HTTP Client(Java21)

Certainly! Here are the detailed notes with all code snippets included:

---

1. Introduction to Java 11 HTTP Client:

- Java 11 introduced a standardized HTTP client API.
- It replaces the legacy `HttpUrlConnection` class.
- The `HttpUrlConnection` class was known for being low-level and lacking features.
- Previously, developers relied on third-party libraries like Apache HttpClient, Jetty, and Spring's RestTemplate.

2. Background:

- The change was implemented as part of JEP 321.
- Major changes include standardizing the HTTP API and incorporating it into the Java SE API.
- HTTP/2 protocol is introduced for better performance, featuring stream multiplexing, header compression, and push promises.
- The API is fully asynchronous, utilizing `CompletableFuture`.
- Core classes/interfaces include `HttpClient`, `HttpRequest`, `HttpResponse`, and `WebSocket`.
- Problems with the pre-Java 11 HTTP client include outdated design, blocking nature, and maintenance issues.

3. HTTP Client API Overview:

- Provides synchronous and asynchronous request mechanisms.
- Core classes include `HttpRequest`, `HttpClient`, and `HttpResponse`.

4. HttpRequest:

- Represents the request to be sent via `HttpClient`.

- Created using `HttpRequest.newBuilder()`.
- Methods like `uri()`, `GET()`, `POST()`, `PUT()`, `DELETE()` used for configuration.

5. Setting URI:

- URI can be set using constructor or `uri()` method.

```
HttpRequest.newBuilder(new URI("<https://postman-echo.com/get
>"))
```

6. Specifying the HTTP Method:

- Methods like `GET()`, `POST()`, `PUT()`, `DELETE()` specify the HTTP method.

```
HttpRequest.newBuilder()
    .uri(new URI("<https://postman-echo.com/get>"))
    .GET()
```

7. Setting HTTP Protocol Version:

- The default is HTTP/2 but can be explicitly set.

```
HttpRequest.newBuilder()
    .uri(new URI("<https://postman-echo.com/get>"))
    .version(HttpClient.Version.HTTP_2)
```

8. Setting Headers:

- Additional headers can be added using `headers()` or `header()` methods.

```
HttpRequest.newBuilder()
    .uri(new URI("<https://postman-echo.com/get>"))
    .headers("key1", "value1", "key2", "value2")
```

9. Setting a Timeout:

- Timeout can be set using the `timeout()` method.

```
HttpRequest.newBuilder()
    .uri(new URI("<https://postman-echo.com/get>"))
    .timeout(Duration.of(10, SECONDS))
```

10. Setting a Request Body:

- Body can be added using `POST()`, `PUT()`, `DELETE()` methods.
- BodyPublishers like `ofString()`, `ofInputStream()`, `ofByteArray()`, `ofFile()` are provided.

```
HttpRequest.newBuilder()
    .uri(new URI("<https://postman-echo.com/post>"))
    .POST(HttpRequest.BodyPublishers.noBody())
```

11. HttpClient:

- Used to send requests and receive responses.
- Methods like `send()`, `sendAsync()` are provided.

12. Handling Response Body:

- Methods like `ofByteArray()`, `ofString()`, `ofFile()` used for handling body.

```
HttpClient.newHttpClient()
    .send(request, HttpResponse.BodyHandlers.ofString());
```

13. Setting a Proxy:

- Proxy can be defined using the `proxy()` method.

```
HttpClient.newHttpClient()
    .proxy(ProxySelector.getDefault())
```

14. Setting the Redirect Policy:

- Redirect policy can be defined using `followRedirects()`.

```
HttpClient.newHttpClient()
    .followRedirects(HttpClient.Redirect.ALWAYS)
```

15. Setting Authenticator for a Connection:

- Authenticator can be set using the `authenticator()` method.

```
HttpClient.newHttpClient()
    .authenticator(new Authenticator() {
        @Override
        protected PasswordAuthentication getPasswordAuthentic
ation() {
            return new PasswordAuthentication("username", "pa
ssword".toCharArray());
        }
    })
```

16. Send Requests – Sync vs Async:

- `send()` for synchronous, `sendAsync()` for asynchronous calls.
- Asynchronous calls return a `CompletableFuture`.

```
HttpClient.newHttpClient()
    .sendAsync(request, HttpResponse.BodyHandlers.ofString())
```

17. Setting Executor for Asynchronous Calls:

- Executor can be defined to control the threads used for asynchronous calls.

```
HttpClient.newHttpClient()
    .executor(executorService)
```

18. Defining a CookieHandler:

- CookieHandler can be set using `cookieHandler()`.

```
HttpClient.newHttpClient()
    .cookieHandler(new CookieManager(null, CookiePolicy.ACCEP
T_NONE))
```

19. HttpResponse Object:

- Represents the response from the server.
- Provides methods like `statusCode()`, `body()`, `uri()`, `headers()`, `version()`.

20. Handling Push Promises in HTTP/2:

- Push promises are supported through `PushPromiseHandler` interface.
- `sendAsync()` method supports handling push promises.

21. Conclusion:

- Java 11 HttpClient API standardizes and enhances HTTP client capabilities.
- Features like HTTP/2 support, asynchronous processing, and push promises improve performance and usability.

These detailed notes cover all the sections of the article along with their corresponding code snippets. Let me know if you need further assistance!