Características de la herramienta PMD

Introducción:

La herramienta PMD es un analizador de código fuente estático de código abierto, e informa de los problemas encontrados dentro del código fuente de una aplicación. Informa principalmente de malos hábitos de programación y de partes del código que son ineficientes. Su análisis está basado en reglas, es capaz de analizar una gran variedad de lenguajes de programación e integra con multitud de herramientas. No requiere apenas de infraestructura y es gratuita.

Características:

PMD es un analizador de código estático que aplica reglas de validación al código que escribimos. Por defecto PMD ofrece multitud de reglas para cada uno de los lenguajes que soporta, siendo las reglas definidas de calidad. Permite también definir reglas propias, lo que facilita enormemente automatizar comprobaciones en el código que no estén recogidas en las reglas iniciales. Es muy cómodo de usar, convierte un proceso que sería manual (y muy tedioso) en uno automático.

PMD se puede usar de cuatro formas:

1. Mediante línea de comandos (siendo su uso poco amigable).
2. Integrado en un entorno de desarrollo como Visual Studio, Welkin Suite, Intellij, Eclipse (es como usaremos PMD), etc.
3. Siendo parte de un entorno de Integración Continua que tenga un gestor como Maven, Ant, Gradle, etc.
4. Formando parte de un producto comercial en Cloud u OnPremise como Codacy, Code-Scan, etc.

Características más relevantes:

- Requiere de un listado de reglas que alguien ha definido y programado previamente para detectar incumplimientos.
- Las reglas detectan situaciones que queremos anunciar, indicando si se trata de un error, una advertencia, una información… Y un valor de prioridad.
- PMD en una estructura de ficheros examina fichero a fichero el código fuente y va aplicando las reglas, de estas se obtiene como resultado incumplimientos a las mismas.
- El término Ruleset hace referencia al conjunto de reglas definido. El éxito que podamos tener en el uso de PMD depende directamente de la calidad de las reglas, de las que hayamos habilitado para el proyecto actual, y del proceso de mejora continua que hayamos pensado para las reglas.
- El incumplimiento a las reglas definidas se puede obtener en diversos formatos: fichero de texto plano, XML, HTML, en una herramienta…

Nos centraremos en el lenguaje de programación Java para proporcionar un ejemplos de Ruleset por defecto.

Las reglas para el lenguaje de programación Java se clasifican en varias temáticas:

- Best Practices
    - Son reglas que hacen cumplir buenas prácticas de programación que son aceptadas.
    - Un ejemplo de regla para esta temática es:
        - MissingOverride => Anotar los métodos sobrescritos con @Override asegura que en tiempo de compilación el método realmente sobrescribe uno, lo que ayuda a refactorizar y aclara la intención.
        - Esta regla se incumple si un método que se sobrescribe no lleva la anotación @Override.
- Code Style
    - Son reglas que hacen cumplir un estilo de codificación especifico.
    - Un ejemplo de regla para esta temática es:
        - AtLeastOneConstructor => Cada clase que no es estática debe declarar al menos un constructor. Las clases con elementos solamente estáticos se ignoran.

- Design
    - Son reglas que te ayudan a descubrir fallos de diseño.
    - Un ejemplo de regla para esta temática es:
        - TooManyMethods => Una clase con demasiados métodos es probablemente candidata para ser refactorizada, con el fin de reducir su complejidad y encontrar un camino para tener objetos de granularidad más fina.
- Documentation
    - Son reglas relacionadas con la documentación del código.
    - Un ejemplo de regla para esta temática es:
        - CommentSize => Determina si las dimensiones de los comentarios sin encabezado encontrados están dentro de los límites especificados.
        - Se pueden determinar estos límites.
- Error Prone
    - Son reglas para detectar construcciones rotas, muy confusas o propensas a errores en tiempo de ejecución.
    - Un ejemplo de regla para esta temática es:
        - AvoidCatchingNPE => El código nunca debería lanzar NullPointerException en circunstancias normales. Un bloque catch debe ocultar el error original, causando otros problemas más sutiles más adelante.
- Multithreading
    - Son reglas que indican problemas al tratar con múltiples hilos de ejecución.
    - Un ejemplo de regla para esta temática es:
        - AvoidSynchronizedAtMethodLevel => La sincronización a nivel de método puede causar problemas cuando nuevo código es añadido al método. La sincronización a nivel de bloque ayuda a asegurar que solo el código que necesita sincronización la obtenga.
- Performance
    - Son reglas que detectan código que no es óptimo.
    - Un ejemplo de regla para esta temática es:
        - AddEmptyString => La conversión de literales a strings concatenándolos con strings vacíos es ineficiente. Es mucho mejor usar uno de los métodos toString() para cada tipo específico en su lugar.
        - Se entiende que si uno quiere pasar un entero a string se use el método toString() de la clase Integer en lugar de hacer: String ejemplo= entero+””; donde entero es un número del tipo primitivo int o un objeto de la clase Integer.

- Security
    - Son reglas que detectan potenciales defectos de seguridad.
    - Un ejemplo de regla para esta temática es:
        - HardCodedCryptoKey => No utilice valores fuertemente codificados para operaciones criptográficas. Guarde las claves fuera del código fuente por favor.
        - Esto quiere decir que no es recomendable almacenar claves de seguridad en el propio código fuente ya que vulnera la misma.

Hay más reglas por defecto, pero están obsoletas y no pertenecen a una temática clara.

Las mostradas anteriormente solo son una pequeña parte del total de reglas que trae por defecto la herramienta dentro de cada temática.

El potencial de esta herramienta reside en la cantidad y calidad de las reglas definidas por defecto y más aún en la posibilidad de añadir más reglas sin límite según las necesidades del desarrollador o empresa.

Por supuesto se pueden observar el resto de reglas que ofrece la herramienta por defecto a través del siguiente enlace:

Java Rules | PMD Source Code Analyzer