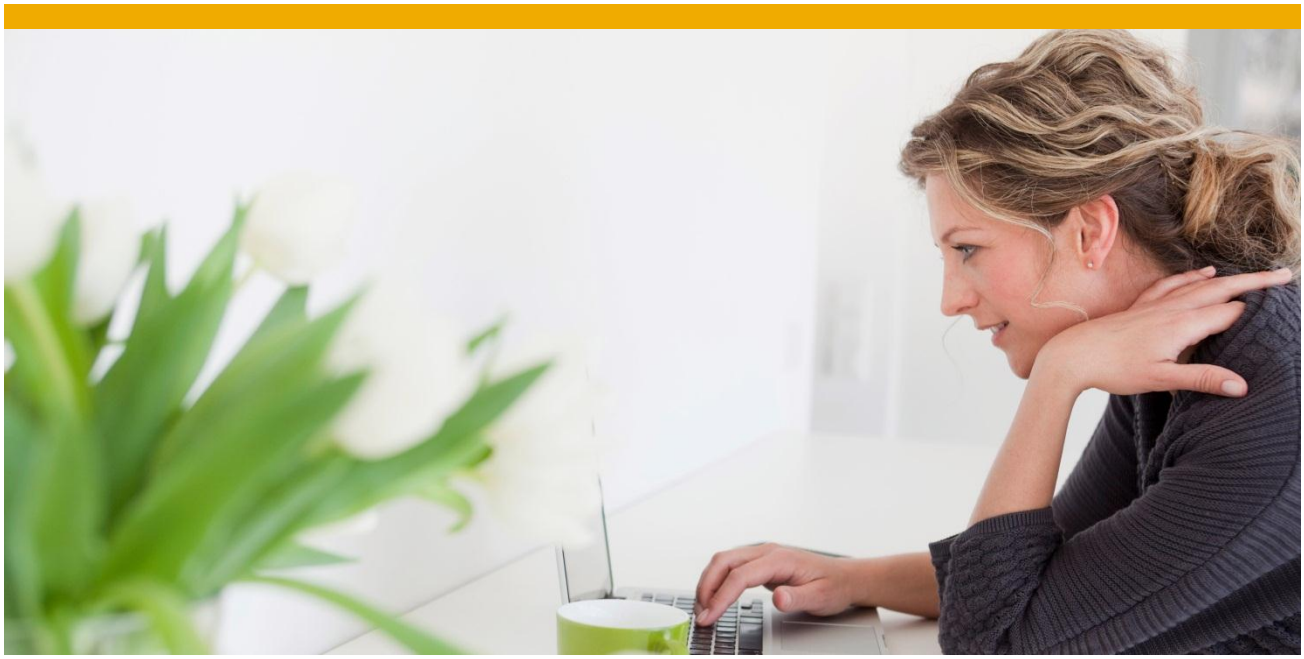


LAB3b: Transaction Overlay Processing

Change the Custom Transaction Flow Overlay

TABLE OF CONTENTS

AGENDA

1. Change the Custom Transaction Flow Overlay
2. Install and Test the Transaction Flow Overlay

BEFORE YOU START…

Please start the Mobiliser 5.1 Lab Virtual machine.

The Login with user is "mobiliser" and password "sybase".

MODIFY THE CUSTOM TRANSACTON OVERLAY

We are going to extend the existing custom transaction flow overlay implementation by changing the `afterInitTransaction();`. We are going to access the database retrieving Department information based on the authorization request.

(For more information please see Mobiliser_Framework_5.1_Customization_Guide.pdf section **3.5 Transaction Flow Overlay**)

1. Navigate to the *com.sybase365.mobiliser.custom.project.businesslogic.impl* bundle
2. Open the custom transaction flow overlay implementation (**CustomTransactionFlowOverlay.java**) to add the additional logic.
3. In the new logic we would try to get Department information stored in the database, as the Authorization request moves through different stages of the transaction.
4. Database communication to get Department information would require to getting access to Custom DaoFactory as well as TransactionTemplate. Spring configuration for those injections will come later.

```
(...)

import
com.sybase365.mobiliser.money.businesslogic.transaction.demarcation.impl.MobiliserTransaction
Template;
import com.sybase365.mobiliser.custom.project.persistence.dao.factory.api.DaoFactory;

(...)
```

5. Add DaoFactory and MobiliserTransactionTemplateFactory as member variables

```
(...)

public class CustomTransactionFlowOverlay implements
       ITransactionFlowOverlay<IMoneyRequest, ITransactionResponse> {

DaoFactory daoFactory;
MobiliserTransactionTemplate transactionTemplate;

(...)
```

6. Add Getters and Setters for injected properties (member variables).

```
(...)

public void setDaoFactory(DaoFactory daoFactory) {
        this.daoFactory = daoFactory;
}

public void setTransactionTemplate(MobiliserTransactionTemplate transactionTemplate) {
        this.transactionTemplate = transactionTemplate;
}

public DaoFactory getDaoFactory() {
        return this.daoFactory;
}

public MobiliserTransactionTemplate getTransactionTemplate() {
        return this.transactionTemplate;
}

(...)
```

7. Format the code: “Ctrl+A” then “Ctrl+Shift+F” to format the contents of the file.
8. Organize the imports: “Ctrl+Shift+O” then verify the imports.
9. Save the CustomTransactionFlowOverlay.java

10. Modify the existing Spring configuration to wire DaoFactory and TransactionTemplate objects.

bundle-context.xml

```
(...)

<bean id="customTxnFlow"
class="com.sybase365.mobiliser.custom.project.businesslogic.impl.CustomTransactionFlowOverlay
">
        <property name="daoFactory" ref="daoFactory" />
        <property name="transactionTemplate" ref="transactionTemplate" />
</bean>

(...)
```

11. Please indent the XML contents by pressing “Ctrl+A” and then “Ctrl+I”.
12. Save the file **bundle-context.xml.**
13. Replace the logic in `afterInitTransaction()` method with the following logic.

CustomTransactionFlowOverlay.java

Packages to import:

org.springframework.transaction.TransactionStatus
org.springframework.transaction.support.TransactionCallback
com.sybase365.mobiliser.custom.project.persistence.model.Department

```
(...)

@Override
public void afterInitTransaction(IMoneyRequest request,
               ITransactionResponse response,
               AtomicReference<CallerInformation> callerRef,
               AtomicReference<Long> authIdRef,
               boolean persist) {

   LOG.info("#afterInitTransaction");
   if (request instanceof IAuthorisationRequest) {
   final IAuthorisationRequest authRequest = ((IAuthorisationRequest) request);
   LOG.info("useCase: " + authRequest.getUsecase());
   final StringBuffer sbDepartmentId = new StringBuffer();

   this.transactionTemplate.execute(new TransactionCallback<Boolean>() {
      @Override
      public Boolean doInTransaction(TransactionStatus status) {
         try {
            final Department department = getDaoFactory().getDepartmentDao()
               .findDepartmentByName(authRequest.getText());
            sbDepartmentId.append(department.getId());
         } catch (Exception e) {
            // Nothing to do...
         }
         return null;
      }
   });

   if (sbDepartmentId.length() != 0) {
      response.getUnstructuredData().add(new IKeyValue() {
            @Override
            public String getValue() {
               return sbDepartmentId.toString()
                  + " is the associated Department ID";
            }
            @Override
            public String getKey() {
               return "DepartmentID";
            }
         });
      }
   } else {
      LOG.info("unknown request type: " + request.getClass().getName());
   }
}
```

14. Format the code: “Ctrl+A” then “Ctrl+Shift+F” to format the contents of the file.
15. Organize the imports: “Ctrl+Shift+O” then verify the imports.
16. Save the CustomTransactionFlowOverlay.java

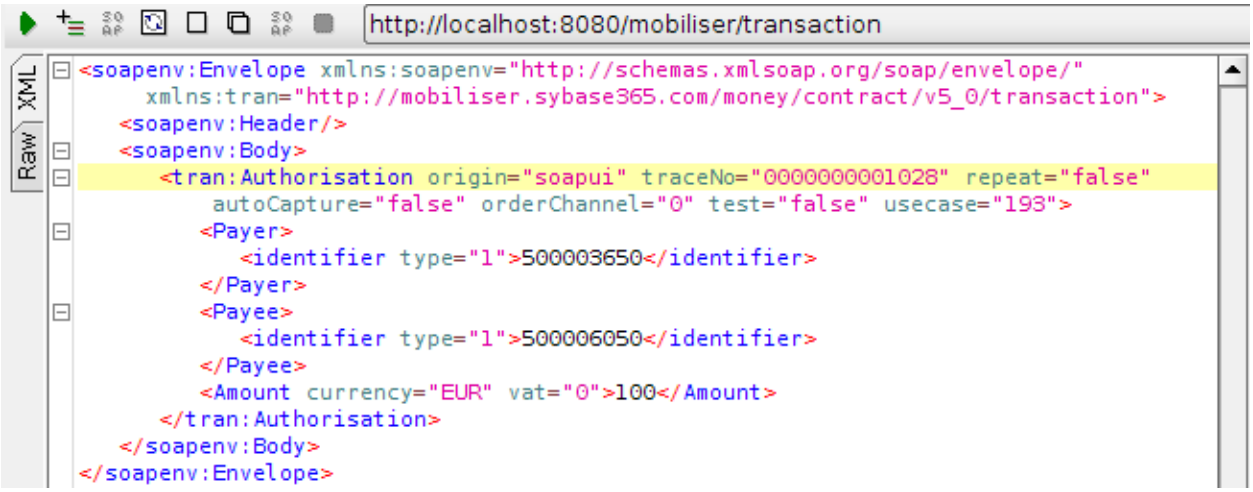
BUILD THE BUNDLE

BusinessLogic.Impl is the only bundle that gets modified – build this bundle and use hot deployment technique to deploy this bundle. For instruction please refer to Lab 3a – Hot deployment section.

TEST

We will be using SoapUI to do the validation of our code.

1. Make sure that your money container is running.
2. Verify that the Transaction.wsdl is loading correctly by enter the following URL into your browser: http://localhost:8080/mobiliser/transaction/Transaction.wsdl
3. Launch SoapUI and create a new project.
4. Initial WSDL/WADL: http://localhost:8080/mobiliser/transaction/Transaction.wsdl
5. We will be creating an Authorization request.
6. Modify the tran:Authorization tag: delete the following attributes callback, conversationId, and sessionId. Set the other attributes to these values: origin="soapui" traceNo="000000000100" (this can be any number), orderChannel="0" usecase="193". The other attributes can stay at the default values.
7. You can remove the following tags and sub tags: AuditData, UnstructuredData, OrderID, Timestamp, and attribute.
8. Remove any remaining tags so the code in the SoapUI editor looks similar to the following:



9. You will need two customer id numbers. You should have at least one, the one you created earlier. Please refer to Lab1 on how to create customers.
10. The Payer and Payee identifier type = "1". The Amount tag has the following values. currency="EUR" vat="0" the value is the lowest denomination for that currency (cent) to enter 1EUR you would enter 100.
11. Open the SQL Developer. The department information is in table MOBR5.CUS_DEPARTMENT. The customer information is in MOBR5.MOB_CUSTOMERS.
12. Remember to add the username=mobiliser, password=secret, and Authentication Type=Preemptive.

13. You are now ready to submit the request. You should receive three (3) UnstructredData tags in the soapenv

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
          xmlns:tran="http://mobiliser.sybase365.com/money/contract/v5_0/transaction">
  <soapenv:Header/>
  <soapenv:Body>
    <tran:Authorisation origin="soapui" traceNo="0000000001028" repeat="false"
          autoCapture="false" orderChannel="0" test="false" usecase="193">
      <Payer>
        <identifier type="1">500003650</identifier>
      </Payer>
      <Payee>
        <identifier type="1">500006050</identifier>
      </Payee>
      <Amount currency="EUR" vat="0">100</Amount>
    </tran:Authorisation>
  </soapenv:Body>
</soapenv:Envelope>
```

www.sap.com