```
File - /Users/beathoven/Library/Mobile Documents/com~apple~CloudDocs/Projects/translation-tool/src/main/java/net/wiredclub/translation/TranslationTool.java
 1 package net.wiredclub.translation;
 2
 3 import com.fasterxml.jackson.databind.JsonNode;
 4 import com.fasterxml.jackson.databind.node.ArrayNode;
 5 import com.flipkart.zisonpatch.JsonDiff:
 6 import com.flipkart.zjsonpatch.JsonPatch;
 7 import org.slf4j.Logger;
 8 import org.slf4j.LoggerFactory;
10 import java.io.IOException;
11 import java.util.Iterator;
12
13 import static net.wiredclub.translation.DeepLHelper.DeepLUsage;
14 import static net.wiredclub.translation.TranslationStatusCode.STATUS_BAD_AS_HELL;
15 import static net.wiredclub.translation.TranslationStatusCode.STATUS_FILE_NOT_FOUND;
16 import static net.wiredclub.translation.TranslationStatusCode.STATUS_JSON_INVALID;
17 import static net.wiredclub.translation.TranslationStatusCode.STATUS_OK;
18 import static net.wiredclub.translation.TranslationStatusCode.STATUS_TRANSLATION_FILE_INVALID;
19
20 /**
21 * Translate all modified keys from source language into target language.
22 * The previous version with translations is taken from previous commit.
23 */
24 public class TranslationTool {
25
26
       private static final Logger LOG = LoggerFactory.getLogger(TranslationTool.class);
27
       private final JsonHelper jsonHelper;
28
29
       private final DeepLHelper deepLHelper;
       private final FileHelper fileHelper;
30
31
       private final CommandLineHelper commandLineHelper;
32
33
       private TranslationConfig cfg;
34
35
36
        * Without using CDI we instantiate all required classes here. For mocking
37
        * of some classes the second constructor is used.
```

```
File - /Users/beathoven/Library/Mobile Documents/com~apple~CloudDocs/Projects/translation-tool/src/main/java/net/wiredclub/translation/Translation/Tool.java
38
         */
39
       TranslationTool() {
40
            this.jsonHelper = new JsonHelper();
41
            this.deepLHelper = new DeepLHelper(jsonHelper);
            this.fileHelper = new FileHelper();
42
            this.commandLineHelper = new CommandLineHelper(deepLHelper, fileHelper);
43
       }
44
45
46
       /**
47
         * Constructor for tests.
48
49
         * @param isonHelper either real or mock class
50
         * @param deepLHelper either real or mock class
        * @param fileHelper either real or mock class
51
52
        * @param commandLineHelper either real or mock class
53
         */
54
       TranslationTool(JsonHelper jsonHelper, DeepLHelper deepLHelper, FileHelper fileHelper,
55
                        CommandLineHelper commandLineHelper) {
56
            this.jsonHelper = jsonHelper;
57
            this.deepLHelper = deepLHelper;
58
            this.fileHelper = fileHelper;
59
            this.commandLineHelper = commandLineHelper;
60
       }
61
62
       public static void main(String[] args) {
            System.exit(new TranslationTool().run(args).exitCode());
63
64
       }
65
66
       TranslationStatusCode run(String[] args) {
            try {
67
68
                cfg = commandLineHelper.getTranslationConfig(args);
                processTranslation();
69
            } catch (TranslationException e) {
70
                String message = e.getMessage();
71
                if (message ≠ null && !message.isBlank()) {
72
73
                    LOG.warn(message);
74
                }
```

```
File - /Users/beathoven/Library/Mobile Documents/com~apple~CloudDocs/Projects/translation-tool/src/main/java/net/wiredclub/translation/TranslationTool.java
 75
                return e.statusCode();
 76
            } catch (Throwable e) {
                 LOG.error(e.getMessage(), e);
 77
                 return STATUS_BAD_AS_HELL;
 78
 79
            return STATUS_OK;
 80
        }
 81
 82
         /**
 83
         * <01>
 84
                Read source file and find differences to the previous version
 85
         *
                For every target language
 86
 87
         *
                <01>
                    Read target file and find differences to source file
 88
                    Translate text and create patch operation changes between target to source
 89
                    Translate remaining source text changes and create patch operation for source \langle li>
 90
 91
                    Write output file
 92
                </01>
 93
         * 
 94
         * Look at the activity diagram in documentation folder for a graphical overview.
 95
 96
         * @throws TranslationException throws exception if translation is not possible
 97
         * @throws IOException throws exception if an error during file IO occurs
 98
 99
         */
        private void processTranslation() throws TranslationException, IOException {
100
            // run information for devs
101
            if (LOG.isDebugEnabled()) {
102
103
                 LOG.info("Translation tool started.");
                 LOG.info("Source language: {}", cfg.sourceLanguage());
104
                 LOG.info("Target language(s): {}", cfg.targetLanguages());
105
                 LOG.info("Translations directory: {}", cfq.translationsDirectory());
106
                 DeepLUsage usage = deepLHelper.usage();
107
                 LOG.info("DeepL translations possible: {}/{}", usage.characterCount(), usage.characterLimit());
108
            }
109
110
            // read actual main.json
111
```

```
File - /Users/beathoven/Library/Mobile Documents/com~apple~CloudDocs/Projects/translation-tool/src/main/java/net/wiredclub/translation/Translation/Tool.java
            JsonNode sourceJson = qetTranslationFile(cfq.sourceFileName());
112
113
            // find all changes from previous version of main.json to actual main.json
114
            JsonNode sourceDiffPatch = findChangesInSource(sourceJson):
115
116
            for (String targetLanguage : cfg.targetLanguages()) {
117
                 boolean translationsFileChanged = false:
118
119
120
                 JsonNode targetJson = getTranslationFile(cfg.targetFileName(targetLanguage));
121
                // This call is a bit weird, because we use target json as first parameter (source) and source as
122
                // second (target). This is because the names are used in a different context. We want to know which keys
123
                // need to be added to or removed from target ison in comparison to source ison. The target ison will be
124
125
                // transformed into the same structure as source ison.
                 JsonNode targetDiffPatch = JsonDiff.asJson(targetJson, sourceJson);
126
                 // LOG.debug("target to source diff patch: {}", targetKeyDiffPatch.toPrettyString());
127
128
                 if (!targetDiffPatch.isEmpty()) {
129
                     // create patch with add or remove fields (field values will be translated).
130
                     JsonNode translationPatch = translateTargetDiffPatch(targetDiffPatch, targetLanguage);
131
132
                     if (!translationPatch.isEmpty()) {
                         translationsFileChanged = true;
133
                         LOG.info("Created patch (KEYS DIFF) with {} operation(s)/translation(s) for '{}'.",
134
135
                                 translationPatch.size(), cfg.targetFileName(targetLanguage));
                         // LOG.debug("{}", translationPatch.toPrettyString());
136
137
                         // add or remove keys in target json
138
                         JsonPatch.applyInPlace(translationPatch, targetJson);
139
140
                     }
                 }
141
142
                 if (!sourceDiffPatch.isEmpty()) {
143
                     // create patch with replace operations
144
145
                     JsonNode translationPatch = translateSourceDiffPatch(sourceDiffPatch, targetLanguage);
                     if (!translationPatch.isEmpty()) {
146
147
                         translationsFileChanged = true;
                         LOG.info("Created patch (VALUE DIFF) with {} translation(s) for '{}'.",
148
```

```
File - /Users/beathoven/Library/Mobile Documents/com~apple~CloudDocs/Projects/translation-tool/src/main/java/net/wiredclub/translation/TranslationTool.java
                                  translationPatch.size(), cfg.targetFileName(targetLanguage));
149
                         // LOG.debug("{}", translationPatch.toPrettuString());
150
151
                         // replace keys in target ison
152
                         JsonPatch.applyInPlace(translationPatch, targetJson);
153
                     }
154
                 }
155
156
157
                 // write result into target directory and overwrite existing translation file.
158
                 if (translationsFileChanged) {
                     writeTargetTranslationFile(targetJson, targetLanguage);
159
                 }
160
            }
161
162
             LOG.info("Translation process finished but files were not committed and pushed."
                     + "Please verify translation files and commit and push them.");
163
        }
164
165
166
        private JsonNode getTranslationFile(String filename) throws TranslationException {
167
            try {
                 String source = fileHelper.readFile(cfg.repositoryDirectory() + "/" + filename);
168
169
                 return jsonHelper.convertStringToJson(source);
                 // LOG.debug("source json: {}", sourceJson.toPrettyString());
170
171
             } catch (TranslationJsonProcessingException e) {
172
                 throw new TranslationException(
                         "Error: Invalid Json. Please verify that the file '" + filename + "' is valid json. "
173
174
                                  + "Cause: " + e.getMessage(),
                         STATUS_JSON_INVALID);
175
            } catch (TranslationFileNotFoundException e) {
176
177
                 throw new TranslationException("Error: '" + filename + "' not found. "
                         + "Please verify that the file exists.", STATUS_FILE_NOT_FOUND);
178
179
             }
180
        }
181
182
        private JsonNode findChangesInSource(JsonNode sourceJson) throws IOException, TranslationException {
183
             String sourceFileName = cfg.sourceFileName();
184
             try {
185
                 return createDiffPatch(cfq.repositoryDirectory(), sourceFileName, sourceJson);
```

```
File - /Users/beathoven/Library/Mobile Documents/com~apple~CloudDocs/Projects/translation-tool/src/main/java/net/wiredclub/translation/TranslationTool.java
                 // LOG.debug("diff json patch: {}", diffPatch.toPrettyString());
186
187
            } catch (TranslationFileNotFoundException e) {
                 throw new TranslationException("Error: '" + sourceFileName + "' not found. "
188
                         + "Please verify that the previous version of file exists in git.", STATUS_FILE_NOT_FOUND);
189
190
            } catch (TranslationJsonProcessingException e) {
191
                 throw new TranslationException(
                         "Error: Invalid Json. Please verify that the file '" + sourceFileName + "' is valid json.",
192
193
                         STATUS_JSON_INVALID);
194
195
        }
196
197
        JsonNode createDiffPatch(String repositoryDirectory, String previousTranslationsFileName,
198
                                   JsonNode actualTranslationsJson)
199
                 throws TranslationFileNotFoundException, TranslationJsonProcessingException, IOException {
200
            String previousTranslations =
                     fileHelper.readPreviousFileFromHistory(repositoryDirectory, previousTranslationsFileName, 1);
201
202
203
            JsonNode previousTranslationsJson = jsonHelper.convertStringToJson(previousTranslations);
204
            return JsonDiff.asJson(previousTranslationsJson, actualTranslationsJson);
        }
205
206
         /**
207
         * Translates target diff patch. Only add and remove operations are handled here.
208
209
         * Replace operation will be handled by translate source diff patch.
         * All other operations are not needed.
210
211
         *
         * @param diffPatch changes of source file
212
         * @param targetLanguage the target language
213
214
215
         * @return a translation patch
216
         * @throws TranslationException thrown if the translation patch is an invalid json
217
218
         * <u>@throws</u> IOException thrown if an error occurs during file access
219
         */
220
        private JsonNode translateTargetDiffPatch(JsonNode diffPatch, String targetLanguage)
221
                 throws TranslationException, IOException {
222
            ArrayNode translationPatch = jsonHelper.createNewTranslationPatch();
```

```
File - /Users/beathoven/Library/Mobile Documents/com~apple~CloudDocs/Projects/translation-tool/src/main/java/net/wiredclub/translation/TranslationTool.java
223
224
             if (diffPatch.isArrav()) {
225
                 for (int i = 0; i < diffPatch.size(); i++) {</pre>
                     JsonNode command = diffPatch.get(i);
226
                      // LOG.debug("command: {}", command);
227
228
                     String op = command.get("op").asText();
229
230
                     String path = command.get("path").asText();
                      switch (op) {
231
                          case "add":
232
                              JsonNode value = command.get("value");
233
234
                              translationPatch.add(jsonHelper.createPatchOperationAdd(path, value));
                              traverse(translationPatch, targetLanguage, path, value);
235
236
                              break;
237
                          case "remove":
                              translationPatch.add(jsonHelper.createPatchOperationRemove(path));
238
239
                              break:
                          default:
240
241
                              // replace would do an unnecessary translation.
242
                              // translations for operation move and copy is also not needed.
243
                 }
244
             }
245
246
             return translationPatch;
247
248
         }
249
250
251
         * Translates source diff patch. Only the replace operation is handled here.
          * All other operations are already handled or not needed.
252
253
          * @param diffPatch changes of source file
254
          * @param targetLanguage the target language
255
256
          *
          * @return a translation patch
257
258
          * @throws TranslationException thrown if the translation patch is an invalid json
259
```

```
File - /Users/beathoven/Library/Mobile Documents/com~apple~CloudDocs/Projects/translation-tool/src/main/java/net/wiredclub/translation/Translation/Tool.java
260
          * Athrows IOException thrown if an error occurs during file access
261
         */
         private JsonNode translateSourceDiffPatch(JsonNode diffPatch, String targetLanguage)
262
                 throws TranslationException, IOException {
263
             ArrayNode translationPatch = jsonHelper.createNewTranslationPatch();
264
265
             if (diffPatch.isArrav()) {
266
                 for (int i = 0; i < diffPatch.size(); i++) {</pre>
267
                     JsonNode command = diffPatch.get(i);
268
                     // LOG.debug("command: {}", command);
269
270
                     String op = command.get("op").asText();
271
                     // for operations add, remove, move, and copy translation is not needed
272
                     if ("replace".equals(op)) {
273
274
                         traverse(translationPatch, targetLanguage, command.get("path").asText(), command.get("value"));
                     }
275
276
                 }
             }
277
278
279
             return translationPatch;
280
         }
281
         /**
282
283
          * Recursive approach to iterate through json tree.
284
         * Oparam patch operations how to change the target ison will be added to the patch
285
         * @param targetLanguage the target language
286
         * <u>Oparam</u> path the path is a unique identifier. it is build from all successor field names and the actual field name.
287
         * @param jsonNode the json node to be evaluated
288
289
         * Othrows TranslationException thrown if an array is defined in ison, or translation has an invalid ison
290
         * @throws IOException thrown if an error occurs during file access
291
292
          */
         private void traverse(ArrayNode patch, String targetLanguage, String path, JsonNode jsonNode)
293
                 throws TranslationException, IOException {
294
             if (jsonNode.isObject()) {
295
                 Iterator<String> fieldNames = jsonNode.fieldNames();
296
```

```
File - /Users/beathoven/Library/Mobile Documents/com~apple~CloudDocs/Projects/translation-tool/src/main/java/net/wiredclub/translation/TranslationTool.java
297
                 while (fieldNames.hasNext()) {
                     String fieldName = fieldNames.next();
298
299
                     JsonNode fieldValue = jsonNode.get(fieldName);
                     traverse(patch, targetLanguage, path + "/" + fieldName, fieldValue);
300
301
302
             } else if (jsonNode.isArray()) {
                 throw new TranslationException("Error: Arrays are not allowed in translation file 'main.json'.",
303
                         STATUS_TRANSLATION_FILE_INVALID);
304
305
             } else {
                 String textToTranslate = isonNode.asText():
306
                 String translation = deepLHelper.translate(textToTranslate, cfg.sourceLanguage(), targetLanguage);
307
                 JsonNode command = jsonHelper.createPatchOperationReplace(path, translation);
308
                 patch.add(command);
309
310
311
        }
312
313
         /**
         * Oparam appliedTranslationPatch a json that holds all values which should be written to an output file
314
315
         * Oparam targetLanguage the desired target language
316
         *
317
         * Othrows TranslationException thrown if the translation patch is an invalid ison
318
         * Othrows IOException thrown if an error occurs during file access
319
320
        private void writeTargetTranslationFile(JsonNode appliedTranslationPatch, String targetLanguage)
                 throws TranslationException, IOException {
321
322
             try {
323
                 String target = jsonHelper.convertJsonToString(appliedTranslationPatch);
                 String targetFileName = cfq.repositoryDirectory() + "/" + cfq.targetFileName(targetLanguage);
324
325
                 fileHelper.writeFile(targetFileName, target);
326
                 LOG.info("File written to '{}'.", targetFileName);
             } catch (TranslationJsonProcessingException e) {
327
328
                 throw new TranslationException(
329
                         "Error: Could not create a valid json file. Something has gone wrong. Please check.",
330
                         STATUS_JSON_INVALID);
331
        }
332
333 }
```