链滴

JDK11 新特性

作者：2457081614
原文链接：https://ld246.com/article/1577237752536
来源网站：链滴

1、新增 HttpClient 客户端

简介

这个功能在 JDK9 中引入并在 JDK10 中得到了更新，最终在 JDK11 中发布，支持 HTTP1.1 和 HTTP/2。

常用工具类和接口：

- HttpClient.Builder HttpClient 构建 工具类
- HttpRequest.Builder HttpRequest 构建 工具类
- HttpRequest.BodyPublisher 将 Java 对象转换为可发送的 HTTP request body 字节流, 如 form 单提交
- HttpResponse.BodyHandler 处理接收到的 Response Body

Get 请求

```
package jdk11;

import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.time.Duration;

/**
 * @author by xw
 * @Description TODO
 */
```

```
public class HttpClientTest {


    private static final URI uri = URI.create("http://www.baidu.com");


    public static void main(String[] args) {
        testGet();
    }

    //GET请求
    private static void testGet() {
        //var httpClient = HttpClient.newHttpClient();
        //设置建立连接超时 connect timeout
        var httpClient =
                HttpClient.newBuilder().connectTimeout(Duration.ofMillis(5000)).build();
        //设置读取数据超时 read timeout
        var request =
                HttpRequest.newBuilder().timeout(Duration.ofMillis(3000))
                        .header("key1", "v1")
                        .header("key2", "v2")
                        .uri(uri).build();
        System.out.println(request);
        try {
            var response = httpClient.send(request,
                    HttpResponse.BodyHandlers.ofString());
            System.out.println(response.body());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

POST 请求

```
private static void testPost() {
        var httpClient = HttpClient.newHttpClient();
        var request = HttpRequest.newBuilder()
                .uri(uri)
                .header("Content-Type", "application/x-www-formurlencoded")
                .POST(HttpRequest.BodyPublishers.ofString("account=123456&pwd=1234567890"))
                .build();
        try {
            var response = httpClient.send(request,
                    HttpResponse.BodyHandlers.ofString());
            System.out.println(response.body());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

异步 Get 请求

```
/**
 * 异步请求
 */
private static void testAsynGet() {
    var httpClient = HttpClient.newBuilder().build();
    var request =
            HttpRequest.newBuilder().timeout(Duration.ofMillis(3000))
                    .header("key1", "v1")
                    .header("key2", "v2")
                    .uri(uri).build();
    try {
        CompletableFuture<String> result = httpClient.sendAsync(request,
                HttpResponse.BodyHandlers.ofString())
                .thenApply(HttpResponse::body);
        System.out.println(result.get());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

HTTP2 请求

Http2 百度百科：

HTTP2 协议的强制要求 https，如果 目标 URI 是 HTTP 的，则 无法使 用 HTTP 2 协议。

```
private static void testHttp2() {
    var httpClient = HttpClient.newBuilder()
            .connectTimeout(Duration.ofMillis(3000))
            //指定版本
            .version(HttpClient.Version.HTTP_2)
            .build();
    var request = HttpRequest.newBuilder()
            .timeout(Duration.ofMillis(3000))
            .header("key1", "v1")
            .header("key2", "v2")
            .uri(uri1)
            .build();
    try {
        var response = httpClient.send(request,
                HttpResponse.BodyHandlers.ofString());
        System.out.println(response.body());
        System.out.println(response.version());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

2、javac 和 Java 命令优化

我们在 jdk11 以前运行 Java 程序，命令如下

```
#编译
javac xxx.java
#运行
```

```
java xxx
```

jdk11 以后运行 Java 程序，命令如下

```
# 本地不会产生class文件
java xxx.java
```

3、ZGC(可伸缩的低延迟垃圾回收器)-实验阶段

ZGC，是一个可伸缩的低延迟垃圾收集器。 它旨在实现以下目标:

- 暂停时间不要超过 10 毫秒
- 暂停时间不会随堆大小的增加而增加，能够在很大的堆内存中进行高效垃圾回收。

Zgc 的核心是一个并发垃圾收集器，这意味着所有繁重的提升工作(标记、压缩、引用处理、字符串清理等)都是在 Java 线程继续执行时完成的。 这大大限制了垃圾收集对应用程序响应时间的影响。

4、Epsilon GC

Epsilon GC 是一种新的实验性无运行垃圾收集器。 Epsilon GC 只处理内存分配，不实现任何内存回机制。 它对于性能测试非常有用，可以对比其他 gc 的成本及效率。 它可以用来在测试中方便地确定存占用和内存压力。

5、移除方法

方法 Thread.destroy ()和 Thread.stop (Throwable)已被删除。

参考文档:

官网文档

项目代码地址