Atitit http参数 模式：request-line方式与request-body方式。

：request-line方式与request-body方式。

request-line

request-line方式是指在请求行上通过URI直接提供参数。

equest-line方式是指在请求行上通过URI直接提供参数。

（1）

我们可以在生成request对象时提供带参数的URI，如：

HttpUriRequest request = new HttpGet(

        "http://localhost/index.html?param1=value1&param2=value2");

（2）

另外，HttpClient程序包为我们提供了URIUtils工具类，可以通过它生成带参数的URI，如：

URI uri = URIUtils.createURI("http", "localhost", -1, "/index.html",

    "param1=value1&param2=value2", null);

HttpUriRequest request = new HttpGet(uri);

System.out.println(request.getURI());

与request-line方式不同，request-body方式是在request-body中提供参数，此方式只能用于POST请求。在 HttpClient程序包中有两个类可以完成此项工作，它们分别是UrlEncodedFormEntity类与MultipartEntity类。这两个类均实现了HttpEntity接口。

（1）

使用最多的是UrlEncodedFormEntity类。通过该类创建的对象可以模拟传统的HTML表单传送POST请求中的参数。如下面的表单：

<form action="http://localhost/index.html" method="POST">

    <input type="text" name="param1" value="中国"/>

    <input type="text" name="param2" value="value2"/>

    <inupt type="submit" value="submit"/>

</form>

我们可以用下面的代码实现：

List formParams = new ArrayList();

formParams.add(new BasicNameValuePair("param1", "中国"));

formParams.add(new BasicNameValuePair("param2", "value2"));

HttpEntity entity = new UrlEncodedFormEntity(formParams, "UTF-8");

HttpPost request = new HttpPost(“http://localhost/index.html”);

request.setEntity(entity);

当然，如果想查看HTTP数据格式，可以通过HttpEntity对象的各种方法取得。如：

List formParams = new ArrayList();

formParams.add(new BasicNameValuePair("param1", "中国"));

formParams.add(new BasicNameValuePair("param2", "value2"));

UrlEncodedFormEntity entity = new UrlEncodedFormEntity(formParams, "UTF-8");

System.out.println(entity.getContentType());

System.out.println(entity.getContentLength());

System.out.println(EntityUtils.getContentCharSet(entity));

System.out.println(EntityUtils.toString(entity));

上例的打印结果如下：

    Content-Type: application/x-www-form-urlencoded; charset=UTF-8

    39

    UTF-8

    param1=%E4%B8%AD%E5%9B%BD&param2=value2

（2）

除了传统的application/x-www-form-urlencoded表单，我们另一个经常用到的是上传文件用的表单，这种表单的类型为 multipart/form-data。在HttpClient程序扩展包（HttpMime）中专门有一个类与之对应，那就是 MultipartEntity类。此类同样实现了HttpEntity接口。如下面的表单：

<form action="http://localhost/index.html" method="POST"

        enctype="multipart/form-data">

    <input type="text" name="param1" value="中国"/>

    <input type="text" name="param2" value="value2"/>

    <input type="file" name="param3"/>

    <inupt type="submit" value="submit"/>

</form>

我们可以用下面的代码实现：

MultipartEntity entity = new MultipartEntity();

entity.addPart("param1", new StringBody("中国", Charset.forName("UTF-8")));

entity.addPart("param2", new StringBody("value2", Charset.forName("UTF-8")));

entity.addPart("param3", new FileBody(new File("C:\\1.txt")));

HttpPost request = new HttpPost(“http://localhost/index.html”);

request.setEntity(entity);

HTTP响应

HttpClient程序包对于HTTP响应的处理较之HTTP请求来说是简单多了，其过程同样使用了HttpEntity接口。我们可以从 HttpEntity对象中取出数据流（InputStream），该数据流就是服务器返回的响应数据。需要注意的是，HttpClient程序包不负责解析数据流中的内容。如：

HttpUriRequest request = ...;

HttpResponse response = httpClient.execute(request);

// 从response中取出HttpEntity对象

HttpEntity entity = response.getEntity();

// 查看entity的各种指标

System.out.println(entity.getContentType());

System.out.println(entity.getContentLength());

System.out.println(EntityUtils.getContentCharSet(entity));

// 取出服务器返回的数据流

InputStream stream = entity.getContent();

// 以任意方式操作数据流stream

// 调用方式 略