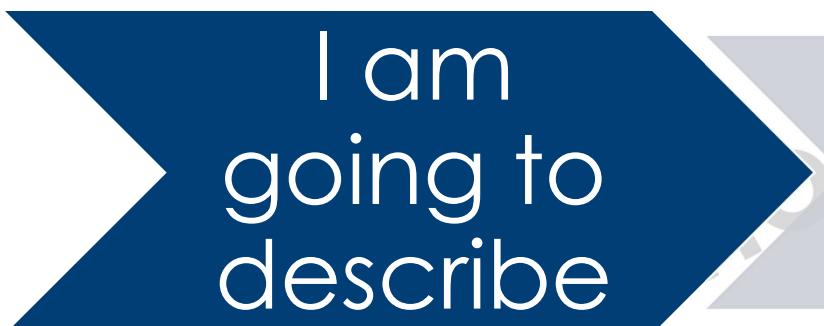
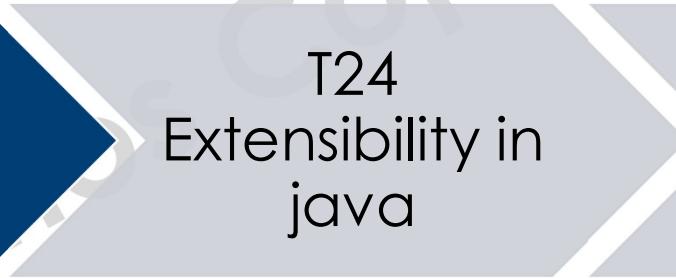


| T24 Extensibility in Java

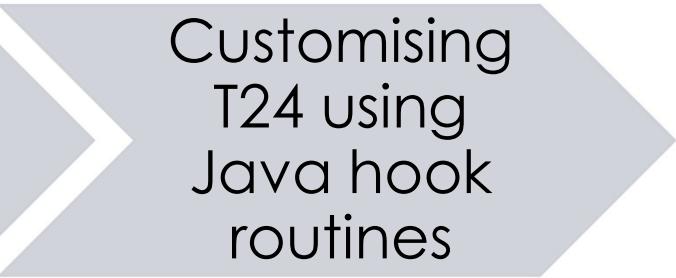
I Lesson Overview



I am
going to
describe



T24
Extensibility in
java



Customising
T24 using
Java hook
routines

| Your Course » Objective and Learning Outcomes

This course will introduce you to:

- T24 Extensibility in Java

In particular you will be able to:

- Understand T24 extensibility and customisation using Java
- Setup and Use Design Studio to Create and execute VERSION, ENQUIRY, SERVICE hooks
- Create and execute hooks attached to a local application
- Debug the Java code in Design Studio

| Your Course » Timetable



Day 1

- Introduction to T24 Extensibility in Java
- Setup Design Studio
- VERSION hooks



Day 2

- DEBUG the Java code in Design Studio
- ENQUIRY hooks
- SERVICE hooks
- Local application using EB.TABLE.DEFINITION



| Lesson 1. Introduction

T24 Extensibility in Java



| Agenda - T24 Extensibility in Java

- Why do we need T24 Extensibility in Java ?
- APIs and Hooks
- T-Types
- DataAccess
- Complex classes
- Setup Design Studio

| Why do we need T24 Extensibility in Java ?

- Scalability and ownership
 - There are more java developers than jBC developers
 - Banks can use their own developers
- Simplify the T24 APIs
 - The developer should not need to know about STORE.END.ERROR, R.NEW or the different ways of storing an amount field in T24
- Governance
 - We can use the framework to protect t24. for e.g. prevent a select on a large table that can lead to performance issue for all users

Prerequisites

- JD product must be installed

The screenshot shows a software interface titled "SPF SYSTEM" with a navigation bar at the top. The main area displays a list of configuration parameters. One parameter, "Extensible Customisation", is highlighted with a red box and has a dropdown menu open. The dropdown menu contains three options: "No" (selected), "Yes", and "[None]". There are also three radio buttons below the dropdown: "[None]" (selected), "No", and "Yes".

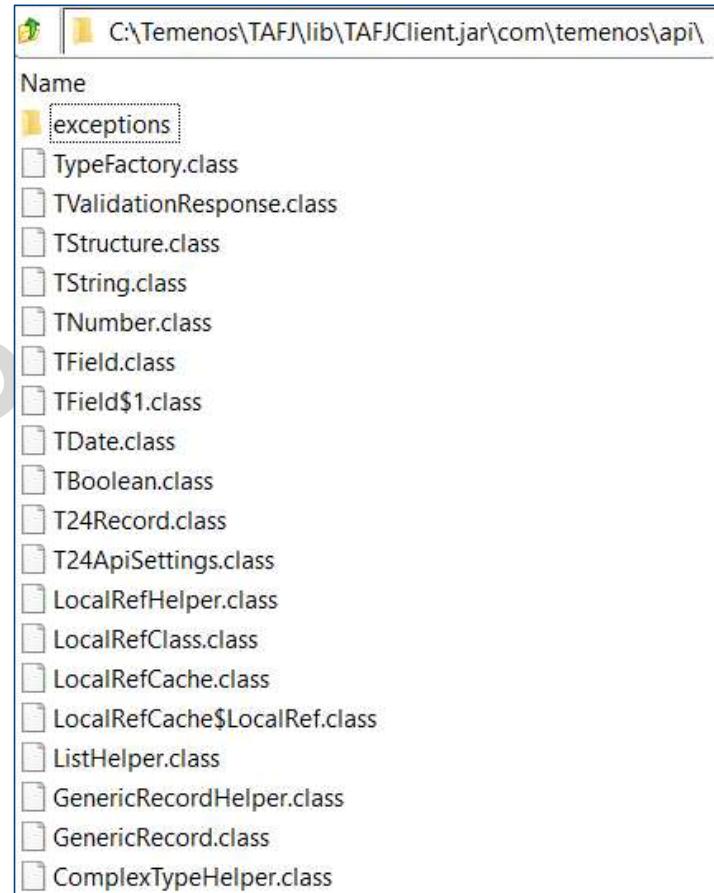
Parameter	Value
Restructure Tables.1	
Last Cache Reset Period.1	
Enquiry Sample Size	
Extensible Customisation	No
Ext Security Frame	
Deployment Mode	
Infinity	
Reserved 4	

APIs and Hooks

- APIs are code Temenos has written that L3 developers can call
- Hooks are code L3 developers have written that T24 can call
- A developer can attach own logic to T24 to be called from core application exits using HOOKS
- Each hook has been provided with methods to carry out the task
- For e.g the Java developer will use the validateField method defined in com.temenos.t24.api.hook.system.RecordLifeCycle class for field validation exits

'T' types in T24

- Classes introduced by Temenos for Java developers in TAFJClient.jar
- TStructure
- TField
- TValidationResponse
- Many more like TString, TNumber, TBoolean, TDate ...



'T' types in T24 - TStructure

- The “**TStructure**” is a generic type containing a record object
- TStructure must be ‘cast’ to the correct record type
- Maps a jBC dynamic array to java object
- To access a record, the java developer must construct an instance of a record or complex type from the received parameter

```
public TValidationResponse validateRecord(String application, String recordId, TStructure record,
                                         TStructure lastLiveRecord) {
    FundsTransferRecord fr = new FundsTransferRecord(record);
    TField f1 = fr.getDebitCurrency();
    TField f2 = fr.getCreditCurrency();

    if (!f1.getValue().equals(f2.getValue()))
    {
        f2.setError("CREDIT CURRENCY IS NOT EQUAL TO DEBIT CURRENCY");
    }
    return fr.getValidationResponse();
}
```

'T' types in T24 - TField

TField offers getters and setters for Enrichment, Error, field values.

Every field in a record is an internal type **TField** (not String by default)

```
FTVersionDemo.java ✘
package com.newbank;

import java.util.List;

public class FTVersionDemo extends RecordLifecycle {

    @Override
    public TValidationResponse validateRecord(String application, String recordId, TStructure record,
                                              TStructure lastLiveRecord) {
        FundsTransferRecord fr = new FundsTransferRecord(record);
        TField f1 = fr.getDebitCurrency();
        TField f2 = fr.getCreditCurrency();

        if (!f1.getValue().equals(f2.getValue()))
        {
            f2.setError("CREDIT CURRENCY IS NOT EQUAL TO DEBIT CURRENCY");
        }
        return fr.getValidationResponse();
    }

}
```

'T' types in T24 - TValidationResponse

Record validation is a common mechanism in T24 to allow custom validation on records.

```
FTVersionDemo.java ✘
package com.newbank;

import java.util.List;

public class FTVersionDemo extends RecordLifecycle {

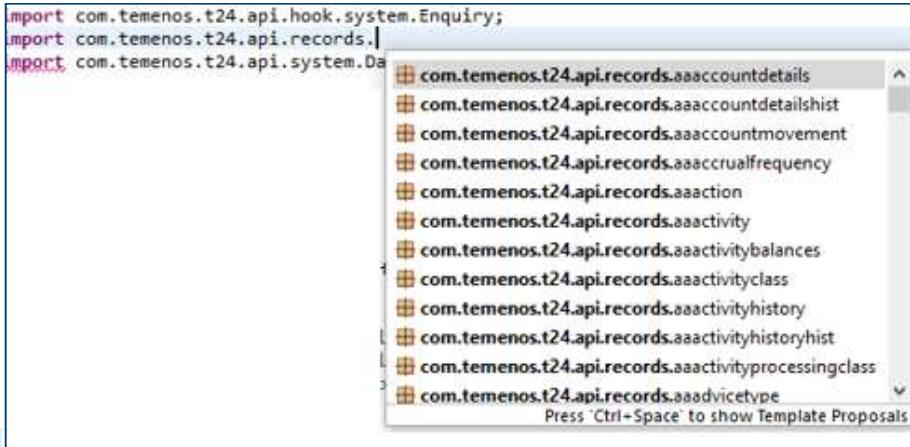
    @Override
    public TValidationResponse validateRecord(String application, String recordId, TStructure record,
                                              TStructure lastLiveRecord) {
        FundsTransferRecord fr = new FundsTransferRecord(record);
        TField f1 = fr.getDebitCurrency();
        TField f2 = fr.getCreditCurrency();

        if (!f1.getValue().equals(f2.getValue()))
        {
            f2.setError("CREDIT CURRENCY IS NOT EQUAL TO DEBIT CURRENCY");
        }
        return fr.getValidationResponse();
    }

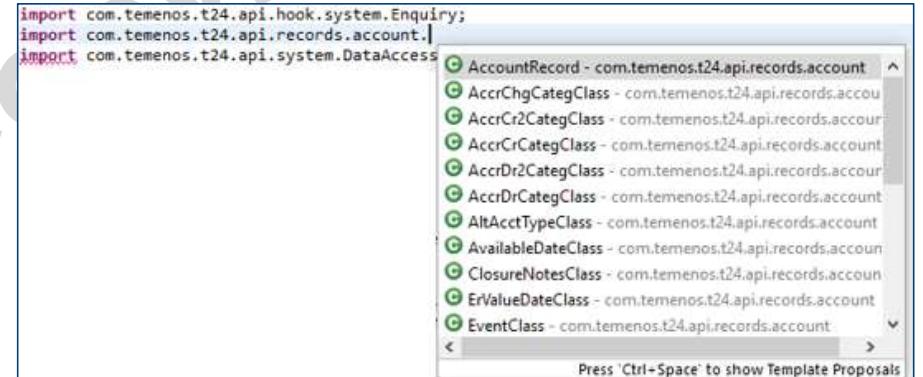
}
```

DataAccess

- The system.DataAccess class is a utility class to read, select and access data in T24.
- The api.records package is used to hold records from T24 applications



```
DataAccess da = new DataAccess(this);
AccountRecord AccRec = new AccountRecord(da.getHistoryRecord("ACCOUNT", currentId));
TField title = AccRec.getAccountTitle1(0);
return title.toString();
```



‘T’ types in T24 – T24Context

- T24Context is the way to establish a connection between Java application and T24
- TAFJClient.jar is required in JAVA_PROJECT to establish the connection.(available in TAFJ_HOME/lib).
- T24Context is used to set the credentials that pass through T24 Validation.
- Hook routines do not need a T24Context to connect to T24.

'T' types in T24 – T24Context

```
package com.newbank;

import com.temenos.t24.api.complex.st.customerapi.PersonalInfo;
import com.temenos.t24.api.party.Customer;
import com.temenos.tafj.api.client.impl.T24Context;

public class GetT24Context {

    public static void main(String[] args) {
        System.setProperty("tafj.home",
                           "C:/Temenos/TAFJ");
        T24Context ctx = new T24Context("tafj");
        ctx.setPassword("123456");
        ctx.setUser("INPUTTT");

        Customer customer = null;
        customer = new Customer(ctx);
        customer.setCustomerId("100352");

        PersonalInfo personalInfo = customer.getPersonalInfo();
        System.out.println("Nationality " + personalInfo.getNationality());
        System.out.println("Residence " + personalInfo.getResidence());
        System.out.println("Date of Birth " + personalInfo.getDateOfBirth());
    }
}
```

TAFJ
properties file

OUTPUT

(OFS.INITIALISE.SOURCE) : JAVA.FRAMEWORK
Nationality AU
Residence AU
Date of Birth 19711225

Complex class in T24

- No need to set/get related information in separate parameters.
- For e.g. Customer PersonalInfo is made up of DOB, nationality and residence

```
package com.newbank;

import com.temenos.t24.api.complex.st.customerapi.PersonalInfo;
import com.temenos.t24.api.party.Customer;
import com.temenos.tafj.api.client.impl.T24Context;

public class GetT24Context {

    public static void main(String[] args) {
        System.setProperty("tafj.home",
                           "C:/Temenos/TAFJ");
        T24Context ctx = new T24Context("tafj");
        ctx.setPassword("123456");
        ctx.setUser("INPUTTT");

        Customer customer = null;
        customer = new Customer(ctx);
        customer.setCustomerId("100352");

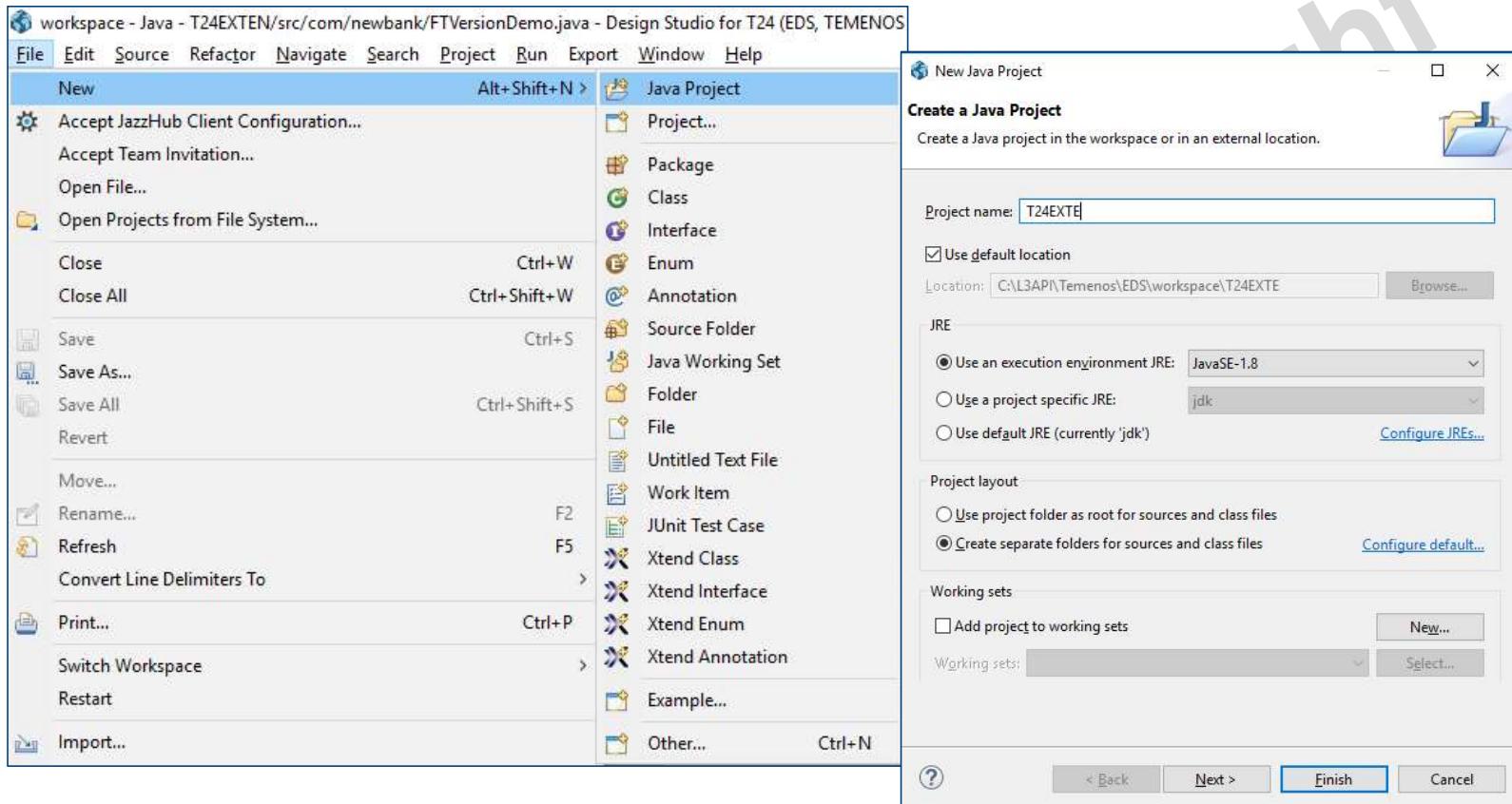
        PersonalInfo personalInfo = customer.getPersonalInfo();
        System.out.println("Nationality " + personalInfo.getNationality());
        System.out.println("Residence " + personalInfo.getResidence());
        System.out.println("Date of Birth " + personalInfo.getDateOfBirth());
    }
}
```

PersonalInfo
{
 dateOfBirth : date
 nationality : string
 residence : string
}

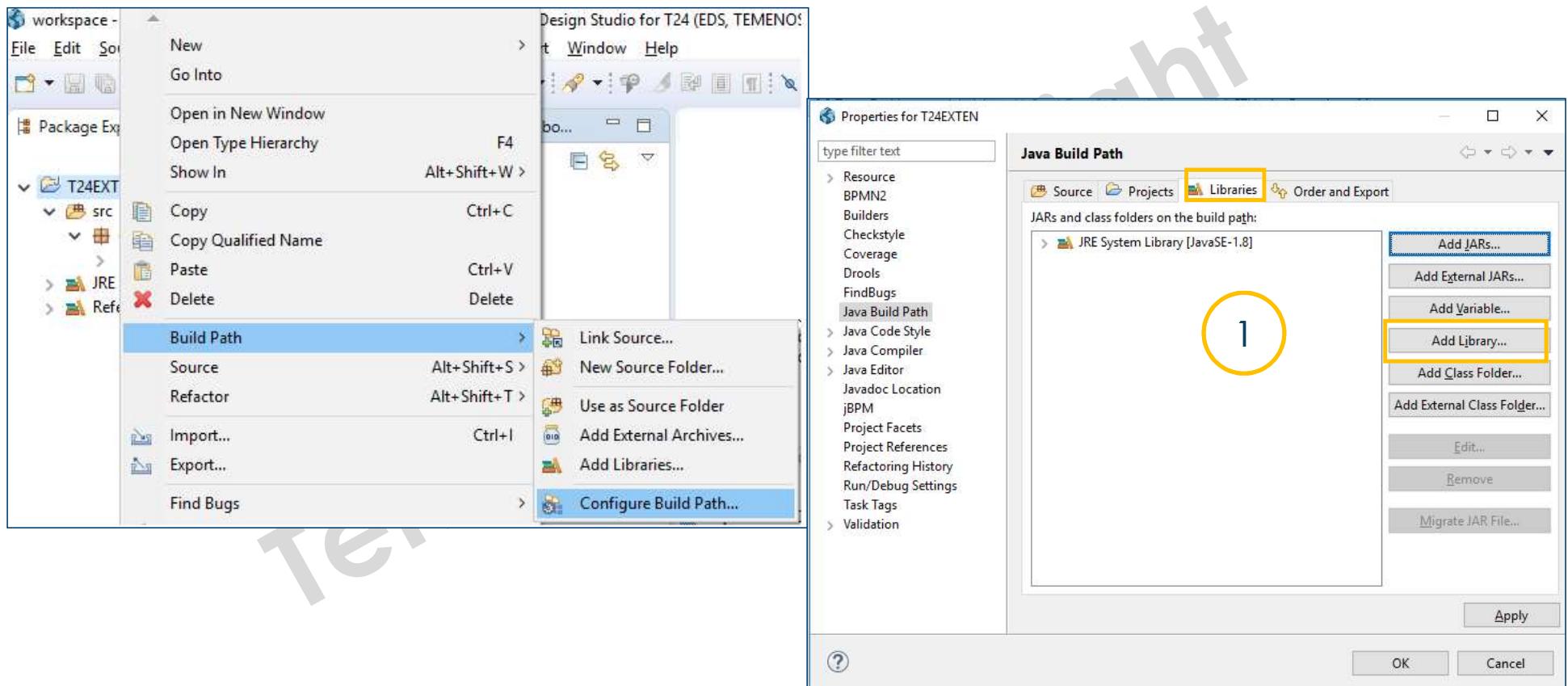
(OFS.INITIALISE.SOURCE) : JAVA.FRAMEWORK
Nationality AU
Residence AU
Date of Birth 19711225

Setup - Creating java project in Design Studio

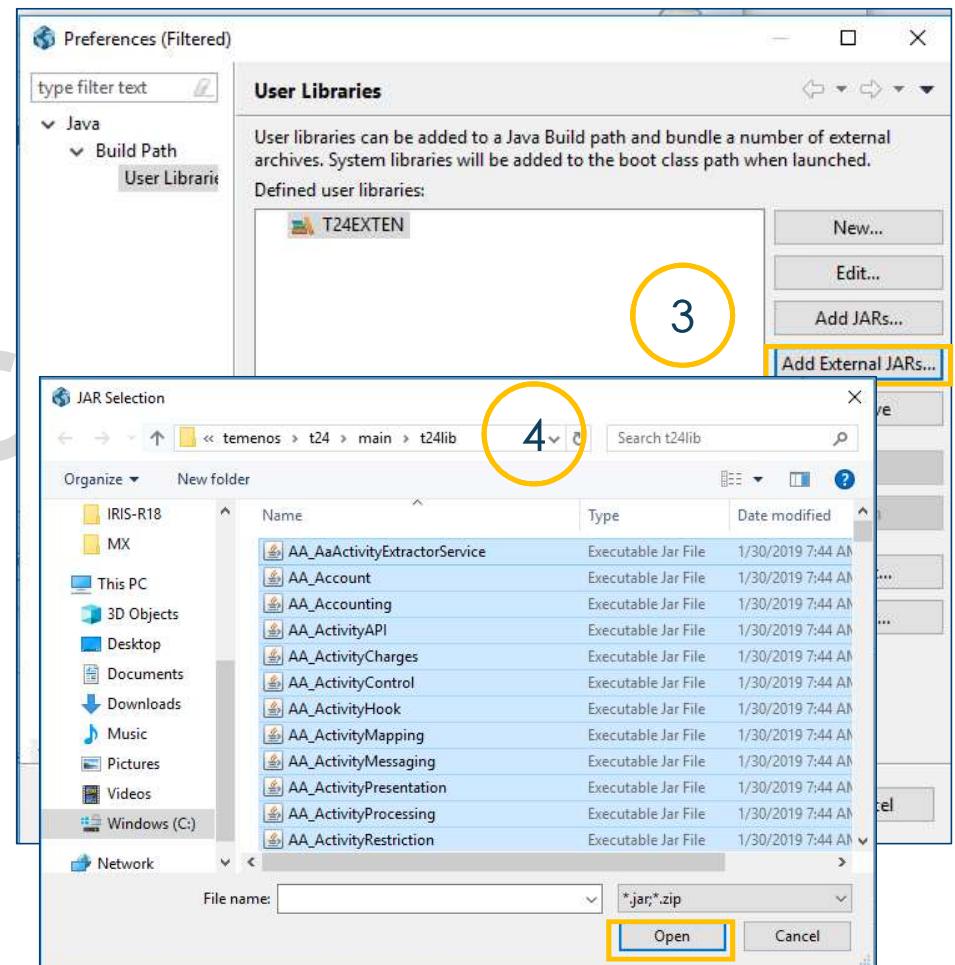
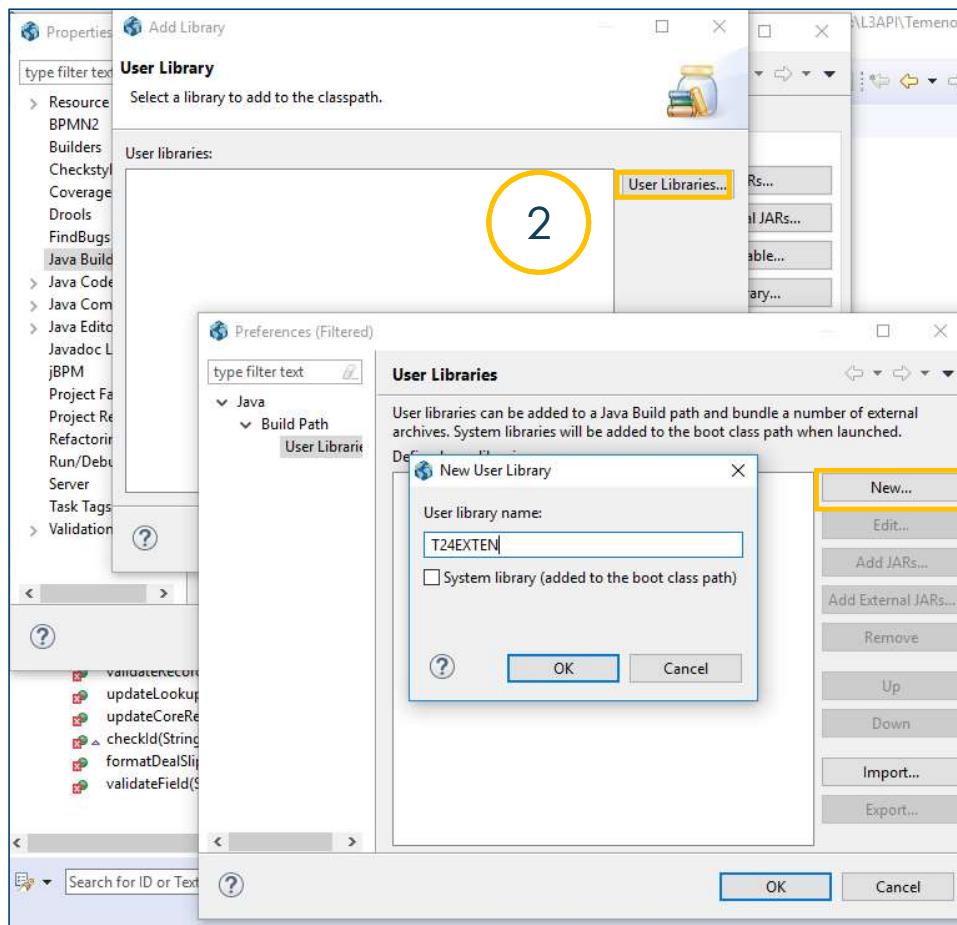
- File → New → Java Project (ProjectName)



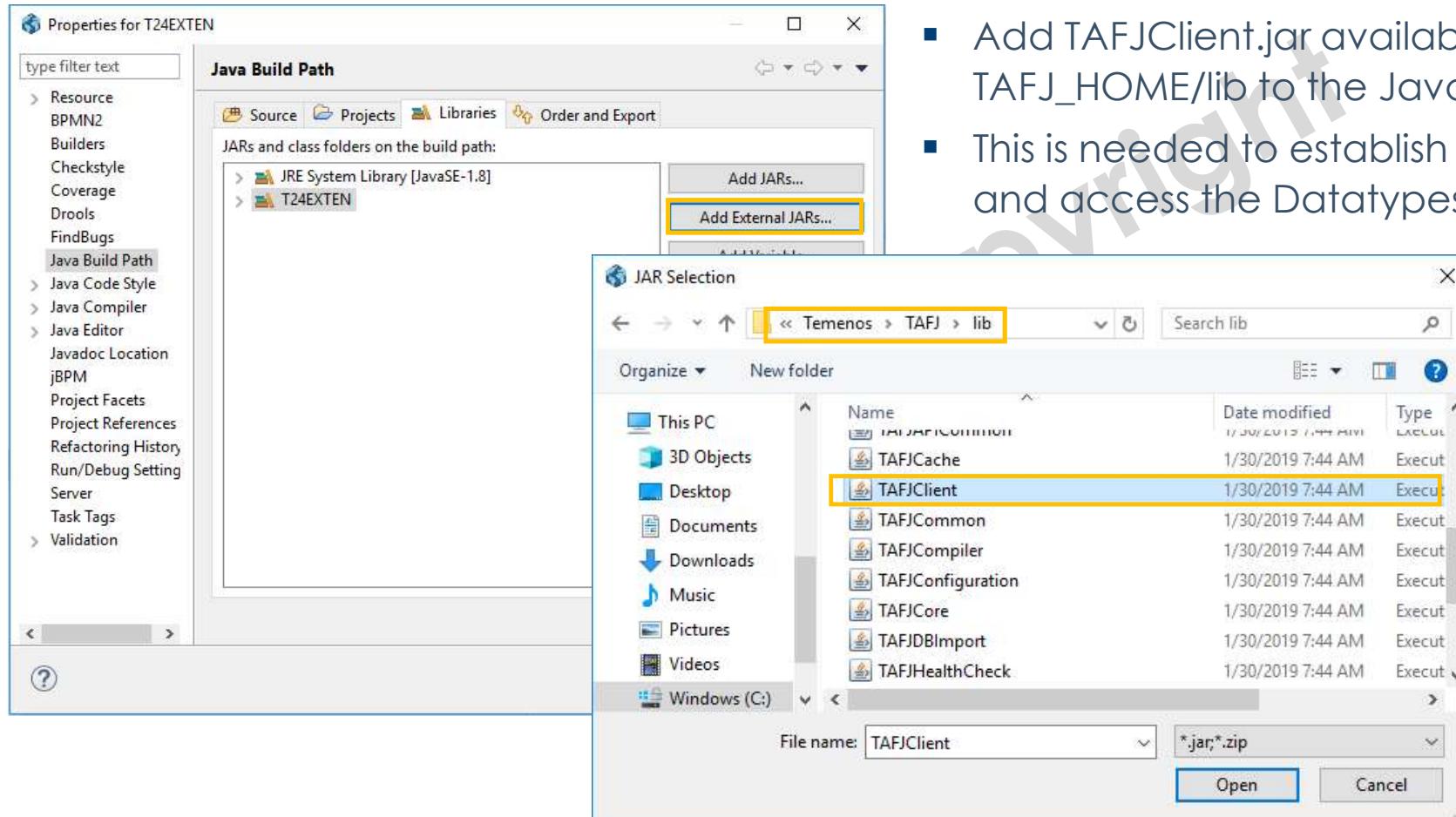
Setup - Configure Build Path – T24 Precompile



Configure Build Path – T24 Precompile



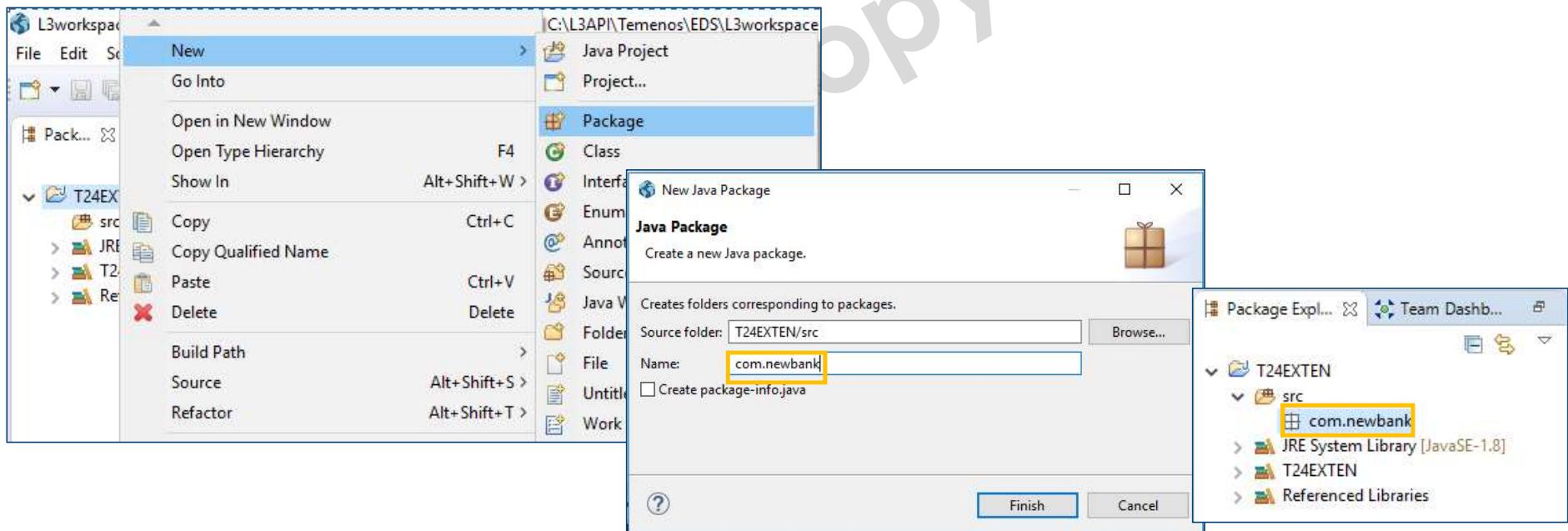
Configure Build Path - TAFJClient



- Add TAFJClient.jar available in TAFJ_HOME/lib to the Java Build Path
- This is needed to establish connection and access the Datatypes

Create package

- Java package is a mechanism for organizing Java classes into namespaces similar to the T24 modules.
- All classes go into a package
- Right Click on your project → New → Package → PackageName.

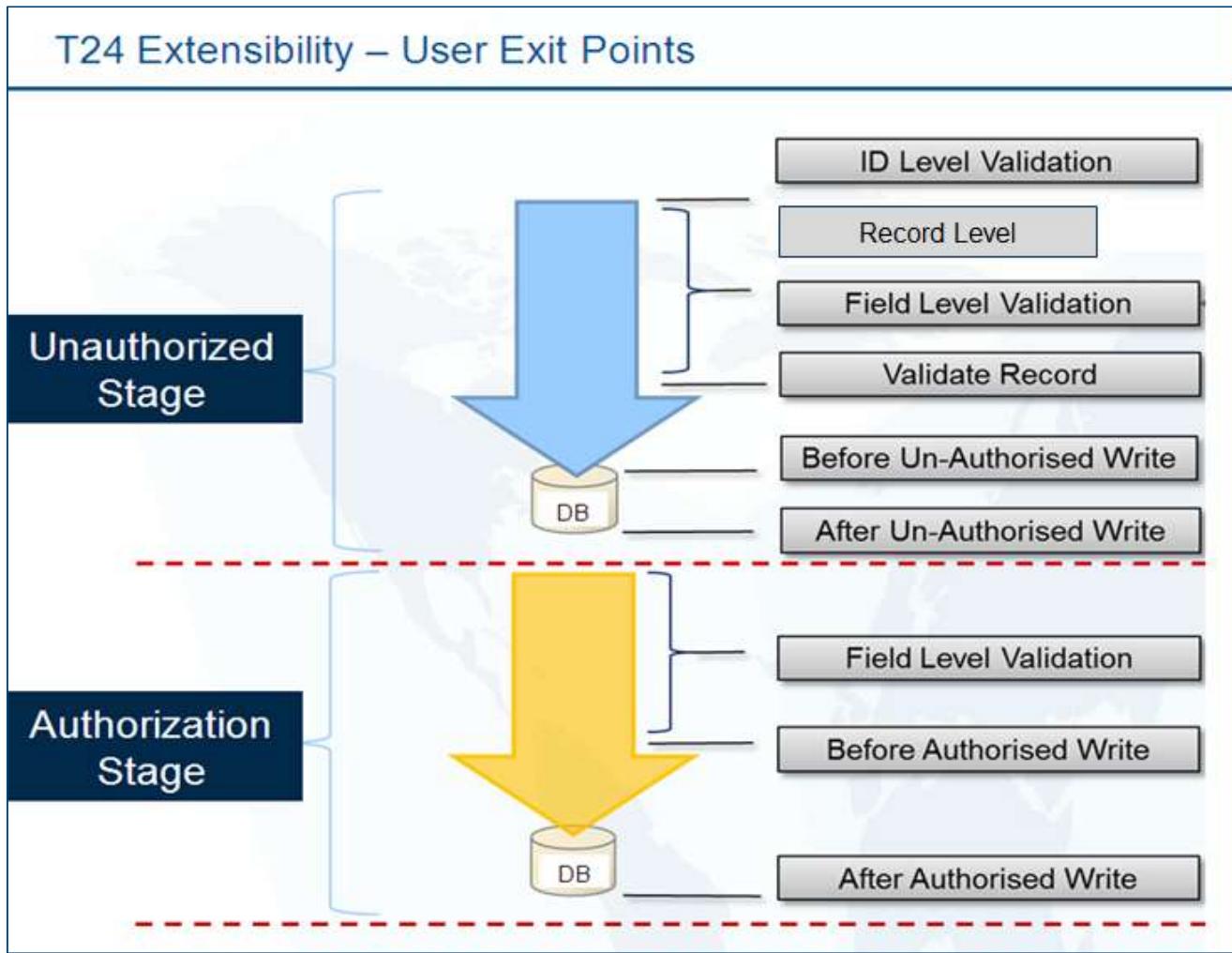


Lesson 2. VERSION Hooks in Java

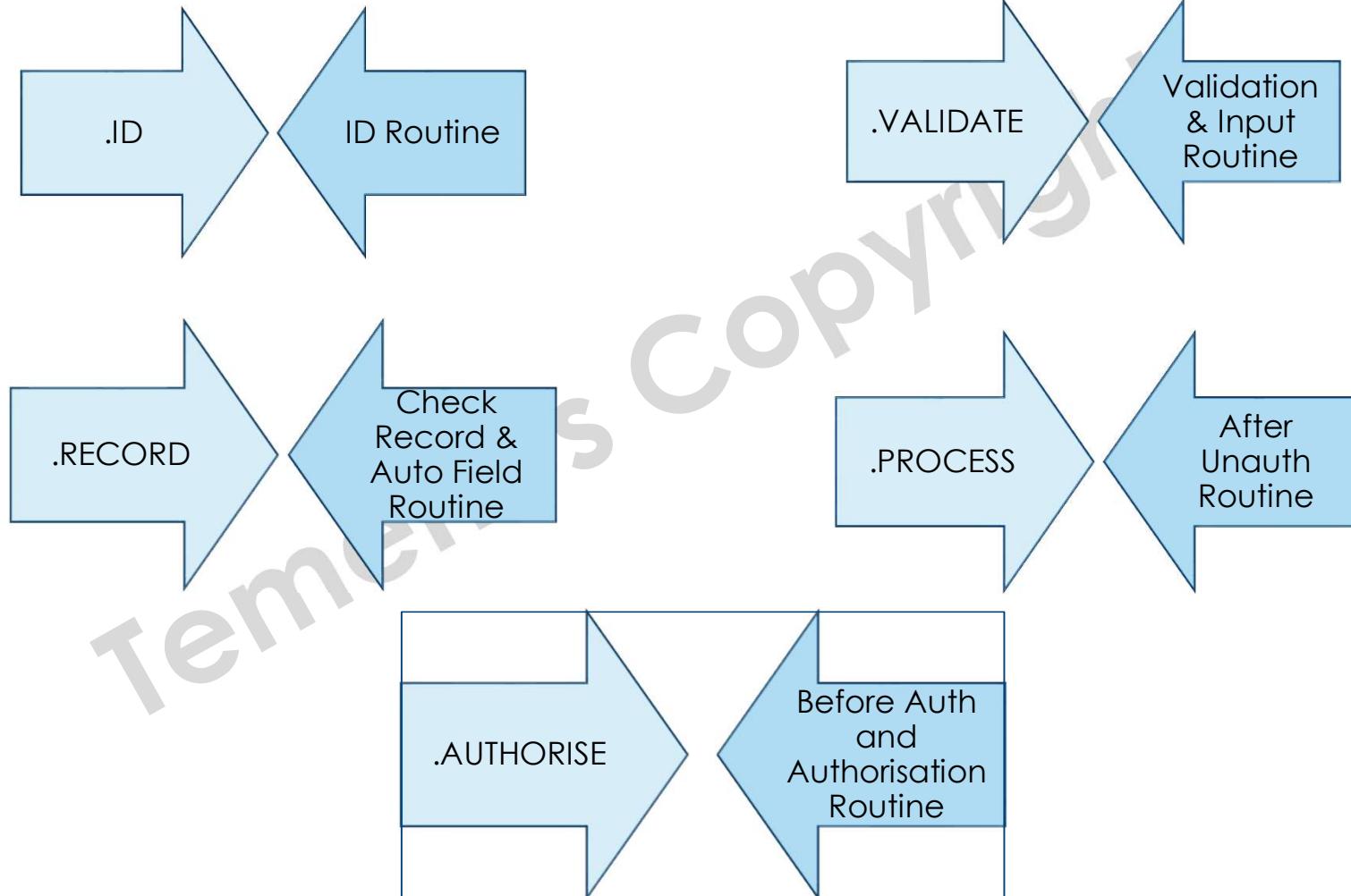
T24 Extensibility in Java



T24 Extensibility



Routine Invocation Stages in the Template Life Cycle



| VERSION HOOKS in T24

- Version hooks are attached to a VERSION
- **com.temenos.t24.api.hook.system.RecordLifecycle** has the following methods
 - checkId
 - defaultFieldValues
 - formatDealSlip
 - getDataDefinition
 - updateCoreRecord
 - updateLookupTable
 - validateField
 - validateRecord

DEMO – VERSION HOOK

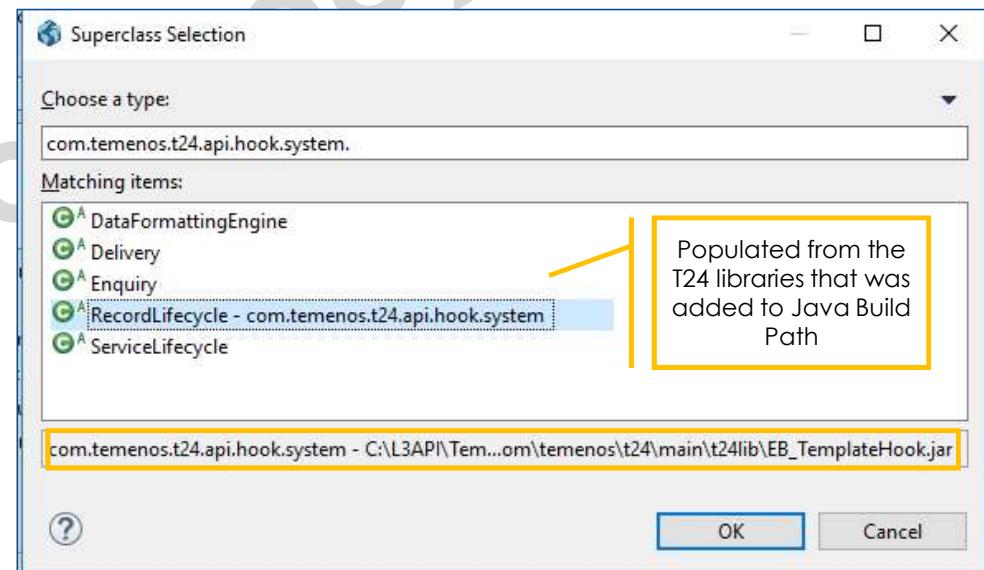
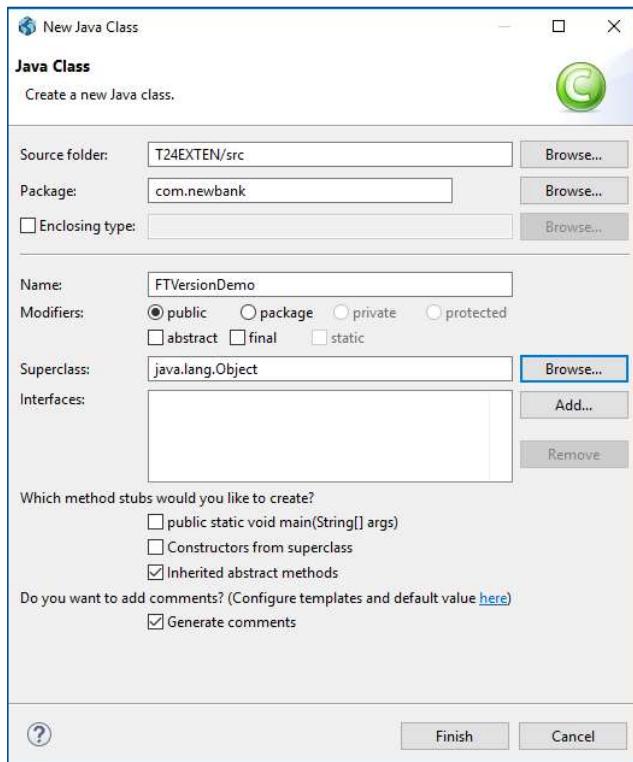
For a FUNDS.TRANSFER transaction, error must be raised when the DEBIT.CURRENCY and CREDIT.CURRENCY are not equal

Solution

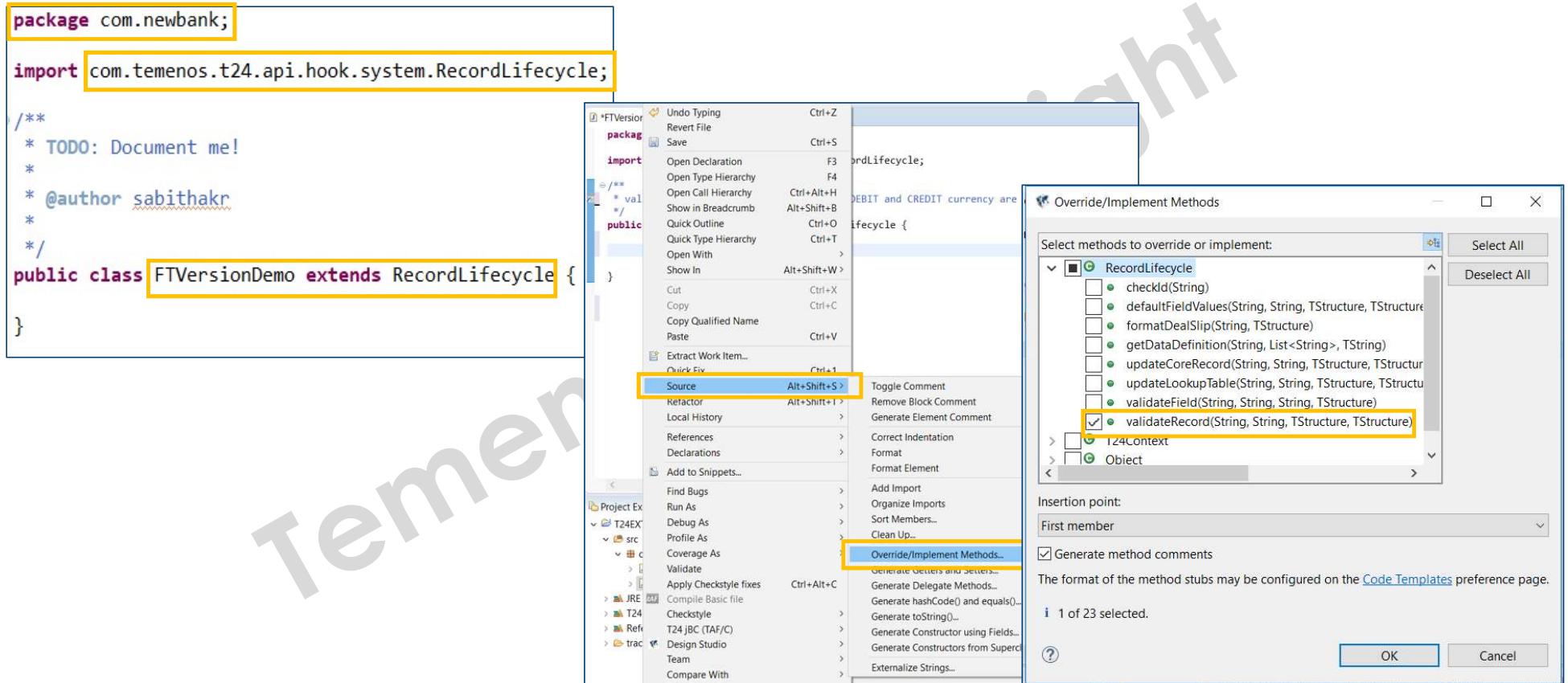
1. Create a class that extends **RecordLifecycle** and override the validateRecord()
2. Create JAR, add in module.xml and restart jBoss
3. Make an entry in EB.API
4. Create a VERSION and attach the EB.API record to the Input routine to validate if DEBIT.CURRENCY and CREDIT.CURRENCY are the same

Step 1 – Create class

- Right Click on your Package → New → Class



Step 1 – implement validateRecord



Step 1 – implement validateRecord

The screenshot shows the 'Override/Implement Methods' dialog box from a Java IDE. On the left, a tree view lists methods under 'RecordLifecycle'. The 'validateRecord(String, String, TStructure, TStructure)' method is selected and highlighted with a yellow border. On the right, the generated Java code is displayed:

```
package com.newbank;

import com.temenos.api.TStructure;
import com.temenos.api.TValidationResponse;
import com.temenos.t24.api.hook.system.RecordLifecycle;

/**
 * validateRecord is used to check if the DEBIT and CREDIT currency are the same
 */
public class FTVersionDemo extends RecordLifecycle {

    @Override
    public TValidationResponse validateRecord(String application, String recordId, TStructure record,
                                              TStructure lastLiveRecord) {
        // TODO Auto-generated method stub
        return super.validateRecord(application, recordId, record, lastLiveRecord);
    }
}
```

Step 1 – implement validateRecord

```
package com.newbank;

import com.temenos.api.TField;
import com.temenos.api.TStructure;
import com.temenos.api.TValidationResponse;
import com.temenos.t24.api.hook.system.RecordLifecycle;
import com.temenos.t24.api.records.fundstransfer.FundsTransferRecord;

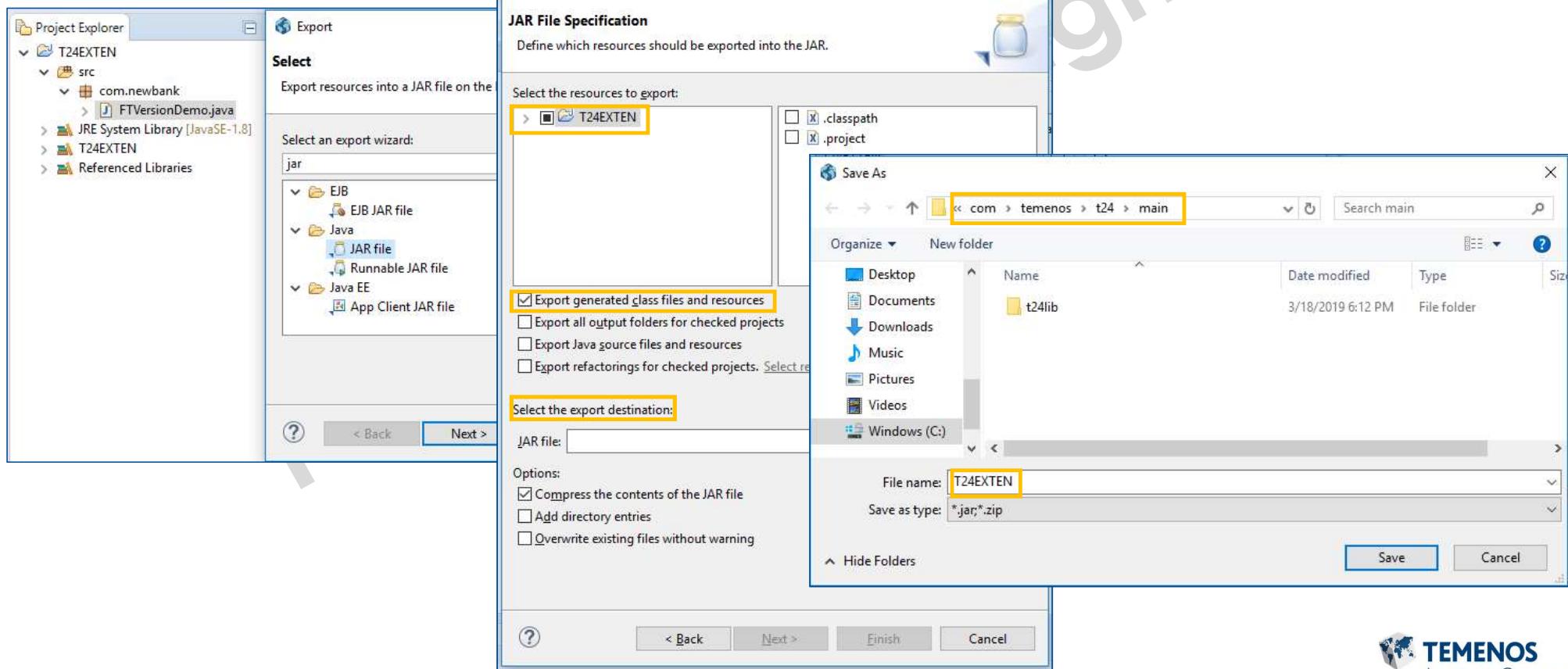
/**
 * validateRecord is used to check if the DEBIT and CREDIT currency are the same
 */
public class FTVersionDemo extends RecordLifecycle {

    @Override
    public TValidationResponse validateRecord(String application, String recordId, TStructure record,
                                              TStructure lastLiveRecord) {
        FundsTransferRecord fr = new FundsTransferRecord(record);
        TField f1 = fr.getDebitCurrency();
        TField f2 = fr.getCreditCurrency();

        if (!f1.getValue().equals(f2.getValue()))
        {
            f2.setError("CREDIT CURRENCY IS NOT EQUAL TO DEBIT CURRENCY");
        }
        return fr.getValidationResponse();
    }
}
```

Step 2 – Create the JAR and add to module.xml

- The .class file must be exported as a JAR and added to the module.xml in JBoss



Step 2 – Create the JAR and add to module.xml

- Add the JAR to the jBoss module.xml and restart jBoss

The screenshot shows a Windows File Explorer window. The address bar indicates the path: This PC > Windows (C:) > Temenos > jboss > modules > com > temenos > t24 > main. Inside the 'main' folder, there are three items: 't24lib' (File folder), 'module' (XML Document), and 'T24EXTEN' (Executable Jar File). The 'module' file is selected.

```
<module xmlns="urn:jboss:module:1.0" name="com.temenos.t24">
    <resources>
        <!-- Insert resources here -->
        <resource-root path=".//T24EXTEN.jar" />
        <resource-root path=".//t24lib/AA_ARAccountsData.jar" />
```

Step 3 – EB.API

- Hooks are invoked when a transaction is validated/committed
- Input Hooks are attached to the field INPUT.ROUTINE in the VERSION application
- All hooks must have an entry in EB.API

EB.API FTVERSIO.N.DEMO

Description	EN FT Version Hook
Protection Level *	<input checked="" type="radio"/> Full <input type="radio"/> Partial <input type="radio"/> None
Source Type *	<input type="radio"/> Basic <input type="radio"/> Java <input type="radio"/> Hook <input checked="" type="radio"/> Method
Java Method	validateRecord
Java Class	FTVersionDemo
Java Package	com.newbank

| Step 4 – Create/Modify version

- Modify the version FUNDS.TRANSFER.AC

Version Ds **FUNDS.TRANSFER.AC**

D Slip Trigger OI Rq

Input Routine.1 FTVERSION.DEMO FT Version Hook▼  

Auth Routine.1  

EB.API record is specified as the INPUT.ROUTINE

Launching the VERSION

- Error is raised when DEBIT.CURRENCY is not equal to CREDIT.CURRENCY

Transfer Between Accounts FT18107K2MR8

Please Select GO

Transfer Between Accounts Audit

Debit Account *	19461 WILLIAM HEWLETT	Debit Amount	123.00
Debit Currency	USD US Dollar	Debit Narrative	
Debit Value Date	17 APR 2018 17 APR 2018	Cheque Number	
Cheque Type		Credit Amount	
Ordered By.1			
Credit Account *	11223 NIKE		
Credit Currency	GBP Pound Ster		

CREDIT CURRENCY IS NOT EQUAL TO DEBIT CURRENCY

Please resolve the errors below to proceed:

X Credit Currency: CREDIT CURRENCY IS NOT EQUAL TO DEBIT CURRENCY

Launching the VERSION

- Error is not raised when DEBIT.CURRENCY equal to CREDIT.CURRENCY

Transfer Between Accounts FT18107K2MR8

✓ ⓘ ⏪ ⏴ ⓘ - Please Select

Information:
Field has been validated

Transfer Between Accounts		Audit
Debit Account *	19461	WILLIAM HEWLETT
Debit Currency	USD	US Dollar
Debit Value Date	17 APR 2018	17 APR 2018
Cheque Type		
Ordered By.1		
Credit Account *	21261	ZURICH EQUITY F
Credit Currency	USD	US Dollar
Credit Value Date	17 APR 2018	17 APR 2018
Debit Amount	123.00	
Debit Narrative		
Cheque Number		
Credit Amount		
Credit Narrative		

Lesson 3. DEBUG Java code in Design Studio

T24 Extensibility in Java



| DEMO – DEBUG the Java code

- The Design Studio Java IDE provides many debugging tools and views grouped in the Debug Perspective
- To debug the program, define breakpoints. By adding breakpoints in the source code we can specify where the execution of the program should pause
- To set breakpoints in the source code double click on the small left margin in the source code editor.
- DS uses eclipse standard for debugging
 - Step In : F5
 - Step Over : F6
 - Step Out : F7
 - Continue : F8



DEBUG the Java code

- Remote Debug jBoss from Design Studio
- Set JAVA_OPTS

For Windows

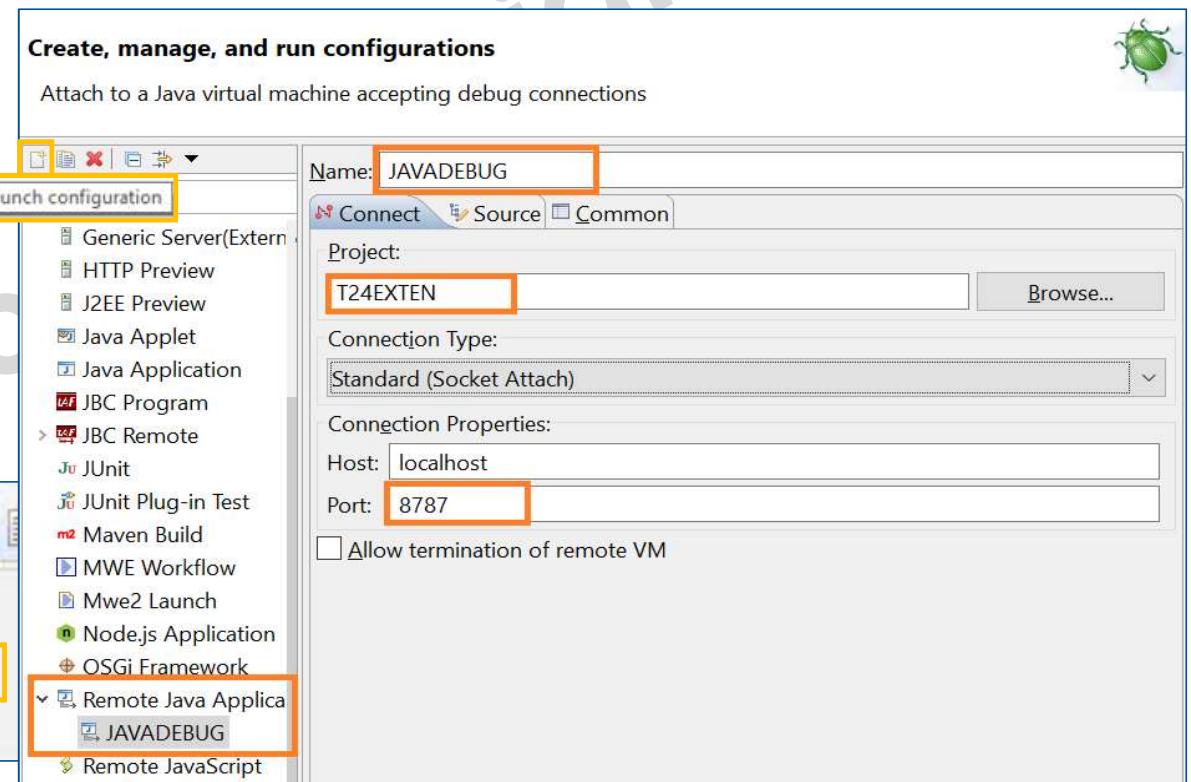
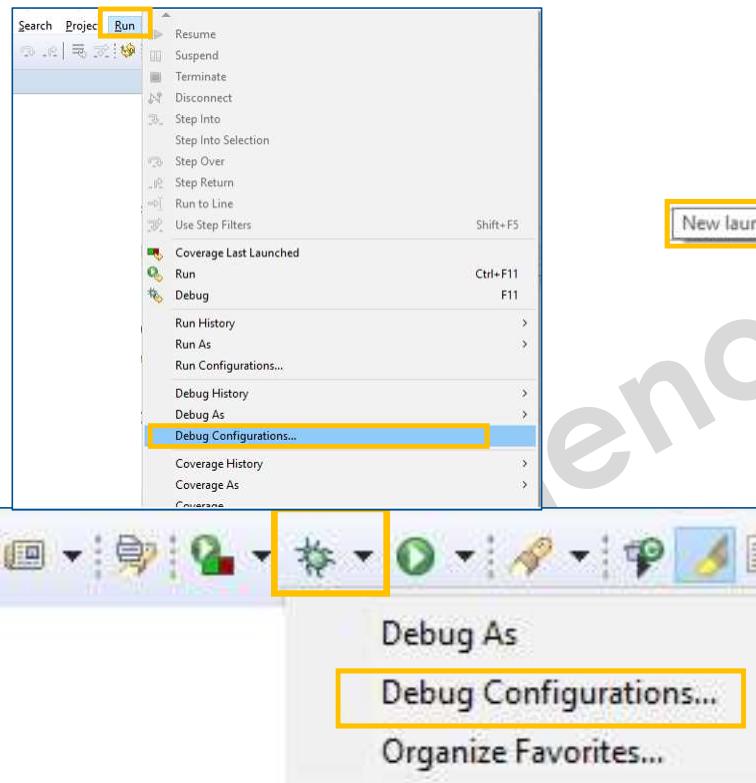
- SET JAVA_OPTS=-Xdebug -Xnoagent -Xrunjdwp:transport=dt_socket, address=8787,server=y,suspend=n%JAVA_OPTS%

For Linux

- JAVA_OPTS="-Xdebug -Xnoagent -Xrunjdwp:transport=dt_socket, address=8787,server=y,suspend=n\$JAVA_OPTS"

DEBUG the Java code

- Create the DEBUG configuration



DEBUG the java code

The screenshot shows a Java application interface for "Transfer Between Accounts" with the identifier FT18107370J1. The application has tabs for "Transfer Between Accounts" (selected) and "Audit". The "Transfer Between Accounts" tab contains fields for "Debit Account" (19461, WILLIAM HEWLETT), "Debit Currency" (USD, US Dollar), "Debit Value Date" (DD MMM YYYY), "Cheque Type", "Ordered By", and "Credit Account" (11223). The "Audit" tab is currently inactive.

The main area displays a stack trace for a suspended thread:

```
Thread [default task-15] (Suspended (breakpoint at line 20 in FTVersionDemo))
  FTVersionDemo.validateRecord(String, String, TStructure, TStructure) line: 20
  NativeMethodAccessorImpl.invoke0(Method, Object, Object[])
  NativeMethodAccessorImpl.invoke(Object, Object[])
  DelegatingMethodAccessorImpl.invoke(Object, Object[])
  Method.invoke(Object, Object[])
  component_EB_TemplateHook_19_cl.validateRecord(Object, Object, Object, Object, Object) line: not available
  EB_TEMPLATE_HOOK_VALIDATE_RECORD_INVOKER_cl.main() line: not available
  EB_TEMPLATE_HOOK_VALIDATE_RECORD_INVOKER_cl.invoke(Object, Object, Object) line: not available
  EB_TEMPLATE_HOOK_VALIDATE_RECORD_INVOKER_cl.invoke(Object...) line: not available
  component_EB_TemplateHook_19_cl.validateRecordInvoker(Object, Object, Object) line: not available
```

A yellow box highlights the line "Method.invoke(Object, Object...) line: 498" with the label "Debugger view".

The right side of the interface includes several views:

- Variables and expressions view:** Shows variables and their values, including recordId, record, lastLiveRecord, fr, f1, and f2. A yellow box highlights "f1" with the label "Variables and expressions view".
- Outline view:** Shows the class structure of FTVersionDemo, specifically the validateRecord method.
- Code editor:** Displays the source code of FTVersionDemo.java, highlighting the current instruction pointer at line 20.

A yellow box highlights the line "TField f1 = fr.getDebitCurrency();" with the label "The current instruction pointer".

```
@Override
public TValidationResponse validateRecord(String application, String recordId, TStructure record,
    TStructure lastLiveRecord) {
    FundsTransferRecord fr = new FundsTransferRecord(record);
    TField f1 = fr.getDebitCurrency();
    TField f2 = fr.getCreditCurrency();

    if (!f1.getValue().equals(f2.getValue()))
    {
        f2.setError("CREDIT CURRENCY IS NOT EQUAL TO DEBIT CURRENCY");
    }
    return fr.getValidationResponse();
}
```

Practice – Version Hook CheckRec

- Create a VERSION for ACCOUNT application that allows user to edit records belonging to CATEGORY 1001 only
- Expected Output



The screenshot shows a TEMENOS application window with the title bar "ACCOUNT,CAT1001| 19461". The main area displays a "Current Account" record for "WILLIAM HEWLETT". The record includes fields for Customer ID (100366), Product Code (1001), Currency (USD), Mnemonic (HEWWILLUSD), Account Name 1 (EN WILLIAM HEWLETT), and Account Name 2 (EN). A yellow arrow points from the "19461" account number to a callout box containing the text "ACCOUNT in CATEGORY 1001 is editable".

Practice – Version Hook for ID

- Create a VERSION for the CUSTOMER that prefixes “99” to the ID given by the user
- Expected Output

The screenshot shows a TEMENOS application window titled "CUSTOMER, ID99177". The title bar also displays "TEMENOS". The main area is labeled "Basic Details" and contains the following fields:

- Title:** A dropdown menu showing "- Please Select".
- Given Name:** An input field containing "EN".
- Family Name:** An input field containing "EN".
- Full Name ***: An input field containing "EN".
- Full Name-2**: An input field containing "EN".
- Short Name ***: An input field containing "EN".

Lesson 3. ENQUIRY Hooks in Java

T24 Extensibility in Java



| Enquiry hooks

We can customize ENQUIRY by

modifying the dynamic selection box – BUILD hook

modifying the data before display – CONVERSION hook

Extend the **com.temenos.t24.api.hook.system.Enquiry** class

setRecord()

setFilterCriteria()

setValue()

setIds()

| DEMO – Enquiry Build hook

- The Build hook in ENQUIRY, is used to modify the selection criteria
- We can set the Field Name, Operator and the value and add these to the filter criteria

Create an enquiry to display Customer Id, Account Id and the respective working balance after accepting category from the user.

Based on category, additional filtering has to be done

- For category 1001 - Balance should be in the range 0 and 49999
- For category 6001 - Balance should be in the range 50000 and 100000
- For category greater than 7000 - Balance should be in the range 100000 and 500000

Step 1 – Create the class

The screenshot illustrates the process of creating a new Java class named `ACEnquiryDemo` that extends `Enquiry`. The steps are as follows:

- New Java Class Dialog:** Shows the configuration for the new class. Source folder is `T24EXTEN/src`, package is `com.newbank`, and superclass is `java.lang.Object`.
- Superclass Selection Dialog:** Shows the selection of `Enquiry` as the superclass. The matching item is `com.temenos.t24.api.system.Enquiry`.
- Override/Implement Methods Dialog:** Shows the selection of methods to override or implement. Under the `Enquiry` node, the `setFilterCriteria` method is selected.

The generated code in the editor is:

```
package com.newbank;

import com.temenos.t24.api.hook.system.Enquiry;

/**
 * demo for Build routine
 */
public class ACEnquiryDemo extends Enquiry { }
```

Step 1 – Create the class

```
package com.newbank;

import java.util.List;

import com.temenos.t24.api.complex.eb.enquiryhook.FilterCriteria;
import com.temenos.t24.api.hook.system.Enquiry;

/**
 * demo for ENQUIRY Build hook
 */
public class ACEnquiryDemo extends Enquiry {

    @Override
    public List<FilterCriteria> setFilterCriteria(String enquiryName, List<FilterCriteria> filterCriteria) {
        FilterCriteria fc = new FilterCriteria();
        String s1 = filterCriteria.get(0).getFieldname();
        String s3 = filterCriteria.get(0).getValue();

        fc.setFieldname("WORKING.BALANCE");
        fc.setOperand("RG");
        if (s1.equals("CATEGORY")) {
            if (Integer.parseInt(s3) > 7000) {
                fc.setValue("50000 100000");
            } else {
                switch (s3) {
                    case "1001":
                        fc.setValue("0 10000");
                        break;
                    case "6001":
                        fc.setValue("10000 50000");
                        break;
                    default:
                }
            }
        }
        filterCriteria.add(fc);
        return filterCriteria;
    }
}
```

Step 2 – Create JAR and update module.xml

The screenshot shows the Eclipse IDE interface with three main windows:

- Project Explorer**: Shows the project structure with a package named `com.newbank` containing files `ACEquiryDemo.java` and `FTVersionDemo.java`.
- Export**: A wizard window titled "Select" for exporting resources into a JAR file. It lists export wizards: JAR file, Javadoc, Runnable JAR file, Java EE (App Client JAR file, EAR file), and FAR file.
- JAR Export**: A configuration window for specifying resources to export. It shows the project `T24EXTEN` selected, with its source folder `src` and package `com.newbank` checked. Other options like `.classpath` and `.project` are available but unchecked. There are also checkboxes for generated class files, refactoring, and compressing the JAR.
- File Explorer**: A table showing the contents of the current directory:

Name	Date modified	Type	Size
t24lib	1/30/2019 7:44 AM	File folder	
module	2/18/2019 6:31 PM	XML Document	71 KB
T24Exten	2/18/2019 10:03 PM	Executable Jar File	3 KB
- module.xml**: An XML editor showing the configuration for the module. The code is as follows:

```
<module xmlns="urn:jboss:module:1.0" name="com.temenos.t24">
<resources>
    <!-- Insert resources here -->
    <resource-root path="./T24Exten.jar" />
    <resource-root path="./t24lib/AA_ARAccountsData.jar" />
    <resource-root path="./t24lib/AA_ARC.jar" />
```

Step 3 – EB.API

```
package com.newbank;

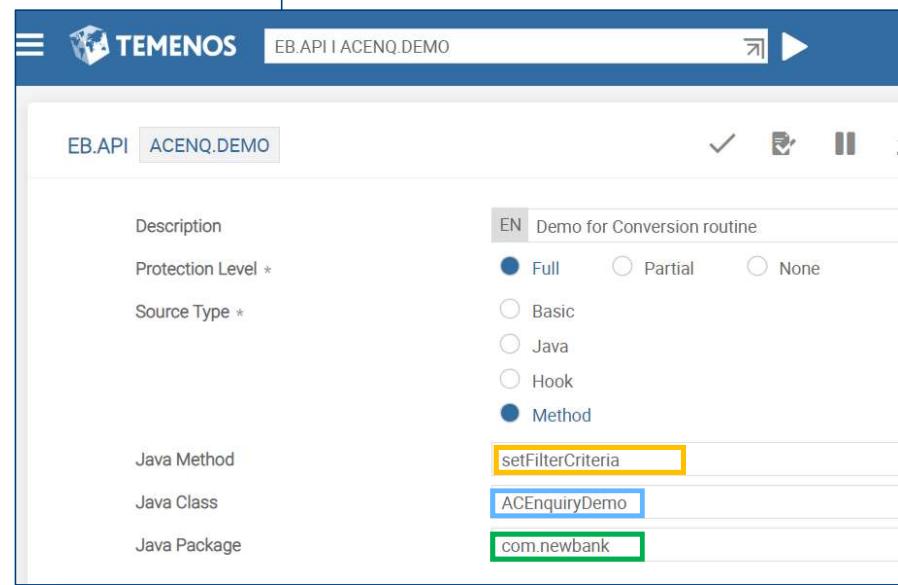
import java.util.List;

import com.temenos.t24.api.complex.eb.enquiryhook.FilterCriteria;
import com.temenos.t24.api.hook.system.Enquiry;

/**
 * demo for ENQUIRY Build hook
 */
public class ACEnquiryDemo extends Enquiry {

    @Override
    public List<FilterCriteria> setFilterCriteria(String enquiryName, List<FilterCriteria> filterCriteria) {
        FilterCriteria fc = new FilterCriteria();
        String s1 = filterCriteria.get(0).getFieldname();
        String s3 = filterCriteria.get(0).getValue();

        fc.setFieldname("WORKING.BALANCE");
        fc.setOperand("RG");
        if (s1.equals("CATEGORY")) {
            if (Integer.parseInt(s3) > 7000) {
                fc.setValue("50000 100000");
            } else {
                switch (s3) {
                    case "1001":
                        fc.setValue("0 10000");
                        break;
                    case "6001":
                        fc.setValue("10000 50000");
                        break;
                    default:
                }
            }
        }
        filterCriteria.add(fc);
        return filterCriteria;
    }
}
```



Step 4 – Create ENQUIRY and attach hook



EMQUIRY,DS | ACENQ ➔

Enquiry Ds ACENQ

Sort.1 WORKING.BALANCE DSND

+ Open Bracket.1 Yes
Selection Flds.1 CATEGORY
Sel Label.1 EN
Sel Fld Oper.1
Required Sel.1
Close Bracket.1 Yes
Rel Next Field.1 Or And

Build Routine.1 ACENQ.DEMO

Header
+ @^(1,1)Build routine Demo

A large diagonal watermark "copyright" is visible across the form.

Launch the ENQUIRY

The image displays three separate TEMENOS application windows, each showing a table of account information under a "Build routine Demo" header. The windows are arranged horizontally.

Top Window (Left): ENQ ACENQ CATEGORY EQ 1001

Account Number	Category	Ccy	Working Balance
74918	1001	USD	10,000.00
76341	1001	EUR	10,000.00
77868	1001	GBP	10,000.00
78007	1001	USD	10,000.00
81884	1001	CAD	10,000.00
83666	1001	USD	10,000.00
76945	1001	USD	9,960.00
78883	1001	GBP	9,420.25
81744	1001	EUR	9,000.00
76748	1001	USD	8,013.26
77445	1001	USD	8,000.00
78581	1001	USD	8,000.00
81752	1001	USD	8,000.00
74567	1001	GBP	7,658.74
74586	1001	GBP	7,658.74
78794	1001	GBP	7,658.74
74578	1001	GBP	7,558.74
78042	1001	USD	7,500.00
78514	1001	USD	7,000.00
77698	1001	GBP	6,000.00
34533	1001	USD	5,300.00
75515	1001	USD	5,040.00
81768	1001	GBP	4,500.00
83224	1001	USD	4,500.00

Middle Window (Center): ENQ ACENQ CATEGORY EQ 6001

Account Number	Category	Ccy	Working Balance
82384	6001	USD	50,000.00
78937	6001	USD	46,245.00
82414	6001	USD	45,000.00
78905	6001	USD	44,000.00
78867	6001	USD	20,885.00
74381	6001	USD	18,000.00
76422	6001	USD	17,194.73
78719	6001	USD	15,650.00
74365	6001	USD	15,000.00
17531	6001	EUR	14,753.43
78875	6001	USD	13,000.00
78751	6001	USD	12,000.00
78573	6001	GBP	10,482.55

Bottom Window (Right): ENQ ACENQ CATEGORY GT 7001

Account Number	Category	Ccy	Working Balance
GBPUSD1902201010001	19022	GBP	100,000.00
USD1661100040001	16611	USD	95,000.00
USD1422000060001	14220	USD	94,298.93
USD1660400060001	16604	USD	78,750.00
EUR1419700010001	14197	EUR	67,377.08
EUR1401000010001	14010	EUR	59,155.44
USD1660400050001	16604	USD	55,150.00

Practice – Enquiry Conversion hook

- Create an ENQUIRY TO LIST the LD records that are in foreign currency.
List the ID, CUSTOMER, AMOUNT and Equivalent amount in LCCY.
- Use conversion hook to convert the amount in foreign currency to LCCY
- Expected output

The screenshot displays three components illustrating the conversion process:

- Currency Conversion Window:** Shows "CURRENCY JPY Japanese Yen" and "Mid Reval Rate.1" with the value **110.624307**, highlighted by a yellow box.
- Calculator Window:** Shows the calculation **75000 ÷ 110.624307 = 677.9703487769645**.
- TEMENOS Application Output:** Shows the query **ENQ LDENQ CURRENCY EQ JPY** and the results table titled "Conversion routine demo". The table includes columns: Contract Number, Product, Customer Id, Currency, Loan Amount, and Loan Amount (highlighted by a yellow box). The data rows are:

Contract Number	Product	Customer Id	Currency	Loan Amount	Loan Amount
LD1808014199	21003	190002	JPY	75,000.00	677.97
LD1809452323	21010	190114	JPY	40,000,000.00	361,584.19
LD1809497668	21074	190114	JPY	4,000,000.00	36,158.42

NOFILE Enquiry

- What if there are multiple files involved in creation of an enquiry?
- What if the record ID of another application is not part of the base application?
- This can be achieved using NOFILE enquiries
- Since more than one file might be required to extract data, it is a NOFILE enquiry. We can extract data from any number of T24 applications
- Subroutine is written to extract all required data and form the record
- **ENQUIRY** application is used to display extracted data

DEMOS – NOFILE enquiry

- Create an ENQUIRY to display the consolidated balance for a CUSTOMER.
- Hint : Accumulate all the balance from all the accounts available in CUSTOMER.ACOUNT.
- Expected Output

AUTHORISER Last signed on 15 APR 2019 at 10:28 with 0 attempt(s)
Help Tools Sign Off
ENQ NOFILE.ENQ1

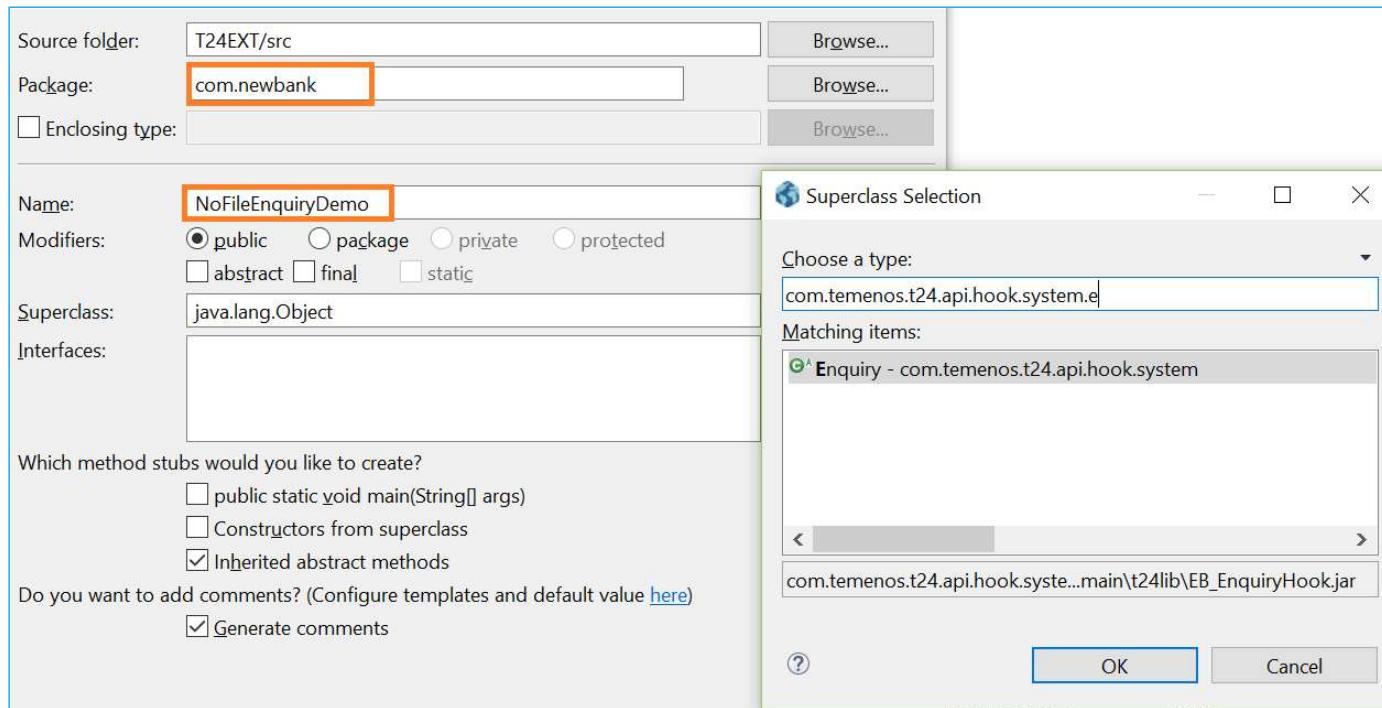
NOFILE.ENQ1 - Google Chrome
localhost:9089/BrowserWeb/servlet/BrowserServlet

Results 1 - 1 of 1

CUSTOMER	CONSOLIDATED.BALANCE
100100	507,263.22

Favourites More Options Clear Selection Find
CUSTOMER equals * 100100
GB0010001 : NOFILE.ENQ1

Step 1 – Create the class



The screenshot shows a Java code editor with the tab NoFileEnquiryDemo.java selected.

```
@Override  
public List<String> setIds(List<FilterCriteria> filterCriteria) {  
    Customer cusrec = new Customer(this);  
    DataAccess da = new DataAccess(this);  
    String s1 = filterCriteria.get(0).getFieldname();  
    String s3 = filterCriteria.get(0).getValue();  
    List<String> retId = new ArrayList<String>();
```

Step 1 – Create the class

```
NoFileEnquiryDemo.java
package com.newbank;

import java.util.ArrayList;

public class NoFileEnquiryDemo extends Enquiry {

    @Override
    public void setRecord(String value, String currentId, TStructure currentRecord,
        List<FilterCriteria> filterCriteria) {
    }

    @Override
    public List<FilterCriteria> setFilterCriteria(String enquiryName, List<FilterCriteria> filterCriteria) {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public String setValue(String value, String currentId, TStructure currentRecord,
        List<FilterCriteria> filterCriteria) {
        // TODO Auto-generated method stub
        return null;
    }

    public List<String> setIds(List<FilterCriteria> filterCriteria) {
        Customer cusrec = new Customer(this);
        DataAccess da = new DataAccess(this);
        String s1 = filterCriteria.get(0).getFieldname();
        String s3 = filterCriteria.get(0).getValue();
        List<String> retId = new ArrayList<String>();
        Double consAmt = 0.0;
        if (s1.equals("CUSTOMER")) {
            if (!s3.equals("")) {
                cusrec.setCustomerId(s3);
                List<String> accnum = null;
                try {
                    accnum = cusrec.getAccountNumbers();
                    for (String acnum : accnum) {
                        AccountRecord ar = new AccountRecord(da.getRecord("ACCOUNT", acnum));
                        String amt = ar.getWorkingBalance().getValue();
                        if (amt.isEmpty() || amt.equals("")) {
                            continue;
                        } else {
                            double amt1 = Double.valueOf(amt);
                            consAmt += amt1;
                        }
                    }
                } catch (Exception e) {
                }
                retId.add(Double.toString(consAmt));
            }
        }
        return retId;
    }
}
```

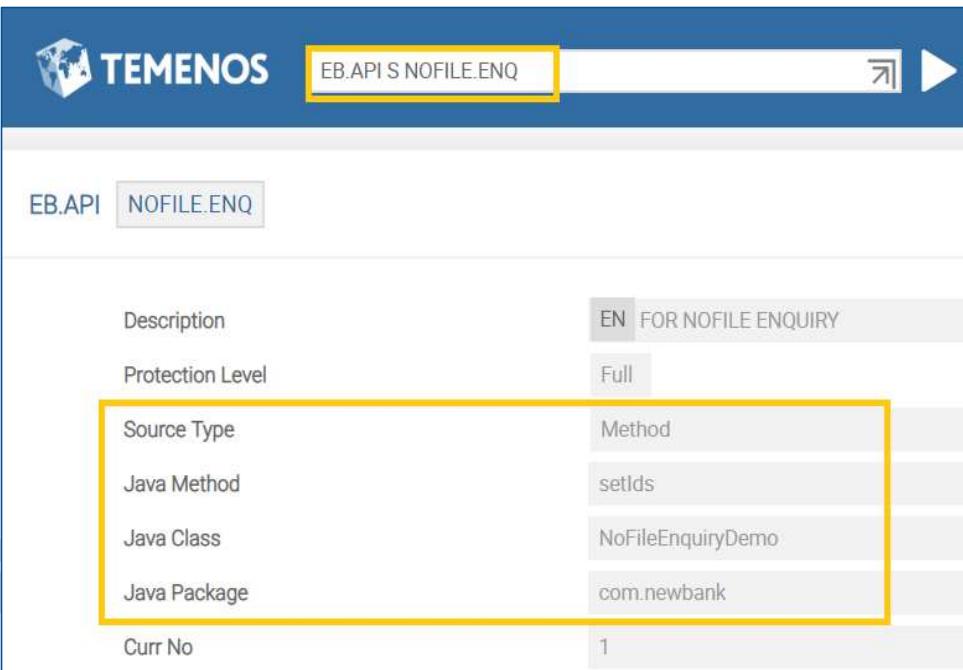
Step 2 – Create JAR and update module.xml

The screenshot shows the Eclipse IDE interface with the following components:

- Project Explorer:** Shows the project structure with a package named `T24EXTEN` containing a source folder `src` with files `ACEnquiryDemo.java` and `FTVersionDemo.java`.
- Export Dialog:** A modal window titled "Select" for exporting resources into a JAR file. It lists export wizards: "JAR file", "Javadoc", "Runnable JAR file", "Java EE" (with "App Client JAR file" and "FAR file" options), and "Next >" and "Finish" buttons.
- JAR Export Dialog:** A modal window titled "JAR Export" for specifying resources to export. It shows the selected project `T24EXTEN` and its `src` folder, along with checked options for ".classpath" and ".project". It also includes checkboxes for "Export generated class files and resources" and "Export all output folders for checked projects".
- module.xml Editor:** An XML editor showing the configuration for the module. The XML code is:

```
1 <module xmlns="urn:jboss:module:1.0" name="com.temenos.t24">
2   <resources>
3     <!-- Insert resources here -->
4     <resource-root path=".//T24Exten.jar" />
5     <resource-root path=".//t24lib/AA_ARAccountsData.jar" />
6     <resource-root path=".//t24lib/AA_ARC.jar" />
```

Step 3 – EB.API



The screenshot shows the TEMENOS EB.API configuration interface. The top navigation bar has tabs for 'EB.API' and 'NOFILE.ENQ'. The 'NOFILE.ENQ' tab is selected and highlighted with a yellow box. The main content area displays the following configuration details:

Description	EN FOR NOFILE ENQUIRY
Protection Level	Full
Source Type	Method
Java Method	setIds
Java Class	NoFileEnquiryDemo
Java Package	com.newbank
Curr No	1

Creating the ENQUIRY Record

- A valid application name should be given in the field FILE.NAME which is checked against **STANDARD.SELECTION**
- A NOFILE enquiry does not have a single associated file
- Create a record in **STANDARD.SELECTION** with **NOFILE** prefix for the ID e.g. NOFILE.CUSBALANCE
- This ID should be given in FILE.NAME in the **ENQUIRY** application

NOFILE Enquiry - STANDARD.SELECTION

The screenshot shows the TEMENOS interface for 'STANDARD.SELECTION' with the route 'SS S NOFILE.DEMO.NEW'. The left panel lists various user fields, and the right panel shows their corresponding values in the 'CUSTOMER' and 'CONSOLIDATED.BALANCE' tables.

User Field Name	Value
Usr Field Name.1	CUSTOMER
Usr Type.1	S
Usr Display Fmt.1	10L
Usr Single Mult.1	S
Usr Field Name.2	CONSOLIDATED.BALANCE
Usr Type.2	R
Usr Field No.2.1	NOFILE.ENQ
Usr Display Fmt.2	100L
Usr Single Mult.2	S

Annotations with yellow boxes and arrows:

- A box labeled "Selection Field" points to the first row in the CUSTOMER table.
- A box labeled "Field created by the routine" points to the first row in the CONSOLIDATED.BALANCE table.
- A box labeled "ID of the EB.API" points to the value "NOFILE.ENQ" in the CONSOLIDATED.BALANCE table.

NoFile Enquiry - ENQUIRY

ENQUIRY NOFILE.DEMO

Page Size	4,20
File Name ►	NOFILE.DEMO
Fixed Selection.1	CONSOLIDATED.BAL NE*
Selection Flds.1	CUSTOMER
Required Sel.1	Y
Field Name.1	CUSTOMER
Operation.1 .1	SELECTION CUSTOMER
Column.1	1
Length Mask.1	10L
Single Multi.1	S
Field Name.2	CONSOLIDATED.BAL
Operation.2 .1	0

ENQUIRY NOFILE.DEMO

Field Name.2	CONSOLIDATED.BAL
Operation.2 .1	0
Column.2	2
Conversion.2 .1	F *,2,1
Page Fields	2
Multi Fields	0
Break Fields	0
Process Breaks	0
Total Fields	0
Curr No	13
Inputter.1	202_INPUTTER_OFS_BROWSERTC
Date time.1	07 AUG 19 13:41
Authoriser	202 INPUTTER_OFS_BROWSERTC

NoFile Enquiry – Execution

INPUTTER Last signed on 27 MAR 2019 at 16:31 with 0 attempt(s)

[Help](#) [Tools](#) [Sign Off](#)

ENQ NOFILE.ENQ1

NOFILE.ENQ1 - Google Chrome — □ ×

localhost:9089/BrowserWeb/servlet/BrowserServlet

Favourites

NOFILE.ENQ1 [More Options](#) [Clear Selection](#)

CUSTOMER	equals	* <input type="text" value="100100"/>
NO.SORT.OPTION	equals	
INCLUDE.DL	equals	

GB0010001 : NOFILE.ENQ1

NoFile Enquiry – OUTPUT

AUTHORISER Last signed on 15 APR 2019 at 10:28 with 0 attempt(s)
[Help](#) [Tools](#) [Sign Off](#)

ENQ NOFILE.ENQ1

NOFILE.ENQ1 - Google Chrome – □ ×

localhost:9089/BrowserWeb/servlet/BrowserServlet

Results 1 - 1 of 1     

CUSTOMER CONSOLIDATED.BALANCE	
100100	507,263.22

Favourites 

More Options [Clear Selection](#) [Find](#)

CUSTOMER equals * 100100

GB0010001 : NOFILE.ENQ1

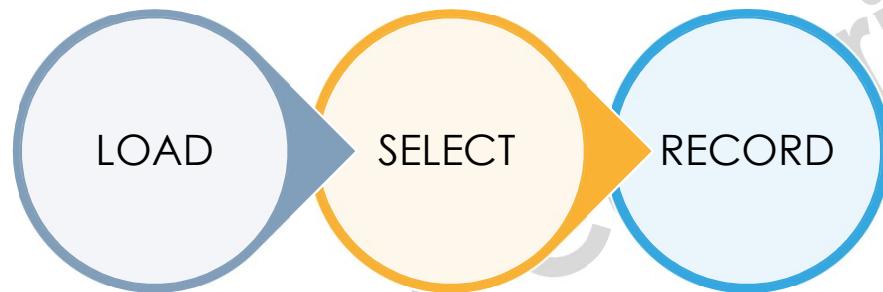
Lesson 4. SERVICE Hooks in Java

T24 Extensibility in Java



Services in T24

- Multithreaded processes
- Can be run independently or as part of COB



- Must implement **com.temenos.t24.api.hook.system.ServiceLifecycle**
- Equivalent of
 - LOAD – initialize()
 - SELECT – getIds() or getTableName()
 - RECORD – inputRecord() or process()

| DEMO- Service

- Simple service that updates the SHORT.NAME of CUSTOMERS in SECTOR 1001 as TEST + existing MNEMONIC
- EXPECTED OUTPUT

CUSTOMER 130131	
Customer Mnemonic	KATRIN
Short Name	EN TEST KATRIN
Full Name	EN Abbey Adalwine
Street Name	EN FRANKFURT

1. Create the class

```
package com.newbank;

import java.util.List;

import com.temenos.api.TBoolean;
import com.temenos.api.TStructure;
import com.temenos.t24.api.complex.eb.servicehook.ServiceData;
import com.temenos.t24.api.hook.system.ServiceLifecycle;
import com.temenos.t24.api.records.customer.CustomerRecord;
import com.temenos.t24.api.system.DataAccess;

/* Service DEMO */

public class MyServiceDemo extends ServiceLifecycle {
    @Override
    public void inputRecord(String id, ServiceData serviceData, String controlItem, TBoolean setZeroAuth,
                           List<String> versionNames, List<String> recordIds, List<TStructure> records) {
        DataAccess da= new DataAccess(this);
        versionNames.add("CUSTOMER");
        recordIds.add(id);
        CustomerRecord cusRec = new CustomerRecord(da.getRecord("CUSTOMER", id));
        cusRec.setShortName("TEST "+cusRec.getMnemonic().toString(), 0);
        records.add(cusRec.toStructure());
        setZeroAuth.set(true);
    }
    @Override
    public List<String> getIds(ServiceData serviceData, List<String> controlList) {
        List<String> recIds = null;
        DataAccess da = new DataAccess(this);
        recIds=da.selectRecords("BNK", "CUSTOMER", "", "WITH MNEMONIC EQ KATRIN K MART KOS ");
        return recIds;
    }
}
```

Export JAR and
restart jBoss

2. Create the EB.API records

```
public class MyServiceDemo extends ServiceLifecycle {  
    @Override  
    public void inputRecord(String id, ServiceData serviceData,  
                           List<String> versionNames, List<String> recordIds,  
                           DataAccess da= new DataAccess(this);  
                           versionNames.add("CUSTOMER");  
                           recordIds.add(id);  
                           CustomerRecord cusRec = new CustomerRecord(da.getRecords());  
                           cusRec.setShortName("TEST "+cusRec.getMnemonic().toString());  
                           records.add(cusRec.toStructure());  
                           setZeroAuth.set(true);  
  
    }  
    @Override  
    public List<String> getIds(ServiceData serviceData, List<String>  
                               versionNames, List<String> recordIds,  
                               List<String> recIds = null;  
                               DataAccess da = new DataAccess(this);  
                               recIds=da.selectRecords("BNK", "CUSTOMER", "", "WITH MN");  
                               return recIds;  
    }  
}
```

EB.API SERVICE.DEMO	
Description	EN SERVICE DEMO
Protection Level	Full
Source Type	Method
Java Method	inputRecord
Java Class	MyServiceDemo
Java Package	com.newbank

EB.API SERVICE.DEMO.SELECT	
Description	EN SERVICE DEMO SELECT
Protection Level	Full
Source Type	Method
Java Method	getIds
Java Class	MyServiceDemo
Java Package	com.newbank

3. Create the PGM.FILE, BATCH & TSA.SERVICE records

The image displays three separate configuration screens side-by-side, each showing fields for creating a record. A large, diagonal watermark reading "Copyright" is overlaid across all three screens.

PGM.FILE SERVICE.DEMO

Type	B
Batch Job.1	@BATCH.JOB.CONTROL
Product	EB

BATCH BNK/SERVICE.DEMO

Process Status	O
Batch Environment	F BACKGROUND
Job Name.1	SERVICE.DEMO
Frequency.1	D
Data.1.1	BULK.OFS

TSA.SERVICE BNK/SERVICE.DEMO

Description.1	SERVICE DEMO
Server Name.1	
Work Profile.1	
Server Status.1	
User *	INPUTTER
Service Control	<input type="radio"/> Stop <input checked="" type="radio"/> Start <input type="radio"/> Auto

4. Run the TSM and TSA

localhost:9089/TAFJEE/Execute

TEMENOS

Execute servlet

You can post command to the ExecQueue by submiting your command in the field below.

i.e. to execute TSM service within the application server post command START.TSM

Command:

- When the service is run, we can see the SHORT.NAME is updated

Customer No	Mnemonic	Name	Account Officer
130131	KATRIN	TEST KATRIN	Implementation
100284	KMART	TEST KMART	Branch Operations Manager
100329	KOS	TEST KOS	Derivatives Front Office Dealer

| Lesson 5. Local APPLICATION

T24 Extensibility in Java



| DEMO Creating Local Application

- We can create a new application using EB.TABLE.DEFINITION. OVERRIDE must be raised if the phone / email is used by existing customer

TABLE.NAME	WALKIN.CUSTOMER
PRODUCT	ST
FILE.TYPE	H
CLASSIFICATION	CUS

@ID, OVERRIDE, AUDIT fields are added automatically

Field Name	Data Type	Remarks
NAME	A	
PHONE		Raise override if this PHONE is available in CUSTOMER
EMAIL		Raise override if this EMAIL is available in CUSTOMER

1. Creating the TABLE – EB.TABLE.DEFINITION

- Create the table using EB.TABLE.DEFINITION
- ETD creates the required artefacts – PGM.FILE, FILE.CONTROL, STANDARD.SELECTION, DATA FILES

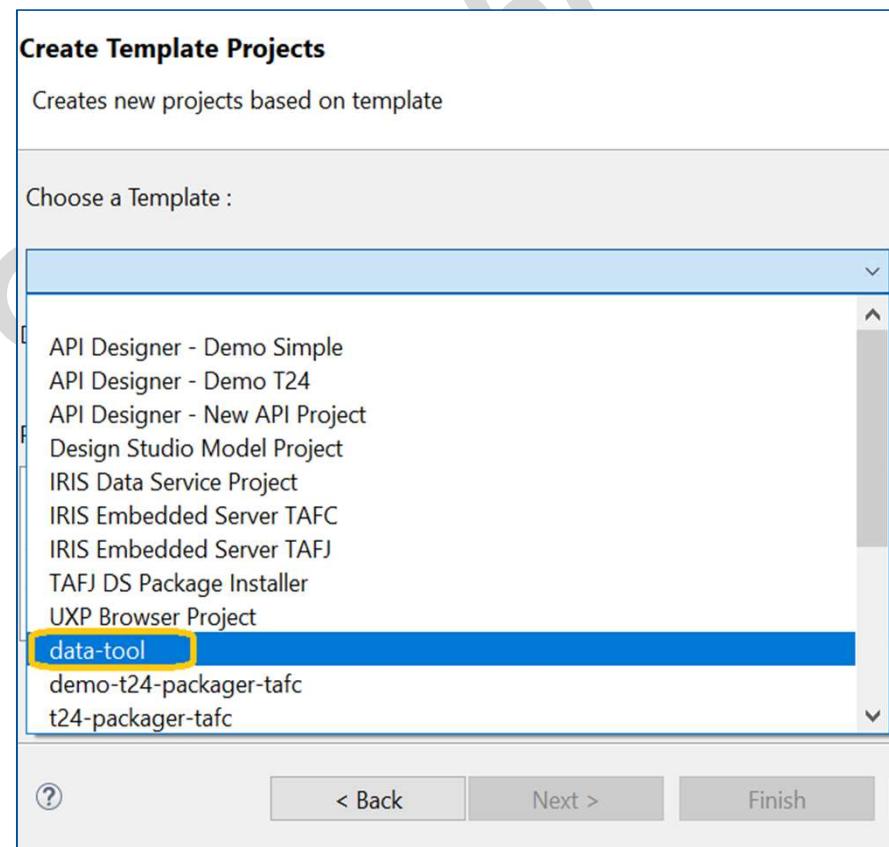
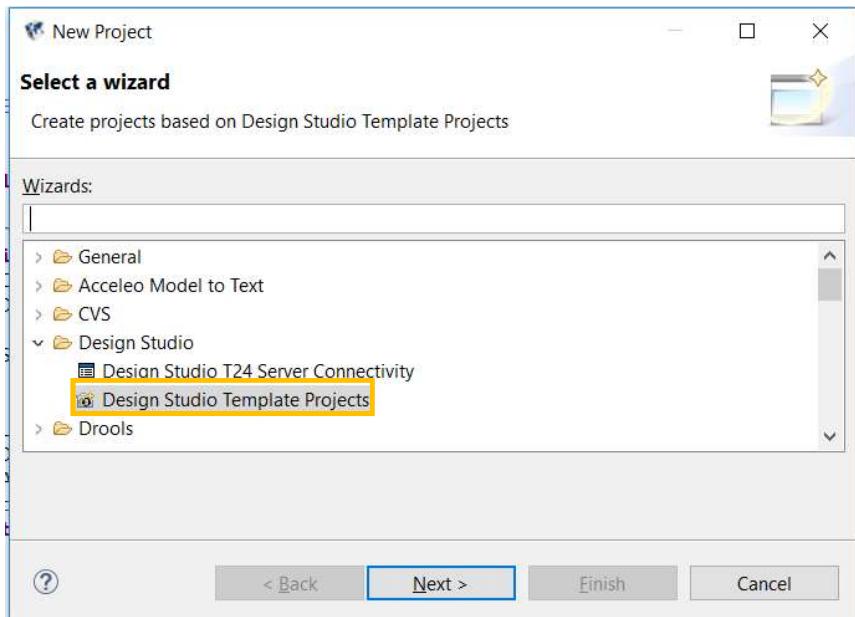
EB.TABLE.DEFINITION WALKIN.CUSTOMER		
Description	EN WALKIN CUSTOMER DEMO	
Product	ST	System Tables
Field Name.1	NAME	
Description.1	NAME.OF.CUST	
Max Char.1	35	
Data Type.1	ALPHANUMERIC	Alpha Numeric Charactes
Field Name.2	PHONE	
Description.2	PHONE	
Max Char.2	20	
Data Type.2	ALPHANUMERIC	Alpha Numeric Charactes
Field Name.3	EMAIL	
Description.3	EMAIL	
Max Char.3	35	
Data Type.3	ALPHANUMERIC	Alpha Numeric Charactes
Prefix	ST.WAL26	
File Type	Cus	
Link To Wfl	No	

| 2. Generate the API for the new Table

- Using Design Studio, we can generate the API for the APPLICATION created using EB.TABLE.DEFINITION
- we need a server project – to establish connection with T24
- We need a models project – to import the application from T24

2a. Create the -DataTool project

- Creating the server project
- File→New→Project



2b. Create the DS-DataTool Project

- Create the DS-DataTool project

The image shows two screenshots illustrating the creation of a DS-DataTool project.

New Design Studio Template Projects Dialog:

- Title:** New Design Studio Template Projects
- Section:** Create Template Projects
- Description:** Creates new projects based on template
- Choose a Template:** data-tool
- Description:** T24 Data Tool (EXPERIMENTAL)
- Parameters:** A table showing configuration parameters:

Name	Value	Description
project-name	BRP	Name of the project
product-name	XX	Product's name
component-name	Sample	Component's name
host	localhost	Host
ws-port	9089	Port
t24username	INPUTT	User name
t24password	123456	Password
t24username-authorizer	AUTHOR	Authoriser's user name
t24password-authorizer	123456	Authoriser's password
tafjHome	C:\Temenos\TAFJ	TAFJ home (tafj.home in tafj.prope
insertDir	C:\Temenos\t24home\default\RG.BP	Insert BASIC sources directory (tem
libDir	C:\Temenos\jboss\modules\com\temenos\t24\main	Precompile classes directory (tem
- Buttons:** ? (Help), < Back, Next >, Finish (highlighted in blue), Cancel

Package Explorer View:

- Project:** BRP-data-code
- Nodes:** BRP-models (selected), src, Applications, domain, JRE System Library [jdk], Documentation, Models, pom.xml, BRP-models-gen, BRP-packager, data-tool, t24-server (selected), config, server.properties

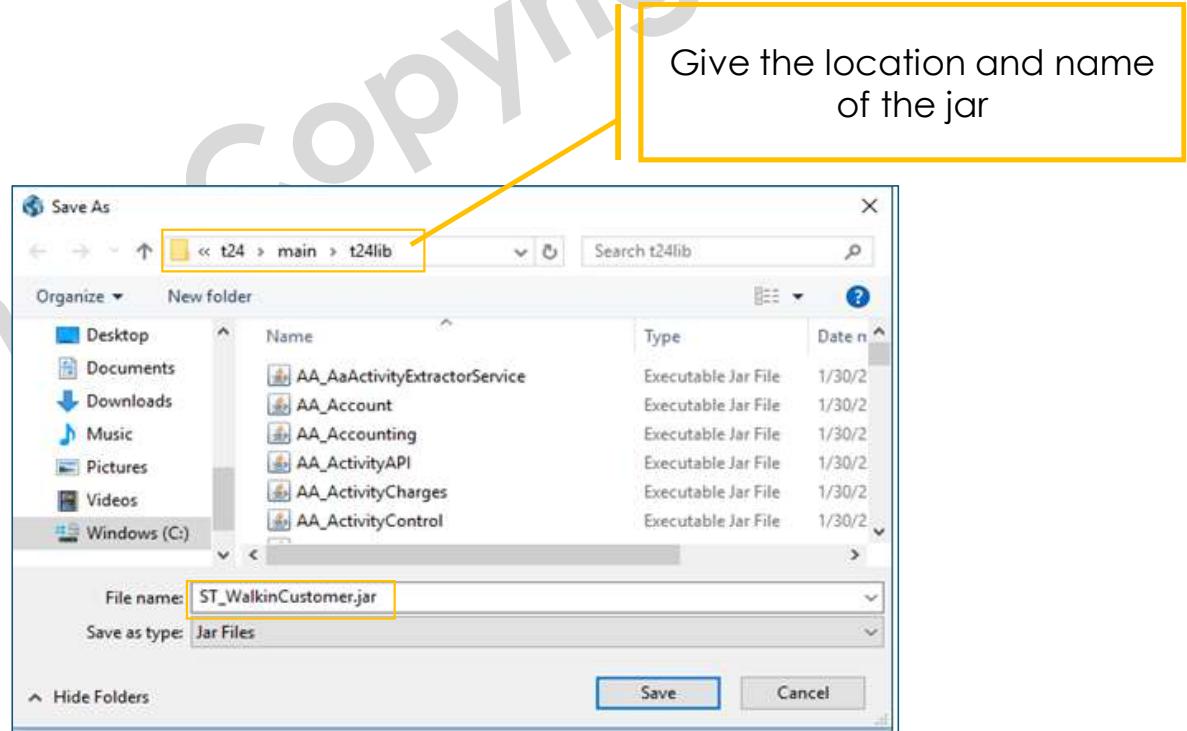
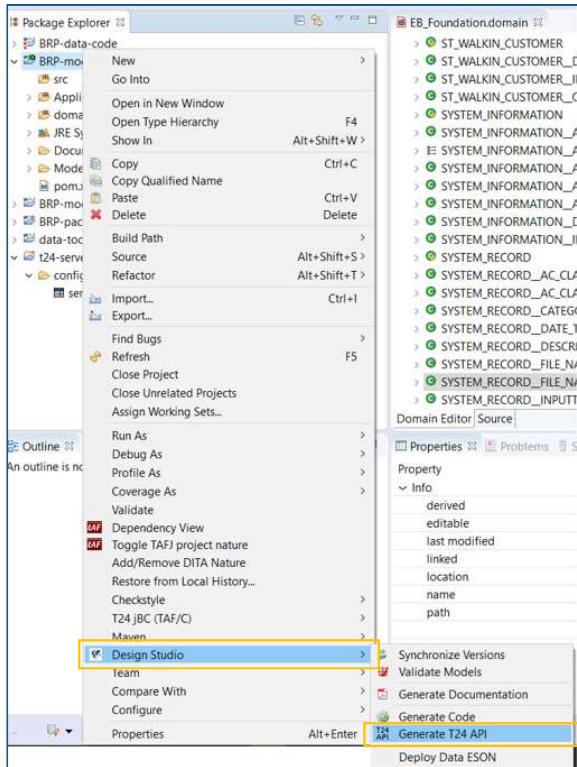
2c. Import the Local Application

The screenshot illustrates the process of importing a local application into a T24 environment. It shows four windows:

- Package Explorer:** Shows a tree view of projects: BRP-data-code, BRP-models, and I24-server. The BRP-models project is selected.
- Import Context Menu:** A context menu is open over the BRP-models project, with the "Import..." option highlighted.
- Select Application Wizard:** This window lists "Import T24 Applications By Name" as the selected import wizard. It shows the "app" application selected under "Application Selection".
- T24 Application Name Selection Wizard:** This window displays the "T24 Application Name Selection" step, which imports an application named "ST.WALKIN.CUSTOMER".
- T24 Applications Import Wizard:** This window is titled "Select the destination of imported Applic". It shows the "Select the destination folder:" section, where the "BRP-models" project is selected. It also includes sections for "Projects visibility" (radio button for "Only model projects") and "Folder Structure" (set to "FLAT").
- EB_Foundation.domain:** This window shows the contents of the imported application. It lists several components:
 - ST_WALKIN_CUSTOMER
 - ST_WALKIN_CUSTOMER_DATE_TIME
 - ST_WALKIN_CUSTOMER_INPUTTER
 - ST_WALKIN_CUSTOMER_OVERRIDE
 - SYSTEM_INFORMATION
 - SYSTEM_INFORMATION_ANALYS_ROUTINE
 - SYSTEM_INFORMATION_ANALYS_ROUTINE_ANALYS_ROUTINE
 - SYSTEM_INFORMATION_ANALYS_ROUTINE_ANALYS_VALUE
 - SYSTEM_INFORMATION_APPLICATION

2d. Generate the API for the new Table

- Toggle the DS project as a TAFJ project
- Right click on project → Design Studio → Generate T24 API



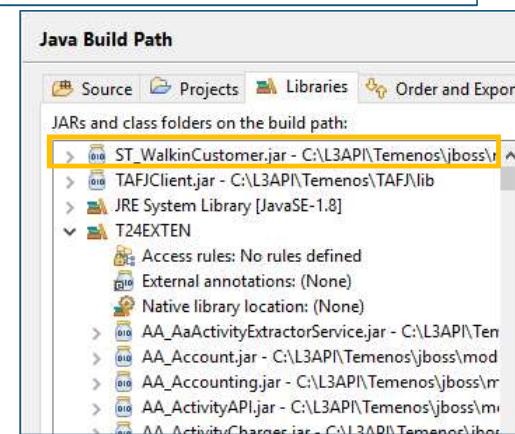
2e. Update module.xml and Build path

C:\Temenos\jboss\modules\com\temenos\t24\main\t24lib\ST_WalkinCustomer.jar				
Name	Size	Packed Size	Modified	
WalkinCustomerRecord.class	16 803	6 058	2019-02-22 13:34	
WalkinCustomerRecord.java	6 397	955	2019-02-22 13:34	
WalkinCustomerTable.class	4 095	1 517	2019-02-22 13:34	
WalkinCustomerTable.java	2 082	517	2019-02-22 13:34	

- Update module.xml and restart jBoss

```
<module xmlns="urn:jboss:module:1.0" name="com.temenos.t24">
  <resources>
    <!-- Insert resources here -->
    <resource-root path="../T24Exten.jar" />
    <resource-root path="../t24lib/ST_WalkinCustomer.jar" />
    <resource-root path="../t24lib/AA_ARAccountsData.jar" />
    <resource-root path="../t24lib/AA_ARC.jar" />
```

- Add jar to the java Build path in DS



3. Create the class

```
*WalkInCustomerOverride.java ✘
```

```
package com.newbank;
import java.util.List;
import com.temenos.api.TBoolean;
import com.temenos.api.TString;
import com.temenos.api.TStructure;
import com.temenos.api.TValidationResponse;
import com.temenos.t24.api.hook.system.RecordLifecycle;
import com.temenos.t24.api.system.DataAccess;
import com.temenos.t24.api.tables.walkincustomer.WalkinCustomerRecord;

public TValidationResponse validateRecord(String application, String recordId, TStructure record,
    TStructure lastLiveRecord) {
    // validate phone and raise override
    DataAccess da = new DataAccess(this);
    String cusIds = "";
    List<String> recIds;
    int len;
    WalkinCustomerRecord WCR = new WalkinCustomerRecord(record);
    String phone = WCR.getPhone().toString();
    if (!phone.isEmpty()) {
        recIds = da.selectRecords("BNK", "CUSTOMER", "", "WITH PHONE.1 EQ '" + WCR.getPhone().toString() + "'");
        len = recIds.size();
        if (len > 0) {
            for (int i = 0; i < len; i++)
                cusIds = cusIds + " " + recIds.get(i);
            WCR.getPhone().setOverride("This phone number is used by CUSTOMER " + cusIds);
        }
    }
    // VALIDATE email and raise override
    String email = WCR.getEmail().toString();
    if (!email.isEmpty()) {
        recIds = da.selectRecords("BNK", "CUSTOMER", "", "WITH EMAIL.1 EQ '" + email + "'");
        len = recIds.size();
        cusIds = "";
        if (len > 0) {
            for (int i = 0; i < len; i++)
                cusIds = cusIds + " " + recIds.get(i);
            WCR.getEmail().setOverride("This email is used by CUSTOMER " + cusIds);
        }
    }
    return WCR.getValidationResponse();
}
```

4. Create the EB.API, VERSION

EB.API WALKIN.CUSTOMER.OVERRIDE	
Description	EN WALKIN CUST
Protection Level	Full
Source Type	Method
Java Method	validateRecord
Java Class	WalkinCustomerOverride
Java Package	com.newbank

Version ST.WALKIN.CUSTOMER,DEMO	
Records Per Page	1
Fields Per Line	Multi
Language Code.1	1
Field No.1	NAME
Text Char Max.1	35
Pre Syndication ID.1 .1	Name
Enrichm Char.1	35
Table Line.1	0
Field No.2	EMAIL
Text Char Max.2	35
Pre Syndication ID.2 .1	Email
Enrichm Char.2	35
Table Line.2	1
Field No.3	PHONE
Text Char Max.3	10
Pre Syndication ID.3 .1	Phone
Enrichm Char.3	10
Table Line.3	2
Input Routine.1	WALKIN.CUSTOMER.OVERRIDE

5. Launch the VERSION

The screenshot shows a comparison of two customer records. The top record is for 'CUSTOMER 100100' and the bottom record is for 'St Walkin Customer Demo 829'. Both records show identical contact information: Name (HARRY THOMAS CRISP), Email (harrycrisp@gmail.com), and Phone (206 553-6947). A yellow box highlights the phone number in both records. A modal dialog box is overlaid on the screen, stating: '= This phone number is used by CUSTOMER 100100' and 'This email is used by CUSTOMER 100100'. It includes 'CANCEL' and 'ACCEPT' buttons.

CUSTOMER	100100
Phone No.1	206 553-6947
Mobile Phone Numbers.1	206 553-6947
Email.1	harrycrisp@gmail.com

St Walkin Customer Demo	829
Name	HARRY THOMAS CRISP
Email	harrycrisp@gmail.com
Phone	206 553-6947
Override.1	This phone number is used by CUSTOMER 100100
Override.2	This email is used by CUSTOMER 100100

= This phone number is used by
CUSTOMER 100100
This email is used by CUSTOMER 100100

CANCEL ACCEPT

Using EB.TABLE.PROCEDURES

The screenshot illustrates the configuration of EB API, EB TABLE PROCEDURES, and PGM FILES.

EB.API * (Top Left):

Key	Description	Protection Level	Source Type	Java Method
WALKIN.CUSTOMER.DEFAULT	WALKIN CUST	FULL	METHOD	defaultFieldValues
WALKIN.CUSTOMER.OVERRIDE	WALKIN CUST	FULL	METHOD	validateRecord

EB.TABLE.PROCEDURES (Bottom Left):

	ST.WALKIN.CUSTOMER
Description *	EN WALKIN CUSTOMER
Initialise Proc.1	
Check Id Proc.1	
Check Rec Proc.1	WALKIN.CUSTOMER.DEFAULT WALKIN CUST ▾
Prs Fld Proc.1	
Check Fld Proc.1	
Crossval Proc.1	WALKIN.CUSTOMER.OVERRIDE WALKIN CUST ▾

PGM.FILE (Top Right):

WALKIN.CUSTOMER.DEFAULT	
Type	S
Product	EB

PGM.FILE (Bottom Right):

WALKIN.CUSTOMER.OVERRIDES	
Type	S
Product	EB

Using EB.TABLE.PROCEDURES

```
public class WalkInCustomerOverride extends RecordLifecycle {  
  
    public String checkId(String arg0) {..  
  
        @Override  
        public void defaultFieldValues(String application, String recordId, TStruct record) {  
            WalkinCustomerRecord WCR = new WalkinCustomerRecord(record);  
            if (WCR.getEmail().toString().isEmpty()) {  
                WCR.setEmail("abc@abc.com");  
                record.set(WCR.toStructure());  
            }  
        }  
    }  
}
```

St Walkin Customer Demo 456

Name

Email

Phone

| Local Table

Lesson - 6



| DEMO Creating Local Table

- Create a local field called CUS.HOBBIESTS
- Attach field to CUSTOMER
- The local field gets its values from a LOCAL APPLICATION
- Use version to input hobbies for authorized CUSTOMER
- Minimum of 3 hobbies must be input

1. Creating the Local Application

- Create local application CUS.HOBBIES and input few records

EB.TABLE.DEFINITION CUS.HOBBIES	
Description	EN CUSTOMER HOBBIES
Product	ST
Field Name.1	DESCRIPTION
Description.1	HOBBY
Max Char.1	30
Data Type.1	ALPHANUMERIC
Prefix	ST.CUS86
File Type	Cus

List of Hobbies	
Deal Number	Description
DANCING	Dancing
GARDENING	Gardening
MOVIES	Watching Movies
ORIGAMI	Origami
READING	Reading Books

2. Create LOCAL.TABLE called CUS.HOBBIES

The screenshot shows the LocalFieldsDefinition.domain editor interface. In the left pane, under 'LocalFieldsDefinition.domain' > 'CUS.HOBBIES', there is a list of fields:

- [0-1] COLLECTION.RATING (LocalFieldsEnumeration:LocalField_COLLECTION_RATING)
- [0-1] CREATION.DATE (T24BusinessTypes:A)
- [0-1] CREDITOR.ID (T24BusinessTypes:A)
- [0-1] CU.EFF.DATE (T24BusinessTypes:D)
- [0-1] CUS.HOBBIES (EB_Foundation:ST_CUS_HOBBIES) **(highlighted)**
- [0-1] CUSTOMER.ID (T24BusinessTypes:NUMERIC)

In the main pane, the 'Association' section is displayed. It includes fields for 'Name' (CUS.HOBBIES), 'Type' (EB_Foundation:ST_CUS_HOBBIES), 'Modifiers' (Primary Key, Business Key, Required), 'Containment' (By Value, By Reference **(highlighted)**), and 'Multiplicity' (One, Many). The 'Type' field is highlighted with a yellow box.

3. Add the LOCAL.TABLE to CUSTOMER

The screenshot shows the domain editor for 'X_CUSTOMER.domain'. The tree view on the left lists various entities and associations. The 'CUS_HOBIES' association is selected and highlighted with a yellow box. The 'Association' tab is active at the bottom, showing properties for the 'CUS_HOBIES' association:

- Name: CUS_HOBIES
- Type: EB_Foundation:ST_CUS_HOBIES
- Modifiers:
 - Primary Key
 - Business Key
 - Required
- Containment:
 - By Value
 - By Reference
- Multiplicity:
 - One
 - Many

4. Create the class

```
package com.newbank;

import java.util.List;
import com.temenos.api.LocalRefClass;
import com.temenos.api.TField;
import com.temenos.api.TStructure;
import com.temenos.api.TValidationResponse;
import com.temenos.t24.api.hook.system.RecordLifecycle;
import com.temenos.t24.api.records.customer.CustomerRecord;
import com.temenos.t24.api.system.DataAccess;

/** VERSION CUSTOMER,HOBBY is used to add a minimum of 3 hobbies for authorised customers */
public class CustomerHobbies extends RecordLifecycle {

    @Override
    public TValidationResponse validateRecord(String application, String recordId, TStructure record,
                                              TStructure lastLiveRecord) {
        CustomerRecord cusRec = new CustomerRecord(record);
        LocalRefClass lrf = cusRec.getLocalRef("CUS.HOBBIES");
        List<TField> hobbies = lrf.get();
        TField dummy = new TField("");
        if (hobbies.size() <= 2) {
            lrf.set(dummy, 0); // to overcome array index out of bounds when no hobby is entered
            lrf.get(0).setError("Please select minimum of 3 hobbies");
        }
        return cusRec.getValidationResponse();
    }

    @Override
    public String checkId(String idNew) {
        DataAccess da = new DataAccess(this);
        CustomerRecord cusRec = null;
        try {
            cusRec = new CustomerRecord(da.getRecord("CUSTOMER", idNew));
        } catch (Exception e) {
            throw new Error("This VERSION can be used to edit Authorised customers only or Invalid Customer ID");
        }
        return idNew;
    }
}
```

Create class , Export JAR and restart jBoss

5. Create the EB.API records

EB.API *	CUS.H				
	Key	Description	Protection Level	Source Type	Java Method
	CUS.HOBBIES	CUS HOBBIES	FULL	METHOD	validateRecord
	CUS.HOBBIES.ID	CUS HOBBIES ID	FULL	METHOD	checkId

6. Create/Modify version

- Create the version CUSTOMER,HOBBY in DS and generate code. Modify the VERSION and add the hooks

Version CUSTOMER,HOBBY		
Records Per Page	1	
Fields Per Line	Multi	
Language Code.1	1	
Field No.1	MNEMONIC	
Text Char Max.1	10	
Enrichm Char.1	10	
Table Line.1	0	
Field No.2	CUS.HOBBIES-1	
Text Char Max.2	50	
Enrichm Char.2	50	
Table Line.2	1	
No Of Auth	1	
Multi Possible	Y	
Local Ref Field	LOCAL.REF	
Input Routine.1	CUS.HOBBIES	CUS HOBBIES
Report Locks	Yes	
Id Rtn.1	CUS.HOBBIES.ID	CUS HOBBIES ID

Launching the VERSION

The screenshot shows the TEMENOS Customer Hobby interface for Customer ID 100283. The Mnemonic field contains 'DELL'. Under 'Cus Hobbies', three entries are listed: 'GARDENING' (Gardening), 'DANCING' (Dancing), and 'READING' (Reading Books). Each hobby entry has a dropdown menu and a '+' button.

The screenshot shows the TEMENOS Customer Hobby interface for Customer ID 919919. A red message at the top states: "Runtime Exception Error This VERSION can be used to edit Authorised customers only or Invalid Customer ID".

The screenshot shows the TEMENOS Customer Hobby interface for Customer ID 100283. The Mnemonic field contains 'DELL'. Under 'Cus Hobbies', only 'GARDENING' is selected. A validation message below the hobbies says: "Please select minimum of 3 hobbies". A red error box on the right side lists: "Please resolve the errors below to proceed:" and "Cus Hobbies.1: Please select minimum of 3 hobbies".

Course Summary

- We have seen
 - Need for T24 Extensibility
 - T types
 - How to attach VERSION hooks
 - Attaching Enquiry hooks
 - Creating and executing a multi threaded SERVICE
 - Creating a local table using EB.TABLE.DEFINITION and customizing the application
 - How to debug the Java code



TEMENOS

Learning Community

thank.you

tlc.temenos.com