AEM Invoke API – How to use HTTP Client Factory

Kiran Sg AEM 8th Nov 202129th Oct 2022 3 Minutes

Problem Statement:

We have created an HTTP Client factory but how to execute requests and handle responses?

Requirement:

Show some examples to build the request and make a request to the external system and handle the response.

Introduction:

HTTP Client factory is an OSGi Service that creates an HTTP connection based on the external system based on the domain and connection configuration.

In order to use this service,

Build URI:

we need to build the key value map using NameValuePair and add these params to **URIBuilder**. Once the request is created

```
@Override
  public String buildURL(@NotNull String apiEndpointURI, boolean buildExternalLink,
    Map < String, String > parameterMap) throws MalformedURLException {
    if (MapUtils.isNotEmpty(parameterMap)) {
      URIBuilder builder = new URIBuilder();
      List < NameValuePair > nvpList = new ArrayList < > (parameterMap.size());
      parameterMap.entrySet().stream()
        .filter(entry -> StringUtils.isNoneBlank(entry.getKey(), entry.getValue()))
        .forEach(entry -> nvpList.add(new BasicNameValuePair(entry.getKey(), entry.getValue())));
      return returnApiEndpointURI(apiEndpointURI, buildExternalLink, builder, nvpList);
    }
    return returnApiEndpointURI(apiEndpointURI, buildExternalLink, null, null);
  }

  private String returnApiEndpointURI(String apiEndpointURI, boolean buildExternalLink,
    URIBuilder builder, List < NameValuePair > nvpList) {
    if (buildExternalLink) {
      return StringUtils.join(httpClientFactory.getApiStoreLocatorHostName(),
        httpClientFactory.getExternalURIType(), apiEndpointURI,
        null != builder ? builder.addParameters(nvpList).toString() : StringUtils.EMPTY);
    } else {
      return StringUtils.join(apiEndpointURI,
        null != builder ? builder.addParameters(nvpList).toString() : StringUtils.EMPTY);
    }
  }
```

Make a Request:

we can make use of request methods like POST

```
entFactory.post(buildURL(apiEndPointURI, false, params)).addHeader("Content-Type", contentType)).handleResponse(HANDLER
```

Response Handler:

Get the response and handle the request using **StringObjectResponseHandler**.

```
1  package com.mysite.core.http.impl;
2
3  import java.io.IOException;
4  import org.apache.http.HttpResponse;
5  import org.apache.http.client.ClientProtocolException;
6  import org.apache.http.client.ResponseHandler;
7  import org.apache.http.client.utils.HttpClientUtils;
8  import org.apache.http.impl.client.BasicResponseHandler;
9
10 /**
11  * Handling response using Basic response Handler
12  */
13 public class StringObjectResponseHandler implements ResponseHandler < String > {
14
15   private BasicResponseHandler handler = new BasicResponseHandler();
16
17   @Override
18   public String handleResponse(HttpResponse httpResponse) throws
19   ClientProtocolException,
20   IOException {
21     String responseString = handler.handleResponse(httpResponse);
22     HttpClientUtils.closeQuietly(httpResponse);
23     return responseString;
24   }
25 }
```

Complete file for implementation:

```
1  package com.mysite.core.services.impl;
2
3  import com.drew.lang.annotations.NotNull;
4  import com.mysite.core.http.impl.StringObjectResponseHandler;
5  import com.mysite.core.services.APIInvoker;
6  import com.mysite.core.services.HttpClientFactory;
7  import org.apache.commons.collections4.MapUtils;
8  import org.apache.commons.lang3.StringUtils;
9  import org.apache.http.NameValuePair;
10 import org.apache.http.client.ClientProtocolException;
11 import org.apache.http.client.utils.URIBuilder;
12 import org.apache.http.message.BasicNameValuePair;
13 import org.apache.sling.api.servlets.HttpConstants;
14 import org.osgi.service.component.annotations.Component;
15 import org.osgi.service.component.annotations.Reference;
16 import org.slf4j.Logger;
17 import org.slf4j.LoggerFactory;
18 import java.io.IOException;
19 import java.net.MalformedURLException;
20 import java.util.ArrayList;
21 import java.util.List;
22 import java.util.Map;
23
24 @Component(service = APIInvoker.class)
25 public class ExampleAPIInvoker implements APIInvoker {
26   private static final Logger LOGGER = LoggerFactory.getLogger(ExampleAPIInvoker.class);
27   private static final StringObjectResponseHandler HANDLER = new StringObjectResponseHandler();
28
29   @Reference
30   private HttpClientFactory httpClientFactory;
31
32   @Override
33   public String invokeAPI(String apiEndPointURI, String httpMethod, Map < String, String > params,
```

```
34      String bodyText, String contentType) {
35      try {
36        if (StringUtils.isNotEmpty(bodyText)) {
37          LOGGER.info("API Request Params {}", bodyText);
38        } else {
39          LOGGER.info("API Request Params {}", params);
40        }
41        if (StringUtils.equalsAnyIgnoreCase(httpMethod, HttpConstants.METHOD_POST) &&
42          StringUtils.isEmpty(bodyText)) {
43          String responseString =
44            httpClientFactory.getExecutor().execute(httpClientFactory.post(buildURL(apiEndPointURI, false, params)
45          return responseString;
46        }
47      } catch (MalformedURLException exception) {
48        LOGGER.debug("MalformedURLException while invoking API, {}", exception.getMessage());
49      } catch (ClientProtocolException exception) {
50        LOGGER.debug("ClientProtocolException while invoking API, {}", exception.getMessage());
51      } catch (IOException exception) {
52        LOGGER.debug("IOException while invoking API, {}", exception.getMessage());
53      }
54      return StringUtils.EMPTY;
55    }
56
57    @Override
58    public String buildURL(@NotNull String apiEndpointURI, boolean buildExternalLink,
59      Map < String, String > parameterMap) throws MalformedURLException {
60      if (MapUtils.isNotEmpty(parameterMap)) {
61        URIBuilder builder = new URIBuilder();
62        List < NameValuePair > nvpList = new ArrayList < > (parameterMap.size());
63        parameterMap.entrySet().stream()
64          .filter(entry -> StringUtils.isNoneBlank(entry.getKey(), entry.getValue()))
65          .forEach(entry -> nvpList.add(new BasicNameValuePair(entry.getKey(), entry.getValue())));
66        return returnApiEndpointURI(apiEndpointURI, buildExternalLink, builder, nvpList);
67      }
68      return returnApiEndpointURI(apiEndpointURI, buildExternalLink, null, null);
69    }
70
71    private String returnApiEndpointURI(String apiEndpointURI, boolean buildExternalLink,
72      URIBuilder builder, List < NameValuePair > nvpList) {
73      if (buildExternalLink) {
```

```
74          return StringUtils.join(httpClientFactory.getApiStoreLocatorHostName(),
75            httpClientFactory.getExternalURIType(), apiEndpointURI,
76            null != builder ? builder.addParameters(nvpList).toString() : StringUtils.EMPTY);
77        } else {
78          return StringUtils.join(apiEndpointURI,
79            null != builder ? builder.addParameters(nvpList).toString() : StringUtils.EMPTY);
80        }
81      }
```

Create REST service using HTTP Client factory

— ***Check this out URI (https://kirantech58867409.wordpress.com/2021/11/08/%ef%bf%bcaem-rest-service-using-http-client-factory/)***

Tagged:
AEM,
AEM as a Cloud Service,
aemaacs,
API,
Best practices,
HttpClientBuilderFactory,
HttpClientFactory,
Invoke API

Published by Kiran Sg

My name is Kiran SG. I am currently working as an AEM architect and handling site revamp and migration. I try to learn new tricks and techniques every day to improve my coding and deliver value-added projects. All the tricks and techniques sometimes seem to be straightforward working code with best practices. But sometimes it won't satisfy the developer's needs. But most of the time code is either lagging to meet the requirement or doesn't follow best practices. Hence I am taking the initiative to put all my learnings and tricks in my blog series to share the working code with best practices. Hoping to help other developers like me. I have been working on AEM for 8 years now. I started my career as a Java developer. I have delivered many projects with a core customer-centric and strong focus on infrastructure and architecture. If you have any queries related to my blog topics please do connect with me directly on my email and also you can connect with me over LinkedIn and Facebook. View all posts by Kiran Sg

2 thoughts on “AEM Invoke API – How to use HTTP Client Factory”

Pingback: AEM Invoke API – REST service using HTTP Client factory – AEM best practices and development guide
Pingback: Exporting AEM Experience Fragment/Page Content for A/B Testing and External Systems without HTML Tags – AEM best practices and development guide