Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Laboratorio de Desarrollo y Herramientas:

Herramientas de Calidad del Producto Software y Documentación. SonarQube, Maven y Doxygen

Mario Alfonso Clavijo Mojica
(alu0101330457@ull.edu.es)

Índice:

1. Introducción.

En el presente informe se procederá a explicar y/o resolver los apartados propuestos por el docente, entre ellos, expondremos los plugins que se han utilizado para la configuración del **pom.xml** y por consiguiente nuestro proyecto **ExpositoTOP.** Además de la utilización de estos plugins también se utilizó la herramienta **sonarQube** junto con **maven** para la realización de un análisis del código del proyecto nombrado anteriormente.

2. Configuración de plugins soportados en maven.

1. CheckStyle

El complemento Checkstyle genera un informe sobre el estilo de código utilizado por los desarrolladores.

Summary

Files	Info	Warnings	Errors
11	0	0	947

2. PMD

El complemento PMD le permite ejecutar automáticamente la herramienta de análisis de código PMD en el código fuente de su proyecto y generar un informe del sitio con sus resultados. También es compatible con la herramienta Copiar/Pegar Detector (o CPD) que se distribuye con PMD.

Violations By Priority

Priority 3

ExpositoTOP/src/es/ull/esit/utilities/ExpositoUtilities.java

Rule	Violation	Line
UnusedPrivateMethod	Avoid unused private methods such as 'getFirstAppearance(int,int)'.	27
CollapsibleIfStatements	These nested if statements could be combined	94-97
EmptyCatchBlock	Avoid empty catch blocks	221-222
EmptyCatchBlock	Avoid empty catch blocks	230-231

3. JXR

El complemento JXR produce una referencia cruzada de las fuentes del proyecto. Los informes generados facilitan al usuario la referencia o la búsqueda de líneas de código específicas. También es útil cuando se usa con el complemento PMD para hacer referencia a errores encontrados en el código.

4. **Surefire Report**

   El complemento Surefire Report analiza el TEST-* generado. xml en ${basedir}/target/surefire-reports y los representa mediante DOXIA, que crea la versión de la interfaz web de los resultados de la prueba. Mapa vial. Objetivos de liberación.

5. **JDepend**

   JDepend es una herramienta que atraviesa directorios de archivos de clase de Java y genera métricas de calidad de diseño para cada paquete de Java. JDepend le permite medir automáticamente la calidad de un diseño en términos de su extensibilidad, reutilización y mantenibilidad para administrar las dependencias de los paquetes de manera efectiva.

Summary

[summary] [packages] [cycles] [explanations]

Package	TC	CC	AC	Ca	Ce	A	I	D	V
ExpositoTOP.src.es.ull.esit.utilities	3	3	0	1	5	0.0%	83.0%	17.0%	1
ExpositoTOP.src.es.ull.esit.utils	1	1	0	0	2	0.0%	100.0%	0.0%	1
ExpositoTOP.src.top	9	9	0	0	4	0.0%	100.0%	0.0%	1

6. **TagList**

   Escanea los archivos de origen en busca de etiquetas y genera un informe sobre sus ocurrencias.

7. **JavaDoc**

   El complemento Javadoc se utiliza para generar el documento java de nuestro proyecto. Podemos agregar el complemento maven javadoc agregando el código del complemento en el archivo pom. archivo xml.

8. **OWASP**

   La herramienta OWASP (Open Web Application Security Project) Dependency-Check es una herramienta de análisis de composición de software de código abierto que intenta escanear las dependencias de su proyecto de software y compararlas en busca de vulnerabilidades conocidas.

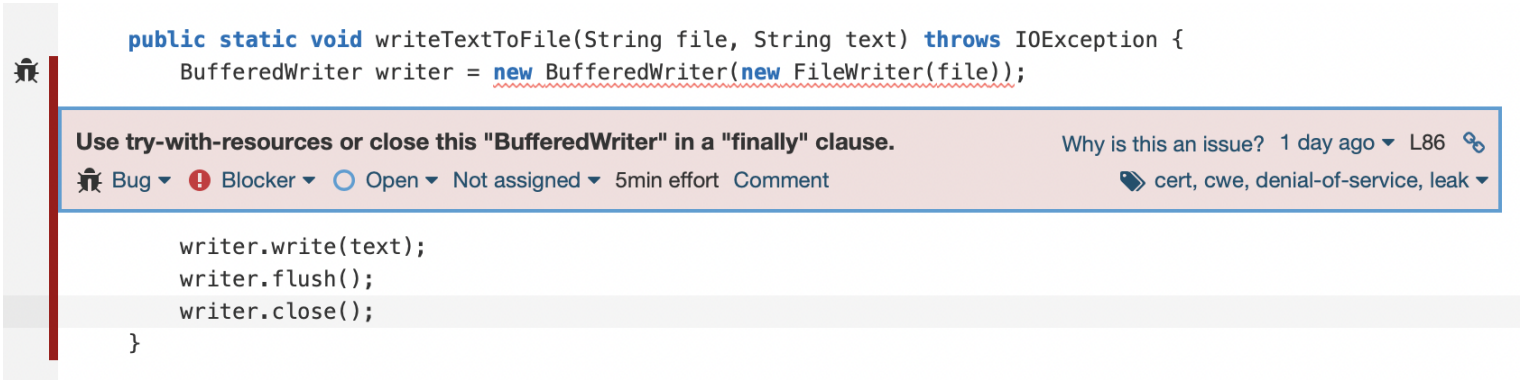
3. Identificación y Corrección de problemas de seguridad en el código.

Al combinar herramientas tales como SonarQube y Maven tenemos resultados que nos permiten evaluar nuestro código. En este caso, mediante la identificación de vulnerabilidades que proporciona SonarQube destacamos la siguiente.

Procedemos a solucionar el “Bug” de seguridad.

```
public int aleatorySelectionRCL(int maxTRCL) {
	SecureRandom r = new SecureRandom();
	int low = 0;
	int high = maxTRCL;
	int posSelected = r.nextInt(high-low) + low;
	return posSelected;
}
```

Por otro lado, también tenemos otros tipos de “Bugs” como por ejemplo:



Ahora, para solucionarlo tendremos que agregar las siguiente líneas de código.

```
public static void writeTextToFile(String file, String text) throws IOException {
     try ( BufferedWriter writer = new BufferedWriter(new FileWriter(file));){
         writer.write(text);
         writer.flush();
     }
 }
```

Para concluir, vemos que el uso de estas herramientas nos proporciona un conocimiento muy global, de tal manera que nos sirve para tener unas importantes nociones sobre cómo está estructurado el código y cómo se comunican los módulos de alto nivel (aquellos que llevan la lógica del negocio). De esta manera reducimos el tiempo que nos tomaría comprender el código que no hemos desarrollado nosotros.

4. Referencias.

1. https://maven.apache.org/plugins/index.html