链滴

Java11 新特性及代码示例（转）

作者：michael

原文链接：https://ld246.com/article/1669972018267

来源网站：链滴

本文主要内容

- HTTP客户端API
- 无需编译即可启动单文件程序
- 字符串API更新
- Collection.toArray
- Files.readString() 和 Files.writeString()
- Optional.isEmpty()

Java 11（2018 年 9 月发布）包含许多重要且有用的更新。让我们看看它为开发人员和架构师带来的功能和改进。

1. HTTP客户端API

Java 使用HttpURLConnection进行HTTP通信已经很长一段时间了。但随着时间的推移，要求变得来越复杂，应用程序的要求也越来越高。在 Java 11 之前，开发人员不得不求助于功能丰富的库，如Apache HttpComponents或OkHttp等。

我们看到Java 9发布包含一个HttpClient实现作为实验性功能。它随着时间的推移而发展，现在是 Java 11 的最终功能。现在 Java 应用程序可以进行 HTTP 通信，而无需任何外部依赖。

1.1 如何使用HttpClient

java.net.http模块和典型 HTTP 交互如下所示：

- 创建 **HttpClient**的实例并根据需要进行配置。
- 创建一个 **HttpRequest**实例并填充信息。
- 将请求传递给客户端，执行请求并检索 **HttpResponse**的实例。
- 处理包含在 HttpResponse中的信息.

HTTP API 可以处理同步和异步通信。让我们看一个简单的例子。

1.2 同步请求示例

请注意 http 客户端 API 如何使用**构建器模式**来创建复杂对象。

```
package cn.dayangshuo.http;

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

/**
 * @author DAYANG
 */
```

```
public class HttpClientTest {
    public static void main(String[] args) {
        HttpClient httpClient = HttpClient.newBuilder()
                .connectTimeout(Duration.ofSeconds(10))
                .build();

        try {
            String urlEndpoint = "https://www.baidu.com/s";
            URI uri = URI.create(urlEndpoint + "?wd=java11");
            HttpRequest request = HttpRequest.newBuilder()
                    .uri(uri)
                    .build();
            HttpResponse<String> response = httpClient.send(request,
                    HttpResponse.BodyHandlers.ofString());

            System.out.println("Status code: " + response.statusCode());
            System.out.println("Headers: " + response.headers().allValues("content-type"));
            System.out.println("Body: " + response.body());
        } catch (IOException | InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
}
```

1.3 异步请求示例

如果我们不想等待响应，异步通信很有用。我们提供回调处理程序，当响应可用时执行。

注意使用 **sendAsync()** 方法发送异步请求。

```
package cn.dayangshuo.http;

import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.stream.Stream;

import static java.util.stream.Collectors.toList;

/**
 * @author DAYANG
 */
public class HttpClientTest {
    public static void main(String[] args) {
        final List<URI> uris = Stream.of(
                "https://www.baidu.com/",
                "https://www.zhihu.com/people/da-yang-12-48",
                "https://dayangshuo.cn"
        ).map(URI::create).collect(toList());
```

```
        HttpClient httpClient = HttpClient.newBuilder()
                .connectTimeout(Duration.ofSeconds(10))
                .followRedirects(HttpClient.Redirect.ALWAYS)
                .build();
        //回调设置
        CompletableFuture[] futures = uris.stream()
                .map(uri -> verifyUri(httpClient, uri))
                .toArray(CompletableFuture[]::new);

        CompletableFuture.allOf(futures).join();
    }

    private static CompletableFuture<Void> verifyUri(HttpClient httpClient, URI uri) {
        HttpRequest request = HttpRequest.newBuilder()
                .timeout(Duration.ofSeconds(5))
                .uri(uri)
                .build();

        return httpClient.sendAsync(request, HttpResponse.BodyHandlers.ofString())
                .thenApply(HttpResponse::statusCode)
                .thenApply(statusCode -> statusCode == 200)
                .exceptionally(ex -> false)
                .thenAccept(valid -> {
                    if (valid) {
                        System.out.println("[SUCCESS] Verified " + uri);
                    } else {
                        System.out.println("[FAILURE] Could not " + "verify " + uri);
                    }
                });
    }
}
```

2. 无需编译即可启动单文件程序

传统上，对于我们想要执行的每个程序，我们都需要先编译它。对于用于测试目的的小程序来说，这乎是不必要的冗长过程。

Java 11 改变了它，现在我们可以执行包含在单个文件中的 Java 源代码，而无需先编译它。

编写一个简单的HelloWorld.java

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

要执行上述类，请直接使用java命令运行它：

```
//控制台
$ java HelloWorld.java

Hello World!
```

请注意，该程序**不能使用**任何外部依赖除了使用java.base module模块化. 并且程序只能是 **单文件程序**。

3. 字符串API更新

3.1 String.repeat(Integer)

此方法只是重复一个字符串n次。它返回一个字符串，其值为重复 N 次的给定字符串的串联。

如果此字符串为空或计数为零，则返回空字符串。

```
public class HelloWorld {
    public static void main(String[] args) {
      String str = "1".repeat(5);
      //打印出：11111
      System.out.println(str);
    }
}
```

3.2. String.isBlank()

此方法指示字符串是空的还是仅包含空格。以前，我们一直在用 Apache 的StringUtils.java.

```
public class HelloWorld {
    public static void main(String[] args) {
      "1".isBlank();  //false
      "".isBlank(); //true
      "    ".isBlank(); //true
    }
}
```

3.3. String.strip()

此方法负责删除头和尾空格。

我们可以通过使用String.stripLeading()仅删除头字符，使用 String.stripTrailing ( )仅删除尾字符。

```
public class HelloWorld {
    public static void main(String[] args) {
      "   hi  ".strip();  //"hi"
      "   hi  ".stripLeading();  //"hi   "
      "   hi  ".stripTrailing(); //"   hi"
    }
}
```

3.4. String.lines()

此方法有助于将多行文本作为Stream处理。

```
public class HelloWorld {
    public static void main(String[] args) {
```

```
    String testString = "hello\nworld\nis\nexecuted";
    List<String> lines = new ArrayList<>();
    testString.lines().forEach(line -> lines.add(line));
    assertEquals(List.of("hello", "world", "is", "executed"), lines);
  }
}
```

4. Collection.toArray

在 Java 11 之前，将集合转换为数组并不简单。Java 11 使转换更加方便。

```
public class HelloWorld {
  public static void main(String[] args) {
    List<String> names = new ArrayList<>();
    names.add("alex");
    names.add("brian");
    names.add("charles");
    String[] namesArr1 = names.toArray(new String[names.size()]);  //Java 11之前
    String[] namesArr2 = names.toArray(String[]::new);          //Java 11
  }
}
```

5. Files.readString() 和 Files.writeString()

使用这些重载方法，Java 11 旨在减少大量样板代码，从而更容易读取和写入文件。

```
public class HelloWorld {
  public static void main(String[] args) {
    //读取文件内容为字符串
    URI txtFileUri = getClass().getClassLoader().getResource("helloworld.txt").toURI();
    String content = Files.readString(Path.of(txtFileUri),Charset.defaultCharset());
    //把字符串写入文件
    Path tmpFilePath = Path.of(File.createTempFile("tempFile", ".tmp").toURI());
    Path returnedFilePath = Files.writeString(tmpFilePath,"Hello World!",
            Charset.defaultCharset(), StandardOpenOption.WRITE);
  }
}
```

6. Optional.isEmpty()

Optional是一个容器对象，它可能包含也可能不包含非空值。如果不存在任何值，则该对象被认为是的。

isPresent()方法如果值存在则返回true，否则返回false。

isEmpty()方法与isPresent()方法相反，如果存在值则返回false，否则返回true。

所以我们无论如何都不要写否定条件。适当时使用这两种方法中的任何一种。

```
public class HelloWorld {
  public static void main(String[] args) {
    String currentTime = null;
    assertTrue(!Optional.ofNullable(currentTime).isPresent());
```

```
        assertTrue(Optional.ofNullable(currentTime).isEmpty());
        currentTime = "12:00 PM";
        assertFalse(!Optional.ofNullable(currentTime).isPresent());
        assertFalse(Optional.ofNullable(currentTime).isEmpty());
    }
}
```

转自https://zhuanlan.zhihu.com/p/480291145