SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Curso

TÉCNICO EM DESENVOLVIMENTO

DE SISTEMAS

Pesquisa Métodos equals e hashCode e o uso de Lombok

Ana Carolina Castro Ribeiro

Sorocaba
Novembro – 2024

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Ana Carolina Castro Ribeiro

Pesquisa Métodos equals e hashCode e o uso de Lombok

Pesquisa e levantamento de
Conhecimento (Métodos equals e
hashCode e o uso de Lombok)
Prof. – Emerson Magalhães, Daniel Cintori

Sorocaba
Novembro – 2024

SUMÁRIO

Métodos equals e hashCode e o uso de Lombok

INTRODUÇÃO

Os métodos equals e hashCode são elementos essenciais no desenvolvimento em Java, pois eles promovem uma correta comparação e manipulação de objetos.

Estes métodos, herdados da classe Object, direcionam o comportamento de coleções como: HashSet e HashMap. Além de terem um papel fundamental em frameworks, como por exemplo o Spring, uma vez que gerenciam entidades para operações de persistência e caching.

Nisto surge a biblioteca Lombok, uma ferramenta que visa simplificar a implementação desses métodos, automatizando tarefas e reduzindo o código.

Dessa forma, essa pesquisa tem como objetivo promover o aprendizado da importância e o funcionamento dos métodos equals e hashCode, vendo no que consistem, analisando suas aplicações práticas, e demonstrando como o Lombok pode facilitar o desenvolvimento em Java, mostrando vantagens e desvantagens.

1. Desenvolvimento

Inicialmente, os métodos equals e hashCode são métodos herdados da classe Object em Java e desempenham papéis cruciais na comparação de objetos e na determinação de suas posições em coleções baseadas em hashing, como HashMap e HashSet.

Eles possuem grande relevância em frameworks como Spring, pois uma vez que é feito a implementação correta desses métodos é possível o uso das funcionalidades como caching, além da persistência e verificação de integridade de entidades.

Por fim a Lombok nada mais é do que uma biblioteca que reduz o código boilerplate em Java por meio de anotações, simplificando a criação de métodos comuns, como equals, hashCode, toString, etc.

1.1. Fundamentos Teóricos

Esses métodos possuem um contrato fundamental que rege seu funcionamento. De acordo com o contrato:

- Se dois objetos são iguais de acordo com o método equals, eles devem ter o mesmo valor de hashCode.

Ademais a implementação de equals deve respeitar os seguintes princípios:

- Reflexividade, simetria, transitividade e consistência para equals;
- Consistência e a regra que se equals retornar false, hashCode não precisa necessariamente ser diferente.

Já o método hashCode deve fornecer um valor constante para um objeto durante a execução do programa, desde que os dados usados em sua implementação não sejam modificados. Entretanto, uma implementação incorreta desses métodos pode levar a alguns problemas, como a presença de elementos duplicados em coleções ou a impossibilidade de recuperar objetos ou a busca correta por um deles.

1.2. Utilização Prática em Coleções e no Spring

No contexto das coleções Java, HashSet e HashMap dependem dos métodos equals e hashCode para armazenar e recuperar objetos.

Como exemplo há a criação de uma classe Pessoa onde a implementação correta dos métodos assegura que não haverá qualquer duplicidade de elementos em um HashSet. Se dois objetos Pessoa diferentes, mas com as mesmas propriedades, forem inseridos em uma HashSet, o uso adequado de equals e hashCode garantirá que apenas um objeto será armazenado.

Agora, em frameworks como o Spring, esses métodos também são fundamentais para a gestão de entidades em operações de persistência e caching. Quando o Spring lida com entidades ele precisa verificar a igualdade de objetos para evitar duplicidade, assegurando a integridade dos dados. Ao ser inserido corretamente é evitado problemas na falha em identificar objetos idênticos em listas de resultados e a duplicação de entradas em caches de entidades.

Exemplo com HashSet e HashMap:

```java
import java.util.HashSet;
import java.util.Objects;

public class Pessoa {
    private String nome;
    private int idade;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Pessoa pessoa = (Pessoa) o;
        return idade == pessoa.idade && Objects.equals(nome, pessoa.nome);
    }

    @Override
    public int hashCode() {
        return Objects.hash(nome, idade);
    }
}
```

FIGURA 1- Código exemplificando HashSet e HashMap

Esse exemplo mostra uma implementação manual de equals e hashCode.

1.3. Lombok: Simplificação do Código

A biblioteca Lombok é muito utilizada para reduzir o código em projetos Java, facilitando a escrita e a manutenção do código. Através de anotações como: @EqualsAndHashCode e @Data, o Lombok gera automaticamente os métodos equals e hashCode.

A anotação @EqualsAndHashCode permite que o desenvolvedor fale quais campos devem ser incluídos na comparação, já a anotação @Data inclui @EqualsAndHashCode, além de outros métodos como toString e getters/setters.

Como exemplo pode-se imaginar mais uma vez a classe Pessoa que, sem Lombok, exigiria a implementação manual de equals e hashCode. Com o uso de @EqualsAndHashCode, o código é simplificado, reduzindo erros comuns e garantindo eficiência.

Exemplo com Lombok:

```
import lombok.Data;

@Data
public class Pessoa {
    private String nome;
    private int idade;
}
```

FIGURA 2- Código exemplificando Lombok

Com a anotação @Data, a classe Pessoa terá os métodos equals e hashCode gerados automaticamente.

1.4. Vantagens e Desvantagens de Usar Lombok

1.4.1. Vantagens:

- Reduz o código boilerplate, possuindo menos código, e possibilitando menor chance de erros;

- Contribui na legibilidade, tornando o código mais limpo e fácil de manter.

1.4.2. Desvantagens:

- Existe a dependência externa, sendo necessário incluir Lombok como uma dependência;
- Cria dificuldades em debugging e análise de código gerado.

É muito importante na hora da implementação do Lombok que os desenvolvedores avaliem a complexidade do projeto. Com isso o uso de Lombok deve ser balanceado, preferindo este quando a manutenção do código e a simplicidade são fundamentais. Entretanto, em projetos críticos, é importante avaliar os riscos de dependências externas e a transparência do código.

CONCLUSÃO

A implementação correta dos métodos equals e hashCode é de suma importância para garantir o funcionamento adequado de coleções e a integridade em frameworks como o Spring.

Já o uso de Lombok oferece uma forma de simplificar e automatizar a criação desses métodos, deixando um código mais limpo e de fácil manutenção.

No entanto, é importante considerar tanto suas vantagens, quanto os seus obstáculos no desenvolvimento dos códigos. Por isso é tão importante uma análise para decidir entre a implementação manual e o uso de Lombok. Assim será possível o desenvolvimento de aplicações Java mais eficientes e escaláveis.

BIBLIOGRAFIA

- https://blog.algaworks.com/entendendo-o-equals-e-hashcode/
- https://angeliski.com.br/equals-e-hashcode?x-host=angeliski.com.br
- https://www.devmedia.com.br/sobrescrevendo-o-metodo-hashcode-em-java/26488
- https://www.youtube.com/watch?v=T008B4vURk4
- https://www.youtube.com/watch?v=SlC1thsNauU
- https://www.dio.me/articles/como-usar-o-lombok-em-projetos-java
- https://pt.stackoverflow.com/questions/141294/o-que-%C3%A9-o-lombok
- https://www.devmedia.com.br/uma-visao-sobre-o-projeto-lombok/28321