POLITECNICO
MILANO 1863

Computer Science and Engineering

A.A. 2016/2017

Software Engineering 2 Project:

# "PowerEnJoy"

**C**ode **I**spection

January 29, 2017

Prof.Luca Mottola

Matteo Michele Piazzolla Matr. 878554

Andrea Millimaggi Matr. 876062

## Revision History

| **Revision** | **Date** | **Author(s)** | **Description** |
|---|---|---|---|
| 1.0 | 29/01/17 | Piazzolla Millimaggi | First document issue |
| | | | |
| | | | |

# Contents

**1 Assigned Classes 5**
1.1 OFBizSolrRedirectServlet . . . . . 5
1.2 TraverseSubContentCacheTransform . . . . . 5

**2 Functional role 6**

**3 List of issues OFBizSolrRedirectServlet 7**
3.1 Naming Conventions . . . . . 7
3.2 Indention . . . . . 7
3.3 Braces . . . . . 7
3.4 File Organization . . . . . 7
3.5 Wrapping Lines . . . . . 8
3.6 Comments . . . . . 8
3.7 Java Source Files . . . . . 8
3.8 Package and Import Statements . . . . . 8
3.9 Class and Interface Declarations . . . . . 8
3.10 Initialization and Declarations . . . . . 9
3.11 Method Calls . . . . . 9
3.12 Arrays . . . . . 9
3.13 Object Comparison . . . . . 9
3.14 Output Format . . . . . 9
3.15 Computation, Comparisons and Assignments . . . . . 10
3.16 Exceptions . . . . . 10
3.17 Flow of Control . . . . . 10
3.18 Files . . . . . 10

**4 List of issues TraverseSubContentCacheTransform 11**
4.1 Naming Conventions . . . . . 11
4.2 Indention . . . . . 11
4.3 Braces . . . . . 11
4.4 File Organization . . . . . 11
4.5 Wrapping Lines . . . . . 12
4.6 Comments . . . . . 12
4.7 Java Source Files . . . . . 13
4.8 Package and Import Statements . . . . . 13
4.9 Class and Interface Declarations . . . . . 13
4.10 Initialization and Declarations . . . . . 13
4.11 Method Calls . . . . . 14
4.12 Arrays . . . . . 14
4.13 Object Comparison . . . . . 14
4.14 Output Format . . . . . 14
4.15 Computation, Comparisons and Assignments . . . . . 14
4.16 Exceptions . . . . . 15
4.17 Flow of Control . . . . . 15

4.18 Files . . . . . . . . . . 15

**5 Appendix 16**

5.1 Used software . . . . . . . . . . 16

5.2 Source Code . . . . . . . . . . 17

5.3 OFBizSolrRedirectServlet . . . . . . . . . . 17

5.4 TraverseSubContentCacheTransform . . . . . . . . . . 19

5.5 Time effort . . . . . . . . . . 25

# 1 Assigned Classes

## 1.1 OFBizSolrRedirectServlet

**Class:** `OFBizSolrRedirectServlet`
**File:** OFBizSolrRedirectServlet.java
**Package:** `org.apache.ofbiz.solr.webapp`
**Path:** specialpurpose/solr/src/main/java/org/apache/ofbiz/solr/webapp
**Description:**

## 1.2 TraverseSubContentCacheTransform

**Class:** `TraverseSubContentCacheTransform`
**File:** TraverseSubContentCacheTransform.java
**Package:** `org.apache.ofbiz.content.webapp.ftl`
**Path:** applications/content/src/main/java/org/apache/ofbiz/content/webapp/ftl
**Description:** an URL template model for FreeMarker.

## 2 Functional role

Both the classes assigned to our group belong to the package org.ofbiz.solr.webapp. **Solr** is a standalone enterprise server with a REST-like API. Solr allows to put documents in the server and to query them. The documents can be put on the server via JSON, XML, CSV or over HTTP. Queries can be done using HTTP GET request method and the result are given in the format used for the document upload. From the name of the package we can easily deduce that the classes are components of the Solr web application.

The functional role of the **OFBizSolrRedirectServlet** is clearly explained by the javadoc:

```
/**
 * OFBizSolrRedirectServlet.java - Master servlet for the ofbiz-solr application.
 */
```

The javadoc of the **TraverseSubContentCacheTransform** says:

```
/**
 * TraverseSubContentCacheTransform - Freemarker Transform for URLs (links)
 */
```

Apache FreeMarker is a template engine, that is a library that generates text output based on templates. So we can deduce that the TraverseSubContent-CacheTransform class is a class for the generation of text representing URLs.

# 3 List of issues OFBizSolrRedirectServlet

## 3.1 Naming Conventions

1 No issue

2 No issue - No one-character variable is used

3 No issue

4 No issue - There isn't any interface

5 Issue at line 46 - The first letter of the method "DoGet(HttpServletRequest request, HttpServletResponse response)" is capitalized

6 No issue

7 No issue - There isn't any constant

## 3.2 Indention

8 No issue - Always three spaces are used for indention

9 No issue - No tabs found in the code

## 3.3 Braces

10 No issue - Kernighan and Ritchie style is used consistently.

11 No issue - There are only *if* statements and they are all surrounded by curly braces.

## 3.4 File Organization

12 No issue - All sections are separated by a blank line.

13
- Issue at line 35 - Length = 82 - It is impractical to split the last word.
- Issue at line 40 - Length = 82 - It is impratical to split the method "getName()"
- Issue at line 43 - Length = 128 - "javax.servlet.http.HttpServletResponse)" could be put in the following line
- Issue at line 46 - Length = 119 - impractical to put the *throw* declaration in the following line
- Issue at line 55 - Length = 119 - impractical to put the *throw* declaration in the following line
- Issue at line 57 - Length = 83 - impractical to split the method.
- Issue at line 70 - Length = 87 - impractical to split the method
- Issue at line 74 - Length = 87 - impractical to split the method

14 Issue at line 43 - Length = 128 - See [13]

## 3.5 Wrapping Lines

15 No issue - No line breaks

16 No issue - No line breaks

17 No issue - Expressions are all aligned.

## 3.6 Comments

18 Comments are used only for briefly explain what the class does and to give a reference to the method that the "doGet" method overrides. There are no comments for the "forwardUrl" method and for blocks of code.

19 No issue - There is no commented out code

## 3.7 Java Source Files

20 No issue

21 No issue

22 No issue

23 No issue

## 3.8 Package and Import Statements

24 **Lines 31:** `import org.apache.ofbiz.security.Security;`
**Unused import.**
**Solution:** Remove the unused import line.

## 3.9 Class and Interface Declarations

25
- a No issue
- b No issue
- c No issue - No implementation comments
- d No issue
- e No issue - No instance variables
- f No issue - No constructors
- g No issue

26 No issue - There are only two methods

27
- No duplicates
- Methods aren't too long (24 lines for forwardUrl and 7 lines for doGet)

  - The class isn't too big (45 lines)
  - No breaking encapsulation - The only attribute (module) is final.
  - The cohesion is high, because all the methods work to implement the main functionality of the server, that is get and forward information.
  - Coupling is low, because the methods are independent.

### 3.10 Initialization and Declarations

28 No issue

29 No issue

30 No issue

31 No issue

32 No issue

33 No issue

### 3.11 Method Calls

34 No issue

35 No issue

36 No issue

### 3.12 Arrays

- No arrays are used in the class behavior.

### 3.13 Object Comparison

40 There isn't any comparison

### 3.14 Output Format

There is no displayed output.

### 3.15 Computation, Comparisons and Assignments

44 No issue - All the operations seem to be effective and concise and there is no evidence of the use of brute force solutions

45-46 No issue - The only operations present in the code are the AND operations at lines 70 and 74 and they involve only one operation between two elements, so they don't need the management of precedences.

47 No issue - There isn't any division

48 No issue - There isn't any arithmetic expression

49 No issue- There isn't any comparison

50 No issue - There isn't any throw-catch expression

51 No issue - There is only one type conversion and is explicit (line 57):

```
GenericValue userLogin = (GenericValue) session.getAttribute("userLogin");
```

## 3.16 Exceptions

52 No issue - The two methods throw appropriate exceptions, declared in their header

53 No issue - There isn't any throw-catch statement. The exception handled are declared in the headers and they are well-known and reliable java exceptions

## 3.17 Flow of Control

No `switch` statement are used in the class.
No loop statements are used in the class.

## 3.18 Files

No files are used in the class.

# 4 List of issues TraverseSubContentCacheTransform

## 4.1 Naming Conventions

1 No issue

2 No issue - No one-character variable is used

3 No issue

4 No issue - There isn't any interface

5 No issue

6 No issue

7 **Lines 52:**
`public static final String module = TraverseSubContentCacheTransform.class.getName;`.
**Solution:** rename to `MODULE`.

## 4.2 Indention

8 No issue - Always four spaces are used for indention

9 No issue - No tabs found in the code

## 4.3 Braces

10 No issue - Kernighan and Ritchie style is used consistently.

11 No issue

## 4.4 File Organization

12 No issue - All sections are separated by a blank line.

13
- Issue at line 51 - Length = 90
- Issue at line 68 - Length = 89
- Issue at line 84 - Length = 93
- Issue at line 86 - Length = 89
- Issue at line 89 - Length = 89
- Issue at line 90 - Length = 89
- Issue at line 91 - Length = 94
- Issue at line 93 - Length = 92
- Issue at line 95 - Length = 85

- Issue at line 96 - Length = 94
- Issue at line 104 - Length = 90
- Issue at line 107 - Length = 83
- Issue at line 112 - Length = 87
- Issue at line 113 - Length = 85
- Issue at line 157 - Length = 102
- Issue at line 159 - Length = 87
- Issue at line 165 - Length = 90
- Issue at line 166 - Length = 95
- Issue at line 182 - Length = 97
- Issue at line 192 - Length = 88
- Issue at line 195 - Length = 97
- Issue at line 209 - Length = 116
- Issue at line 215 - Length = 84
- Issue at line 226 - Length = 107
- Issue at line 231 - Length = 86
- Issue at line 237 - Length = 92

14
- Issue at line 53 - Length = 218 - Strings can be split in multiple lines.
- Issue at line 76 - Length = 97 - Function parameters can be split.
- Issue at line 94 - Length = 123 - Impractical to split.
- Issue at line 97 - Length = 147 - Line can be split after before `&&`
- Issue at line 102 - Length = 142 - Function parameters can be split.
- Issue at line 151 - Length = 131 - Function parameter can be split in a second line.
- Issue at line 210 - Length = 122 - Impractical to split.
- Issue at line 222 - Length = 128 - Function parameter can be split in a second line.

## 4.5 Wrapping Lines

15 No issue - No line breaks

16 No issue - No line breaks

17 No issue - Expressions are all aligned.

## 4.6 Comments

18 No JavaDoc.

19 No issue - There is no commented out code

## 4.7 Java Source Files

20 No issue

21 No issue

22 No issue

23 No JavaDoc.

## 4.8 Package and Import Statements

24 No issue

## 4.9 Class and Interface Declarations

25 a No issue

b No issue

c No issue - No implementation JavaDoc

d No issue

e No issue

f No issue

g No issue

26 No issue

27 No issue

## 4.10 Initialization and Declarations

28 No issue - **Lines 52-53:** `public static final String [] upSaveKeyNames [...] public static final String [] saveKeyNames = [...]`
*The keyword* `final` *for arrays does not reflect to the content of the array. There is no good reason to have a mutable object as* `public`.
**Solution:** Make this member `protected`.

29 No issue

30 No issue

31 No issue

32 No issue

33 No issue

## 4.11 Method Calls

34 No issue

35 No issue

36 No issue

## 4.12 Arrays

37 No issue - No arrays are used in the class behavior.

38 No issue - No arrays are used in the class behavior.

39 No issue - No arrays are used in the class behavior.

## 4.13 Object Comparison

40 No issue

## 4.14 Output Format

41 No issue

42 No issue

43 No issue

## 4.15 Computation, Comparisons and Assignments

44 **Line 153:** `if (globalNodeTrail.size() > 0) {`
`isEmpty()` *generally is O(1),* `size()` *is O(n).*
**Solution:** Use isEmpty() makes the code more readable and can be more performant.

45 No issue

46 No issue

47 No issue

48 No issue

49 No issue

50 No issue

51 No issue

## 4.16 Exceptions

52 **Line 235:** `boolean bEquals = contentIdStart.equals(contentIdEnd);`
*NullPointerException might be thrown as 'contentIdStart' is nullable here.*
**Solution:** Check the `null` value before de-reference `contentIdStart`.

53 **Lines 98-105:** Catch `GeneralException` (org.ofbiz.base.util.GeneralException) and retrow a `RuntimeException`.
*Using such generic exceptions as Error, RuntimeException, Throwable, and Exception prevents calling methods from handling true, system-generated exceptions differently than application-generated errors.*
**Solution:** Define an application-specific exception.

## 4.17 Flow of Control

54 No issue - No `switch` statement are used in the class.

55 No issue - No `switch` statement are used in the class.

56 No issue - No loop statements are used in the class.

## 4.18 Files

57 No issue - No files are used in the class behavior.

58 No issue - No files are used in the class behavior.

59 No issue - No files are used in the class behavior.

60 No issue - No files are used in the class behavior.

# 5 Appendix

## 5.1 Used software

1. **TeXstudio:** `http://www.texstudio.org/` to redact this document in LaTeX format.
2. **SonarQube:** `http://www.sonarqube.org/` open source platform for continuous inspection of code quality.
3. **CheckStyle 7.3:** `http://checkstyle.sourceforge.net/` automated code review tool.

## 5.2 Source Code

## 5.3 OFBizSolrRedirectServlet

```
/*******************************************************************************
 * 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.
 *******************************************************************************/
package org.apache.ofbiz.solr.webapp;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.solr.servlet.RedirectServlet;
import org.apache.ofbiz.base.util.UtilValidate;
import org.apache.ofbiz.entity.GenericValue;
import org.apache.ofbiz.security.Security;
import org.apache.ofbiz.webapp.control.LoginWorker;

/**
 * OFBizSolrRedirectServlet.java - Master servlet for the ofbiz-solr application.
 */
@SuppressWarnings("serial")
public class OFBizSolrRedirectServlet extends RedirectServlet {

    public static final String module = OFBizSolrRedirectServlet.class.getName();

    /**
     * @see
         javax.servlet.http.HttpServlet#doGet(javax.servlet.http.HttpServletRequest,
         javax.servlet.http.HttpServletResponse)
     */
    @Override
    public void DoGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
        boolean isForwarded = forwardUrl(request, response);
        if (isForwarded) {
            return;
        }

        super.doGet(request, response);
    }

    protected static boolean forwardUrl(HttpServletRequest request,
         HttpServletResponse response) throws IOException {
        HttpSession session = request.getSession();
        GenericValue userLogin = (GenericValue) session.getAttribute("userLogin");
        boolean forwardToLogin = false;
        if (UtilValidate.isEmpty(userLogin)) {
            forwardToLogin = true;
        } else {
            if (!LoginWorker.hasBasePermission(userLogin, request)) {
                forwardToLogin = true;
            }
        }

        if (forwardToLogin) {
            String contextPath = request.getContextPath();
            String uri = request.getRequestURI();
            if (UtilValidate.isNotEmpty(contextPath) && uri.startsWith(contextPath))
                 {
                uri = uri.replaceFirst(request.getContextPath(), "");
            }
            String servletPath = request.getServletPath();
            if (UtilValidate.isNotEmpty(servletPath) && uri.startsWith(servletPath)) {
                uri = uri.replaceFirst(servletPath, "");
            }
            response.sendRedirect(contextPath + "/control/checkLogin" + uri);
            return true;
        }

        return false;
    }
}
```

### 5.4 TraverseSubContentCacheTransform

```
/*******************************************************************************
 * 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.
 *******************************************************************************/
package org.apache.ofbiz.content.webapp.ftl;

import java.io.IOException;
import java.io.Writer;
import java.sql.Timestamp;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import org.apache.ofbiz.base.util.GeneralException;
import org.apache.ofbiz.base.util.UtilDateTime;
import org.apache.ofbiz.base.util.UtilGenerics;
import org.apache.ofbiz.base.util.UtilValidate;
import org.apache.ofbiz.base.util.template.FreeMarkerWorker;
import org.apache.ofbiz.content.content.ContentWorker;
import org.apache.ofbiz.entity.Delegator;
import org.apache.ofbiz.entity.GenericValue;
import org.apache.ofbiz.webapp.ftl.LoopWriter;

import freemarker.core.Environment;
import freemarker.template.TemplateModelException;
import freemarker.template.TemplateTransformModel;
import freemarker.template.TransformControl;

/**
 * TraverseSubContentCacheTransform - Freemarker Transform for URLs (links)
 */
```

```
public class TraverseSubContentCacheTransform implements TemplateTransformModel {

    public static final String module =
        TraverseSubContentCacheTransform.class.getName();
    public static final String [] upSaveKeyNames = {"globalNodeTrail"};
    public static final String [] saveKeyNames = {"contentId", "subContentId",
        "subDataResourceTypeId", "mimeTypeId", "whenMap", "locale",
        "wrapTemplateId", "encloseWrapText", "nullThruDatesOnly",
        "globalNodeTrail"};

    /**
     * @deprecated use FreeMarkerWorker.getWrappedObject()
     * A wrapper for the FreeMarkerWorker version.
     */
    @Deprecated
    public static Object getWrappedObject(String varName, Environment env) {
        return FreeMarkerWorker.getWrappedObject(varName, env);
    }

    /**
     * @deprecated use FreeMarkerWorker.getArg()
     */
    @Deprecated
    public static String getArg(Map<String, Object> args, String key, Environment
        env) {
        return FreeMarkerWorker.getArg(args, key, env);
    }

    /**
     * @deprecated use FreeMarkerWorker.getArg()
     */
    @Deprecated
    public static String getArg(Map<String, Object> args, String key, Map<String,
        Object> ctx) {
        return FreeMarkerWorker.getArg(args, key, ctx);
    }

    @SuppressWarnings("unchecked")
    public Writer getWriter(final Writer out, Map args) {
        final StringBuilder buf = new StringBuilder();
        final Environment env = Environment.getCurrentEnvironment();
        final Map<String, Object> templateRoot =
            FreeMarkerWorker.createEnvironmentMap(env);
        final Map<String, Object> savedValuesUp = new HashMap<String,
            Object>();
        FreeMarkerWorker.saveContextValues(templateRoot, upSaveKeyNames,
            savedValuesUp);
        final Map<String, Object> savedValues = new HashMap<String, Object>();
        FreeMarkerWorker.overrideWithArgs(templateRoot, args);
        String startContentAssocTypeId =
            (String)templateRoot.get("contentAssocTypeId");
        final Delegator delegator = FreeMarkerWorker.getWrappedObject("delegator",
            env);
        final HttpServletRequest request =
            FreeMarkerWorker.getWrappedObject("request", env);
        FreeMarkerWorker.getSiteParameters(request, templateRoot);
        final GenericValue userLogin =
            FreeMarkerWorker.getWrappedObject("userLogin", env);
        List<Map<String, ? extends Object>> globalNodeTrail =
            UtilGenerics.checkList(templateRoot.get("globalNodeTrail"));
        String strNullThruDatesOnly =
            (String)templateRoot.get("nullThruDatesOnly");
        String contentAssocPredicateId =
            (String)templateRoot.get("contentAssocPredicateId");
        Boolean nullThruDatesOnly = (strNullThruDatesOnly != null &&
            strNullThruDatesOnly.equalsIgnoreCase("true")) ? Boolean.TRUE
            :Boolean.FALSE;
        try {
            // getCurrentContent puts the "current" node on the end of
                globalNodeTrail.
            // It may have already been there, but getCurrentContent will compare its
                contentId
            // to values in templateRoot.
            ContentWorker.getCurrentContent(delegator, globalNodeTrail, userLogin,
                templateRoot, nullThruDatesOnly, contentAssocPredicateId);
        } catch (GeneralException e) {
            throw new RuntimeException("Error getting current content. " +
                e.toString());
        }

        final Map<String, Object> traverseContext = new HashMap<String,
            Object>();
        traverseContext.put("delegator", delegator);
        Map<String, Object> whenMap = new HashMap<String, Object>();
        whenMap.put("followWhen", templateRoot.get("followWhen"));
        whenMap.put("pickWhen", templateRoot.get("pickWhen"));
        whenMap.put("returnBeforePickWhen",
            templateRoot.get("returnBeforePickWhen"));
        whenMap.put("returnAfterPickWhen",
            templateRoot.get("returnAfterPickWhen"));
        traverseContext.put("whenMap", whenMap);
        env.setVariable("whenMap", FreeMarkerWorker.autoWrap(whenMap, env));
        String fromDateStr = (String)templateRoot.get("fromDateStr");
        String thruDateStr = (String)templateRoot.get("thruDateStr");
        Timestamp fromDate = null;
        if (UtilValidate.isNotEmpty(fromDateStr)) {
            fromDate = UtilDateTime.toTimestamp(fromDateStr);
        }
        traverseContext.put("fromDate", fromDate);
        Timestamp thruDate = null;
        if (UtilValidate.isNotEmpty(thruDateStr)) {
            thruDate = UtilDateTime.toTimestamp(thruDateStr);
        }
        traverseContext.put("thruDate", thruDate);
        traverseContext.put("contentAssocTypeId", startContentAssocTypeId);
        String direction = (String)templateRoot.get("direction");
        if (UtilValidate.isEmpty(direction)) {
            direction = "From";
        }
        traverseContext.put("direction", direction);

        return new LoopWriter(out) {

            @Override
            public void write(char cbuf[], int off, int len) {
                buf.append(cbuf, off, len);
            }

            @Override
            public void flush() throws IOException {
                out.flush();
            }

            @Override
            public int onStart() throws TemplateModelException, IOException {
                List<Map<String, ? extends Object>> nodeTrail = null;
                Map<String, Object> node = null;
                List<Map<String, ? extends Object>> globalNodeTrail =
                    UtilGenerics.checkList(templateRoot.get("globalNodeTrail"));
                if (globalNodeTrail.size() > 0) {
                    int sz = globalNodeTrail.size() ;
                    nodeTrail = new LinkedList<Map<String,? extends Object>>();
                    node = UtilGenerics.checkMap(globalNodeTrail.get(sz - 1));
                    Boolean checkedObj = (Boolean)node.get("checked");
                    Map<String, Object> whenMap =
                        UtilGenerics.checkMap(templateRoot.get("whenMap"));
                    if (checkedObj == null || !checkedObj.booleanValue()) {
                        ContentWorker.checkConditions(delegator, node, null,
                            whenMap);
                    }
                } else {
                    throw new IOException("Empty node trail entries");
                }

                Boolean isReturnBeforePickBool =
                    (Boolean)node.get("isReturnBeforePick");
                if (isReturnBeforePickBool != null &&
                    isReturnBeforePickBool.booleanValue()) {
                    return TransformControl.SKIP_BODY;
                }

                ContentWorker.selectKids(node, traverseContext);
                nodeTrail.add(node);
                traverseContext.put("nodeTrail", nodeTrail);
                Boolean isPickBool = (Boolean)node.get("isPick");
                Boolean isFollowBool = (Boolean)node.get("isFollow");
                boolean isPick = true;
                if ((isPickBool == null || !isPickBool.booleanValue())
                   && (isFollowBool != null && isFollowBool.booleanValue())) {
                    isPick = ContentWorker.traverseSubContent(traverseContext);
                }
                if (isPick) {
                    populateContext(traverseContext, templateRoot);
                    FreeMarkerWorker.saveContextValues(templateRoot,
                        saveKeyNames, savedValues);
                    return TransformControl.EVALUATE_BODY;
                } else {
                    return TransformControl.SKIP_BODY;
                }
            }

            @Override
            public int afterBody() throws TemplateModelException, IOException {
                FreeMarkerWorker.reloadValues(templateRoot, savedValues, env);
                boolean inProgress =
                    ContentWorker.traverseSubContent(traverseContext);
                if (inProgress) {
                    populateContext(traverseContext, templateRoot);
                    FreeMarkerWorker.saveContextValues(templateRoot,
                        saveKeyNames, savedValues);
                    return TransformControl.REPEAT_EVALUATION;
                } else {
                    return TransformControl.END_EVALUATION;
                }
            }

            @Override
            public void close() throws IOException {
                FreeMarkerWorker.reloadValues(templateRoot, savedValuesUp, env);
                String wrappedContent = buf.toString();
                out.write(wrappedContent);
            }

            public void populateContext(Map<String, Object> traverseContext,
                 Map<String, Object> templateContext) {
                List<Map<String, ? extends Object>> nodeTrail =
                    UtilGenerics.checkList(traverseContext.get("nodeTrail"));
                int sz = nodeTrail.size();
                Map<String, ? extends Object> node = nodeTrail.get(sz - 1);
```

```
                GenericValue content = (GenericValue)node.get("value");
                String contentId = (String)node.get("contentId");
                String contentAssocTypeId = (String)node.get("contentAssocTypeId");
                envWrap("contentAssocTypeId", contentAssocTypeId);
                envWrap("contentId", contentId);
                envWrap("content", content);
                String mapKey = (String)node.get("mapKey");
                envWrap("mapKey", mapKey);
                envWrap("subContentDataResourceView", null);
                List<Map<String, ? extends Object>> globalNodeTrail =
                     UtilGenerics.checkList(templateContext.get("nodeTrail"));
                String contentIdEnd = null;
                String contentIdStart = null;
                if (globalNodeTrail != null) {
                    Map<String, ? extends Object> ndEnd =
                         globalNodeTrail.get(globalNodeTrail.size() - 1);
                    contentIdEnd = (String)ndEnd.get("contentId");
                    Map<String, ? extends Object> ndStart = nodeTrail.get(0);
                    contentIdStart = (String)ndStart.get("contentId");
                } else {
                    globalNodeTrail = new LinkedList<Map<String,? extends
                         Object>>();
                }
                boolean bIdEnd = UtilValidate.isNotEmpty(contentIdEnd);
                boolean bIdStart = UtilValidate.isNotEmpty(contentIdStart);
                boolean bEquals = contentIdStart.equals(contentIdEnd);
                if (bIdEnd && bIdStart && bEquals) {
                    List<Map<String, ? extends Object>> subList =
                         nodeTrail.subList(1, sz);
                    globalNodeTrail.addAll(subList);
                } else {
                    globalNodeTrail.addAll(nodeTrail);
                }
                int indentSz = globalNodeTrail.size();
                envWrap("indent", Integer.valueOf(indentSz));
                String trailCsv = ContentWorker.nodeTrailToCsv(globalNodeTrail);
                envWrap("nodeTrailCsv", trailCsv);
                envWrap("globalNodeTrail", globalNodeTrail);
            }

            public void envWrap(String varName, Object obj) {
                templateRoot.put(varName, obj);
                env.setVariable(varName, FreeMarkerWorker.autoWrap(obj, env));
            }
        };
    }
}
```

## 5.5 Time effort

The estimated time spent by us to redact this document:

| | |
|---|---|
| Andrea Millimaggi | 15h |
| Matteo Michele Piazzolla | 15h |