LINT REPORT D03

INFORMACIÓN DE GRUPO

Group: C1.043

Repository: https://github.com/DaniFdezCab/DP2-2324-C1-043.git

Student #1
UVUS: danfercab
Contact: danfercab@alum.us.es

Student #2
UVUS: alvmarmun1
Contact: alvmarmun1@alum.us.es

Student #3
UVUS: pabberima
Contact: pabberima@alum.us.es

Student #4
UVUS: alepingar
Contact: alepingar@alum.us.es

Student #5
UVUS: fracapgar1
Contact: fracapgar1@alum.us.es

Date: Sevilla Mayo 27,02, 2024

TABLA DE CONTENIDO

RESUMEN EJECUTIVO

Este informe proporcionará una lista de problemas detectados por Sonar Lint en el proyecto, junto con explicaciones de por qué pueden no ser motivo de preocupación. Se utilizará un lenguaje claro y directo para asegurar una fácil comprensión y ayudar a garantizar un producto final de alta calidad.

TABLA DE REVISIONES

Revisión	Fecha	Descripción
1	2024-05-27	Se realizó el informe en su completitud.

INTRODUCCIÓN

Usaré el plugin SonarLint para Eclipse para analizar todos los archivos que he creado como Estudiante 4 para mis tareas individuales. Específicamente, revisaremos el contenido de los paquetes Java llamados "entities.sponsorships," "features.sponsor," "features.any.sponsorship," y "features.authenticated.sponsor." Adicionalmente, analizaremos el rol de Sponsor y el formulario SponsorDashboard. Los archivos de pobladores como Sponsorship, Invoice y Sponsor también serán analizados, junto con los archivos de vista (forms.jsp, list.jsp y archivos i18n) que se encuentran en las carpetas "views/sponsor," "views/any/sponsorship," y "views/authenticated/sponsor." Dado que la mayoría de los malos olores son comunes a varios archivos, analizaré olor a código por olor a código en lugar de clase por clase.

CONTENIDOS

Sobrescribir el método "equals" en esta clase

Esta sugerencia surge porque es una práctica común sobrescribir el método "equals" en las clases de Java. Sin embargo, dado que no se utiliza en nuestra implementación y ha sido comentado en la clase, no es necesario realizar ninguna corrección.

Eliminar el literal Booleano innecesario

Este mal olor se debe a las validaciones que verifican si una entidad está publicada: **"object.isPublished() == false"**. Es cierto que solo se puede escribir **"!object.isPublished()"**, pero dado que es parte de una aserción más grande, considero que es más legible con **== false**. En cualquier caso, esto solo ocurre un par de veces y creo que vale la pena una expresión un poco más larga por el bien de la comprensibilidad.

Usar una sintaxis concisa de clase de caracteres '\d' en lugar de '[0-9]'

Esta recomendación se refiere al código – como atributos. Sin embargo, la implementación actual que usa '[0-9]' en la expresión regular es válida y, posiblemente, más comprensible para representar rangos de números latinos. Por lo tanto, no es necesaria ninguna corrección.

Renombrar el nombre del paquete para que coincida con la expresión regular '^[a-z_]+(.[a-z_][a-z0-9_])$'

Aunque hay una sugerencia para renombrar el paquete, el nombre actual se adhiere al estándar del framework y sigue lo enseñado por los profesores. Por lo tanto, no se necesita ninguna corrección.

Definir una constante en lugar de duplicar este literal

Esta recomendación sugiere definir una constante en lugar de duplicar un valor literal varias veces en el código. Sin embargo, en este caso, duplicar el literal puede no ser un problema significativo. La justificación para su naturaleza inofensiva es que el literal se usa solo en unos pocos lugares (2 o 3 veces como máximo).

Reemplazar esta aserción con una verificación adecuada

La aserción **`object != null;`** está marcada, pero es la implementación recomendada tanto por los profesores como por el framework. Por lo tanto, no es necesaria ninguna modificación.

CONCLUSIONES

En conclusión, si bien el análisis de código ha señalado varias áreas potenciales de mejora, tras un examen más detenido, muchas de estas sugerencias se consideran inocuas o incluso preferibles según el contexto específico del proyecto. Por ejemplo, sugerencias como sobrescribir el método "equals", utilizar una sintaxis de clase de caracteres concisa y renombrar nombres de paquetes se consideran innecesarias, ya que se alinean con los estándares y prácticas del proyecto enseñados por los profesores. Además, las decisiones de mantener ciertas estructuras de código, como usar comparaciones booleanas explícitas o duplicar literales, están justificadas por su contribución a la legibilidad del código y debe decirse que solo ocurren un par de veces. En general, si bien es esencial considerar y evaluar las sugerencias de código, es igualmente importante sopesarlas con el contexto y los estándares del proyecto para determinar el curso de acción más apropiado. En este caso, las prácticas de código existentes parecen ser adecuadas para las necesidades del proyecto.

BIBLIOGRAFÍA

Intencionalmente en blanco.