From: Victor Tortorello Neto <vtneto@gmail.com>
To: Ferrero Rocher <arthurseveral@yahoo.com.br>
Subject: Re: Dúvida no Hibernate
Date: Wed, 21 Sep 2011 10:43:13 -0300

Entendo, Arthur... Na realidade, você está certo: o bom é usar um framework como o Spring. Vou estudar para ver se implemento.

Só uma dúvida simples: qual é a diferença entra utilizar o inner join ou o inner join fetch no Hibernate?

Abraços,

Victor

Em 20 de setembro de 2011 20:32, Ferrero Rocher <arthurseveral@yahoo.com.br> escreveu:
Olá Victor,

Isso é um comportamento arriscado, pois depende da implementação. A especificação diz

It is implementation-dependent as to whether callback methods are invoked before or after the cascading of the lifecycle events to related entities. Applications should not depend on this ordering.

É específico da implementação se métodos de callback devem ser invocados antes ou após o cascateamento dos eventos do ciclo de vida. Applicações não devem depender dessa ordem.

Embora havia funcionado na implementação do Hibernate, o mesmo não é válido para a implementação de referência TopLink. Eu já havia testado. Quando digo que isso é um comportamento arriscado deve-se ao fato que se um desenvolvedor prover manutanção ao sistema e, digamos que por questões de performance, ele decida usar uma outra implementação que não funcione como no Hibernate, o mesmo pode ficar a ver navios. Aí ele sai por aí como muitos iniciantes em Java dizendo que a linguagem é um #&@$ e por aí vai.

Na verdade, esses eventos foram projetados para lidar com questões específicas da aplicação, como validação, conversão e assim por diante. Para o propósito de lidar com questões além da aplicação, como logging, segurança etc, bem como o problema de configurar a chave-primária, como é o seu caso, os EJB's foram designados para tal tarefa.

Seria mais ou menos assim: O cascateamento para a classe filho seria removido, uma vez que, para estratégia de acesso via campo, o callback @PostPersist é arriscado como demonstrado acima. Posteriormente, vc criaria um EJB que agisse como um DAO, como a seguir

```
@Stateless

public class PaiDAO implements DAO<Pai> {


    @PersistenceContext
    private EntityManager em;



    public void add(Pai pai) {
        em.persist(pai);
    }



}
```

Logo após, vc atribuiria um interceptador ao método add, o qual percorreria a lista de filhos e configuraria cada chave-composta, persistindo cada entidade. Não lembro de cor como se configura um

interceptador com EJB. A mesma estratégia poderia ser usada se vc usasse Spring, Google Guice e assim por diante.

Se os getter's e setter's são indesejáveis para a sua aplicação, use Groovy

@Entity

class Pai implements Serializable {

@Id

Integer id;

}

Quando esse código rodar, getter's e setter's são automaticamente adicionados. Se vc quiser brincar com Groovy, use sua interface online

Atenciosamente
Arthur Ronald F D Garcia

De: Victor Tortorello Neto <vtneto@gmail.com>

Para: Ferrero Rocher <arthurseveral@yahoo.com.br>

Enviadas: Terça-feira, 20 de Setembro de 2011 15:43

Assunto: Re: Dúvida no Hibernate

Arthur,

Já pensou em utilizar o @PostPersist no pai para setar os valores dos filhos? (ou o @PrePersist nos filhos?)

Eu estava fazendo umas modificações aqui e funcionou...

Abraços,

Neto

Em 26 de agosto de 2011 02:22, Ferrero Rocher <arthurseveral@yahoo.com.br> escreveu:
Bem, se for do seu interesse, vc pode evitar @IndexColumn do Hibernate, bastando para isso criar um método auxiliar que retorne suas instâncias de acordo com o índice do Filho na lista como a seguir

```
public class Pai {

    @Transient
    public List<Filho> getFilhoListOrederedByIndex() {
        Collections.sort(filhoList, new Comparator<Filho>() {
        public int compare(Filho o1, Filho o2) {
                    // Integer implements Comparable<T>
                    return o1.getId().getIndex().compareTo(o2.getId().getIndex());
                }
        });

        return this.filhoList;
    }
```

Puro JPA e funciona em ambas as versões: 1.0 e 2.0

--- Em sex, 26/8/11, Victor Tortorello Neto <vtneto@gmail.com> escreveu:

De: Victor Tortorello Neto <vtneto@gmail.com>
Assunto: Re: Dúvida no Hibernate
Para: "Ferrero Rocher" <arthurseveral@yahoo.com.br>

Data: Sexta-feira, 26 de Agosto de 2011, 1:31

Deu certo!

Agora sim acho que ficou consistente.

O único porém é que tive que utilizar a coluna @IndexColumn que é específica do Hibernate, já que não existe na especificação do JPA 2.0 (pelo menos não na versão que estou utilizando).

Ficou ótimo!

Obrigado pela ajuda.

Forte abraço,

Neto

Em 25 de agosto de 2011 19:53, Ferrero Rocher <arthurseveral@yahoo.com.br> escreveu:
Olá Victor,

Não posso garantir com certeza que essa solução funcionará usando JPA. Pelo que eu sei, pelo menos na versão 1.0, JPA @Embeddable não aceita entidades como attributos embora algumas implementações aceitem. Em contra-partida, se vc usar a versão JPA 2.0, funcionará pq @Embeddable aceita entidade como attributos. Isso é possível pq JPA implementações fazem uso de Java reflection para acessar campos e propriedades como vc pode ver nessa minha questão

Atenciosamente
Arthur Ronald F D Garcia

--- Em qui, 25/8/11, Victor Tortorello Neto <vtneto@gmail.com> escreveu:

De: Victor Tortorello Neto <vtneto@gmail.com>

Assunto: Re: Dúvida no Hibernate
Para: "Ferrero Rocher" <arthurseveral@yahoo.com.br>
Data: Quinta-feira, 25 de Agosto de 2011, 17:01

Arthur, só para constar, também tentei desta maneira: http://stackoverflow.com/questions/6744451/hibernate-nm-extracthashcode-throws-nullpointerexception/7195594#7195594.

Nunca tinha visto usar o mappedBy composto desta maneira. Aqui esta mudança resultou no mesmo erro citado pelo autor da pergunta.

Abraço,

Neto

Em 25 de agosto de 2011 15:41, Victor Tortorello Neto <vtneto@gmail.com> escreveu:
Olá novamente, Arthur!

Em 25 de agosto de 2011 15:08, Ferrero Rocher <arthurseveral@yahoo.com.br> escreveu:

Olá Victor,

Pelo que eu vi no seu código, vc está usando Java Persisence, não Hibernate. Não sei se vc sabe mas Java Persistence é apenas uma especificação, enviada em anexo, não uma implementação. Qualquer fabricante pode implementar a especifição desde que atenda aos requisitos da mesma. NO seu caso, vc deve estar usando a implementação do HIbernate

Sim, estou utilizando JPA com a implementação do Hibernate. Achei melhor comparada à implementação do EclipseLink, que faz muitos SELECTs, enqto o Hibernate trabalha com INNER e OUTER JOINs -- por esta razão relacionei meu problema ao Hibernate em si, e não ao JPA.

Voltando ao seu problema

Problema ao executar o persist para salvar um objeto pai que tem uma lista de objetos filhos

Bem, Filho possui uma chave-composta. Até onde eu sei, nenhuma solução de persistência provê suporte para geração automática de chaves-compostas.

Na realidade, a geração automática é apenas para a chave primária de Pai (funciona corretamente). Na chave composta de Filho, o programa deve setar os dois valores, porém, um destes valores é uma foreign key.

Como há um relacionamento bi-direcional, vc deve especificar o atributo mappedBy que nada mais faz do que informar qual propriedade deve ser usada para configurar o link entre as entidades. Um link a nível de banco de dados fica a cargo de chave-estrangeira (foreign key). No seu caso, na entidade Pai, vc configura mappedBy="pai", ou seja, se o campo pai da entidade Filho estiver configurada, crie um link entre a entidade Pai e Filho.

Certo.

No entanto, a entidade Filho define duas colunas com o mesmo nome: uma utilizada pela chave-composta e outra utilizada pelo campo pai. Dessa forma, vc deve especificar qual das duas deve ser usada para refletir o estado da entidade. No seu caso, paiId da chave-composta foi usado.

```
    @ManyToOne
    @JoinColumn(name="PAI_ID", insertable=false, updatable=false)
    private Pai pai;

    @Embeddable
    public class FilhoId implements Serializable {

       @JoinColumn(name="PAI_ID")
        public Integer paiId;
    }
```

Correto. Tentei inverter esta situação e utilizar o 'insertable="false", updatable="false"' na coluna que está em FilhoId (@Embeddable), porém, o Hibernate não aceitou.

Vc havia dito: não configuro 'filho.getId().setPaiId()' porque teoricamente o Hibernate deveria preenche-lo...?

De modo algum, não é pq o Hibernate configurou pai.id que filho.id.paiId será refletido automaticamente. Como pode o Hibernate supor que filho.id.paiId se refere ao mesmo valor da propriedade pai.id.

Eu achei que ele deveria refletir automaticamente, até porque se trata da mesma coluna. Erro meu.

Bem, a sua solução não é possível usando field access strategy, ou seja, JPA acessa o estado da sua entidade usando seus campos ao invés de usar seus getters and setters.

Um outro detalhe, se vc usar um relacionamento bi-direcional, use um helper para configurar a bi-direcionalidade como a seguir

```
    addFilho(Filho filho) {
        filho.setPai(this);

        pai.getFilhoList().add(filho);
    }
```

Realmente, nunca utilizei desta maneira. Obrigado pela dica!

A sua solução segue em anexo, porém usando property access strategy.

Muito obrigado, Arthur. Irei analisar a sua implementação.
Para mim, é difícil encontrar alguém para debater sobre estes assuntos, portanto faço ou "adivinho" sempre tudo sozinho rsrs... Caso quiser um parceiro para estes assuntos meu MSN é neto.t2@hotmail.com.

Atenciosamente
Arthur Ronald F D Garcia

Obrigado novamente. Abraços,

Neto

--- Em qui, 25/8/11, Victor Tortorello Neto <vtneto@gmail.com> escreveu:

De: Victor Tortorello Neto <vtneto@gmail.com>
Assunto: Dúvida no Hibernate
Para: arthurseveral@yahoo.com.br
Data: Quinta-feira, 25 de Agosto de 2011, 12:00

Olá, Arthur.

Sou um desenvolvedor Java (atualmente em Catanduva, interior de SP) e vi algumas respostas suas sobre Hibernate no stackoverflow.

Estou com um problema ao executar o persist para salvar um objeto pai que tem uma lista de objetos filhos.

Caso você quiser e puder me ajudar, a estrutura é mais ou menos assim:
(Pseudo-código. Considere que existem os getters, setters, equals e hashCode).

```
@Entity
Pai {

@Id @GeneratedValue
Integer id;

@OneToMany(cascade=CascadeType.ALL, mappedBy="pai")
private List<Filho> filhos;

}

@Entity
Filhos {

@EmbeddedId
private FilhoId id;

@ManyToOne
@JoinColumn(name="pai_id", insertable="false", updatable="false")
private Pai pai;

@Column(...)
private String nome;

}

@Embeddable
FilhoId {

@Column(...)
Integer paiId;

@Column(...)
Integer num;

}
```

```
        Pai = new Pai();

        Filho filho1 = new Filho();
        filho1.setId(new FilhoId());
        // Não seto 'filho1.getId().setPaiId()' porque teoricamente o Hibernate deveria preenche-lo...?
        filho1.getId().setNum(1);
        filho1.setNome('Primeiro Filho');

        pai.setFilhos(new ArrayList<Filho>());
        pai.getFilhos().add(filho1);

        entityManager.persist(pai);
```

Problema: o Hibernate busca o id do Pai (que é um SequenceGenerator), mas não insere este valor nos filhos.

Caro Arthur, você conhece uma solução para este problema?

Obrigado.

Abraços,

Neto