Step-by-step example

Created by Unknown User (ludbac), last modified by Joakim Engerstam1 on Jul 03, 2020

This example demonstrates step-by-step how to create a test case for entering and verifying a deal.

1. Create a new class named **TcEnterDealExample**, for example under package **com.nasdaq.nv.testcases.functional.clearing.template**, and extend **CoreTestCase**. Add two methods, **tcSetup** and **testEnterDeal** . The **tcSetup** method should override the **tcSetup** method from **CoreTestCase** and will be used to initialize the data needed to create and enter the trade, make sure to call super to setup common configuration. The **testEnterDeal** method will handle entering and verifying the deal.

   TcEnterDealExample

```
public class TcEnterDealExample extends CoreTestCase {

    @Override
    public void tcSetup() {
        super.tcSetup();
    }

    public void testEnterDeal() {

    }

}
```

2. To enter the deal an existing buy- and sell-account as well as a tradeable series need to be added. These will be dependent on the backend node the test is run against and should therefore preferably be configurable. To achieve this the omexus properties-file located at */batches/testrunner/backends/OMexusConfiguration.properties* can be used.

   a. In OMexusConfiguration.properties, add configuration according to the below. The account parameters need to be on the format <Exchange> <Participant> <AccountName> (separated by white space that is). For this case only the instrument type will be specified, not the exact series. This will allow the test case to pick a random series according to the specified instrument type. Valid accounts and instrument types can be found via the RDM Core Desktop application, the below ones are only example values.

```
### Parameters for TcEnterDealExample
com.nasdaq.nv.testcases.functional.clearing.template.TcEnterDealExample.buy_account=SA GCM2 TA_C_GROSS
com.nasdaq.nv.testcases.functional.clearing.template.TcEnterDealExample.sell_account=SA NCM2 TA_C_GROSS
com.nasdaq.nv.testcases.functional.clearing.template.TcEnterDealExample.instrument_type=FI_CBL
```

   b. In TcEnterDealExample, add instance variables for the buy account, sell account and series. For the series the \@TCInstrumentTypeProp annotation is used to enable injection of the actual series-object. Note that the value-field of the annotation need to have the same name as the field in the omexus properties file. In this case it's also specified that a random series should be picked and that only tradeable series should be eligible. The account-values are fetched from the properties file using the iOmexusConfiguration-field available from CoreTestCase, and are then converted to Account-objects using the utility-class AccountUtil.

   TcEnterDealExample

```
import nff.nvs.message.gen.structure.Account;

import om.omexus.toolbox.cdb.CDBSeriesInfo;
import om.omexus.functionarea.annotations.setup.TCInstrumentTypeProp;
import om.omexus.functionarea.account.AccountUtil;

public class TcEnterDealExample extends CoreTestCase {

    @TCInstrumentTypeProp(random=true, tradeable=true, value="instrument_type")
    private CDBSeriesInfo series;

    private Account buyAccount;
    private Account sellAccount;

    @Override
    public void tcSetup() throws OMexusException {
        super.tcSetup();
        buyAccount  = AccountUtil.toAccount(iOmexusConfiguration.getTestCaseParam("buy_account"));
        sellAccount = AccountUtil.toAccount(iOmexusConfiguration.getTestCaseParam("sell_account"));
    }
```

3. Add a TradeToolDelux and a Collector as instance-variables. These are utility classes that will make it more convenient to enter and verify the deal. TradeToolDelux provides methods for sending the deal transaction (CB0) and Collector is used to subscribe and collect the broadcasts that will be used in the verification (CB10001). Note that both classes have a dependency to the current NVS-session which is fetched using the SessionHandler-class. TradeToolDelux also needs the business date and the system date of the backend node, which both are fetched using the SystemInfoUtil-class.

TcEnterDealExample

```
import nff.nvs.message.gen.structure.Account;

import com.nasdaq.nv.framework.Collector;

import om.omexus.toolbox.cdb.CDBSeriesInfo;
import om.omexus.functionarea.annotations.setup.TCInstrumentTypeProp;
import om.omexus.functionarea.account.AccountUtil;
import om.omexus.functionarea.productData.tradedata.TradeToolDelux;
import om.omexus.toolbox.util.SystemInfoUtil;
import om.omexus.toolbox.util.SessionHandler;

public class TcEnterDealExample extends CoreTestCase {

    @TCInstrumentTypeProp(random=true, tradeable=true, value="instrument_type")
    private CDBSeriesInfo series;

    private Account buyAccount;
    private Account sellAccount;
    private TradeToolDelux tradeTool;
    private Collector collector;

    @Override
    public void tcSetup() throws OMexusException {
        super.tcSetup();
        buyAccount  = AccountUtil.toAccount(iOmexusConfiguration.getTestCaseParam("buy_account"));
        sellAccount = AccountUtil.toAccount(iOmexusConfiguration.getTestCaseParam("sell_account"));
        tradeTool = new TradeToolDelux(SessionHandler.getNvsSession(), SystemInfoUtil.getBusinessDate(), Syste
        collector = new Collector(SessionHandler.getNvsSession());
    }
```

4. Add the code for entering the trade to the testEnterDeal-method. First start subscribing to CB1001-broadcasts by calling *CB10001Collector()*, then use tradeTool to create a TradeData object. This is a data container that is provided to the enterTrade-method of tradeTool together with the series, a quantity and a price to actually enter the deal (note that trade and deal are used interchangeably here).

TcEnterDealExample

```
public void testEnterDeal() throws OMexusException {
        // Before entering the trade start subscribing to CB10001 broadcasts to be used for verification.
        CB10001Collector cb10001Collector = collector.CB10001Collector();

        // Enter the trade using the trade tool utility-class and our previously initialized accounts and seri
        int qty = 10;
        int price = 150;
        TradeData tradeData = tradeTool.create().buyAccount(buyAccount).sellAccount(sellAccount).freeText("EN
        tradeTool.enterTrade(series, qty, price, tradeData);

    }
```

5. Collect the broadcast and verify that it contains the same trade details as the entered tradeData. To do this the CB1001Verifier-class is used, which will verify trade details such as quantity, direction, price etc. and on any difference throw an *OMexusException* with a detailed error message causing the test case to fail. If no exception is thrown the test will pass.

TcEnterDealExample

```
...
// Verify the broadcast against the entered trade data.
CB10001Collection collection = cb10001Collector.collect();
CB10001Verifier cb10001Verifier = new CB10001Verifier(tradeData);
cb10001Verifier.verifyTrade(collection.get(BoughtOrSold.BOUGHT), collection.get(BoughtOrSold.SOLD));
...
```

6. To run the test case in batch runner, create an xml-config for it pointing to the test-case class and save it in the /batches folder. Below is an example using CDC_INT_1 as the clearing user, which is needed to enter the deal. As this is done the test case can be tested locally by running BatchRunner and loading the xml.

Batch Config

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--To use the xsl-stylesheet it must be in the same directory as the XML file.-->
<?xml-stylesheet type="text/xsl"?>
<TestBatch name="EnterDealExample" type="Functionality" version="6.0.0000" customer="ptp" description="">
    <TestCase id="com.nasdaq.nv.testcases.functional.clearing.template.TcEnterDealExample" type="Functionality
        <Thread threadNumber="0">
            <User portNumber="8024" userDescription="CDC_INT_1" maxUsersAtPrimaryNG="30" secondaryHostname=""
        </Thread>
    </TestCase>
</TestBatch>
```

The full code for the test case:

TcEnterDealExample

```
package com.nasdaq.nv.testcases.functional.clearing.template;

import com.nasdaq.nv.framework.Collector;
import com.nasdaq.nv.message.CB10001Collector;
import com.nasdaq.nv.message.collection.CB10001Collection;
import com.nasdaq.nv.verification.trade.CB10001Verifier;

import nff.nvs.message.gen.structure.Account;

import om.omexus.functionarea.account.AccountUtil;
import om.omexus.functionarea.annotations.setup.TCInstrumentTypeProp;
import om.omexus.functionarea.productData.tradedata.TradeData;
import om.omexus.functionarea.productData.tradedata.TradeToolDelux;
import om.omexus.message.cl.constants.BoughtOrSold;
import om.omexus.omex.omnet.OMexusException;
import om.omexus.testcases.CoreTestCase;
import om.omexus.toolbox.cdb.CDBSeriesInfo;
import om.omexus.toolbox.util.SessionHandler;
import om.omexus.toolbox.util.SystemInfoUtil;

public class TcEnterDealExample extends CoreTestCase {

    @TCInstrumentTypeProp(random=true, tradeable=true, value="instrument_type")
    private CDBSeriesInfo series;
    private Account buyAccount;
    private Account sellAccount;
    private TradeToolDelux tradeTool;
    private Collector collector;

    @Override
    public void tcSetup() throws OMexusException {
        super.tcSetup();
        buyAccount  = AccountUtil.toAccount(iOmexusConfiguration.getTestCaseParam("buy_account"));
        sellAccount = AccountUtil.toAccount(iOmexusConfiguration.getTestCaseParam("sell_account"));
        tradeTool = new TradeToolDelux(SessionHandler.getNvsSession(), SystemInfoUtil.getBusinessDate(), SystemInfo
        collector = new Collector(SessionHandler.getNvsSession());
    }

    public void testEnterDeal() throws OMexusException {
        // Before entering the trade start subscribing to CB10001 broadcasts to be used for verification.
        CB10001Collector cb10001Collector = collector.CB10001Collector();

        // Enter the trade using the trade tool utility-class and our previously initialized accounts and series.
        int qty = 10;
        int price = 150;
        TradeData tradeData = tradeTool.create().buyAccount(buyAccount).sellAccount(sellAccount).freeText("ENTER_DE
        tradeTool.enterTrade(series, qty, price, tradeData);

        // Verify the broadcast against the entered trade data.
        CB10001Collection collection = cb10001Collector.collect();
        CB10001Verifier cb10001Verifier = new CB10001Verifier(tradeData);
        cb10001Verifier.verifyTrade(collection.get(BoughtOrSold.BOUGHT), collection.get(BoughtOrSold.SOLD));
    }

}
```

Using Framework Components

With Frameworks Components the instance creation of commonly used framework classes is delegated to the framework resulting in less boiler plate code and cleaner code.

The same example using Framework Component injection and JUnit:

TcEnterDealExample

```
public class TcEnterDealExample extends CoreTestCase {

    @TCInstrumentTypeProp(random=true, tradeable=true, value="instrument_type")
    private CDBSeriesInfo series;
    private Account buyAccount;
    private Account sellAccount;

    @Framework
    private TradeDataFactory tradeDataFactory;
    @Framework
    private TradeFunctionsNew tradeFunctions;
    @Framework
    private Collector collector;
    @Framework
    private Verifiers verifiers;

    @Override
    public void tcSetup() throws OMexusException {
        super.tcSetup();
        buyAccount  = AccountUtil.toAccount(iOmexusConfiguration.getTestCaseParam("buy_account"));
        sellAccount = AccountUtil.toAccount(iOmexusConfiguration.getTestCaseParam("sell_account"));
    }

    @Test
    public void testEnterDeal() throws OMexusException {

        // Create the test data using Trade Data factory component
        int qty = 10;int price = 150;
        TradeData tradeData = tradeDataFactory.create(series, qty, price)
            .buyAccount(buyAccount)
            .sellAccount(sellAccount)
            .freeText("ENTER_DEAL_EXAMPLE");

        // Enter the trade using Trade functions component
        CB10001Collection collection = tradeFunctions.enterTrade(tradeData);

        // Verify collected messages using Verifier component
        verifiers.cb10001Verifier(tradeData)
            .verifyTrade(collection.get(BoughtOrSold.BOUGHT), collection.get(BoughtOrSold.SOLD));
    }

}
```

No labels