arthurgregorio / web-budget Public

Code Issues 15 Pull requests 26 Actions Projects 1 Wiki Security In

Commit

⚠ This commit does not belong to any branch on this repository, and may belong to a fork outside of the repository.

Add da anotação @DaTa do Lombok e refatoração e rm de @ desnecessarias

Browse files

deniserisia committed on Jun 14

1 parent 5ea2103 commit 5e5d4a3

Showing **1 changed file** with **10 additions** and **99 deletions**.

Whitespace Ignore whitespace Split Unified

109 src/main/java/br/com/webbudget/domain/entities/financial/Movement.java

```
20  20    import br.com.webbudget.domain.entities.registration.Contact;
21  21    import br.com.webbudget.domain.exceptions.BusinessLogicException;
22  22    import br.com.webbudget.infrastructure.utils.RandomCode;
23      - import lombok.EqualsAndHashCode;
24      - import lombok.Getter;
25      - import lombok.Setter;
26      - import lombok.ToString;
    23  + import lombok.*;
27  24    import org.hibernate.envers.AuditTable;
28  25    import org.hibernate.envers.Audited;
29  26
41  38    import static br.com.webbudget.infrastructure.utils.DefaultSchemes.FINANCIAL_AUDIT;
42  39    import static javax.persistence.CascadeType.REMOVE;
43  40
44      - /**
45      -  * This is a superclass for a {@link FixedMovement} and {@link PeriodMovement}. This class also
          represents the basic
46      -  * type of financial transaction in the application
47      -  *
48      -  * @author Arthur Gregorio
49      -  *
50      -  * @version 2.0.0
51      -  * @since 1.0.0, 04/03/2014
52      -  */
53  41    @Entity
54  42    @Audited
55  43    @Table(name = "movements", schema = FINANCIAL)
56  44    @Inheritance(strategy = InheritanceType.SINGLE_TABLE)
57  45    @AuditTable(value = "movements", schema = FINANCIAL_AUDIT)
    46  + @Data
58  47    @ToString(callSuper = true, exclude = {"apportionments", "deletedApportionments"})
59  48    @EqualsAndHashCode(callSuper = true, exclude = {"apportionments", "deletedApportionments"})
60  49    @NamedEntityGraph(name = "Movement.full", attributeNodes = @NamedAttributeNode(value =
          "apportionments"))
```

```
61  50    @DiscriminatorColumn(name = "discriminator_value", length = 15, discriminatorType =
          DiscriminatorType.STRING)
62  51    public class Movement extends PersistentEntity {
63  52
64      -     @Getter
65  53        @Column(name = "code", nullable = false, length = 6, unique = true)
66  54        private String code;
67      -     @Getter
68      -     @Setter
    55  +
69  56        @NotBlank(message = "{movement.identification}")
70  57        @Column(name = "identification", nullable = false, length = 90)
71  58        private String identification;
72      -     @Getter
73      -     @Setter
    59  +
74  60        @Column(name = "description", columnDefinition = "TEXT")
75  61        private String description;
76      -     @Getter
77      -     @Setter
    62  +
78  63        @NotNull(message = "{movement.value}")
79  64        @Column(name = "value", nullable = false)
80  65        private BigDecimal value;
81  66
82      -     @Getter
83      -     @Setter
84  67        @ManyToOne
85  68        @JoinColumn(name = "id_contact")
86  69        private Contact contact;
87  70
88      -     @OneToMany(mappedBy = "movement", cascade = REMOVE)
    71  +     @OneToMany(mappedBy = "movement", cascade = CascadeType.REMOVE)
89  72        private Set<Apportionment> apportionments;
90  73
91  74        @Transient
92  75        private Set<Apportionment> deletedApportionments;
93  76
94      -     /**
95      -      * Constructor...
96      -      */
97  77        public Movement() {
98  78            this.code = RandomCode.alphanumeric(6);
99  79            this.apportionments = new HashSet<>();
100 80            this.deletedApportionments = new HashSet<>();
101 81        }
102 82
103     -     /**
104     -      * Getter for the apportionments
105     -      *
106     -      * @return an unmodifiable {@link List} of the {@link #apportionments}
107     -      */
108     -     public Set<Apportionment> getApportionments() {
109     -         return Collections.unmodifiableSet(this.apportionments);
110     -     }
111     -
112     -     /**
113     -      * Getter for the deleted apportionments
114     -      *
```

```
115       -      * @return an unmodifiable {@link Set} of the {@link #deletedApportionments}
116       -      */
117       -     public Set<Apportionment> getDeletedApportionments() {
118       -         return Collections.unmodifiableSet(this.deletedApportionments);
119       -     }
120       -
121       -     /**
122       -      * Get only the name of the {@link Contact} of this movement
123       -      *
124       -      * @return the name of the linked {@link Contact}
125       -      */
126   83        public String getContactName() {
127   84            return this.contact != null ? this.contact.getName() : "";
128   85        }
129   86
130       -     /**
131       -      * To check if this movement is a expense
132       -      *
133       -      * @return true if is, false otherwise
134       -      */
135   87        public boolean isExpense() {
136       -         return this.apportionments
137       -                 .stream()
      88  +          return this.apportionments.stream()
138   89                  .findFirst()
139   90                  .map(Apportionment::isExpense)
140   91                  .orElse(false);
141   92        }
142   93
143       -     /**
144       -      * To check if this movement is a revenue
145       -      *
146       -      * @return true if is, false otherwise
147       -      */
148   94        public boolean isRevenue() {
149       -         return this.apportionments
150       -                 .stream()
      95  +          return this.apportionments.stream()
151   96                  .findFirst()
152   97                  .map(Apportionment::isRevenue)
153   98                  .orElse(false);
154   99        }
155  100
156       -     /**
157       -      * Method used to add new an new {@link Apportionment} to this movement
158       -      *
159       -      * @param apportionment the {@link Apportionment} to be added
160       -      */
161  101        public void add(Apportionment apportionment) {
162  102            this.apportionments.add(apportionment);
163  103        }
164  104
165       -     /**
166       -      * Same function of {@link #add(Apportionment)} but this one takes a {@link Set} as parameter
167       -      *
168       -      * @param apportionments to be added
169       -      */
170  105        public void addAll(Set<Apportionment> apportionments) {
171  106            this.apportionments.addAll(apportionments);
```

```
172  107      }
173  108
174       -  /**
175       -   * Remove an {@link Apportionment} from the list of apportionments
176       -   *
177       -   * @param apportionment the {@link Apportionment} to be removed
178       -   */
179  109      public void remove(Apportionment apportionment) {
180  110          if (apportionment.isSaved()) {
181  111              this.deletedApportionments.add(apportionment);
182  112          }
183  113          this.apportionments.remove(apportionment);
184  114      }
185  115
186       -  /**
187       -   * Calculate the amount not divided into the apportionments
188       -   *
189       -   * @return the total possible to be divided by an {@link Apportionment}
190       -   */
191  116      public BigDecimal calculateRemainingTotal() {
192       -
193       -        final BigDecimal remaining = this.value.subtract(this.calculateApportionmentsTotal());
     117  +        BigDecimal remaining = this.value.subtract(calculateApportionmentsTotal());
194  118
195  119          if (remaining.signum() <= 0) {
196  120              throw new BusinessLogicException("error.period-movement.no-value-to-divide");
199  123          return remaining;
200  124      }
201  125
202       -  /**
203       -   * Sum all the apportionments and give the total
204       -   *
205       -   * @return the total of all {@link Apportionment}
206       -   */
207  126      private BigDecimal calculateApportionmentsTotal() {
208       -        return this.apportionments
209       -                .stream()
     127  +        return this.apportionments.stream()
210  128                  .map(Apportionment::getValue)
211  129                  .reduce(BigDecimal.ZERO, BigDecimal::add);
212  130      }
213  131
214       -  /**
215       -   * Copy the {@link Apportionment} to a new list without reference to this movement
216       -   *
217       -   * This method is mainly used at the process of transforming a {@link FixedMovement} in a
         {@link PeriodMovement}
218       -   *
219       -   * @return a new {@link Set} of new {@link Apportionment} objects
220       -   */
221  132      public Set<Apportionment> copyApportionments() {
222  133          return this.apportionments.stream()
223  134                  .map(Apportionment::copyOf)
```

0 comments on commit `5e5d4a3`

Please sign in to comment.