OpenOdonto / sco Public

Code Issues Pull requests Actions Projects Wiki Security Ins

Commit

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

Refactor UniqueCpfValidator

Browse files

Felipe-Gs committed on Jun 14

1 parent a4efa6b commit ee603d9

Showing **1 changed file** with **68 additions** and **57 deletions**.

Whitespace Ignore whitespace Split Unified

125

openodonto-web/src/main/java/br/ueg/openodonto/validator/UniqueCpfValidator.java

```
...  ...  @@ -1,9 +1,5 @@
 1    1   package br.ueg.openodonto.validator;
 2    2
 3      - import java.lang.reflect.Field;
 4      - import java.sql.SQLException;
 5      - import java.util.HashMap;
 6      - import java.util.LinkedList;
 7    3   import java.util.List;
 8    4   import java.util.Map;
 9    5
21   17   import br.ueg.openodonto.dominio.constante.PessoaFisica;
22   18   import br.ueg.openodonto.util.WordFormatter;
23   19
24      - public class UniqueCpfValidator<T extends Entity & PessoaFisica<T>> extends
        AbstractValidator implements ObjectValidatorType{
25      -
26      -         public UniqueCpfValidator(T value) {
27      -                 this(null,value);
28      -         }
29      -
30      -         public UniqueCpfValidator(CpfValidator next,T value) {
31      -                 super(next, value);
32      -         }
33      -
```

```
-	@Override
-	@SuppressWarnings("unchecked")
-	public T getValue() {
-		return (T) super.getValue();
-	}
-
-	@Override
-	@SuppressWarnings("unchecked")
-	protected boolean validate() {
-		Class<T> type = (Class<T>) getValue().getClass();
-		String cpf = WordFormatter.clear(getValue().getCpf());
-		EntityManager<T> dao = (EntityManager<T>) ManageBeanGeral.getDao(type);
-		Map<String,Object> params = new HashMap<String, Object>(1);
-		params.put("cpf", cpf);
-		List<Field> keyFields = OrmResolver.getKeyFields(new LinkedList<Field>(), type, true);
-		String[] fields = new String[keyFields.size()];
-		for(int i = 0;i < keyFields.size();i++){
-			fields[i] = keyFields.get(i).getName();
-		}
-		IQuery query = CrudQuery.getSelectQuery(getValue().getClass(), params, fields);
-		try {
-			List<Map<String,Object>> result = dao.getSqlExecutor().executarUntypedQuery(query);
-			if(result.size() == 1){
-				OrmFormat format = new OrmFormat(getValue());
-				Map<String,Object> already = result.get(0);
-				Map<String,Object> local = format.format(fields);
-				boolean isSamePf = true;
-				OrmTranslator translator = new OrmTranslator(keyFields);
-				for(String field : fields){
-					isSamePf = isSamePf && already.get(translator.getColumn(field)).equals(local.get(field));
-				}
-				if(!isSamePf){
-					setErrorMsg("CPF já está sendo usado.");
-				}
-				return isSamePf;
-			}else if(result.size() > 1){
-				throw new IllegalStateException("Falha de integridade.Permitiso apenas uma pessoa por CPF.");
-			}
-		} catch (SQLException e) {
-			e.printStackTrace();
-		}
-		return true;
-	}
+ public class UniqueCpfValidator<T extends Entity & PessoaFisica<T>> extends AbstractValidator implements ObjectValidatorType {
```

```
+     private static final String ERROR_MSG = "CPF já está sendo usado.";
+
+     public UniqueCpfValidator(T value) {
+         super(value);
+     }
+
+     @Override
+     protected boolean validate() {
+         T value = getValue();
+         String cpf = WordFormatter.clear(value.getCpf());
+         UniqueCpfChecker<T> checker = new UniqueCpfChecker<>(value.getClass(), cpf);
+         return checker.check();
+     }
+
+     private static class UniqueCpfChecker<T extends Entity & PessoaFisica<T>> {
+         private final Class<T> type;
+         private final String cpf;
+
+         public UniqueCpfChecker(Class<T> type, String cpf) {
+             this.type = type;
+             this.cpf = cpf;
+         }
+
+         public boolean check() {
+             EntityManager<T> dao = ManageBeanGeral.getDao(type);
+             List<Map<String, Object>> result = getQueryResult(dao);
+             if (result.size() == 1) {
+                 return isSamePf(result.get(0));
+             } else if (result.size() > 1) {
+                 throw new IllegalStateException("Falha de integridade. Permitido apenas uma pessoa por CPF.");
+             }
+             return true;
+         }
+
+         private List<Map<String, Object>> getQueryResult(EntityManager<T> dao) {
+             IQuery query = createQuery(dao);
+             try {
+                 return dao.getSqlExecutor().executarUntypedQuery(query);
+             } catch (Exception e) {
+                 e.printStackTrace();
+                 return null;
+             }
+         }
+
+         private IQuery createQuery(EntityManager<T> dao) {
+             Map<String, Object> params = Map.of("cpf", cpf);
+             List<String> keyFieldNames = OrmResolver.getKeyFieldNames(type, true);
+             String[] fields = keyFieldNames.toArray(new String[0]);
+             return CrudQuery.getSelectQuery(type, params, fields);
+         }
+
```

```
+        private boolean isSamePf(Map<String, Object> already) {
+            T value = getValue();
+            OrmFormat format = new OrmFormat(value);
+            List<String> keyFieldNames = OrmResolver.getKeyFieldNames(type, true);
+            Map<String, Object> local = format.format(keyFieldNames.toArray(new String[0]));
+            OrmTranslator translator = new OrmTranslator(type, keyFieldNames);
+            for (String fieldName : keyFieldNames) {
+                Object alreadyValue = already.get(translator.getColumn(fieldName));
+                Object localValue = local.get(fieldName);
+                if (!alreadyValue.equals(localValue)) {
+                    setErrorMsg(ERROR_MSG);
+                    return false;
+                }
+            }
+            return true;
+        }
+    }
 }
```

0 comments on commit `ee603d9`

Please sign in to comment.