API

Businesstest-Skript

Jedes Businesstest-Skript ist automatisch von der Klasse org.nuclos.server.businesstest.execution.BusinessTestScript abgeleitet und hat dadurch unter anderem folgende Methoden zur Verfügung:

void attempt(Callable c)

Führt das übergebene Callable (z.B. eine Groovy Closure) aus und fängt auftretende Exceptions ab. Dabei wird ggf. der "Fehler"- oder "Warnung"-Zähler erhöht, abhängig vom Typ der Exception.
Die Ausführung des Scripts an sich wird dabei jedoch nicht abgebrochen, wie das normalerweise bei Exceptions der Fall wäre.

void fail(String message)

Lässt den Test fehlschlagen und trägt die angegebene message als Ergebnis ein.
Sollte nur in Ausnahmefällen verwendet werden, da meistens Assertions die bessere Alternative sind (welche jedoch einen Fehler melden, wenn sie fehlschlagen, statt nur einer Warnung).

<T> T withUser(String user, Callable<T> c)

Führt das übergebene Callable als der angegebene User aus, wobei entsprechend alle Rechte und Einschränkungen des Users gelten.
Die Methode liefert den Rückgabewert des Callables zurück.

Beispiel:

```
withUser('test') {
    Order order = new Order(orderNumber: 123).save()
    assert order.createdBy == 'test'
}
```

Entitätsklassen

Für jedes vorhandene BO wird eine entsprechende Entitätsklasse generiert, über die auf die Attribute des BOs zugegriffen werden kann, und die einige Hilfsmethoden bereitstellt.

Getter und Setter für Attribute

Analog zur Serverregel-API: Für jedes Attribut eines BOs gibt es einen Getter und Setter. Für Referenz-Attribute zusätzlich einen Getter und Setter für die ID.
In Groovy können diese Methoden wie einfache Properties benutzt werden.

Beispiel:

```
Order order = new Order()
order.orderNumber = 10020149
println order.orderNumber
```

get(Long id) - Datensatz über die ID holen

Statische Methode der Entitätsklasse - entspricht QueryProvider.getById() der Serverregel-API.

Beispiel:

```
Order order = Order.get(12345)
```

save() - Datensatz speichern

Speichert den aktuellen Datensatz - entspricht BusinessObjectProvider.insert() oder BusinessObjectProvider.update(), je nachdem ob der Datensatz neu ist.
Eine manuelle Unterscheidung ist hier nicht mehr nötig.

Beispiel:

```
Order order = new Order()
order.orderNumber = 10020149
order.save()
order.orderNumber++
order.save()
```

delete() - Datensatz löschen

Löscht den aktuellen Datensatz - entspricht BusinessObjectProvider.delete().
Der Datensatz muss hierfür bereits gespeichert gewesen sein.

Beispiel:

```
Order.get(12345).delete()
```

first() - Ersten Datensatz holen

Holt den ersten Datensatz dieses BOs, den die Datenbank liefert (ohne bestimmte Reihenfolge).
Wird vor allem in den automatisch generierten Tests verwendet, wenn keine sinnvolleren Suchkriterien zur Verfügung stehen.

Beispiel:

```
Order order = Order.first()
```

list(int limit) - Mehrere Datensätze holen

Analog zu first(), jedoch werden gleich mehrere Datensätze geholt. *limit* ist optional - falls nicht angegeben, ist der Default-Wert 100.

Beispiel:

```
List<Order> orders = Order.list(10)
```

query(String where) - Komplexe SQL-ähnliche Abfragen

Hierüber können komplexe Suchen durchgeführt werden. Der Parameter muss ein Query-String sein, wie ihn auch der REST-Service akzeptiert (siehe dazu Dokumentation des "where" Parameters unter: 4. Businessobjekte (BO) lesen).

Beispiel:

```
List<Order> orders = Order.query('''
    example_rest_Order_customer = 40000294
AND example_rest_Order_orderDate >= '2014-06-05'
AND example_rest_Order.id IN (
        SELECT example_rest_OrderPosition_order
        FROM example_rest_OrderPosition
        WHERE example_rest_OrderPosition_price > 800
    )
''')
```

Dependents (Subform-Datensätze)

Wenn ein BO von anderen BOs referenziert wird, wird für jede dieser Referenzen eine Methode der Form get<ReferenzBO>By<Attributname>() auf dem referenzierten BO generiert. Diese Methode liefert eine Collection der Dependents, welcher auch neue Datensätze hinzugefügt werden können.

Beispiel:

Hinzufügen und Löschen eines Subform-Datensatzes

```
Customer bo = new Customer()
bo.customerNumber = 123
bo.name = 'Customer 123'
bo.active = true

// Add dependent
bo.getCustomerAddressByCustomer().add(new CustomerAddress(
      street: 'Street 1',
      city: 'Munich',
      zipCode: '12345'
))

bo.save()

assert bo.getCustomerAddressByCustomer().size() == 1

// Delete first dependent
bo.getCustomerAddressByCustomer().remove(
      bo.getCustomerAddressByCustomer().first()
)

bo.save()

assert bo.getCustomerAddressByCustomer().empty
```

BOs mit Statusmodell

Falls das BO ein Statusmodell hat:

changeStat(int state) - Statuswechsel

Wechselt in den angegebenen Zielstatus - entspricht StatemodelProvider.changeState().

Beispiel:

```
Order.get(12345).changeState(80)
```

BOs mit Benutzerregeln

Falls dem BOs Benutzerregeln (CustomRules) zugewiesen sind, steht für jede Regel eine Methode der Form execute<Regel-Name>() zur Verfügung.

Beispiel (Order hat die CustomRule "LockOrder"):

```
Order.get(12345).executeLockOrder()
```

BOs mit "Aktionen"

Falls dem BOs "Aktionen" (aka NuclosProcess) zugewiesen sind, steht für jede Aktion eine Methode der Form changeProcess<Aktionsname>() zur Verfügung. Zusätzlich gibt es eine Method unsetProcess(), um die Aktion zu entfernen.

Beispiel (Order hat die Aktion "Priority order"):

```
Order.get(12345).changeProcessPriorityorder()
```