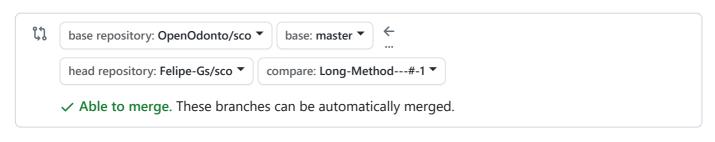
	penC)donto	/ sco (Public
,M-		o o i i co	, 500	

<> Code	Issues	! Pull requests	Actions		₩ Wiki	Security	<u>l∼</u> In
V/ Couc	0 133463	8 8 Tull requests	Actions	Ш тюјеста	WIKI	• Security	···

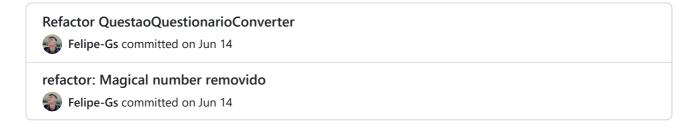
Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more about diff comparisons.



-ত- 2 commits 🚊 1 file changed 🔍 1 contributor

-o- Commits on Jun 14, 2023



Showing 1 changed file with 55 additions and 51 deletions.

```
Split Unified
```

```
106
...nto-web/src/main/java/br/ueg/openodonto/visao/converter/QuestaoQuestionarioConverter.java
  15
         15
                 import br.ueg.openodonto.controle.ManterQuestionarioAnamnese;
  16
         16
                 import br.ueg.openodonto.util.bean.QuestionarioQuestaoAdapter;
  17
               - public class QuestaoQuestionarioConverter implements Converter{
  18
  19
  20
  21
                         private Map<String,Integer>
                                                         indexMap;
                         private boolean
  22
                                                                         isIndexed;
  23
                         private void doIndexMap(FacesContext context, UIComponent component){
  24
                                 indexMap = new HashMap<String, Integer>();
  25
                                 UIOrderingList orderingList = (UIOrderingList) component;
  26
  27
                                 String clientId = orderingList.getBaseClientId(context);
                         ExternalContext externalContext = context.getExternalContext();
  28
                                 Map<String, String[]> requestParameterValuesMap =
  29
                 externalContext.getRequestParameterValuesMap();
```

```
String[] strings = (String[])
30
               requestParameterValuesMap.get(clientId);
                               if (strings != null && strings.length != 0) {
31
                                        for(String string : strings){
32
                                                boolean fwl =
33
               Character.isLetter(string.charAt(0));
34
                                                boolean swl =
               Character.isLetter(string.charAt(1));
                                                int indexDivisor = string.indexOf(":");
35
36
                                                int indexReal = fwl ? swl ? 2 : 1 : 0;
                                                int indexVirtual = indexDivisor+1;
37
                                                Integer real =
38
               Integer.valueOf(string.substring(indexReal,indexDivisor));
                                                String virtual = string.substring(indexVirtual);
39
                                                indexMap.put(virtual, real);
40
41
                                        }
42
                               }
43
                               isIndexed = true;
44
                       }
45
                       @Override
46
                       public Object getAsObject(FacesContext context, UIComponent component,
47
               String value) {
                               if(!isIndexed){
48
                                        doIndexMap(context,component);
49
50
51
                               Integer id = indexMap.get(value);
                               ManterQuestionarioAnamnese mBean = getMbean(context);
52
                               List<QuestionarioQuestaoAdapter> questoes =
53
               mBean.getManageQuestao().getQuestoesAdapter();
                               QuestionarioQuestaoAdapter questao = questoes.get(id);
54
55
                               return questao;
                       }
56
57
                       @Override
58
                       public String getAsString(FacesContext context, UIComponent component,
59
               Object value) {
                               QuestionarioQuestaoAdapter adapter =
60
               (QuestionarioQuestaoAdapter) value;
                               return adapter.getQqa().getIndex().toString();
61
62
                       }
63
64
                       private ManterQuestionarioAnamnese getMbean(FacesContext context){
                               ELContext elctx = context.getELContext();
65
                               ManterQuestionarioAnamnese mBean =
66
               (ManterQuestionarioAnamnese)elctx.getELResolver().getValue(elctx, null,
               "manterQuestionarioAnamnese");
67
                               return mBean;
68
                       }
              public class QuestaoQuestionarioConverter implements Converter {
        18
        19
        20
                   private Map<String, Integer> indexMap;
```

```
21
                   private boolean isIndexed;
69
       22
                       private static final int INDEX REAL FIRST LETTER = 1;
       23
                   private static final int INDEX_REAL_SECOND_LETTER = 2;
       24
                   private static final int INDEX REAL DEFAULT = 0;
       25
       26
       27
                       private Map<String, Integer> createIndexMap(UIOrderingList orderingList,
               Map<String, String[]> requestParameterValuesMap, String clientId) {
                       Map<String, Integer> indexMap = new HashMap<>();
       28
       29
                       String[] strings = requestParameterValuesMap.get(clientId);
       30
                       if (strings != null && strings.length != 0) {
       31
       32
                           for (String string : strings) {
                               boolean firstCharIsLetter =
       33
               Character.isLetter(string.charAt(0));
       34
                               boolean secondCharIsLetter =
               Character.isLetter(string.charAt(1));
       35
                               int indexDivisor = string.indexOf(":");
                               int indexReal = firstCharIsLetter ? (secondCharIsLetter ?
       36
               INDEX_REAL_SECOND_LETTER : INDEX_REAL_FIRST_LETTER) : INDEX_REAL_DEFAULT;
                               int indexVirtual = indexDivisor + 1;
       37
                               Integer real = Integer.valueOf(string.substring(indexReal,
       38
               indexDivisor));
                               String virtual = string.substring(indexVirtual);
       39
                               indexMap.put(virtual, real);
       40
       41
                           }
       42
                       }
       43
                       return indexMap;
       44
                   }
       45
                   private void doIndexMap(FacesContext context, UIComponent component) {
       46
       47
                       indexMap = createIndexMap(context, component);
       48
                       isIndexed = true;
       49
                   }
       50
                   @Override
       51
                   public Object getAsObject(FacesContext context, UIComponent component,
       52
               String value) {
                       if (!isIndexed) {
       53
       54
                           doIndexMap(context, component);
       55
                       }
       56
                       Integer id = indexMap.get(value);
       57
                       ManterQuestionarioAnamnese mBean = getMbean(context);
                       List<QuestionarioQuestaoAdapter> questoes =
       58
               mBean.getManageQuestao().getQuestoesAdapter();
       59
                       QuestionarioQuestaoAdapter questao = questoes.get(id);
                       return questao;
       60
       61
                   }
       62
       63
                   public String getAsString(FacesContext context, UIComponent component,
       64
               Object value) {
```

```
04/12/2023, 10:46
```

```
QuestionarioQuestaoAdapter adapter = (QuestionarioQuestaoAdapter) value;
       65
                       return adapter.getQqa().getIndex().toString();
       66
       67
                  }
       68
                  private ManterQuestionarioAnamnese getMbean(FacesContext context) {
       69
       70
                       ELContext elctx = context.getELContext();
       71
                       ManterQuestionarioAnamnese mBean = (ManterQuestionarioAnamnese)
               elctx.getELResolver().getValue(elctx, null, "manterQuestionarioAnamnese");
                       return mBean;
       72
       73
                  }
70
       74
               }
```