4.4.2 Custom Sonar Rules

This document provides information about Sonar and how to develop Sonar rules for use in KS development. It contains the following topics:

Sonar and Related Tools

Sonar is a framework for running a variety of checking rules and collecting their results. The scope of the KS code base makes it prohibitive for someone to manually inspect and resolve undesirable coding patterns. Sonar can help by identifying locations that need to be cleaned up and that cleanup can then be assigned to a developer as a single unit of work.

sonar.kuali.org is the sonarqube instance for the Kuali projects. There is a special Jenkins job set up using the Jenkin's Sonar Plugin that will run when the code is updated and will run the code through the processing rules and filters defined in the Sonar instance.

```
$ mvn sonar:sonar
```

PMD

PMD is a source code analyzer. It finds common programming flaws. Refer to the following PMD resources.

URL	Description
http://pmd.sourceforge.net/pmd-4.3/rules/index.html	PMD 4.3 Rules

XPath

XPath is a search mechanism for examining hierarchical data structures like the Javascript DOM and in the Sonar case the Abstract Syntax Tree of the code being analyzed. Refer to the fofllowing XPath resources.

URL	Description
http://pmd.sourceforge.net/pmd-5.0.4/xpathruletutorial.html	XPath Rule Tutorial
http://www.w3schools.com/xpath/xpath_syntax.asp	The syntax options for XPath
http://www.w3schools.com/xpath/xpath_functions.asp#string	The available functions in XPath
http://www.google.com/url?q=http%3A%2F%2Fpmd.sourceforge.net%2Fpmd-5.0.4%2Frules%2Findex.html%23JavaBeans&sa=D&sntz=1&usg=AFQjCNEILhLktaYXk2qeqgV0WUImC398mw	Examples from PMD v5.0.4 ***Note:*** *Sonar uses PMD 4.3.*

http://docs.codehaus.org/display/SONAR/Extending+Coding+Rules#ExtendingCodingRules-extendingRulesXPath	XPath Rule Tutorial from the Sonar project wiki

CheckStyle

CheckStyle is a tool to help developers stick to a consistent coding style. Use CheckStyle to apply regular expression matching to single, multiple, or comment lines. Refer to the following CheckStyle resources.

URL	Description
http://checkstyle.sourceforge.net/index.html	The home page of the CheckStyle project
http://checkstyle.sourceforge.net/availablechecks.html	List of available checks

The names are not directly queryable in the rule search screen, but you should be able to pick up on the key function of the check and search on that instead.

Developing XPath Rules

To make sure that the XPath String you develop is accurate, you need to test it. The best way to do this is to use the PMD Rule Designer. Even though the latest available version is v5.0.4, v4.3 is still supported by Sonar. However, the v5.0.4 editor is better, so you might want to install both. If you do this, make sure commands work properly in v4.3 before creating or updating an existing Sonar rule. Refer to the following resources for the PMD Rule Designer.

Version	Url	Descripotion
4.3	http://pmd.sourceforge.net/pmd-4.3/	Main Site
4.3	http://sourceforge.net/projects/pmd/files/pmd/4.3/	Download Site
5.0.4	http://pmd.sourceforge.net/pmd-5.0.4/	Main Site
5.0.4	http://sourceforge.net/projects/pmd/files/pmd/5.0.4/	Download Site

Determine XPath version 1.0 rule compatibility with PMD 4.3.

PMD XPath Functions

The following table describes some Common XPath functions:

Function Name	Usage	Description
starts-with	(Attribute, 'pattern')	Matches the string pattern to the value of the given attribute. For example, @Image.
@Attribute	@Attribute=somevalue	Attribute being one of the queryable Attributes, such as @Image.
ancestor	ancestor::<AST Node>	Match where the ancestor of the current location is the given AST node.
descendant	descendant::<AST Node>	Match where the descendant of the current location is the given AST node.

PMD Designer Overview

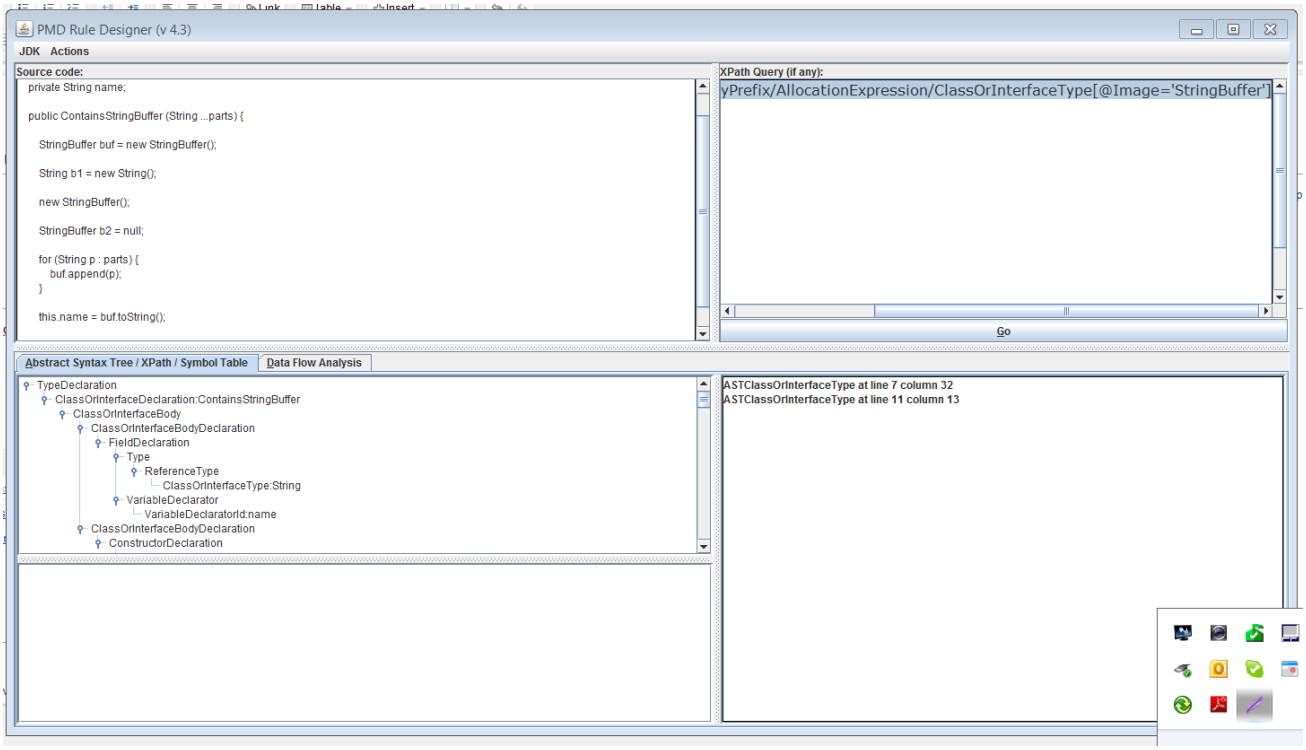
There are five primary areas in the application, as described in the following table:

Location	Description	Next Step
Top Left	Paste or Write in Code example.	Define XPath
Top Right	Paste or Write in the XPath.	Click **Go** Button

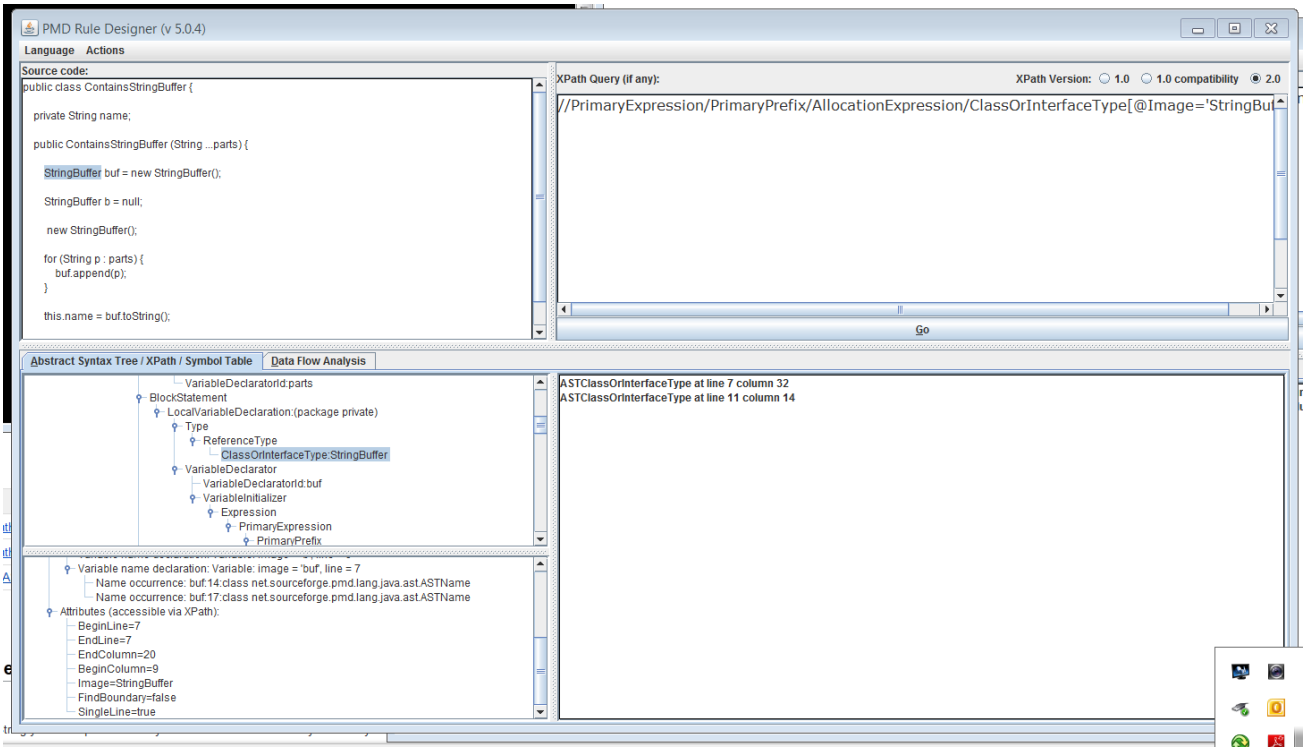
Bottom Left - Upper	Abstract Syntax Tree for the Code	N/A
Bottom Left - Lower	Node Details	N/A
Bottom Right	Match Details of XPath against the Code	N/A

As illustrated in the following screen shots, these are basically the same in both versions 4.3 and 5.0.4.

PMD Rule Designer v4.3



PMD Rule Designer v5.0.4



The main benefit of 5.0.4 is that when you select a node you can see which properties are **Accessible via XPath**. Note that not all nodes have queryable paths so you can expand the tree to find the nodes that contain the details you need. Clicking on the Node or a match will highlight it in the Source Code window.

Developing CheckStyle Rules that use Regular Expressions

Use the following for development of regular expressions for different kinds of CheckStyle rules:

Check Style Rule Name	Sonar Rule Name	Description
RegexpMultilineCheck	Regexp Multiline	A check for detecting matches across multiple lines. This check can be used when the regular expression can be span multiple lines.
RegexpSinglelineCheck	Regexp Singleline	A check for detecting single lines that match a supplied regular expression. Works with any file type. This check can be used to prototype checks and to find common bad practice such as calling *ex.printStacktrace()*, *System.out.println()*, *System.exit()*, etc.
RegexpSinglelineJavaCheck	Regexp Singleline Java	This class is a variation on *RegexpSingleline* for detecting single lines that match a supplied regular expression in Java files. It supports suppressing matches in Java comments.
TodoCommentCheck	Comment pattern matcher	This rule allows you to find any kind of pattern inside comments like TODO, NOPMD, etc., except NOSONAR.

Developing Java based PMD and CheckStyle Rules

Sonar provides plugins that connect it with the PMD and CheckStyle rule systems. It's possible to write new rules that can then be bundled using this mechanism. Java based rules require custom code to be inserted into the Sonar instance and it being restarted for the rules to appear in the user interface for final configuration.

The example plugins are located here: https://github.com/SonarSource/sonar-examples/tree/master/plugins.

Refer to the Unit Testing section below for information on PMD or CheckStyle testing.

Unit Testing Custom Regular Expression Based Rules

The build time of the KS Sonar CI job is over two hours so you will want to make sure that your rule syntax is correct. Both XPath and CheckStyle rules can be tested against a static source file to be sure they work before the rule is applied to the Sonar instance. For now only Regular Expression based rules are being tested, but it should be possible to test Java based rules. Use the following high level process:

Check Out the *ks-sonar* Maven Project

There are several modules to the *ks-sonar* project located at https://svn.kuali.org/repos/student/tools/ks-sonar/trunk. The unit tests are placed into the *ks-sonar-rule-test* artifact.

Test Code Locations

Test code locations are as follows:

Path	Package	Description
ks-sonar/ks-sonar-rule-test/src/test/java	org.kuali.student.checkstyle	Base location for the CheckStyle tests
ks-sonar/ks-sonar-rule-test/src/test/java	org.kuali.student.pmd	Base location for the PMD tests

Test Source Code Location

Test source code can be found in the following location:

Path	Description
ks-sonar/ks-sonar-rule-test/src/test/resources	Location of the Test Files

As the number of files being used in the test suite grows, subdirectories may be required.

Create a PMD Unit Test

Only XPath is supported. Create a new class that extends from *AbstractXPathTest* as illustrated below:

```
public class TestContainsStringBuffer extends AbstractXPathTest {

    @Test
    public void testContainsStringBufferXPath() throws FileNotFoundException,
PMDException {
           Report report = super.processXPath(

"//PrimaryExpression/PrimaryPrefix/AllocationExpression/ClassOrInterfaceType[@Image='S
tringBuffer']",
               "StringBuffer is not a preferred type", new
FileReader("src/test/resources/ContainsStringBuffer.java"),
               SourceType.JAVA_16);

           ReportTree vt = report.getViolationTree();

           Assert.assertEquals(2, vt.size());


    }
}
```

The *processXPath* helper will load the PMD XPathRule, inject the xpath expression, and apply it against the *ContainsStringBuffer* source file.

```
public class ContainsStringBuffer {

    public ContainsStringBuffer (String ...parts) {

        StringBuffer buf = new StringBuffer();

        new StringBuffer();

  StringBuffer buf2;

    }

}
```

This rule matches the *new StringBuffer()* part. It occurs twice in the file, which is why the assert specifies a size of 2.

To verify the rule worked as expected, you may want to include more than one source file and to check that no false positives are being detected. Depending on what happens when the XPath/Regular Expression is used in Sonar, the unit tests may need to be adapted to add in additional scenarios.

Creating a Sonar Rule using the Developed XPath

To create new rules, you will need the Sonar Credentials or create a JIRA with the details that will be needed to fill in the form and then assign it to someone with access to the Sonar instance.

New Rule Details

The following details are required for a new rule:

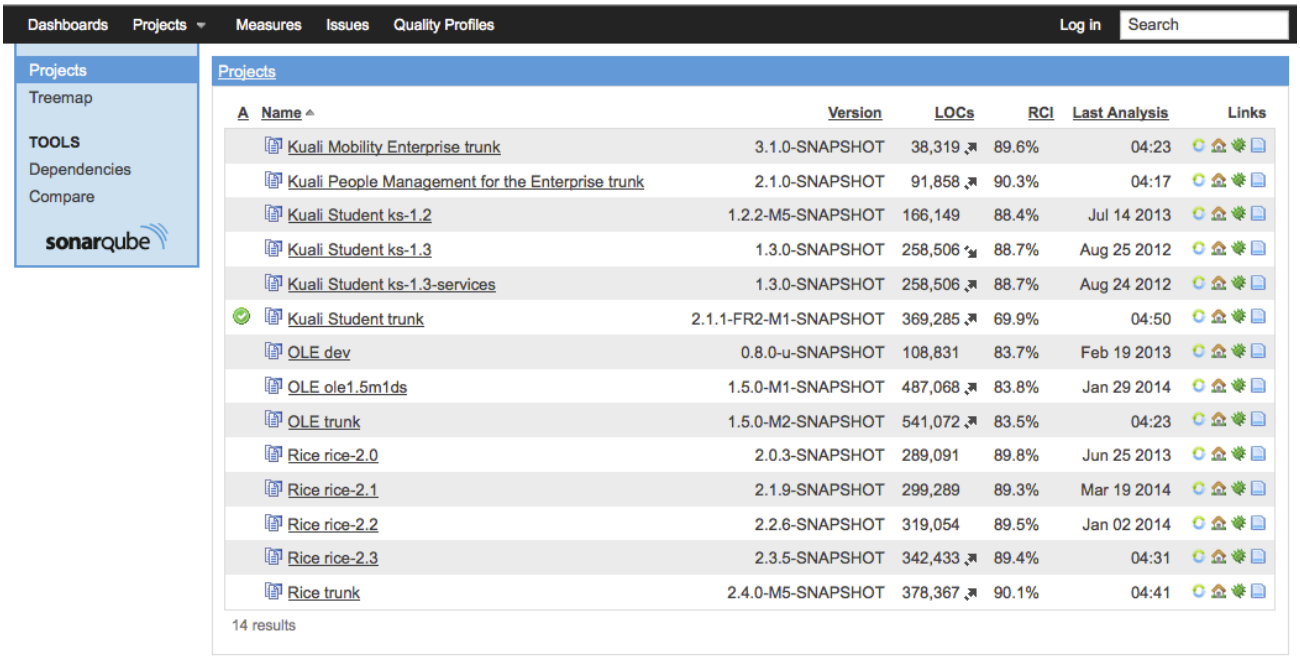
Name	Description
Name	The name of the rule.
Message	The short message to display when the rule is detected.
Description	A longer description that can include HTML markup.
Severity	Info, Minor, Major, Critical, or Blocker.
XPath	The XPath query string.

Other PMD rules may already exist that are close to what you need. In some cases, certain OR scenarios need to be implemented with several discrete rules that each exercise a piece of the problem.

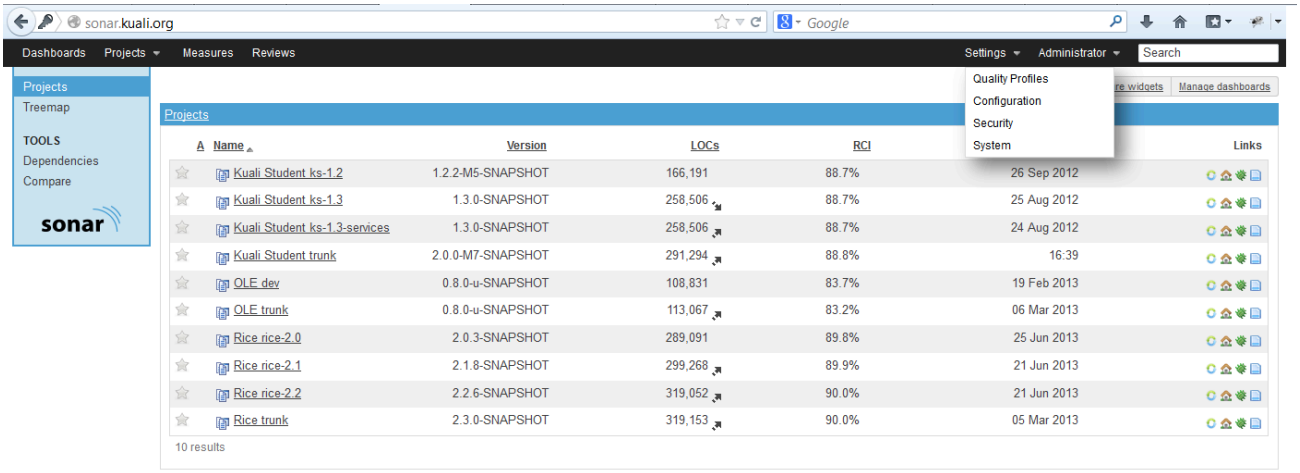
Create the Rule

Use the following basic procedure to create the rule.

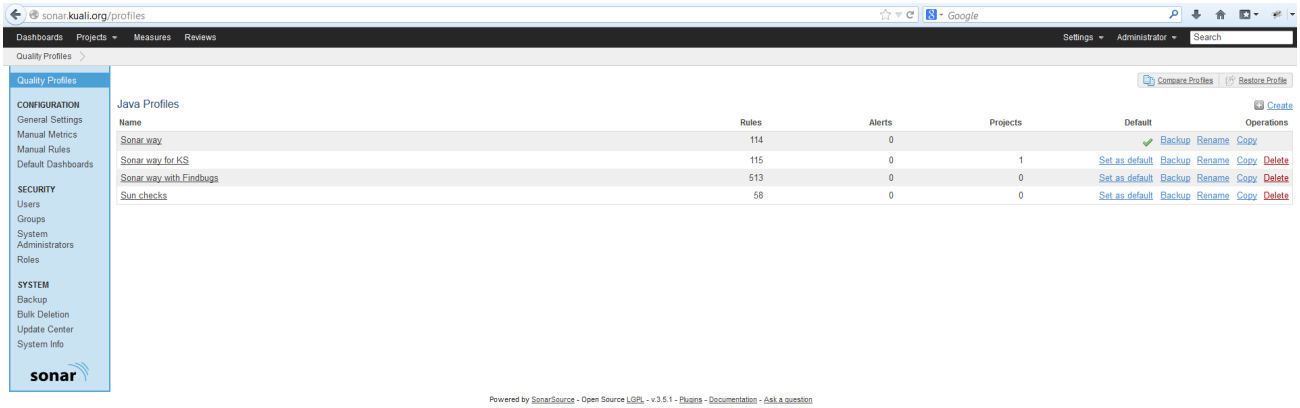
1. Access sonar.kuali.org. You see a screen similar to the following:



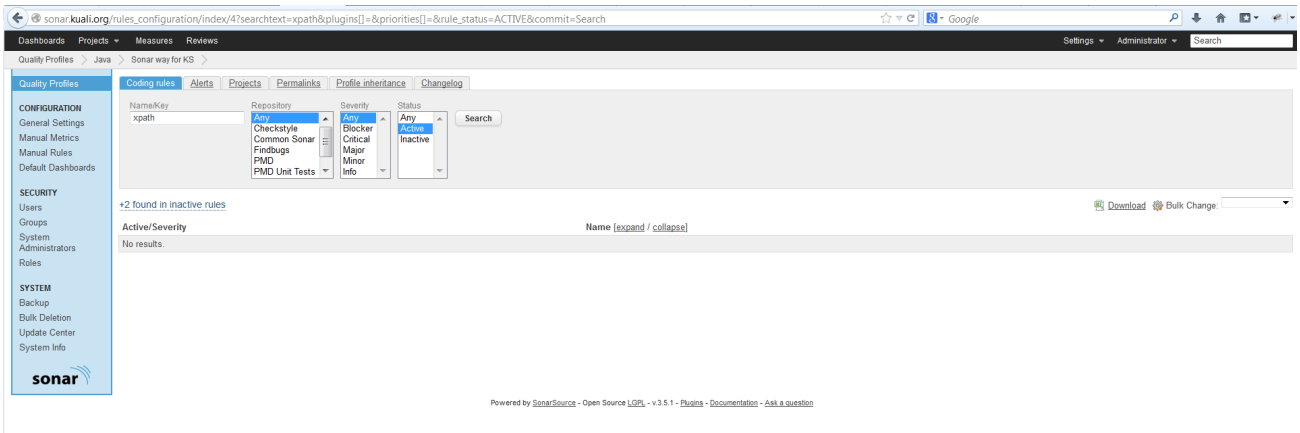
2. From the tool menu bar, select **Log in**.
3. Enter your login/password and click **Log in**. You see a Projects screen similar to the following:



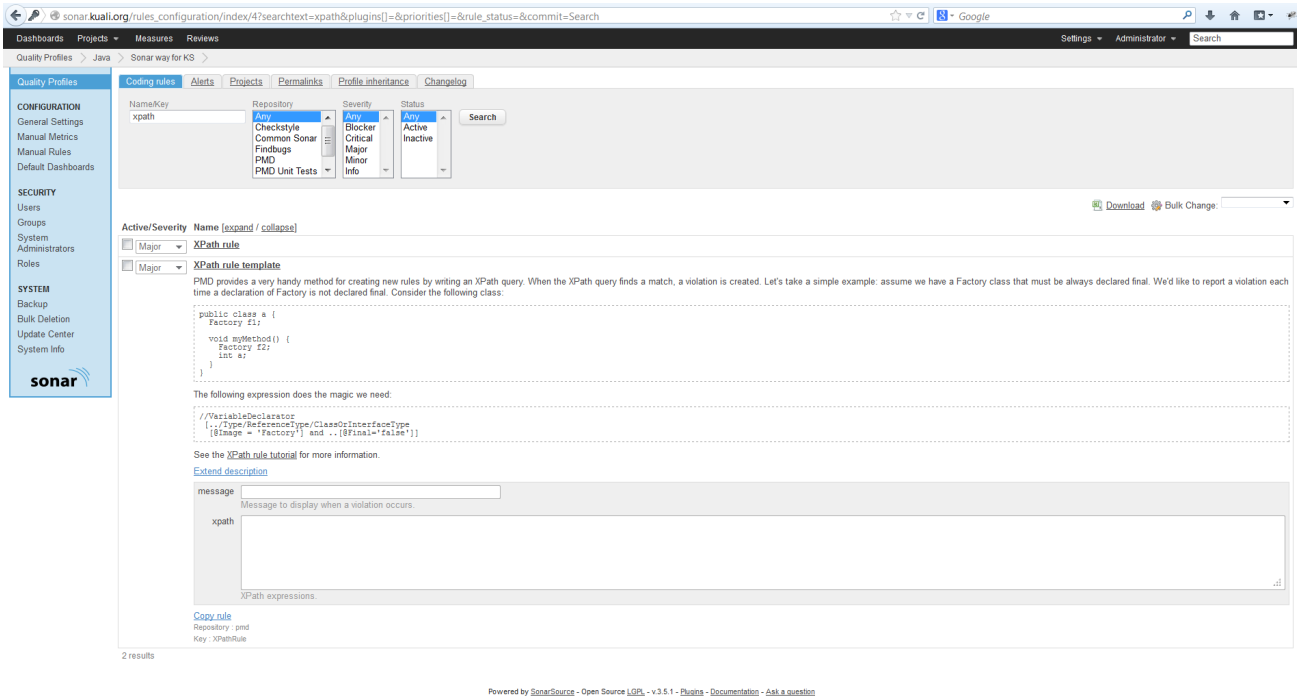
4. From the **Settings** drop-down select the **Quality Profiles** option. You see a Profiles screen similar to the following:



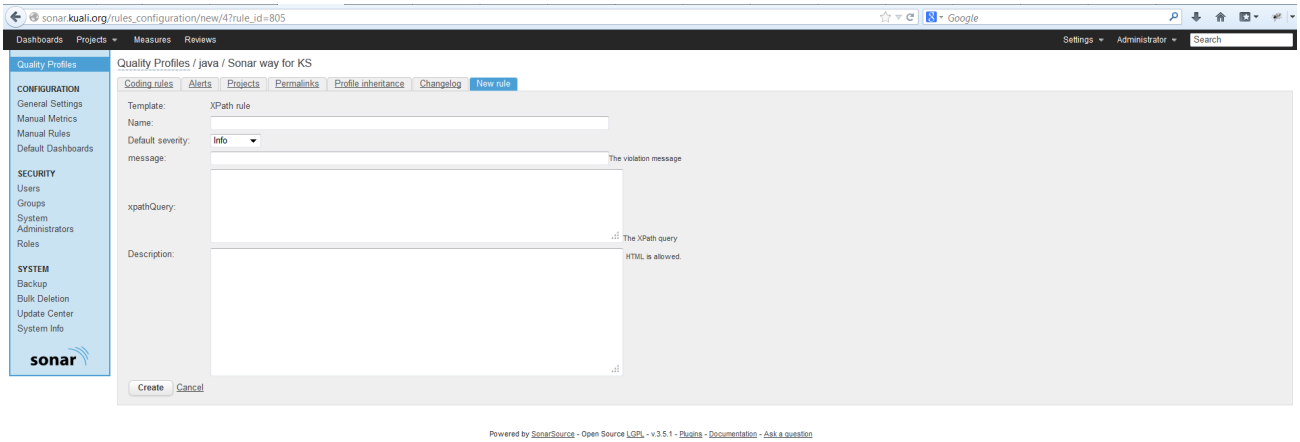
5. Select **Sonar way for KS**. This is the link in the **Name** column of the **Java Profiles** table.



6. From the Coding Rules tab, enter **xpath** into the **Name/Key** and click **Search**.



7. Click on the **+2 found in inactive rules** link to reveal the xpath template rules.



8. Click to expand the **XPath rule template**, then click the **Copy Rule** link
   ⚠ ***Don't insert your rule data into the template.***
9. Enter the Rule data into the fields and click **Create**.
10. Find the rule you just created and click the checkbox to enable it.

Verify Rule is Enabled

After creating and activating the rule, check these links to make sure Sonar knows to include it in the next Sonar build.

http://sonar.kuali.org/profiles/export?language=java&name=Sonar%2520way%2520for%2520KS

Search for either the Message or XPath Expression. The following example illustrates the AvoidStringBuffer Rule.

```
<rule>
    <repositoryKey>pmd</repositoryKey>
    <key>XPathRule_1372855392</key>
    <priority>CRITICAL</priority>
    <parameters>
        <parameter>
        <key>xpath</key>

<value>//PrimaryExpression/PrimaryPrefix/AllocationExpression/ClassOrInterfaceType[@Im
age='StringBuffer']</value>
        </parameter>
        <parameter>
        <key>message</key>
        <value>Avoid Using StringBuffer, use StringBuilder instead.</value>
        </parameter>
    </parameters>
</rule>
```

It will be used during the next build of the ks-enr-1.0-sonar jenkins job.