Apache  HttpAsyncClient

**1. Обзор**

В этом руководстве мы проиллюстрируем наиболее распространенные случаи использования Apache *HttpAsyncClient*- от базового использования до того, как **настроить прокси**, как использовать **SSL-сертификат**и, наконец, - как **аутентифицироваться**с помощью асинхронного клиента. ,

**2. Простой пример**

Сначала - давайте посмотрим, как использовать *HttpAsyncClient*в простом примере - отправим запрос GET

@Test

**public** **void** **whenUseHttpAsyncClient\_\_thenCorrect**() **throws** Exception {

CloseableHttpAsyncClient client = HttpAsyncClients.createDefault();

client.start();

HttpGet request = **new** HttpGet("http://www.google.com");

Future<HttpResponse> future = client.execute(request, **null**);

HttpResponse response = future.get();

assertThat(response.getStatusLine().getStatusCode(), equalTo(200));

client.close();

}

Обратите внимание, как **нам нужно *стартовать*асинхронный клиент перед его использованием**; без этого мы получили бы следующее исключение:

java.lang.IllegalStateException: Request cannot be executed; I/O reactor status: INACTIVE

at o.a.h.u.Asserts.check(Asserts.java:46)

at o.a.h.i.n.c.CloseableHttpAsyncClientBase.

ensureRunning(CloseableHttpAsyncClientBase.java:90)

**3. Многопоточность с *HttpAsyncClient***

Теперь давайте посмотрим, как использовать *HttpAsyncClient*для одновременного выполнения нескольких запросов.

В следующем примере - мы отправляем три запроса GET на три разных хоста, используя *HttpAsyncClient*и *PoolingNHttpClientConnectionManager*:

@Test

**public** **void** **whenUseMultipleHttpAsyncClient\_\_thenCorrect**() **throws** Exception {

ConnectingIOReactor ioReactor = **new** DefaultConnectingIOReactor();

PoolingNHttpClientConnectionManager cm =

**new** PoolingNHttpClientConnectionManager(ioReactor);

CloseableHttpAsyncClient client =

HttpAsyncClients.custom().setConnectionManager(cm).build();

client.start();

String[]toGet = {

"http://www.google.com/",

"http://www.apache.org/",

"http://www.bing.com/"

};

GetThread[]threads = **new** GetThread[toGet.length];

**for** (**int** i = 0; i < threads.length; i++) {

HttpGet request = **new** HttpGet(toGet[i]);

threads[i]= **new** GetThread(client, request);

}

**for** (GetThread thread : threads) {

thread.start();

}

**for** (GetThread thread : threads) {

thread.join();

}

}

Вот наша реализация *GetThread*для обработки ответа:

**static** **class** **GetThread** **extends** **Thread** {

**private** CloseableHttpAsyncClient client;

**private** HttpContext context;

**private** HttpGet request;

**public** **GetThread**(CloseableHttpAsyncClient client,HttpGet req){

**this**.client = client;

context = HttpClientContext.create();

**this**.request = req;

}

@Override

**public** **void** **run**() {

**try** {

Future<HttpResponse> future = client.execute(request, context, **null**);

HttpResponse response = future.get();

assertThat(response.getStatusLine().getStatusCode(), equalTo(200));

} **catch** (Exception ex) {

System.out.println(ex.getLocalizedMessage());

}

}

}

**4. Прокси с *HttpAsyncClient***

Далее - давайте посмотрим, как настроить и использовать **proxy**с *HttpAsyncClient*.

В следующем примере - мы отправляем запрос HTTP **GET через прокси**:

@Test

**public** **void** **whenUseProxyWithHttpClient\_\_thenCorrect**() **throws** Exception {

CloseableHttpAsyncClient client = HttpAsyncClients.createDefault();

client.start();

HttpHost proxy = **new** HttpHost("74.50.126.248", 3127);

RequestConfig config = RequestConfig.custom().setProxy(proxy).build();

HttpGet request = **new** HttpGet("https://issues.apache.org/");

request.setConfig(config);

Future<HttpResponse> future = client.execute(request, **null**);

HttpResponse response = future.get();

assertThat(response.getStatusLine().getStatusCode(), equalTo(200));

client.close();

}

**5. SSL-сертификат с *HttpAsyncClient***

Теперь давайте посмотрим, как использовать **SSL-сертификат**с *HttpAsyncClient*.

В следующем примере - мы настраиваем *HttpAsyncClient*на **принятие всех сертификатов**:

@Test

**public** **void** **whenUseSSLWithHttpAsyncClient\_\_thenCorrect**() **throws** Exception {

TrustStrategy acceptingTrustStrategy = **new** TrustStrategy() {

**public** **boolean** **isTrusted**(X509Certificate[]certificate, String authType) {

**return** **true**;

}

};

SSLContext sslContext = SSLContexts.custom()

.loadTrustMaterial(**null**, acceptingTrustStrategy).build();

CloseableHttpAsyncClient client = HttpAsyncClients.custom()

.setSSLHostnameVerifier(SSLConnectionSocketFactory.ALLOW\_\_ALL\_\_HOSTNAME\_\_VERIFIER)

.setSSLContext(sslContext).build();

client.start();

HttpGet request = **new** HttpGet("https://mms.nw.ru/");

Future<HttpResponse> future = client.execute(request, **null**);

HttpResponse response = future.get();

assertThat(response.getStatusLine().getStatusCode(), equalTo(200));

client.close();

}

**6. Печенье с *HttpAsyncClient***

Далее - давайте посмотрим, как использовать куки с *HttpAsyncClient*.

В следующем примере - мы **устанавливаем значение cookie перед отправкой запроса**:

@Test

**public** **void** **whenUseCookiesWithHttpAsyncClient\_\_thenCorrect**() **throws** Exception {

BasicCookieStore cookieStore = **new** BasicCookieStore();

BasicClientCookie cookie = **new** BasicClientCookie("JSESSIONID", "1234");

cookie.setDomain(".github.com");

cookie.setPath("/");

cookieStore.addCookie(cookie);

CloseableHttpAsyncClient client = HttpAsyncClients.custom().build();

client.start();

HttpGet request = **new** HttpGet("http://www.github.com");

HttpContext localContext = **new** BasicHttpContext();

localContext.setAttribute(HttpClientContext.COOKIE\_\_STORE, cookieStore);

Future<HttpResponse> future = client.execute(request, localContext, **null**);

HttpResponse response = future.get();

assertThat(response.getStatusLine().getStatusCode(), equalTo(200));

client.close();

}

**7. Аутентификация с помощью HttpAsyncClient**

Далее - давайте посмотрим, как использовать аутентификацию с *HttpAsyncClient*.

В следующем примере - мы используем *CredentialsProvider*для доступа к хосту через обычную аутентификацию:

@Test

**public** **void** **whenUseAuthenticationWithHttpAsyncClient\_\_thenCorrect**() **throws** Exception {

CredentialsProvider provider = **new** BasicCredentialsProvider();

UsernamePasswordCredentials creds = **new** UsernamePasswordCredentials("user", "pass");

provider.setCredentials(AuthScope.ANY, creds);

CloseableHttpAsyncClient client =

HttpAsyncClients.custom().setDefaultCredentialsProvider(provider).build();

client.start();

HttpGet request = **new** HttpGet("http://localhost:8080");

Future<HttpResponse> future = client.execute(request, **null**);

HttpResponse response = future.get();

assertThat(response.getStatusLine().getStatusCode(), equalTo(200));

client.close();

}

**8. Заключение**

В этой статье мы проиллюстрировали различные варианты использования асинхронного клиента Apache Http.

Реализация всех этих примеров и фрагментов кода **может быть найдена в [my[my project github](https://github.com/eugenp/tutorials/tree/master/httpclient" \l "readme)**- это проект на основе Eclipse, поэтому он легко импортировать и запускать как есть