取大批量sqlite数据信息。为什么呢？答案大家都明白，用户体验问题。

在这里，首先我构造一个进度条对话框，用来显示下载进度，然后开辟一个线程去下载图片数据，下载数据完毕后，通知主UI线程去更新显示我们的图片。

Handler是沟通Activity 与Thread/runnable的桥梁。而Handler是运行在主UI线程中的，它与子线程可以通过Message对象来传递数据。具体代码如下：

/\*\*这里重写handleMessage方法，接受到子线程数据后更新UI\*\*/

private Handler handler=new Handler(){

@Override

public void handleMessage(Message msg){

switch(msg.what){

case 1:

//关闭

ImageView view=(ImageView)frameLayout.findViewById(R.id.image);

view.setImageBitmap(bitmap);

dialog.dismiss();

break;

}

}

};

我们在这里弹出进度对话框，使用HTTP协议来获取数据。

//前台ui线程在显示ProgressDialog，

//后台线程在下载数据，数据下载完毕，关闭进度框

@Override

public void onClick(View view) {

switch(view.getId()){

case R.id.btnFirst:

dialog = ProgressDialog.show(this, "",

"下载数据，请稍等 …", true, true);

//启动一个后台线程

handler.post(new Runnable(){

@Override

public void run() {

//这里下载数据

try{

URL url = new URL(params);

HttpURLConnection conn = (HttpURLConnection)url.openConnection();

conn.setDoInput(true);

conn.connect();

InputStream inputStream=conn.getInputStream();

bitmap = BitmapFactory.decodeStream(inputStream);

Message msg=new Message();

msg.what=1;

handler.sendMessage(msg);

} catch (MalformedURLException e1) {

e1.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

});

break;

如此以来，你会发现很好的完成了我们的下载目标了，你可以把它应用到其他方面去，举一反三。