ARTIGO: 11381
HasMany
Introdução

Nesse artigo veremos como usar uma lista HasMany no veloster.

Uso da anotação

A notação ***@HasMany*** é usada para atributos do tipo ***List***. Suas opções são:

String reference(): Nome da referência presente no outro lado da relação.

int batchSize(): Quantidade de itens carregados. O valor padrão é 20 se a lista for *lazy=true*.

boolean cascadeRemoveOnDelete(): Remove os ítems da relação quando a entidade dona da lista for removida (remove).

boolean cascadeMergeOrInsertOnSave(): Faz insert ou update nos ítems da lista quando a entidade dona da lista for salva (save).

boolean cascadeMergeOrInsertOnUpdate(): Faz insert ou update nos ítems da lista quando a entidade dona da lista for alterada (update).

boolean cascadeAll(): Habilita todas as opções de cascade.

boolean lazy(): habilita o lazy load na lista.

Caso de Uso:

Java Code

```
/**
 * Entity with reference
 *
 */
@Table
public class LazyListReference extends EntityImpl {

    @Column
    private String value;
    @Column
    @JoinColumn
    private LazyListEntity lazyListEntity;

    public LazyListReference() {
    }

    public LazyListReference(String value, LazyListEntity lazyListEntity) {
        this.value = value;
        this.lazyListEntity = lazyListEntity;
    }

    public String getValue() {
        return value;
    }

    public void setValue(String value) {
        this.value = value;
    }

    public LazyListEntity getLazyListEntity() {
        return lazyListEntity;
    }

    public void setLazyListEntity(LazyListEntity lazyListEntity) {
        this.lazyListEntity = lazyListEntity;
    }
}
```

Java Code

```
/**
 * Entity owner of the list
 *
 */
@Table
public class LazyListEntity extends EntityImpl {

    @Column
    private String value;
    @Column
    @HasMany(reference = "lazyListEntity")
    private List<LazyListReference> list;

    public LazyListEntity() {
    }

    public LazyListEntity(String value) {
        this.value = value;
    }

    public String getValue() {
        return value;
```

```
    }

    public void setValue(String value) {
        this.value = value;
    }

    public List<LazyListReference> getList() {
        return list;
    }

    public void setList(List<LazyListReference> list) {
        this.list = list;
    }
}
```

Java Code

```
/**
 *  Teste Case to LazyList
 *
 */
public class HasManyTestCase{

    @Test
    public void testLoadHasMany() {

        Veloster<LazyListEntity> velosterEntity = VelosterRepository.getVeloster(LazyListEntity.class);
        Veloster<LazyListReference> velosterReference = VelosterRepository.getVeloster(LazyListReference.class);

        LazyListEntity entity = new LazyListEntity("value abc");
        velosterEntity.save(entity);

        List<LazyListReference> list = new LinkedList<LazyListReference>();

        for (int i = 0; i < 10; i++) {
            list.add(new LazyListReference("value " + i, entity));
        }

        for (LazyListReference it : list) {
            velosterReference.save(it);
        }

        LazyListEntity e = velosterEntity.load(entity.getId());

        List<LazyListReference> items = e.getList();
        Assert.assertTrue(items.size() == 10);
        for (int i = 0; i < 10; i++) {
            Assert.assertTrue(items.get(i).getValue().equals("value " + i));
        }
    }
}
```

A lista usada para o lazy load é uma instância de *br.com.mobilemind.veloster.orm.model.ListLazy<T extends Entity>*, que estende *LinkedList<T>*. Essa lista pode ser usar sem a necessidade de uma entidade dona, como no exemplo a seguir:

Java Code

```
Veloster<Person> veloster = VelosterRepository.getVeloster(Person.class);

Criteria<Person> criteria = veloster.createCriteria();
criteria.add("name", new Eq("veloster"));

List<Person> items = new ListLazy(criteria, Person.class, 30, true);
```

Nesse exemplos, estamos criando uma lista lazy load, com páginas de tamanho 30 e que devem obedecer, como crit ério de carregamento, as condições especificadas na Criteria.