GET和POST请求

HttpURLConnection登录之GET,POST

GET请求特有:

客户端发送数据以UTF-8的编码发送,服务器以ISO8859-1的编码进行接收解码,所以在服务器端要先以ISO8859-1方式编码,以再UTF-8解码,然后返回给客户端.

GET：

1、获取用户名和密码

String username = et_username.getText().toString().trim();

2、对用户名进行URL按UTF-8编码

参数一：需要编码的中文　　参数二：按什么字符集编码　　返回值：编码后的内容

username = URLEncoder.encode(username,"UTF-8");

3、创建访问服务器的路径

String path = "http://192.168.56.1:8080/day11_web/UserServlet?username="+username+"&password="+password;

4、创建URL对象

URL url = new URL(path);

5、获取与服务器连接的对象

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

6、设置提交数据的方式，用大写字母，设置一个超时时是，单位毫秒　　连接服务器（可省）

conn.setRequestMethod("GET")　　conn.setConnectTimeout(5000);　　conn.connect();

7、获取响应状态码

int code = conn.getResponseCode();

8、对获取的服务器响应码进行判断如果是200那么就通过 getInputStream() 方法获取服务器响应给客户端的内容

InputStream is = conn.getInputStream()

9、创建一个容器用来获取服务器响应给客户端的数据

ByteArrayOutputStream baos = new ByteArrayOutputStream(); 这个其实就是一个输入流将InputStream读取到的数据写入其中保存

10、通过循环将服务器响应的数据，放入容器中，即读写操作

byte[] buf = new byte[1024];　int len = 0;　while((len=is.read(buf))>0){　baos.write(buf,0,len);　baos.flush();

11、将容器中的数据转换为字符串或其它数据类型String

String　string =　new String(baos.toByteArray(),"UTF-8");

12、对数据做其它的操作，比如加入LIstview组件显示或者直接输入等操作

MainActivity.this.runOnUiThread(new Thread(){　在子线程中用主线程来提示用户的方法，不需要用handler这个助手处理类

13、关闭流

```
public void getGET() {
    new Thread() {
        public void run() {
            Message msg = new Message();
            HttpURLConnection con = null;
            ByteArrayOutputStream baos = null;
            InputStream in = null;
            // 获取用户名密码 String name =
            String name = usname.getText().toString().trim();
            String password = paword.getText().toString().trim();
            // 获取访问地址
            try {
                String username = URLEncoder.encode(name, "UTF-8");
                String path = "http://192.168.2.102:8080/Servlets_02/servlet/MyServlets?username="
                        + username + "&password=" + password;
                // 创建访问路径
                URL url = new URL(path);
                // 获取与服务器的连接对象
                con = (HttpURLConnection) url.openConnection();
                // 设置访问方式及响应时间
                con.setRequestMethod("GET");
                con.setConnectTimeout(5000);
                // 链接服务器
                con.connect();
                //获取响应状态码
                int code = con.getResponseCode();
                if (code == 200) {
                    in = con.getInputStream();
                    baos = new ByteArrayOutputStream();
                    int len;
                    byte[] brr = new byte[1024];
                    while ((len = in.read(brr)) != -1) {
                        baos.write(brr, 0, len);

                    String str = new String(baos.toByteArray());
                    msg.obj = str;
                    msg.what = 1;
                    handler.sendMessage(msg);
                } else {
                    msg.obj = "";
                    msg.what = 2;
                    handler.sendMessage(msg);
                }
            } catch (Exception e) {

            } finally {
                try {
                    in.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                try {
                    baos.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

    }.start();
```

POST：

1、获取访问服务要用的，用户名和密码

String username = et_username.getText().toString().trim();

2、对用户名进行URL按UTF-8编码

参数一：需要编码的中文　　参数二：按什么字符集编码　　返回值：编码后的内容
username = URLEncoder.encode(username,"UTF-8");
3、创建一个容器保存要传递给服务器的数据
String data = "username="+username+"&password="+password;
4、创建访问服务器的路径
String path = "http://192.168.56.1:8080/day11_web/UserServlet?
username="+username+"&password="+password;
5、创建URL对象
URL url = new URL(path);
6、获取与服务器连接的对象
HttpURLConnection conn =(HttpURLConnection) url.openConnection();
7、设置提交数据的方式，用大写字母，设置一个超时时是，单位毫秒　　连接服务器（可省）
conn.setRequestMethod("GET")　conn.setConnectTimeout(5000);　　conn.connect();
8、设置POST二个专用请求头
conn.setRequestProperty("content-type","application/x-www-form-urlencoded");
访问的类型及编码　这两个数据可通过浏览器中的网络监控的请求头去获取
conn.setRequestProperty("content-length",data.length()+"");　访问的提交数据的长度
9、获取输出流对象，通过输出流将要提交给服务器的数据传递出去
OutputStream os = conn.getOutputStream();　　os.write(data.getBytes("UTF-8"));
os.flush();
10、获取响应状态码
int code = conn.getResponseCode();
11、对获取的服务器响应码进行判断如果是200那么就通过 getInputStream() 方法获取服务器响应给客户端的内容
12、创建一个容器用来获取服务器响应给客户端的数据
ByteArrayOutputStream baos = new ByteArrayOutputStream(); 这个其实就是一个输入流将InputStream读取到的数据写入其中保存
13、通过循环将服务器响应的数据，放入容器中，即读写操作
byte[] buf = new byte[1024];　int len = 0;　while((len=is.read(buf))>0){　baos.write(buf,0,len);
baos.flush();
114、将容器中的数据转换为字符串或其它数据类型String
String　string =　new String(baos.toByteArray(),"UTF-8");
15、对数据做其它的操作，比如加入LIstview组件显示或者直接输入等操作
Message message = new Message(); 消息获取类对象
message.what = UPDATE_UI;　　获取消息的类型 用常量表示数值为整数
message.obj = content;　　获取消息的内容，最后在handler这个类中的handleMessage方法获取数据msg.obj.tostring()
handler.sendMessage(message);　　将消息传递给handler类来处理。
MainActivity.this.runOnUiThread(new Thread(){　在子线程中用主线程来提示用户的方法，不需要用handler这个助手处理类

16、关流

```
public void getPOST() {
    new Thread() {
        public void run() {
            Message mag = new Message();
            HttpURLConnection con = null;
            InputStream in = null;
            ByteArrayOutputStream baos = null;
            String username = usname.getText().toString().trim();
            String password = paword.getText().toString().trim();
            try {
                // 服务器要接收的数据
                username = URLEncoder.encode(username, "utf-8");
                String data = "username=" + username + "&password="
                        + password;
                String path = "http://192.168.2.102:8080/Servlets_02/servlet/MyServlets";
                URL url = new URL(path);
                con = (HttpURLConnection) url.openConnection();
                con.setRequestMethod("POST");
                con.setConnectTimeout(5000);
                //设置请求头
                con.setRequestProperty("Content-Type",
                        "application/x-www-form-urlencoded");
                con.setRequestProperty("Content-Length",
                        data.length()+"");//
                //获取输出流
                OutputStream stream = con.getOutputStream();
                //用输出流将data变量输出到服务器
                stream.write(data.getBytes());
                // 要不要关?
                stream.flush();

                int code = con.getResponseCode();
                if (code == 200) {
                    in = con.getInputStream();
                    baos = new ByteArrayOutputStream();
                    int len;
                    byte[] brr = new byte[1024];
                    while ((len = in.read(brr)) != -1) {
                        baos.write(brr, 0, len);
                    }
                    String str1 = new String(baos.toByteArray());
                    mag.obj = str1;
                    mag.what = 1;
                    handler.sendMessage(mag);
                }
            } catch (Exception e) {

            } finally {
                try {
                    con.disconnect();
                    in.close();
                    baos.close();
                } catch (Exception e2) {
                }
            }
        }
    }.start();
```

这个类是一个容器用不保存读取到的数据

HttpClient登录之GET,POST

是Apache Jakarta Common下的子项目，用来提供高效的、最新的、功能丰富的支持HTTP协议的客户端编程工具包，并且它支持HTTP协议最新的版本和协议。使用HttpClient时，无需导入项目的jar包或源码。

GET：

1、获取访问服务器要用的数据

String username = et_username.getText().toString().trim();

2、对用户名进行URL按UTF-8编码

username = URLEncoder.encode(username,"UTF-8");

3、创建访问服务器的路径

String path = "http://192.168.56.1:8080/day11_web/UserServlet?username="+username+"&password="+password;

4、获取HttpClient对象

HttpClient client = new DefaultHttpClient();

5、创建用于GET请求的对象

HttpGet get = new HttpGet(path);

6、执行对服务器的访问操作

HttpResponse response = client.execute(get);

7、通过获取状态行来，获取访问服务器的响应码

response.getStatusLine().getStatusCode()

8、判断如果响应码为200执行获取响应数据操作，获取服务器响应的数据，类型，字符的长度等等

HttpEntity entity = response.getEntity();　　响应请求数据

9、直接将HttpEntity对象响应给客户端的数据转成字符串

String content = EntityUtils.toString(entity);

10、对数据做其它的操作，比如加入LIstview组件显示或者直接输入等操作

　MainActivity.this.runOnUiThread(new Thread(){　在子线程中用主线程来提示用户的方法，不需要用handler这个助手处理类

```java
/**
 * 以POST方式发送请求
 */
private void sendWithPOST() {
    new Thread(){
        public void run() {
            try {
                String username = et_username.getText().toString().trim();//赵君
                String password = et_password.getText().toString().trim();//123456
                //不要手工编码，不用在URL后面加?
                String path = "http://192.168.56.1:8080/day11_web/UserServlet";
                HttpClient client = new DefaultHttpClient();
                HttpPost post = new HttpPost(path);
                //将用户名绑定到指定的变量中，username变量是明文，暂时无需编码
                BasicNameValuePair pair1 = new BasicNameValuePair("username",username);
                BasicNameValuePair pair2 = new BasicNameValuePair("password",password);
                //创建一个集合，用于装所有的BasicNameValuePair对象
                List<NameValuePair> list = new ArrayList<NameValuePair>();
                list.add(pair1);
                list.add(pair2);
                //统一对list集合中的所有数据编码，按UTF-8
                UrlEncodedFormEntity formEntity = new UrlEncodedFormEntity(list,"UTF-8");
                //将FormEntity设置到POST请求中
                post.setEntity(formEntity);
                HttpResponse response = client.execute(post);
                if(response.getStatusLine().getStatusCode()==200){
                    //FormEntity是请求的数据
                    //HttpEntity是响应的数据
                    HttpEntity entity = response.getEntity();
                    //参数为响应的数据
                    final String content = EntityUtils.toString(entity);
                    MainActivity.this.runOnUiThread(new Thread(){
                        @Override
```

```java
/**
 * 以GET方式发送请求
 */
private void sendWithGET() {
    new Thread(){
        public void run() {
            try {
                String username = et_username.getText().toString().trim();
                String password = et_password.getText().toString().trim();
                username = URLEncoder.encode(username,"UTF-8");
                String path = "http://192.168.56.1:8080/day11_web/UserServlet?username="+username+"&password="+password;

                //选中对象，ctrl+T，查询该对象的结构
                HttpClient client = new DefaultHttpClient();
                //创建用于GET请求的对象
                HttpGet get = new HttpGet(path);
                //执行访问操作
                HttpResponse response = client.execute(get);
                //获取响应码
                if(response.getStatusLine().getStatusCode()==200){
                    //获取服务器响应的内容，类型，字符的长度等等
                    HttpEntity entity = response.getEntity();
                    //直接将HttpEntity对象中的内容转成字符串
                    final String content = EntityUtils.toString(entity);
                    MainActivity.this.runOnUiThread(new Thread(){
                        @Override
                        public void run() {
                            Toast.makeText(MainActivity.this,content,0).show();
```

```
public class MainActivity extends Activity implements OnClickListener{
    private EditText et_username,et_password;
    private Button bt_get,bt_post;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        et_username = (EditText) this.findViewById(R.id.et_username);
        et_password = (EditText) this.findViewById(R.id.et_password);
        bt_get = (Button) this.findViewById(R.id.bt_get);
        bt_post = (Button) this.findViewById(R.id.bt_post);
        //为二个Button控件设置单击事件
        bt_get.setOnClickListener(this);
        bt_post.setOnClickListener(this);
    }
    /**
     * 单击后的回调
     */
    @Override
    public void onClick(View v) {
        //区分:
        switch(v.getId()){
            //GET请求
            case R.id.bt_get:
                sendWithGET();
                break;
            //POST请求
            case R.id.bt_post:
                sendWithPOST();
                break;
```

实现接口的多按钮点击事件

R.id.bt_get==v.getId()
这个方法可以判断是哪一个
按钮的点击事件

POST：

1、获取访问服务器要用的数据

String username = et_username.getText().toString().trim();

2、创建访问服务器的路径

String path = "http://192.168.56.1:8080/day11_web/UserServlet";

3、获取HttpClient对象

HttpClient client = new DefaultHttpClient();

4、创建用于POST请求的对象

HttpPost post = new HttpPost(path);

5、将访问服务器的数据绑定到指定的变量中

BasicNameValuePair pair1 = new BasicNameValuePair("username",username);

6、创建一个集合，用于装所有的BasicNameValuePair对象

List<BasicNameValuePair> list = new ArrayList<BasicNameValuePair>();

list.add(pair1);

7、统一对list集合中的所有数据编码，按UTF-8

参数一：list集合　　参数二：指定的编码UTF-8

UrlEncodedFormEntity formEntity = new UrlEncodedFormEntity(list,"UTF-8");

8、将转码后的向服务器请求数据对象设置到POST请求中

post.setEntity(formEntity);
9、执行对服务器的访问操作
HttpResponse response = client.execute(post);
10、获取访问服务器的响应码，如果响应码是200那么就开始获取服务器响应数据
response.getStatusLine().getStatusCode()
11、获取服务器响应的数据，类型，字符的长度等等
HttpEntity entity = response.getEntity();
12、直接将HttpEntity对象响应给客户端的数据转成字符串
String content = EntityUtils.toString(entity);
13、对数据做其它的操作，比如加入LIstview组件显示或者直接输入等操作
MainActivity.this.runOnUiThread(new Thread(){ 在子线程中用主线程来提示用户的方法，不需要用handler这个助手处理类

AsyncHttpClient异步框架GET,POST请求

GET：
AsyncHttpClient是一个开源的网络请求框架，它是专门针对Android在Apache的HttpClient基础上构建的异步的callback based http client。所有的请求全在UI线程之外，而callback发生在创建它的UI线程中，应用了Android的Handler发送消息机制。使用AsyncHttpClient时，要导入对应的jar包或源码。
1_对同步HttpClient的封装，而HttpClient又对同步HttpURLConnection封装
2_采用异步方式
3_请求均在UI线程或主线程之外，是隐式的
4_线程池，重用里面的线程，目的，节省资源，加快速度
5_不需在URL后面接装提交数据
6_不需手工编码
1、获取访问服务器的数据
String username = et_username.getText().toString().trim();
2、判断访问服务的数据是否为空
3、创建异步AsyncHttpClient对象
AsyncHttpClient client = new AsyncHttpClient();
4、获取访问服务器的地址 书写PATH，不需要将数据拼接到PATH后面，不需要手工编码
String path = "http://192.168.56.1:8080/day11_web/UserServlet";
5、创建封装提交数据的对象
RequestParams params = new RequestParams();
6、绑定具体的数据
params.put("username",username);

7、提交访问服务器的数据

参数一：提交到服务器的路径

参数二：提交的数据

参数三：响应处理对象,即如果子线程有结果响应的话，去调用AsyncHttpResponseHandler对象

client.get(path,params,new MyResponseHandler());

8、响应处理对象，可使用有名内部类继承extends AsyncHttpResponseHandler类

重写onSuccess(String content)方法和onFailure(Throwable error, String content)方法

onSuccess(String content)方法代表服务器响应成功

onFailure(Throwable error, String content)方法代表响应失败

方法内写成功或失败后要做的数据，提示处理

```
 * GET请求
 * 1_对同步HttpClient的封装，而HttpClient又对同步HttpURLConnection封装
 * 2_采用异步方式
 * 3_请求均在UI线程或主线程之外，是隐式的
 * 4_线程池，重用里面的线程，目的，节省资源，加快速度
 * 5_不需在URL后面接装提交数据
 * 6_不需手工编码
 */
public void click1(View view) {

    String password = et_password.getText().toString().trim();
    if(!TextUtils.isEmpty(username) && !TextUtils.isEmpty(password)){
        //创建异步HttpClient对象
        AsyncHttpClient client = new AsyncHttpClient();
        //书写PATH，不需要将数据拼接到PATH后面，不需要手工编码
        String path = "http://192.168.56.1:8080/day11_web/UserServlet";
        //创建封装提交数据的对象
        RequestParams params = new RequestParams();
        //绑定具体的数据
        params.put("username",username);
        params.put("password",password);
        //在隐式子线程中访问网络
        //参数一：提交到服务器的路径
        //参数二：提交的数据
        //参数三：响应处理对象,即如果子线程有结果响应的话，去调用AsyncHttpResponseHandler对象
        client.get(path,params,new MyResponseHandler());
    }else{
        Toast.makeText(this,"用户名和密码不能为空",0).show();
    }
```

```
 * 有名内部类
 * 响应处理对象
 * 这二个方法在UI线程或主线程中执行，可以更新UI
 */
private class MyResponseHandler extends AsyncHttpResponseHandler{
    /**
     * 成功，即code==200
     * @param content 服务器响应的内容
     */
    @Override
    public void onSuccess(String content) {//登录成功GET
        //Context上下文
        //参数一：Context，MainActivity.this，即MainActivity的对象
        Toast.makeText(MainActivity.this,content,0).show();
    }
    /**
     * 失败,即code!=200
     */
    @Override
    public void onFailure(Throwable error, String content) {
        Toast.makeText(MainActivity.this,error.getMessage(),0).show();
    }
}
```

POST：

1、获取访问服务器的数据

String username = et_username.getText().toString().trim();

2、判断访问服务的数据是否为空

3、创建异步AsyncHttpClient对象

AsyncHttpClient client = new AsyncHttpClient();

4、获取访问服务器的地址　书写PATH，不需要将数据拼接到PATH后面，不需要手工编码

String path = "http://192.168.56.1:8080/day11_web/UserServlet";

5、创建封装提交数据的对象

RequestParams params = new RequestParams();

6、绑定具体的数据

params.put("username",username);

7、提交访问服务器的数据

参数一：提交到服务器的路径

参数二：提交的数据

参数三：响应处理对象,即如果子线程有结果响应的话，去调用AsyncHttpResponseHandler对象

client.post(path,params,new MyResponseHandler());

8、响应处理对象，可使用有名内部类继承extends AsyncHttpResponseHandler类

重写onSuccess(String content)方法和onFailure(Throwable error, String content)方法

onSuccess(String content)方法代表服务器响应成功

onFailure(Throwable error, String content)方法代表响应失败

方法内写成功或失败后要做的数据，提示处理

```
/**
 * POST请求
 */
public void click2(View view) {
    String username = et_username.getText().toString().trim();
    String password = et_password.getText().toString().trim();
    if(TextUtils.isEmpty(username) || TextUtils.isEmpty(password)){
        Toast.makeText(this,"用户名和密码不能为空",0).show();
        return;
    }
    AsyncHttpClient client = new AsyncHttpClient();
    String path = "http://192.168.56.1:8080/day11_web/UserServlet";
    RequestParams params = new RequestParams();
    params.put("username", username);
    params.put("password", password);
    client.post(path,params,new MyResponseHandler());
}
```

Android版本多线程下载

xUtils开源项目简介及框架下载，具有上传功能

xUtils最初源于Afinal框架，进行了大量重构，使得xUtils支持大文件上传，，拥有更加灵活的ORM，更多的事件注解支持且不受混淆影响。xUitls最低兼容android 2.2 (api level 8)

xUtils主要有以下四大模块：DbUtils模块、ViewUtils模块、HttpUtils模块、BitmapUtils模块。xUtils可以在github上搜索进行下载：

1、导入条三方工具包xUtils

2、创建HttpUtils对象 HttpUtils utils = new HttpUtils();

3、设置下载路径：String path = "http://192.168.143.1:8080/a.jpg";

4、开始下载：utils.download(path , "/mnt/sdcard/a.jpg" , true , new MyHandler());

参数一：下载的路径

参数二：保存位置

参数三：断点续传，true表示支持断点续传;false不表示支持断点续传

参数四：下载响应处理类

5、定义下载响应处理类：任意类继承 RequestCallBack<File>

private class MyHandler extends RequestCallBack<File>

6、重写

onFailure(HttpException error, String msg)方法 表示响应失败状态码!=200方法处理

onLoading(long total, long current, boolean isUploading)方法 表示响应成功，如果需要处理进度条，可以重写这个方法

参数一：total表示文件的总长度

参数二：current表示当前这个线程已经下载的字节

参数三：是否上传，如果是下载的话，用false

```
//创建HttpUtils对象
HttpUtils utils = new HttpUtils();
//下载路径
String path = "http://192.168.143.1:8080/a.jpg";
//下载
//参数一：下载路径
//参数二：保存在手机中的位置，/mnt/sdcard只限于模拟器
//参数三：true表示支持断点续传;false不表示支持断点续传
//参数四：响应处理类
//注意：下面代码是由隐式子线程运行
utils.download(path , "/mnt/sdcard/a.jpg" , true , new MyHandler());
```

```
private class MyHandler extends RequestCallBack<File>{
    /**
     * 响应状态码=200
     */
    @Override
    public void onSuccess(ResponseInfo<File> responseInfo) {
        Toast.makeText(MainActivity.this,"下载文件成功",0).show();
    }
    /**
     * 响应状态码!=200
     */
    @Override
    public void onFailure(HttpException error, String msg) {
        Toast.makeText(MainActivity.this,"下载文件失败",0).show();
    }
    /**
     * 如果需要处理进度条，可以重写这个方法
     * 参数一：total表示文件的总长度，如1471左右
     * 参数二：current表示当前这个线程已经下载的字节
     * 参数三：是否上传，如果是下载的话，用false
     */
    @Override
    public void onLoading(long total, long current, boolean isUploading) {
        //操作进度条
        //进度条的总长度:
        pb.setMax((int)total);
        //进度条的刻度
        pb.setProgress((int)current);
```