SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Curso

TÉCNICO EM DESENVOLVIMENTO

DE SISTEMAS

Métodos equals e hashCode em Java e o uso de Lombok.

Ana Julia Sanches de Souza

Cidade
Mês – Ano

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Ana Julia Sanches de Souza

Métodos equals e hashCode em Java e o uso de Lombok.

Métodos equals e hashCode em Java e o uso de Lombok para otimização de códigos.

Prof. – Emerson

Sorocaba
Novembro – 2024

Métodos equals e hashCode em Java e o uso de Lombok.

OBJETIVO

O objetivo desse projeto é explorar a importância e o funcionamento dos métodos equals e hashCode em Java, explicando como eles afetam o comportamento de coleções como o hashMap e hashSet, e como são aplicados no gerenciamento de entidades em frameworks como o Spring.

De mesma forma, será explorado a biblioteca Lombok, e como a mesma pode tornar a implementação desses métodos mais simples e melhorar a produtividade no desenvolvimento.

INTRODUÇÃO

O método hashCode é de extrema importância para a formação de tabelas de dispersão (empalhamento), cujo dados são armazenados de acordo com o cálculo de um número hash, a partir das especificações de determinada informação, ou seja, é muito simples procurar um dado especifico em meio a muitos outros dados. Assim, torna-se fácil recuperar essas informações, pois mesmo que os dados colidam, o espaço de busca é reduzido.

Quando há a colisão dos dados, ou seja, quando duas ou mais informações tem características semelhantes, é utilizado o método equals. O método equals consiste em basicamente determinar se esses objetos são iguais e se possuem o mesmo valor dentre seus atributos congêneres.

Esses métodos são de suma importância para coleções e frameworks, pois os mesmos dependem dessa comparação entre dados para que a operação a ser realizada seja feita de forma mais eficiente possível, e vale a pena mencionar que ambos serão mais funcionais ao serem utilizados juntos.

Um exemplo seria o uso desses métodos no Spring Data, cuja necessidade de realizar comparações de forma simples entre determinadas informações de diferentes objetos (entidades) é muito comum.

1. Fundamentos Teóricos

Explicação do contrato entre equals e hashCode.

1.1. Regras de Implementação

Algumas das regras que governam a implementação dos métodos equal e hashCode são: quando objetos são iguais eles devem apresentar o mesmo valor em seu hashCode, o método equals deve sempre retornar “true” e em caso de comparação, se um objeto retorna “true” o outro também deve retornar esse valor.
Caso o equals retorne o valor “false”, é indicado que os valores entre os objetos sejam distintos, e o valor que foi retornado pelo hashCode deve ser o mesmo enquanto o objeto existir. Caso seja modificado de modo a alterar de forma conjunta o equals e hashCode, a possibilidade de o dado em questão ficar inacessível e ser de difícil conhecimento sua localização entre as coleções e frameworks.

.

1.2. Como o contrato entre equals e hashCode afeta o comportamento das coleções (ex.: HashMap, HashSet).

Os métodos equals e hashCode são importantes para o comportamento das coleções, pois é por meio deles que um objeto pode ser adicionado a um hashSet e caso exista algum outro dado de valor semelhante pode ser utilizado o método equals para garantir que não trata-se do mesmo objeto.

Em outras palavras, caso sejam adicionados dois objetos considerados de mesmo valor pelo equals, só sera permitido a duplicação em caso de concordância entre hashCode e o equals, e não exclusivamente pelo hashSet.

Para o hashMap, o hashCode é utilizado como um localizador para identificar se o objeto retornado é de fato o objeto requisitado. Assim, se houver dois localizadores com o mesmo valor em seu hashCode, mas com diferentes

características é usado o método equals para garantir que se trata de localizadores de valor igual realmente.

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

É de extrema importância que a implementação de equals e hashCode em entidades de aplicações Java seja feita de maneira correta, pois trata-se de um fator fundamental para a integridade das informações sejam preservadas em contexto de coleções e frameworks como o Spring Data.

Diversas coleções são integralmente dependentes de tais métodos para armazenar e realizar a comparação de determinados objetos de forma conveniente.

Um exemplo dessa má implementação seria a duplicação de objetos ou localização incorreta do mesmo, já que o hashCode é responsável por determinar o local no qual o objeto será armazenado e o equals por realizar essa comparação de mesmos valores de hashCode.

Quando esses métodos são utilizados, a comparação das informações pode acabar por tornar-se dependente de tais. Se esses não forem implementados de forma correta, ocorrerá certa dificuldade no momento de manusear tais dependências. Também há o risco de dados acabarem se cruzando o que torna o processo lento e menos eficiente.

2. Utilização Prática em Coleções Java e no Spring.

2.1. Exemplo do método equals.

```java
public class Pessoa {
  String nome;
  String sobrenome;
  String idade;
  @Override
  public boolean equals(Object obj) {
    if (!(obj instanceof Pessoa))
      return false;
    Pessoa other = (Pessoa) obj;
    if (!this.nome.equals(other.nome) || !this.idade.equals(other.idade))
      return false;
    return true;
  }
}
```

Pessoa.java hosted with ❤ by GitHub view raw

2.2. Exemplo do método equals e hashCode.

```java
import java.util.List;
public class BuscaPessoa {
  public PessoaFisica buscaPessoaPorCpf(List pessoas, String cpf) {
    PessoaFisica pessoaFisica = new PessoaFisica();
    pessoaFisica.cpf = cpf;
    int indexOfPessoa = pessoas.indexOf(pessoaFisica);
    if (indexOfPessoa > 0) {
      return pessoas.get(indexOfPessoa);
    }
    return null;
  }
  class PessoaFisica {
    String cpf;
    @Override
    public int hashCode() {
      final int prime = 31;
      int result = 1;
      result = prime * result + getOuterType().hashCode();
      result = prime * result + ((cpf == null) ? 0 : cpf.hashCode());
      return result;
    }
```

```java
    @Override
    public boolean equals(Object obj) {
      if (this == obj)
        return true;
      if (obj == null)
        return false;
      if (getClass() != obj.getClass())
        return false;
      PessoaFisica other = (PessoaFisica) obj;
      if (!getOuterType().equals(other.getOuterType()))
        return false;
      if (cpf == null) {
        if (other.cpf != null)
          return false;
      } else if (!cpf.equals(other.cpf))
        return false;
      return true;
    }
    private BuscaPessoa getOuterType() {
      return BuscaPessoa.this;
    }
  }
}
```

BuscaPessoa.java hosted with ❤ by GitHub view raw

3. Introdução a Lombok.

A biblioteca Lombok ajuda a aumentar a produtividade do desenvolvimento e a reduzir o código do projeto que muitas vezes pode ser repetitivo.

Ela permite que que sejam adicionadas anotações que realizam o trabalho desejado, o que acaba por gerar métodos construtores como e outros como os getters e setters, contribuindo assim para a otimização do código de forma mais simplificada, ou seja, ela ajuda a evitar erros que ocorrem ao escrever métodos como *equals* e *hashCode* manualmente que pode levar a problemas de desempenho e dificultar na procura desses erros. Assim, ao deixar a programação mais organizada, o desenvolvedor pode focar mais nos processos lógicos da classe em questão.

Urge também a necessidade de utilizar dependências tanto por via Maven ou Gradle, pois traz diversos benefícios principalmente em projetos maiores, cuja a manutenção pode se tornar mais dificil.

3.1. Análise das anotações @EqualsAndHashCode e @Data.

@EqualsAndHashCode: é usado para simpificar a implementação dos métodos equals e hashCode em uma classe Java. Ao serem implementados, tais métodos devem seguir a risca suas regras para evitar problemas em coleções que utilizam hashing, como HashMap e HashSet.
A anotação @EqualsAndHashCode do Lombok, gera os métodos de forma automatica, o que torna o processo de desenvolvimento mais simples e eficaz.

Já a anotação @Data é utilizada para gerar de forma automática os métodos getters e setters, equals e hashCode. Dessa forma, ao invés de escrever de forma repetitiva métodos manuais é possivel apenas utilizar tal anotação, reduzindo assim o código de forma simples, facilitando sua manutenção e diminuindo a probablidade de erros no projeto.

3.2. Exemplo prático de implementação de uma entidade com Lombok, comparando com uma implementação manual.

```java
import java.util.Objects;

public class Usuario {
    private Long id;
    private String nome;
    private String email;

    // Getters e Setters
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
```

```java
    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    // equals
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Usuario usuario = (Usuario) o;
        return Objects.equals(id, usuario.id) &&
               Objects.equals(nome, usuario.nome) &&
               Objects.equals(email, usuario.email);
    }
```

```java
    }

    // hashCode
    @Override
    public int hashCode() {
        return Objects.hash(id, nome, email);
    }

    // toString
    @Override
    public String toString() {
        return "Usuario{" +
               "id=" + id +
               ", nome='" + nome + '\'' +
               ", email='" + email + '\'' +
               '}';
    }
}
```

4. Vantagens e Desvantagens de Usar Lombok para equals e hashCode.

4.1. Vantagens:

Redução de Código Boilerplate: Eliminação do código repetitivo, conhecido como "boilerplate". Métodos como getters, setters não contribuem diretamente para a lógica e o Lombok permite gerar esses métodos automaticamente por meio de anotações, o que facilita um desenvolvimento mais leve para o programador.

Aumento da Produtividade: Com menos tempo gasto em tarefas mecânicas os programadores podem se concentrar mais na lógica do negócio, o que pode ser vantajoso pois aumenta a produtividade do desenvolvedor.

Melhor Legibilidade e Manutenção: Um código mais curto e limpo é mais fácil de entender. Com o Lombok a legibilidade do código é aprimorada, o que facilita tanto a revisão quanto a manutenção, além de evitar erros de digitação.

4.2. Desvantagens:

Dependência Externa: Ao adicionar o Lombok ao seu projeto, você está incorporando uma biblioteca externa. Isso pode ser um problema, pois algumas bibliotecas tem certas limitações e restrições o que pode afetar a compatibilidade com o Java.

Dificuldade no Debugging: O lombok gera código automaticamente durante a compilação, o que faz com que os métodos gerados não estejam visíveis no código-fonte, ou seja, o desenvolvedor pode ter dificuldade em localizar a origem do problema, já que o código real não é exibido diretamente no ambiente de desenvolvimento.

CONCLUSÃO

A implementação correta dos métodos equals e hashCode em Java é essencial, pois os mesmos asseguram a comparação eficaz de objetos e a organização correta de dados, melhorando a eficiência das operações.

A utilização da biblioteca Lombok pode simplificar a implementação desses métodos, reduzindo a quantidade de código repetitivo e aumentando a produtividade do desenvolvedor, ao gerar automaticamente os métodos com anotações como @EqualsAndHashCode, tornando o código mais limpo e fácil de manter. Entretanto, podem haver problemas na compatibilidade entre a biblioteca, pois ao depender de bibliotecas externas pode haver a não concordância entre versões.

BIBLIOGRAFIA

https://angeliski.com.br/equals-e-hashcode?x-host=angeliski.com.br

https://blog.algaworks.com/entendendo-o-equals-e-hashcode/

https://www.dio.me/articles/como-lombok-pode-transformar-seu-codigo-java

https://pt.linkedin.com/advice/1/what-benefits-drawbacks-using-lombok-generate?lang=pt

https://artefatox.com/annotations-do-lombok/#:~:text=A%20annotation%20%40EqualsAndHashCode%20gera%20automaticamente,e%20ter%C3%A3o%20o%20mesmo%20hashcode.

https://artefatox.com/annotations-do-lombok/#:~:text=A%20annotation%20%40EqualsAndHashCode%20gera%20automaticamente,e%20ter%C3%A3o%20o%20mesmo%20hashcode.

https://www.dio.me/articles/como-lombok-pode-transformar-seu-codigo-java

https://www.devmedia.com.br/uma-visao-sobre-o-projeto-lombok/28321

https://rdrblog.com.br/java/introducao-ao-lombok/