Item 111
git_comments:

1. * *
2. response
3. * * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
4. **comment:** use multiple values for the foo header in the reply
   label: code-design
5. * *
6. * * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
7. * *
8. * * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
9. * *
10. response
11. * * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
12. **comment:** use multiple values for the foo header in the reply
    label: code-design
13. mapping the content-type
14. use http helper to extract parameter value as it may contain multiple values
15. there may be multiple values for the same name
16. **comment:** use an iterator as there can be multiple values. (must not use a delimiter)
    label: code-design
17. if its a multi value then check each value if we can add it and for multi values they should be combined into a single value
18. **comment:** use an iterator as there can be multiple values. (must not use a delimiter)
    label: code-design
19. the value to add as request header
20. use the default toString of a ArrayList to create in the form [xxx, yyy] if multi valued, for a single value, then just output the value as is
21. add the value(s) as a http request header

22. **comment:** if we bridge endpoint then we need to skip matching headers with the HTTP_QUERY to avoid sending duplicated headers to the receiver, so use this skipRequestHeaders as the list of headers to skip
    label: code-design
23. use http helper to extract parameter value as it may contain multiple values
24. preserve headers from in by copying any non existing headers to avoid overriding existing headers with old values
25. **comment:** we should not add headers for the parameters in the uri if we bridge the endpoint as then we would duplicate headers on both the endpoint uri, and in HTTP headers as well
    label: code-design
26. remove the [ ] markers
27. trim value before checking for multiple parameters
28. * * Extracts the parameter value. * <p/> * This implementation supports HTTP multi value parameters which * is based on the syntax of <tt>[value1, value2, value3]</tt> by returning * a {@link List} containing the values. * <p/> * If the value is not a HTTP mulit value the value is returned as is. * * @param value the parameter value * @return the extracted parameter value, see more details in javadoc.
29. * * Appends the key/value to the headers. * <p/> * This implementation supports keys with multiple values. In such situations the value * will be a {@link java.util.List} that contains the multiple values. * * @param headers headers * @param key the key * @param value the value
30. there may be multiple values for the same name
31. **comment:** use an iterator as there can be multiple values. (must not use a delimiter)
    label: code-design
32. must use entrySet to ensure case of keys is preserved
33. if its a multi value then check each value if we can add it and for multi values they should be combined into a single value
34. **comment:** use an iterator as there can be multiple values. (must not use a delimiter)
    label: code-design
35. the value to add as request header
36. use the default toString of a ArrayList to create in the form [xxx, yyy] if multi valued, for a single value, then just output the value as is
37. add the value(s) as a http request header
38. **comment:** if we bridge endpoint then we need to skip matching headers with the HTTP_QUERY to avoid sending duplicated headers to the receiver, so use this skipRequestHeaders as the list of headers to skip
    label: code-design
39. use http helper to extract parameter value as it may contain multiple values
40. preserve headers from in by copying any non existing headers to avoid overriding existing headers with old values
41. **comment:** we should not add headers for the parameters in the uri if we bridge the endpoint as then we would duplicate headers on both the endpoint uri, and in HTTP headers as well
    label: code-design
42. remove the [ ] markers
43. trim value before checking for multiple parameters
44. * * Extracts the parameter value. * <p/> * This implementation supports HTTP multi value parameters which * is based on the syntax of <tt>[value1, value2, value3]</tt> by returning * a {@link List} containing the values. * <p/> * If the value is not a HTTP mulit value the value is returned as is. * * @param value the parameter value * @return the extracted parameter value, see more details in javadoc.
45. * * Appends the key/value to the headers. * <p/> * This implementation supports keys with multiple values. In such situations the value * will be a {@link java.util.List} that contains the multiple values. * * @param headers headers * @param key the key * @param value the value
46. must use response fields to get the http headers as httpExchange.getHeaders() does not work well with multi valued headers
47. preserve headers from in by copying any non existing headers to avoid overriding existing headers with old values
48. if its a multi value then check each value if we can add it and for multi values they should be combined into a single value
49. **comment:** use an iterator as there can be multiple values. (must not use a delimiter)
    label: code-design
50. propagate headers as HTTP headers
51. use the default toString of a ArrayList to create in the form [xxx, yyy] if multi valued, for a single value, then just output the value as is

52. add the value(s) as a http request header
53. **comment:** if we bridge endpoint then we need to skip matching headers with the HTTP_QUERY to avoid sending duplicated headers to the receiver, so use this skipRequestHeaders as the list of headers to skip
    label: code-design
54. the values to add as a request header
55. **comment:** we should not add headers for the parameters in the uri if we bridge the endpoint as then we would duplicate headers on both the endpoint uri, and in HTTP headers as well
    label: code-design
56. these tests does not run well on Windows
57. **comment:** response use multiple values for the foo header in the reply
    label: code-design
58. give Jetty time to startup properly

git_commits:

1. **summary:** CAMEL-4212: Add support for multi value headers for the http related components.
   message: CAMEL-4212: Add support for multi value headers for the http related components. git-svn-id: https://svn.apache.org/repos/asf/camel/trunk@1146782 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** http components should support parameters with multiple values when mapping to Camel headers
   description: All the http components in Camel should support mapping HttpServletRequest parameters to Camel Message headers where there can be multiple values for the same key. So in that case the value of the Camel Message header should be a List.
   label: requirement
2. **summary:** http components should support parameters with multiple values when mapping to Camel headers
   description: All the http components in Camel should support mapping HttpServletRequest parameters to Camel Message headers where there can be multiple values for the same key. So in that case the value of the Camel Message header should be a List.
   label: code-design
3. **summary:** http components should support parameters with multiple values when mapping to Camel headers
   description: All the http components in Camel should support mapping HttpServletRequest parameters to Camel Message headers where there can be multiple values for the same key. So in that case the value of the Camel Message header should be a List.
   label: code-design
4. **summary:** http components should support parameters with multiple values when mapping to Camel headers
   description: All the http components in Camel should support mapping HttpServletRequest parameters to Camel Message headers where there can be multiple values for the same key. So in that case the value of the Camel Message header should be a List.
   label: code-design
5. **summary:** http components should support parameters with multiple values when mapping to Camel headers
   description: All the http components in Camel should support mapping HttpServletRequest parameters to Camel Message headers where there can be multiple values for the same key. So in that case the value of the Camel Message header should be a List.

6. **summary:** http components should support parameters with multiple values when mapping to Camel headers
   description: All the http components in Camel should support mapping HttpServletRequest parameters to Camel Message headers where there can be multiple values for the same key. So in that case the value of the Camel Message header should be a List.
7. **summary:** http components should support parameters with multiple values when mapping to Camel headers
   description: All the http components in Camel should support mapping HttpServletRequest parameters to Camel Message headers where there can be multiple values for the same key. So in that case the value of the Camel Message header should be a List.

jira_issues_comments:

1. **body:** Work in progress patch
   label: requirement
2. **body:** I have attached a patch. It covers most of the stuff. It needs a bit polishing in the HttpHelper in the extractParameterName method etc. The camel-ahc, and camel-http4 components has not been migrated. And I think a few more unit tests would be appropriate.
   label: code-design
3. **body:** This also applies for HTTP headers. And on the producer side, you may have a Camel Message that has a header which contains a list with multiple values. It seems so the idiom is to do a .toString on the value. Which in Java outputs a: [xxx, yyy, zzz] for an ArrayList.
   label: code-design
4. **body:** Okay was a bit more icky to implement as we have in Camel - 4 http producers - 2 consumers And they all got a bit different API, and some works better out of the box with multi values and others dont. Even Jetty on the consumer side had issues, so I had to use a different API to grab multi values. camel-ahc worked the best out of the box, as it supported it without any code changes needed.
   label: code-design
5. This was fixed between the original 2.8.0 release attempt and the redo so it's part of 2.8.0