'TESTOUTSIDEGIT'

VERSION 1.0.0-BUILD-SNAPSHOT

CODE ANALYSIS

By: default

2023-08-02

'testoutsidegit'

CONTENT

'testoutsidegit'

INTRODUCTION

This document contains results of the code analysis of 'testoutsidegit'.

CONFIGURATION

- Quality Profiles
    - Names: Sonar way [CSS]; Sonar way [Java]; Sonar way [JavaScript]; Sonar way [JSP]; Sonar way [HTML]; Sonar way [XML];
    - Files: AYlzScn1f_Qmj1ZxJua6.json; AYlzSd28f_Qmj1ZxJvYY.json; AYlzSdBlf_Qmj1ZxJu[illegible]L.json; AYlzScy9f_Qmj1ZxJuc3.json; AYlzSd9jf_Qmj1ZxJvg1.json; AYlzSeBlf_Qmj1ZxJviF.json;
- Quality Gate
    - Name: AYlEo0Ql0S8MdK3ohUdJ
    - File: AYlEo0Ql0S8MdK3ohUdJ.xml

'testoutsidegit'

SYNTHESIS

ANALYSIS STATUS

Reliability	Security	Security Review	Maintainability
C	D	E	A

QUALITY GATE STATUS

Quality Gate Status	Passed

METRICS

Coverage	Duplication	Comment density	Median number of lines of code per file	Adherence to coding standard
0.0 %	0.1 %	2.8 %	58.0	99.4 %

TESTS

Total	Success Rate	Skipped	Errors	Failures
0	0 %	0	0	0

DETAILED TECHNICAL DEBT

Reliability	Security	Maintainability	Total
0d 0h 12min	0d 0h 10min	0d 2h 44min	**0d 3h 6min**

'testoutsidegit'

'testoutsidegit'

METRICS RANGE

	Cyclomatic Complexity	Cognitive Complexity	Lines of code per file	Comment density (%)	Coverage	Duplication (%)
Min	0.0	0.0	2.0	0.0	0.0	0.0
Max	45.0	4.0	59242.0	72.7	0.0	57.1

VOLUME

Language	Number
CSS	57426
Java	197
JSP	308
HTML	1226
XML	263
Total	59420

'testoutsidegit'

ISSUES

CHARTS

ISSUES COUNT BY SEVERITY AND TYPE

Type / Severity	INFO	MINOR	MAJOR	CRITICAL	BLOCKER
BUG	0	2	1	0	0
VULNERABILITY	0	0	0	1	0
CODE_SMELL	0	2	21	1	0

ISSUES LIST

Name	Description	Type	Severity	Number
Tables should have headers	Why is this an issue?Assistive technologies, such as screen readers, use &lt;th&gt; headers to provide some context when users navigates a table. Withoutit the user gets rapidly lost in the flow of data.Headers should be properly associated with the corresponding &lt;td&gt;&nbsp;cells by using either a scope attribute or	BUG	MAJOR	1

	headers and id attributes. See&nbsp;W3C WAI&nbsp;Web AccessibilityTutorials&nbsp;for more information.This rule raises an issue whenever a &lt;table&gt; does not contain any&nbsp;&lt;th&gt;&nbsp;elements.Noncompliant code example&lt;table&gt; &lt;!-- Noncompliant --&gt; &lt;tr&gt; &lt;td&gt;Name&lt;/td&gt; &lt;td&gt;Age&lt;/td&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td&gt;John Doe&lt;/td&gt; &lt;td&gt;24&lt;/td&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td&gt;Alice Doe&lt;/td&gt; &lt;td&gt;54&lt;/td&gt; &lt;/tr&gt;&lt;/table&gt; Compliant solution&lt;table&gt; &lt;tr&gt; &lt;th scope="col"&gt;Name&lt;/th&gt; &lt;th scope="col"&gt;Age&lt;/th&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td&gt;John Doe&lt;/td&gt; &lt;td&gt;24&lt;/td&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td&gt;Alice Doe&lt;/td&gt; &lt;td&gt;54&lt;/td&gt; &lt;/tr&gt;&lt;/table&gt; ExceptionsNo issue will be raised on &lt;table&gt; used for layout purpose, i.e. when it contains a role attribute set to"presentation" or "none". Note that using &lt;table&gt; for layoutpurpose is a bad practice.No issu will be raised on &lt;table&gt; containing an aria-hidden attribute set to "true".Resources WCAG2, 1.3.1&nbsp; &nbsp;Info and Relationships WCAG2,&nbsp;H51 - Using table markup to present tabular information			
"<table>" tags should have a description	Why is this an issue?In order to be accessible to visually impaired users, it is important that tables provides a description of its content before the data isaccessed.The simplest way to do it, and also the one recommended by WCAG2 is to add a&lt;caption&gt; element inside the &lt;table&gt;.Other technics this rule accepts are: adding a concise description via aria-label or aria-labelledby attributes in the &lt;table&gt;. referencing a description element with an aria-describedby attribute in the &lt;table&gt;. embedding the &lt;table&gt; inside a &lt;figure&gt; which also contains a &lt;figcaption&gt;. adding a summary attribute to the &lt;table&gt; tag. However note that this attribute has been deprecated in HTML5. See&nbsp;W3C WAI&nbsp;Web Accessibility Tutorials&nbsp;for more information.This rule raises an issue when a &lt;table&gt; has neither of the previously mentioned description mechanisms.Noncompliant code example&lt;table&gt; &lt;!-- Noncompliant --&gt; ...&lt;table&gt;Compliant solutionAdding a &lt;caption&gt; element.&lt;table&gt; &lt;caption&gt;New York City Marathon Results 2013&lt;/caption&gt; ...&lt;/table&gt;Adding an aria-describedby attribute.&lt;p id="mydesc"&gt;New York City Marathon Results 2013&lt;/p&gt;&lt;table aria-	BUG	MINOR	2

	describedby="mydesc"&gt; ...&lt;/table&gt;Embeddir the table in a &lt;figure&gt; which also contains a &lt;figcaption&gt;.&lt;figure&gt; &lt;figcaption&gt;Nev York City Marathon Results 2013&lt;/figcaption&gt; &lt;table&gt; ... &lt;/table&gt;&lt;/figure&gt;Addir summary attribute.&nbsp;However note that this attribute has been deprecated in HTML5.&lt;table summary="New York City Marathon Results 2013"&gt; ...&lt;/table&gt;ExceptionsNo issue will be raised on &lt;table&gt; used for layout purpose, i.e. when it contains a role attribute set to"presentation" or "none". Note that using &lt;table&gt; for layout purpose is a bad practice.No issue will be raised either on &lt;table&gt; containing an aria-hidden attribute set to "true". Resources WCAG2, 1.3.1&nbsp;-&nbsp;Info and Relationships WCAG2,&nbsp;H39 - Using caption elements to associate data table captions with data tables			
String literals should not be duplicated	Why is this an issue?Duplicated string literals make the process of refactoring error-prone, since you must be sure to update all occurrences.On the other hand, constants can be referenced from many places, but only need to be updated in a single place.Noncompliant code exampleWith the default threshold of 3:public void run { prepare("action1"); // Noncompliant - "action1" is duplicated 3 times execute("action1"); release("action1");}@SuppressWarning("all") // Compliant - annotations are excludedprivate void method1() { /* ... */ }@SuppressWarning("all")private void method2() { /* ... */ }public String method3(String a { System.out.println("'" + a + "'"); // Compliant - literal "'" has less than 5 characters and is excluded return ""; // Compliant - literal "" has less than 5 characters and is excluded}Compliant solutionprivate static final String ACTION_1 = "action1"; // Compliantpublic void run() { prepare(ACTION_1); // Compliant execute(ACTION_1); release(ACTION_1); ExceptionsTo prevent generating some false-positives, literals having less than 5 characters are excluded.	CODE_SMELL	CRITICA L	1
Attributes deprecated in HTML5 should not be used	Why is this an issue?With the advent of HTML5, many old attributes were deprecated. To ensure the best user experience, deprecated attributes should not be used. Thisrule checks for the following deprecated attributes, where CSS should be used instead. A Removed from accept form caption, col, div, embed, h1-h6, hr, iframe, img, input, legend, object, p, table, tbody, thead, tfoot, td, th, tr alink body allowtransparency ifram	CODE_SMELL	MAJOR	5

'testoutsidegit'	archive� object� � � axis� td, th� � � bacl body, table, thead, tbody, tfoot, tr, td,� th bgcolor� body, table, td, th, tr� � � border� img (border="0" allowed), object� � � bordercolor� ta � � cellpadding� table� � � cellspacing� tak char� col, tbody, thead, tfoot, td, th, tr� � � charo col, tbody, thead, tfoot, td, th, tr� � � charset� a, l � � classid� object� � � clear� br� � � object� � � codebase� object� � � codetype object� � � color� hr� � � compact� dl, o coords� a� � � datafld� a, applet, button, div, fieldset, frame,� iframe, img, input, label, legend, marquee, object,� param, select, span, textarea dataformatas� button, div, input, label, legend, marquee,� object, option, select, span, table datapagesize� table� � � datasrc� a, applet, bu div, frame, iframe, img,� input, label, legend, marquee, object, option,� select, span, table, textarea declare� object� � � event� script� � � fc script� � � frame� table� � � frameborder iframe� � � height� td, th� � � hspace� e iframe, img, input, object� � � ismap� input langauge� script (language="javascript", case insensitive, allowed)� � � link� body� � � lo img� � � marginbottom� body� � � marginh body, iframe� � � marginleft� body marginright� body� � � margintop� body marginwidth� body, iframe� � � methods� a, li � name� a (name="[a's element id]" allowed), embed img, option� � � nohref� area� � � noshade � � nowrap� td, th� � � profile� head� � table� � � scheme� meta� � � scope� td scrolling� iframe� � � shape� a� � � size� � standby� object� � � summary� table target� link� � � text� body� � � type� param, ul� � � urn� a, link� � � usemap� � valign� col, tbody, thead, tfoot, td, th, tr valuetype� param� � � version� html� � � body� � � vspace� embed, iframe, img, input, obje � � width� col, hr, pre, table, td, th� � ��Resource W3C, Differences in HTML5 � WHATWG, Obsolete Features			
"aria-label" or "aria-labelledby" attributes should be used to differentiate similar elements	Why is this an issue?�If a page contains multiple &lt;nav&gt;&nbsp;or &lt;aside&gt; elements, each one should have an aria-label�or aria-labelledby attribute so that they can be differentiated. The same rule applies when multiple elements have�a&nbsp;role attribute with the same "landmark" value.�Landmark roles are: banner, complementary, contentinfo, form, main,�navigation,	CODE_SMELL	MAJOR	2

	search, application.&nbsp;The use of ARIA markup helps users of&nbsp;screen readers navigate across blocks of content. For example it makes groups of links easier to locateor skip.Noncompliant code exampleMultiple &lt;nav&gt; element&lt;nav&gt; &lt;!-- Noncompliant --&gt; &lt;ul&gt; &lt;li&gt;A list of navigation links&lt;/li&gt; &lt;/ul&gt;&lt;/nav&gt;&lt;article&gt &lt;nav&gt; &lt;!-- Noncompliant --&gt; Another list of navigation links &lt;/nav&gt;&lt;/article&gt; Repeated "landmark" role "navigation"&lt;div id="mainnav" role="navigation"&gt; &lt;!-- Noncompliant --&gt; &lt;h2 id="mainnavheading"&gt;Site Navigation&lt;/h2&gt; &lt;ul&gt; &lt;li&gt;List of links&lt;/li&gt; &lt;/ul&gt;&lt;/div&gt;&lt;div id="secondarynav" role="navigation"&gt; &lt;!-- Noncompliant --&gt; &lt;h2 id="secondarynavheading"&gt;Related links&lt;/h2&gt; &lt;ul&gt; &lt;li&gt;List of links&lt;/li&gt; &lt;/ul&gt &lt;/div&gt;Compliant solution&lt;nav aria-label="Sit menu"&gt; &lt;ul&gt; &lt;li&gt;A list of navigation links&lt;/li&gt; &lt;/ul&gt;&lt;/nav&gt;&lt;article&gt &lt;nav aria-label="Related links"&gt; Another list of navigation links &lt;/nav&gt;&lt;/article&gt;&lt;div id="mainnav" role="navigation" aria-labelledby="mainnavheading"&gt; &lt;h2 id="mainnavheading"&gt;Site Navigation&lt;/h2&gt; &lt;ul&gt; &lt;li&gt;List of links&lt;/li&gt; &lt;/ul&gt &lt;/div&gt;&lt;div id="secondarynav" role="navigation" aria-labelledby="secondarynavheading"&gt; &lt;h2 id="secondarynavheading"&gt;Related links&lt;/h2&gt; &lt;ul&gt; &lt;li&gt;List of links&lt;/li&gt; &lt;/ul&gt &lt;/div&gt;Resources WCAG2, ARIA11 - Using ARIA landmarks to identify regions of a page WCAG2, H97 - Grouping related links using the nav element WCAG2 1.3.1 Info and Relationships			
Constructors should not be used to instantiate "String", "BigInteger", "BigDecimal" and primitive-wrapper classes	Why is this an issue?Constructors for String, BigInteger, BigDecimal and the objects used to wrap primitives should never beused. Doing so is less clear and uses more memory than simply using the desired value in the case of strings, and using valueOf foreverything else. Noncompliant code exampleString empty = new String(// Noncompliant; yields essentially "", so just use that. String nonempty = new String("Hello world"); // NoncompliantDouble myDouble = new Double(1.1); // Noncompliant; use valueOfInteger integer = new Integer(1); // NoncompliantBoolean bool = new Boolean(true); // NoncompliantBigInteger bigInteger1 = new BigInteger("3"); // NoncompliantBigInteger bigInteger2 = new BigInteger("9223372036854775807");	CODE_SMELL	MAJOR	2

'testoutsidegit'

	// NoncompliantBigInteger bigInteger3 = new BigInteger("111222333444555666777888999"); // Compliant, greater than Long.MAX_VALUECompliant solutionString empty = "";String nonempty = "Hello world";Double myDouble = Double.valueOf(1.1);Integer integer = Integer.valueOf(1);Boolean bool = Boolean.valueOf(true);BigInteger bigInteger1 = BigInteger.valueOf(3);BigInteger bigInteger2 = BigInteger.valueOf(9223372036854775807L);BigInteger bigInteger3 = new BigInteger("111222333444555666777888999"); ExceptionsBigDecimal constructor with double argumen is ignored as using valueOf instead might change resultingvalue. See S2111 .			
"Preconditions" and logging arguments should not require evaluation	Why is this an issue?Passing message arguments that require further evaluation into a Guava com.google.common.base.Preconditions check can result in aperformance penalty. That’s because whether or not they’re needed, each argument must be resolved before the method is actually called.Similarly, passing concatenated strings into a logging method can also incur a needless performance hit because the concatenation will be performedevery time the method is called, whether or not the log level is low enough to show the message.Instead, you should structure your code to pass static or pre-computed values into Preconditions conditions check and loggingcalls. Specifically, the built-in string formatting should be used instead of string concatenation, and if the message is the result of a method call,then Preconditions should be skipped altogether, and the relevant exception should be conditionally thrown instead.Noncompliant code examplelogger.log(Level.DEBUG, "Something went wrong: " + message); // Noncompliant; string concatenation performed even when log level too high to show DEBUG messageslogger.fine("An exception occurred with message: " + message); // Noncompliant LOG.error("Unable to open file " + csvPath, e); // NoncompliantPreconditions.checkState(a &gt; 0, "Arg must be positive, but got " + a); // Noncompliant. String concatenation performed even when a &gt; 0 Preconditions.checkState(condition, formatMessage()); // Noncompliant. formatMessage() invoked regardless of conditionPreconditions.checkState(condition, "message: %s", formatMessage()); // NoncompliantCompliant solutionlogger.log(Level.SEVERE, "Something went wrong: {0} ", message); // String formatting only applied if neededlogger.fine("An exception occurred with message: {}", message); // SLF4J, Log4j	CODE_SMELL	MAJOR	5

'testoutsidegit'

	logger.log(Level.SEVERE, () -&gt; "Something went wrong: " + message); // since Java 8, we can use Supplier , which will be evaluated lazilyLOG.error("Unable to ope file {0}", csvPath, e);if (LOG.isDebugEnabled()) { LOG.debug("Unable to open file " + csvPath, e); // this is compliant, because it will not evaluate if log level is above debug.}Preconditions.checkState(arg &gt; 0, "Ar must be positive, but got %d", a); // String formatting only applied if neededif (!condition) { throw new IllegalStateException(formatMessage()); // formatMessage() only invoked conditionally}if (!condition) { throw new IllegalStateException("message: " + formatMessage());}Exceptionscatch blocks are ignored, because the performance penalty is unimportant on exceptional paths (catch block should not be a part ofstandard program flow). Getters are ignored as well as methods called on annotations which can be considered as getters. This rule accounts forexplicit test-level testing with SLF4J methods isXXXEnabled and ignores the bodies of such if statements.			
Printf-style format strings should be used correctly	Why is this an issue?Because printf-style format strings are interpreted at runtime, rather than validated by the compiler, they can contain errors thatresult in the wrong strings being created. This rule statically validates the correlation of printf-style format strings to their arguments when calling the format(...) methods of java.util.Formatter, java.lang.String,java.io.PrintStream, MessageFormat, and java.io.PrintWriter classes and the printf(...) methods ofjava.io.PrintStream or java.io.PrintWriter classes.Noncompliant code example String.format("First {0} and then {1}", "foo", "bar"); //Noncompliant. Looks like there is a confusion with the use of {{java.text.MessageFormat}}, parameters "foo" and "bar" will be simply ignored here String.format("Display %3$d and then %d", 1, 2, 3); //Noncompliant; the second argument '2' is unused String.format("Too many arguments %d and %d", 1, 2, 3); //Noncompliant; the third argument '3' is unused String.format("First Line\n"); //Noncompliant; %n should be used in place of \n to produce the platform-specific line separatorString.format("Is myObject null ? %b", myObject); //Noncompliant; when a non-boolean argument is formatted with %b, it prints true for any nonnull value, and false for null. Even if intended, this is misleading. It's better to directly inject the boolean value (myObject == null in this case)String.format("value is " + value); // NoncompliantString s = String.format("string without arguments"); // Noncompliant	CODE_SMELL	MAJOR	5

'testoutsidegit'	MessageFormat.format("Result '{0}'.", value); // Noncompliant; String contains no format specifiers. (quote are discarding format specifiers) MessageFormat.format("Result {0}.", value, value); // Noncompliant; 2nd argument is not used MessageFormat.format("Result {0}.", myObject.toString()); // Noncompliant; no need to call toString() on objectsjava.util.Logger logger; logger.log(java.util.logging.Level.SEVERE, "Result {0}.", myObject.toString()); // Noncompliant; no need to call toString() on objects logger.log(java.util.logging.Level.SEVERE, "Result.", new Exception()); // compliant, parameter is an exception logger.log(java.util.logging.Level.SEVERE, "Result '{0}'", 14); // Noncompliant - String contains no format specifiers.logger.log(java.util.logging.Level.SEVERE, "Result " + param, exception); // Noncompliant; Lambda should be used to differ string concatenation. org.slf4j.Logger slf4jLog;org.slf4j.Marker marker; slf4jLog.debug(marker, "message {}"); slf4jLog.debug(marker, "message", 1); // Noncompliant - String contains no format specifiers. org.apache.logging.log4j.Logger log4jLog; log4jLog.debug("message", 1); // Noncompliant - String contains no format specifiers.Compliant solution String.format("First %s and then %s", "foo", "bar"); String.format("Display %2$d and then %d", 1, 3); String.format("Too many arguments %d %d", 1, 2); String.format("First Line%n");String.format("Is myObject null ? %b", myObject == null);String.format("value is %d", value);String s = "string without arguments"; MessageFormat.format("Result {0}.", value); MessageFormat.format("Result '{0}' = {0}", value); MessageFormat.format("Result {0}.", myObject); java.util.Logger logger; logger.log(java.util.logging.Level.SEVERE, "Result {0}.", myObject);logger.log(java.util.logging.Level.SEVERE, "Result {0}'", 14); logger.log(java.util.logging.Level.SEVERE, exception, () -&gt; "Result " + param);org.slf4j.Logger slf4jLog; org.slf4j.Marker marker;slf4jLog.debug(marker, "message {}");slf4jLog.debug(marker, "message {}", 1); org.apache.logging.log4j.Logger log4jLog; log4jLog.debug("message {}", 1);Resources CERT, FI C. - Use valid format strings			
"@Deprecated" code marked for removal should never be used	Why is this an issue?Java 9 introduced a flag for the @Deprecated annotation, which allows to explicitly say if the deprecated code is planned to beremoved at some point or not. This is done using forRemoval=true as	CODE_SMELL	MAJOR	2

	annotation parameter. The javadoc of the annotation explicitlymention the following: If true, it means that tl API element is earmarked for removal in a future release. If false, the API element is deprecated, but there is currently no intention to remove it in a future release. While usually deprecated classes, interfaces, and their deprecated members should be avoided rather than used, inherited or extended, those alreadymarked for removal are much more sensitive to causing trouble in your code soon. Consequently, any usage of such deprecated code should be avoided orremoved. Noncompliant code example/** * @deprecated As of release 1.3, replaced by {@link #Fee}. Will be dropped with release 1.4. */@Deprecated(forRemoval=true) public class Foo { ... }public class Bar { /** * @deprecated As of release 1.7, replaced by {@link #doTheThingBetter()} */ @Deprecated(forRemoval=true) public void doTheThing() { ... } public void doTheThingBetter() { ... } /** * @deprecated As of release 1.14 due to poor performances. */ @Deprecated(forRemoval=false) public void doTheOtherThing() { ... }}public class Qix extends Bar { @Override public void doTheThing() { ... } // Noncompliant; don't override a deprecated method marked for removal}public class Bar extends Foo { // Noncompliant; Foo is deprecated and will be removed public void myMethod() { Bar bar = new Bar(); // okay; the class isn't deprecated bar.doTheThing(); // Noncompliant; doTheThing method is deprecated and will be removed bar.doTheOtherThing(); // Okay; deprecated, but not marked for removal }}Resources MITRE, CWE-477 - Use of Obsolete Functions CERT, MET02-J. - Do not use deprecated or obsolete classes or methods RSPEC-1874 for standard deprecation use			
Composed "@RequestMapping" variants should be preferred	Why is this an issue?Spring framework 4.3 introduced variants of the @RequestMapping annotation to better represent the semantics of the annotated methods.The use of @GetMapping, @PostMapping, @PutMapping, @PatchMapping and @DeleteMappingshould be preferred to the use of the raw @RequestMapping(method = RequestMethod.XYZ). Noncompliant code example@RequestMapping(path = "/greeting", method = RequestMethod.GET) // Noncompliantpublic Greeting greeting(@RequestParam(value = "name", defaultValue = "World") String name) {...}Compliant solution @GetMapping(path = "/greeting") // Compliantpublic Greeting greeting(@RequestParam(value = "name", defaultValue = "World") String name) {...}	CODE_SMELL	MINOR	2

'testoutsidegit'

Persistent entities should not be used as arguments of "@RequestMapping" methods	Why is this an issue?On one side, Spring MVC automatically bind request parameters to beans declared as arguments of methods annotated with @RequestMapping. Because of this automatic binding feature, it’s possible to feed some unexpected fields on the arguments of the@RequestMapping annotated methods.On the other end, persistent objects (@Entity or @Document) are linked to the underlying database and updatedautomatically by a persistence framework, such as Hibernate, JPA or Spring Data MongoDB.These two facts combined together can lead to malicious attack: if a persistent object is used as an argument of a method annotated with@RequestMapping, it’s possible from a specially crafted user input, to change the content of unexpected fields into the database.For this reason, using @Entity or @Document objects as arguments of methods annotated with @RequestMappingshould be avoided.In addition to @RequestMapping, this rule also considers the annotations introduced in Spring Framework 4.3: @GetMapping,@PostMapping, @PutMapping, @DeleteMapping, @PatchMapping. Noncompliant code exampleimport javax.persistence.Entity;@Entitypublic class Wish { Lo productId; Long quantity; Client client;}@Entitypul class Client { String clientId; String name; String password;}import org.springframework.stereotype.Controller;import org.springframework.web.bind.annotation.RequestMapping;@Controllerpublic class WishListController { @PostMapping(path = "/saveForLater") public String saveForLater(Wish wish) { session.save(wish); } @RequestMapping(path = "/saveForLater", method = RequestMethod.POST) public String saveForLater(Wish wish) { session.save(wish); }}Compliant solutionp class WishDTO { Long productId; Long quantity; Long clientId;}import org.springframework.stereotype.Controller;import org.springframework.web.bind.annotation.RequestMapping;@Controllerpublic class PurchaseOrderController { @PostMapping(path = "/saveForLater") public String saveForLater(WishDTO wish) { Wish persistentWish = new Wish(); // do the mapping between "wish" and "persistentWish" [...] session.save(persistentWish); @RequestMapping(path = "/saveForLater", method = RequestMethod.POST) public String saveForLater(WishDTO wish) { Wish persistentWish = new Wish(); // do the mapping between "wish" and "persistentWish" [...] session.save(persistentWish);ExceptionsNo issue is reported when the parameter is annotated with @PathVariable from Spring Framework, since the lookup will be done via id,the object cannot be	VULNERABILITY	CRITICAL	1

'testoutsidegit'

forged on client side.Resources OWASP Top 10 2021 Category A8 - Software and Data Integrity Failures OWASP Top 10 2017 Category A5 - Broken Access Control MITRE, CWE-915 - Improperly Controlled Modification of Dynamically-Determined Object Attributes Two Security Vulnerabilities in the Spring Framework’s MVC by Ryan Berg and Dinis Cruz

'testoutsidegit'

SECURITY HOTSPOTS

SECURITY HOTSPOTS COUNT BY CATEGORY AND PRIORITY

Category / Priority	LOW	MEDIUM	HIGH
LDAP Injection	0	0	0
Object Injection	0	0	0
Server-Side Request Forgery (SSRF)	0	0	0
XML External Entity (XXE)	0	0	0
Insecure Configuration	0	0	0
XPath Injection	0	0	0
Authentication	0	0	0
Weak Cryptography	0	0	0
Denial of Service (DoS)	0	0	0
Log Injection	0	0	0
Cross-Site Request Forgery (CSRF)	0	0	2
Open Redirect	0	0	0
Permission	0	0	0
SQL Injection	0	0	0
Encryption of Sensitive Data	0	0	0
Traceability	0	0	0
Buffer Overflow	0	0	0
File Manipulation	0	0	0
Code Injection (RCE)	0	0	0

'testoutsidegit'

Cross-Site Scripting (XSS)	0	0	0
Command Injection	0	0	0
Path Traversal Injection	0	0	0
HTTP Response Splitting	0	0	0
Others	2	0	0

SECURITY HOTSPOTS LIST

Category	Name	Priority	Severity	Count
Others	Authorizing an opened window to access back to the originating window is security-sensitive	LOW	MINOR	2
Cross-Site Request Forgery (CSRF)	Allowing both safe and unsafe HTTP methods is security-sensitive	HIGH	MINOR	2