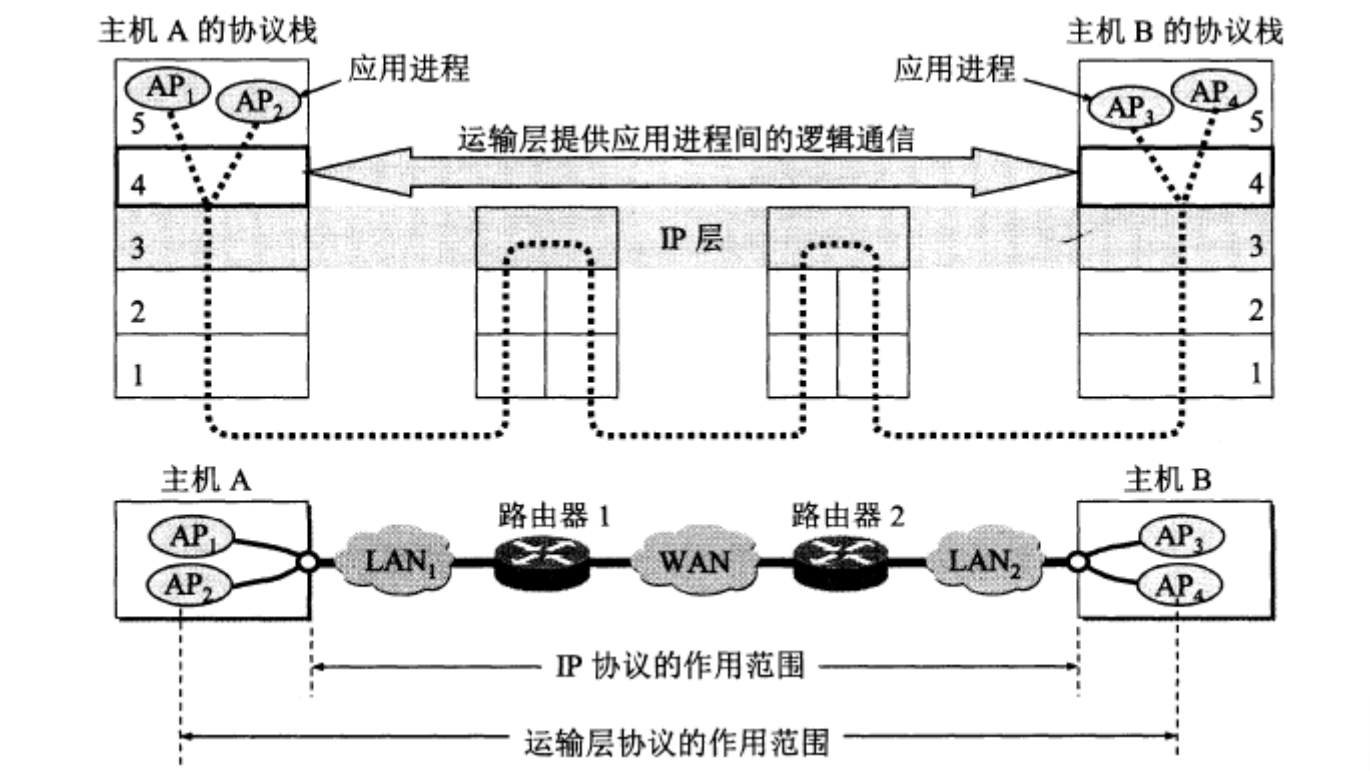
ANDROID移动互联网应用开发

傅晓，河海大学计算机与信息学院
2019年3月

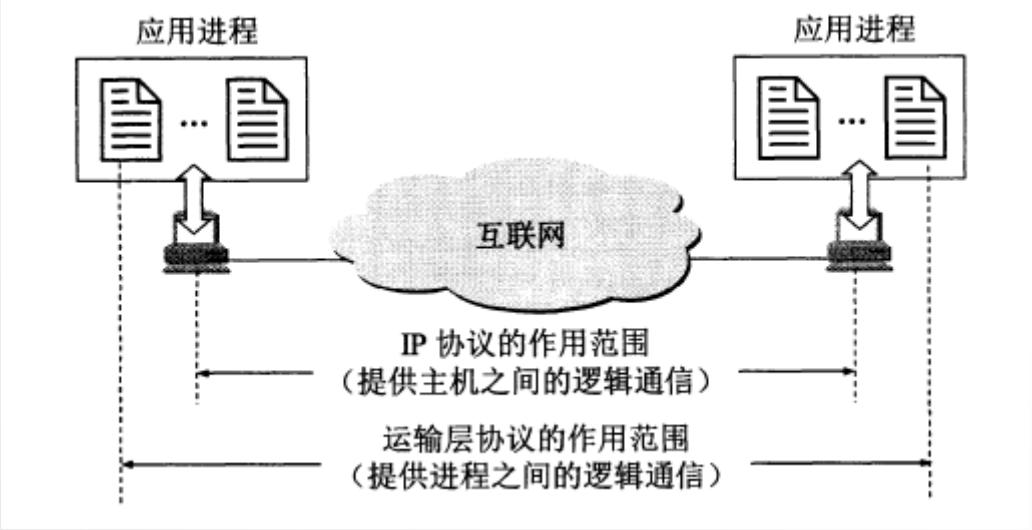
运输层协议

- 好像这种通信沿水平方向上直接传送数据，但是事实上这两个运输层之间并没有一条水平方向的物理连接，数据的传送是沿着虚线方向经过多个层次传送的：



运输层协议

- 网络层为主机之间提供逻辑通信，而运输层为应用程序之间提供端到端的逻辑通信；
- 运输层对收到的报文进行差错检查；
- 运输层向高层用户屏蔽了下面网络核心的细节，采用TCP协议时，相当于一条全双工的可靠信道，采用UDP协议时，仍然是一条不可靠信道。



运输层协议

- 1，用户数据包协议UDP（User Datagram Protocol）：不需要先建立连接；
- 2，传输控制协议TCP（Transmission Control Protocol）：提供面向连接的服务：

应用	应用层协议	运输层协议
名字转换	DNS（域名系统）	UDP
文件传送	TFTP（简单文件传送协议）	UDP
路由选择协议	RIP（路由信息协议）	UDP
IP 地址配置	DHCP（动态主机配置协议）	UDP
网络管理	SNMP（简单网络管理协议）	UDP
远程文件服务器	NFS（网络文件系统）	UDP
IP 电话	专用协议	UDP
流式多媒体通信	专用协议	UDP
多播	IGMP（网际组管理协议）	UDP
电子邮件	SMTP（简单邮件传送协议）	TCP
远程终端接入	TELNET（远程终端协议）	TCP
万维网	HTTP（超文本传送协议）	TCP
文件传送	FTP（文件传送协议）	TCP

端口

- 复用：应用层所有应用进程都可以通过运输层再传送到IP层；
- 分用：运输层从IP层受到发送给各应用进程的数据后，必须分别交付指明的各应用进程；
- 运输层使用协议端口号（protocol port number）或端口（port）识别不同的应用进程，这种在协议栈层间的抽象的协议端口是软件端口；
- 硬件端口是不同硬件进行交互的接口，软件端口是应用层的各种协议进程与运输实体进行层间交互的一种地址；
- 端口号只具有本地意义，16位端口号允许有65535个不同的的端口号

端口

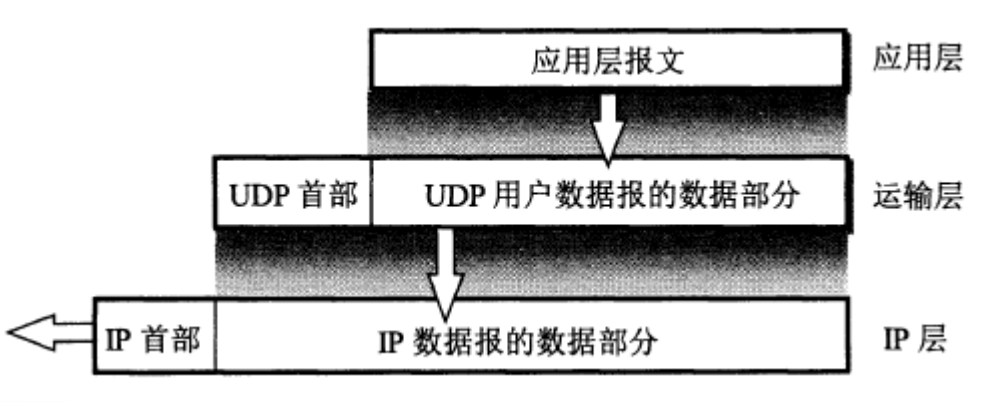
- 端口号分为两大类：
- 1，服务器使用的端口号：分为两类：
- 熟知端口号（系统端口号）：0～1023；
- 登记端口号：1024～49151；

应用程序	FTP	TELNET	SMTP	DNS	TFTP	HTTP	SNMP	SNMP (trap)	HTTPS
熟知端口号	21	23	25	53	69	80	161	162	443

- 2，客户端使用的端口号（短暂端口号）：49152~65535。

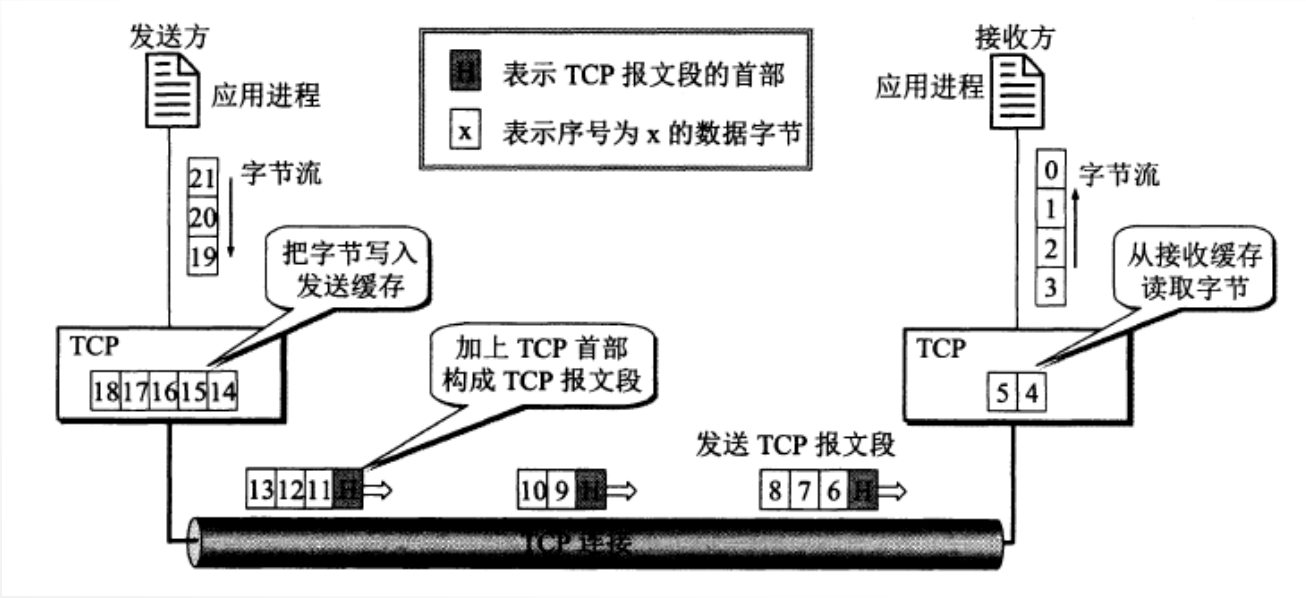
用户数据报协议UDP

- UDP主要特点:
- 1，UDP是无连接的;
- 2，UDP使用尽最大努力交付;
- 3，UDP是面向报文的;
- 4，UDP没有拥塞控制;
- 5，UDP支持一对一、一对多、多对一、多对多的交互通信;
- 6，UDP的首部开销小。



传输控制协议TCP

- TCP主要特点：
- 1，TCP是面向连接的运输层协议；
- 2，每一条TCP连接只能是点对点的；
- 3，TCP提供可靠交付服务；
- 4，TCP提供全双工通信；
- 5，面向字节流（stream）。



套接字SOCKET

- TCP把连接作为最基本的抽象；
- TCP连接的端点叫做套接字（socket），端口号拼接到IP地址即构成套接字，其表示方法是在点分十进制的IP地址后面加上端口号，中间以冒号隔开，如192.3.4.5：80；

$$\text{套接字 socket} = (\text{IP 地址: 端口号})$$

- 每一条TCP连接唯一的被通信两端的两个端点（套接字）确定：

$$\text{TCP 连接} ::= \{\text{socket}_1, \text{socket}_2\} = \{(\text{IP}_1: \text{port}_1), (\text{IP}_2: \text{port}_2)\}$$

套接字SOCKET

- 套接字（socket）可以表示多种不同意思：
- 1，允许应用程序访问连网协议接口的应用编程接口API称为socket API；
- 2，在socket API中使用的一个函数名叫做socket；
- 3，调用socket函数的端点称为socket；
- 4，调用socket函数时，其返回值称为socket描述符；
- 5，在OS内核中连网协议的Berkeley实现，称为socket实现。

套接字SOCKET

- 客户端Socket：
- Socket(String host, int port) throws UnknownHostException, IOException：创建一个流套接字并将其连接到指定主机上的指定端口号。
- getInputStream()：获取此套接字的输入流，收到的数据就在这里。
- getOutputStream()：传送此套接字的输出流， 要发送的数据放到这里。
- Close()：断开此套接字的连接，释放相关资源。

套接字SOCKET

- 服务器端ServerSocket：
- ServerSocket(int port)：创建服务端的监听port端口的套接字方法。
- Socket accept() throws IOException：侦听并接受到此套接字的连接。此方法在连接传入之前一直阻塞。服务端通过这个方法拿到与客户端建立端到端的连接的socket。
- Close()：断开此套接字的连接，释放相关资源。

套接字SOCKET


Server
创建服务端的监听
指定端口的服务端对象
监听端口，一直阻塞，
等待端口返回数据
收到对应端口的数据，
拿到对应的socket对象
从socket拿到客户端发
送的数据（在InputStream中）
将数据放入socket的
OutputStream中返回给客户端
关闭socket以及其他资源
建立连接
数据传输
Client
创建客户端的对应（端
口+IP地址）的socket对象
将数据放入socket的
OutputStream中发送给服务器
从socket拿到服务器返回
的数据（在InputStream中）
关闭socket以
及其他资源

套接字SOCKET

- Socket异常：
- BindException：如果无法与本机指定的IP地址或端口绑定，就会抛出此异常。
- UnknownHostException：如果指定的主机名或IP地址无法识别，就会抛出此异常。
- ConnectException：服务器没有监听指定的端口，或服务器socket指定的backlog队列已满。
- SocketTimeoutException：如果在timeout之间没有能连接成功，就会抛出此异常。

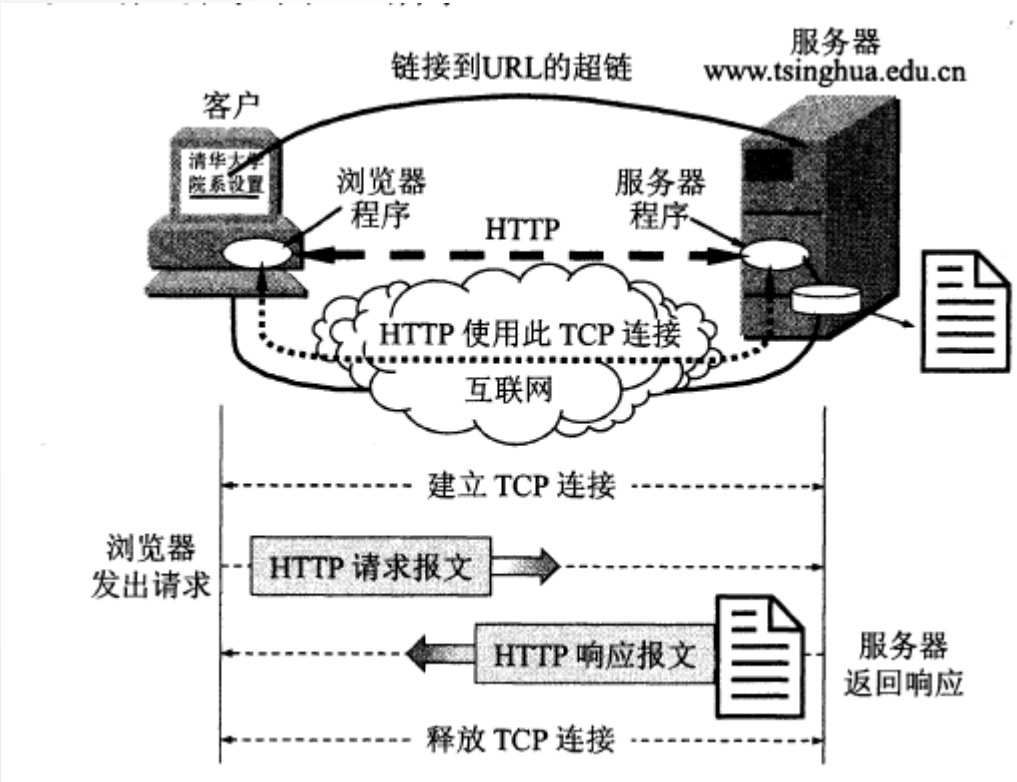
统一资源定位符URL

- 统一资源定位符URL（Uniform Resource Locator），有时也被俗称为网页地址（网址），如同在网络上的门牌，是因特网上标准的资源的地址（Address）。它最初是由蒂姆·伯纳斯-李发明用来作为万维网的地址。现在它已经被万维网联盟编制为因特网标准RFC 1738；
- 在互联网的历史上，统一资源定位符的发明是一个非常基础的步骤。统一资源定位符的语法是一般的，可扩展的，它使用ASCII代码的一部分来表示因特网的地址。统一资源定位符的开始，一般会标志着一个计算机网络所使用的网络协议。统一资源定位符的标准格式如下：
- 协议类型:[//服务器地址[:端口号]][/资源层级UNIX文件路径]文件名[?查询][#片段ID]
- 统一资源定位符的完整格式如下：
- 协议类型:[//[访问资源需要的凭证信息@]服务器地址[:端口号]][/资源层级UNIX文件路径]文件名[?查询][#片段ID]
- 其中【访问凭证信息@；:端口号；?查询；#片段ID】都属于选填项。

超文本传送协议HTTP

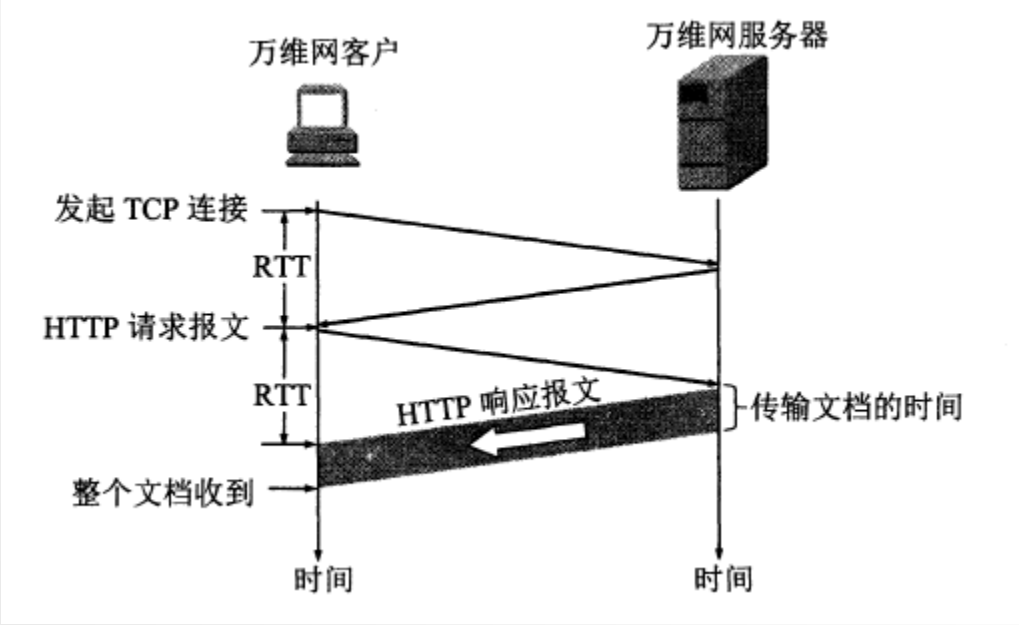
- 超文本传送协议HTTP（HyperText Transfer Protocol）是一种用于分布式、协作式和超媒体信息系统的应用层协议。HTTP是万维网的数据通信的基础。设计HTTP最初的目的是为了提供一种发布和接收HTML页面的方法。通过HTTP或者HTTPS协议请求的资源由统一资源标识符URI （Uniform Resource Identifiers）来标识；
- HTTP是一个客户端终端（用户）和服务器端（网站）请求和应答的标准（TCP）。通过使用网页浏览器、网络爬虫或者其它的工具，客户端发起一个HTTP请求到服务器上指定端口（默认端口为80）。我们称这个客户端为用户代理程序（user agent）。应答的服务器上存储着一些资源，比如HTML文件和图像。我们称这个应答服务器为源服务器（origin server）。在用户代理和源服务器中间可能存在多个“中间层”，比如代理服务器、网关或者隧道（tunnel）。

超文本传送协议HTTP



超文本传送协议HTTP

- HTTP协议是无连接的，虽然HTTP使用了TCP连接，但是通信双方在交换HTTP报文之前不需要先建立HTTP连接；
- HTTP协议是无状态的，同一个客户第二次访问同一个服务器上的页面时，服务器的响应与第一次被访问时相同，服务器不记得曾经访问的客户，也不记得为客户服务过多少次。



SOCKET与HTTP

- HTTP：采用请求—响应方式。即建立网络连接后，当客户端向服务器发送请求后，服务器端才能向客户端返回数据。
- 可理解为：是客户端有需要才进行通信。
- Socket：采用服务器主动发送数据的方式即建立网络连接后，服务器可主动发送消息给客户端，而不需要由客户端向服务器发送请求。
- 可理解为：是服务器端有需要才进行通信。

ASYNCTASK

- 在Android4.0之后，在主线程里面执行Http请求都会报NetworkOnMainThreadException的异常，正常来说，是不应该再主线程中进行网络请求和UI操作的，这些操作往往耗费大量时间，会造成主线程的阻塞，应使用Handler+Thread或AsyncTask。
- AsyncTask是一个抽象类，它是由Android封装的一个轻量级异步类它的内部封装了两个线程池(SerialExecutor和THREAD_POOL_EXECUTOR)和一个Handler(InternalHandler)，其中SerialExecutor线程池用于任务的排队，THREAD_POOL_EXECUTOR线程池才真正地执行任务，InternalHandler用于从工作线程切换到主线程。
- 与Handler+Thread比较， AsyncTask使用简单，代码简洁，过程可控，可调用cancel()取消任务。

ASYNCTASK

- 参数：
- public abstract class AsyncTask<Params, Progress, Result>
- Params: 传入参数类型，即doInBackground()方法中的参数类型。
- Progress: 异步任务执行过程中返回的下载进度类型，即publishProgress()和onProgressUpdate()方法中传入的参数类型。
- Result: 异步任务执行完返回的结果类型，即doInBackground()方法中返回值的类型。

ASYNCTASK

- 方法：
- onPreExecute(): 在主线程中执行，异步任务执行前，此方法会调用，可做一些准备工作；
- doInBackground(): 在线程池中执行，核心方法，用于执行异步操作，此方法中可调用publishProgress()方法来更新进度，publishProgress()方法会调用onProgressUpdate()方法，此方法的返回结果会传给onPostExecute()方法；
- onProgressUpdate(): 在主线程中执行，publishProgress()方法会调用此方法，用于更新进度；
- onPostExecute(): 在主线程中执行，在异步任务执行结束后，此方法会调用，一般用于处理返回结果。
- onCancelled() :它同样在主线程中执行，当异步任务取消时，onCancelled()会被调用。

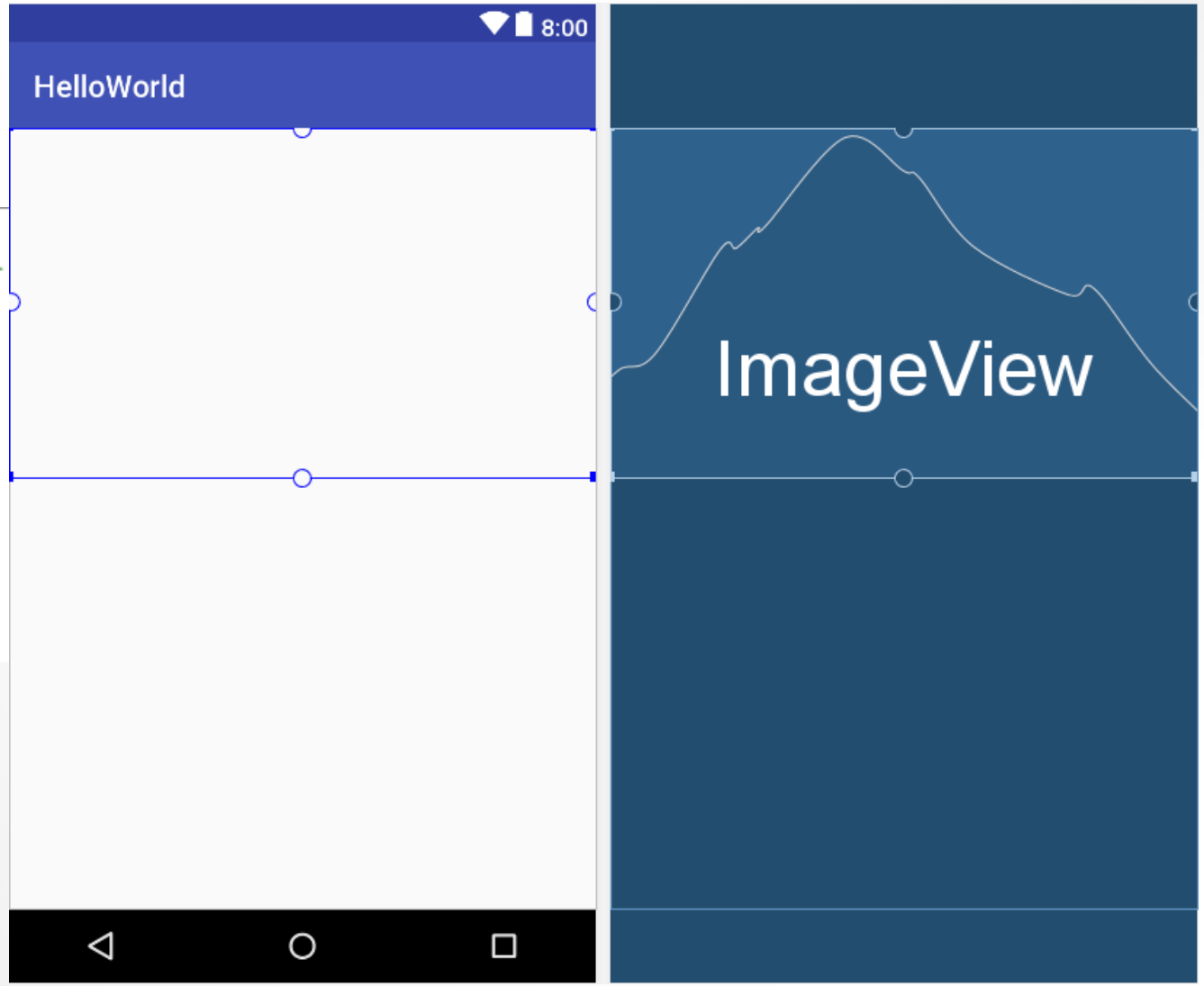
ASYNCTASK

- 使用AsyncTask访问网络资源需在AndroidManifest.xml中申请android.permission.INTERNET权限：

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bishop.helloworld">
    <uses-permission android:name="android.permission.INTERNET"></uses-permission>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloWorld"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

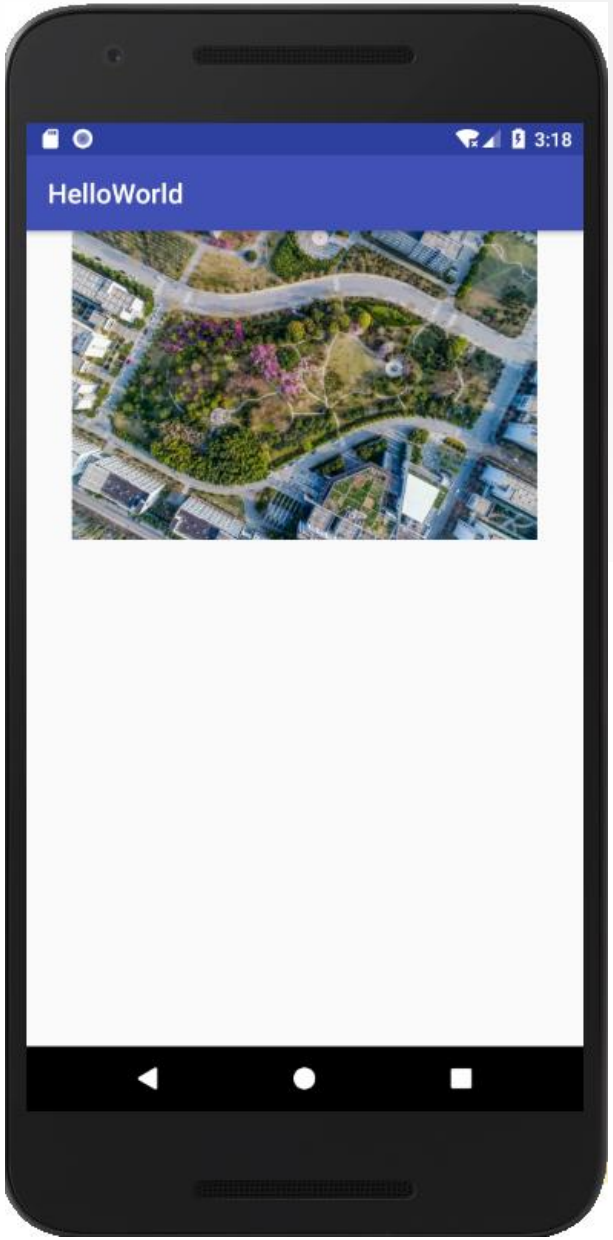
ASYNCTASK

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.
    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.example.bishop.helloworld.MainActivity">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="229dp"
        android:id="@+id/imageView"/>
</android.support.constraint.ConstraintLayout>
```



ASYNCTASK

```
public void ShowPicture(String url){
    new AsyncTask<String, Void, String>(){
        @Override
        protected String doInBackground(String... params){
            try{
                URL url=new URL(params[0]);
                InputStream is=url.openStream();
                bm= BitmapFactory.decodeStream(is);
                is.close();
            }catch (MalformedURLException e){
                e.printStackTrace();
            }catch (Exception e){
                e.printStackTrace();
            }
            return null;
        }
        @Override
        protected void onPostExecute(String s){
            super.onPostExecute(s);
            img.setImageBitmap(bm);
        }
    }.execute(url);
}
```




```
package com.example.bishop.helloworld;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.ImageView;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
public class MainActivity extends AppCompatActivity {
    ImageView img;
    Bitmap bm;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        img=(ImageView)findViewById(R.id.imageView);
        ShowPicture( url: "https://mmbiz.qpic.cn/mmbiz_jpg/8ccm0WAxBbhuUlAunFDMfoqThWfCpqOoibx6xCBJL4hsuMpSL90uiaPdibgB1qYsb2He207Z9Tiax1udjpevUmWFicw/640?wx_fmt=jpeg&wxfrom=5&wx_lazy=1");
    }
```

URLCONNECTION

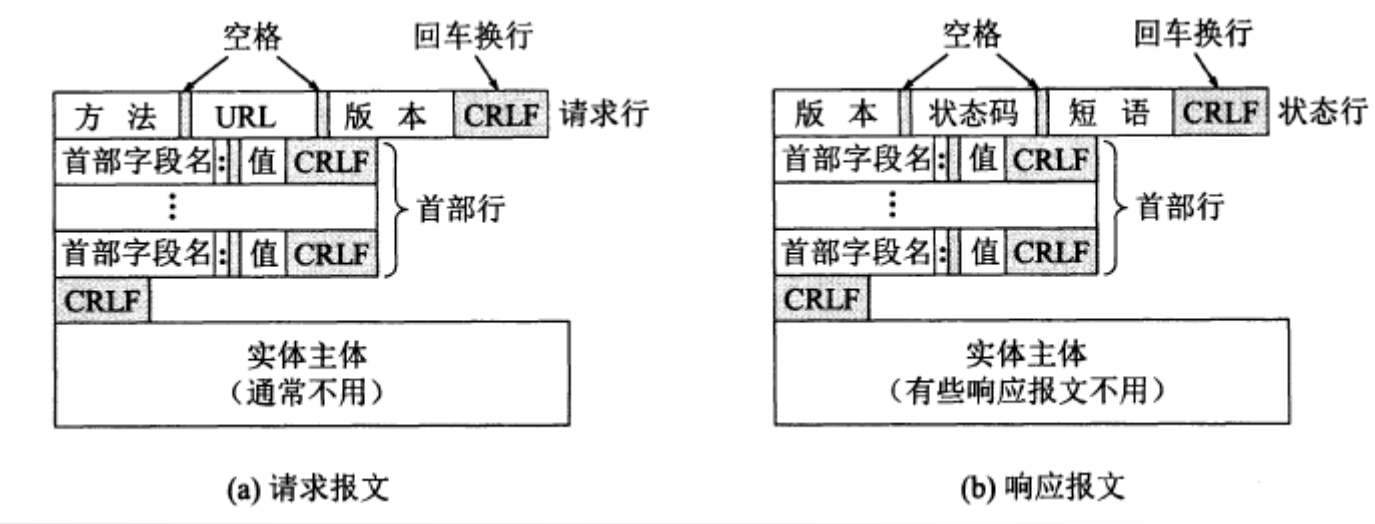
- 通常创建一个和URL的连接，并发送请求、读取此URL引用的资源需要如下几个步骤：
- 通过调用URL对象openConnection()方法来创建URLConnection对象
- 设置URLConnection的参数和普通请求属性。
- 如果只是发送GET方法请求，使用connect方式建立和远程资源之间的实际连接即可；如果需要发送POST方式的请求，需要获取URLConnection实例对应的输出流来发送请求参数。
- 远程资源变为可用，程序可以访问远程资源的头字段，或通过输入流读取远程资源的数据。
- 关闭输入流。

URLCONNECTION

- Object getContent()：获取该URLConnection的内容。
- String getHeaderField(String name)：获取指定响应头字段的值。
- getInputStream()：返回该URLConnection对应的输入流，用于获取URLConnection响应的内容。
- getOutputStream()：返回该URLConnection对应的输出流，用于向URLConnection发送请求参数。
- setAllowUserInteraction：设置该URLConnection的allowUserInteraction请求头字段的值。
- setDoInput：设置该URLConnection的doinput请求头字段的值。
- setDoOutput：设置该URLConnection的dooutput请求头字段的值。
- setUseCaches：设置该URLConnection的useCaches请求头字段的值。

超文本传送协议HTTP

- HTTP有两类报文：
- 1，请求报文：从客户向服务器发送请求报文；
- 2，响应报文：从服务器到客户的回答；



超文本传送协议HTTP

- 由于HTTP是面向文本的（text-oriented），报文中每一字段都是一些ASCII码串，各个字段长度都是不确定的：
- 1，开始行：区分请求报文和响应报文，在请求报文中叫请求行，在响应报文中叫状态行；
- 2，首部行：说明浏览器、服务器、报文主体的信息；
- 3，实体主体：请求报文中一般不用，响应报文中可能没有。

超文本传送协议HTTP

- 由于HTTP是面向文本的（text-oriented），报文中每一字段都是一些ASCII码串，各个字段长度都是不确定的：
- 1，开始行：区分请求报文和响应报文，在请求报文中叫请求行，在响应报文中叫状态行；
- 2，首部行：说明浏览器、服务器、报文主体的信息；
- 3，实体主体：请求报文中一般不用，响应报文中可能没有。

超文本传送协议HTTP

- 请求行有三个内容：
- 方法（method），请求资源的URL，HTTP的版本。

方法（操作）	意义
OPTION	请求一些选项的信息
GET	请求读取由 URL 所标志的信息
HEAD	请求读取由 URL 所标志的信息的首部
POST	给服务器添加信息（例如，注释）
PUT	在指明的 URL 下存储一个文档
DELETE	删除指明的 URL 所标志的资源
TRACE	用来进行环回测试的请求报文
CONNECT	用于代理服务器

```
GET http://www.xyz.edu.cn/dir/index.htm HTTP/1.1
```

```
GET /dir/index.htm HTTP/1.1                    {请求行使用了相对 URL}
Host: www.xyz.edu.cn                {此行是首部行的开始。这行给出主机的域名}
Connection: close                   {告诉服务器发送完请求的文档后就可释放连接}
User-Agent: Mozilla/5.0             {表明用户代理是使用火狐浏览器 Firefox}
Accept-Language: cn                 {表示用户希望优先得到中文版本的文档}
                                    {请求报文的最后还有一个空行}
```

超文本传送协议HTTP

- 状态行包括三个内容：

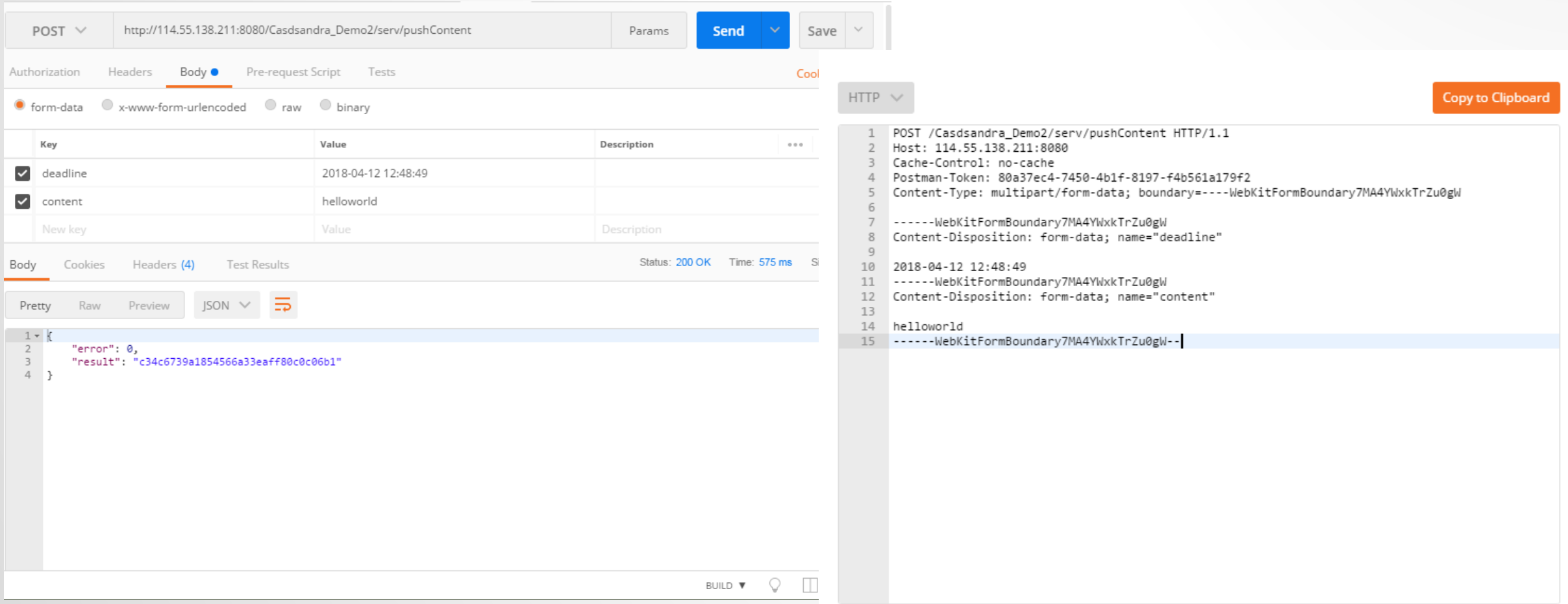
```
HTTP/1.1 301 Moved Permanently          {永久性地转移了}
Location: http://www.xyz.edu/ee/index.html    {新的 URL}
```

- HTTP版本，状态码，解释状态码的简单短语。
- HTTP状态码（Status Code）是用以表示网页服务器超文本传输协议响应状态的3位数字代码，所有状态码的第一个数字代表了响应的五种状态之一：

	类别	原因短语
1XX	Informational（信息性状态码）	接收的请求正在处理
2XX	Success（成功状态码）	请求正常处理完毕
3XX	Redirection（重定向状态码）	需要进行附加操作以完成请求
4XX	Client Error（客户端错误状态码）	服务器无法处理请求
5XX	Server Error（服务器错误状态码）	服务器处理请求出错

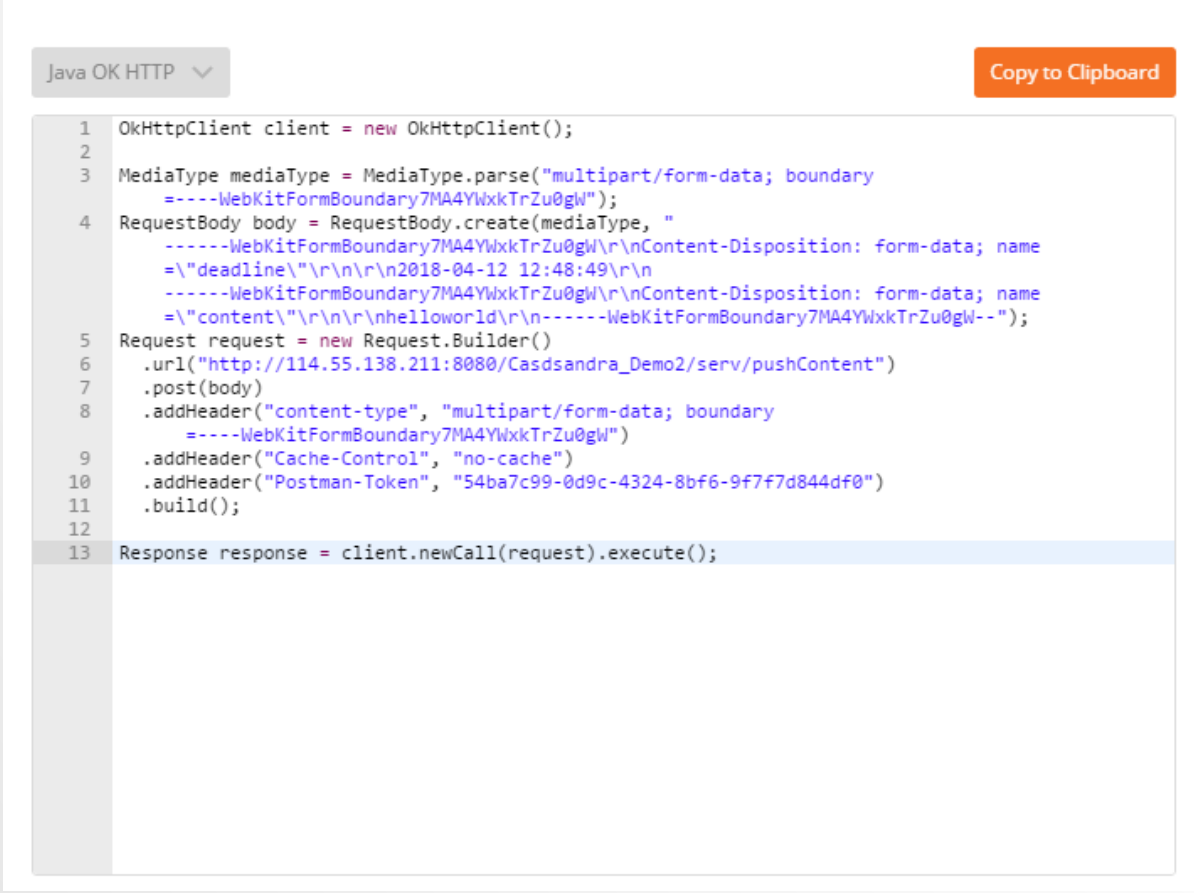
超文本传送协议HTTP

- 使用Postman测试HTTP请求数据，并获取响应值：



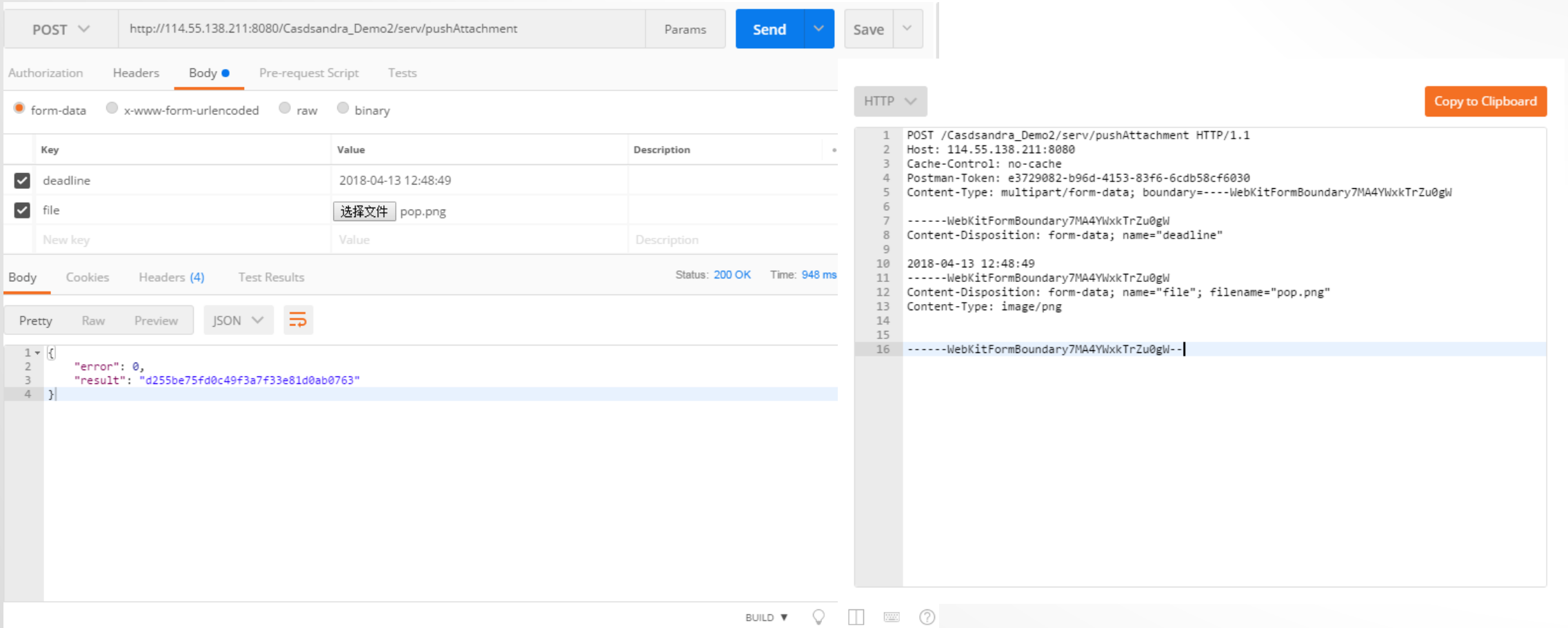
超文本传送协议HTTP

- 在Postman中，可根据HTTP请求数据自动生成包括OkHTTP在内的多种开发环境的代码，只需要复制粘贴至Android Studio中即可：



超文本传送协议HTTP

- Postman中可以在Content-Disposition中设置Multipart/data-form类型，同时在一次HTTP请求中发送文件和键值对：



超文本传送协议HTTP

- 在Android Studio中使用OkHTTP类发送文件和键值对的代码：

Java OK HTTP

Copy to Clipboard

```
OkHttpClient client = new OkHttpClient();

MediaType mediaType = MediaType.parse("multipart/form-data; boundary
    =----WebKitFormBoundary7MA4YWxkTrZu0gW");
RequestBody body = RequestBody.create(mediaType, "
    ------WebKitFormBoundary7MA4YWxkTrZu0gW\r\nContent-Disposition: form-data; name
    =\"deadline\"\r\n\r\n2018-04-13 12:48:49\r\n
    ------WebKitFormBoundary7MA4YWxkTrZu0gW\r\nContent-Disposition: form-data; name=\"file\";
    filename=\"C:\\Users\\Bishop\\Pictures\\pop.png\"\r\nContent-Type: image/png\r\n\r\n\r\n
    ------WebKitFormBoundary7MA4YWxkTrZu0gW--");
Request request = new Request.Builder()
  .url("http://114.55.138.211:8080/Casdsandra_Demo2/serv/pushAttachment")
  .post(body)
  .addHeader("content-type", "multipart/form-data; boundary
      =----WebKitFormBoundary7MA4YWxkTrZu0gW")
  .addHeader("Cache-Control", "no-cache")
  .addHeader("Postman-Token", "53024638-6cc1-443c-a0d4-03e79f60b182")
  .build();

Response response = client.newCall(request).execute();
```

URLCONNECTION

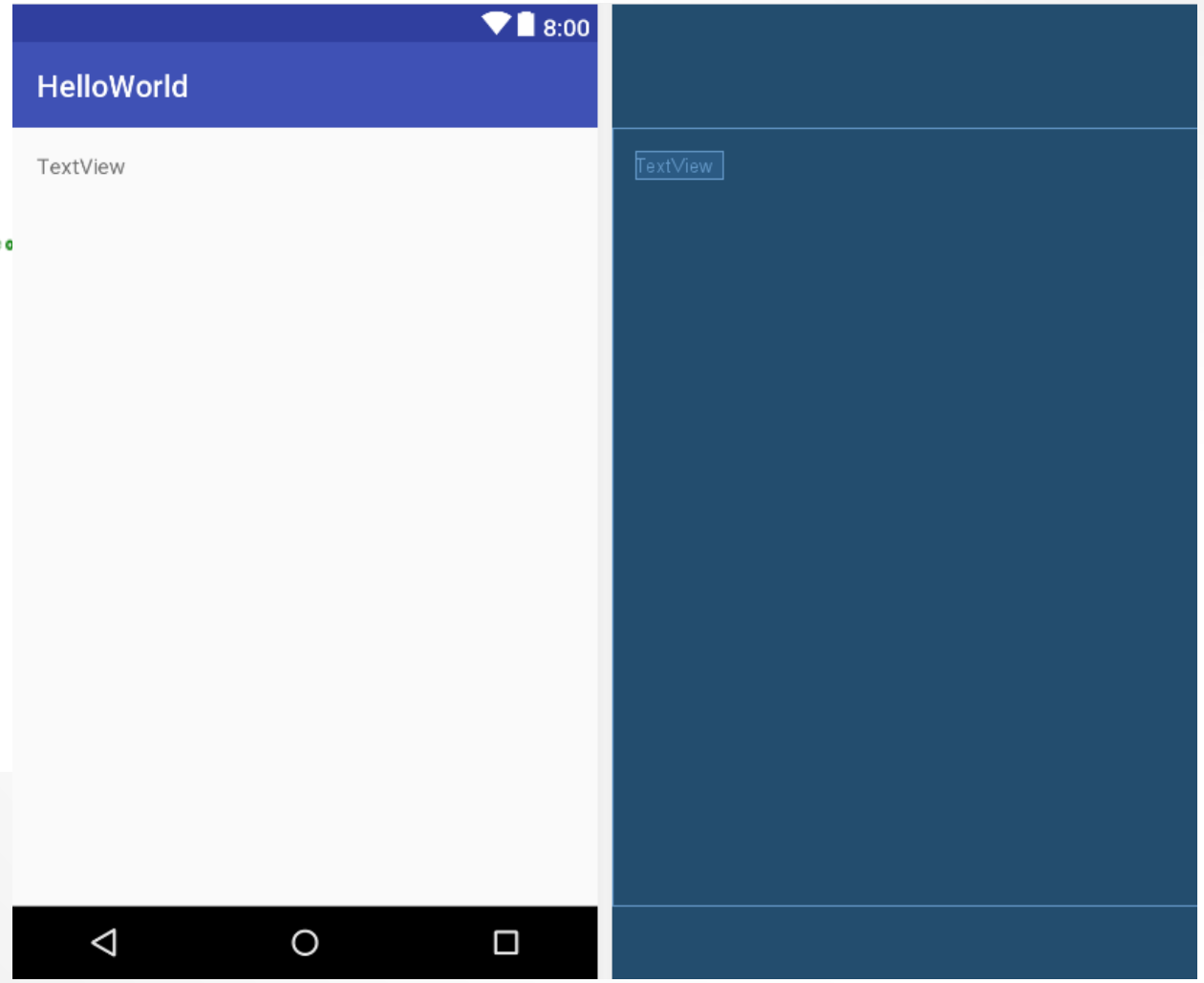
- Get请求：
- String urlString = url + "?" + params;
- URL realUrl = new URL(urlString);
- URLConnection conn = realUrl.openConnection();
- conn.setRequestProperty("accept", "*/*");
- conn.setRequestProperty("connection", "Keep-Alive");
- conn.setRequestProperty("user-agent", "Mozilla/~~~~~~~~~~~~");
- conn.connect();

URLCONNECTION

- Post请求：
- URL realUrl = new URL(url);
- URLConnection conn = realUrl.openConnection();
- conn.setRequestProperty("accept", "*/*");
- conn.setRequestProperty("connection", "Keep-Alive");
- conn.setRequestProperty("user-agent", "Mozilla/~~~~~~~~~~~~");
- conn.setDoOutput(true);
- conn.setDoInput(true);
- out = new PrintWriter(conn.getOutputStream());
- out.print(params);
- out.flush();

URLCONNECTION

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.co
    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.example.bishop.helloworld.MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        tools:layout_editor_absoluteX="16dp"
        tools:layout_editor_absoluteY="16dp" />
</android.support.constraint.ConstraintLayout>
```



URLCONNECTION

```java
package com.example.bishop.helloworld;

import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.TextView;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;

public class MainActivity extends AppCompatActivity {
    TextView text;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        text=(TextView) findViewById(R.id.textView);
        login( name: "admin", password: "123456");
    }
```


```java
    public void login(String name,String password){
        final String url="http://mail.hhu.edu.cn/coremail/index.jsp";
        final String param="uid="+name+"&"+"password="+password;
        new AsyncTask<String, Void,String>(){
            @Override
            protected String doInBackground(String... params){
                PrintWriter out=null;
                BufferedReader in=null;
                String result=null;
                try{
                    URL url=new URL(params[0]);
                    URLConnection conn=url.openConnection();
                    conn.setRequestProperty("accept", "*/*");
                    conn.setRequestProperty("connection", "Keep-Alive");
                    conn.setRequestProperty("user-agent", "Mozilla/~~~~~~~~~~~~");
                    conn.setDoInput(true);
                    conn.setDoOutput(true);
                    out=new PrintWriter(conn.getOutputStream());
                    out.print(param);
                    in=new BufferedReader(new InputStreamReader(conn.getInputStream()));
                    String line;
                    while((line=in.readLine())!=null){
                        result+=line;
                    }
                }catch (MalformedURLException e){
                    e.printStackTrace();
                }catch (Exception e){
                    e.printStackTrace();
                }
                return result;
            }
            @Override
            protected void onPostExecute(String result){
                if(result!=null){
                    text.setText(result);
                }
                super.onPostExecute(result);
            }
        }.execute(url);
    }
}
```

HTTPURLCONNECTION

- Android 6.0(API 23) SDK后，Android的网络请求强制使用HttpUrlConnection，并且SDK中也移除了HttpClient库，同时也移除了SSL 和Notification的setLatestEventInfo方法。
- HttpURLConnection继承了URLConnection，因此也可用于向指定网站发送Get请求、Post请求。它在URLConnection的基础上提供了如下便捷方法：

HTTPURLCONNECTION

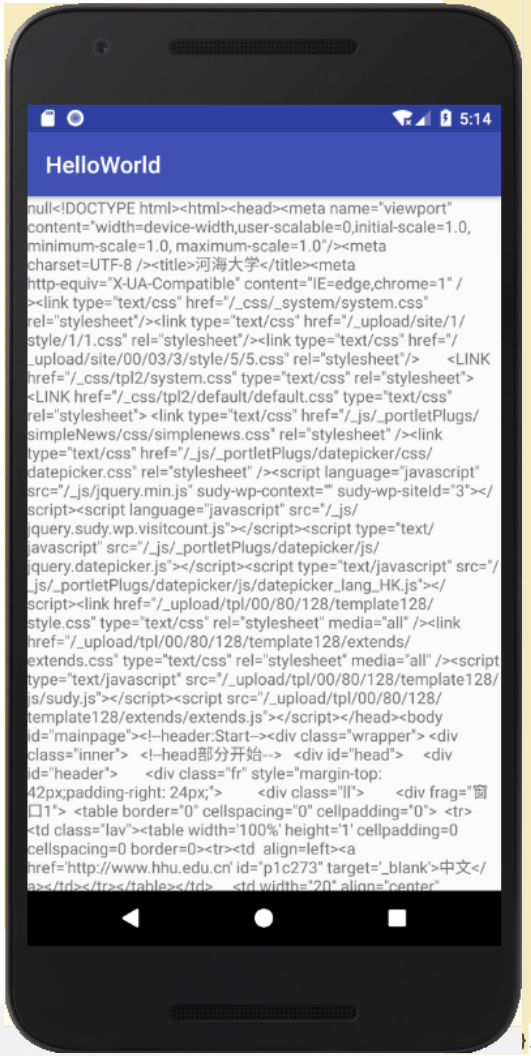
- getContentLength()：获取指定url的资源大小。
- getResponseCode()：获取服务器的响应代码。
- getResponseMessage()：获取服务器的响应信息。
- getRequestMethod()：获取发送请求的方法。
- setRequestMethod(method)：设置发送请求的方法。
- setConnectTimeOut()：设置连接超时时间。
- setRequestProperty()：设置请求的普通属性。

HTTPURLCONNECTION

- 示例:
- URL url = new URL("https://www.csdn.net");
- connection = (HttpURLConnection)url.openConnection();
- connection.setRequestMethod("GET");
- connection.setConnectTimeout(8000);
- connection.setReadTimeout(8000);
- InputStream in = connection.getInputStream();
- ...

HTTPURLCONNECTION

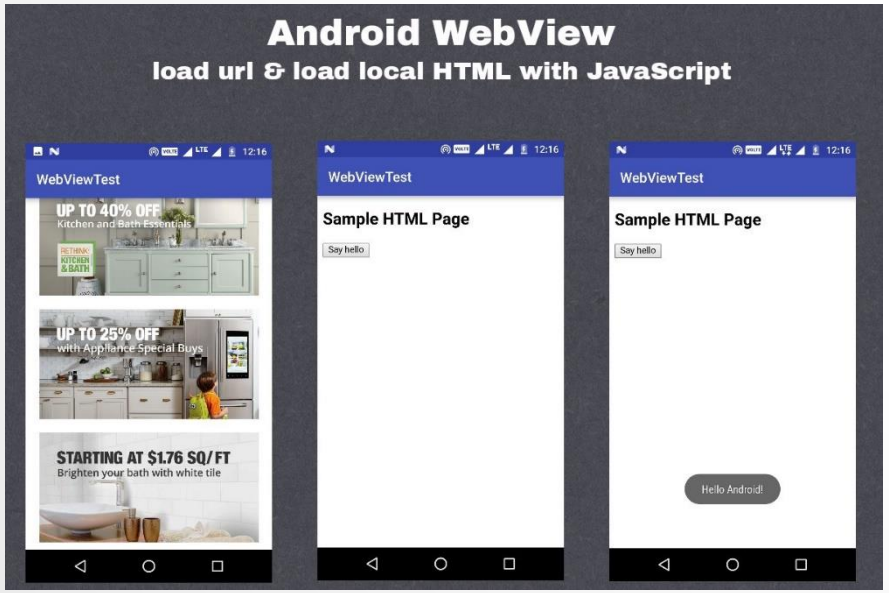
```java
package com.example.bishop.helloworld;

import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.TextView;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;

public class MainActivity extends AppCompatActivity {
    TextView text;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        text=(TextView) findViewById(R.id.textView);
        SimpleDateFormat dateFormat=new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss");
        Date date=new Date(System.currentTimeMillis()+28800*1000+3600*1000);
        push( content: "helloworld",dateFormat.format(date));
    }
```




```java
public void push(final String content, final String deadline){
    final String url="http://www.hhu.edu.cn";
    new AsyncTask<String, Void, String>(){
        @Override
        protected String doInBackground(String... params) {
            String result = null;
            try {
                URL url = new URL(params[0]);
                HttpURLConnection conn = (HttpURLConnection) url.openConnection();
                conn.setRequestProperty("accept", "*/*");
                conn.setRequestProperty("connection", "Keep-Alive");
                conn.setRequestProperty("user-agent", "Mozilla/~~~~~~~~~~~~");
                conn.setDoInput(true);
                conn.setDoOutput(true);
                conn.setRequestMethod("POST");
                conn.setUseCaches(false);
                PrintWriter out= new PrintWriter(conn.getOutputStream());
                final String param="content="+content+"&"+"deadline="+deadline;
                out.print(param);
                int responseCode = conn.getResponseCode();
                String i=String.valueOf(responseCode);
                if (responseCode == HttpURLConnection.HTTP_OK) {
                    BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
                    String line;
                    while ((line = in.readLine()) != null) {
                        result += line;
                    }
                }
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
            return result;
        }
        @Override
        protected void onPostExecute(String result){
            if(result!=null){
                text.setText(result);
            }
            super.onPostExecute(result);
        }
    }.execute(url);
    }
}
```

WEBVIEW

- WebView是一个基于webkit引擎、展现web页面的控件。
- Android的Webview在低版本和高版本采用了不同的webkit版本内核，4.4后直接使用了Chrome。
- WebView控件功能强大，除了具有一般View的属性和设置外，还可以对url请求、页面加载、渲染、页面交互进行强大的处理。



WEBVIEW

- WebView的作用：
- 显示和渲染Web页面。
- 直接使用html文件（网络上或本地assets中）作布局。
- 可和JavaScript交互调用。

WEBVIEW

- 常用方法：
- WebView.loadUrl（“http://www.google.com/”）：加载一个网页。
- WebView.loadUrl（“file:///android_asset/test.html”）：加载apk包中的html页面 。
- WebView.loadUrl（“content://com.android.htmlfileprovider/sdcard/test.html”）：加载手机本地的html页面。
- WebView.loadData(String data, String mimeType, String encoding)：以标签形式加载 HTML 页面的一小段内容。

WEBVIEW

- WebView.onResume()：激活WebView为活跃状态，能正常执行网页的响应。
- WebView.onPause()：通过onPause动作通知内核暂停所有的动作，比如DOM的解析、plugin的执行、JavaScript执行
- WebView.pauseTimers()：暂停应用程序内所有Webview的layout，parsing，javascripttimer。降低CPU功耗。
- WebView.resumeTimers()：恢复pauseTimers状态。
- RootLayout.removeView(webView)：从父容器中移除webview。
- WebView.destroy()：销毁webview。

WEBVIEW

- WebView.canGoBack()：是否可以后退。
- WebView.goBack()：后退网页。
- WebView.canGoForward():是否可以前进。
- WebView.goForward()：前进网页。
- WebView.goBackOrForward(intsteps)：以当前的index为起始点前进或者后退到历史记录中指定的steps，如果steps为负数则为后退，正数则为前进。

WEBVIEW

- 在不做任何处理前提下，浏览网页时点击系统的“Back”键,整个Browser 会调用 finish()而结束自身。
- 如果需要在点击返回按钮后，实现网页回退而不是退出浏览器，则需要在当前Activity中处理并消费掉该 Back 事件：

```
public boolean onKeyDown(int keyCode, KeyEvent event){
    if ((keyCode == KEYCODE_BACK) && webView.canGoBack()) {
        webView.goBack();
        return true;
    }
    return super.onKeyDown(keyCode, event);
}
```

WEBVIEW

- Webview.clearCache(true)：清除网页访问留下的缓存，由于内核缓存是全局的因此这个方法不仅仅针对webview而是针对整个应用程序。
- Webview.clearHistory()：只会清除Webview访问历史记录里的所有记录，除了当前访问记录。
- Webview.clearFormData()：清除自动完成填充的表单数据，并不会清除WebView存储到本地的数据。

WEBVIEW

- 使用WebView必须在AndroidManifest.xml中添加访问网络权限：

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bishop.helloworld">
    <uses-permission android:name="android.permission.INTERNET"></uses-permission>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloWorld"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

WEBVIEW

- 在Activity的layout文件里添加webview控件：

```
<WebView
    android:id="@+id/webView"
    android:layout_width="368dp"
    android:layout_height="460dp"
    tools:layout_editor_absoluteX="8dp"
    tools:layout_editor_absoluteY="43dp" />
```

- 在Activity中实例化：
- WebView webView = new WebView(this);
- 绑定该控件：
- WebView webview = (WebView) findViewById(R.id.webView);

WEBVIEW

- WebSettings子类：
- WebSettings webSettings = webView.getSettings();
- webSettings.setJavaScriptEnabled(true); //设置支持JavaScript
- webSettings.setPluginsEnabled(true); //设置支持插件
- webSettings.setAllowFileAccess(true); //设置访问文件权限
- webSettings.setJavaScriptCanOpenWindowsAutomatically(true); //支持通过JavaScript打开新窗口
- webSettings.setLoadsImagesAutomatically(true); //支持自动加载图片
- webSettings.setDefaultTextEncodingName("utf-8");//设置编码格式

THE END.