**ValidateMethodCondition**

*Functional Role*

This class is part of the minilang package. It is a subclass of MiniLangElements which, as said in Javadoc documentation is a superclass for all XML element models. In particular, it is a subclass of MethodOperation, an abstract class for Mini-language element models, which extends in turn MiniLangElement class. Moreover, it implements Conditional interface, an interface for all conditional element.

**public** **final** **class** ValidateMethodCondition **extends** MethodOperation **implements** Conditional

The constructor of the class takes fields (thanks to FlexibleMapAccessor), method name, class name and a list of sub operation, divided in subOps and elseSubOps, from the xml element, and set them as private final attributes.

The class has 2 main methods:

* CheckCondition (override from Conditional interface)

**public** **boolean** checkCondition(MethodContext methodContext) **throws** MiniLangException

Although Javadoc is not present neither in the interface nor in the class itself, this method seems to invoke the static method that this class aims to check, with fields taken as parameters, and returns a Boolean value which represents the checking of the method condition.

* Exec (override from MiniLangElements superclass)

**public** **boolean** exec(MethodContext methodContext) **throws** MiniLangException

As specified in Javadoc of the superclass this method generally “Executes the operation. Returns <code>true</code> if script execution should continue, or <code>false</code> if script execution should stop”. In this particular case, if the condition is checked executes all subOps in the method context (A container for the Mini-language script engine state), else run all elseSubOps.

Then there is another method that “Updates <code>aic</code> with this element's artifact information” (in particular with suboperation’s artifact information), and two methods for the printing most relevant information about the object, which are helpful for a complete understanding of the structure of the class itself.

Finally, another smaller class is defined in the same file.

**public** **static** **final** **class** ValidateMethodConditionFactory **extends** ConditionalFactory<ValidateMethodCondition> **implements** Factory<ValidateMethodCondition>

It is a static final factory class with the role of generating new ValidateMethodCondition objects.

*List of issues*

L44/L45

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

**private** **static** **final** Class<?>[] ***paramTypes*** = **new** Class<?>[] { String.**class** };

* Declaration of constant without respecting uppercase convention (7)

L18/L19

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**package** org.apache.ofbiz.minilang.method.conditional;

* No blank lines between beginning comments and package import (12)

L40

\* **@see** <a href="https://cwiki.apache.org/confluence/display/OFBADMIN/Mini-language+Reference#Mini-languageReference-{{%3Cifvalidatemethod%3E}}">Mini-language Reference</a>

* Line of 173 characters, exceed 120 (14)

L64

**this**.className = MiniLangValidate.checkAttribute(element.getAttribute("class"), "org.apache.ofbiz.base.util.UtilValidate");

* Line of 132 characters, exceed 120 (14)

L86

fieldVal = MiniLangUtil.convertType(fieldVal, String.**class**, methodContext.getLocale(), methodContext.getTimeZone(), **null**);

* Line of 139 characters, exceed 120 (14)

L161

**public** **static** **final** **class** ValidateMethodConditionFactory **extends** ConditionalFactory<ValidateMethodCondition> **implements** Factory<ValidateMethodCondition> {

* Line of 159 characters, exceed 120 (14)

L163

**public** ValidateMethodCondition createCondition(Element element, SimpleMethod simpleMethod) **throws** MiniLangException {

* Line of 126 characters, exceed 120 (14)

L168

**public** ValidateMethodCondition createMethodOperation(Element element, SimpleMethod simpleMethod) **throws** MiniLangException {

* Line of 132 characters, exceed 120 (14)

L131-L36

messageBuffer.append("validate-method[");

messageBuffer.append(className);

messageBuffer.append(".");

messageBuffer.append(methodName);

messageBuffer.append("(");

messageBuffer.append(**this**.fieldFma);

* Append can be grouped with ‘+’ operator between string (44)

L93-L96

Class<?> valClass = methodContext.getLoader().loadClass(className);

Method valMethod = valClass.getMethod(methodName, ***paramTypes***);

Boolean resultBool = (Boolean) valMethod.invoke(**null**, params);

**return** resultBool.booleanValue();

* Multiple useless variable declaration (44)

L65

Element childElement = UtilXml.firstChildElement(element);

* childElement variable is not declared at the beginning of the block to which it belongs (33)

L71

Element elseElement = UtilXml.firstChildElement(element, "else");

* elseElement variable is not declared at the beginning of the block to which it belongs (33)

L91

Object[] params = **new** Object[] { fieldVal };

* params variable is not declared at the beginning of the block to which it belongs (33)

GENERAL

* Code is not well commented, difficult to understand what methods do (18)
* A java source file contains two public classes: ValidateMethodCondition and ValidateMethodConditionFactory (20)
* Javadoc only on classes, not on methods (23)