SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Curso

TÉCNICO EM DESENVOLVIMENTO

DE SISTEMAS

Introdução aos métodos equals, hashCode e uso de Lombok

Gustavo Machado Barrinha

Sorocaba
Novembro – 2024

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Gustavo Machado Barrinha

Introdução aos métodos equals, hashCode e uso de Lombok

Trabalho solicitado pelo professor
com foco nos métodos equals e
HashCode, e uso de lombok.

Prof. – Emerson Magalhães

Sorocaba
Novembro – 2024

HISTÓRICO DE VERSÕES

Data	Versão	Descrição	Autor
09/11/2024	1.0	Trabalho solicitado pelo professor com foco nos métodos equals e HashCode, e uso de lombok.	Gustavo M. Barrinha

SUMÁRIO

OBJETIVO

Em Java, os métodos equals e hashCode são essenciais para definir como os objetos de uma classe são comparados e manipulados em coleções e frameworks. equals determina se dois objetos são logicamente iguais, enquanto hashCode retorna um valor numérico (o código hash) que representa o objeto. Juntos, esses métodos desempenham um papel crítico em coleções baseadas em hashing, como HashSet e HashMap, que dependem da correta implementação desses métodos para garantir o comportamento esperado.

INTRODUÇÃO

Nas coleções Java, especialmente aquelas baseadas em hashing, como HashMap e HashSet, equals e hashCode determinam onde e como os objetos são armazenados e recuperados. No contexto de frameworks como o Spring, a correta implementação desses métodos influencia diretamente operações de cache e persistência. Por exemplo, ao salvar uma entidade em um banco de dados com Hibernate (ORM utilizado no Spring), a implementação de equals e hashCode garante que as operações de comparação entre entidades funcionem corretamente, evitando inconsistências de dados e duplicações.

1. Fundamentos Teóricos

1.1. Relações entre equals e hashCode

Se dois objetos são iguais de acordo com equals, então o valor retornado por hashCode também deve ser igual para ambos os objetos.

Se dois objetos não são iguais de acordo com equals, eles podem ou não ter o mesmo hashCode, mas ter hash codes diferentes melhora o desempenho em coleções baseadas em hashing.

1.2. Regras entre implementação de equals e hashCode

É importante que hashCode seja consistente: se não houver alterações nos atributos usados para gerar o hash, ele deve retornar o mesmo valor para chamadas subsequentes.

1.3. Impacto da implementação de equals e hashCode em entidades de aplicações Java

=====Coleções=====

Em coleções como HashSet e HashMap, o hashCode de um objeto é usado para determinar em qual "bucket" ou "compartimento" o objeto será armazenado. Depois, equals é usado para verificar se um objeto já existe no bucket, prevenindo duplicações. Assim, uma implementação inconsistente de equals e hashCode pode levar a comportamentos inesperados, como a existência de duplicatas ou dificuldades para encontrar objetos nas coleções.

=====Java=====

A implementação correta de equals e hashCode é essencial em entidades de aplicações Java para garantir a consistência em operações de comparação e armazenamento. Em aplicações corporativas, onde a integridade e a recuperação de dados são cruciais, a falta de uma implementação correta pode causar problemas significativos de desempenho e confiabilidade.

2. Utilização Prática

2.1. Exemplos práticos de equals e hashCode, inseridos em coleções como HashSet e HashMap

Considere uma classe Pessoa com dois atributos, nome e idade. Se essa classe não implementa equals e hashCode, uma coleção HashSet<Pessoa> poderia acabar armazenando duplicatas porque o HashSet usa hashCode para determinar onde armazenar o objeto e equals para verificar igualdade.

==========

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

    // getters, setters, e outros métodos
}
```

==========

Sem equals e hashCode, o Java usa a implementação padrão desses métodos na classe Object, o que significa que dois objetos Pessoa com os mesmos valores para nome e idade ainda seriam considerados diferentes. Abaixo está uma implementação correta:

==========

```
@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 && nome.equals(pessoa.nome);
}

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

==========

2.2. Nome e idade, SEM duplicatas.

Em frameworks como Spring, a implementação de equals e hashCode em classes de entidade é importante para operações de persistência. O Hibernate, por exemplo, usa esses métodos para comparar entidades no contexto de operações de persistência e cache. Um exemplo prático é uma classe Produto em um sistema de e-commerce, onde equals e hashCode garantem que produtos com o mesmo ID sejam tratados como iguais no banco de dados e no cache.

```
@Entity
public class Produto {
    @Id
    private Long id;
    private String nome;
    private double preco;

    // equals e hashCode baseados no ID
}
```

---

3. Introdução ao Lombok

O Lombok reduz a quantidade de código repetitivo em projetos Java, especialmente em classes com muitos atributos e métodos utilitários. Ele faz isso gerando automaticamente os métodos e eliminando a necessidade de escrever manualmente equals e hashCode, entre outros.

3.1. Anotações Lombok

O Lombok oferece a anotação @EqualsAndHashCode que gera os métodos equals e hashCode com base em todos os campos da classe, ou em campos específicos, se definidos. A anotação @Data também é popular, pois gera automaticamente equals, hashCode, toString, getters e setters, ideal para classes de dados.

3.2. Vantagens

O Lombok minimiza o código repetitivo, aumentando a legibilidade e facilitando a manutenção.

3.3. Desvantagens

Dependência de uma biblioteca externa pode ser um problema em projetos que visam ter o mínimo de dependências. Além disso, o uso de Lombok pode dificultar o processo de depuração, pois o código gerado não está visível diretamente no código-fonte.

Em ambientes de produção, é fundamental avaliar se o uso de Lombok é adequado. Em sistemas que exigem alto controle sobre o código, uma implementação manual pode ser preferível.

CONCLUSÃO

O uso correto de equals e hashCode em Java é essencial para garantir a integridade e o desempenho em coleções e frameworks como Spring. O Lombok, com anotações como @EqualsAndHashCode e @Data, simplifica essa implementação, mas deve ser utilizado com cautela, considerando as vantagens e limitações que traz. Entender os benefícios e os desafios desses métodos contribui para um código mais eficiente e escalável, otimizando o desenvolvimento em Java.

BIBLIOGRAFIA

https://codegym.cc/pt/groups/posts/pt.264.metodos-equals-e-hashcode-praticas-recomendadas

https://www.jdevtreinamento.com.br/metodos-equals-e-hashcode/

https://www.treinaweb.com.br/blog/projeto-lombok-acelerando-o-desenvolvimento-java

https://imasters.com.br/back-end/projeto-lombok-escrevendo-menos-codigo-em-java