Politecnico di Milano

A.A. 2016-2017

Software Engineering 2:

OFBiz

Code **I**nspection

Lorenzo Frigerio (mat. 879203)

Martina Perfetto (mat. 808869)

Ivan Vigorito (mat. 879204)

5rd February 2017

Version 1.0

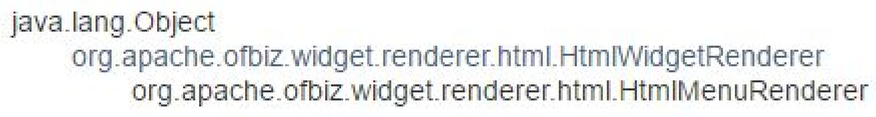
Contents

1. Inspected Classes

In Java, the idea of namespace is embodied in Java packages. The only class inspected is HMTLMenuRenderer that is part of org.apache.ofbiz.widget.renderer.html package.

2. Functional role

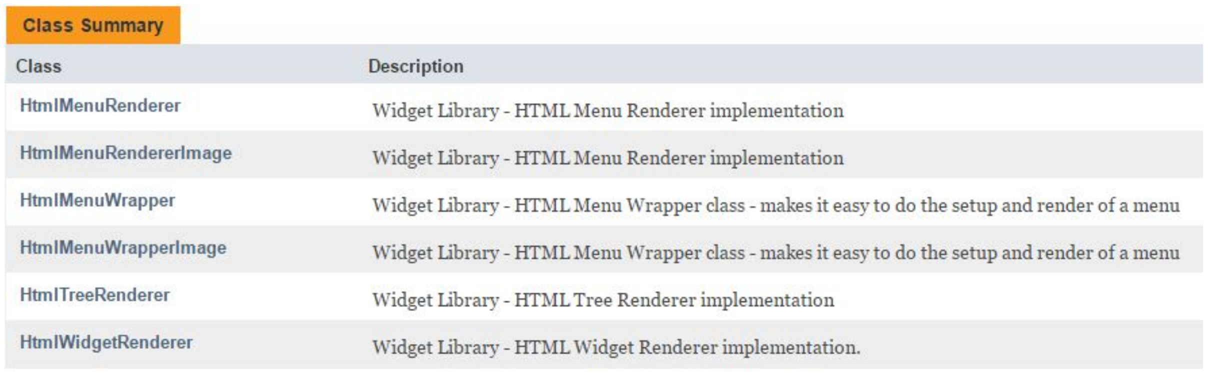
OfBiz gives to the developers the possibility to build an highly personalized user interface. The presentation layer of the application is based on the concept of "screen" that involves the use of the ofBiz widget technology.



```
java.lang.Object
    org.apache.ofbiz.widget.renderer.html.HtmlWidgetRenderer
        org.apache.ofbiz.widget.renderer.html.HtmlMenuRenderer
```

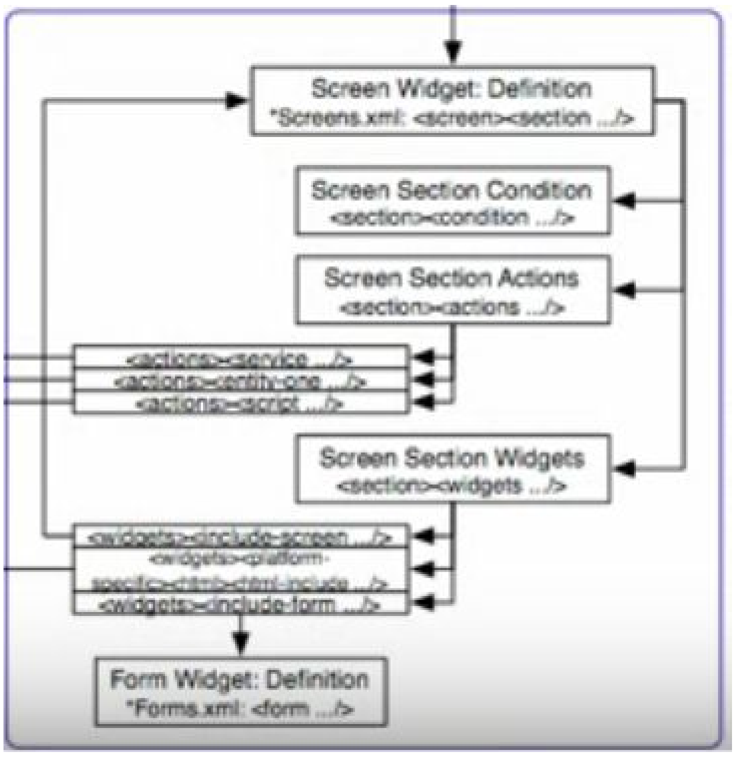
In order to exploit better this service, the platform allows the creation of personalized menu that once developed are stored in an xml file. The role of HTMLMenuRenderer is to convert the information encoded in the xml file in a string html ready to be displayed. This conversion begins with the loading of the xml file in a read only java class (ModelMenu or ModelMenuItem) that the HTMLMenuRenderer encode in a string.

In conclusion, the class has the main role of appending the html string representing a menu end its items to the writer element that will be displayed in the presentation servlets. Moreover, it is able to manage the conversion in html elements of images, possibly added to the menus, links and tooltips.



Class Summary

Class	Description
HtmlMenuRenderer	Widget Library - HTML Menu Renderer implementation
HtmlMenuRendererImage	Widget Library - HTML Menu Renderer implementation
HtmlMenuWrapper	Widget Library - HTML Menu Wrapper class - makes it easy to do the setup and render of a menu
HtmlMenuWrapperImage	Widget Library - HTML Menu Wrapper class - makes it easy to do the setup and render of a menu
HtmlTreeRenderer	Widget Library - HTML Tree Renderer implementation
HtmlWidgetRenderer	Widget Library - HTML Widget Renderer implementation.


Screen Widget: Definition
*Screens.xml: <screen><section .../>
Screen Section Condition
<section><condition .../>
Screen Section Actions
<section><actions .../>
<actions><service .../>
<actions><entity-one .../>
<actions><script .../>
Screen Section Widgets
<section><widgets .../>
<widgets><include-screen .../>
<widgets><platform-specific><html><html-include .../>
<widgets><include-form .../>
Form Widget: Definition
*Forms.xml: <form .../>

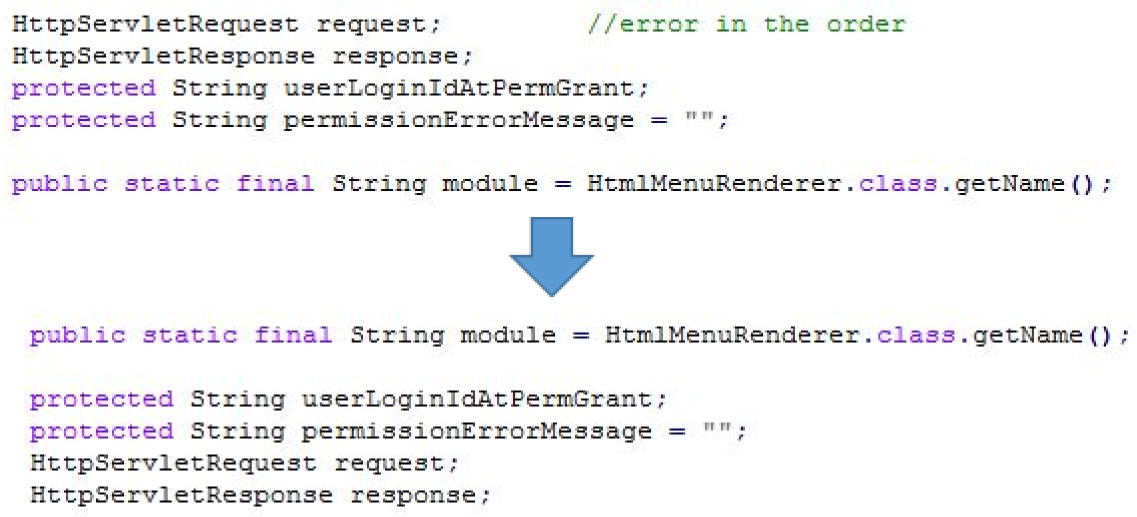
The definition and the role of screens and widget are extracted from:

https://cwiki.apache.org/confluence/display/OFBIZ/Understanding+the+OFBiz+Widget+Toolkit

3. List of issues

public class HTMLMenuRenderer (Line 50)

#25 Line 52: The order of the definition of the attribute of the class is wrong



```
HttpServletRequest request;          //error in the order
HttpServletResponse response;
protected String userLoginIdAtPermGrant;
protected String permissionErrorMessage = "";

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

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

protected String userLoginIdAtPermGrant;
protected String permissionErrorMessage = "";
HttpServletRequest request;
HttpServletResponse response;
```

public static final String module (Line 57)

General Overview: This parameter should be declared before line 52

public void appendOfbizUrl (Line 66)

#14 Line 69: This line length exceeds 120 characters and there is a commented block of code, but with no reason for being commented nor a date when it can be removed from the source file if determined it is no longer needed.

#19 Line 73: (from line 73 to 75) these lines are an else block containing only a comment, so ‘else’ is useless. Moreover, the comment is a commented block of code, but with no reason for being commented nor a date when it can be removed from the source file if determined it is no longer needed.

#19 Line 79: There is a commented block of code, but with no reason for being commented nor a date when it can be removed from the source file if determined it is no longer needed.

#19 Line 82: (from line 82 to 84) these lines are an if block containing only a comment, so ‘if’ is useless. Moreover, the comment is a commented block of code, but with no reason for being commented nor a date when it can be removed from the source file if determined it is no longer needed.

#19 Line 88: (from line 88 to 90) these lines are an if block containing only a comment, so ‘if’ is useless. Moreover, the comment is a commented block of code, but with no reason for being commented nor a date when it can be removed from the source file if determined it is no longer needed.

public void appendContentUrl (Line 94)

#14 Line 97: This line length exceeds 120 characters and there is a commented block of code, but with no reason for being commented nor a date when it can be removed from the source file if determined it is no longer needed.

#19 Line 101: (from line 101 to 103) these lines are an else block containing only a comment, so ‘else’ is useless. Moreover, the comment is a commented block of code, but with no reason for being commented nor a date when it can be removed from the source file if determined it is no longer needed.

#19 Line 107: There is a commented block of code, but with no reason for being commented nor a date when it can be removed from the source file if determined it is no longer needed.

#19 Line 111: (from line 111 to 113) these lines are an if block containing only a comment, so ‘if’ is useless. Moreover, the comment is a commented block of code, but with no reason for being commented nor a date when it can be removed from the source file if determined it is no longer needed.

public void renderMenuItem (Line 144)

#19 Line 146: There is a commented block of code, but with no reason for being commented nor a date when it can be removed from the source file if determined it is no longer needed.

#19 Line 148: There is a commented block of code, but with no reason for being commented nor a date when it can be removed from the source file if determined it is no longer needed.

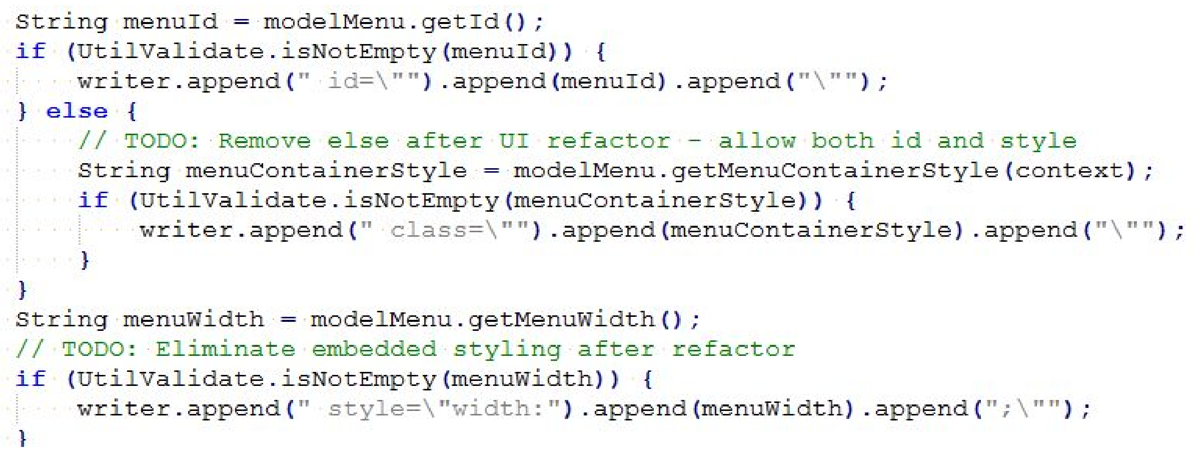
#11 Line 149: Curly braces missing here. To be consistent we should use the “Kernighan and Ritchie" style.

#19 Line 184: There is a commented block of code, but with no reason for being commented nor a date when it can be removed from the source file if determined it is no longer needed.

public void renderMenuOpen (Line 235)

General Overview: This method lacks a lot of information inside the JavaDoc documentation (such as parameters, exceptions and description). In this method there are several line of code which exceed the 80 characters.

Line: 245 and 252: There are two different TODO. As we can see from the code they still need to be completed and we supposed that UI refactor is not executed yet.



```
String menuId = modelMenu.getId();
if (UtilValidate.isNotEmpty(menuId)) {
    writer.append(" id=\"").append(menuId).append("\"");
} else {
    // TODO: Remove else after UI refactor - allow both id and style
    String menuContainerStyle = modelMenu.getMenuContainerStyle(context);
    if (UtilValidate.isNotEmpty(menuContainerStyle)) {
        writer.append(" class=\"").append(menuContainerStyle).append("\"");
    }
}
String menuWidth = modelMenu.getMenuWidth();
// TODO: Eliminate embedded styling after refactor
if (UtilValidate.isNotEmpty(menuWidth)) {
    writer.append(" style=\"width:").append(menuWidth).append(";\"");
}
```

public void renderMenuClose (Line 276)

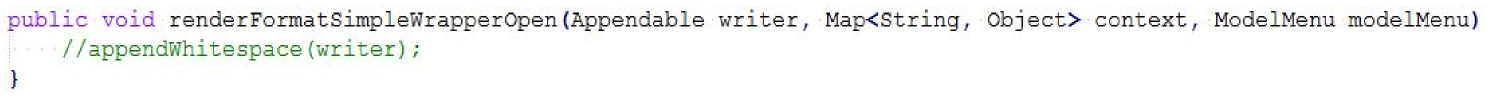
General Overview: This method lacks a lot of information inside the JavaDoc documentation (such as parameters, exceptions and description)

#33 Line 297: This variable declaration should be defined at the top of the method

(line 277)

#14 Line 302: This line is a comment with 168 characters, this is not a proper way to make it legible.

public void renderFormatSimpleWrapperOpen (Line 308)

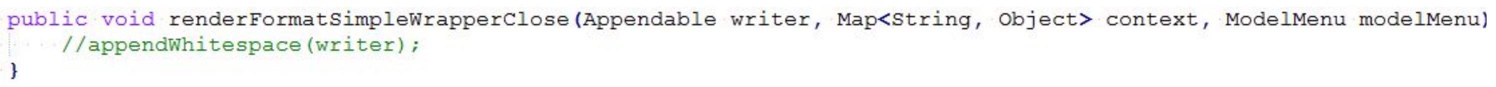


```
public void renderFormatSimpleWrapperOpen(Appendable writer, Map<String, Object> context, ModelMenu modelMenu)
    //appendWhitespace(writer);
}
```

General Overview: This method lacks a lot of information inside the JavaDoc documentation (such as parameters, exceptions and description).

#19 As we can see, this method is useless and incomplete. The body of the method contains only a commented line of code for a whitespace. According to that, this method can be deleted.

public void renderFormatSimpleWrapperClose (Line 312)



```
public void renderFormatSimpleWrapperClose(Appendable writer, Map<String, Object> context, ModelMenu modelMenu)
    //appendWhitespace(writer);
}
```

General Overview: This method lacks a lot of information inside the JavaDoc documentation (such as parameters, exceptions and description).

#19 As we can see, this method is useless and incomplete. The body of the method contains only a commented line of code for a whitespace. According to that, this method can be deleted.

public void setRequest (Line 316)

General Overview: No issues found

public void setResponse (Line 320)

General Overview: No issues found

public void setUserLoginIdAtPermGrant (Line 328)

General Overview: No issues found

public String getUserLoginIdAtPermGrant (Line 333)

General Overview: No issues found

public boolean isHideIfSelected (Line 337)

General Overview: This method lacks a lot of information inside the JavaDoc documentation (such as parameters, exceptions and description).

#14 Line 342: This line is a comment with 168 characters, this is not a proper way to make it legible.

#14 Line 343: This line of code has 143 characters; this line exceeds the max length of

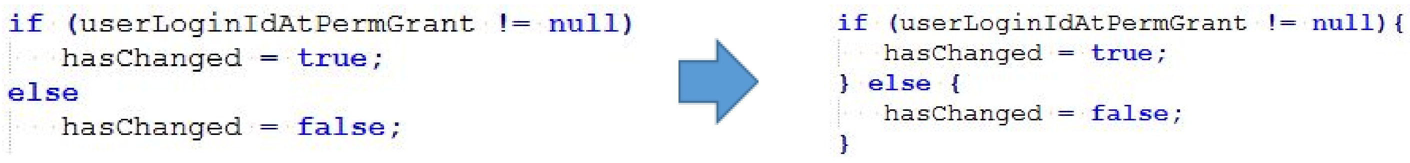
120 characters.

public boolean userLoginIdHasChanged (Line 347)

General Overview: This method lacks a lot of information inside the JavaDoc documentation (such as parameters, exceptions and description).

#13 Line 349: This line exceeds the length of 80 characters.

#11 Line 362: As we can see, the "Kernighan and Ritchi" braces convention is not respected in this case. On the left, is the real code and on the right is the right way to express this if-else.



```
if (userLoginIdAtPermGrant != null)
    hasChanged = true;
else
    hasChanged = false;
```

```
if (userLoginIdAtPermGrant != null){
    hasChanged = true;
} else {
    hasChanged = false;
}
```

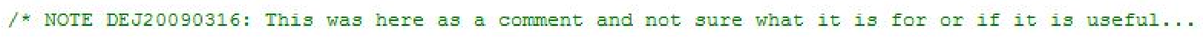
public String getTitle (Line 372)

General Overview: No issues found

public Void renderLink (Line 372)

#13 Lines 392-397-474-478: Lines should be wrapped onto the next line since it exceeds 80 characters but can be practically wrap to the next line.

#19 Line 499: unclear comment that says "not sure what it is for or if it is useful...", probably the code can be safely removed since the retrieval of the style for the element is already performed in the previous lines.

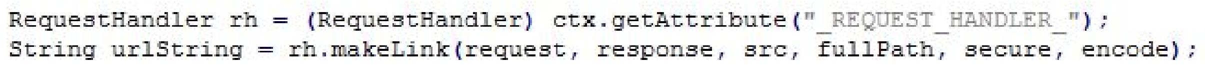


```
/* NOTE DEJ20090316: This was here as a comment and not sure what it is for or if it is useful...
```

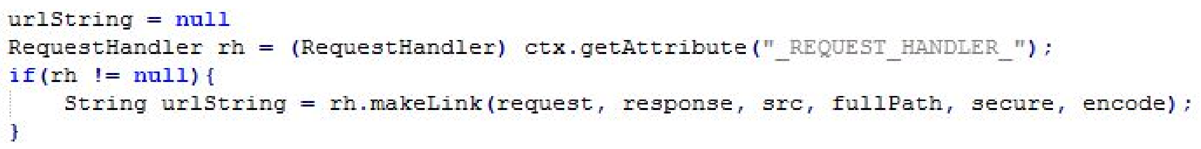
#33 Lines 428-441-447-486: variables declaration should be placed at the beginning of the block (line 394)

public Void renderImage (Line 372)

#31 Line 571: missing check (isNull) on the returned variable rh that is used in the following line. In fact, the attribute "_REQUEST_HANDLER_" may be missed in the context.

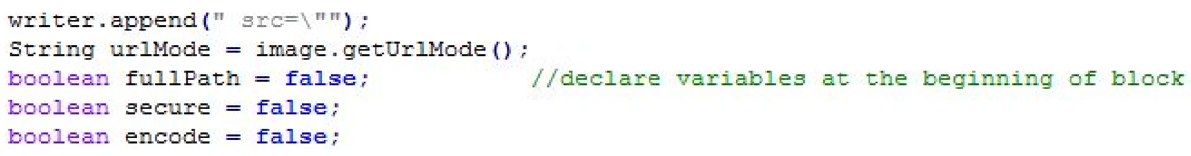


```
RequestHandler rh = (RequestHandler) ctx.getAttribute("_REQUEST_HANDLER_");
String urlString = rh.makeLink(request, response, src, fullPath, secure, encode);
```



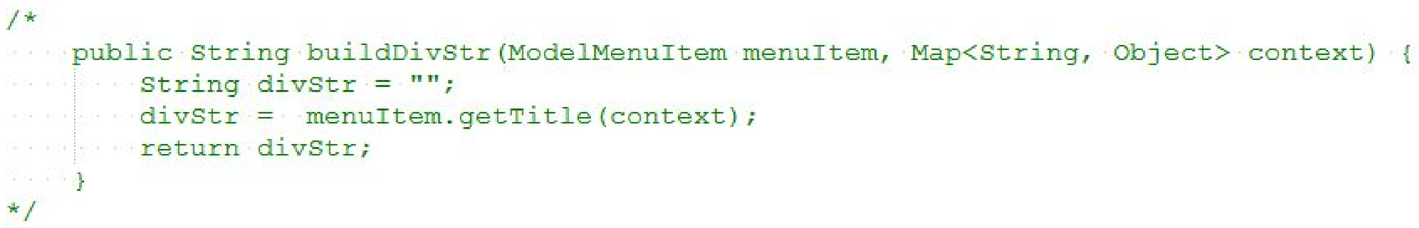
```
urlString = null
RequestHandler rh = (RequestHandler) ctx.getAttribute("_REQUEST_HANDLER_");
if(rh != null){
    String urlString = rh.makeLink(request, response, src, fullPath, secure, encode);
}
```

#33 Lines 535-541-547-553-559-563-565: all the declaration can be put together at the beginning of the block



```
writer.append(" src=\"");
String urlMode = image.getUrlMode();
boolean fullPath = false;          //declare variables at the beginning of block
boolean secure = false;
boolean encode = false;
```

4. Other problems highlighted



```
/*
    public String buildDivStr(ModelMenuItem menuItem, Map<String, Object> context) {
        String divStr = "";
        divStr =  menuItem.getTitle(context);
        return divStr;
    }
*/
```

General Overview: this method is not explained inside the JavaDoc. It is a method fully commented, with a cryptic name and from the name it is difficult to understand its functionality. The body of the method contains only a function which extract the “title” from the “Map<string, Object> context”.

5. Hours of work

Lorenzo Frigerio

23/01/17: 2h

26/01/17: 3h

03/02/17: 1h

Ivan Vigorito

23/01/17: 2h

01/02/17: 2h

03/02/17: 2h

Martina Perfetto

23/01/17: 2h

29/01/17: 1h

03/02/17: 3h