$project.name

$DocumentTitle

Author: $Author

Revision: $Revisions.lastChild.name

|  |  |
| --- | --- |
|  |  |
|  | Date: $date.get(“MMMMM dd, yyyy”) |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Reason For Changes** | **Author** |
| #forrow($rev in $Revisions)$rev.name | $rev.date | $rev.description | $rev.author#endrow |
|  |  |  |  |

**Table of Contents**

Microsoft Word users please click here and press F9 to create Table of Contents.

OpenOffice.org users please remove this text and select Insert Table of Content from menu.

**Table of Figures**

Microsoft Word users please click here and press F9 to create Table of Contents.

OpenOffice.org users please remove this text and select Insert Table of Content from menu.

# Introduction

## Purpose

$Purpose

## Scope

$Scope

## Overview

$Overview

#set($level = 0)

#macro(increaseIndent)

#set($level = $level + 1)

#if($level > 6)

#set($h = 6)

#else

#set($h = $level)

#end

#end

#macro(decreaseIndent)

#set($level = $level - 1)

#if($level < 1)

#set($h = 1)

#else

#set($h = $level)

#end

#end

#macro(writeHeaderText $text)

#if($h == 1)

# $text

#elseif($h == 2)

## $text

#elseif($h == 3)

### $text

#elseif($h == 4)

#### $text

#elseif($h == 5)

##### $text

#else

###### $text

#end

#end

#macro(writeBookmarkHeader $object $text)

#if($h == 1)

# $bookmark.create($object.ID, $text)

#elseif($h == 2)

## $bookmark.create($object.ID, $text)

#elseif($h == 3)

### $bookmark.create($object.ID, $text)

#elseif($h == 4)

#### $bookmark.create($object.ID, $text)

#elseif($h == 5)

##### $bookmark.create($object.ID, $text)

#else

###### $bookmark.create($object.ID, $text)

#end

#end

#set($printInterface = $array.createArray())

#set($printClass = $array.createArray())

#set($printEnum = $array.createArray())

#macro(createInterfaceContent $interface)

#increaseIndent()

#writeBookmarkHeader($interface, $interface.HumanName)

#if($interface.documentation != “”)

$interface.documentation

#end

#set($superInterface = $array.createArray())

#foreach($generalization in $interface.generalization)

#set($tmp = $superInterface.add($generalization.general))

#end

#set($size = $superInterface.size())

#increaseIndent()

#if($superInterface.size() >0)

#writeHeaderText(“SuperInterfaces”)

#foreach($interface in $sorter.humanSort($superInterface))$bookmark.open($interface.ID, $interface.name)#if($size != $velocityCount), #end#end

#end

#set($knownImplementing = $array.createArray())

#foreach($supplier in $interface.supplierDependency)

#if($supplier.hasImplementingClassifier())

#set($tmp = $knownImplementing.add($supplier.implementingClassifier))

#end

#end

#set($size = $knownImplementing.size())

#if($knownImplementing.size() > 0)

#writeHeaderText(“Known Implementing Classes”)

#foreach($class in $sorter.humanSort($knownImplementing))$bookmark.open($class.ID, $class.name)#if($size != $velocityCount), #end#end

#end

package $report.getPackageQualifiedName($interface.owner, $QualifiedNameSeparator.toString())

$interface.visibility interface $interface.name

#set($size = $superInterface.size())

#set($i = 1)

#if($superInterface.size() > 0)

extends #foreach($interface in $sorter.humanSort($superInterface))$interface.name#if($i != $size), #end

#set($i = $i + 1)

#end

#end

#set($interfaceAtt = $array.createArray())

#foreach($a in $interface.ownedAttribute)

#if(!$a.association)

#set($tmp = $interfaceAtt.add($a))

#end

#end

#if($interfaceAtt.size() > 0)

#writeHeaderText(“Attributes”)

#foreach($att in $sorter.humanSort($interfaceAtt))

#printAttr($att)

#if($att.documentation != “”)

$att.documentation

#end

#end

#end

#set($interfaceOper = $interface.ownedOperation)

#if($interfaceOper.size() > 0)

#writeHeaderText(“Operations”)

#foreach($oper in $sorter.humanSort($interfaceOper))

#printOper($oper)

#if($oper.documentation != “”)

$oper.documentation

#end

#end

#end

#set($associationLists = $array.createArray())

#foreach($attribute in $interface.ownedAttribute)

#if($attribute.association)

#set($tmp = $associationLists.add($attribute.association))

#end

#end

#if($associationLists.size() > 0)

#writeHeaderText(“Associations”)

#foreach($attribute in $interface.ownedAttribute)

#if($attribute.association)

#set($association = $attribute.association)

#printAsso($attribute, $association, $interface)

#if($memberEnd.type.documentation != “”)

$memberEnd.type.documentation

#end

#end

#end

#end

#set($printDiagram = $array.createArray())

#if($interface.protocol)

#set($protocol = $interface.protocol)

#if($protocol.ownedDiagram.size() > 0)

#foreach($diagram in $protocol.ownedDiagram)

#if(!$printDiagram.contains($diagram))

#set($tmp = $printDiagram.add($diagram))

$diagram.image

1. $diagram.name

#if($diagram.documentation != “”)

$diagram.documentation

#end

#end

#end

#end

#end

#if($interface.hasHyperlinks())

#foreach($hyperlink in $interface.hyperlinks)

#if($hyperlink.getElement().elementType == “diagram”)

#if(!$printDiagram.contains($hyperlink.getElement()))

#set($tmp = $printDiagram.add($hyperlink.getElement()))

$hyperlink.getElement().image

1. $hyperlink.getElement().name

#if($hyperlink.getElement().documentation != “”)

$hyperlink.getElement().documentation

#end

#end

#end

#end

#end

#end

#macro(createClassContent $class)

#increaseIndent()

#writeBookmarkHeader($class, $class.HumanName)

#if($class.documentation != “”)

$class.documentation

#end

#set($printDiagram = $array.createArray())

#if($class.ownedDiagram.size() > 0)

#foreach($diagram in $class.ownedDiagram)

#if(!$printDiagram.contains($diagram))

#set($tmp = $printDiagram.add($diagram))

$diagram.image

1. $diagram.name

#if($diagram.documentation != “”)

$diagram.documentation

#end

#end

#end

#end

#if($class.elementType != “association”)

#set($implementInterface = $class.realizedInterface)

#set($size = $implementInterface.size())

#increaseIndent()

#if($implementInterface.size() > 0)

#writeHeaderText(“Implemented Interface”)

#foreach($interface in $sorter.humanSort($implementInterface))$bookmark.open($interface.ID, $interface.name)#if($size != $velocityCount), #end#end

#end

#end

#set($baseClassifier = $class.baseClassifier)

#set($size = $baseClassifier.size())

#if($baseClassifier.size() > 0)

#writeHeaderText(“Direct Supertypes”)

#foreach($class in $sorter.humanSort($baseClassifier))$bookmark.open($class.ID, $class.name)#if($size != $velocityCount), #end#end

#end

**package** $report.getPackageQualifiedName($class.owner, $QualifiedNameSeparator.toString())

#if($class.isAbstract)

#set($abstract = “ abstract class”)

#else

#set($abstract = “ class”)

#end

#set($size = $baseClassifier.size())

#set($i = 1)

#if($baseClassifier.size() > 0)

**extends** #foreach($class in $baseClassifier) $bookmark.open($class.ID, $class.name)#if($size != $i), #end#set($i = $i + 1)#end

#end

#set($size = $implementInterface.size())

#set($i = 1)

#if($implementInterface.size() > 0)

implements #foreach($interface in $implementInterface)$interface.name#if($i != $size), #end

#set($i = $i + 1)

#end

#end

#if($class.elementType==”association” || $class.elementType==”associationclass”)

#increaseIndent()

#writeHeaderText(“Association Ends”)

#foreach($end in $class.getMemberEnd())

$report.getIconFor($class) $end.name#if($end.type) : $bookmark.open($end.type.ID, $end.type.name)#end#if($end.multiplicity != “”) [$end.multiplicity]#end #if($subs.size()>0) *Subsets*: #foreach($sub in $subs)$sub.getName():$bookmark.open($sub.type.ID, $sub.type.name) #end #end#if($redefs.size()>0) *Redefines*: #foreach($sub in $redefs)$sub.getName():$bookmark.open($sub.type.ID, $sub.type.name) #end #end

#if($end.documentation != “”)

$end.documentation

#end

#end

#decreaseIndent()

#end

#set($classAtt = $array.createArray())

#foreach($a in $class.getAttribute())

#if(!$a.association)

#set($tmp = $classAtt.add($a))

#end

#end

#if($classAtt.size() > 0)

#writeHeaderText(“Attributes”)

#foreach($att in $classAtt)

#if(!$att.association)

#printAttr($att)

#if($att.documentation != “”)

$att.documentation

#end

#end

#end

#end

#set($classOper = $class.ownedOperation)

#if($classOper.size() > 0)

#writeHeaderText(“Operations”)

#foreach($oper in $classOper)

#printOper($oper)

#if($oper.documentation != “”)

$oper.documentation

#end

#end

#end

#set($associationLists = $array.createArray())

#foreach($attribute in $class.ownedAttribute)

#if($attribute.association)

#set($tmp = $associationLists.add($attribute.association))

#end

#end

#if($associationLists.size() > 0)

#writeHeaderText(“Associations”)

#foreach($attribute in $class.ownedAttribute)

#if($attribute.association)

#set($association = $attribute.association)

#printAsso($attribute, $association, $class)

#if($attribute.documentation != “”)

$attribute.documentation

#end

#end

#end

#end

#if($class.hasHyperlinks())

#foreach($hyperlink in $class.hyperlinks)

#if($hyperlink.getElement().elementType == “diagram-xxx-to-many-diagrams”)

#if(!$printDiagram.contains($hyperlink.getElement()))

#set($tmp = $printDiagram.add($hyperlink.getElement()))

$hyperlink.getElement().image

1. $hyperlink.getElement().name

#if($hyperlink.getElement().documentation != “”)

$hyperlink.getElement().documentation

#end

#end

#end

#end

#end

#end

#macro(createEnumerationContent $enum)

#increaseIndent()

#writeBookmarkHeader($enum, $enum.HumanName)

#if($enum.documentation != “”)

$enum.documentation

#end

#set($directKnown = $enum.baseClassifier)

#increaseIndent()

#set($size = $directKnown.size())

#if($directKnown.size() > 0)

#writeHeaderText(“Direct Known Superclasses”)

#foreach($class in $directKnown)$bookmark.open($class.ID, $class.name)#if($size != $velocityCount), #end#end

#end

#set($ownedLiteral = $enum.ownedLiteral)

package $report.getPackageQualifiedName($enum.owner, $QualifiedNameSeparator.toString())

$enum.visibility enum $enum.name

#set($size = $ownedLiteral.size())

#set($i = 1)

#if($ownedLiteral.size() > 0)

{#foreach($literal in $ownedLiteral)$literal.name#if($size != $i), #end#set($i = $i +1 )#end}

#end

#if($ownedLiteral.size() > 0)

#writeHeaderText(“Literals”)

#foreach($literal in $ownedLiteral)

$report.getIconFor($literal) $literal.name

#if($literal.documentation != “”)

$literal.documentation

#end

#end

#end

#if($enum.hasHyperlinks())

#foreach($hyperlink in $enum.hyperlinks)

#if($hyperlink.getElement().elementType == “diagram”)

$hyperlink.getElement().image

1. $hyperlink.getElement().name

#if($hyperlink.getElement().documentation != “”)

$hyperlink.getElement().documentation

#end

#end

#end

#end

#end

#macro(printAttr $att)

$report.getIconFor($att) $att.name#if($att.type) : $bookmark.open($att.type.ID, $att.type.name)#end#if($att.multiplicity != “”) [$att.multiplicity]#end#if($att.defaultValue) = $att.defaultValue.text#end

#end

#macro(printOper $oper)

#set($paramLists = $oper.ownedParameter)

#set($size = 0)

#foreach($p in $paramLists)

#if($p.direction != “return”)

#set($size = $size +1)

#end

#end

#set($i = 1)

$report.getIconFor($oper) $oper.visibility $oper.name (#foreach($param in $paramLists)#if($param.direction != “return”)$param.name#if($param.type) : $bookmark.open($param.type.ID, $param.type.name)#end#if($param.multiplicity != “”) [$param.multiplicity]#end#if($param.defaultValue) = $param.defaultValue.text#end#if($size != $i), #end#set($i = $i + 1)#end#end)#if($oper.type):$bookmark.open($oper.type.ID, $oper.type.name)#end#if($oper.hasTypeModifier() && $oper.typeModifier != “”)$oper.typeModifier#end

#end

#macro(printAsso $attribute $association $object)

#foreach($member in $association.memberEnd)

#if($member.type != $object)

#set($memberEnd = $member)

#end

#end

#set($subs = $memberEnd.getSubsettedProperty())

#set($redefs = $memberEnd.getRedefinedProperty())

$report.getIconFor($association) #if($attribute.name != “”) $attribute.name#end#if($memberEnd.type):$bookmark.open($memberEnd.type.ID, $memberEnd.type.name)#end#if($memberEnd.multiplicity != “”) [$memberEnd.multiplicity]#end #if($memberEnd.defaultValue) = $memberEnd.defaultValue.text#end#if($subs.size()>0) *Subsets*: #foreach($sub in $subs)$sub.getName():$bookmark.open($sub.type.ID, $sub.type.name) #end #end#if($redefs.size()>0) *Redefines*: #foreach($sub in $redefs)$sub.getName():$bookmark.open($sub.type.ID, $sub.type.name) #end #end

#end

#macro(findNestedElement $object)

#set($innerElement = $report.getInnerElement($object))

#foreach($nested in $innerElement)

#if($nested.elementType != “package” && $nested.elementType != “model” && $nested.elementType != “profile” && $nested.elementType != “smartpackage” )

#if($nested.elementType == “interface”)

#set($tmp = $nestedInterface.add($nested))

#elseif($nested.elementType == “class”)

#set($tmp = $nestedClass.add($nested))

#elseif($nested.elementType == “associationclass”)

#set($tmp = $nestedClass.add($nested))

#elseif($nested.elementType == “association”)

#if($nested.getName() != “”)

#set($tmp = $nestedClass.add($nested))

#end

#elseif($nested.elementType == “enumeration”)

#set($tmp = $nestedEnum.add($nested))

#end

#set($in = $report.getInnerElement($nested))

#if($in.size() > 0)

#findNestedElement($nested)

#end

#end

#end

#end

#set($packageLists = $array.createArray())

#foreach($p in $Package)

#set($tmp = $packageLists.add($p))

#end

#foreach($p in $Profile)

#set($tmp = $packageLists.add($p))

#end

#foreach($m in $Model)

#set($tmp = $packageLists.add($m))

#end

#foreach($m in $SmartPackage)

#set($tmp = $packageLists.add($m))

#end

#foreach($package in $sorter.humanSort($packageLists, “qualifiedName”))

#set($packageInterface = $array.createArray())

#set($packageClass = $array.createArray())

#set($packageEnum = $array.createArray())

#set($nestedInterface = $array.createArray())

#set($nestedClass = $array.createArray())

#set($nestedEnum = $array.createArray())

#if($package.elementType == “package” || $package.elementType == “smartpackage”)

#foreach($element in $report.getOwnedElementsIncludingAdditional($package, true))

#if($element.elementType == “interface”)

#set($tmp = $packageInterface.add($element))

#set($inner = $report.getInnerElement($element))

#if($inner.size() > 0)

#findNestedElement($element)

#end

#elseif($element.elementType == “class”)

#set($tmp = $packageClass.add($element))

#set($inner = $report.getInnerElement($element))

#if($inner.size() > 0)

#findNestedElement($element)

#end

#elseif($element.elementType == “enumeration”)

#set($tmp = $packageEnum.add($element))

#set($inner = $report.getInnerElement($element))

#if($inner.size() > 0)

#findNestedElement($element)

#end

#end

#end

#end

#set($tmp = $array.addCollection($packageInterface, $nestedInterface))

#set($tmp = $array.addCollection($packageClass, $nestedClass))

#set($tmp = $array.addCollection($packageEnum, $nestedEnum))

#set($diagramList = $array.createArray())

#foreach($d in $sorter.humanSort($package.ownedDiagram))

#if($d.diagramType == “Class Diagram”)

#set($tmp = $diagramList.add($d))

#end

#end

#if($diagramList.size() > 0 || $packageClass.size() > 0 || $packageInterface.size() > 0 || $packageEnum.size() > 0)

#if($package != $project.model)

# $report.getQualifiedName($package, $QualifiedNameSeparator.toString())

#if($package.documentation != “”)

$package.documentation

#end

#end

#foreach($diagram in $sorter.humanSort($diagramList))

#if($package == $project.model)

# Diagram: $diagram.name

#else

## Diagram: $diagram.name

#end

$diagram.image

1. $diagram.name

#if($diagram.documentation != “”)

$diagram.documentation

#end

#set($diagramElements = $array.createArray())

#if($package.elementType == “package” || $package.elementType == “smartpackage”)

#set($diagramElements = $package.getPackagedElement())

#end

#set($interfaceLists = $array.createArray())

#set($classLists = $array.createArray())

#set($enumLists = $array.createArray())

#foreach($element in $diagramElements)

#if($element.elementType == “interface”)

#set($tmp = $interfaceLists.add($element))

#set($tmp = $packageInterface.remove($element))

#elseif($element.elementType == “class”)

#set($tmp = $classLists.add($element))

#set($tmp = $packageClass.remove($element))

#elseif($element.elementType == “associationclass”)

#set($tmp = $classLists.add($element))

#set($tmp = $packageClass.remove($element))

#elseif($element.elementType == “association”)

#if($element.getName() != “”)

#set($tmp = $classLists.add($element))

#set($tmp = $packageClass.remove($element))

#end

#elseif($element.elementType == “enumeration”)

#set($tmp = $enumLists.add($element))

#set($tmp = $packageEnum.remove($element))

#end

#end

#if($interfaceLists.size() > 0)

#set($knownInterface = $array.createArray())

#foreach($interface in $sorter.humanSort($interfaceLists))

#if(!$printInterface.contains($interface))

#if($package == $project.model)

#set($level = 1)

#else

#set($level = 2)

#end

#set($tmp = $printInterface.add($interface))

#createInterfaceContent($interface)

#else

#set($tmp = $knownInterface.add($interface))

#end

#end

#if($knownInterface.size() > 0)

#if($package == $project.model)

#set($level = 1)

#else

#set($level = 2)

#end

#increaseIndent()

#writeHeaderText(“Known other interfaces”)

#set($size = $knownInterface.size())

#foreach($i in $knownInterface)$bookmark.open($i.ID, $i.humanName)#if($size != $velocityCount), #end#end

#end

#end##endinterface

#if($classLists.size() > 0)

#set($knownClass = $array.createArray())

#foreach($class in $sorter.humanSort($classLists))

#if(!$printClass.contains($class))

#if($package == $project.model)

#set($level = 1)

#else

#set($level = 2)

#end

#set($tmp = $printClass.add($class))

#decreaseIndent()

#createClassContent($class)

#increaseIndent()

#else

#set($tmp = $knownClass.add($class))

#end

#end

#if($knownClass.size() > 0)

#if($package == $project.model)

#set($level = 1)

#else

#set($level = 2)

#end

#increaseIndent()

#writeHeaderText(“Known other classes”)

#set($size = $knownClass.size())

#foreach($c in $knownClass)$bookmark.open($c.ID, $c.humanName)#if($size != $velocityCount), #end#end

#end

#end##endclass

#if($enumLists.size() > 0)

#set($knownEnum = $array.createArray())

#foreach($enum in $sorter.humanSort($enumLists))

#if(!$printEnum.contains($enum))

#if($package == $project.model)

#set($level = 1)

#else

#set($level = 2)

#end

#set($tmp = $printEnum.add($enum))

#createEnumerationContent($enum)

#else

#set($tmp = $knownEnum.add($enum))

#end

#end

#if($knownEnum.size() > 0)

#if($package == $project.model)

#set($level = 1)

#else

#set($level = 2)

#end

#increaseIndent()

#writeHeaderText(“Known other enumerations”)

#set($size = $knownEnum.size())

#foreach($e in $knownEnum)$bookmark.open($e.ID, $e.humanName)#if($size != $velocityCount), #end#end

#end

#end##endenumeration

#end

#if($packageInterface.size() > 0)

#set($knownInterface = $array.createArray())

#foreach($interface in $sorter.humanSort($packageInterface))

#if(!$printInterface.contains($interface))

#if($package == $project.model)

#set($level = 0)

#else

#set($level = 1)

#end

#set($tmp = $printInterface.add($interface))

#createInterfaceContent($interface)

#else

#set($tmp = $knownInterface.add($interface))

#end

#end

#if($knownInterface.size() > 0)

#if($package == $project.model)

#set($level = 0)

#else

#set($level = 1)

#end

#increaseIndent()

#writeHeaderText(“Known other interfaces”)

#set($size = $knownInterface.size())

#foreach($i in $knownInterface)$bookmark.open($i.ID, $i.humanName)#if($size != $velocityCount), #end#end

#end

#end##endpackageInterface

#if($packageClass.size() > 0)

#set($knownClass = $array.createArray())

#foreach($class in $sorter.humanSort($packageClass))

#if(!$printClass.contains($class))

#if($package == $project.model)

#set($level = 0)

#else

#set($level = 1)

#end

#set($tmp = $printClass.add($class))

#decreaseIndent()

#createClassContent($class)

#increaseIndent()

#else

#set($tmp = $knownClass.add($class))

#end

#end

#if($knownClass.size() > 0)

#if($package == $project.model)

#set($level = 0)

#else

#set($level = 1)

#end

#increaseIndent()

#writeHeaderText(“Known other classes”)

#set($size = $knownClass.size())

#foreach($c in $knownClass)$bookmark.open($c.ID, $c.humanName)#if($size != $velocityCount), #end#end

#end

#end##endpackageClass

#if($packageEnum.size() > 0)

#set($knownEnum = $array.createArray())

#foreach($enum in $sorter.humanSort($packageEnum))

#if(!$printEnum.contains($enum))

#if($package == $project.model)

#set($level = 0)

#else

#set($level = 1)

#end

#set($tmp = $printEnum.add($enum))

#createEnumerationContent($enum)

#else

#set($tmp = $knownEnum.add($enum))

#end

#end

#if($knownEnum.size() > 0)

#if($package == $project.model)

#set($level = 0)

#else

#set($level = 1)

#end

#increaseIndent()

#writeHeaderText(“Known other enumerations”)

#set($size = $knownEnum.size())

#foreach($e in $knownEnum)$bookmark.open($e.ID, $e.humanName)#if($size != $velocityCount), #end#end

#end

#end##endpackageEnum

#end

#end

#if($PrimitiveType.size() > 0)

# Primitive Type

#foreach($primitive in $sorter.humanSort($PrimitiveType))

## $primitive.name

#if($primitive.documentation != “”)

$primitive.documentation

#end

#end

#end

#if($DataType.size() > 0)

# Data Type

#foreach($data in $sorter.humanSort($DataType))

## $data.name

#if($data.documentation != “”)

$data.documentation

#end

#end

#end

#if($Signal.size() > 0)

# Signal

#foreach($signal in $sorter.humanSort($Signal))

## $signal.name

#if($signal.documentation != “”)

$signal.documentation

#end

#end

#end

#if($printClass.size() > 0 || $printInterface.size() > 0 || $printEnum.size() > 0)

# Tree

#if($printClass.size() > 0)

## Class Hierarchy

#foreach ($c in $sorter.humanSort($printClass))

#if (!$c.baseClassifier)

$bookmark.open($c.ID, $report.getPackageQualifiedName($c, $QualifiedNameSeparator.toString()))

#child($c, 0)

#end

#end

#end

#if($printInterface.size() > 0)

## Interface Hierarchy

#foreach($i in $sorter.humanSort($printInterface))

#if(!$i.baseClassifier)

$bookmark.open($i.ID, $report.getPackageQualifiedName($i, $QualifiedNameSeparator.toString()))

#child($i, 0)

#end

#end

#end

#if($printEnum.size() > 0)

## Enumeration Hierarchy

#foreach($e in $sorter.humanSort($printEnum))

#if(!$e.baseClassifier)

$bookmark.open($e.ID, $report.getPackageQualifiedName($e, $QualifiedNameSeparator.toString()))

#child($e, 0)

#end

#end

#end

#end

#macro (child $parent $level)

#foreach ($gen in $sorter.humanSort($parent.get\_generalizationOfGeneral()))

#foreach($i in [0..$level])

#end

#foreach($s in $gen.source)

$bookmark.open($s.ID, $report.getPackageQualifiedName($s, $QualifiedNameSeparator.toString()))

#set ($level = $level + 1)

#child($s, $level)

#set ($level = $level - 1)

#end

#end

#end