DP2 2021-2022

LintReport

Acme Toolkits

Repositorio:

https://github.com/RingoML/Acme-Toolkits

Miembros:

- Caro Albarrán, Francisco Andrés (fracaralb@alum.us.es)
- Gallego Huerta, Alberto(albgalhue@alum.us.es)
- Martín Luque, José Manuel (josmarluq@alum.us.es)
- Reyes Madrid, Francisco (frareymad@alum.us.es)
- Sillero Manchón, Jorge (jorsilman@alum.us.es)

GRUPO E3.06

17 de marzo de 2022

Índice

Resumen ejecutivo

Este informe contendrá los posibles malos olores que reporte Lint sobre nuestro proyecto.

Tabla de revisiones

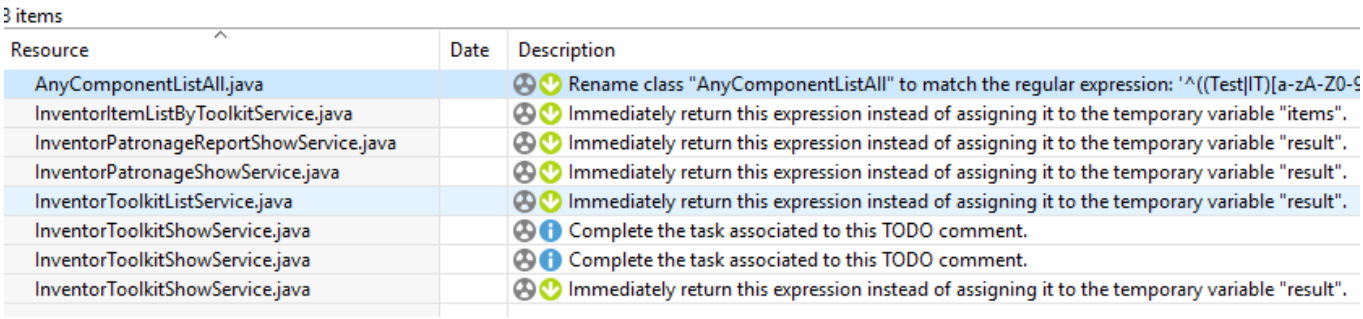
FECHA	VERSIÓN	DESCRIPCIÓN
2022-04-24	1	Versión inicial

Introducción

En este documento se va a detallar la información sobre malos olores de código que reporte Lint. Se justificarán los distintos tipos de malos olores si es necesario.

Contenido

El resultado de ejecutar SonarLint es el siguiente:



3 items

Resource	Date	Description
AnyComponentListAll.java		Rename class "AnyComponentListAll" to match the regular expression: '^((Test|IT)[a-zA-Z0-9
InventorItemListByToolkitService.java		Immediately return this expression instead of assigning it to the temporary variable "items".
InventorPatronageReportShowService.java		Immediately return this expression instead of assigning it to the temporary variable "result".
InventorPatronageShowService.java		Immediately return this expression instead of assigning it to the temporary variable "result".
InventorToolkitListService.java		Immediately return this expression instead of assigning it to the temporary variable "result".
InventorToolkitShowService.java		Complete the task associated to this TODO comment.
InventorToolkitShowService.java		Complete the task associated to this TODO comment.
InventorToolkitShowService.java		Immediately return this expression instead of assigning it to the temporary variable "result".

El primer olor ha sido corregido, ya que hemos renombrado la clase “AnyComponentListAll” como “AnyComponentListAllTest”.

Los siguientes 4 olores junto con el último, son de variables temporales que podríamos evitar crearlas y devolver directamente su valor.

A continuación, un ejemplo de cómo vamos a corregir todos los fallos de este tipo:

Antes:

```
final PatronageReport patronageReport = this.repository.findOnePatronageReportById(id);
final boolean result = patronageReport != null && patronageReport.getPatronage().getInventor().getId()==request.getPrincipal().getActiveRoleId();

return result;
```

Después:

```
return patronageReport != null && patronageReport.getPatronage().getInventor().getId()==request.getPrincipal().getActiveRoleId();
```

Por último, los olores marcados de azul los hemos solucionado borrando las siguientes dos líneas comentadas:

```
@Override
public boolean authorise(final Request<Toolkit> request) {
    // TODO Auto-generated method stub
    assert request != null;

    return true;
}

@Override
public Toolkit findOne(final Request<Toolkit> request) {
    // TODO Auto-generated method stub
    assert request != null;
    final int id  = request.getModel().getInteger("id");
    return this.repository.findOneToolkitById(id);

}
```

Conclusiones

SonarLint nos ha reportado 8 olores de código, los cuáles han sido todos corregidos y solucionados rápidamente. El uso de estas herramientas favorece tanto las buenas prácticas como una ejecución eficiente y un mejor aprovechamiento de los recursos disponibles. Tras volver a pasar SonarLint al proyecto, esta ya no reporta ningún mal olor de código.

Bibliografía

Intencionalmente en blanco.