Apache HTTP Client

Apache HttpClient é uma biblioteca Java que facilita a comunicação com servidores HTTP. Ela suporta todos os métodos HTTP, como GET, POST, PUT, DELETE, entre outros. HttpClient é amplamente utilizado para criar clientes HTTP em aplicações Java, especialmente quando é necessário um controle mais fino sobre os pedidos e respostas HTTP do que o fornecido pela biblioteca java.net.

Conceitos Principais

- **Instância de HttpClient**: Criação de uma instância de HttpClient para gerenciar os pedidos.
- **Métodos HTTP**: Suporte completo para métodos HTTP como GET, POST, PUT, DELETE, HEAD, OPTIONS, e TRACE.
- **Configuração de pedidos**: Personalização de pedidos HTTP, incluindo headers, parâmetros, e tempo de espera.
- **Gestão de Conexões**: Suporte para gestão de conexões, incluindo conexões persistentes e gestão de conexões em aplicações multi-thread.
- **Autenticação**: Suporte para vários esquemas de autenticação, como Basic, Digest, NTLM, e Kerberos.
- **Proxies**: Suporte para conexões através de proxies HTTP, incluindo proxies autenticados.
- **Cookies**: Manipulação automática de cookies, incluindo leitura e envio de cookies.
- **SSL/TLS**: Suporte para conexões seguras usando SSL/TLS.
- **Manipulação de Respostas**: Processamento de respostas HTTP, incluindo leitura de cabeçalhos e corpos de resposta.

Maven

```
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.13</version>
</dependency>
```

Conceito principais

HttpClient:

- Descrição: A interface principal para enviar requisições HTTP e receber respostas.
- Uso: Criação de instâncias de HttpClient para gerenciar requisições HTTP.

HttpRequest:

- Descrição: Representa uma requisição HTTP.
- Uso: Criação de requisições HTTP, como HttpGet, HttpPost, HttpPut, HttpDelete, etc.

HttpResponse:

- Descrição: Representa uma resposta HTTP.
- Uso: Manipulação de respostas HTTP, incluindo leitura de cabeçalhos e corpos de resposta.

HttpEntity:

- Descrição: Representa o corpo de uma requisição ou resposta HTTP.
- Uso: Manipulação de entidades HTTP, como StringEntity, FileEntity, InputStreamEntity, etc.

HttpContext:

- Descrição: Armazena informações de contexto para uma requisição HTTP.
- Uso: Passagem de informações de contexto entre diferentes componentes durante a execução de uma requisição.

HttpClientBuilder:

- Descrição: Utilizado para construir instâncias de HttpClient com configurações personalizadas.
- Uso: Configuração de parâmetros como tempo de espera, autenticação, proxies, etc.

RequestConfig:

- Descrição: Configurações específicas para uma requisição HTTP.
- Uso: Definição de parâmetros como tempo de conexão, tempo de resposta, redirecionamentos automáticos, etc.

CookieStore:

- Descrição: Armazena cookies para gerenciamento de estado HTTP.
- Uso: Manipulação de cookies durante a comunicação HTTP.

ConnectionManager:

- Descrição: Gerencia conexões HTTP, incluindo pooling de conexões.
- Uso: Gerenciamento eficiente de conexões em aplicações multi-thread.

SSLContext:

- Descrição: Configurações de SSL/TLS para conexões seguras.
- Uso: Configuração de certificados e chaves para conexões HTTPS.

Pedido GET

```
import org.apache.hc.client5.http.classic.methods.HttpGet;
import org.apache.hc.client5.http.impl.classic.CloseableHttpClient;
import org.apache.hc.client5.http.impl.classic.CloseableHttpResponse;
import org.apache.hc.client5.http.impl.classic.HttpClients;
import org.apache.hc.core5.http.io.entity.EntityUtils;

import java.io.IOException;

public class HttpClientExample {
    public static void main(String[] args) {
        try (CloseableHttpClient httpClient = HttpClients.createDefault()) {
            HttpGet request = new HttpGet("https://example.com");
            try (CloseableHttpResponse response = httpClient.execute(request)) {
```

```
                System.out.println(EntityUtils.toString(response.getEntity()));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Pedido POST

```
import org.apache.hc.client5.http.classic.methods.HttpPost;
import org.apache.hc.client5.http.entity.UrlEncodedFormEntity;
import org.apache.hc.client5.http.impl.classic.CloseableHttpClient;
import org.apache.hc.client5.http.impl.classic.CloseableHttpResponse;
import org.apache.hc.client5.http.impl.classic.HttpClients;
import org.apache.hc.core5.http.NameValuePair;
import org.apache.hc.core5.http.message.BasicNameValuePair;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class HttpClientPostExample {
    public static void main(String[] args) {
        try (CloseableHttpClient httpClient = HttpClients.createDefault()) {
            HttpPost post = new HttpPost("https://example.com/api");

            List<NameValuePair> urlParameters = new ArrayList<>();
            urlParameters.add(new BasicNameValuePair("param1", "value1"));
            urlParameters.add(new BasicNameValuePair("param2", "value2"));

            post.setEntity(new UrlEncodedFormEntity(urlParameters));

            try (CloseableHttpResponse response = httpClient.execute(post)) {
                System.out.println(EntityUtils.toString(response.getEntity()));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Tempos de espera

```
import org.apache.hc.client5.http.config.RequestConfig;
import org.apache.hc.client5.http.impl.classic.CloseableHttpClient;
import org.apache.hc.client5.http.impl.classic.HttpClients;
import org.apache.hc.core5.util.Timeout;
```

```
public class HttpClientTimeoutExample {
    public static void main(String[] args) {
        RequestConfig requestConfig = RequestConfig.custom()
                .setConnectTimeout(Timeout.ofSeconds(5))
                .setResponseTimeout(Timeout.ofSeconds(5))
                .build();

        try (CloseableHttpClient httpClient = HttpClients.custom()
                .setDefaultRequestConfig(requestConfig)
                .build()) {

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```