说明：

本解读对象为：《HttpClient Tutorial》，基于 HttpClient 4.5.7

更新时间：

2019 年 6 月

第一节：添加 maven 依赖

使用 HttpClient4.5.7 需要的 Maven 依赖，如下所示：

```
<dependency>
   <groupId>org.apache.httpcomponents</groupId>
   <artifactId>httpclient</artifactId>
   <version>4.5.7</version>
</dependency>
```

第二节：了解 HttpClient 的基本用法

```
CloseableHttpClient httpclient = HttpClients.createDefault();
HttpGet httpget = new HttpGet("http://localhost/");
CloseableHttpResponse response = httpclient.execute(httpget);
try {
    <...>
} finally {
    response.close();
}
```

第三节：构建 HTTP 请求

3.1 方法一

```
HttpGet httpget = new HttpGet(
     "http://www.google.com/search?hl=en&q=httpclient&btnG=Google+Search&aq=f&oq=");
```

3.2 方法二

```
URI uri = new URIBuilder()
        .setScheme("http")
        .setHost("www.google.com")
        .setPath("/search")
        .setParameter("q", "httpclient")
        .setParameter("btnG", "Google Search")
        .setParameter("aq", "f")
        .setParameter("oq", "")
        .build();
HttpGet httpget = new HttpGet(uri);
```

第四节：构建 HTTP 响应

```
HttpResponse response = new BasicHttpResponse(HttpVersion.HTTP_1_1,
HttpStatus.SC_OK, "OK");

System.out.println(response.getProtocolVersion());
System.out.println(response.getStatusLine().getStatusCode());
System.out.println(response.getStatusLine().getReasonPhrase());
System.out.println(response.getStatusLine().toString());
```

第五节：添加消息头

```
HttpResponse response = new BasicHttpResponse(HttpVersion.HTTP_1_1,
    HttpStatus.SC_OK, "OK");
response.addHeader("Set-Cookie",
    "c1=a; path=/; domain=localhost");
response.addHeader("Set-Cookie",
    "c2=b; path=\"/\", c3=c; domain=\"localhost\"");
Header h1 = response.getFirstHeader("Set-Cookie");
System.out.println(h1);
Header h2 = response.getLastHeader("Set-Cookie");
System.out.println(h2);
Header[] hs = response.getHeaders("Set-Cookie");
System.out.println(hs.length);
```

第六节：生成 HTTP 实体

6.1 string 类型的实体

```
StringEntity myEntity = new StringEntity("important message",
   ContentType.create("text/plain", "UTF-8"));

System.out.println(myEntity.getContentType());
System.out.println(myEntity.getContentLength());
System.out.println(EntityUtils.toString(myEntity));
System.out.println(EntityUtils.toByteArray(myEntity).length);
```

6.2 file 类型的实体

```
File file = new File("somefile.txt");
FileEntity entity = new FileEntity(file,
    ContentType.create("text/plain", "UTF-8"));

HttpPost httppost = new HttpPost("http://localhost/action.do");
httppost.setEntity(entity);
```

6.3 表单实体

```
List<NameValuePair> formparams = new ArrayList<NameValuePair>();
formparams.add(new BasicNameValuePair("param1", "value1"));
formparams.add(new BasicNameValuePair("param2", "value2"));
UrlEncodedFormEntity entity = new UrlEncodedFormEntity(formparams, Consts.UTF_8);
HttpPost httppost = new HttpPost("http://localhost/handler.do");
httppost.setEntity(entity);
```

第七节：HTTP 代理

7.1 方式一

```
HttpHost proxy = new HttpHost("someproxy", 8080);
DefaultProxyRoutePlanner routePlanner = new DefaultProxyRoutePlanner(proxy);
CloseableHttpClient httpclient = HttpClients.custom()
        .setRoutePlanner(routePlanner)
        .build();
```

7.2 方式二

```
SystemDefaultRoutePlanner routePlanner = new SystemDefaultRoutePlanner(
        ProxySelector.getDefault());
CloseableHttpClient httpclient = HttpClients.custom()
        .setRoutePlanner(routePlanner)
        .build();
```

7.3 方式三

```
HttpRoutePlanner routePlanner = new HttpRoutePlanner() {

    public HttpRoute determineRoute(
            HttpHost target,
            HttpRequest request,
            HttpContext context) throws HttpException {
        return new HttpRoute(target, null,  new HttpHost("someproxy", 8080),
                "https".equalsIgnoreCase(target.getSchemeName()));
    }

};
CloseableHttpClient httpclient = HttpClients.custom()
        .setRoutePlanner(routePlanner)
        .build();
    }
}
```