IBM

WebSphere®

IBM WebSphere Multichannel Bank Transformation Toolkit
Version 8.0

Functional Development Guide

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” section.

This edition applies to Version 8, Release 0, Modification 0, of IBM WebSphere Multichannel Bank Transformation Toolkit (5724-H82) and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You can send to the following address:

IBM China Software Development Lab
Bank Transformation Toolkit Product
Diamond Building, ZhongGuanCun Software Park, Dongbeiwang West Road No.8,
ShangDi, Haidian District, Beijing 100193 P. R. China

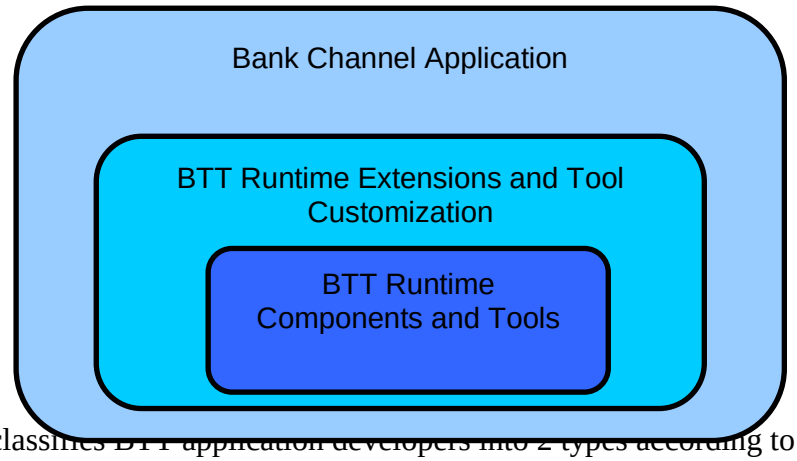
Include the title and order number of this book, and the page number or topic related to your comment.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

1. Overview

As a multi-channel application development toolkit, BTT implements a set of common and reusable components for channel application development. Furthermore, BTT provides tools and facilities for developers to implement channel applications more efficiently and easily. At the same time, for a channel application, there are some project specific reusable components and facilities that need to be implemented. BTT provides capability for application developers to implement project level reusable components and integrate them with BTT framework. The figure below shows the relationship of BTT, BTT extensions and channel application.



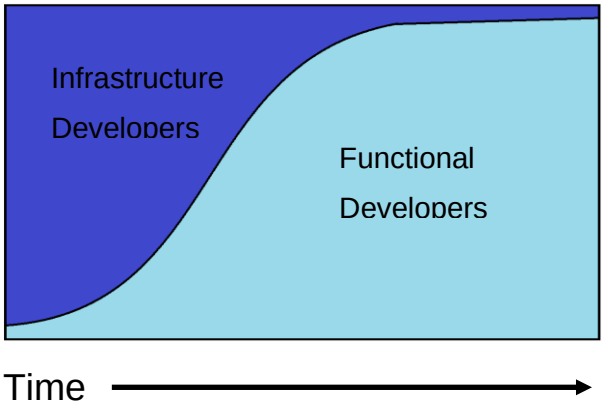
BTT classifies BTT application developers into 2 types according to their roles.

- **Infrastructure developer:** Infrastructure developers supposed to have deep knowledge on BTT and related technologies such as OOP and Java EE. As Infrastructure developers, they are responsible for designing and implementing project specific common components and functions.
- **Functional developer:** Functional developers supposed to have some knowledge on BTT and related technologies. As Functional developers, they are responsible for implementing specific transactions including user interface, operation logic and transaction flow. Development productivity is one of major consideration for Functional developers.

This development mode can be used to leverage reusable components and improve productive of application development.

So for a topical BTT project, it will have two development phases

- **Infrastructure development phase:** this phase requires Infrastructure developers to design and implement project specific reusable components as BTT extensions, and customize BTT tools for these extensions if needed.
- **Incremental development phase:** this phase requires Functional developers to use tools provided by BTT and infrastructure phase extensions to develop each transaction. The figure below demonstrates skill distribution in infrastructure development and incremental development phases.



In this document, we will go through the installation and configuration process of the development environment for functional developers as well as through a detailed description of the tools provided by BTT to be used by the functional developers to implement the required transactions.

2. Installing and configuring the development environment

Use this section to set up a development workstation for a functional developer so that they can develop applications that are based on IBM WebSphere Multichannel Bank Transformation Toolkit V7.1.

WebSphere® Multichannel Bank Transformation Toolkit provides a number of tools that support the development of applications. All the tools are plug-ins of Rational® Application Server.

Application Wizard
The Application Wizard is a plug-in tool that enables you to create WebSphere Multichannel Bank Transformation Toolkit applications rapidly.

Transaction Editor
The Transaction Editor provides a graphical editor for creating and developing WebSphere Multichannel Bank Transformation Toolkit transactions. It also generates the XML configuration files of the transactions and the skeleton code of the transaction classes.

XUI Editor
The XUI Editor is a WYSWYG tool that creates WebSphere Multichannel Bank Transformation Toolkit XML-based UI files.

Web services tooling
The Web services tooling allows you to invoke Web services without requiring you to write any Java code. You can select and retrieve a WSDL file, select the Web service operation described in the WSDL file that you want to invoke, and create data mappings between the data model and the Web services data model so that messages can be exchanged between the two.

Formatter simulator
The Formatter simulator simulates how a string is parsed into a specific data item.

Functional Trace Helper
The Functional Trace is intended for functional developers, and it records execution and error information when code is being tested during the development phase.

In order to proceed with the installation and configuration, follow first the procedure described in the Setting up a Bank Transformation Toolkit development environment section. The result is an installation of WebSphere Multichannel Bank Transformation Toolkit in IBM Rational Application Developer where you can develop solutions that are based on WebSphere Multichannel Bank Transformation Toolkit.

In addition to the tools installed by default, you may want to configure your development environment in order to activate the WebSphere Multichannel Bank Transformation multi-project capabilities. This configuration is a technical developer task. To understand how the multi-project environment may affect the functional developer activities, refer to section Understanding and working in a multi-project environment at the end of the document.

3. The BTT Perspective

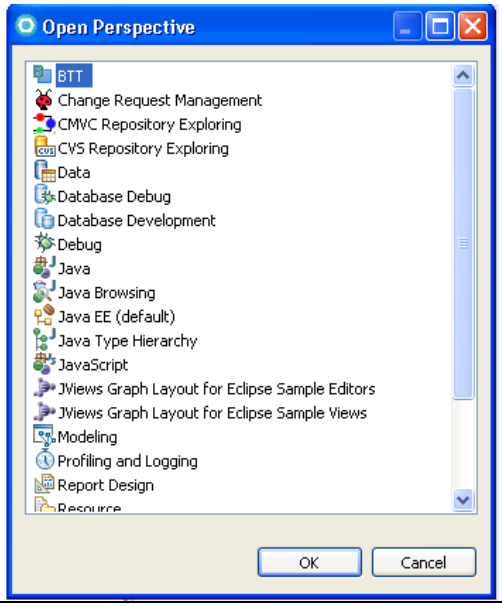
Because BTT is based on RAD, if the Functional developer has access to may accidentally modify generated or infrastructure code or rearrange elements and break the project. In order to avoid this, the BTT tooling provides the functional developers with a specific BTT perspective that hides non-BTT views and editors and hides infrastructure and generated files from the workspace and project explorer outline view. The BTT Perspective shows BTT-specific model artifacts and rearranges them with a more functional structure (grouped by common elements and projects).

Development productivity is one of major consideration for Functional developers. In order to simplify and facilitate the work of the Functional developers, that only have some knowledge of BTT and related technologies, a specific Rational Application Developer perspective, the **BTT Perspective**, is provided as part of the WebSphere® Multichannel Bank Transformation Toolkit development environment. The **BTT Perspective**:

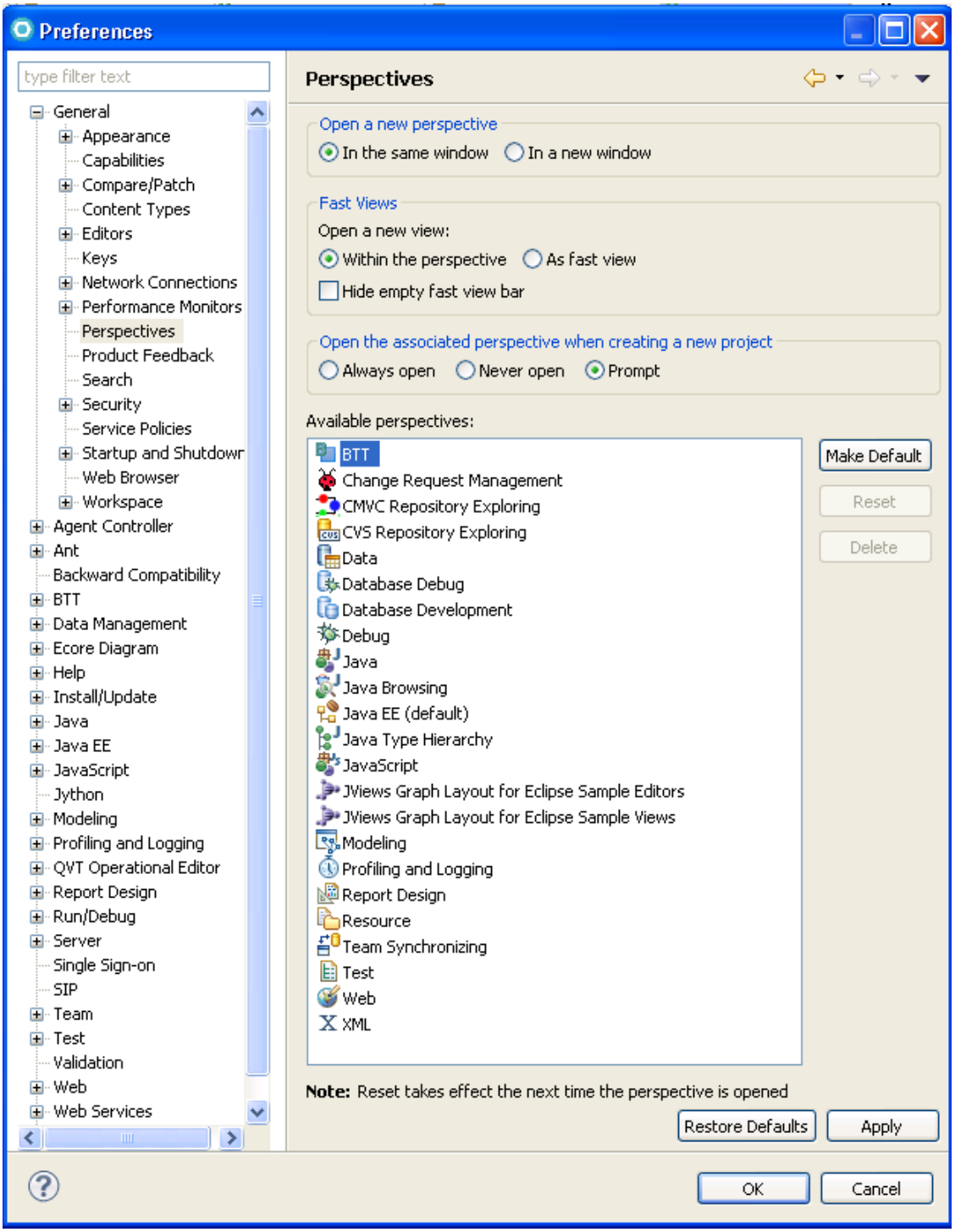
- Hides non-BTT views and editors
- Hides infrastructure and generated files from Workspace and Project Explorer outline view: shows only BTT-specific model artifacts and rearranges them with a more functional structure
- Supports massive build tasks

Functional developers can then focus on creating the business logic and the UI with WebSphere Multichannel Bank Transformation Toolkit tooling. WebSphere Multichannel Bank Transformation Toolkit generates the technical infrastructure code of WebSphere Multichannel Bank Transformation Toolkit framework and transaction definition and classes. In many cases, a function developer might not even be required to write any code. We call it zero code development of massive transactions.

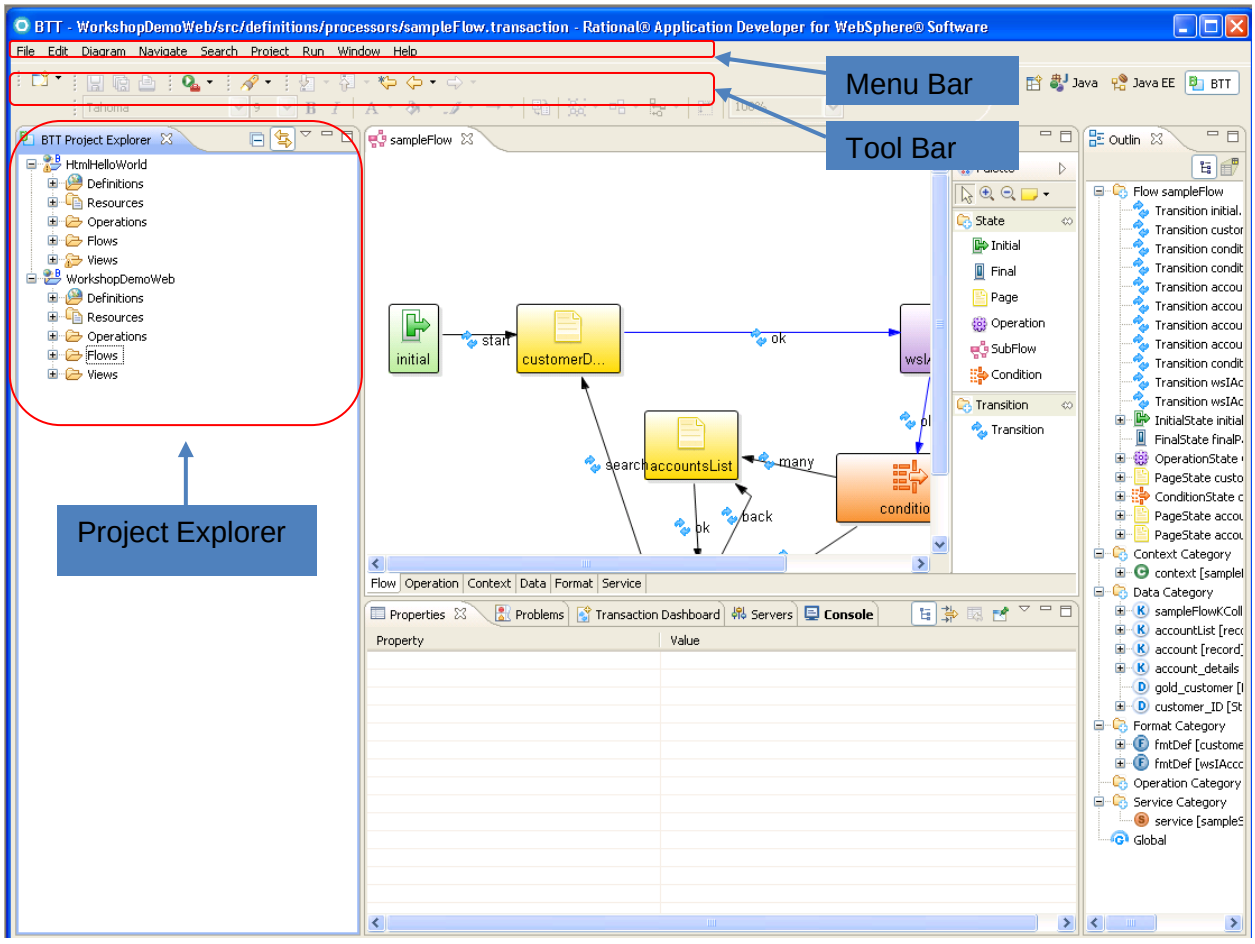
To open the **BTT Perspective** in Rational Application Developer click Window > Open Perspective > Other > BTT.



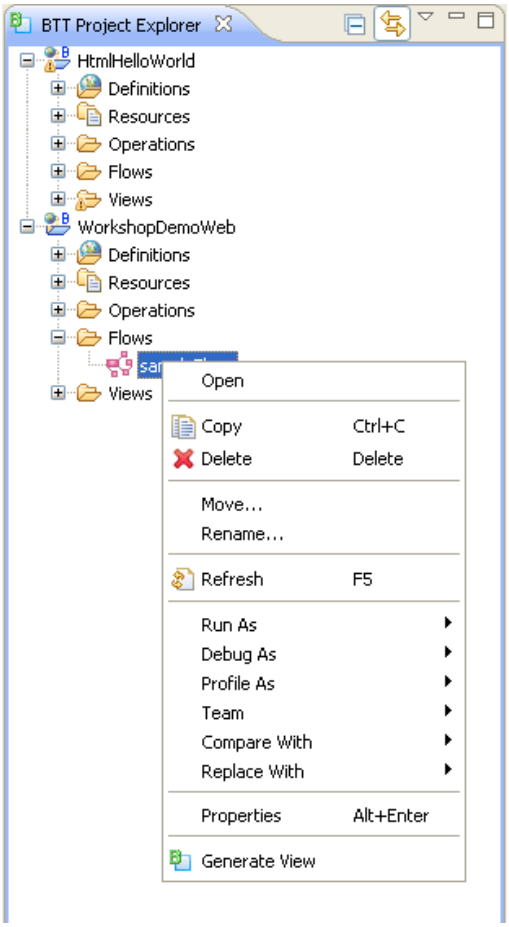
RAD can be configured in order to open on the BTT Perspective by default and hide from the environment those perspectives that we don't want our functional developers to have access to. Use the Windows > Preferences menu option to configure the Perspectives, select the default one and delete the ones that are not needed.



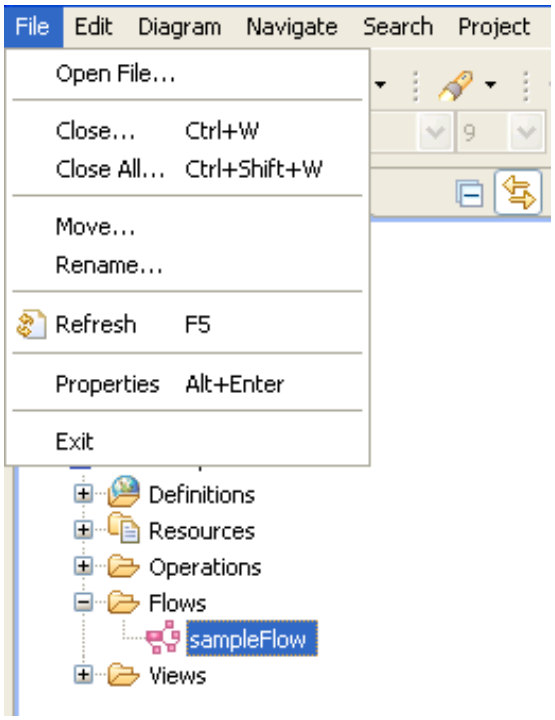
The BTT Perspective provides the functional developers with a specific configuration of several components that are commonly part of other RAD perspectives: the Project Explorer, the Menu Bar and the Tool Bar:



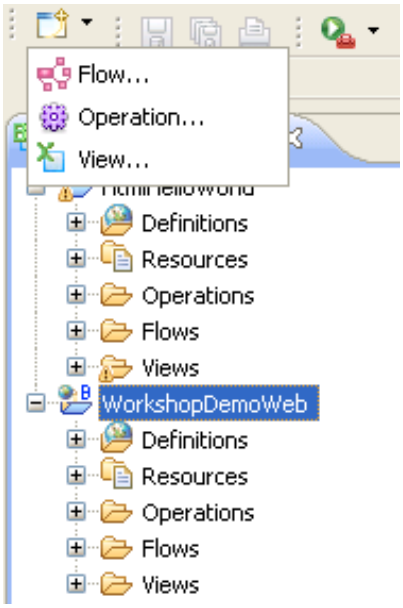
- the **Project Explorer** presents the list of BTT related elements that can be managed by a functional developer and the actions that can be done with them depending on the type of element or group of elements selected (**Context Menu** content)



- the **File** option in the **Menu Bar** has been configured to present the valid options for the selected BTT element or group of elements in the Project Explorer panel.



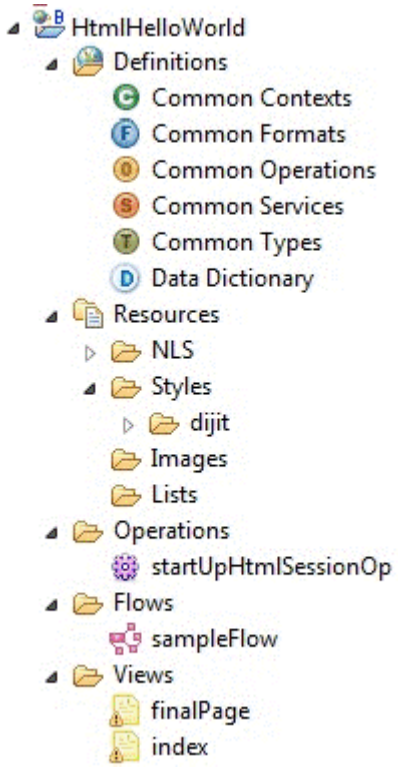
- the **New** option in the **Tool Bar**  does only show, when dropping down the options by clicking on the icon arrow, the element that can be created as a child of the currently selected BTT artifact in the Project Explorer panel.



All actions are consistent independently from where you are launching them. The following section describes in detail the content of the BTT Project Explorer and the actions associated to the different types of elements.

3.1 The BTT Project Explorer

One of the components that have been configured for the BTT Perspective is the Project Explorer. The BTT Project Explorer contains only BTT projects. The BTT projects are the root elements in the BTT Project Explorer tree structure, are labeled with the project Id and are ordered in alphabetical order. For each BTT project, the following information is available:

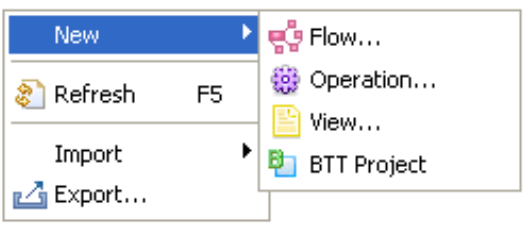


This section will describe contents of the project explorer folders and the actions that can be done when selecting any of the elements that are shown or selecting the blank area.

Note: in some development environments there may be uncontrolled contributions to the project explorer context menu. In this section only the actions that are allowed and make sense on a BTT development environment for a selected element are described, but some others may also appear when right-clicking on the element.

3.1.1 Project Explorer blank area

If you right-click on the Project Explorer blank area (if you want to remove a selection, press Ctrl+left click on the selected element), you are able to create the following BTT elements: flow, operation, view and BTT project.



The **Operation…** option launches the Transaction editor in the Operation page (Operation editor).

The **Flow…** option launches the Transaction editor in the Flow page (Flow editor).

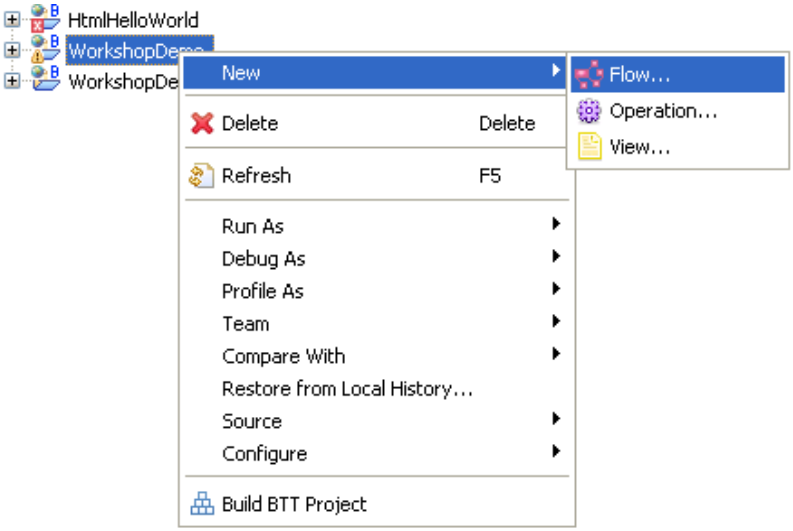
The **View…** option launches the XUI editor.

The **BTT Project** option opens the XUI project create wizard.

The same options are available with no selection in the Project Explorer if you go to the Menu Bar and select the File > New option or you drop-down the list of options in the icon  in the Tool Bar.

3.1.2 Project folder

If you right-click on a specific project folder label, there are two possible BTT specific actions: **New** and **Build BTT Project**.



When selecting the **New** option, the tooling allows to create either a flow, an operation or a view.

The same occurs if having a specific BTT Project folder selected, you go to the Menu Bar and select the File > New option or you drop-down the list of options in the icon  in the Tool Bar.

The **Operation…** option launches the Transaction editor in the Operation page (Operation editor).

The **Flow…** option launches the Transaction editor in the Flow page (Flow editor).

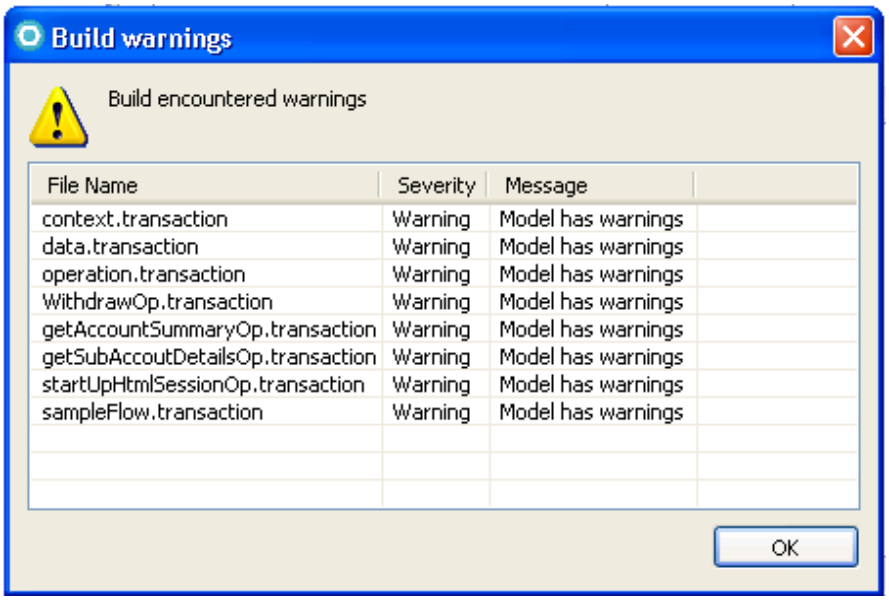
The **View…** option launches the XUI editor.

The new BTT element is created in the folder inside the project that corresponds to its type: Flows, Operation or Views folder.

When selecting **Build BTT Project,** all project components all BTT artifacts that are needed to deploy the application are generated. Normally these artifacts are automatically generated as part of the different elements creation and editing processes when this is done using the development tools. But under certain circumstances you may need to run the Build BTT Project:

- If you are copying some model files directly into the development environment or download any definition from a code repository, you may need to run the Build BTT Project action to ensure that all require artifacts are created before the application deployment.
- When deleting, moving or renaming a BTT element, not all required changes are automatically applied to other BTT elements in your project or workspace. In order to detect possible inconsistencies, you have to run the **Build BTT Project** action and correct the errors found.

After the **Build BTT Project** execution, a report with a list of warning and errors found in the generated artifacts is prompted to the developer, so you can proceed to correct them using the tooling editors.



You can also go to the Problems tab to check for any other possible generation error.

3.1.3 Definitions folder

Before starting the development of the application transactions, there is an environment configuration task that should be run by the technical developers. This task includes the configuration of the different core components (global settings) and the definition of the basic components that will be shared during the transaction development process (context structure, context common data, new data types, reusable operations and formatters, common services, …). All this information is stored in the file btt.xml that is the configuration file for all the BTT components within a specific BTT Project, but this file can only be accessed and edited by the technical developers using the Deployment Descriptor Editor, a graphical editor to enable you to build the btt.xml file rapidly. For a

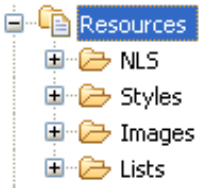
functional developer, all these common and reusable components defined in the BTT xml definition files are available in the **Definitions** folder of each BTT Project in the BTT Project Explorer. From there you have access to the defined common contexts, formats, operations, services and types, as well as to the data dictionary, a list of names of the data elements being used by the predefined set of common BTT components. This list of components can then be complemented with any other BTT component that you may need for your transactions development and are added to the Definitions folder at project level. A functional developer is normally expected to add only new data and formatters to these common definitions.

Right-clicking on any of the children folders of the **Definitions** folder and selecting the option **Open**, the Transaction editor opens showing the specific creation wizard. The same occurs when double-clicking on any of the folders. This is the only action that is allowed for these folders. Follow the links for a details description of the definition of contexts, formats, operations, services, types and data using the **Transaction editor**.

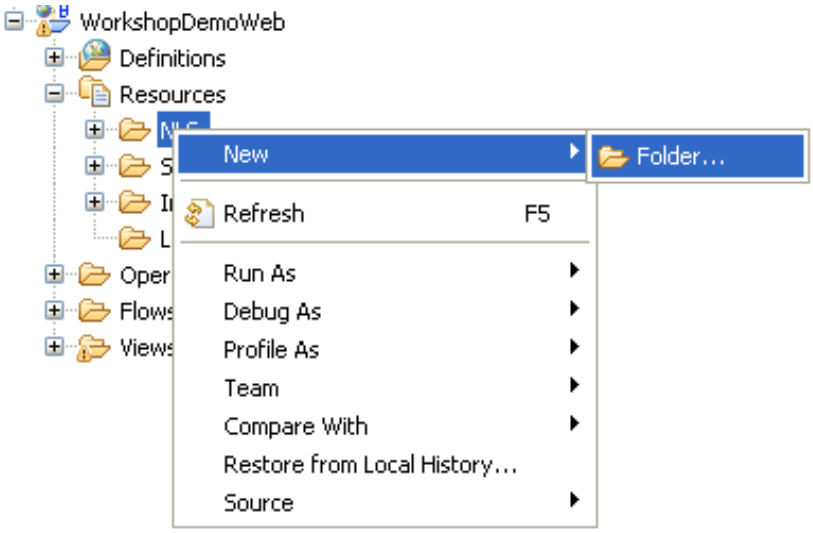
No actions are associated to this folder neither in the Menu Bar nor in the Tool Bar.

3.1.4 Resources folder

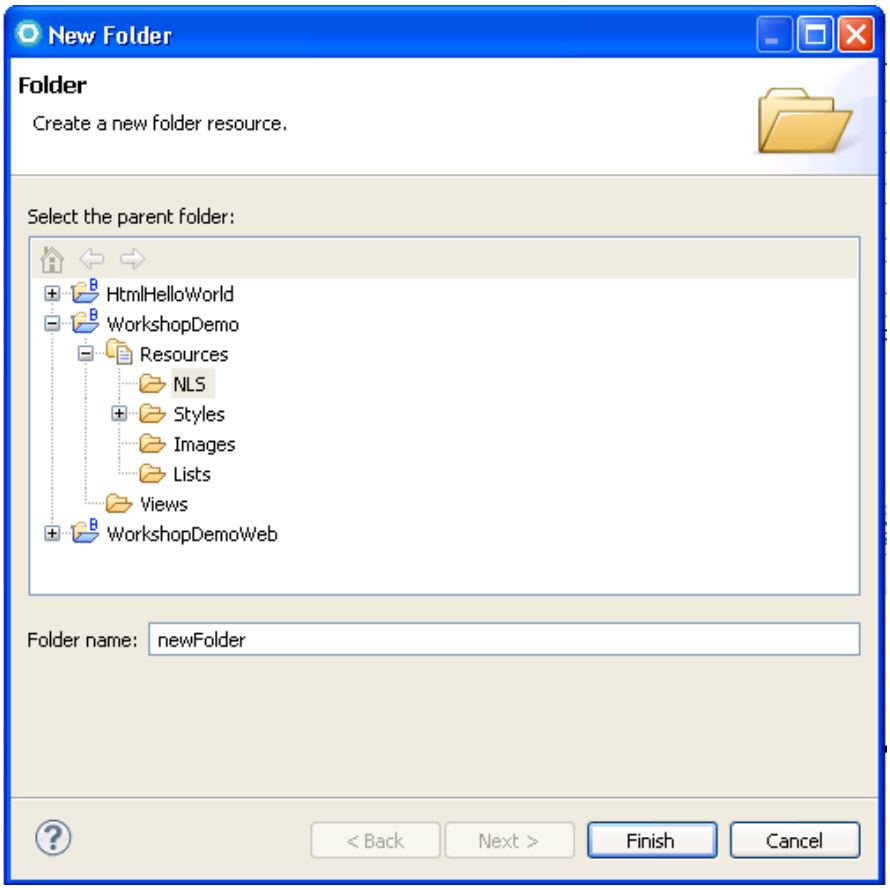
This folder contains several development resources that are needed during the application transactions development: properties files for NLS support, images used during views development, the css style files used to customize the style of the view's widgets and any other shared resources (for example, the Lists folder may contain the files with the valid business values for a combo or select list widgets). These resources are organized in folders, as shown in the Figure below:



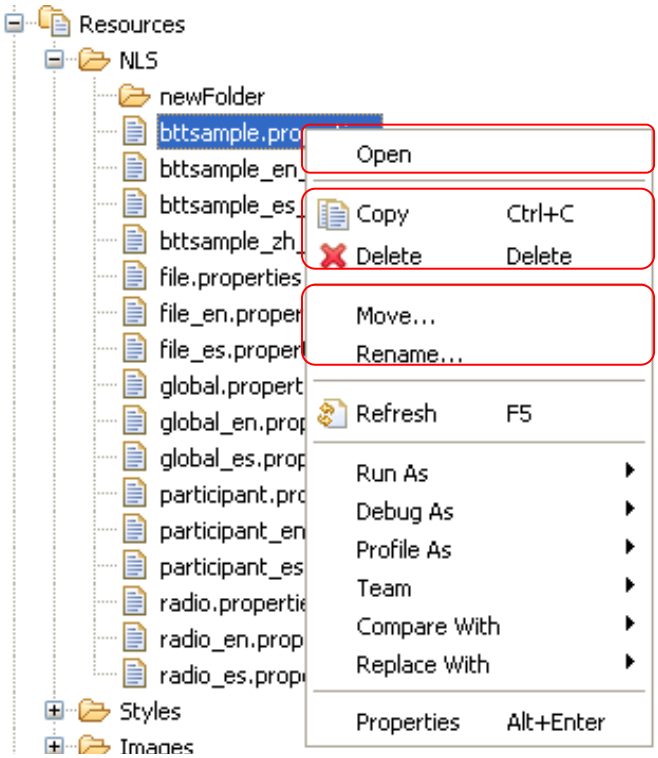
These resources are created and set as part of the development environment by the technical developers, and will be automatically accessed by the editors whenever required to set a specific component property value. No new folders can be created under the Resources folder, but you can add new folders inside each of the default subfolders (NLS, Styles, Images and Lists), that will be able to contain resources of that specific type.



Only the folder creation option is presented if, having the Operations folder selected, you go to the Menu Bar and select the File > New option or you drop-down the list of options in the icon  in the Tool Bar. The **Folder…** option launches the New Folder wizard, which controls that the folder is created in a valid parent folder.



When one of the resources inside any of the default folders is selected, the possible actions shown in the context menu (right-clicking on the operation) are shown and described below:

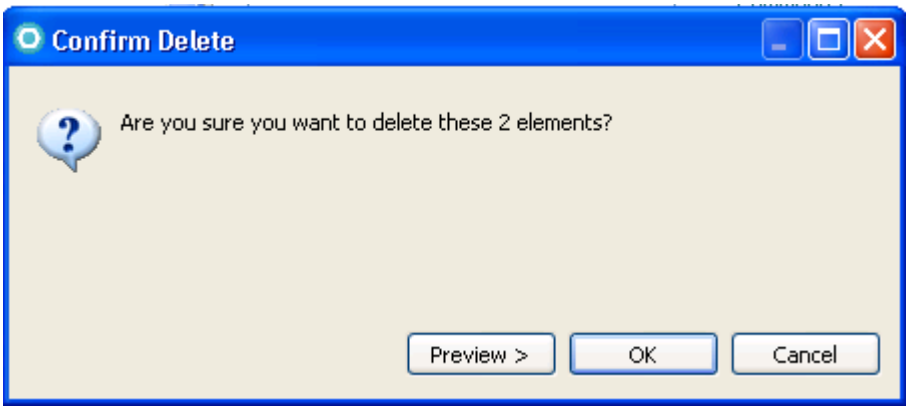


- **Open**- Opens the editor associated to the selected file type. If you select several files, different editor tabs are opened in the perspective. For a single file selection, the Open action behaves the same as double-clicking on the file name.
- **Copy/Paste/Delete**- Allows you to copy, paste or delete the selected files (multiple selection is allowed for files of the same type).

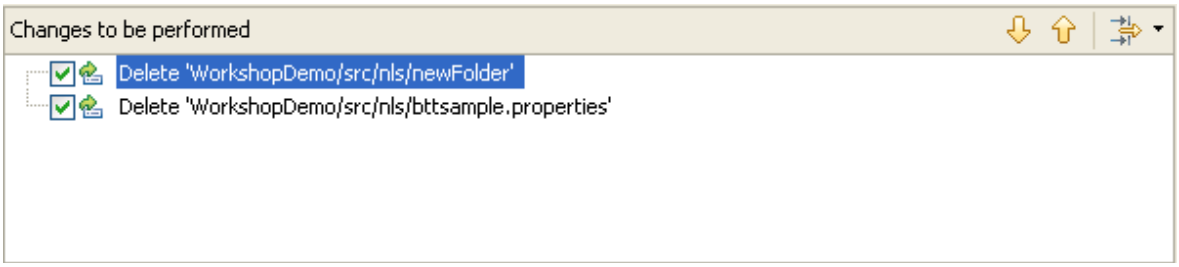
  When you copy a file or a folder, you will only be allowed to paste it on a folder containing resources of the same type, either using the context menu Paste option or using the associated short-key.

  The Paste option is only available when one or several files or folders in this project or any other project in the workspace have been copied to the perspective clipboard. All copied files and folders are pasted to the same destination folder, which should match the files and folders type.

  When you delete a file, you are prompted for confirmation. If you press OK the file will be deleted from your development environment.



If you want to verify the actual files that are going to be deleted from the system, you can click on the Preview button:

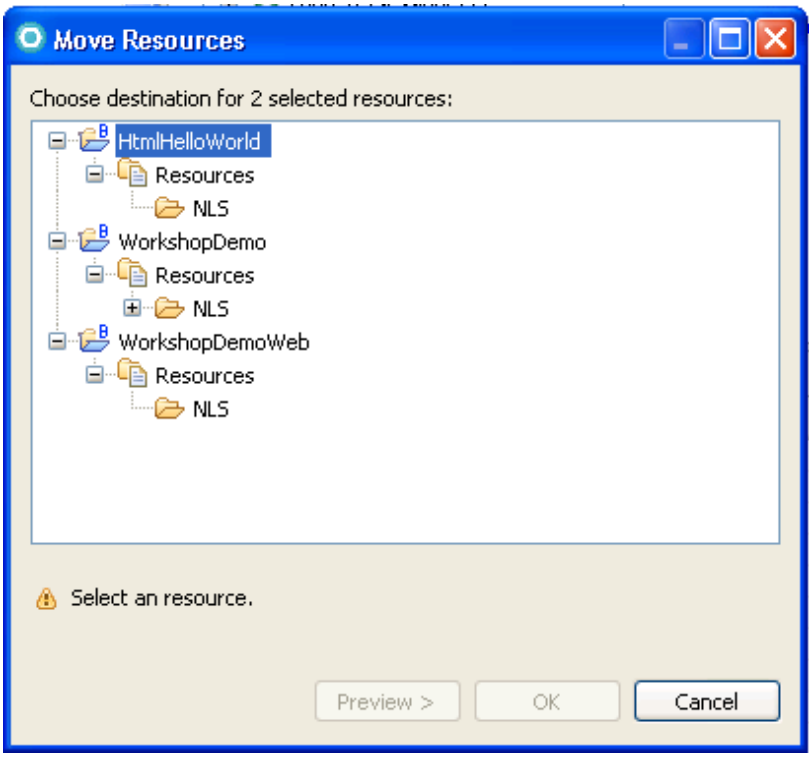


For a folder, all the folder content is deleted.

Note: although you are allowed to uncheck some of the actions that are listed in the Preview view, this is not recommended as may generate inconsistencies in your development environment.

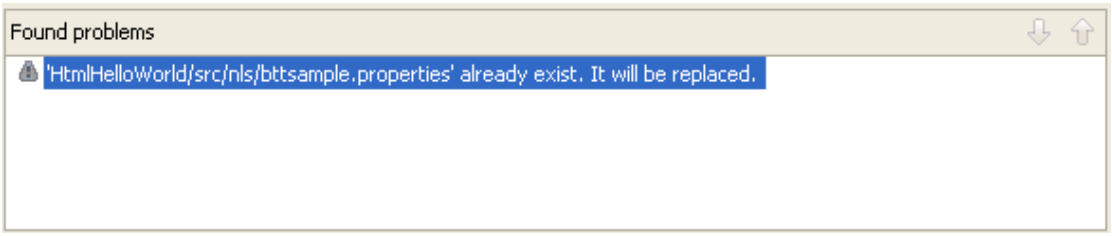
- **Move…/Rename…**. Allows you to move or rename the selected files and folders.

  When selecting the **Move…** option, the Move Resources wizard opens to choose the destination folder. A file or folder containing files can only be moved to another resources folder of the same type and the origin folder cannot be selected. When selecting several files and folders of the same type, all of them will be moved to the same destination folder.

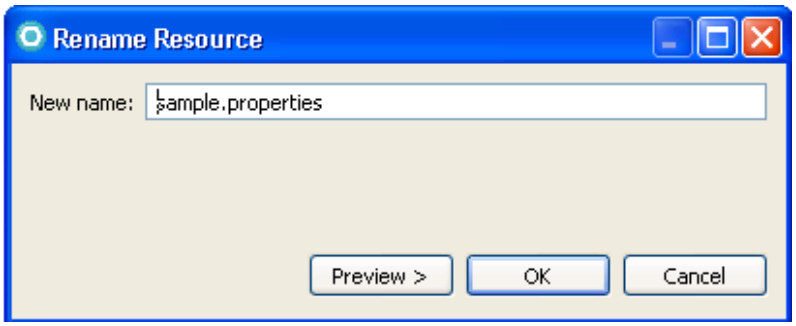


If you want to verify the actual files that are going to be moved as a result of your action or if there is any problem (for instance, a resource with the

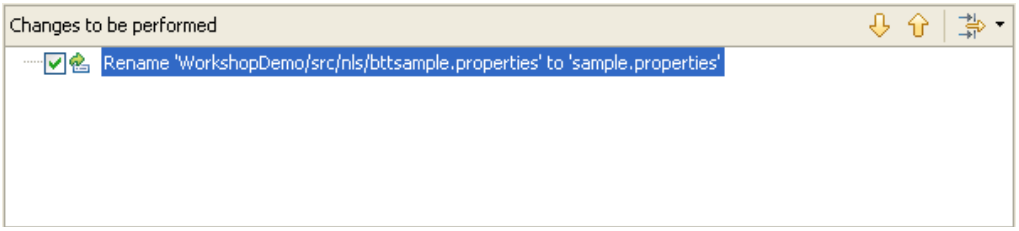
same name already exists in the destination folder), you can click on the Preview button:



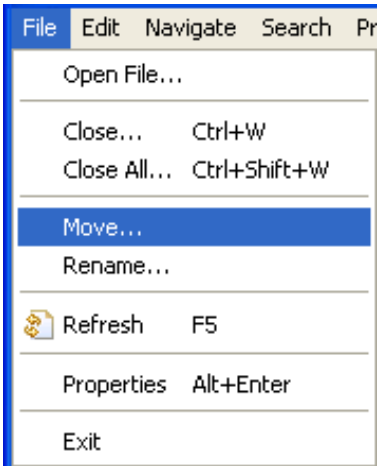
The **Rename…** option is only used with a single selection. If several file or folders are selected, the Rename… will apply to the first file or folder in the list. When selecting the Rename… option, the Rename Resource wizard opens to choose the new name. The name you enter is the full file or folder name, including the extension in the case of files.



If you want to verify the actual files that are going to be renamed as a result of your action or if there is any problem (for instance, a resource with the same name already exists in the destination folder), you can click on the Preview button.



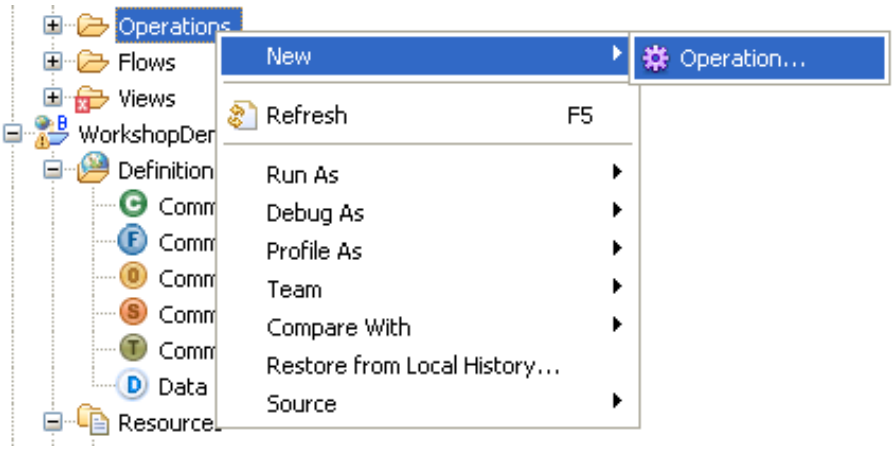
These options are also available when selecting the File option in the Menu Bar:



3.1.5 Operations folder

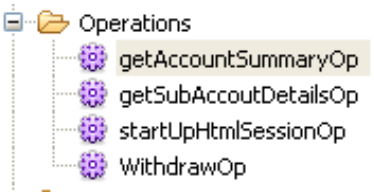
This folder contains all the operations that you create during your development process.

If you right-click on the Operations folder label and select the New option, only the Operation element is available for selection.

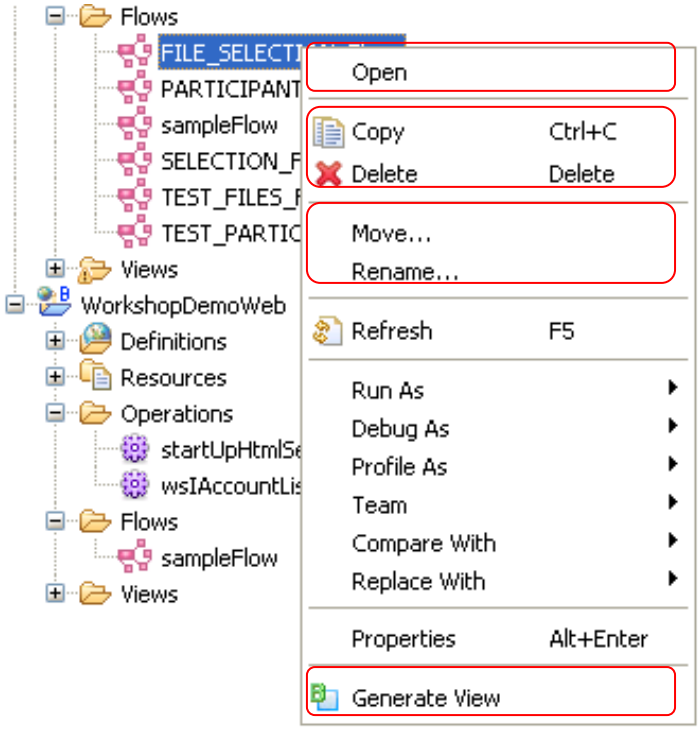


The same occurs if having the Operations folder selected, you go to the Menu Bar and select the File > New option or you drop-down the list of options in the icon  in the Tool Bar. The **Operation…** option launches the Transaction editor in the Operation page (Operation editor).

The operations are labeled using the operation Id and are ordered in alphabetical order.



When an operation is selected, the possible actions shown in the context menu (right-clicking on the operation) are shown and described below:

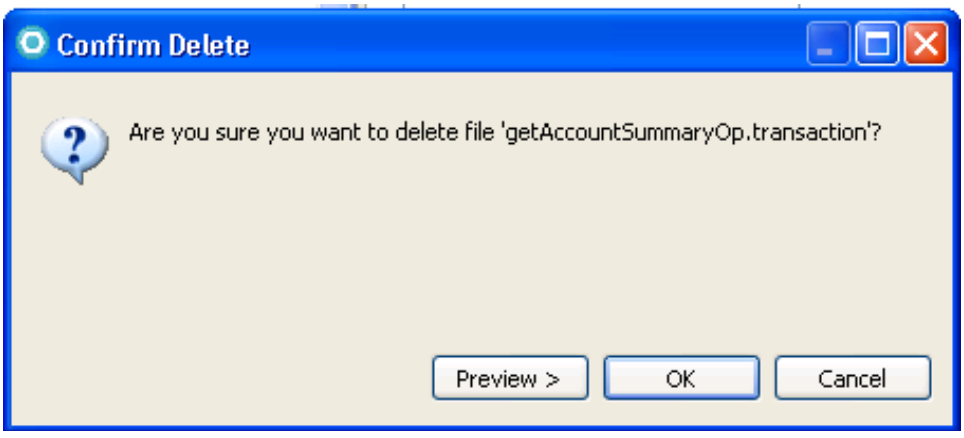


- **Open**- Opens the Operation editor. If you select several operations, different editor tabs are opened in the perspective. For a single operation selection, the Open action behaves the same as double-clicking on the operation name.
- **Copy/Paste/Delete-** Allows you to copy, paste or delete the selected operations (multiple selection is allowed).

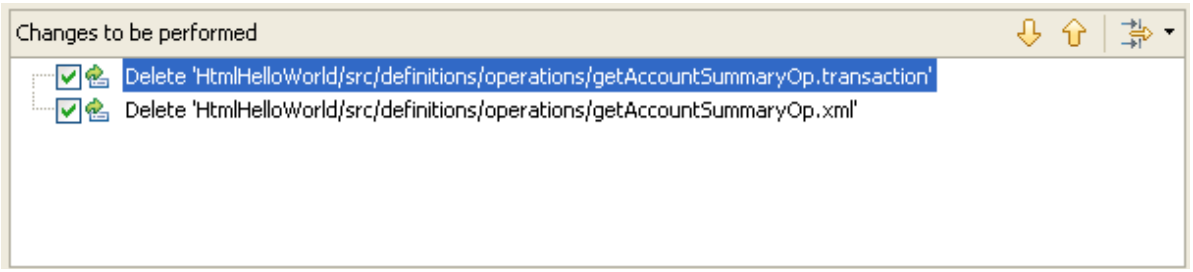
  When you copy an operation, you will only be allowed to paste it on another Operations folder, either using the context menu Paste option or using the associated short-key.

  The Paste option is only available when one or several operations in this project or any other project in the workspace have been copied to the perspective clipboard. All copied operations are pasted to the same destination folder.

  When you delete an operation, you are prompted for confirmation. If you press OK the operation will be deleted from your development environment.



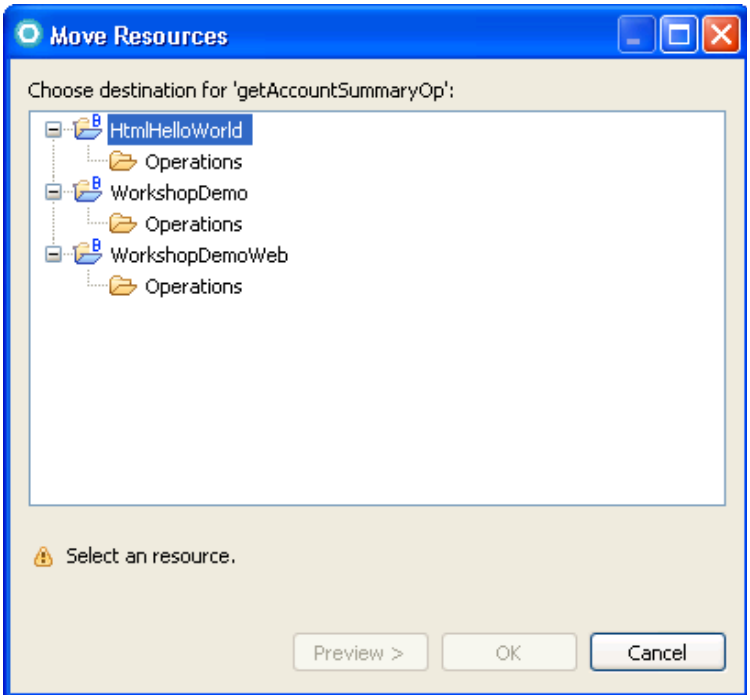
If you want to verify the actual files that are going to be deleted from the system, you can click on the Preview button:



Note: although you are allowed to uncheck some of the actions that are listed in the Preview view, this is not recommended as may generate inconsistencies in your development environment.

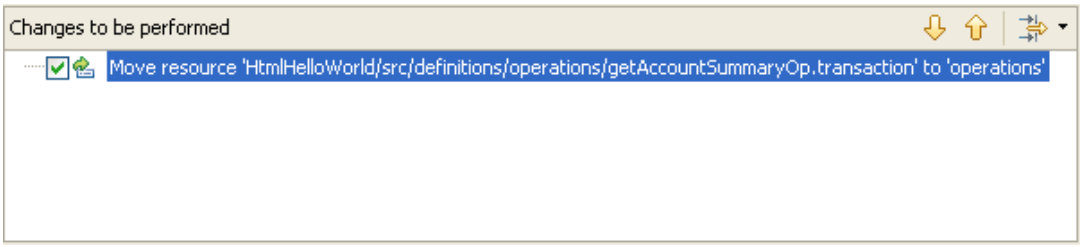
- **Move…/Rename…**. Allows you to move or rename the selected operations.

  When selecting the **Move…** option, the Move Resources wizard opens to choose the destination folder. An operation can only be moved to another Operations folder and the origin folder cannot be selected. When selecting several operations, all of them will be moved to the same destination folder.

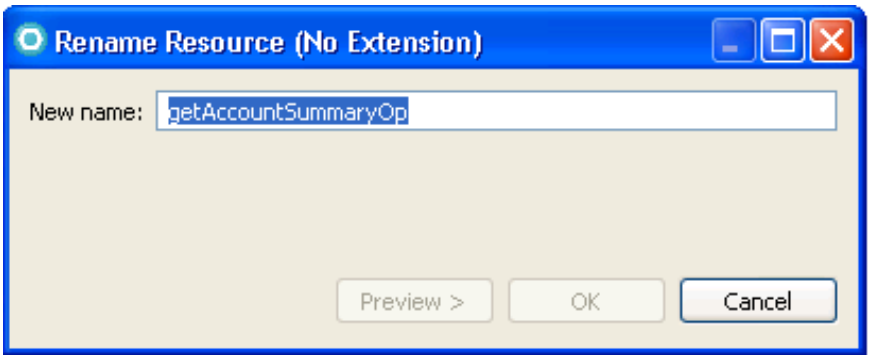


If you want to verify the actual files that are going to be moved as a result of your action or if there is any problem (for instance, a resource with the

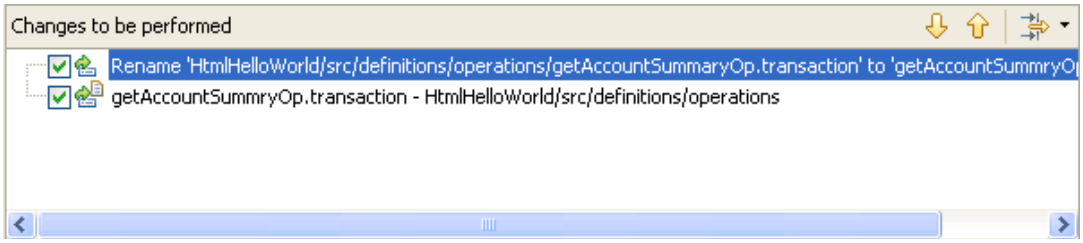
same name already exists in the destination folder), you can click on the Preview button:



The **Rename…** option is only used with a single selection. If several operations are selected, the Rename… will apply to the first operation in the list. When selecting the Rename… option, the Rename Resource wizard opens to choose the new name. The name you enter has to be with no extension.

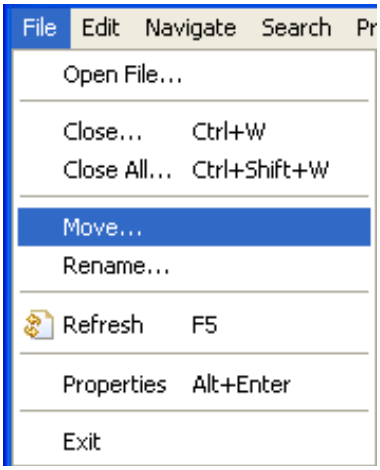


If you want to verify the actual files that are going to be renamed as a result of your action or if there is any problem (for instance, a resource with the same name already exists in the destination folder), you can click on the Preview button



Note: although you are allowed to uncheck some of the actions that are listed in the Preview view, this is not recommended as may generate inconsistencies in your development environment.

These options are also available when selecting the File option in the Menu Bar:

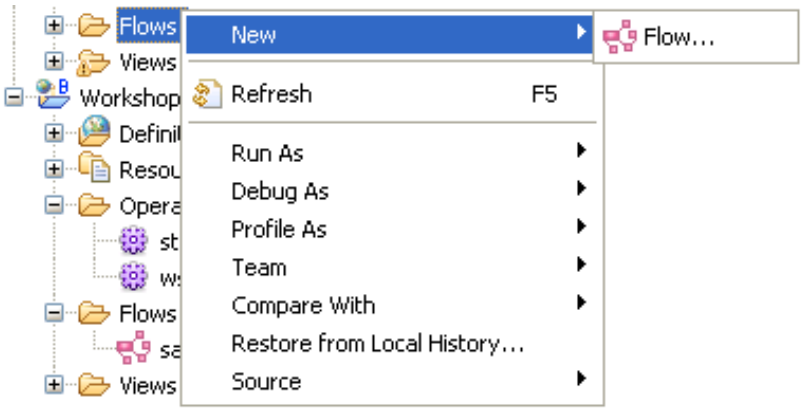


- **Generate View**. This option allows you to generate a view from the operation context data. For more details about this option, go to section Generating a view from the transaction context data.

3.1.6 Flows folder

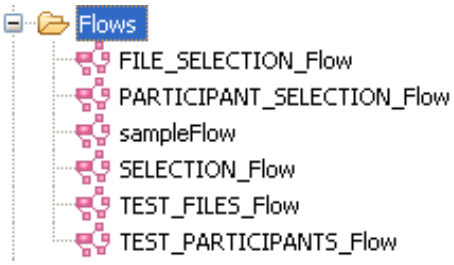
This folder contains all the flows that you create during your development process.

If you right-click on the Flows folder label and select the New option, only the Flow element is available for selection.

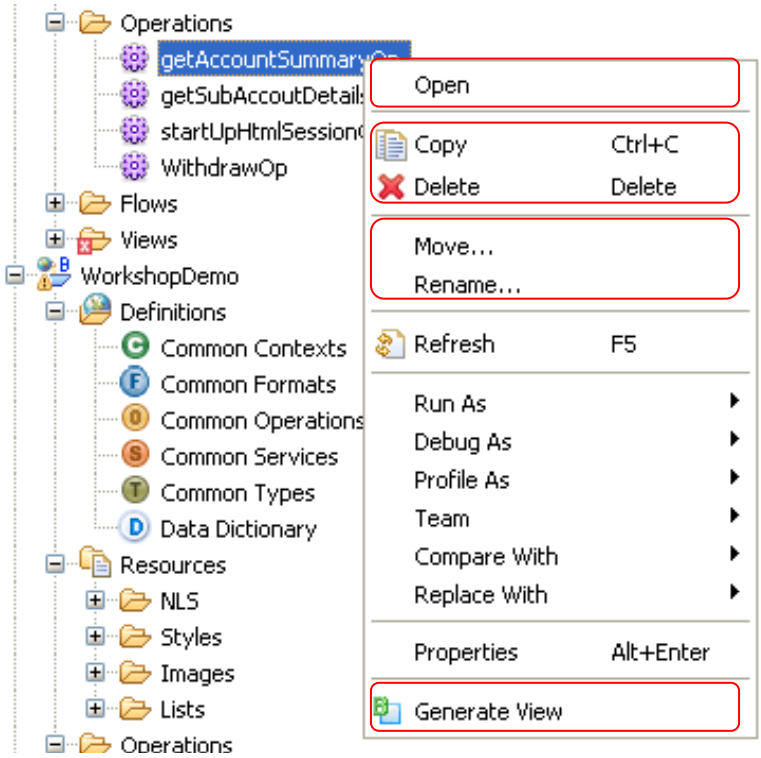


The same occurs if having the Flows folder selected, you go to the Menu Bar and select the File > New option or you drop-down the list of options in the icon  in the Tool Bar. The **Flow…** option launches the Transaction editor in the Flow page (Flow editor).

The flows are labeled using the flow Id and are ordered in alphabetical order.



When a flow is selected, the possible actions shown in the context menu (right-clicking on the operation) are shown and described below:

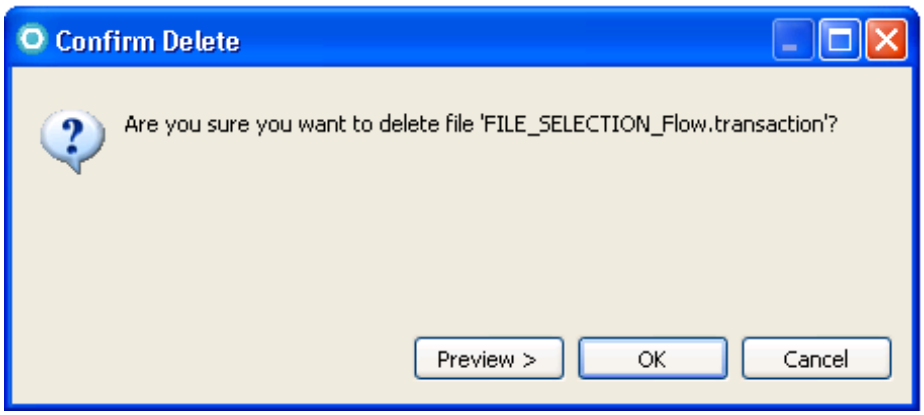


- **Open**- Opens the Flow editor. If you select several flows, different editor tabs are opened in the perspective. For a single flow selection, the Open action behaves the same as double-clicking on the flow name.
- **Copy/Paste/Delete**- Allows you to copy, paste or delete the selected flows (multiple selection is allowed).

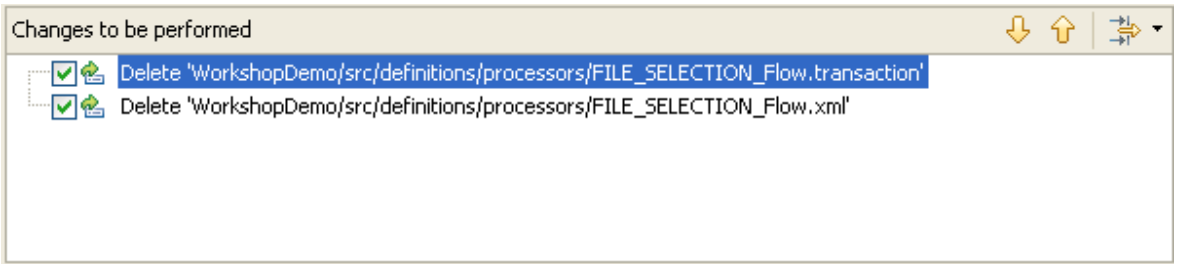
  When you copy a flow, you will only be allowed to paste it on another Flows folder, either using the context menu Paste option or using the associated short-key.

  The Paste option is only available when one or several flows in this project or any other project in the workspace have been copied to the perspective clipboard. All copied flows are pasted to the same destination folder.

  When you delete a flow, you are prompted for confirmation. If you press OK the flow will be deleted from your development environment.



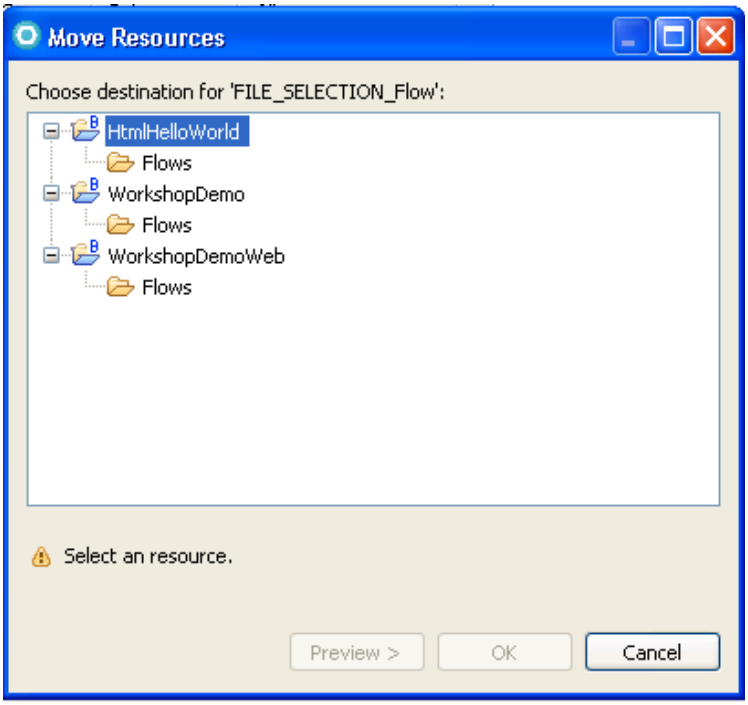
If you want to verify the actual files that are going to be deleted from the system, you can click on the Preview button:



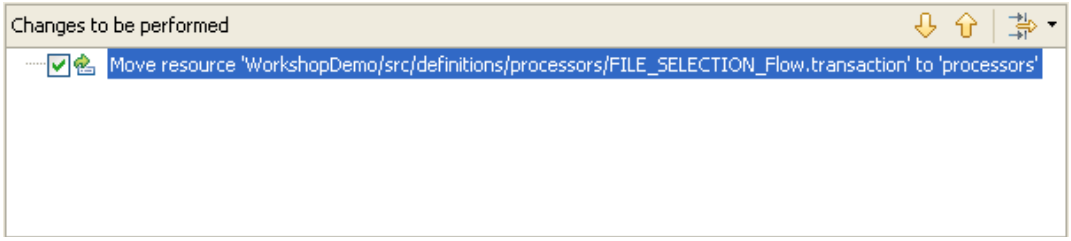
Note: although you are allowed to uncheck some of the actions that are listed in the Preview view, this is not recommended as may generate inconsistencies in your development environment.

- **Move…/Rename…**. Allows you to move or rename the selected flows.

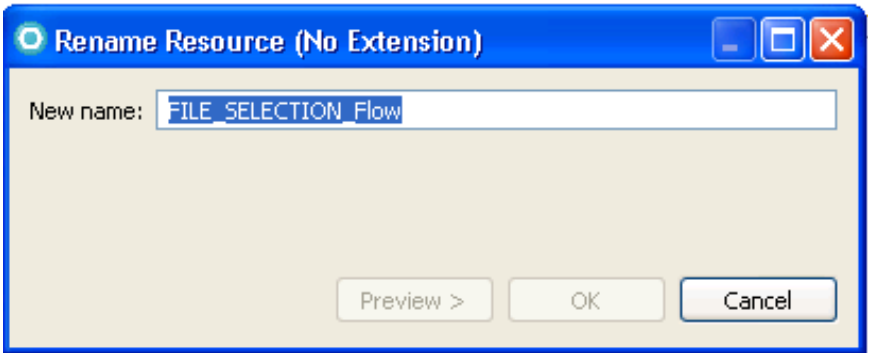
  When selecting the **Move…** option, the Move Resources wizard opens to choose the destination folder. A flow can only be moved to another Flows folder and the origin folder cannot be selected. When selecting several flows, all of them will be moved to the same destination folder.



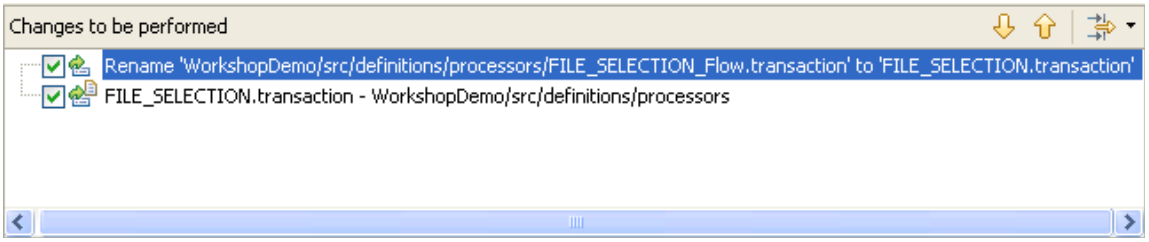
If you want to verify the actual files that are going to be moved as a result of your action or if there is any problem (for instance, a resource with the same name already exists in the destination folder), you can click on the Preview button:



The **Rename…** option is only used with a single selection. If several flows are selected, the Rename… will apply to the first flow in the list. When selecting the Rename… option, the Rename Resource wizard opens to choose the new name. The name you enter has to be with no extension.

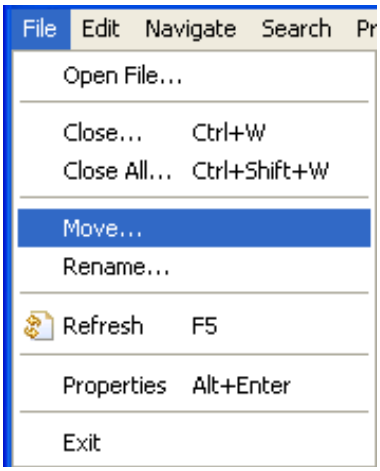


If you want to verify the actual files that are going to be renamed as a result of your action or if there is any problem (for instance, a resource with the same name already exists in the destination folder), you can click on the Preview button.



Note: although you are allowed to uncheck some of the actions that are listed in the Preview view, this is not recommended as may generate inconsistencies in your development environment.

These options are also available when selecting the File option in the Menu Bar:

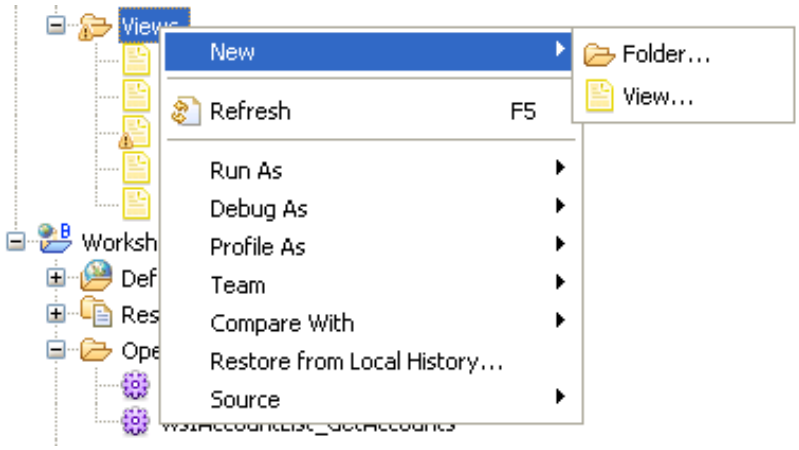


- **Generate View**. This option allows you to generate a view from the flow context data. For more details about this option, go to section Generating a view from the transaction context data.

3.1.7 Views folder

This folder contains all the views that you create during your development process, either using the XUI editor or generating them automatically using any of the existing wizards.

Views can be organized inside the Views folder in a folder structure. Then, if you right-click on the Views folder label and select the New option, you are allowed to create either a folder or a view.

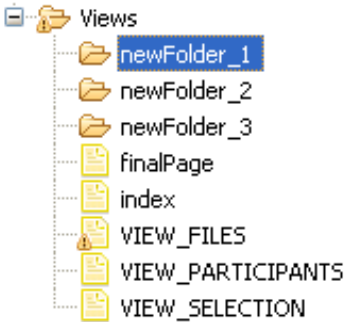


The same occurs if having the Views folder selected, you go to the Menu Bar and select the File > New option or you drop-down the list of options in the icon  in the Tool Bar.

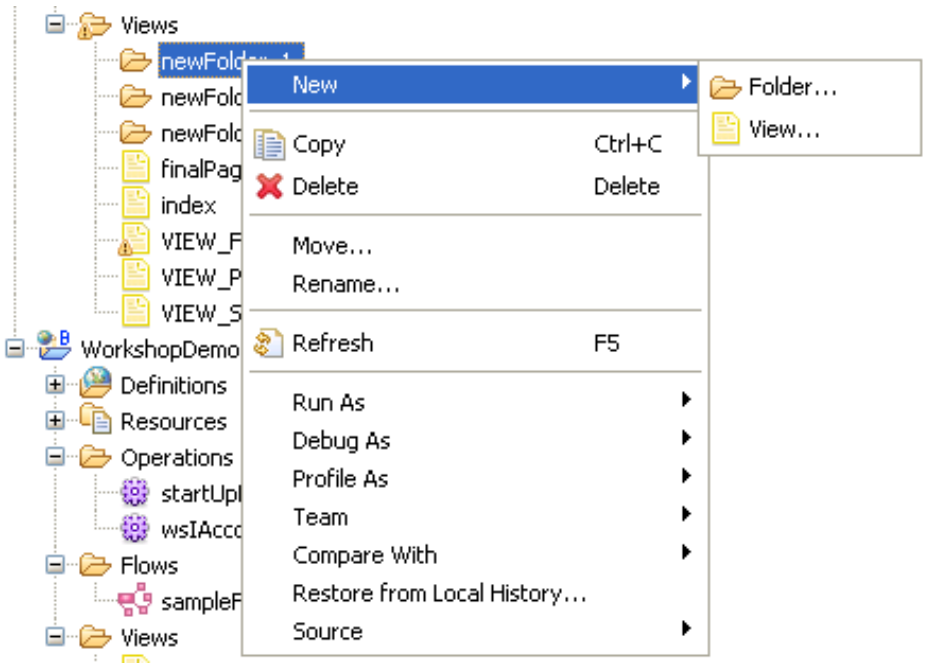
When selecting the **Folder…** option, the New Folder wizard opens and asks for the new folder label and where the folder is to be created; only a Views folder can be selected.

The **View…** option launches the XUI editor.

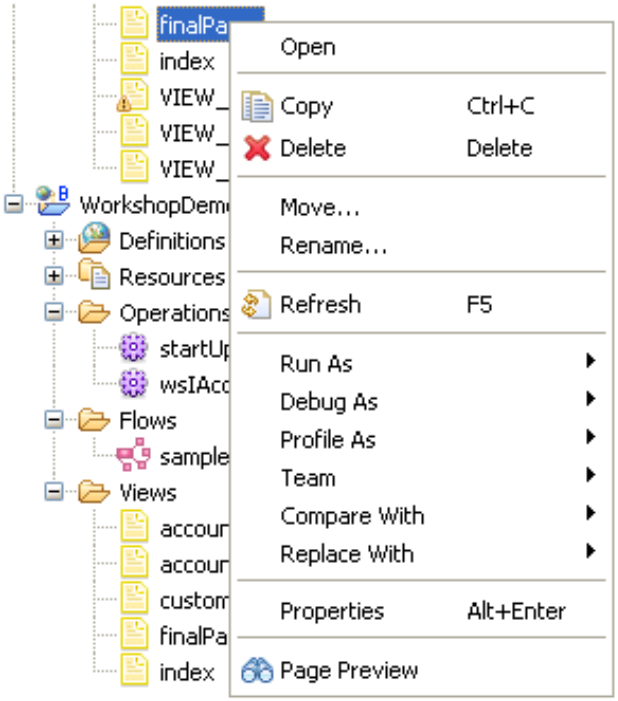
In the Views folder, the views are labeled using the view Id and the folders with the folder name. Folders appear first in the Views folder ordered alphabetically and then you have the views also ordered in alphabetical order; recursively, the views and subfolders inside a folder are ordered following the same criteria.



When a folder is selected, the possible actions shown in the context menu (right-clicking on the folder) are shown below:



When a view is selected, the possible actions shown in the context menu (right-clicking on the view) are shown below:



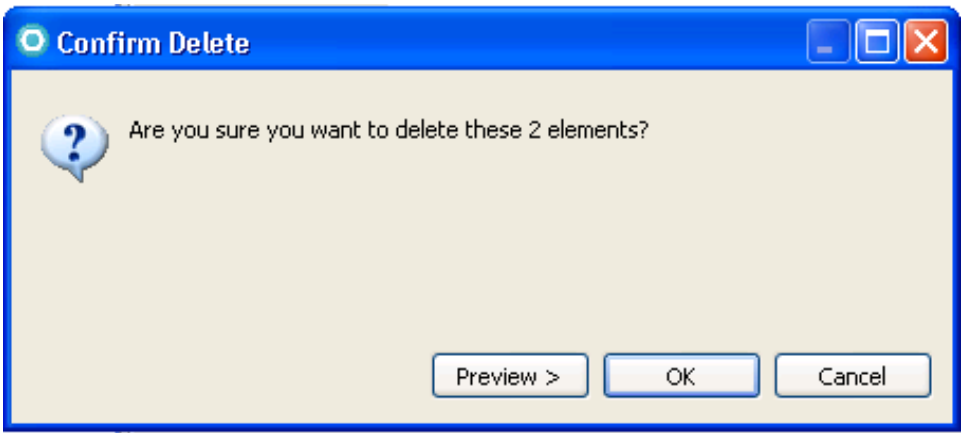
- **New-** Only available if a folder is selected. Inside a folder you can either create another folder or a view.

- **Open**- Only available if a view is selected. Opens the XUI editor. If you select several views, different editor tabs are opened in the perspective. For a single view selection, the Open action behaves the same as double-clicking on the view name.
- **Copy/Paste/Delete-** Allows you to copy, paste or delete the selected views or folders (multiple selection is allowed).

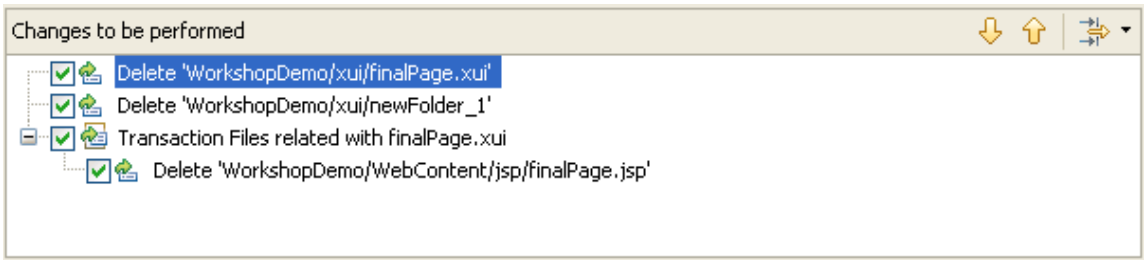
  When you copy a view, you will only be allowed to paste it on another Views folder or subfolder, either using the context menu Paste option or using the associated short-key.

  The Paste option is only available when one or several views or folders in this project or any other project in the workspace have been copied to the perspective clipboard. All copied views and folders are pasted to the same destination folder.

  When you delete a view or folder, you are prompted for confirmation. If you press OK the elements will be deleted from your development environment.



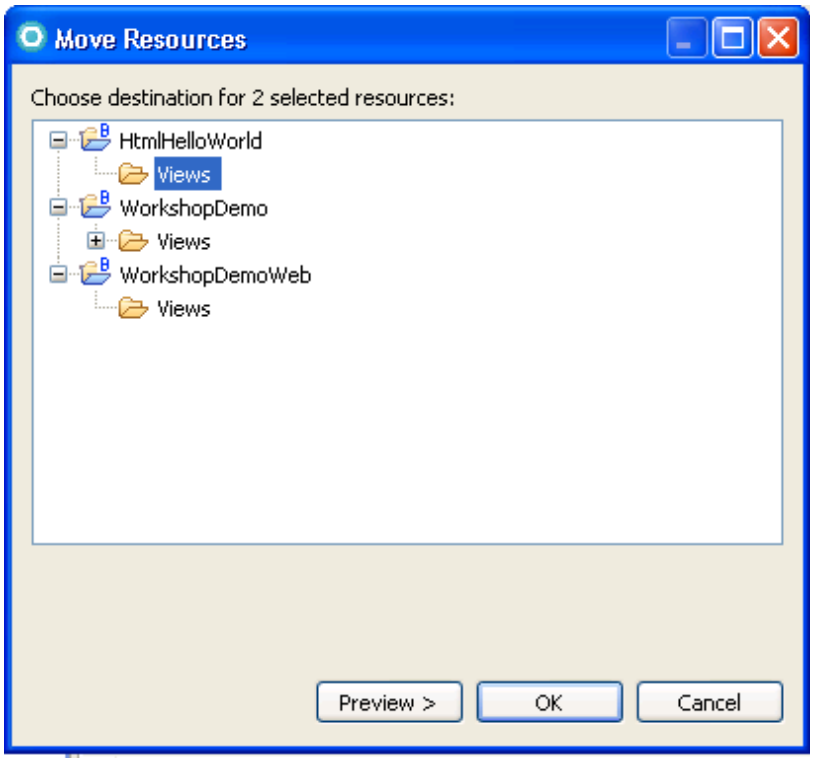
If you want to verify the actual files that are going to be deleted from the system, you can click on the Preview button:



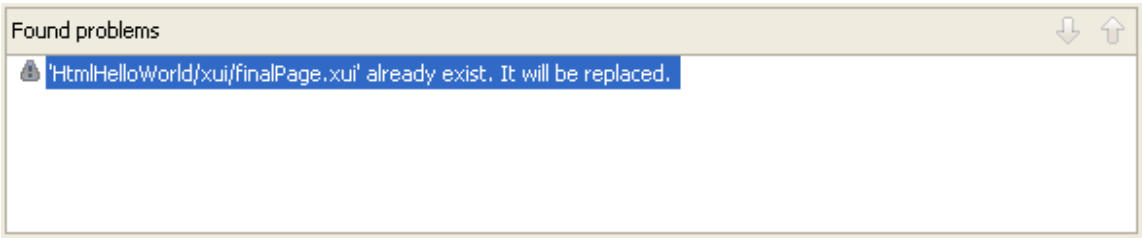
Note: although you are allowed to uncheck some of the actions that are listed in the Preview view, this is not recommended as may generate inconsistencies in your development environment.

- **Move…/Rename…**. Allows you to move or rename the selected view or folders.

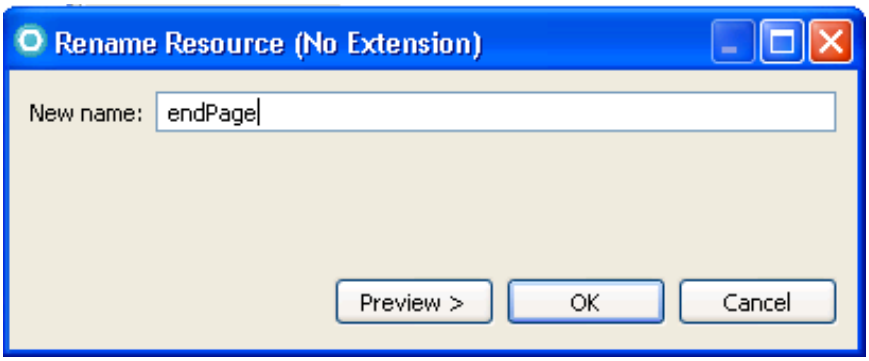
  When selecting the **Move…** option, the Move Resources wizard opens to choose the destination folder. A view or folder can only be moved to another Views folder or subfolder and the origin folder cannot be selected. When selecting several views and folders, all of them will be moved to the same destination folder.



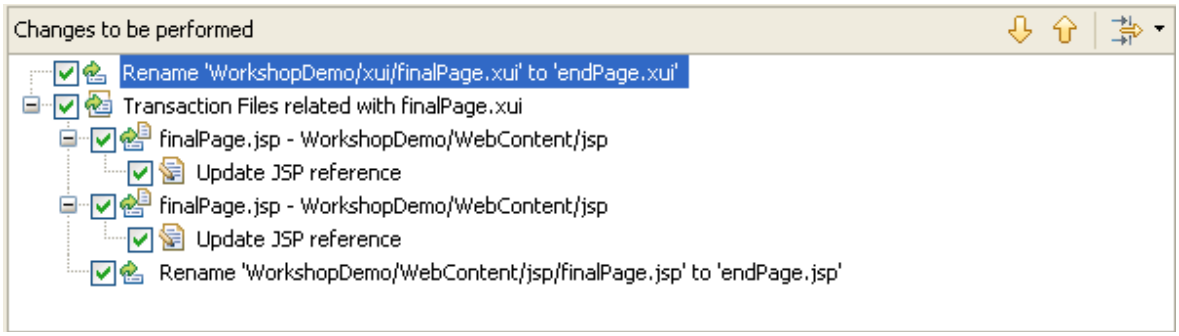
  If you want to verify the actual files that are going to be moved as a result of your action or if there is any problem (for instance, a resource with the same name already exists in the destination folder), you can click on the Preview button:



  The **Rename…** option is only used with a single selection. If several views and folders are selected, the Rename… will apply to the first element in the list. When selecting the Rename… option, the Rename Resource wizard opens to choose the new name. The name you enter has to be with no extension.

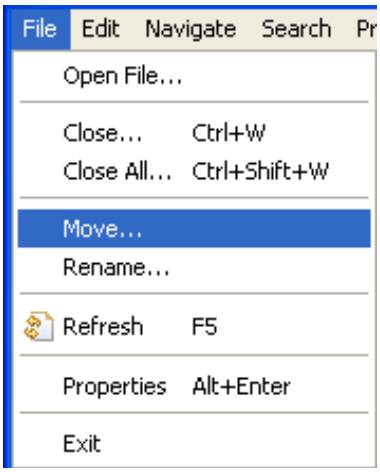


If you want to verify the actual files that are going to be renamed as a result of your action or if there is any problem (for instance, a resource with the same name already exists in the destination folder), you can click on the Preview button.



Note: although you are allowed to uncheck some of the actions that are listed in the Preview view, this is not recommended as may generate inconsistencies in your development environment.

These options are also available when selecting the File option in the Menu Bar:



- **Page Preview**. This option enables you to preview the layout of the page in a web browser. For more details about this option, go to section View Unit Test.

4. The Application Wizard

Application wizard provides a quick way to create BTT Project skeleton. The skeleton includes some necessary elements when you create BTT Application, such as jar, configuration files, JavaScript, servlet and the relationship of BTT Enterprise Application Project. Application Wizard also provides BTT HelloWorld Sample Wizard to create sample project which can help you understand the BTT architecture.

4.1 Creating a BTT Project

To create a BTT Project in the BTT Perspective, do the following steps:

Procedure

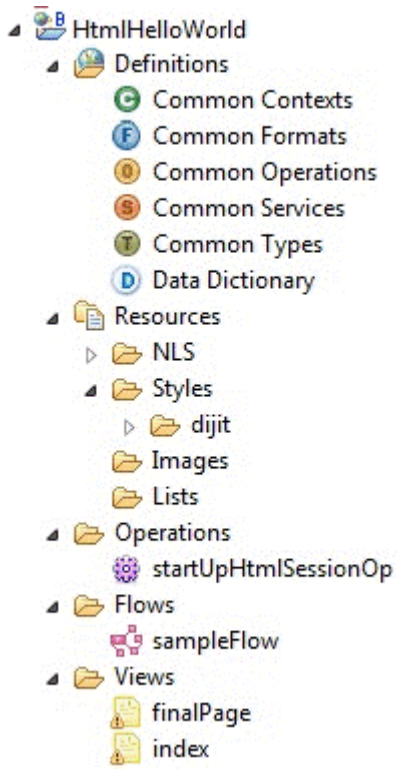
1. Click File > New > BTT Project… The "xui project create wizard" is opened.
2. In the BTT XUI Web Project page, enter details for the XUI web project.
    a. In the Project name field, specify a name for the XUI web project.
    b. In the Project location field, specify the location for the XUI web project. If you do not want to use the default location, clear the Use default location checkbox, and then click Browse to specify a location.
    c. In the Target runtime field, specify the runtime server on which the XUI web project will be deployed.
    d. In the Dynamic web module version field, select the version of Dynamic Web Module facet.
3. Click Next.
4. On the Java page, configure the project for building a Java application. Click Next.
5. On the Web Module page, configure the context root and content directory names for the web project. Click Next.
6. On the BTT Runtime Servlet page, click Finish.

Results

An XUI web project is created. After the XUI web project has been created, the default folders and files are automatically generated. The settings of some default configurations are also automatically generated.

Notice that all generated infrastructure artifacts are hidden in the BTT Perspective and only the BTT projects and the files and folders that may be needed by the Functional developer to proceed with the transactions implementation are shown. Go to section The BTT Project Explorer in this document for a detailed description of a BTT Project

structure in the BTT Perspective and the available contextual menus for each of the different types of resources that can be part of a BTT Project.



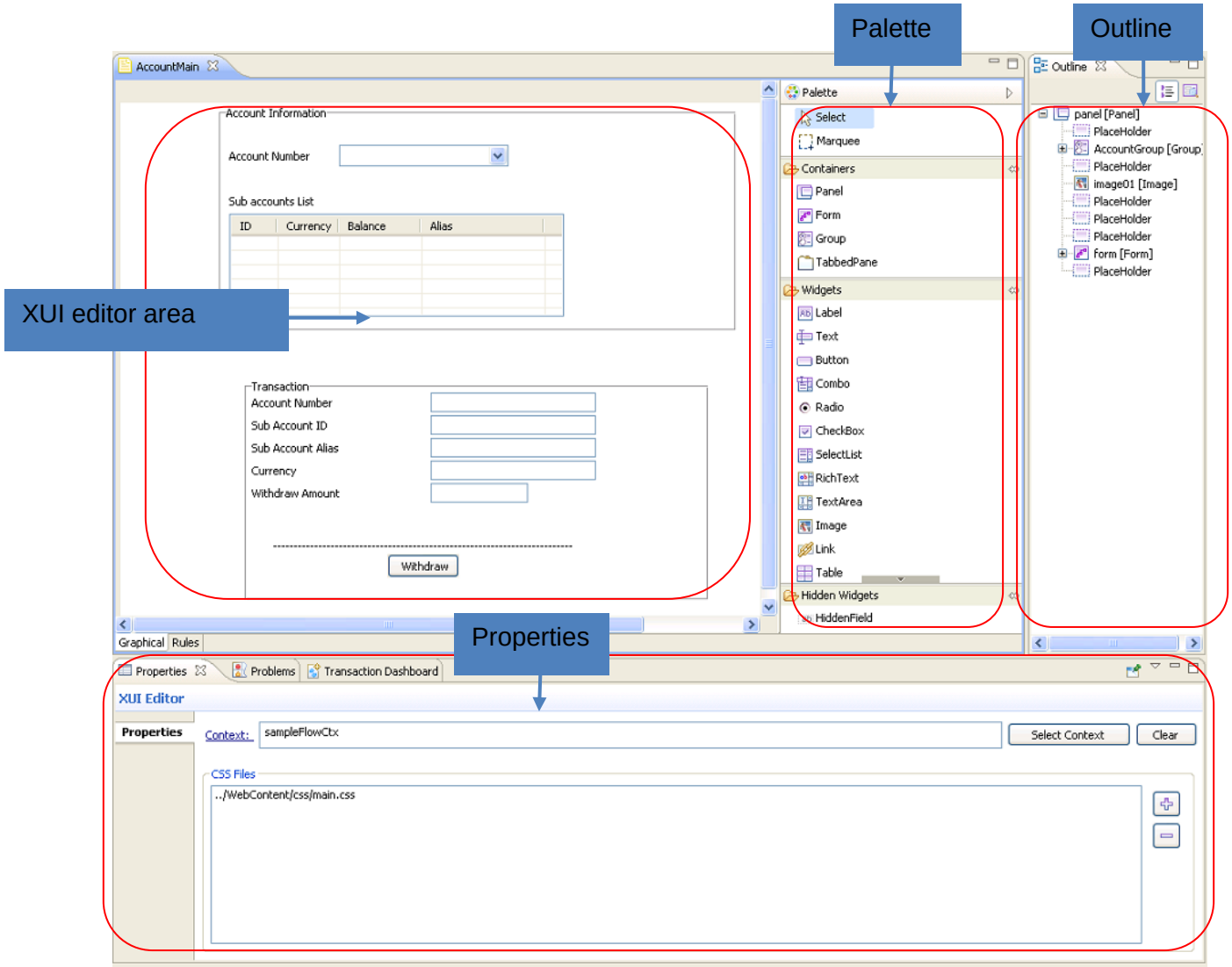
5. Modeling views with the XUI editor

The IBM® WebSphere® Multichannel Bank Transformation Toolkit XUI editor is a WYSIWYG (what you see is what you get) tool that enables you to design and create user interfaces. XUI is an abbreviation for XML User interface.

The XUI editor creates WebSphere Multichannel Bank Transformation Toolkit XML-based transaction UI files. These files, stored in XML format at development time, are then converted to Web 2.0 and SWT widgets at runtime by the WebSphere Multichannel Bank Transformation Toolkit XUI engine.

The Views (sometimes called XUI files) are channel-neutral, which means that they can be converted into channel-specific UI files that comply with the technologies of different channels.

The XUI editor consists of the following four parts:



XUI editor area

The XUI editor area enables you to design GUIs. You can drag and drop the widgets and containers from the Palette to the XUI editor area to design GUIs. Another option to drop the widgets and containers to the XUI editor area is to left-click to select one of them in the Palette and then left-click again to drop the new element in the required position.

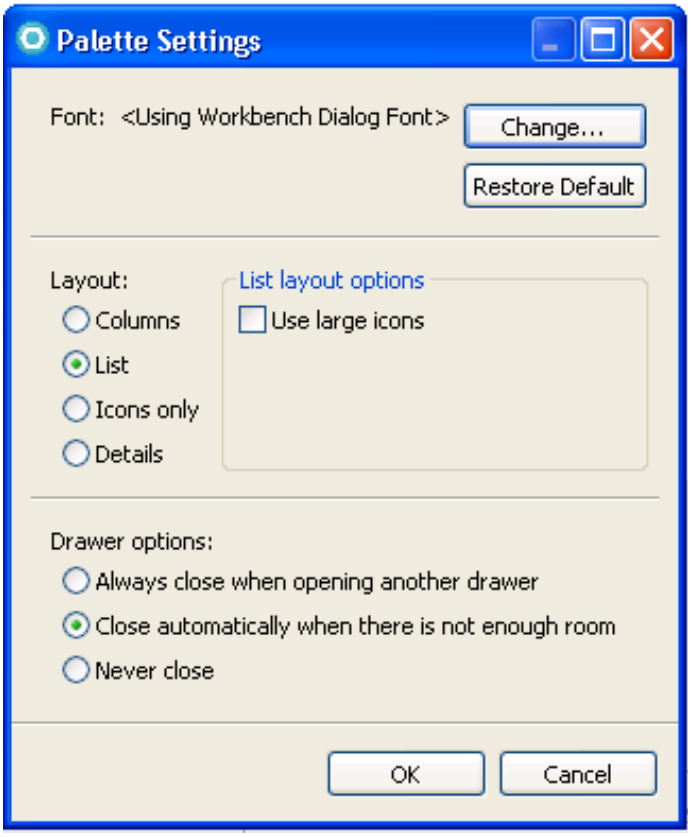
If you click a widget or container element in the XUI editor area, you can edit the properties of the element in the Properties view.

Palette

The Palette view contains the widgets and the containers that enable you to design a view.

To add the Palette view to the current perspective, click Window > Show View > Other…> General > Palette.

The Palette can be configured by right-clicking on it. You can decide how the widgets and containers are shown: organized by columns or in a list, only with icons or with a detailed explanation, select the font for the widgets labels and change the drawer options.



Outline

The Outline view displays an outline of the View that is currently open in the XUI editor area, and it lists the widget and the container elements of the View.

If you click a widget or container element in the Outline view, you can edit the properties of the element in the Properties view.

To add the Outline view to the current perspective, click Window > Show View > Other... > General > Outline.

Properties

The Properties view displays the properties of a selected widget or container. You can edit the properties of the widget and container elements of your View in the Properties view. To add the Properties view to the current perspective, if not available by default when opening the view, click Window > Show View > Other... > General > Properties.

5.1 Creating a view

To access the XUI editor and begin using the widgets, you must first create a View in your BTT Project. There are different ways to create a View:

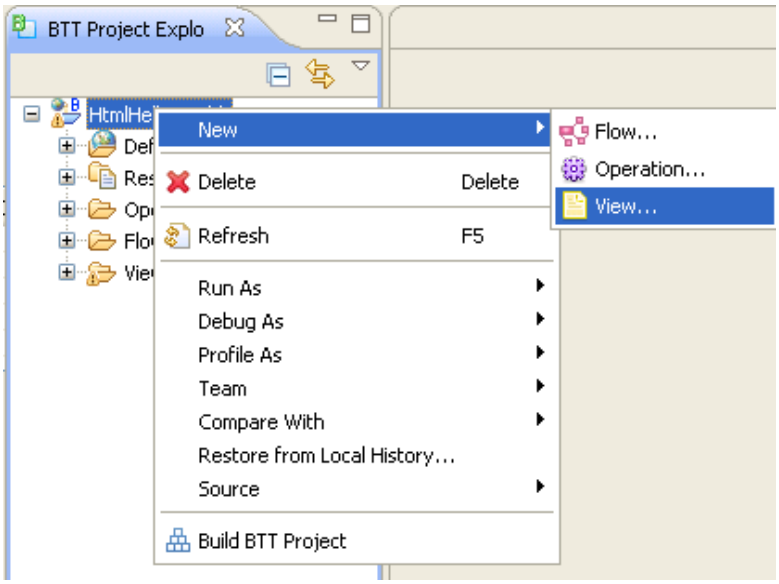
- launching the Create XUI File page from the BTT Project Explorer
- generating the View from the transaction context data

The result of the process in all cases is a View file that is created and is stored in the Views folder of your BTT Project.

5.1.1 Launching the Create XUI File page from the BTT Project Explorer

To create a View file, do the following steps:

1. In the BTT Project Explorer view in Rational® Application Developer, right-click the BTT Project or in the Views folder inside the BTT Project, and then click New > View…. The Create XUI File page is displayed.

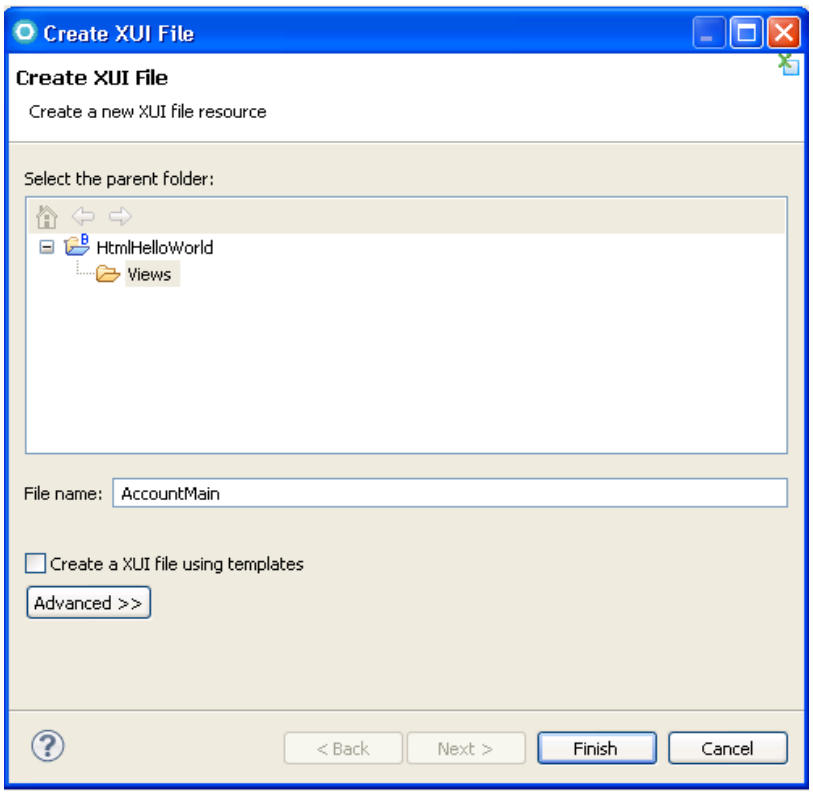


2. Select the parent folder. You can only select a View folder.

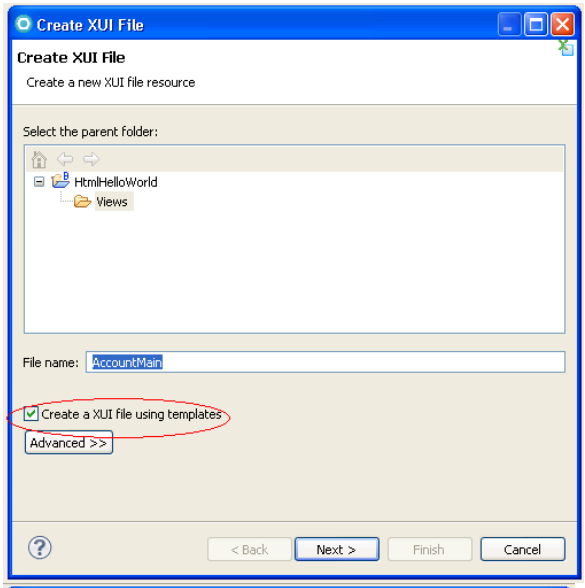
   In the File Name field, specify a name for the view.

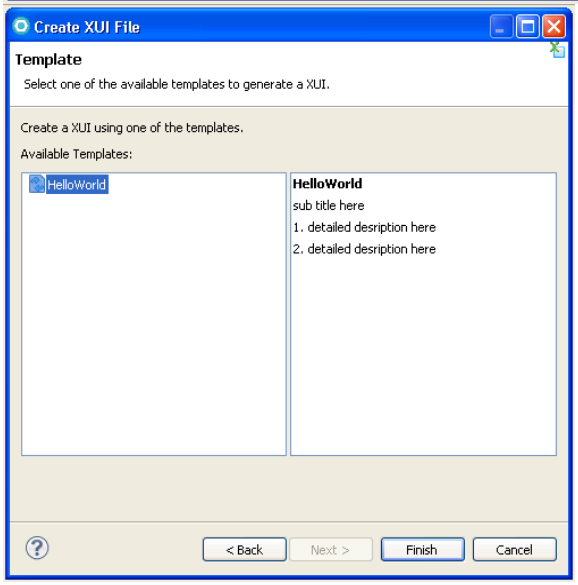
   Note:

   - The file name you specify in the File Name field cannot be duplicated.
   - You must specify a file name without extension. The file will be internally created with an .xui file extension.



3. Optionally, you may want to create a View using an existing XUI template: a predefined view that can be reused. Then, select the **Create a XUI file using template**s check box. Click Next, and then select the template you require.





Note: All the template files can be shared among different projects. The new View inherits all the properties of the original template and can be used as a starting point for your view design. Containers and widgets properties have to be reviewed as they may refer to data names or labels not existing in your BTT Project. Refer to section Using View Templates for more detailed information.

4. Click Finish

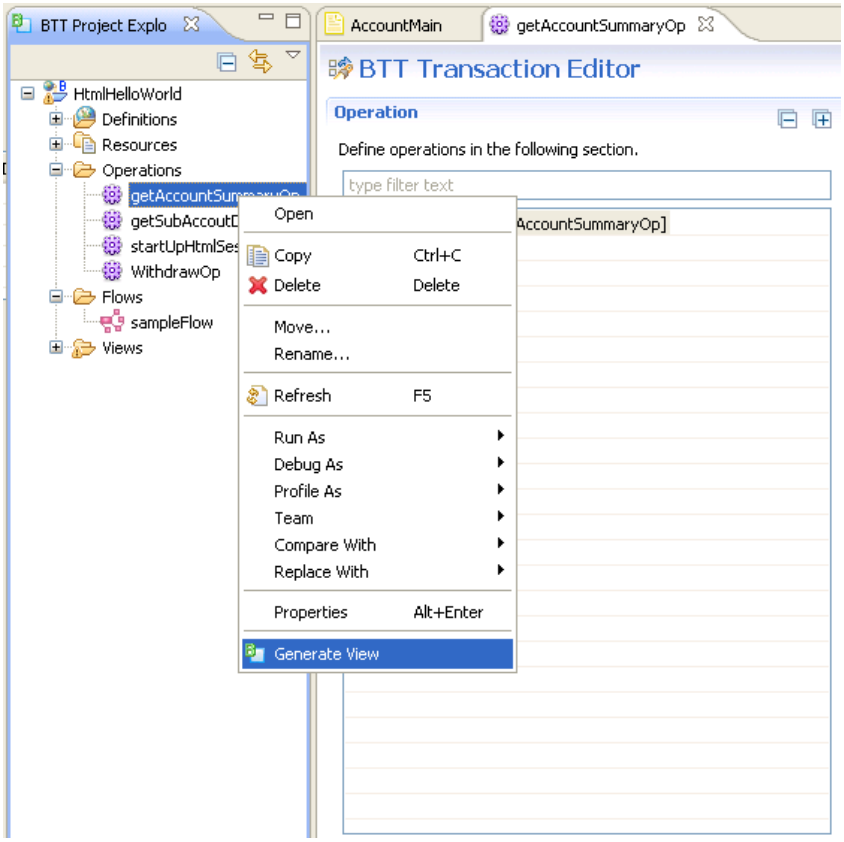
5.1.2 Generating a view from the transaction context data

To use this approach you should first define the data that your transaction is going to manage and the context that keeps the data by using the BTT Transaction Editor (refer

to Transaction Editor section for more details, subsections Defining Context and Defining Data).

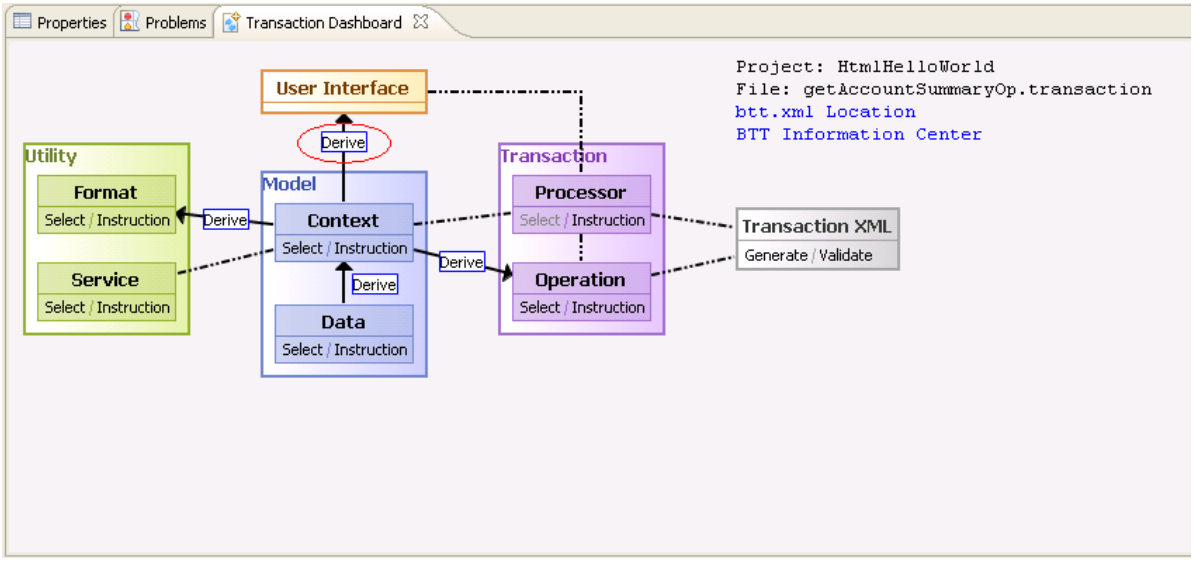
Then, to generate a View by associating the data elements from the context of an operation or a flow to UI widgets, do the following steps:

1. In the Project Explorer, select the operation or the flow file that contains the data elements that you are using to generate the View.
2. Then right-click on the selection and go to the **Generate View** context menu option

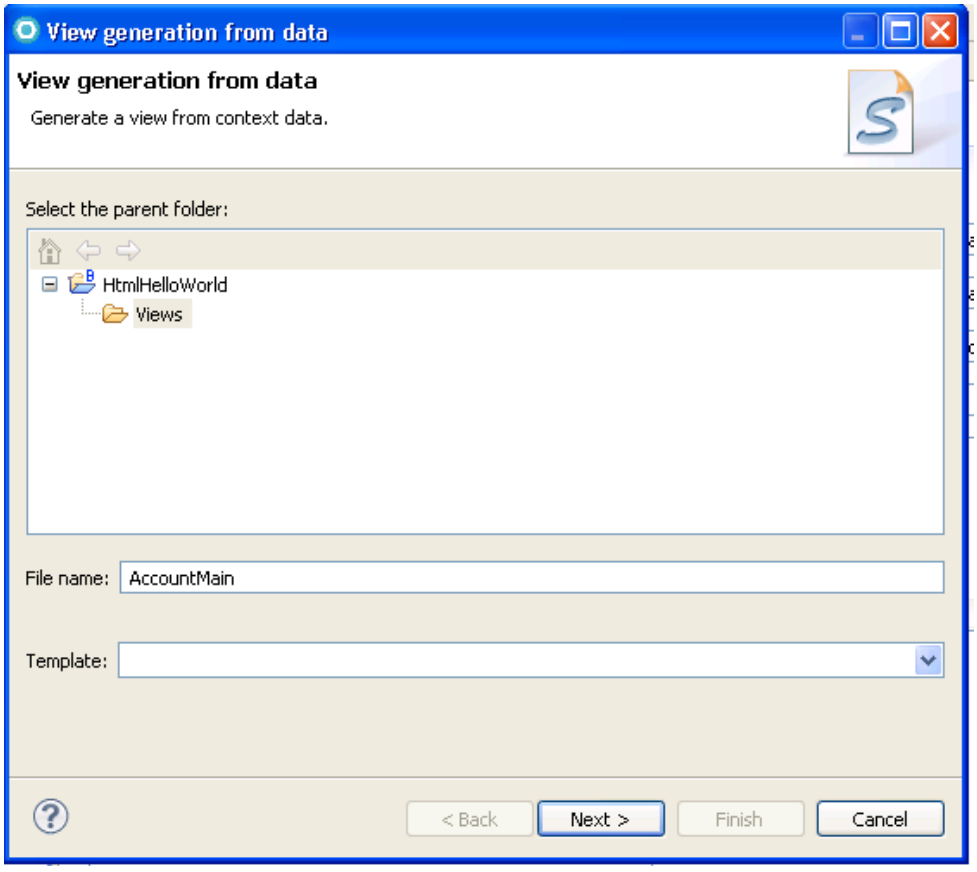


Another approach to generate view is using **Transaction Dashboard**. To open the Transaction Dashboard go to the corresponding tab in the workspace bottom area as shown in the Figure below. In the Transaction Dashboard, you can graphically request for the automatic generation of different resources from your currently defined ones. In this case, double-click the Derive button that is located from context to User Interface.

.



In both cases, the "View generation from data" window opens.



3. In the "Select the parent folder" pane of the "View generation from data" window, select the folder in which you want to save the View that is generated. It must be a Views folder.

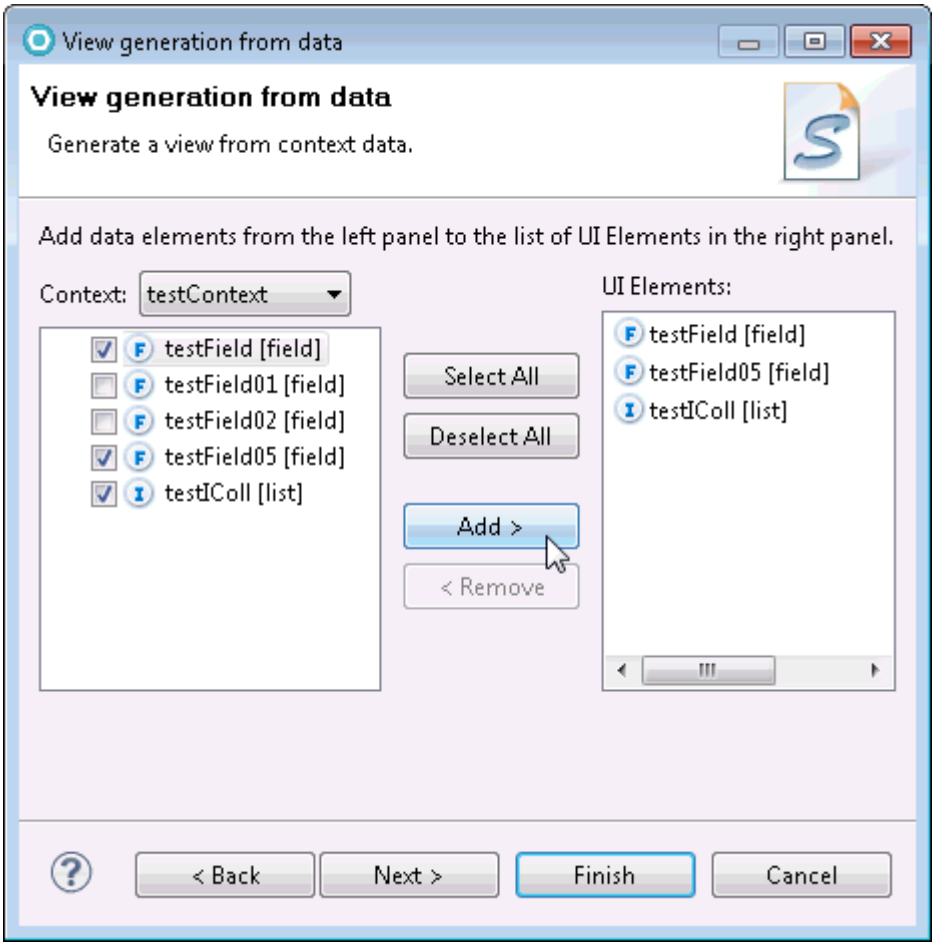
4. In the File name field, enter a name for the View file, either without extension or with a .xui file extension

5. Optionally, from the Template list, you can select the template that you want to use to generate the View file. The template context will be then set, if empty, or updated with the operation context. Refer to section Using View Templates for more detailed information.

6. Click Next.

7. From the Context list, select the data elements that you want to include as UI elements in the View, and then click Add.

   Note: The data elements that are available for selection in the Context list are the data elements from the context that is bound to the flow or operation. If you want to associate data elements from a different context to UI elements, you must change the context that is bound to the flow or operation. For information on how to bind a context to a flow, see Binding a context data to a transaction flow.

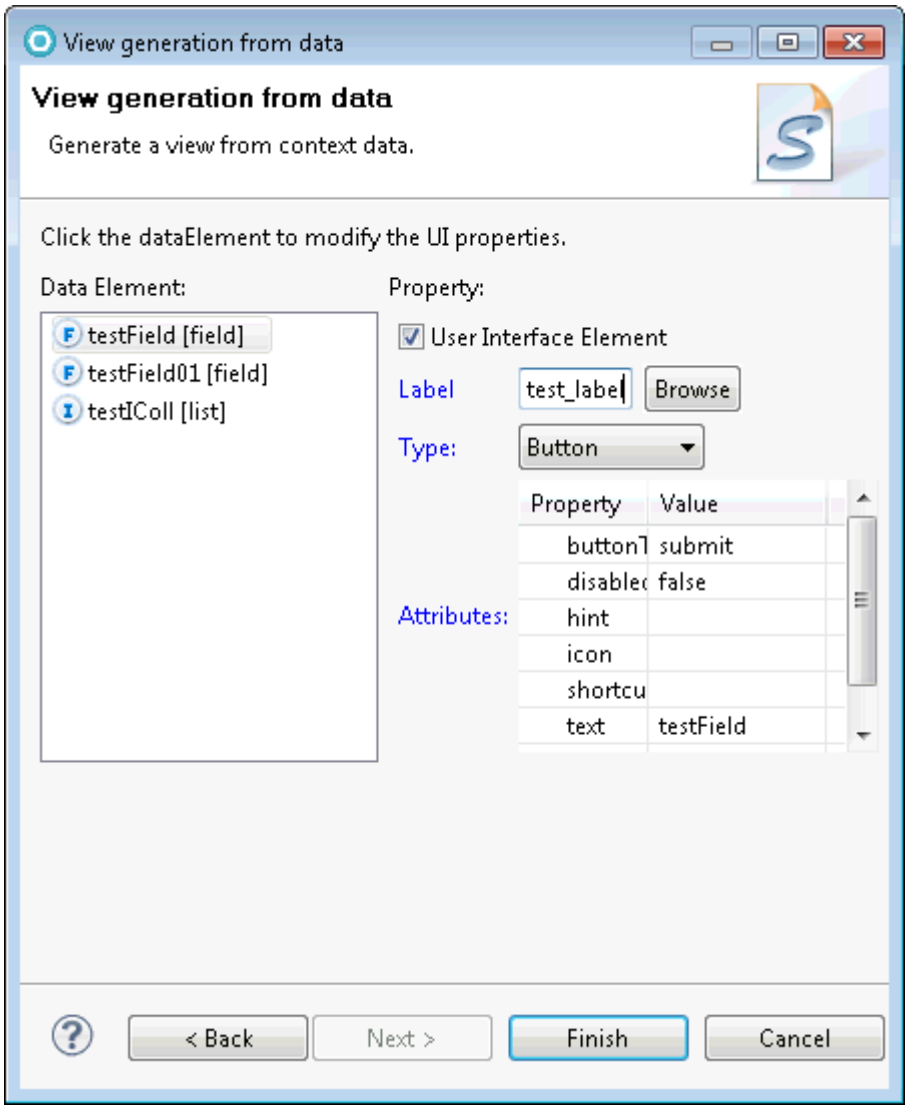


8. Click Next.

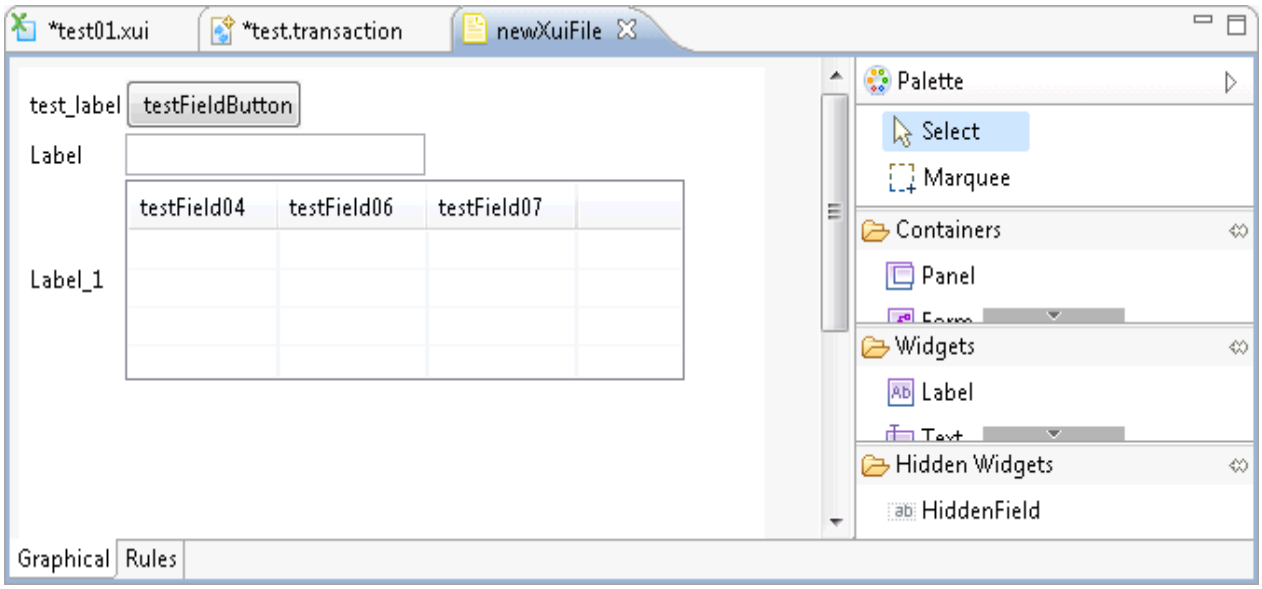
   Note: If you click Finish, the View is generated and displays. If you did not specify the UI widgets that you want to associate to the data elements, the UI widgets that are associated to the data elements by default are

created in the View. To specify the UI widget that you want to associate to a data element, click Next instead of Finish, and proceed to step 9.

9. From the Data Element list, select a data element to modify the UI widget that it is associated to and to modify the properties of the UI widget. In the Label field, specify the text that is displayed on the label next to the UI widget on the View. Click Finish.

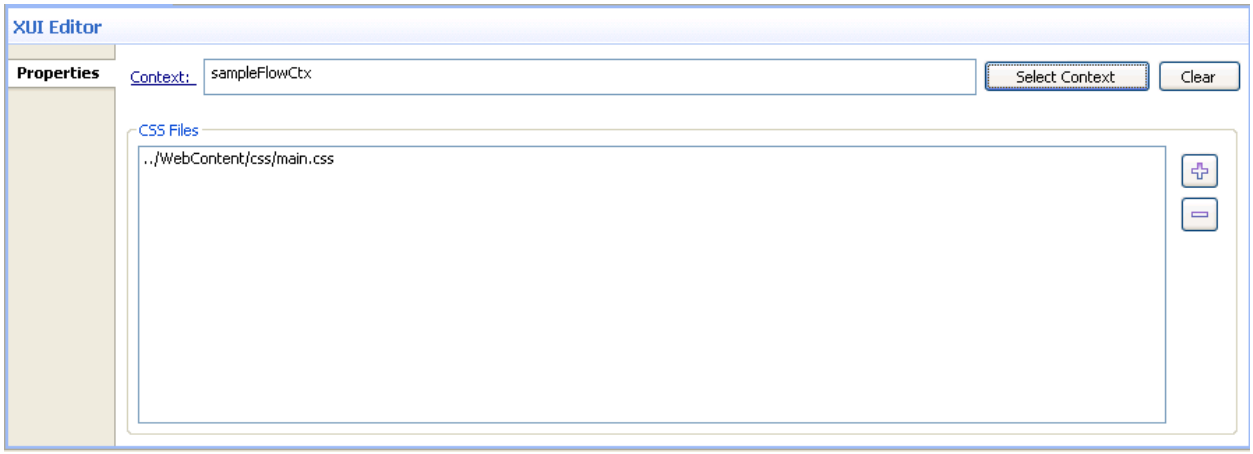


The View generates and displays.



5.2 Mapping a view to a data model

Before editing the view, you should map the view to a data model. This is already done if you have generated the view from already defined context data. This task will enable data to be submitted to a server from the user interface and to be displayed on the user interface. This mapping can be done from the XUI Editor, by clicking outside the root container and going to the Properties tab.

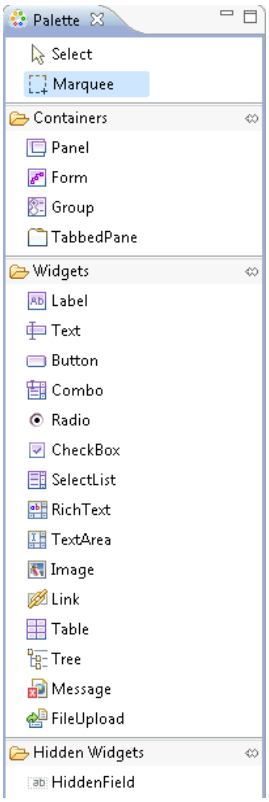


Follow the instructions in section Mapping an XUI file to the data model for a detailed explanation on how to proceed from this screen.

5.3 Editing a view

Once the BTT Project and the view have been created, you can design the view contents without writing code by using the widgets and containers that are provided by the WebSphere Multichannel Bank Transformation Toolkit XUI editor. The XUI containers are objects used to contain widgets in the view.

The following figure shows the widgets and containers that are provided by default by the XUI editor. Any customized widget that may have been developed by the technical developers to suit your project requirements will also appear in the XUI editor palette and can be used as any other widget.



The provided widgets and containers are described, respectively, in sections Widgets Description and Containers Description.

The technical developer uses the BTT Widget Extension framework to enable a customized widget in the XUI editor by defining how a widget is displayed in the XUI editor, the properties of the widget, and how the properties can be edited

5.3.1 Adding widgets and containers

To create a user interface by adding containers and widgets to a view, do the following steps:

1. Open the view.
2. Drag and drop a container from the Palette view to the XUI editor area.

   Note: Each view must contain one root container, and it cannot contain more than one root container. However, a root container can contain other containers as sub-containers.

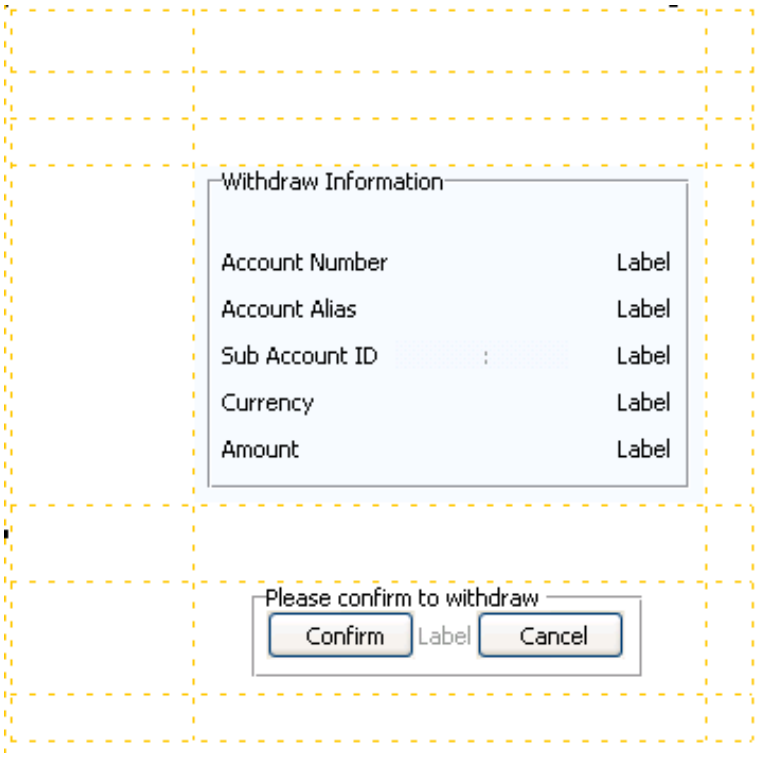
For more information on the containers that are provided by WebSphere® Multichannel Bank Transformation Toolkit, refer to the XUI containers description topic.

3. Drag and drop widgets for the user interface that you are creating from the Palette view to a container in the XUI editor area. For more information on the widgets that are provided by WebSphere Multichannel Bank Transformation Toolkit, see XUI widgets description.

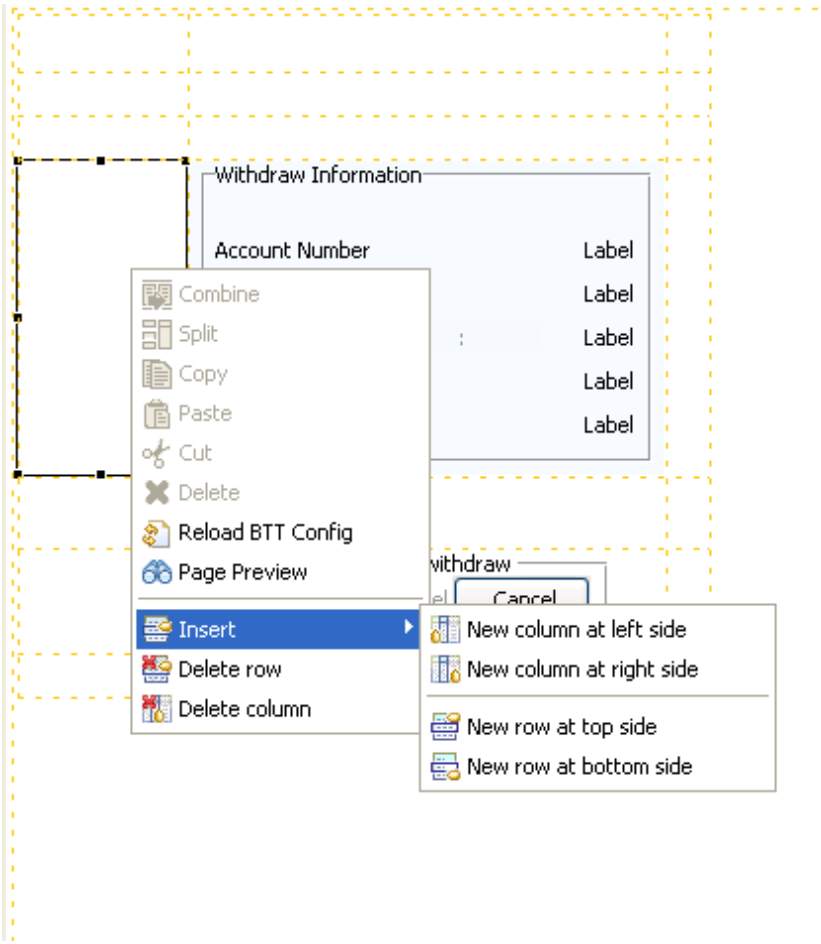
   Note: the HiddenField widget is the only one that cannot be dropped to a container in the XUI editor area. It must be selected (left-click) and then added to a container (left-click) in the XUI outline view.

   The widgets and containers are automatically aligned in the view and their position is always set relative the widgets already part of the view. Then, the new widgets can be placed on the top, the bottom, the right or the left of an existing widget if there is a free space (the final place is decided from the cursor position when you drop the widget to the view) and then a placeholder is generated in this place to keep the widget.

   The placeholders are identified in the XUI editor area as the cells of a dashed-lined table, as shown in the figure. The cells can be empty or already filled with a widget or container.

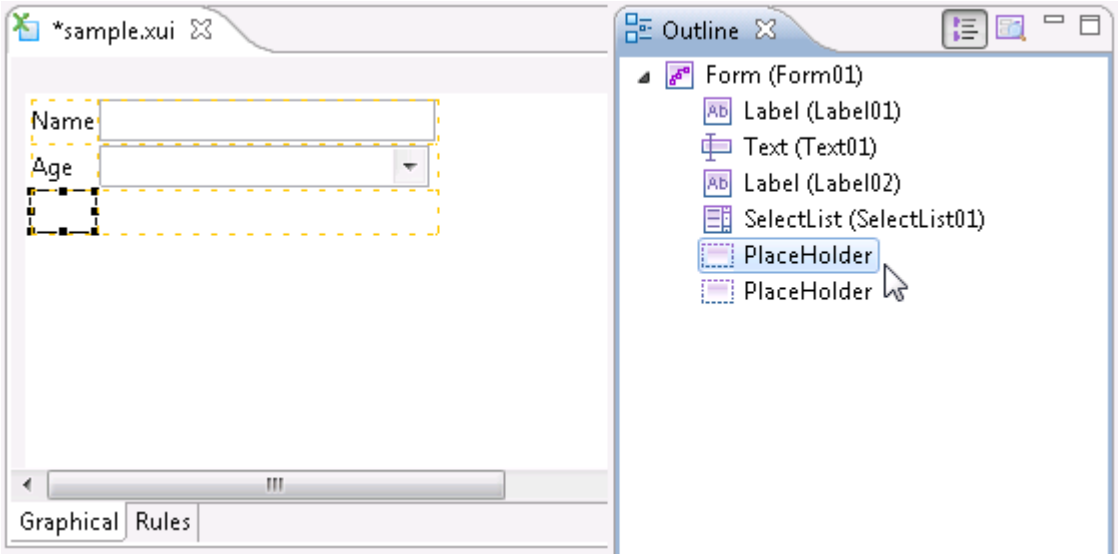


   Another way to generate a placeholder is by using the context menu. The context menu appears by right-clicking in the XUI editor area. By right-clicking in any of the existing placeholders or widgets and selecting where you want to place the new column or row of placeholders relative to the selected item: left side, right side, top side or bottom side, as shown in the figure below:



4. Optional: If you want to delete a widget or a container element, right-click the element that you want to delete in the XUI editor area, and then click Delete. If you want to delete more than one widget or container element at the same time, press Shift and then select the elements that you want to delete. Right-click the selected elements, and then click Delete.

   Note: After you delete a widget, the cell that contained the widget becomes empty but the placeholder is left. Placeholders are displayed in the Outline view and can be use to place new widgets or containers.



If you want to delete a placeholder, you can use the XUI editor context menu. Right-click the placeholder in the XUI editor area, and then click Delete. You can also delete a full row or column of placeholders by right-clicking on any of the items in the column or row and selecting the menu option Delete column or Delete row.

Note: when deleting a column or a row not only empty placeholders are deleted but also any widget or container in the same column or row.

Another way to delete a widget or a placeholder is using the Outline view context menu. Right-click on the widget or placeholder and click Delete.

For a more detailed description on how to modify the layout of your view, go to section XUI Editor Layout Configuration

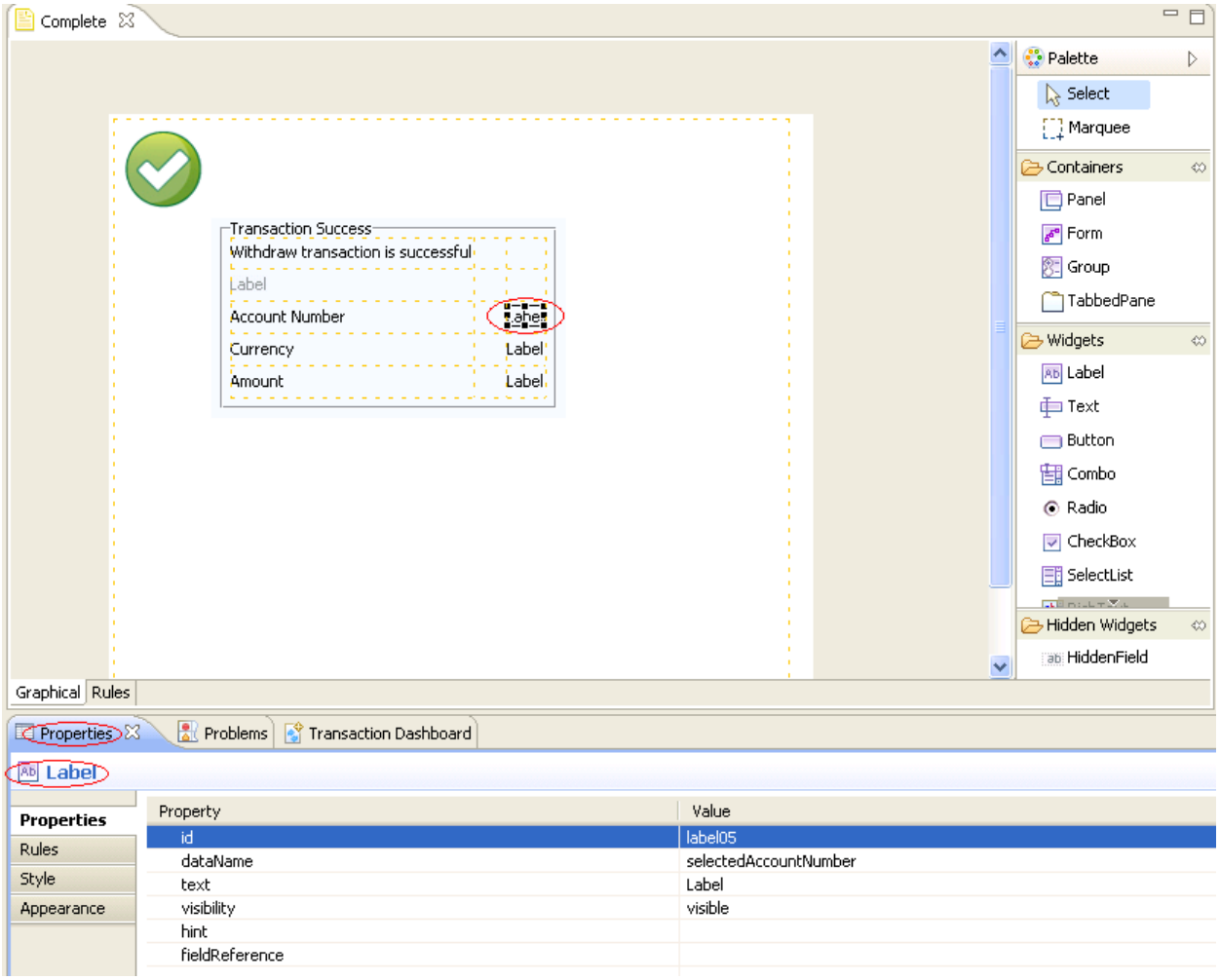
5.3.2 Editing the properties of the view widgets and containers

All the widgets provided by WebSphere Multichannel Bank Transformation Toolkit have a set of properties that are shown in the XUI Editor Properties area and that are organized, as shown in *Figure 1*, in different tabs:

- **Properties**
- **Rules**
- **Style**
- **Appearance**

The **Action** tab is also available in the Properties view for the Link, Button, and Form widgets.
The **Columns** tab is also available in the Properties view for the Table widget. The **Pagination** tab is available for a Table with pagination (isPageable=true).
The **Tabs** tab is also available in the Properties view for the TabbedPane container.
The **xValidations** tab is available in the Properties view for the Form container.

Figure 1: The Properties view of the Label widget



Properties

The Properties tab of the Properties view enables you to edit the properties of the widgets and containers.

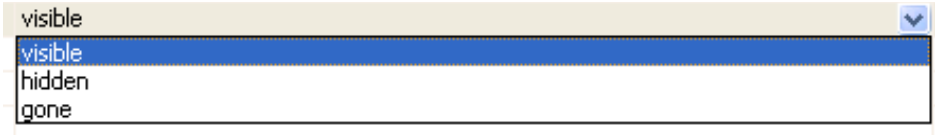
The properties listed in Table 1 are common for all BTT widgets unless otherwise stated. Specific widget properties will be described as part of the detailed explanation of the widget.

Table 1. Widget properties

Id	The id is the identifier for the widget.
dataName	The dataName property is the name of the data element to which the widget is bound. This property is optional. For a full description on how to bind data and context to the XUI widgets go to section Binding data to the XUI widgets

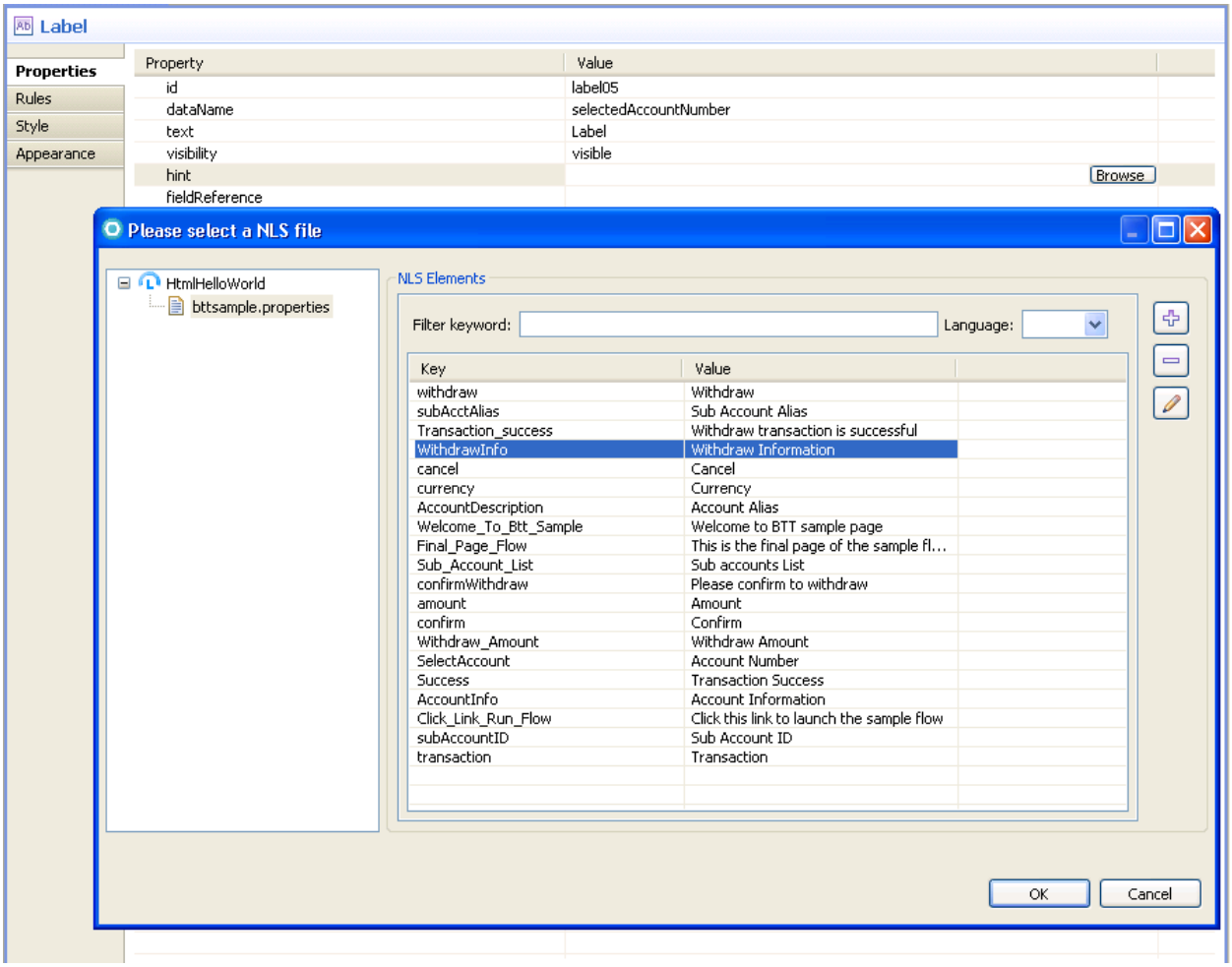
Table 1. Widget properties	
visibility	The visibility property defines how a widget is displayed on a GUI. The following values can be specified for this property: **visible** The widget is displayed on a GUI and users are able to interact with the widget. **hidden** The widget is not displayed on a GUI, but it occupies space on the GUI. **gone** The widget is not displayed on a GUI, and it does not occupy space on the GUI.
hint	The hint is a description of the widget that is displayed as a tooltip for a user. The hint property has multilingual support. For more information about multilingual support, go to section NLS support

These properties are set as pairs of property and value. The values are either directly input by the developer **(**id property is an example), selected from a drop-down list as for the Visibility property



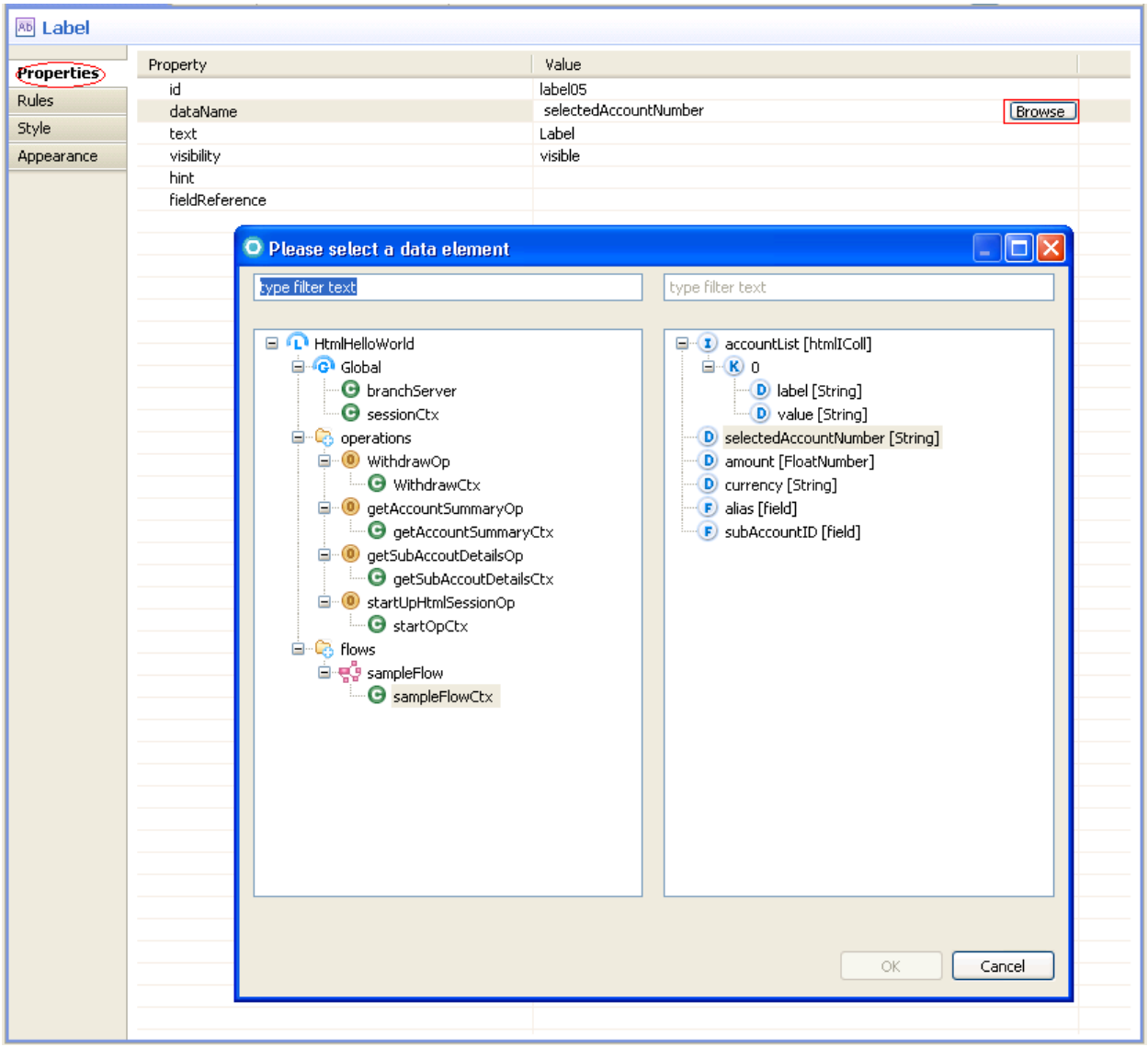
or have a Browse button that launches a wizard helping you to select a valid value, for instance a data element from the transaction context or a text from an existing NLS file. For the Hint property, the NLS support wizard open, as shown in next figure, *Figure 2* (for more details about multilingual support, go to section NLS support)

Figure 2. Setting the Hint property. When clicking the browse button, the NLS support wizard is displayed.



For the DataName property, the data mapping wizard opens, as shown in the figure below, *Figure 3*. For more details about how data and context are mapped to XUI widgets go to section Binding data to the XUI widgets

Figure 3. Setting a DataName property. When clicking the browse button, the data selection wizard is displayed.



For a detailed description on the properties of each widget and container, go to section Widgets Description or Containers Description.

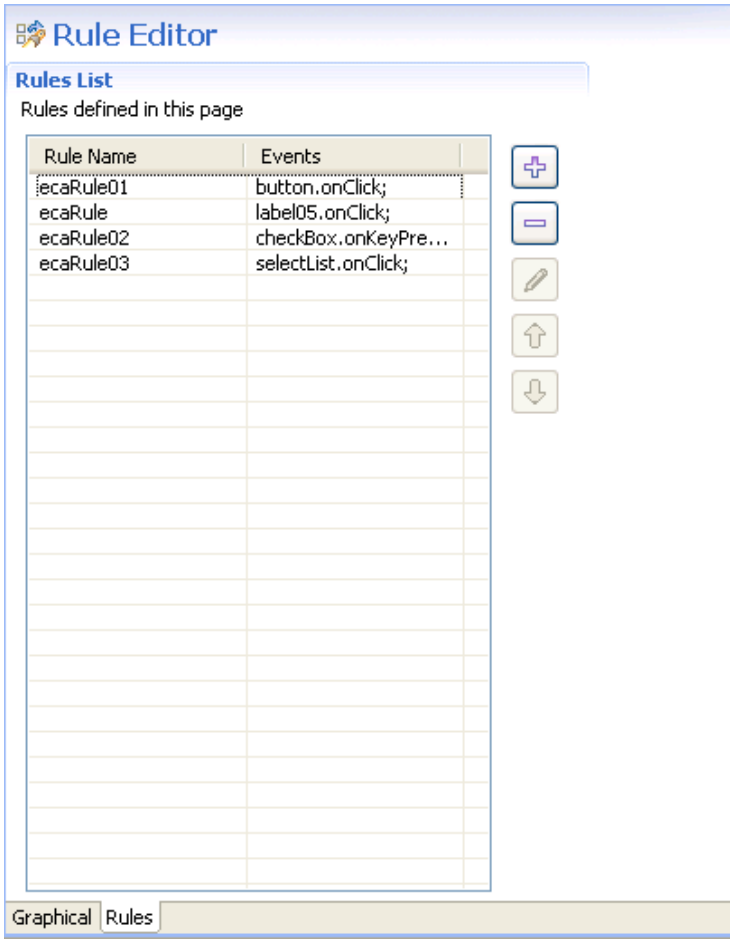
Rules

The ECA tooling (events, conditions and actions) enables a functional developer to handle view events visually. In the Rules tab of the widget properties, new rules can be added that detect a widget related event, evaluates a condition and takes an action. *Figure 4* shows the wizard to create these rules are defined.

Figure 4. Rules definition wizard.

Rules can also be defined by going to the Rules Editor, selecting Rules tab in the XUI editor area as shown in *Figure 5*. By using this tab, you can see in one shot all defined rules for your view and you are able to add or delete rules associated to any of the widgets and containers that are already part of your view.

Figure 5. Viewing all defined rules for your view using the Rules Editor in the XUI editor area



For more information on how to configure the events, conditions, and actions for widgets, refer to the Setting ECA rules to widgets section.

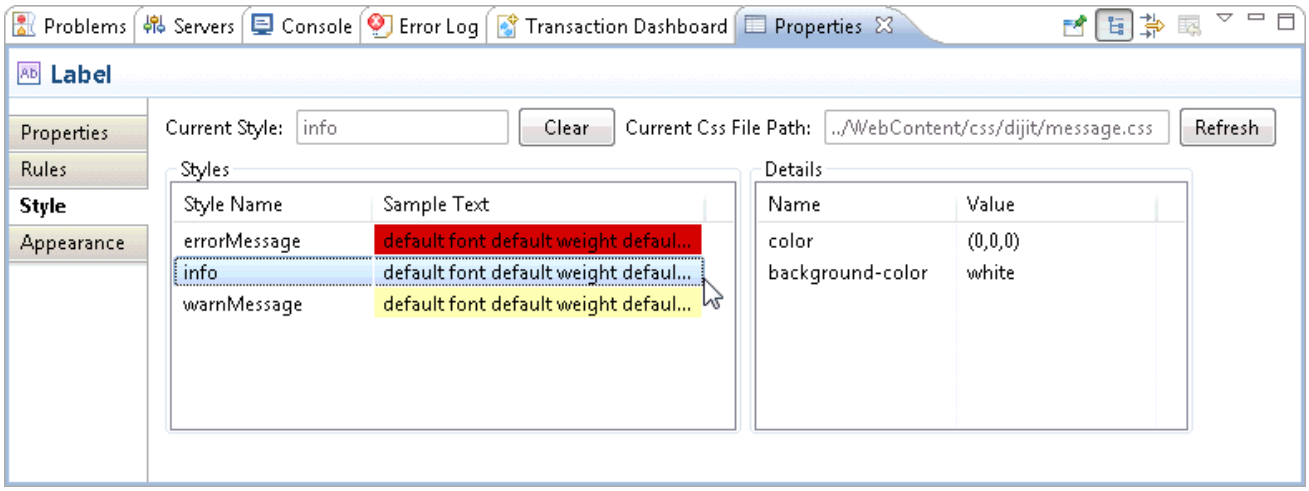
For a detailed description on the events, widget functions and widgets properties that can be used to define the rules using the ECA tool properties of each widget and container, go to section Widgets Description or Containers Description.

Style

The Style tab of the Properties view enables you to edit the style of the widgets. Before you apply CSS styles to a widget, you must first import CSS files to the XUI editor. For information on how to import CSS files to the XUI editor, refer to the Setting CSS style to widgets topic.

As shown in *Figure 6*, to apply a CSS style to a widget, click a style in the Styles panel of the Style tab.

Figure 6. Applying a CSS style to a widget.



The following CSS styles are supported:

- Font size
- Font style
- Font weight
- Text align
- Color
- Font family
- Background color

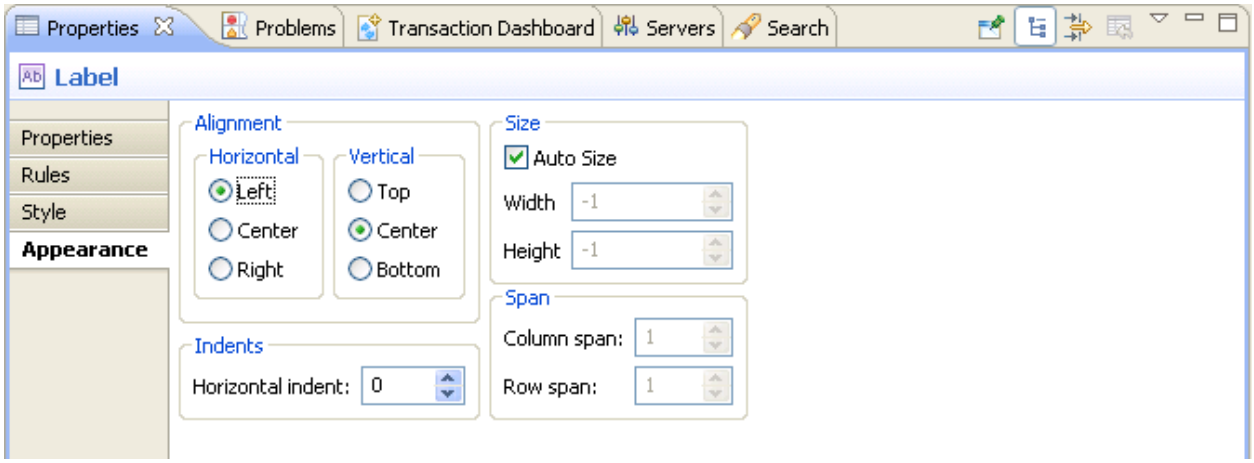
If you click on the link Current CSS file path: in the Style tab, you will get more details on the style you have selected.

Appearance

The Appearance tab of the Properties view enables you to edit the alignment, indentation, size, and span of a widget.

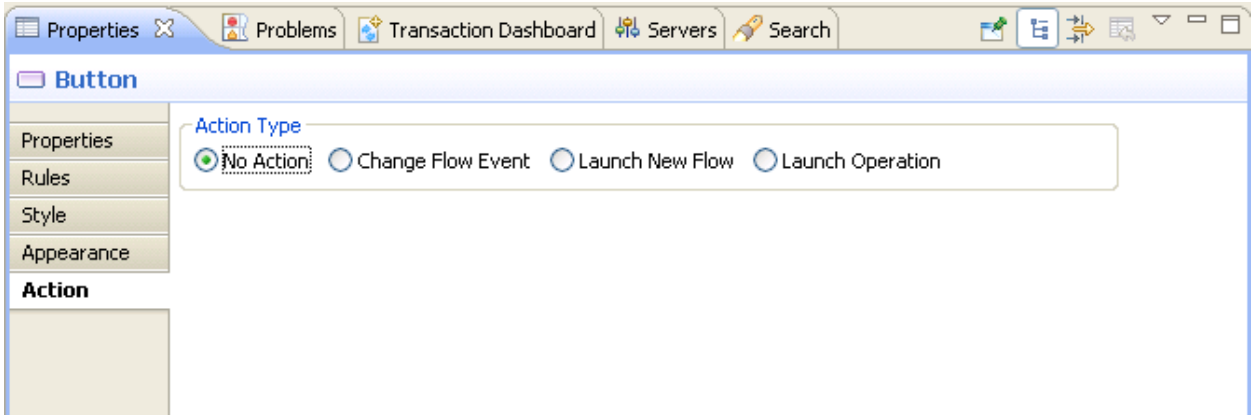
Note:

- To edit the size of a widget, you must first clear the Auto Size checkbox.
- You can also edit the size of a widget by dragging the boundaries of the widget in the XUI editor area. To edit the size of a widget by dragging the boundaries of the widget in the XUI editor, you must first clear the Auto Size checkbox.



Action

The Action tab is available in the Properties view for the Link, Button, and Form widgets.



The following options can be selected in the Action tab for a widget:

- No Action

  No action is initiated after events have been fired and conditions have been evaluated.

- Change Flow Event

This option will allow you to change the flow processor, and select a new event.

- Launch New Flow

  A new flow is launched if the conditions that you configure have been satisfied.

- Launch Operation

  A new operation is launched if the conditions that you configure have been satisfied.

- Launch URL

  A URL is launched if the conditions that you configure have been satisfied.

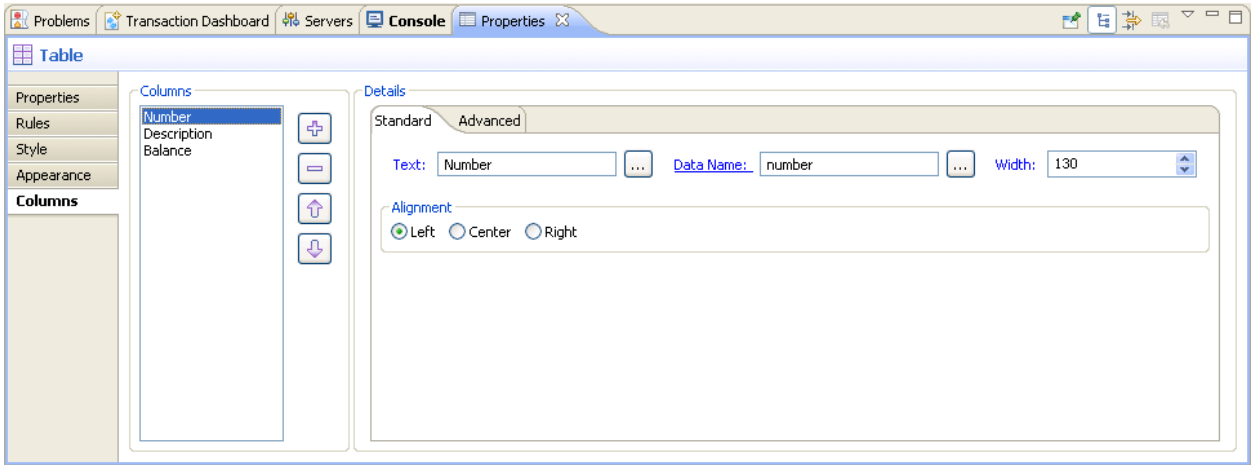
  Note: The Launch URL option is available only for the Link widget.

Columns

The Columns tab is available in the Properties view for the Table widget. Once a new table column is added, you can set the column properties, that are organized in two different tabs:

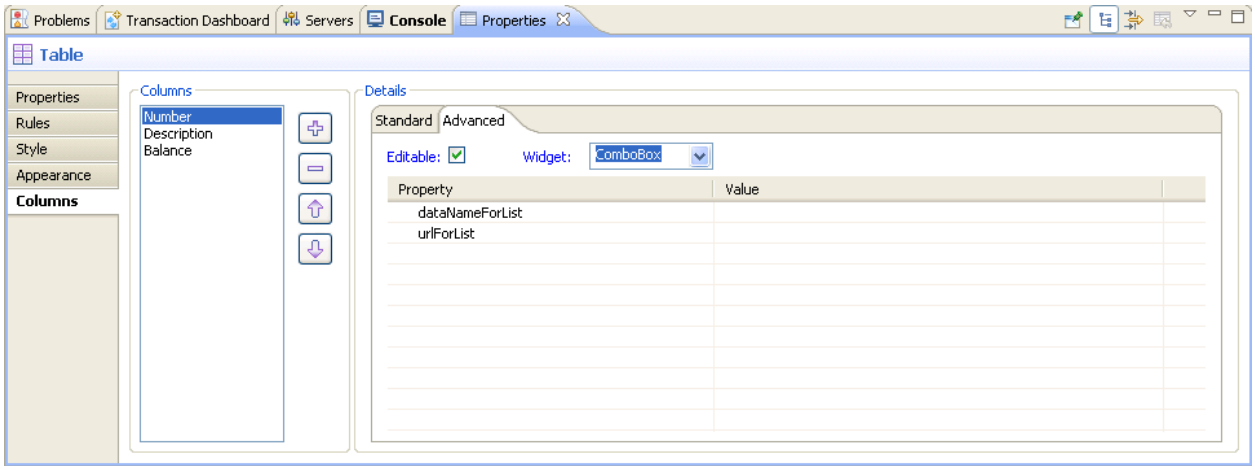
a. In Standard properties tab you can set some basic and common attributes:

- Text: the text field contains the label that will identify your column and will be use as the columns title. It does support Multilanguage.
- Data Name: this field binds the columns context to the data in the transaction context. In order to be able to select a Data Name, the dataNameForList must have been set in the Table properties. For more details about how to bind data to a table widget, refer to section Binding data to a table widget
- Width: the width of the column
- Alignment: how the content of the column will be aligned inside the column cells



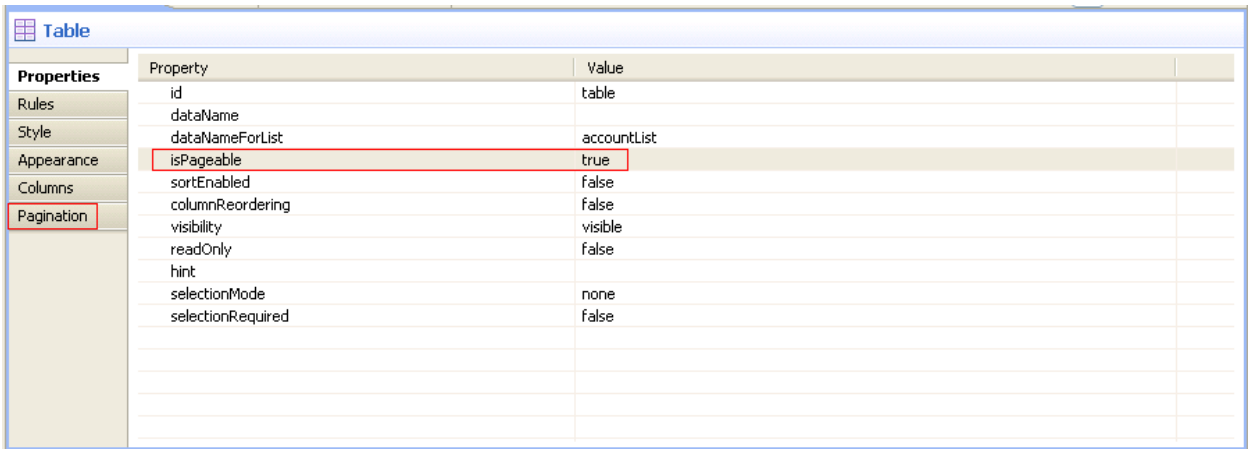
b. In Advanced properties tab, you can set the properties associated with the widget selected for the table column:

- Editable check box: it specifies whether you want the widget selected for the column to be an editable or a non-editable widget
- Widget: the widget that will be used as a container for the column value. If the Editable check box is checked, possible values shown in the drop-down list are: TextBox, CheckBox, ComboBox or SelectList; if the Editable check box is unchecked, then the possible values for the widget are: Image or Label
- Property table: contains the properties associated to the selected widget

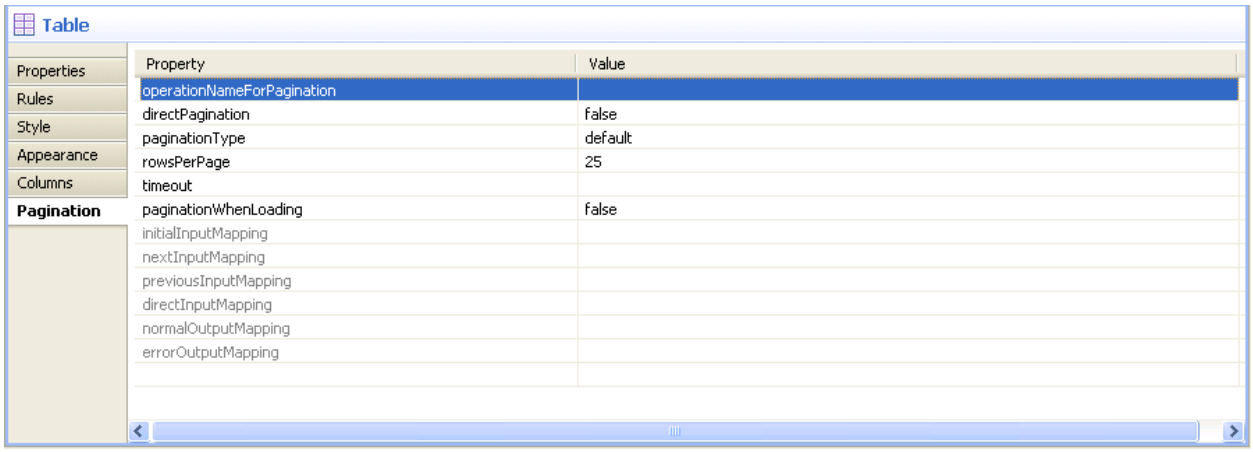


Pagination

The Pagination tab is available in the Properties view for the Table widget when pagination has been selected, as shown in the figure below:



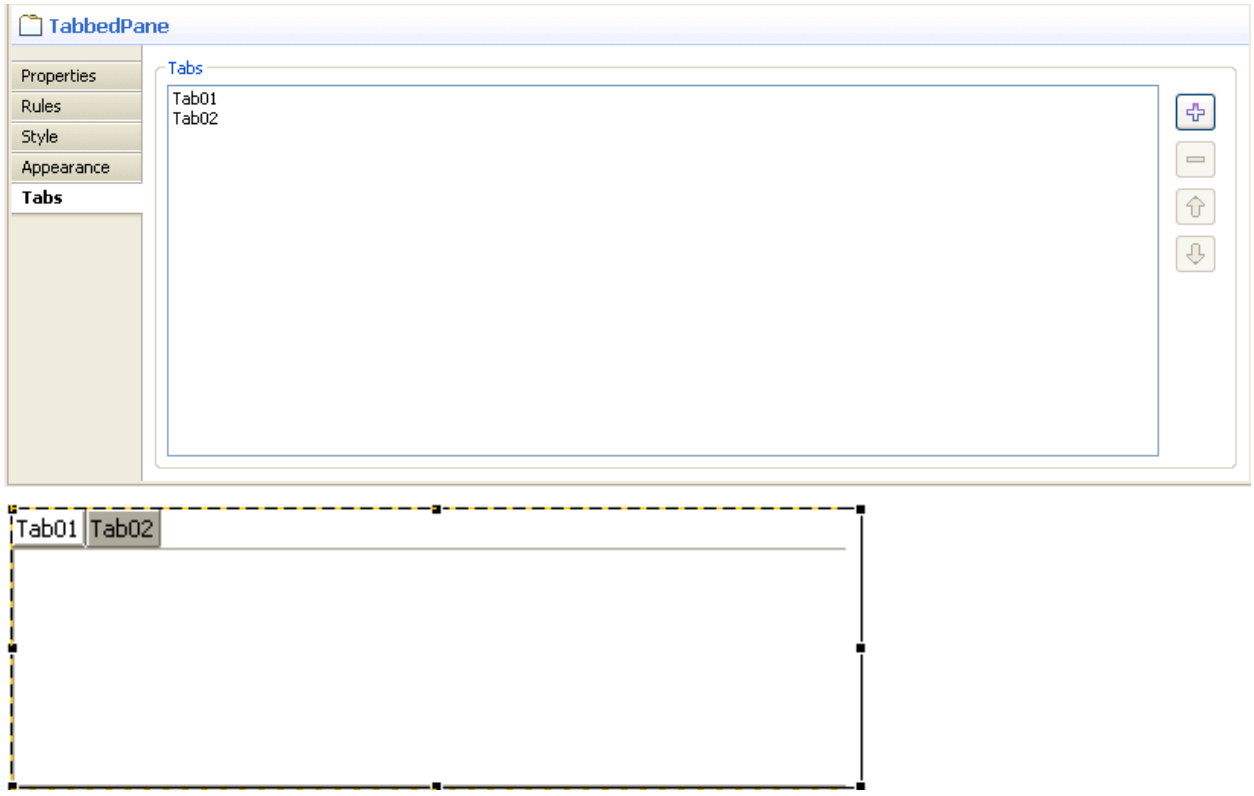
For a description of the properties, refer to section Table**.** For a full explanation on how to create, configure and use paginated tables, go to section Adding a paginated table widget sample



Tabs

The Tabs tab is only available in the Properties view for the TabbedPane widget. It allows you to add and reorder tabs to the pane.

Figure 7. Using Tabs to add tabs to the pane and how the pane is shown in the XUI edit area .

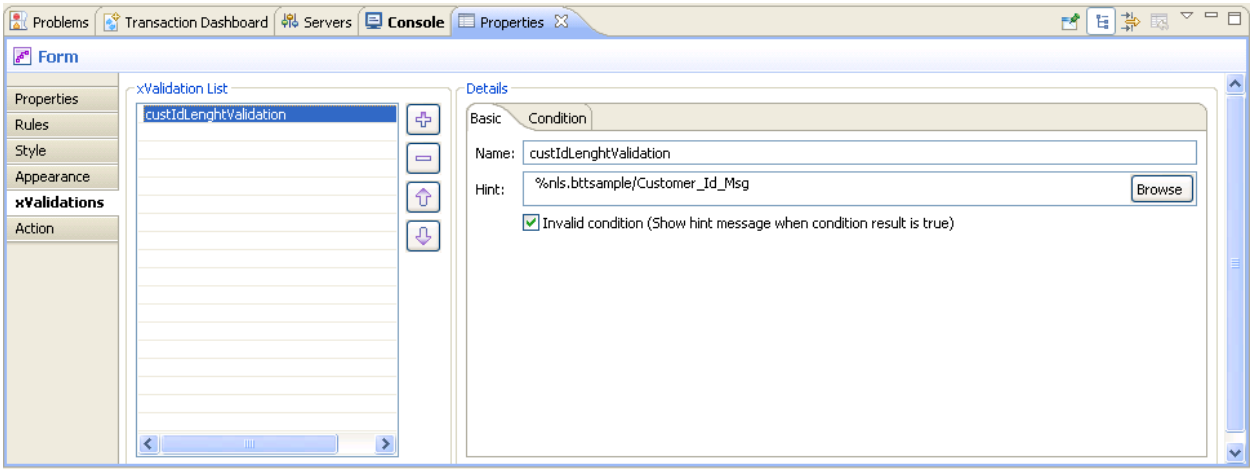


xValidations

Functional developers can define cross field validation (xValidation) rules on a form. A cross field validation rule includes:

- A condition, which determines whether a combination of multiple field contents is valid or not.
- A hint message, which will be shown when the combination of multiple field contents is not valid.

When the 'validateOnSubmit' property of a form is defined as true (refer to **Form** section for details on the container properties), any submit button of the form will be disabled until all the form fields and the cross validation are valid.



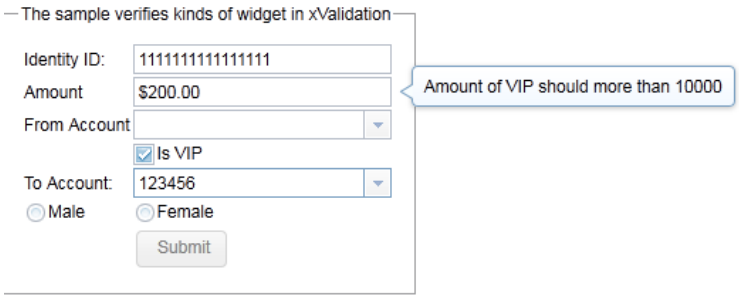
The hint message is shown when any field in a form is changed (lost focus) and no individual validation message need to be shown. The users can know how to correct their input.

If multiple cross field validation rules are not valid, only one hint message of a rule will be selected and shown on form. The message of a rule is selected according to the following criteria:

- if the last changed field belongs to a cross field validation rule and all other mandatory fields in the rule have their values set, the hint message of this rule is selected
- if the previous condition does not occur, then the message of the first invalid rule according to definition sequence is selected.

The selected hint message of the rule will be shown around one of fields which the rule contains. The field is selected according to*:*

- if the last changed field belongs to the rule being evaluated, then the field will be selected.
- if the last changed field does not belong to the rule, then the first field in the rule being evaluated will be selected.



5.3.3 Containers Description

An XUI container is an object that is used to contain widgets in View. This section provides a description of the containers that are provided by IBM® WebSphere® Multichannel Bank Transformation Toolkit.

Each View can have only one container as its root container to hold other widgets or sub-containers. The following four containers are provided by WebSphere Multichannel Bank Transformation Toolkit:

- **Panel**
  The Panel container is used to place various widgets within several panels in one View. However, developers usually do not use panels as the root container of the View. Instead, developers often put a panel in other containers. The only property that can be set for a Panel container is its id and the width and height using the Appearance tab. No events are defined for this type of container.
- **Form**
  The Form container is used to fill with other widgets and submit data. For example, developers usually place widgets like text, label and combo box in a form container. These widgets are often bound to predefined data within the transaction.
- **Group**
  The Group container has a content model which allows data to be grouped together. The Group functions like the menu, because the user can use the Group container to classify different types of data.
- **TabbedPane**
  The TabbedPane container is complex container that has multiple panes (a **ContentPane** per each tab) and can be directly dragged into XUI as the root container, but shows only one pane at a time. It can be different from other containers in that it can be separated into two widgets.

5.3.4 Widgets Description

This section contains the description of the widgets provided by BTT. You can get more detailed information following the link in each widget name. There you will find full description of the widget properties (Properties section), how the widget can be bound to a data element or a context (Data mapping section), and the events, properties and functions that can be configured as part of a rule definition for the widget (ECI editor support section)

- **Label**
  The Label widget displays text that cannot be edited by a user, and it is enabled for multilingual support.

- **Text**
  The Text widget is used to enable users to enter and submit a string of text, or a number, or a date, or a currency. You can bind a Text widget with typed data elements.

- **Button**
  The Button widget is used to invoke an event or action when it is clicked.

- **Combo**
  The Combo widget allows a user to either enter a value or to select an item from a list of choices. The labels and the choices that are available on the ComboBox widget for a user to select are retrieved from a WebSphere Multichannel Bank Transformation Toolkit context or from an external source, for example, a web service. A change in the choice that has been selected by a user in a Combo widget can be detected by the other widgets on the page.

- **Radio**
  A Radio widget is displayed as a circle with text beside it that indicates a fixed set of choices from which only one can be selected. All the radio buttons in a list share the same data, and the change in status of a RadioButton widget can be detected by the other widgets on the page.

- **CheckBox**

  The CheckBox widget allows a user to either check or uncheck an option. A data element is associated to this widget and you can decide which value will be set for this data element when either the box is checked or unchecked.

- **SelectList**
  The SelectList widget allows a user to select an item from a list of choices.

- **RichText**
  The RichText widget allows a user to enter rich text.

- **TextArea**
  The TextArea widget enables users to enter and submit multiples lines of text.

- **Image**
  The Image widget is used to display images. You can also provide a title for the image by specifying a value for the hint property.

- **Link**
  The Link widget is used to initiate an operation, processor, or to invoke an action.

- **Table**
  The Table widget provides the ability to display data in a table.

- **Tree**
  The Tree widget provides the ability to display multiples lines of items in a tabular format.

- **Message**
  The Message widget has the ability to display error messages, warning messages, and hints. Messages and hints are displayed on a Web page and can also be displayed as a pop-up window. Multilingual support is provided for the messages and hints that are displayed on a GUI.

- **FileUpload**
  The FileUpload widget provides the ability to select and upload files to the server-side. It is a composite widget, which means that it consists of a collection of widgets.

- **HiddenField**
  The HiddenField widget is a hidden widget, which means that it is not displayed on a GUI. However, the HiddenField widget contains data that are submitted to a WebSphere Multichannel Bank Transformation Toolkit context and that are retrieved from the context.

5.3.5 XUI editor layout configuration

You can use either of the following two methods to change the layout of widgets in the Views:

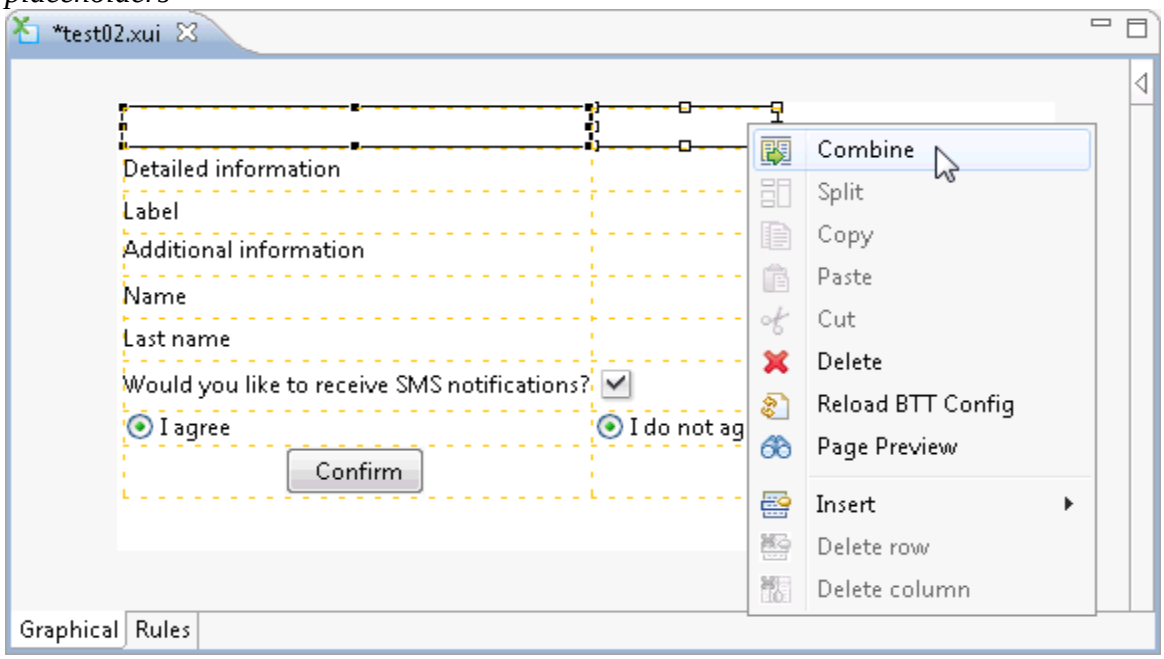
- By using the context menu
- By using the Outline view

Using the context menu to change the layout of widgets

The context menu displays when you right-click the XUI editor area.

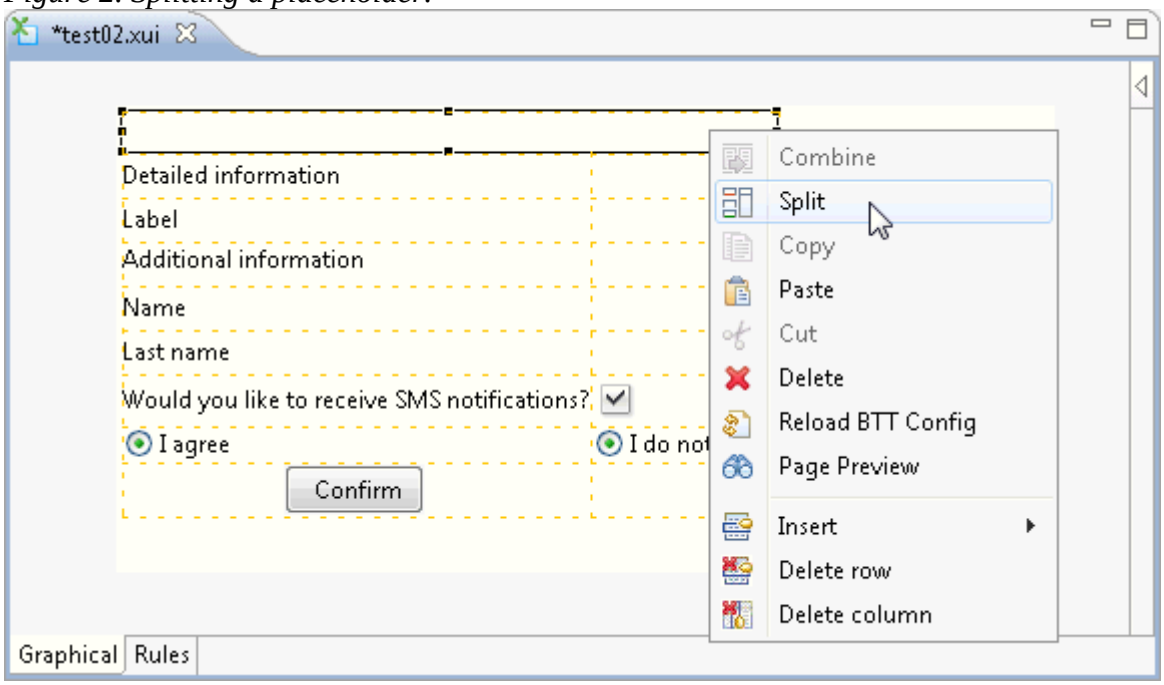
As shown in *Figure 1*, the Combine function in the context menu enables you to merge placeholders. If you want to merge placeholders, press Shift and select the placeholders that you want to merge. Right-click the selected placeholders, and then click Combine.

Figure 1. Using the Combine function to merge placeholders



The Split function in the context menu enables you to split a placeholder. If you want to split a placeholder, right-click the placeholder that you want to split, and then click Split.

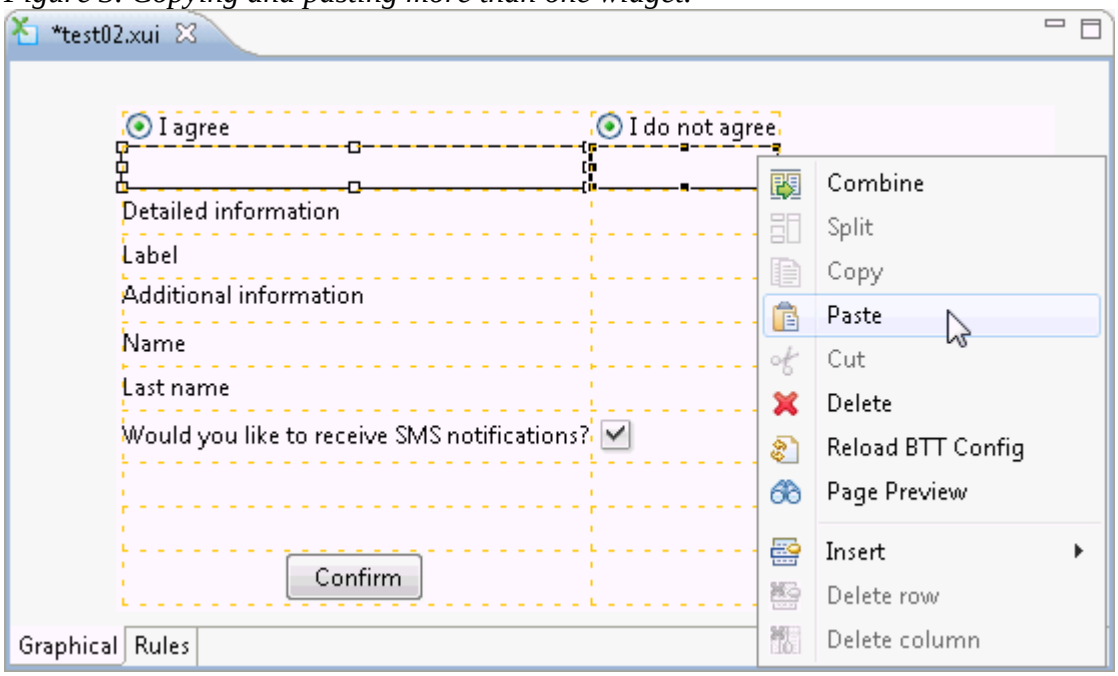
Figure 2. Splitting a placeholder.



When you copy or cut and paste a widget, all the properties of the widget are also copied or cut and pasted. Note that a widget can be pasted only to a placeholder; it cannot be pasted outside a placeholder.

You can copy or cut and paste more than one widget at a time; however, if more than one widget is copy or cut, they can be pasted only to placeholders with the same structure. For example, as shown in *Figure 3*, if you want to copy and paste two widgets that are adjacent to each other, you can paste the two widgets only to placeholders that are also adjacent to each other.

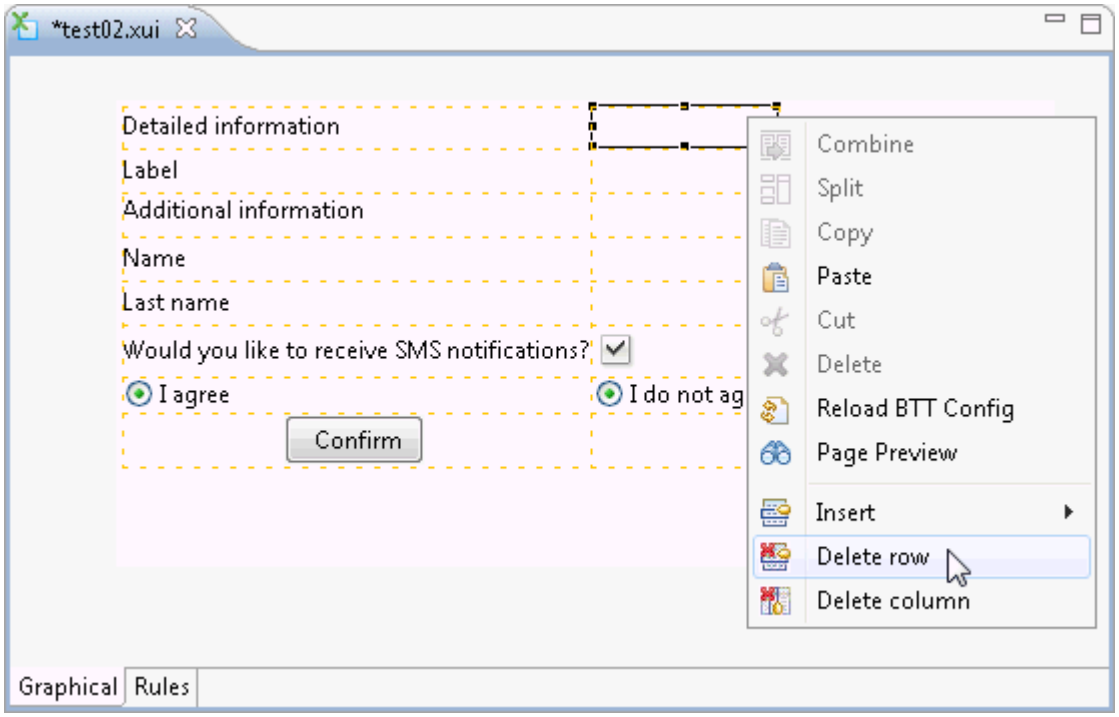
Figure 3. Copying and pasting more than one widget.



If you want to delete a widget or placeholder element, right-click the element and then click Delete. To delete more than one widget or placeholder element at a time, press Shift and select the elements that you want to delete. Right-click the selected elements, and then click Delete.

To delete an entire row or column, use the Delete row or the Delete column function in the context menu as shown in *Figure 4.* Note that when you click Delete row or Delete column, both cells that contain widgets and empty placeholders are deleted in the row or column.

Figure 4. Deleting a row by using the context menu



Using the Outline view to change the layout of widgets

As shown in *Figure 5*, you can use the context menu in the Outline view to delete widgets, and to copy, cut, and paste widgets to placeholders.

You can also change the position of widgets in a View file by dragging and dropping widgets to placeholders as shown in *Figure 6*.

To use a keyboard shortcut to copy and paste widgets, select the widget that you want to copy and press Ctrl+C, and then select the placeholder that you want to paste the widget to and press Ctrl+V.

Note: If you are using the Outline view to copy, cut, and paste widgets, the widgets can be pasted only to empty placeholders.

Figure 5. Copying, cutting, and pasting widgets to placeholders by using the context menu in the Outline view.

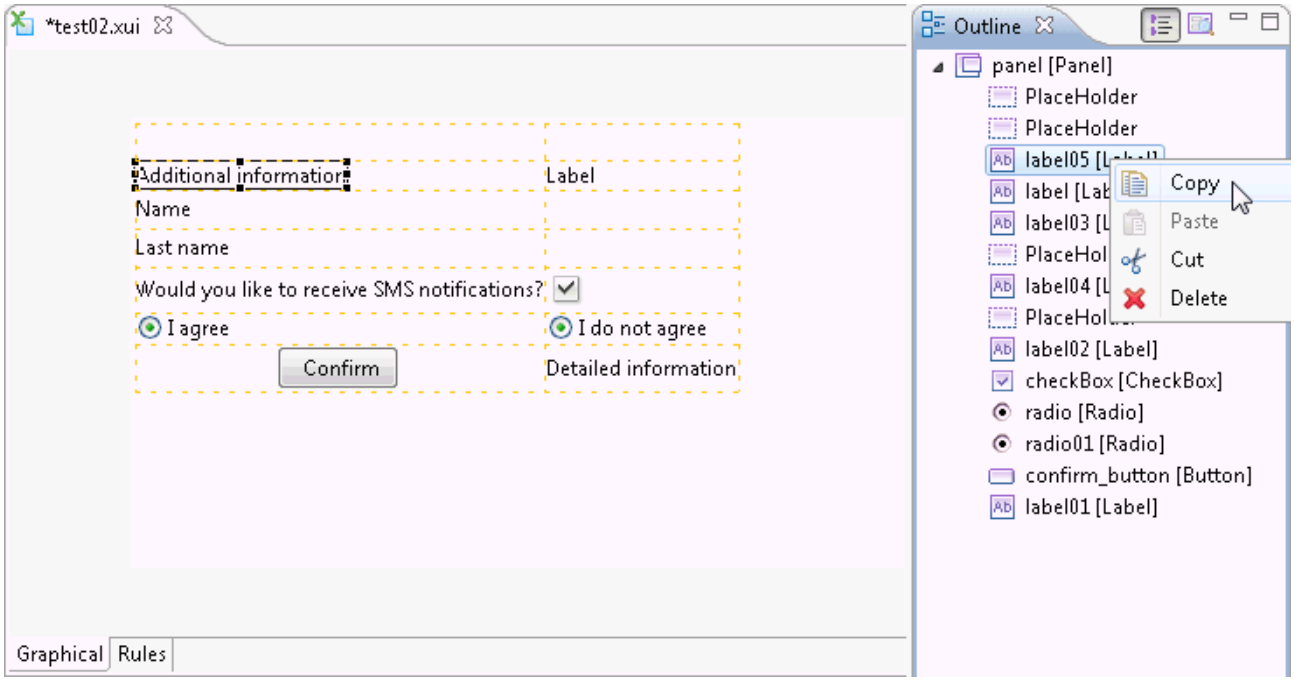
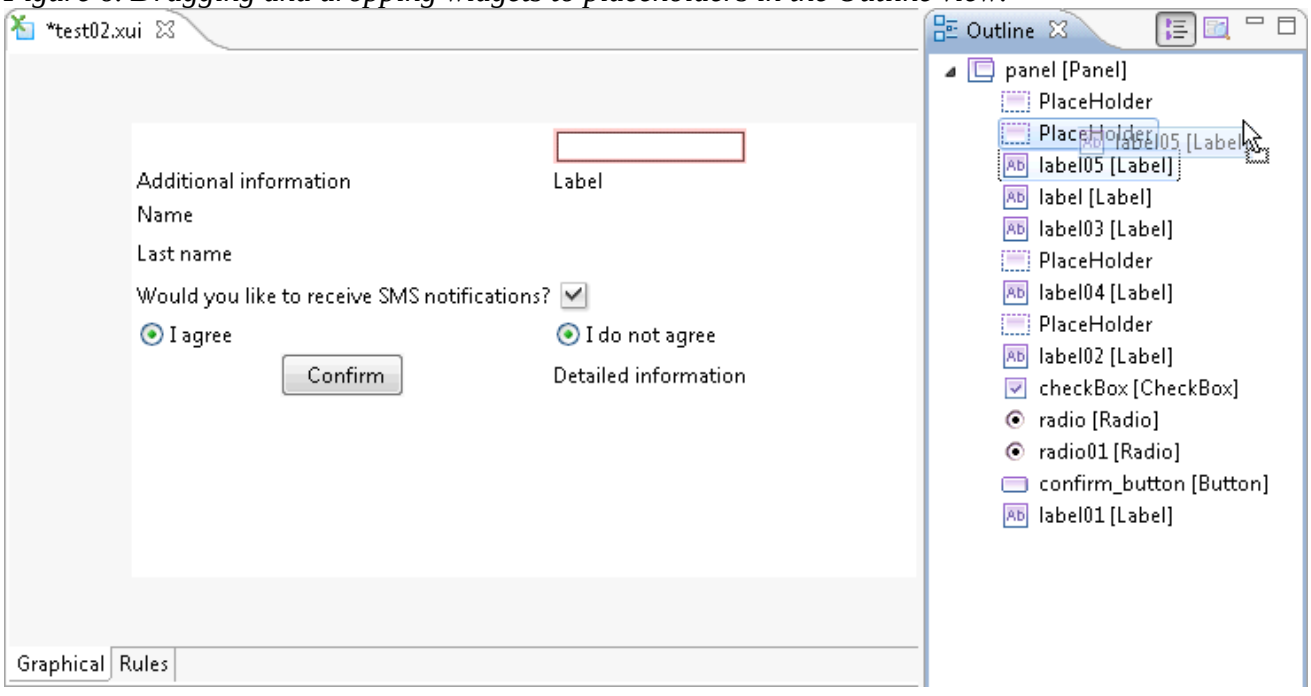


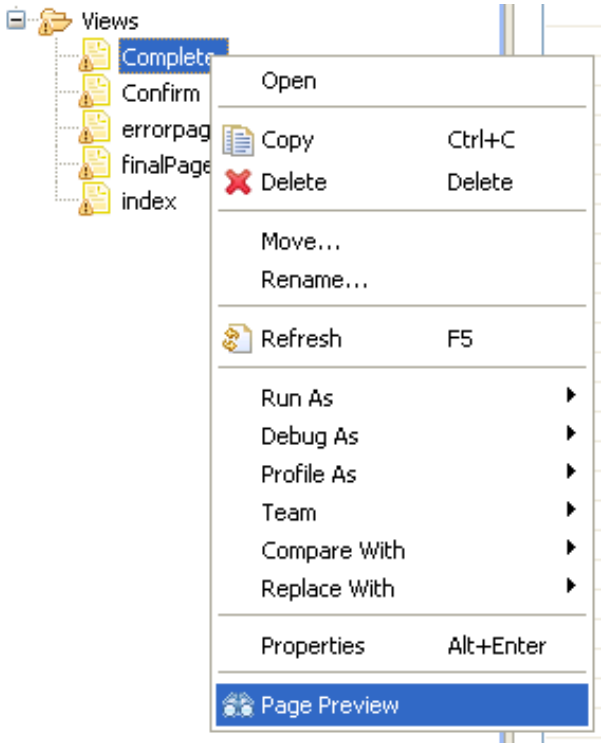
Figure 6. Dragging and dropping widgets to placeholders in the Outline view.



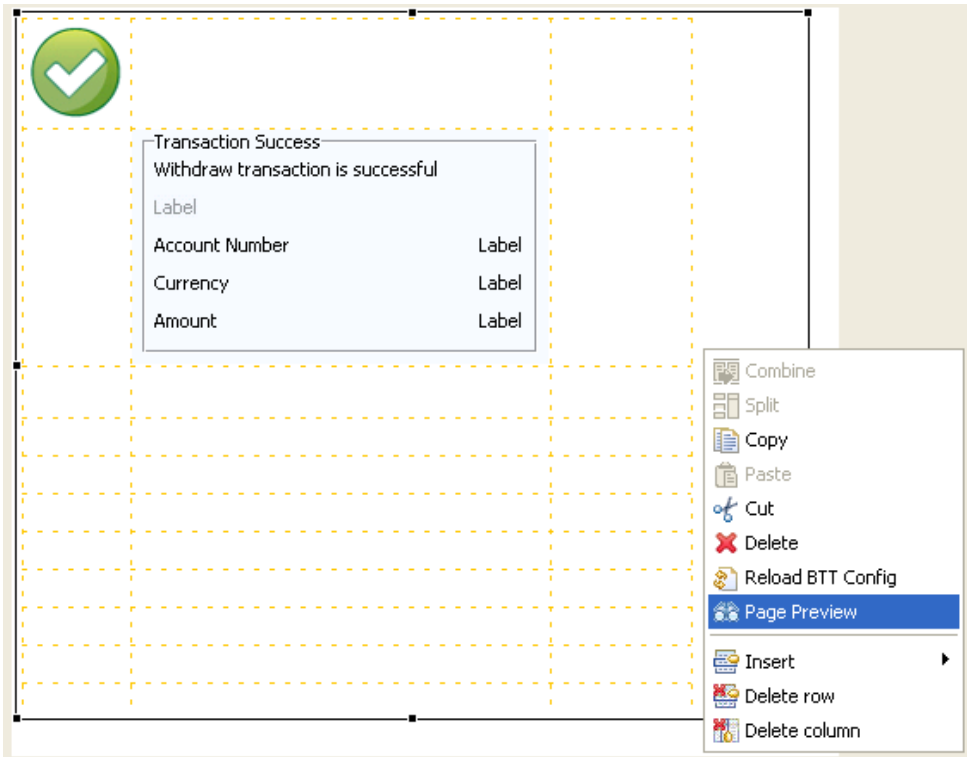
5.4 View Unit Test

Each time you make a change to your view and save it, the JSP and JS files that are going to be deployed to the runtime environment are automatically generated for you. After you complete the construction of the XUI user interface by using the XUI editor, then you may want to test it and verify that the layout, the rules and the input validations you have defined work as expected. Only the ECA rules and input validations that do not require server access can be tested. You can use either of the following two methods to test your view:

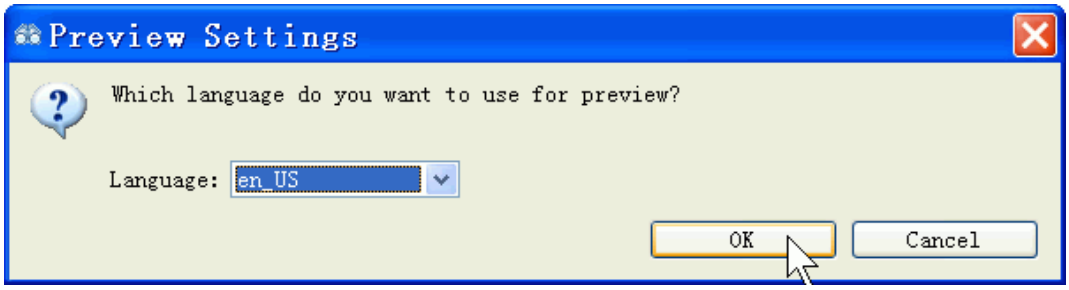
- in the BTT Perspective project explorer context menu, right-click on the view and select the Page Preview menu option.



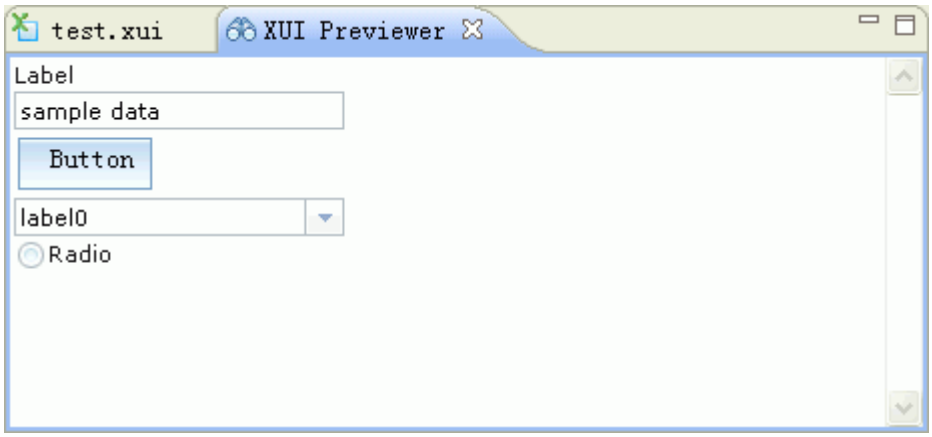
- open the view file using XUI editor, right-click in the blank space to show the context menu and select the Page Preview option..



The Preview Settings window is displayed. In the Language list of the Preview Settings window, select the language that you want to use to preview the XUI page. Click OK.



The view is displayed in the web browser.



Note: The view is displayed in the web browser that has been selected to be used by default when web pages are opened. This is part of the development environment configuration and must be done by a technical developer.

Use the debugging and tracing capabilities provided by BTT to detect and correct possible errors. Go to Client Functional Traces section to understand how tracing capabilities are configured and used in the development environment to test the views.

5.5 Using View Templates

The view templates are built by the technical developers (refer to section Saving a XUI file as a template) and are used by the functional developers as skeletons for the development of transaction views. Normally the templates do only contain common widgets among the views, as title and action buttons, and they do not have, as part of their definition, neither a context nor data to widgets bindings.

If the template is used when generating a view from an operation or flow context then the operation or flow context becomes the view context. If the template context is not empty and if there are already some data mappings defined, the data mapping to UI widgets will have to be redone using the new generated view context data.

Note: it is important to understand how this automatic view generation from a context using a template works:

- the template needs to have a Form container labeled mainForm.
- the mainForm container must be empty. If not, the generation process will prompt an error message.
- all widgets created automatically and associated to the selected data in the operation or flow context will be generated by the wizard inside the mainForm container

6. Modeling flows with the transaction editor

The IBM® WebSphere® Multichannel Bank Transformation Toolkit Transaction editor enables developers to create WebSphere Multichannel Bank Transformation Toolkit transactions rapidly. It provides a graphical editor for creating and developing the WebSphere Multichannel Bank Transformation Toolkit transactions. The Transaction editor also generates the XML definition files of the transactions and the skeleton code of the transaction classes.

There are two types of transactions:

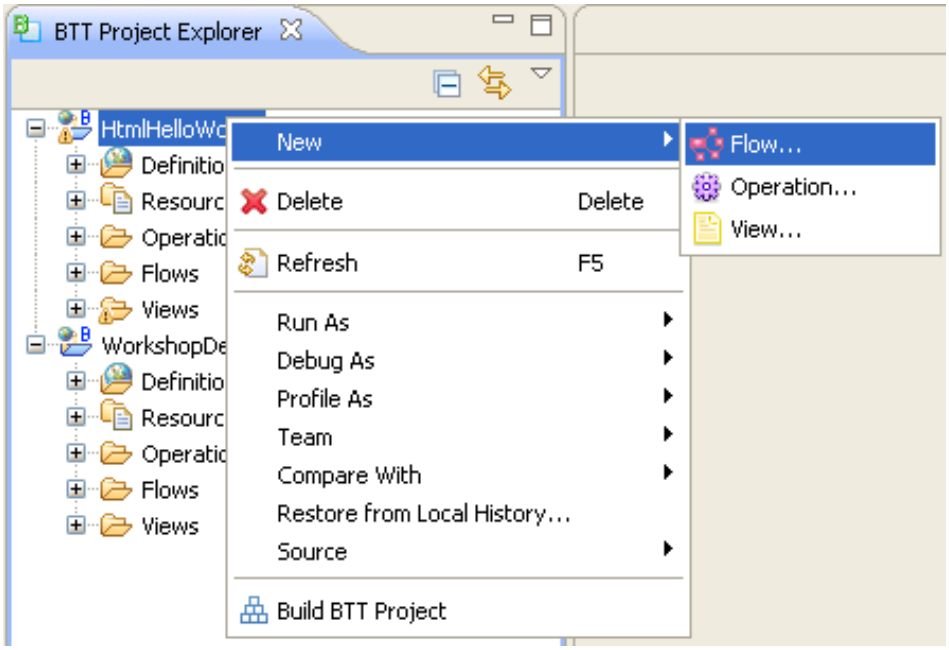
- BTT Operation, a specific task to be executed as part of a business process
- BTT Flow, a sequence of operations, views and transitions between them that describe a specific business process

The Transaction Editor will allow you to define either a BTT Flow or a BTT Operation as well as any other BTT objects that may be part of a flow or operation definition: context, data elements, formatters and service components.

6.1 Creating a flow in the BTT Perspective

To create a Flow, take the following steps:

1. Right-click either in your BTT Project or in the Flows folder inside your BTT Project. Click New > Flow…

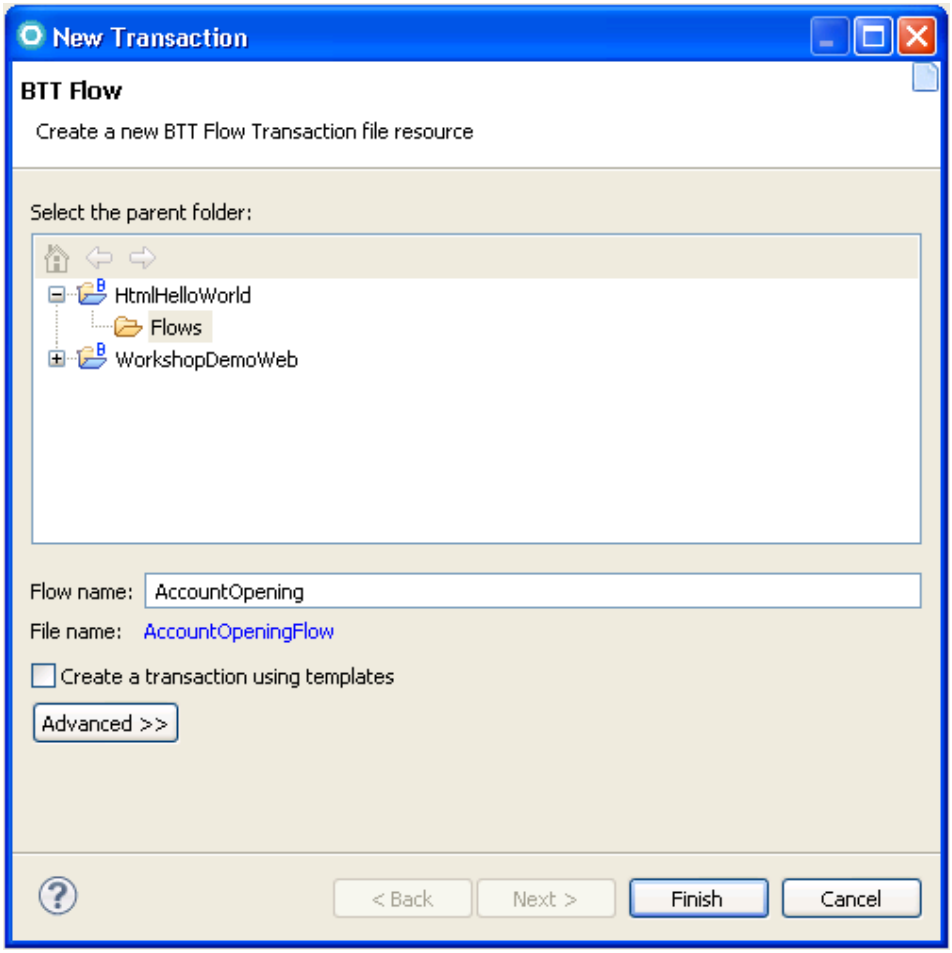


The BTT New Transaction wizard opens.

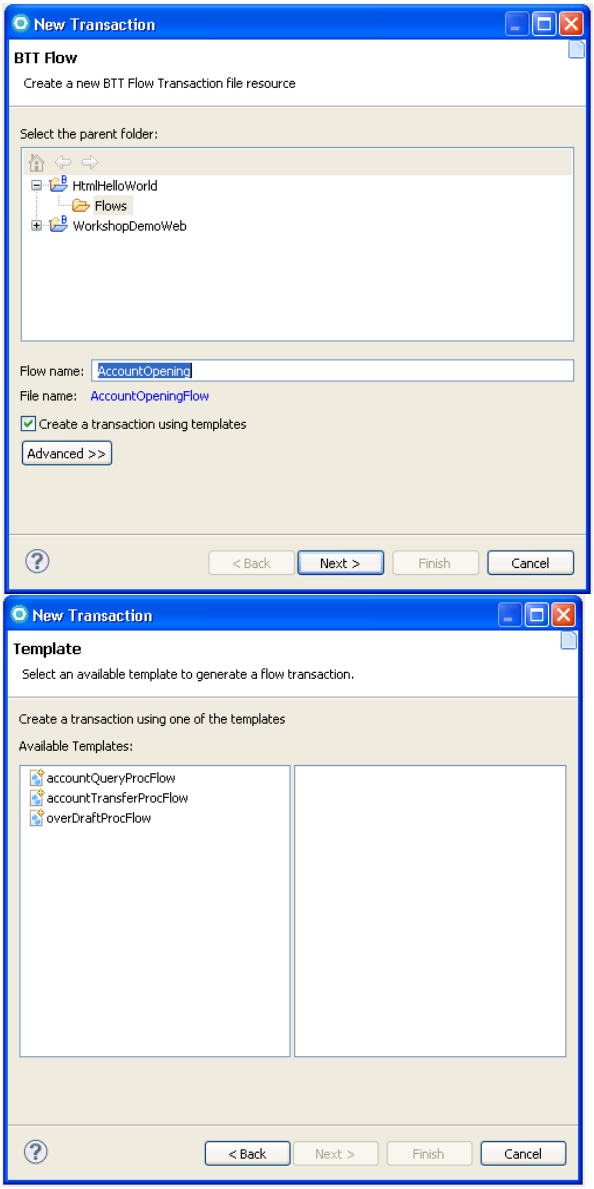
1. In the Flow name field of the New Transaction wizard main page, enter a name for the flow transaction file. Select the appropriate parent folder, which should be the Flows folder of the BTT project (it is already preselected for you). Click Finish.

   Note:

   - The flow name you specify in the Flow name field cannot be duplicated. You will be prompted with a message if a flow with the same name already exists
   - You must specify a flow name without extension. The flow name you enter will be appended with the string 'Flow' and will be internally created with a .transaction file extension ('sample' flow name will be converted, in the file system, to sampleFlow.transaction file). Final file name without extension is shown, while you are typing the flow name, in the File name field of the creation wizard.



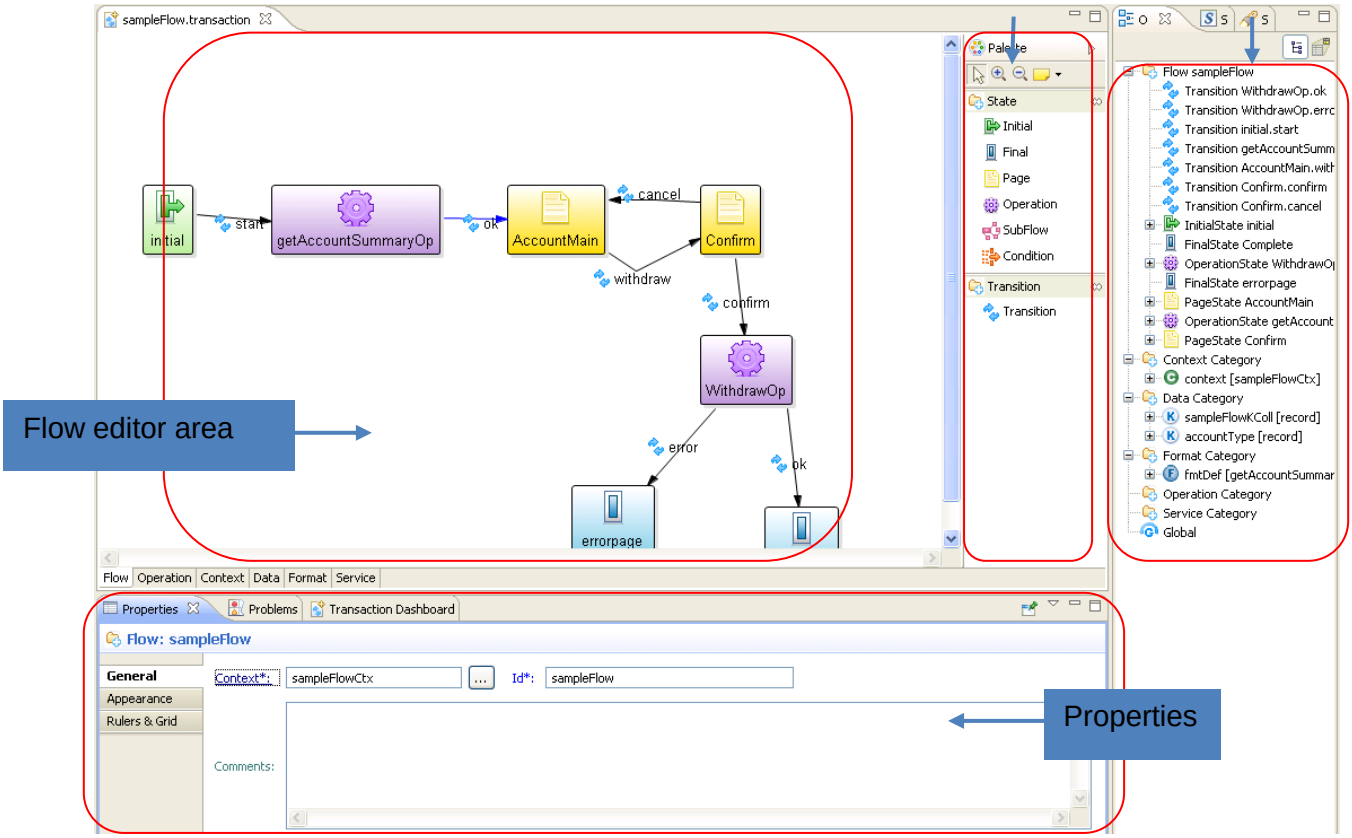
2. Optionally, you may want to create a Flow using an existing transaction template: a predefined transaction that can be reused. Then, select the **Create a transaction using templates** check box. Click Next, and then select the template you require. Click Finish.



Note: All the template files can be shared among different projects. The new Flow inherits all the properties of the original template and can be used as a starting point for your flow design. Refer to section Using Flow Templates for more detailed information.

The Flow Editor is now opened. It consists of the following four parts:

Palette Outline



Flow editor area

The Flow editor area (Flow tab in the Transaction editor) enables you to graphically design flows. You can select a state or transition from the Palette (left-click) and drop it in the editing area by left-clicking in the position you want it to appear.

If you click a state or transition element in the Flow editor area, you can edit the properties of the element in the Properties view.

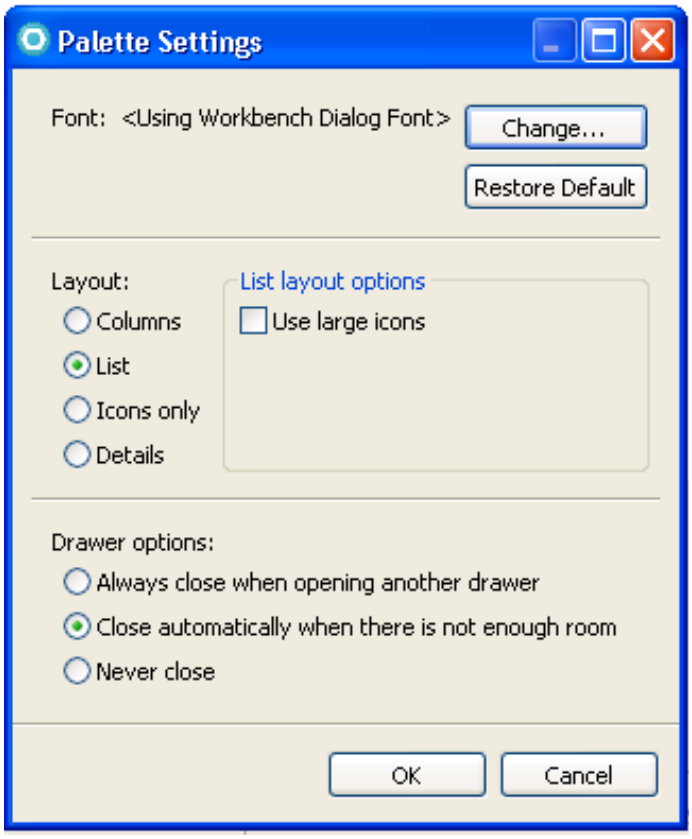
The other tabs in the Transaction editor, Operation, Context, Data, Format and Service are use to create new BTT objects associated with the flow. Follow the links to learn how to proceed to create new context, new data, new formatters or new services as part of the flow definition.

Palette

The Palette view contains the states and transitions that enable you to create a flow.

To add the Palette view to the current perspective, click Window > Show View > Other…> General> Palette.

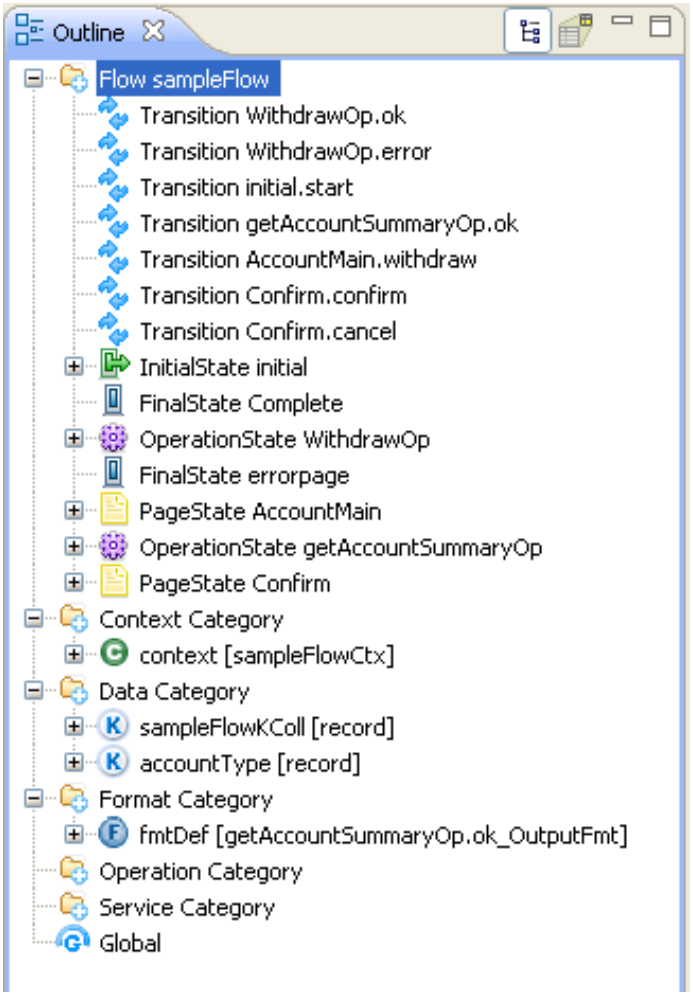
The Palette can be configured by right-clicking on it. You can decide how the elements are shown: organized by columns or in a list, only with icons or with a detailed explanation, select the font for the labels and change the drawer options.



For a full description of the different elements in the Palette that can be used to create a transaction flow, go to section

Outline

The Outline view displays an outline of the Flow that is currently open in the Flow editor area, and it lists the states and transitions elements of the Flow as well as information about the operations, context, data, formatters and services associated to the Flow.



If you click an element in the Outline view, you can edit the properties of the element in the Properties view. Clicking on the button for any of the available elements categories launches the corresponding element creation wizard in the Transaction editor area. Selecting one of the elements launches the corresponding element editor in the Transaction editor area.

To add the Outline view to the current perspective, click Window > Show View > Other... > General > Outline.

Properties

The Properties view displays the flow properties or the properties of the selected state or transition in the Flow editor area.
You can edit the properties of the elements of your Flow in the Properties view.
The Appearance tab in the Properties view allows changing the font and colors of the selected node or all the nodes in the flow.
The Ruler & Grid tab, only available when clicking in the blank area of the Flow editor, allows you to add a ruler or a grid to the editor that may help you to organize the flow states and transitions.

6.2 Creating an operation in the BTT Perspective

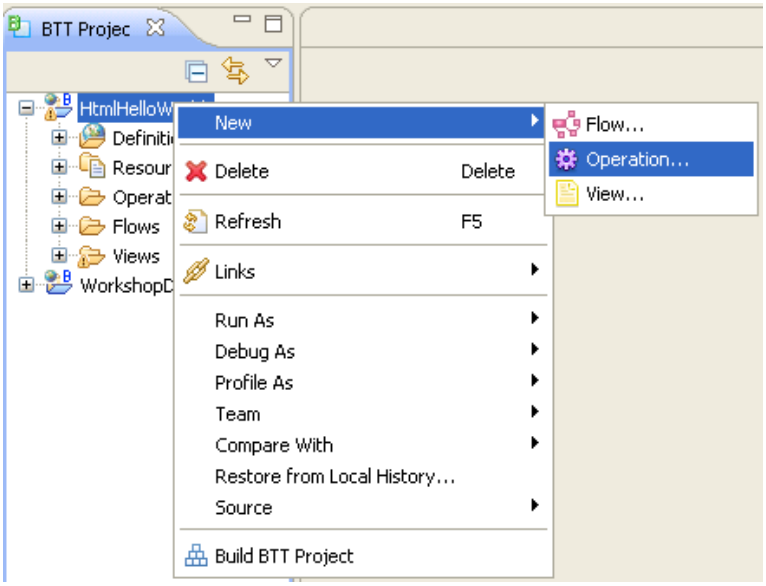
An operation is the entity responsible for performing the set of tasks needed to complete a basic financial operation, including data input and validation, interaction with external services, and management of the results and data received. For a full description of this core component, go to section Operations. The operations are one of the pieces that will be part of a BTT Flow and they can be created as part of the flow definition and then will use the context, data and services associated with the flow (refer to the previous section Creating a flow in the BTT Perspective) or independently, as a separated entity that may be reused by different flows. Then, there are two ways to create an Operation:

- from the BTT Perspective Project Explorer
- from the Flow editor, as part of the Flow creation

6.2.1 Creating an operation from the Project Explorer

To create an Operation from the BTT Perspective Project Explorer, take the following steps:

1. Right-click either in your BTT Project or in the Operations folder inside your BTT Project. Click New > Operation…

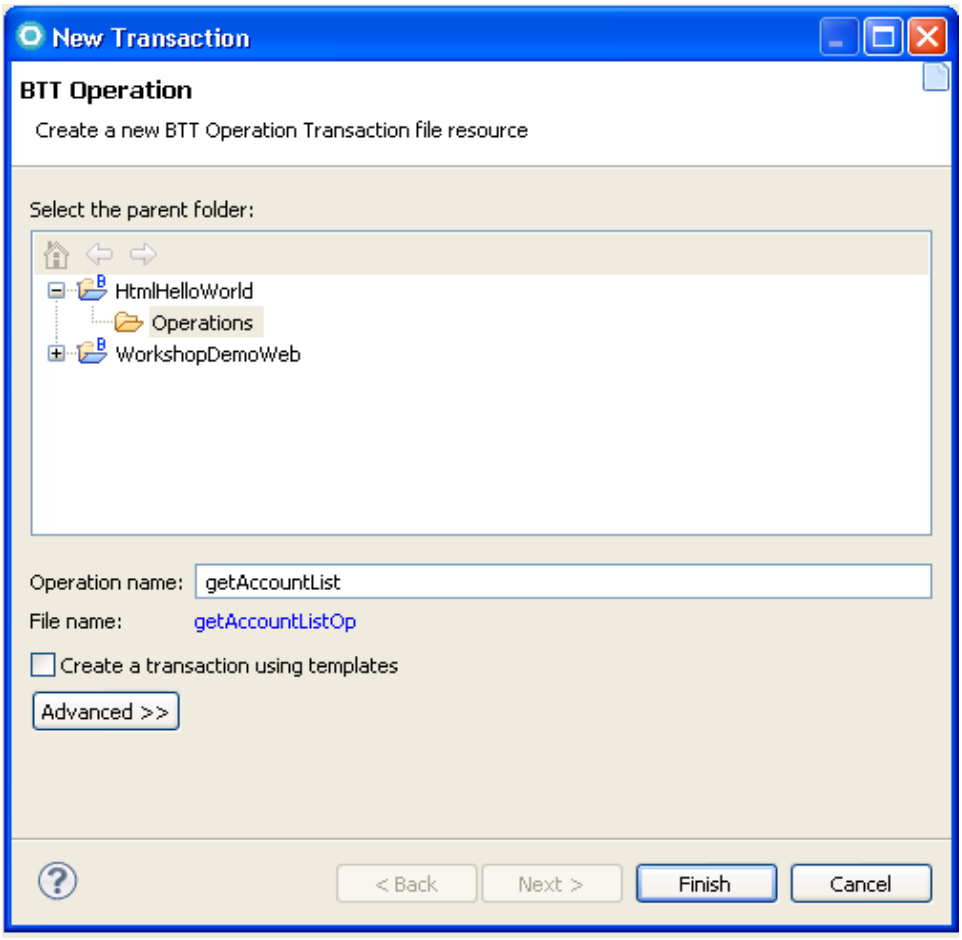


The BTT New Transaction wizard opens

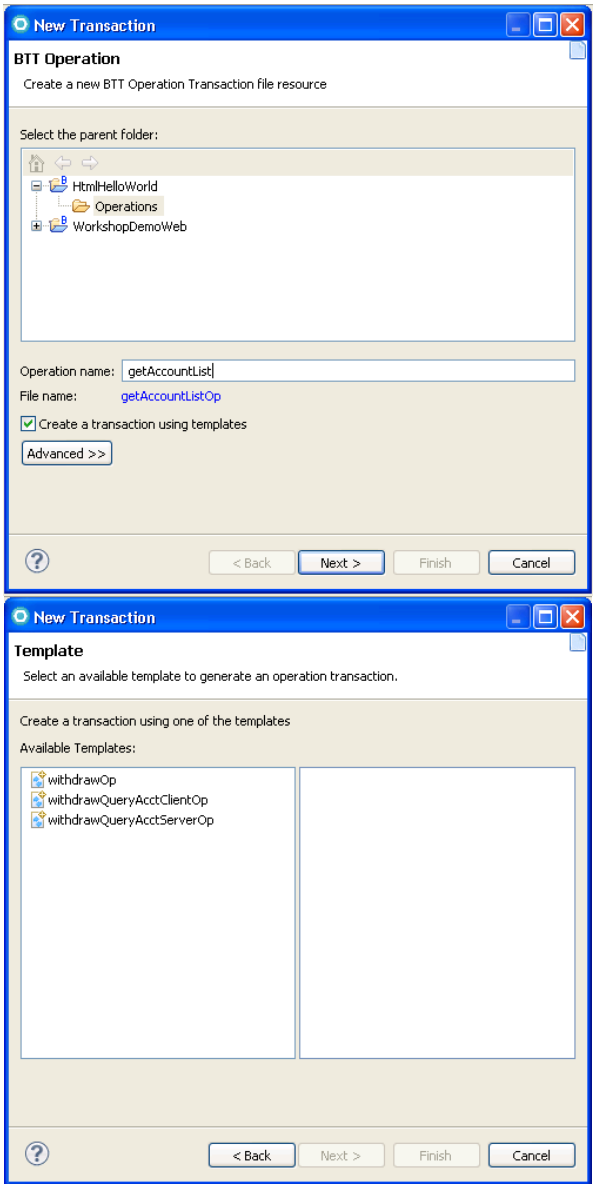
2. In the Operation name field of the New Transaction wizard main page, enter a name for the operation transaction file. Select the appropriate parent folder that should be the Operations folder of the BTT project (it is already preselected for you). Click Finish.

Note:

- The operation name you specify in the Operation name field cannot be duplicated. You will be prompted with a message if an operation with the same name already exists
- You must specify an operation name without extension. The operation name you enter will be appended with the string ‘Op’ and will be internally created with a .transaction file extension (‘sample’ operation name will be converted, in the file system, to sampleOp.transaction file). Final file name without extension is shown, while you are typing the operation name, in the File name field of the creation wizard.

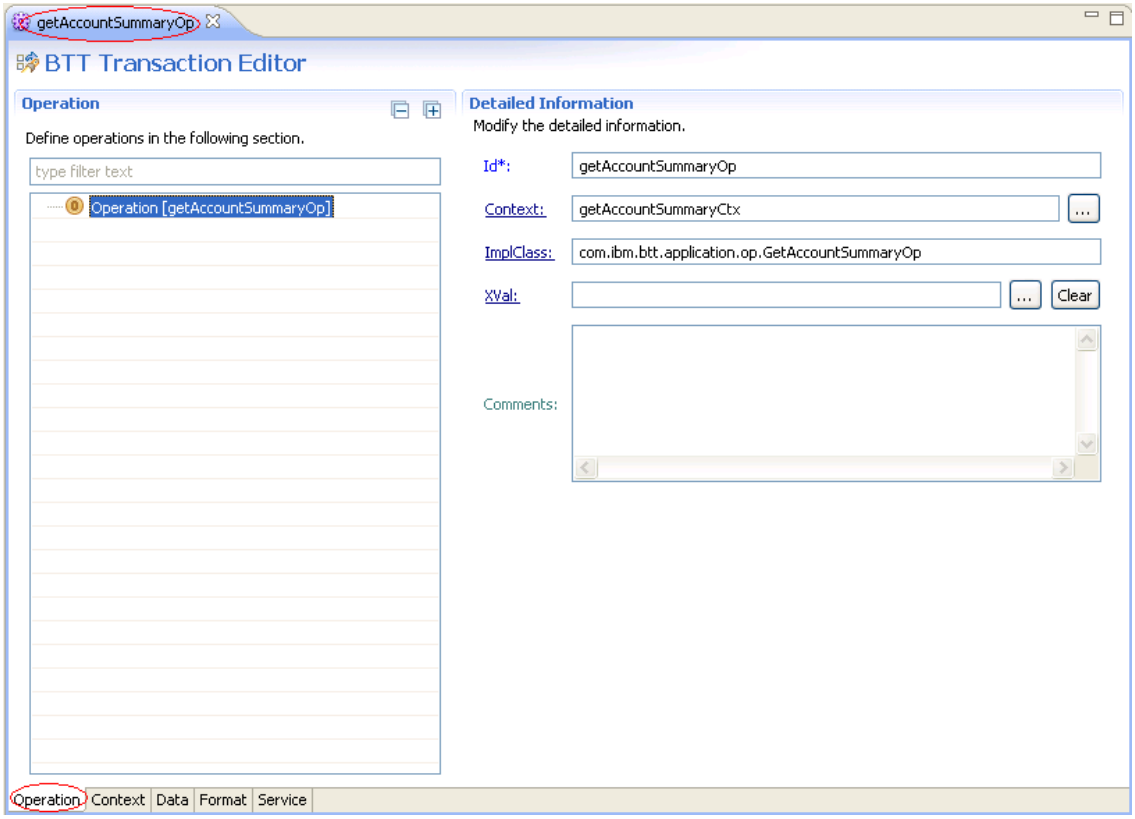


3. Optionally, you may want to create an Operation using an existing transaction template: a predefined transaction that can be reused. Then, select the **Create a transaction using templates** check box. Click Next, and then select the template you require. Click Finish.



Note: All the template files can be shared among different projects. The new Operation inherits all the properties of the original template that can be used as a starting point for your new operation design. Refer to section Using Operation Templates for more detailed information.

The Operation editor is opened.



6.2.2 Creating an operation from the Flow editor

While creating a flow, you have also the option to create a new operation that will be part of it. In the Flow editor wizard, go to the tab Operation and the Operation editor is opened

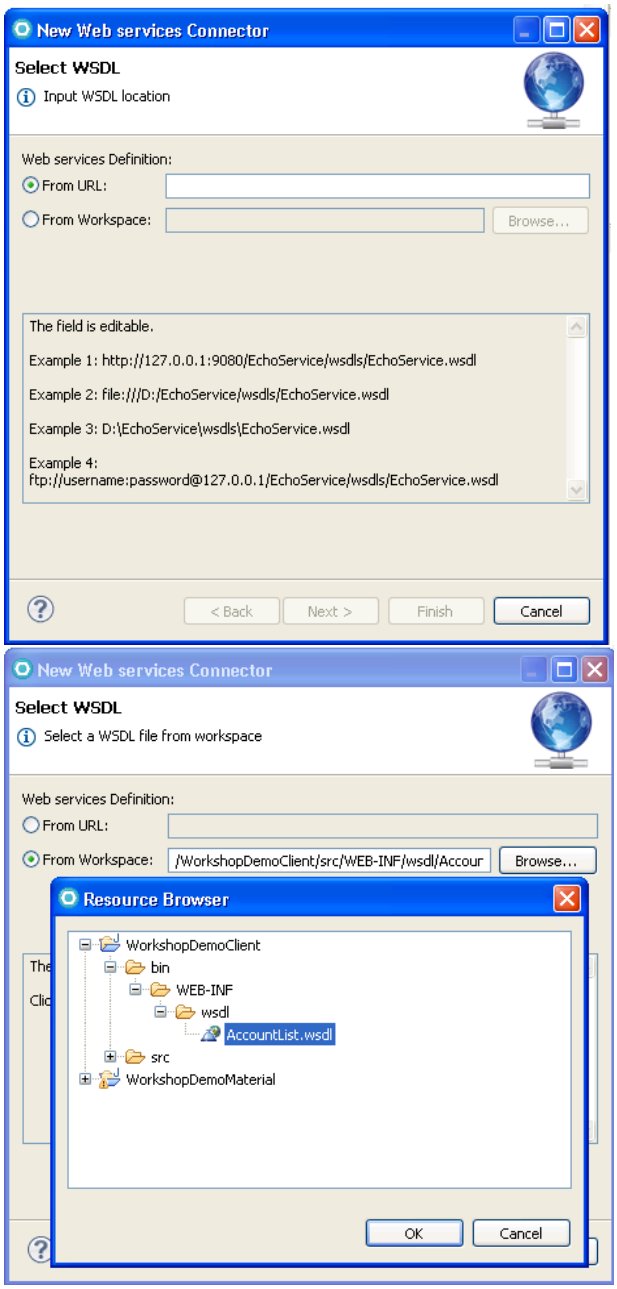
6.2.3 Creating an operation from a Web Service

Creating a new operation from a Web Service

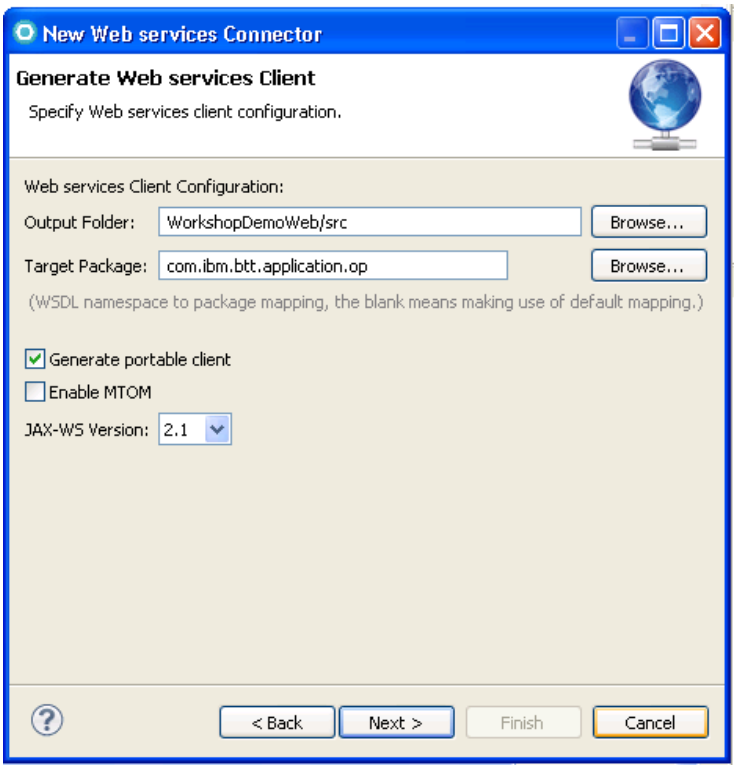
If you have created a Web service connector or you want to access any other published Web service from your transaction, you can create a Web service access operation from the Web service definition file (WSDL file). Do the following steps:

1. Go to the Java EE perspective and select your BTT Project

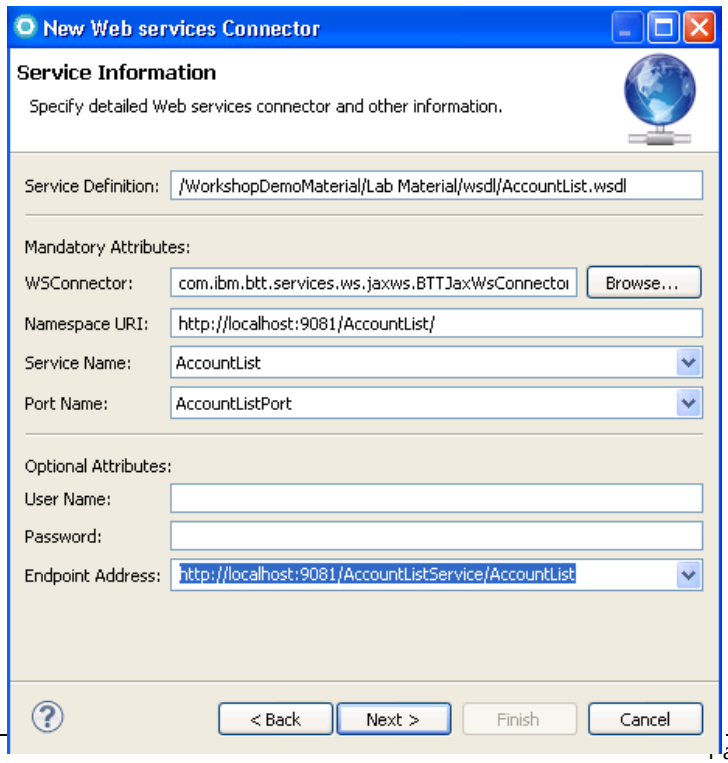
2. Right click on the BTT Project, select BTT Tools → Generate Self-defined Operations. In this wizard you can either select the WSDL file of the web services from a URL or from the workspace, if the web service was previously created in the development environment by the technical developers or the WSDL files for remote web services have been imported into the workspace as part of the environment configuration. In both cases the technical developers should tell you where the WSDL file is located.



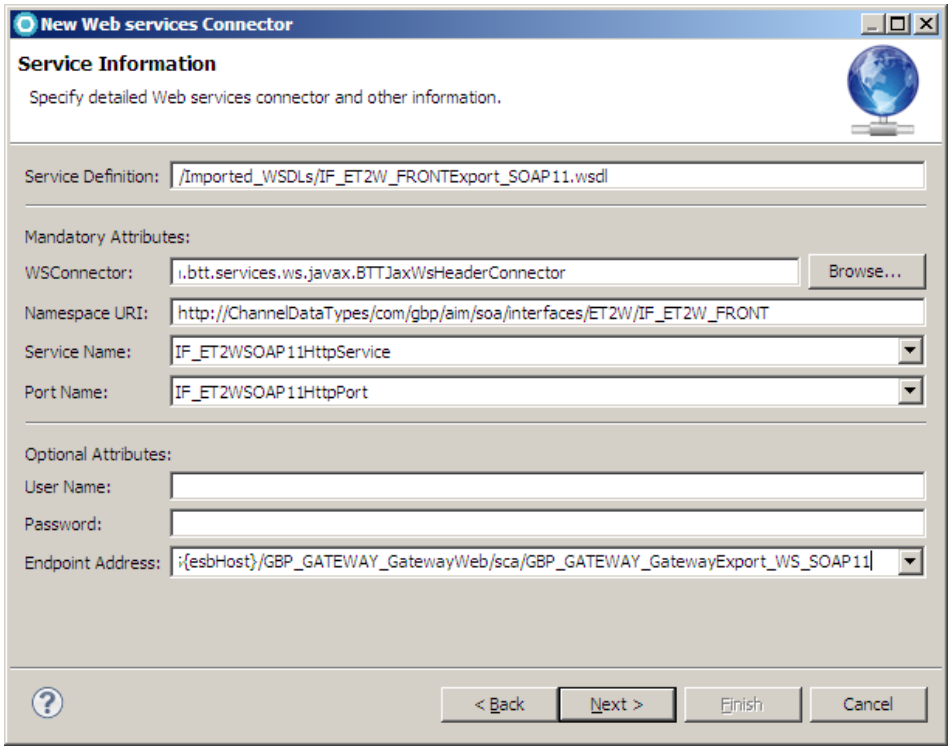
3. Press OK, and then Next. The **Generate Web services Client** wizard opens. You have to select the BTT Project folder in which the operation implementation class will be created and the Target package, meaning the WSDL namespace to package mapping. You can keep this field blank and then the default mapping will be used. For the other fields, you can also keep the default values. Press Next.



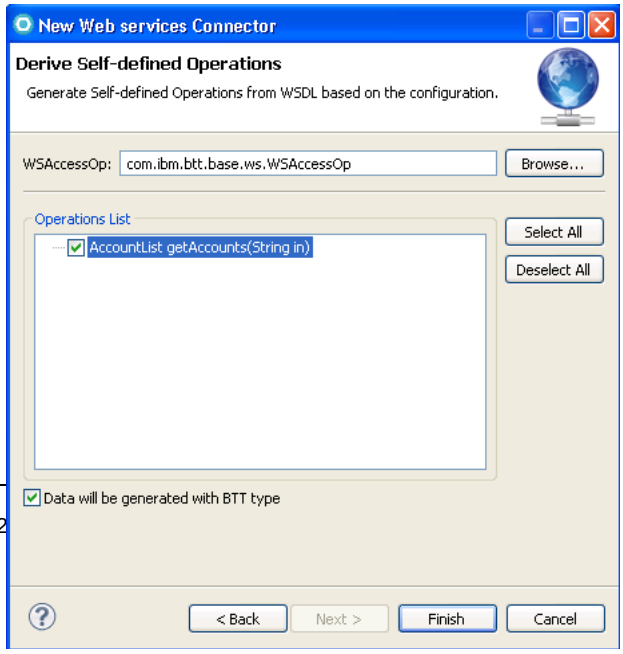
Now the **Service information wizard** opens. Two attributes are specially important in this wizard: the web service connector (WSConnector) and the Endpoint address. The toolkit provides a default WS connector that registers the handlers that add to the web service messages the architecture and security headers: `com.ibm.btt.services.ws.javax.BTTJaxWsHeaderConnector`. This connector is selected by default.



For the EndPoint address, you have to select the address of the web service function you want to generate the operation for. The URL in which the web service is published depending on the different application deployment environments. Your development environment can be configured by the technical developers so that a variable represents the web services server location. The variable for target web services server is resolved to a real URL address during runtime. See below an example of such a variable, esbHost



4. The **Derive Self-defined Operations** panel opens. The default implementation class for all automatically generated web service access operation is **com.ibm.btt.base.ws.WSAccessOp** (a different implementation may be provided by the technical developers and would be specified). Check from the Operation List table, all the web service calls for which you want to generate an operation.



Note: Several methods are listed as possible web service access calls if the web service does provide different functions. For each of the listed methods, an operation will be created for corresponding to a function provided by a web service and in each case the operation data would be automatically mapped to the request and response web service corresponding function data.

You can also check the option 'Data will be generated with BTT type'. If the option is checked, the tooling verifies for each web service request and response data if there is a defined BTT type matching the web service data type and uses it to define the new operation data. If the option is not checked or the corresponding BTT type is not found, the generated BTT data will be one of the default types. For instance, for a complex business object with several attributes, if the type does not exist in BTT, a keyed collection labeled with the business object name and with as many fields as business object attributes is created.

Click Finish. As a result, the operations are created. You can go back to the BTT Perspective, select your BTT Project, go to Operations folder and click on the newly created operation. The **Operation editor** opens. Go to section Using operation editor for a Web service access operation for more details about the operation resulting definition.

Configure an existing operation to access a Web service

If you have already an operation with its data and you want it to be a Web services access operation, then you have to create data mappings between the BTT data model and the Web services data model. Data mappings must be created so that the Web services provider is able to receive and process data sent from the BTT operation context. This is not automatically done as in the previous section, but the mapping is done manually once selected the operation and the web service.

You have to do the following steps:

1. Start the Generate Web Service Access Operation wizard.

   Creating the Web services connector is normally a technical developer task, so most probably you will be given the Dynamic Web Project containing the Web service that needs to be invoked and for which you need to generate the operation. Do the following steps:

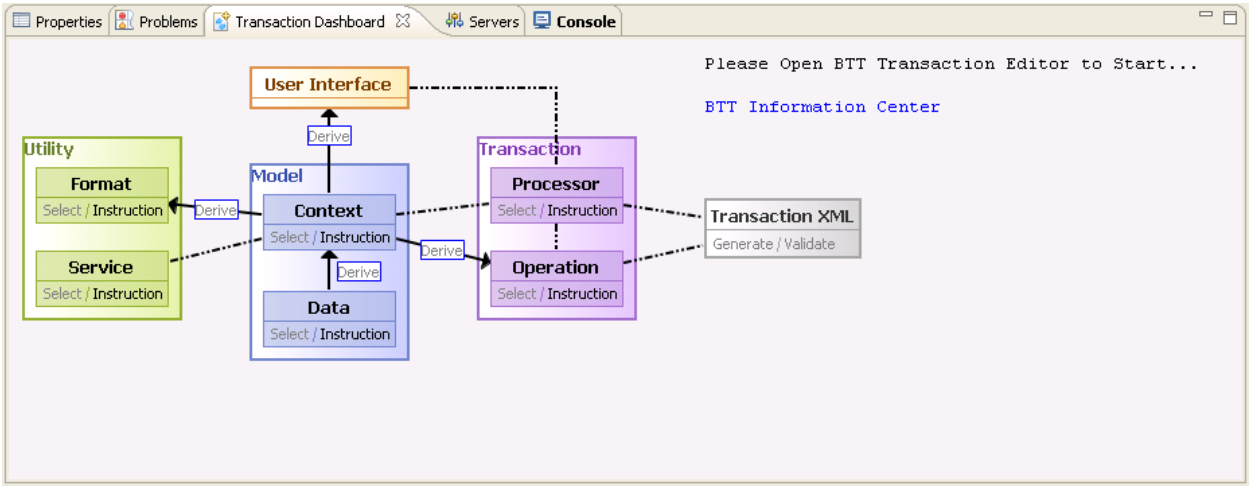
   a. Click Window > Show View > Other…

   b. In the Show View window, expand the Bank Transformation Toolkit folder, and then click Transaction Dashboard. Click OK. The Transaction Dashboard opens.

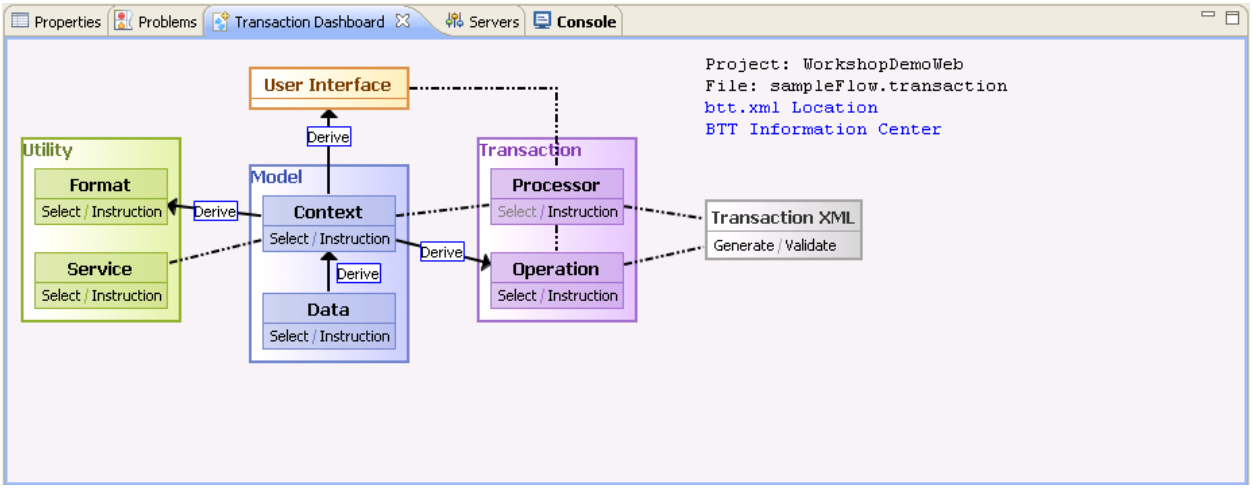
   c. On the Transaction Dashboard, click the **Derive** that is located on the arrow between Context and Operation.

   Note: The operation you are going to create must be associated to a specific flow, so it is required that the BTT Transaction Editor is open for a transaction creation. If it is not opened, a message is prompted ('Please Open BTT

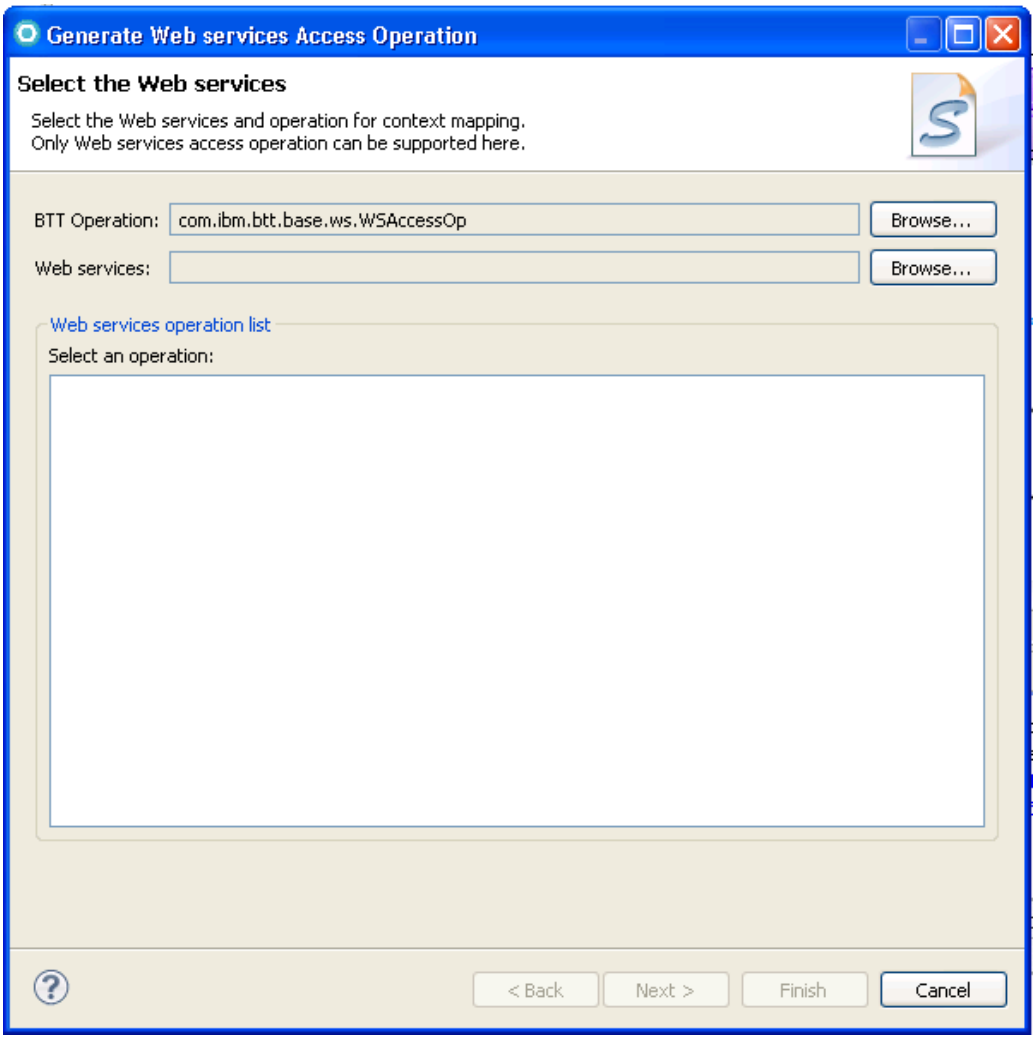
Transaction Editor to Start…') and the Derive buttons are disabled, as shown in the Figure below.



If the BTT Transaction Editor is opened, then the Derive buttons are enabled and you will be prompted with the flow information, as shown below:



d. The Generate Web Services Access Operation wizard opens.



2. In the BTT Operation field of the Select the Web services page, select the BTT operation that contains the data that must be mapped.
3. In the Web Service field, click Browse to select the Web service that you want to invoke.
4. From the Select an operation list in the Web service operation list area, select the Web service operation that you want to invoke, and then click Next.
5. In the Data Mapping page, select the data elements from the WebSphere Multichannel Bank Transformation Toolkit operation context and from the Web services data that require mapping.
    a. From the Context list, select the data elements that must be mapped. The data elements you select will appear in the Context area.
    b. In the Context area, select the data element that must be mapped for the request message.
    c. In the WS Operation area, select the Web services input parameter to which the WebSphere Multichannel Bank Transformation Toolkit context data element must be mapped for the request message, and then click Bind.

d. In the WS Operation area, select the Web services output parameter that requires mapping for the Web service output message.

e. In the Context area, select the data element to which you want the Web services output parameter to be bound to for the Web services output message, and then click Bind.

f. Repeat steps 5a - 5e until you have selected all the BTT operation context data elements and all the Web services data elements that require mapping.

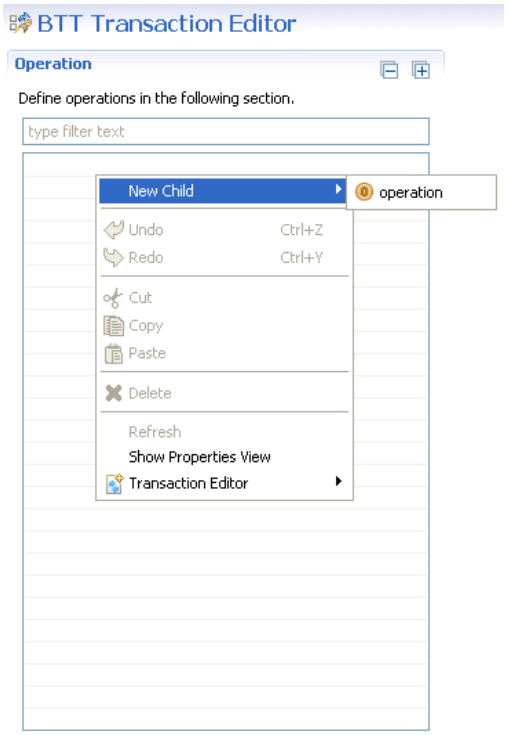
6. Click the Mapping details drop-down arrow to see the data mappings that will be created.

7. Click Finish. You can go through all changes that have been done to the operation definition by selecting the operation in the Project Explorer and opening the **Operation editor**. Refer to Using Operation editor for a Web service access operation for more details.

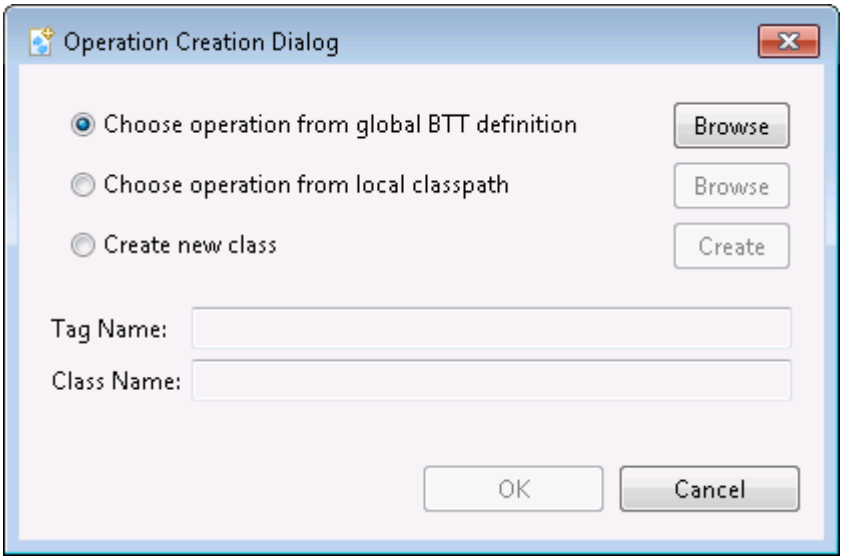
6.2.4 Using the Operation editor

The Operation editor can be used either to create a completely new operation or to create a new operation reusing an implementation already available in the development environment. The Operation editor is also used to display the definition of an existing operation as well as to complement the details of an operation when it has been automatically created either using a template or a web service definition file (refer to section Using the Operation editor for a Web service access operation for more details). To create an operation follow the steps described below:

1. The Operation editor is shown in the following Figure:



2. Right-click in the table under the Operation label, and then click New Child > Operation. The Operation Creation Dialog window opens.



3. If you want to create a fully new operation, in the Operation Creation Dialog window, select Create new class, and then click Create. The Java Class page displays.

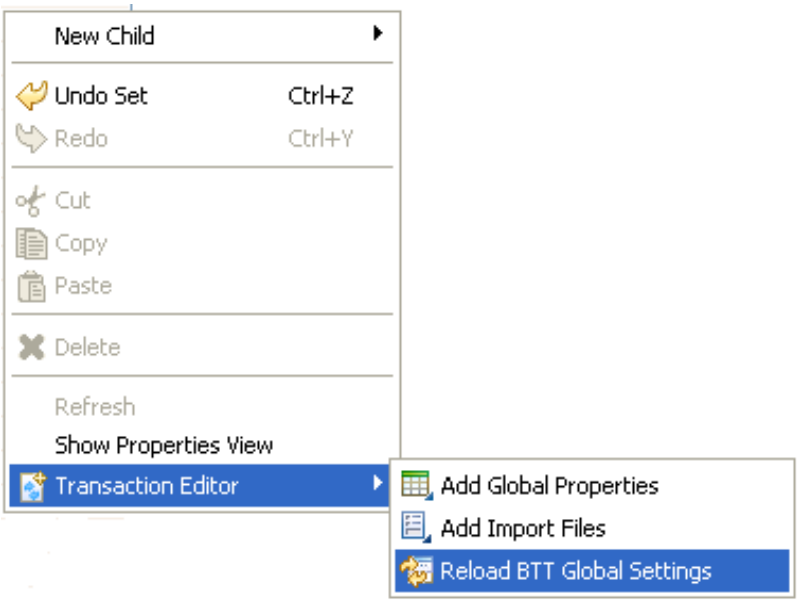
4. In the Java Class page, select the package in which you want to store the Java file.

5. In the Name field of the Java Class page, enter a name for the Java file. Click Finish.

6. In the Operation Creation Dialog window, click OK.

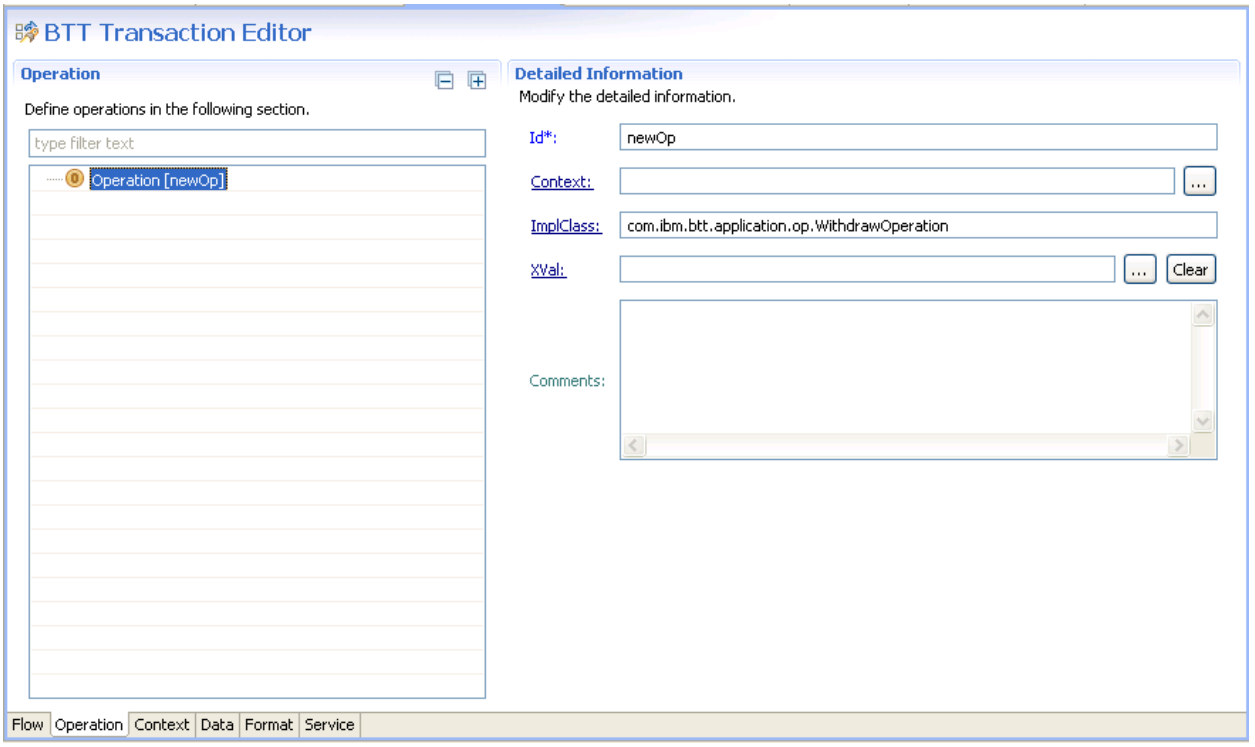
7. Alternatively, you may want to reuse an existing operation implementation class to create your new BTT operation. These classes should either implement the operation interfaces (Operation, ClientOperation or ServerOperation classes) or extend the basic operations (BTTClientOperation and BTTServerOperation). You can then select one of the following options:

   Choose operation from the global BTT definition. Click the Browse button to open the list of operations defined in the btt.xml file. These definitions should be provided by the technical developers. To ensure that you have access to the latest defined implementations, reload the content of the btt.xml file right-clicking in the table under Operations label and then click Transaction Editor > Reload BTT Global Settings, as shown in the figure



- Choose operation from local classpath. Clicking the Browse icon opens the list of Java classes implementing an operation that are accessible from the current BTT project.

In all cases, the implementation class field is prefilled either with the selected class or the new implementation class

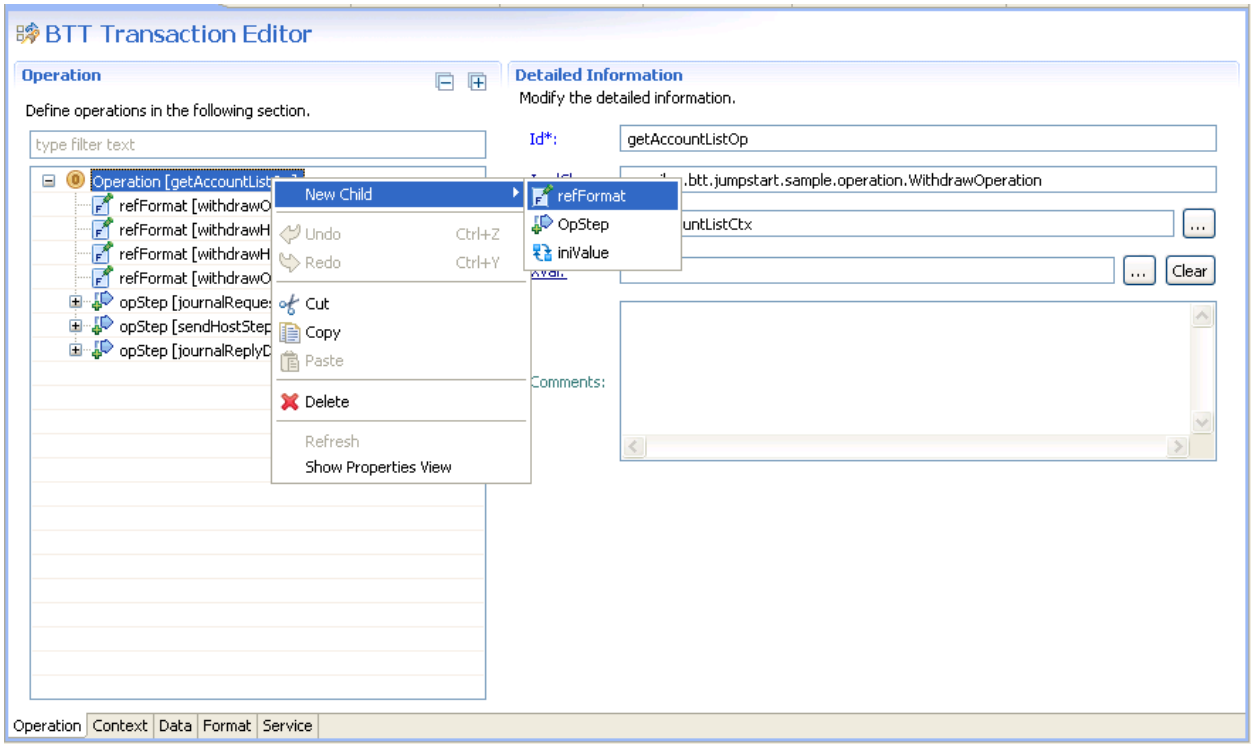


If you created the operation by new-creating operation class, you can then click the ImplClass label to open the operation Java file and add the operation logic into the code. The default Java code for the operation is as following code sample:

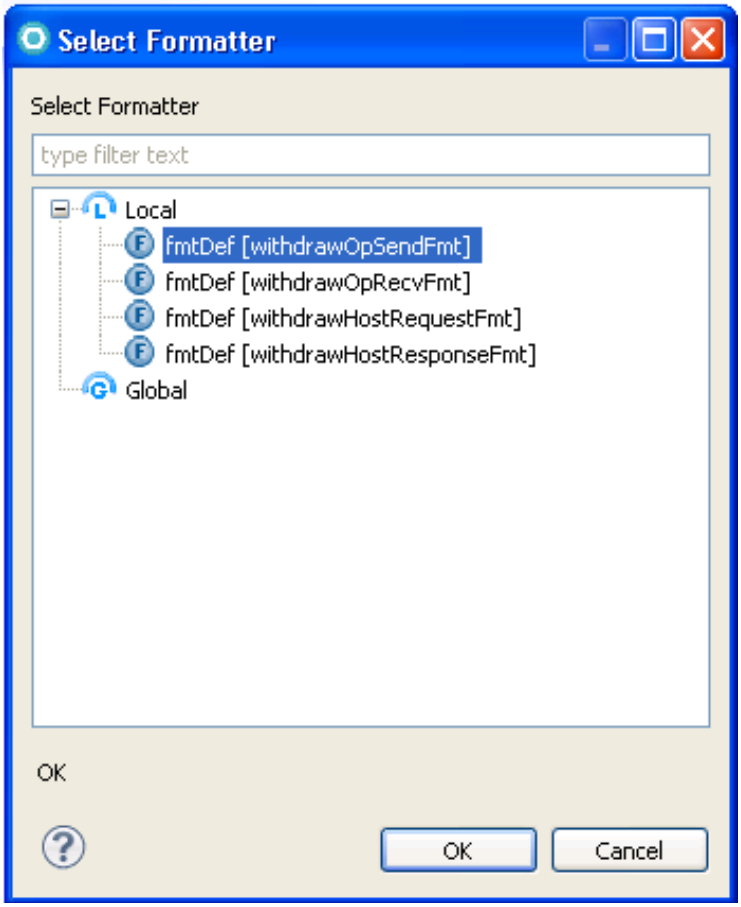
```
package com.ibm.btt.application.op;
import com.ibm.btt.base.BTTServerOperation;
/**
 * Class Generated by BTT Tool
 * Created since: 2011/11/09 16:42:23
 */
public class sampleOp extends BTTServerOperation {
  /**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 */
  public void execute() throws Exception {
  }

    }
```

8. Finally, you can associate a context and a cross-field validation implementation to the new operation by selecting the browse icon  beside the corresponding entry field. For detailed information about these operation properties, go to Server operation external definitions section. To create a new context associated to the operation, refer to section Defining Context in the Transaction editor. Also, follow the links to learn how to proceed to create new data, new formatters or new services to be used by this operation.

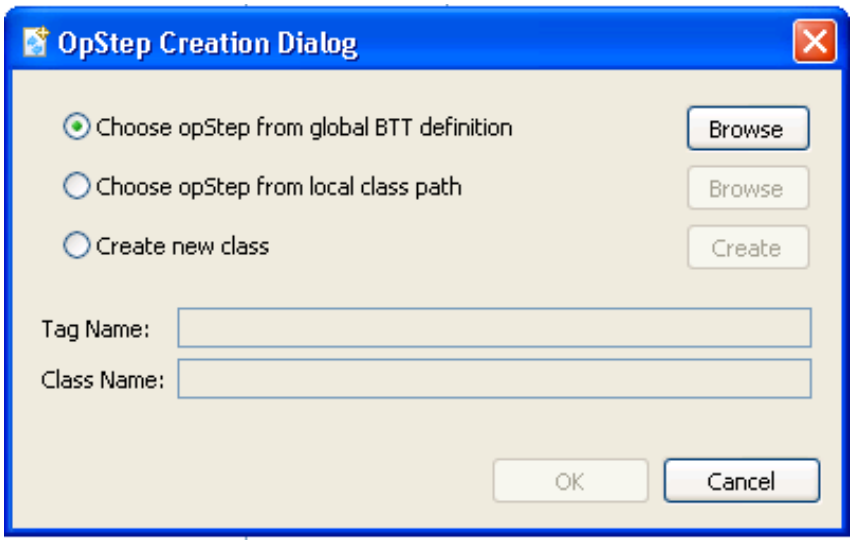
9. For a new operation, the Operation tab of the Transaction editor allows you to add children to the operation being edited: operation steps (OpStep child), formatters (refFormat child) and initial values of specific operation data (iniValue child), as shown in the *Figure* below.



The **formatters** should have been previously defined by the technical developers following the toolkit implementation rules or can be added using the Format tab in the Transaction Editor (refer to section Defining Formatters in the Transaction Editor for more details). A reference to an existing formatter is added to the operation by selecting it from a list of available formatter definitions.



When adding a new **operation steps**, the OpStep Creation Dialog opens. The operation step can either be selected from a list of currently visible implementations in your BTT Project environment or a new one can be created.

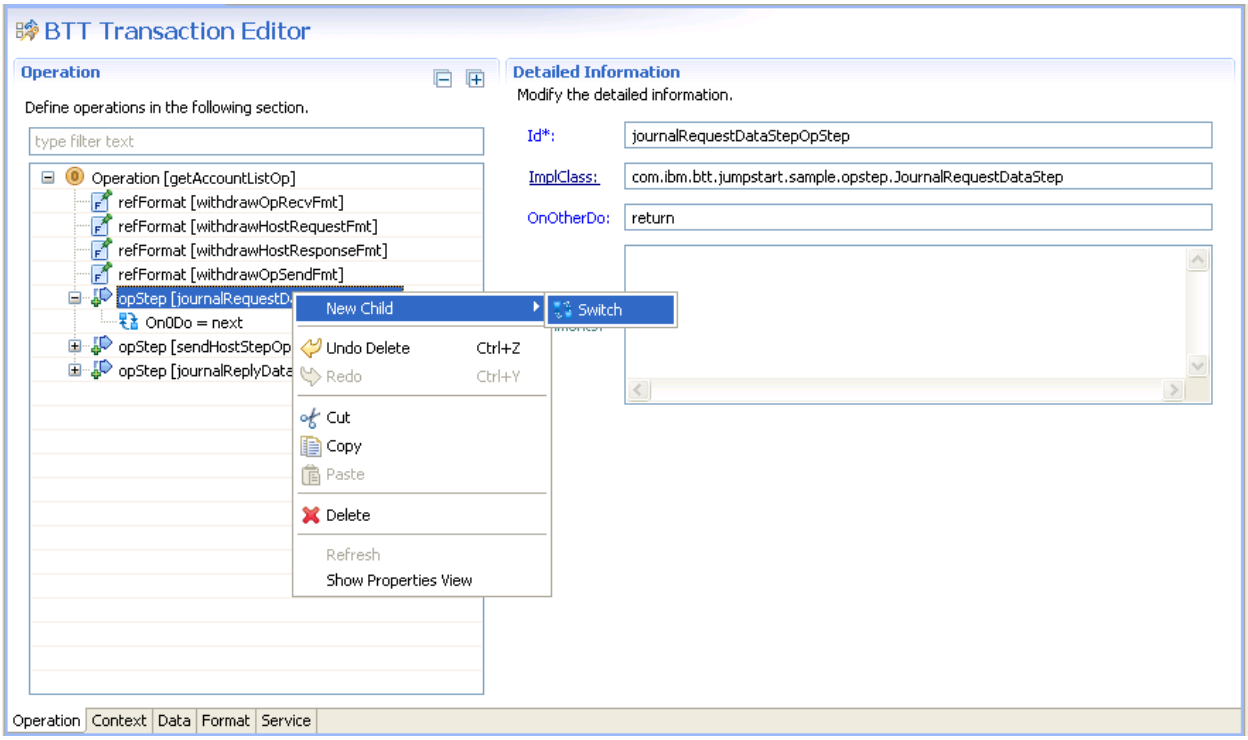


In a similar way than when creating an new operation, when adding a new operation step you are prompted to enter the Java class name and the package in which the class is created. Back to the Operation editor panel, by clicking in the ImplClass label you can add your operation step code to the generated template. An example of the template, for operation step 'sampleOpStep', is shown below:

```
 package com.ibm.btt.application.op;
import com.ibm.btt.base.OperationStep;
/**
 * Class Generated by BTT Tool
 * Created since: 2011/11/09 16:38:34
 */
public class sampleOpStep extends OperationStep {
  /**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 */
  public int execute() throws Exception {
    return 0;
  }

      }
```

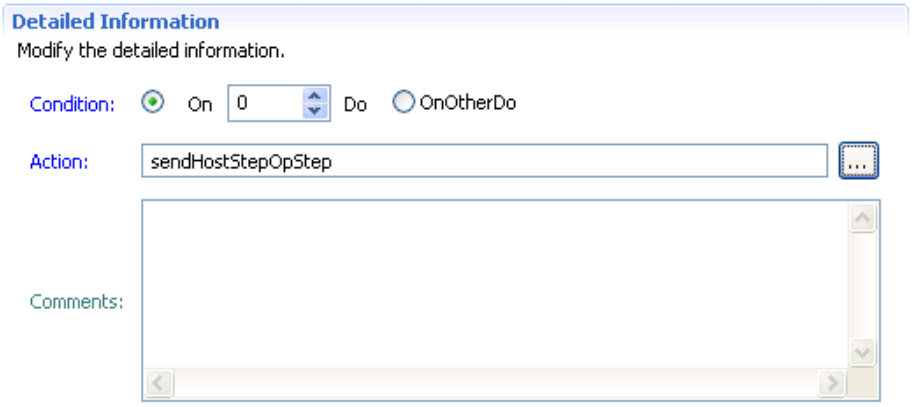
From the global BTT definition or the local class path you may be able to select also a CompareAssertion, ExistsAssertion or NumofElementsAssertion. The properties to be set for any type of operation step are defined in detail in section Operation step external definitions.

The operation steps may have also a **Switch** child that controls how the operation moves from one operation step to the other.

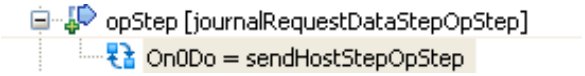


The **Switch** definition allows you to decide which is the next step or action depending on a numeric value that the operation step should return. Possible

actions are ‘next’, meaning that the process will move to the next defined operation step, or ‘return’, meaning that the processing of operation steps stops and returns control to the parent operation execution. An escape action is also available in case there is an error in the operation step execution (OnOtherDo condition). As an example, to add a switch to the journalRequestStepOp step to go to step sentHostStepOp when it return 0, you have to add the values in the switch detailed information as shown next *Figure*

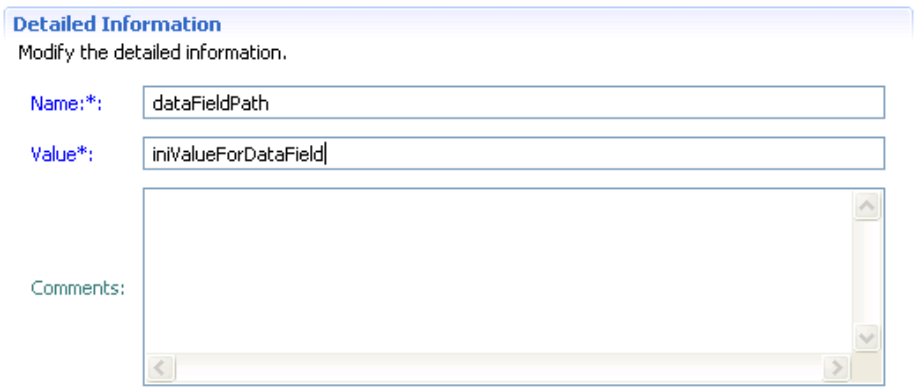


The result entry in your operation definition table will be the following:



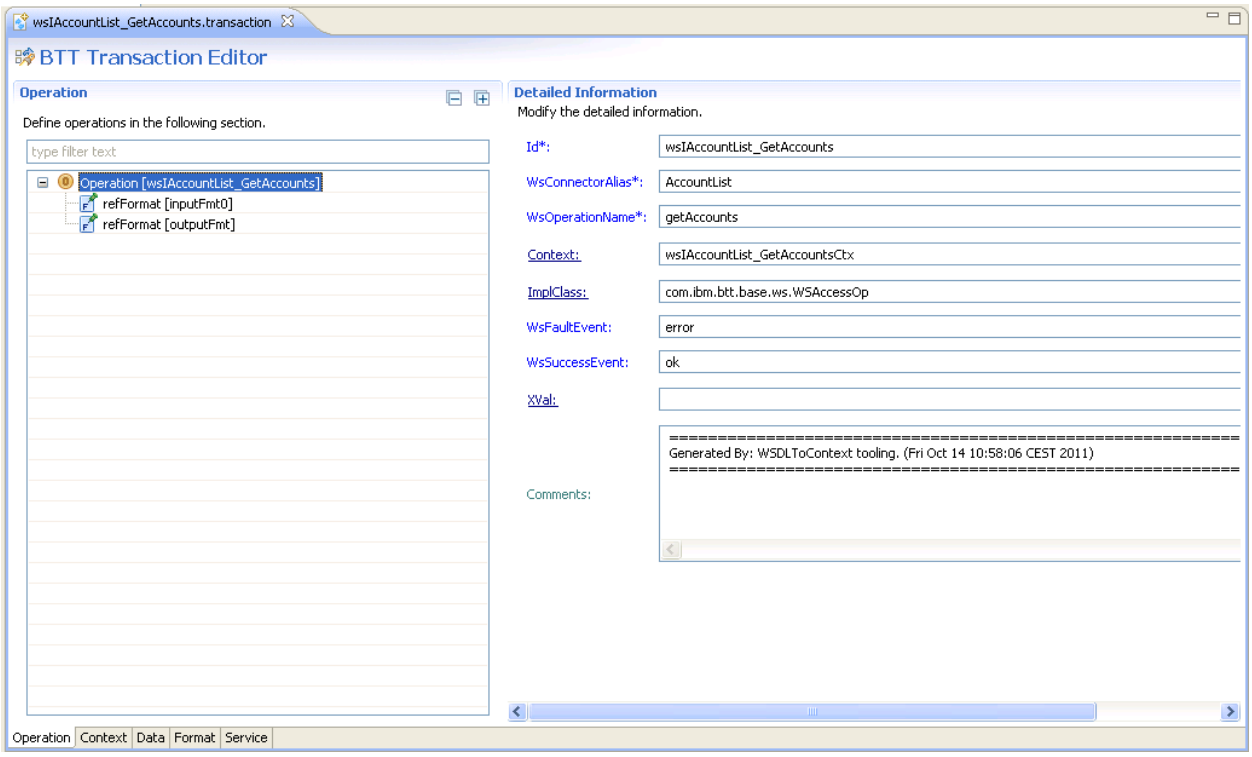
The switch definition is not needed for assertion operation steps, as they have in their own definition what would be the next step or action depending on the Boolean value resulting from the evaluation they run.

The **iniValue** child defines the initial value that must be given to a data element in the operation context during the operation initialization process. The iniValue is defined by a field Name, representing the data field path in the context (simple or composite key of the data) and a field Value, containing the initial value given to the specified data field.



6.2.5 Using the Operation editor for a Web service access operation

A Web service access operation has additional attributes that the other type of operations do not have. An example of the resulting definition of a Web service access operation is shown in the Figure below:



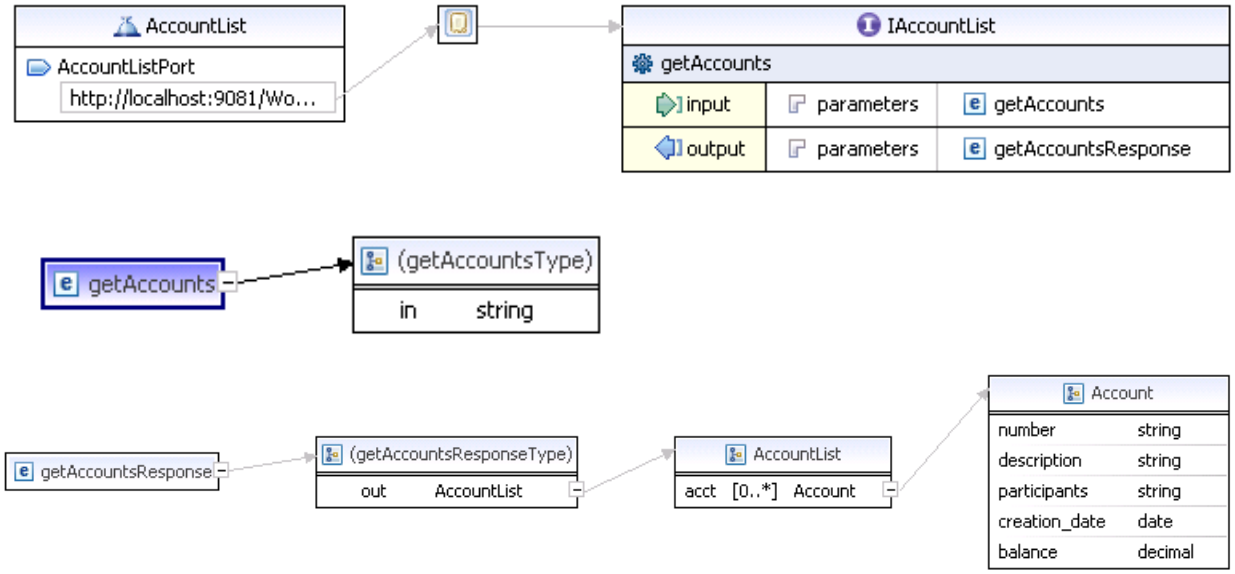
A comment is added in the Comments field so you can identify this operation as being automatically generated by the WSDL to Context tooling.

The implementation class is always com.ibm.btt.base.ws.WSAccessOp if using the default BTT implementation for a web service access operation.

Additional attributes are available in the Operation editor **Detailed Information** panel for a web service access operation:

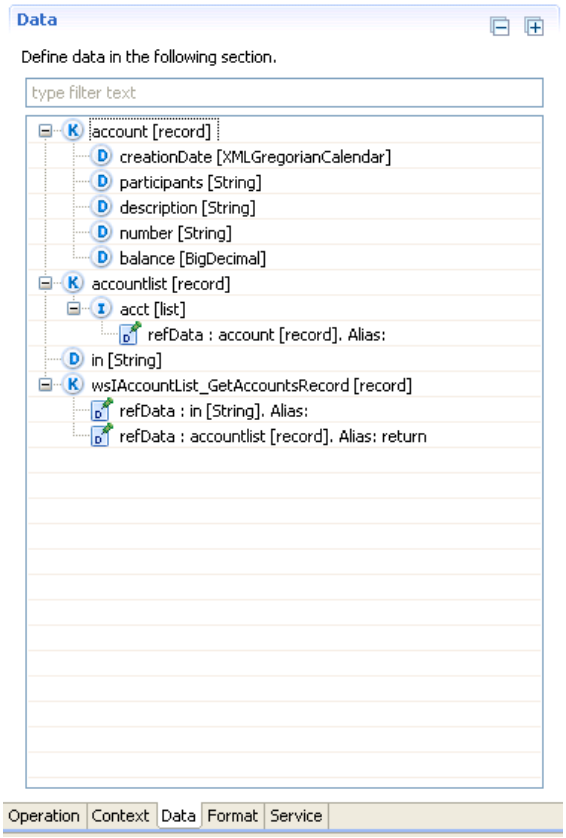
- **WsConnectorAlias**: alias of the web service connector
- **WsOperationName**: name of the web service function associated with the BTT operation
- **WsFaultEvent**: event generated by the operation when there is a failure during the web service call
- **WsSuccessEvent**: event generated by the operation when the web service request ends successfully.

You can now check for any other defined BTT elements associated to this web service access operation: context, data, formatters and services. If we use the example shown in the Creating an operation from a Web Service section, the WSDL file (graphically represented) associated with the web service is the following:



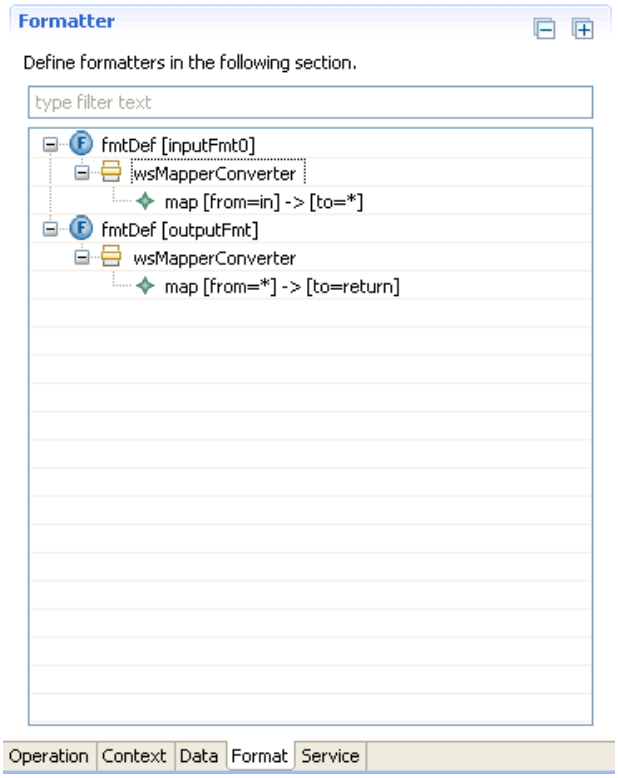
The resulting operation context data, formatter and services are the following:

- **Data**. Data has been generated using the corresponding BTT types of the web services data types.



- **Formatters**. Two formatters are created: an input formatter, to map the request data from the operation context to the web service request format,

and the output formatter, to map the data from the web service response format to the operation context.



The **wsMapperConverter** tag has associated several attributes, already set as result of the automatic operation generation:

isPrimitive

The attribute is mandatory and must have one of the following values:

- true: the Java Object in the data conversion is a Java primitive type (e.g. 'int', 'short', 'double').
- false: the Java Object in the data conversion is not a Java primitive type (e.g. 'java.lang.String', 'int[]' and JavaBean).

javaClass

This is a mandatory attribute. Its value is used to indicate the actual type of the Java Object in Java Web services operation during data conversion.

The **map** tag has the following attributes:

from

The attribute is mandatory. The value of this attribute indicates the source data in the mapping process.

to

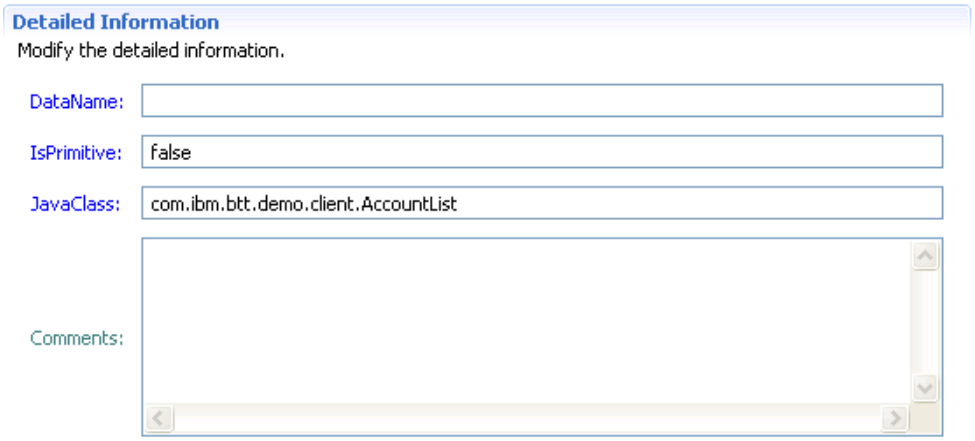
The attribute is mandatory. The value of this attribute indicates the destination data in the mapping process.

byReference

This is an optional attribute. It indicates the Web services mapper runtime whether to make a copy or just reference the source data to construct the target data. The byReference attribute must have one of the following attributes:

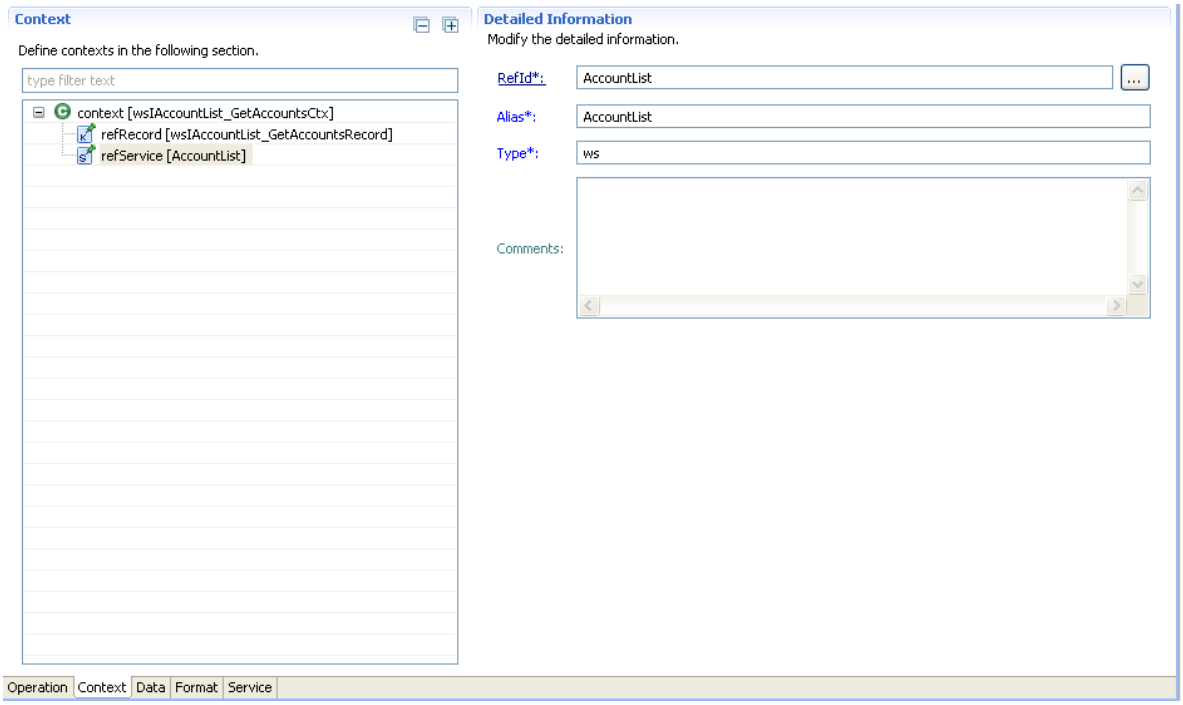
- True: the Web services mapper runtime references data.
- False: the Web services mapper runtime makes a data copy.

Here is an example of the attributes generated for the output format:



In this case, all the mappings are direct as the operation data and the corresponding web service data have the same structure.

Services. No service is defined as part of the operation. The service referenced in the operation context is part of the common services for the BTT project.



The type of service is ‘ws’ and the common class that implements these type of services is *com.ibm.btt.services.ws.jaxws.BTTJaxWsConnector* . The external definition generated for the service, in our example, is the following:

```
<service id="AccountList"
namespaceUri="http://localhost:9081/AccountList/"
serviceName="AccountList" portName="AccountListPort"
serviceEndpointInterface="com.ibm.btt.demo.client.IAccountLi
st" wsdlUrl="/WEB-INF/wsdl/AccountList.wsdl"
implClass="com.ibm.btt.services.ws.jaxws.BTTJaxWsConnector"
mtom="false"
endpointAddress="http://localhost:9081/WorkshopDemoService/A
ccountList"/>
```

6.3 Using flows templates

The flow templates are built by the technical developers and are used by the functional developers as skeletons for the development of transactions.

Normally the templates do only contain states and transitions that are common for all different transactions that are going to be part of a BTT Project, as the initial state, the final state, common pages (views that are reused by different flows and have been already defined in the project) and common operations (defined as common resources in the BTT Project).

Note:

- Pages and Operations being part of a flow template should be accessible from the BTT Project that is going to use the template.

- Operations that are part of a template can be created as empty boxes (with no value set for the operation properties context and implementation class; only the id is mandatory when creating an operation) whose characteristics will be filled when implementing the specific transaction that uses the template.

6.4 Using operation templates

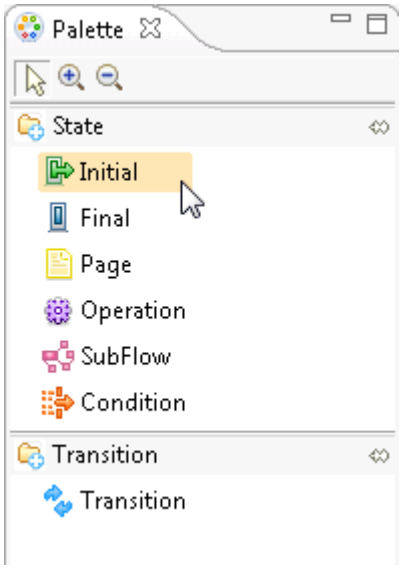
The operation templates are built by the technical developers and are used by the functional developers as skeletons for the development of transactions.

The operation templates can contain context data, formatters and services. When creating an operation from an operation template, all these entities are inherited and a new operation context is created for the new operation with a copy of the data that is defined in the operation template, so the formatters and services will still be valid for the new operation.

6.5 Drawing a flow using the Flow editor

A flow is a representation of a particular route through a business process or business operation. The flow reflects the results of performing different actions and responding to system-generated events within the business process. The flow definition is used at runtime by a flow processor to perform a specific business process based on the defined flow of states. The flow is graphically represented by a statechart diagram that shows a sequence of states and the transitions between them. This section describes how the Flow editor can be used to draw a flow by using and configuring the states and transitions available in the Flow editor palette. For a detailed description on the flow processor and how a flow is executed, go to section How a flow processor works.

In the Flow editor palette, there are State and Transition widgets, as shown in the Figure below



6.5.1 States

A state is a condition or stage of a flow in which the flow processor verifies if a condition is satisfied, performs some activity, or waits for some event to progress to the next state. In the statechart diagram of the flow, each of the nodes relates to a specific state. Within the definition of a flow, there is a definition for every state that the process can attain. The state definition contains all the events that may trigger a state transition. The transitions determine how the processor behaves in response to events that happen while the processor is in that state.

A flow contains an initial state, one or more final states and any number of other states through which it might go.

Initial State

The first state in a transaction flow is called an initial state, A flow must have only one initial state. The initial state is represented in the statechart diagram by the following node:

Final State

The last state in a transaction flow is called a final state. A flow can have as much final states as wanted. All final states must have associated an exit event for the transaction flow to complete. The exit event of a final state means that the flow will send the event specified when the flow comes to the end. The exit event is defined in the

properties view of the final state. A full description of the final state properties can be found in section Creating a final state. The final state is represented in the statechart diagram by the following node:

Page State

A page state is a state that is bound to a view, and it is used to gather information through a view that has been created with the XUI editor. Widgets such as the link or button widget are used to navigate from a page state to another state in the flow. To navigate to other states from a page state, you must bind the view widgets to events and this is done by creating widget rules that trigger events when certain conditions are satisfied (refer to section Setting ECA rules to widgets for more information about how to associate rules to widgets). The events defined for the view's widgets are shown as state events in the Page state events panel. The Page state is represented in the statechart diagram by the following node:

Operation State

An operation state is a state in a transaction flow that is bound to an operation. To be able to navigate from an operation state to another state, the operation state should identify which are the events that may occur during the operation execution. These events become part of the operation state properties.

There are three ways to find out the events that should be part of the operation state definition:

- if the associated operation is an operation already available in your BTT project, it is recommended that the technical developers add an annotation to the Java class that implements the operation to identify the events triggered by this operation. As an example, if the operation triggers the events 'ok' and 'error', then the annotation should be

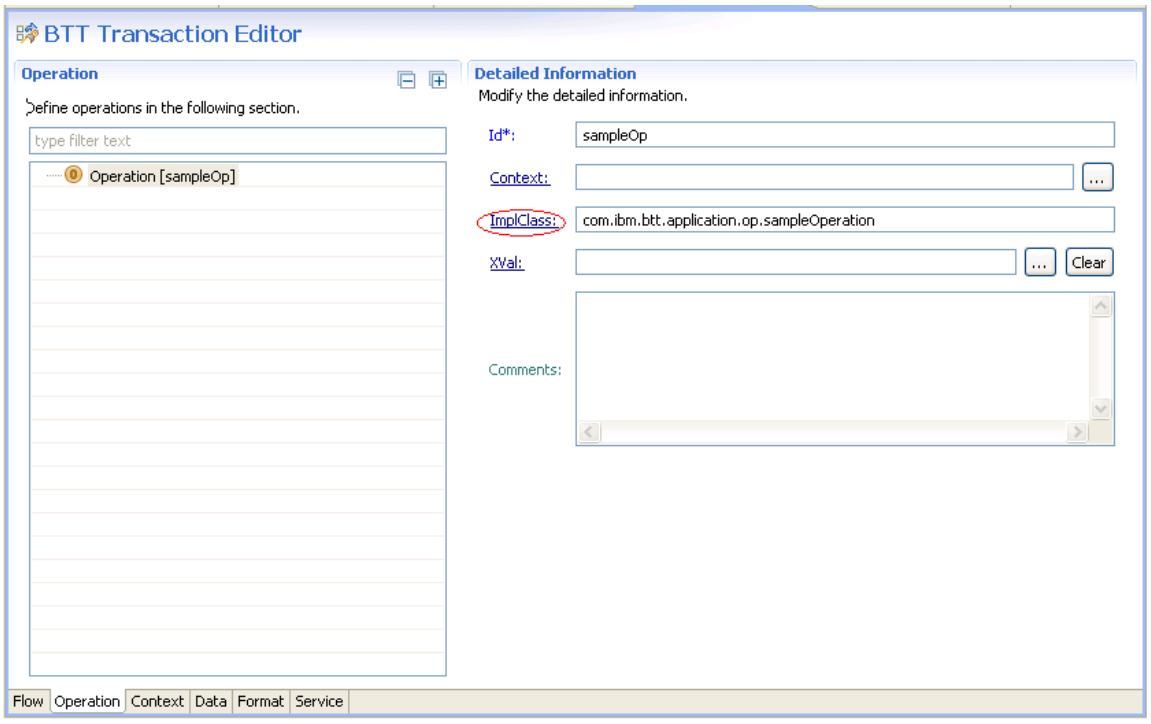
  @EVENT({"ok","error"})

  This way, when the operation state associated to this operation is created, the events panel is automatically populated with these two events and marked with a green arrow, meaning that the events have been populated directly from the operation and then cannot be deleted, although other events may be added manually

Otherwise, the events are marked with a blue arrow.

The advantage of using the EVENT annotation is that whenever the operation events change, this is automatically reflected in the operation state. You can synchronize the operation state events information with the operation by clicking on the icon  .

- you may create your own self-defined operation while creating the flow by going to the Operation tab in the Transaction Editor. Once you have provided the package and class name, you get into the operation editor panel, from where you can start coding your operation by clicking the ImplClass label



The code generated for you is the following:

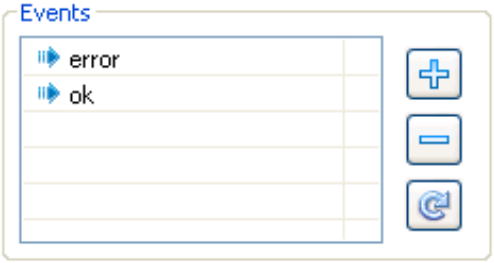
```
package com.ibm.btt.application.op;
import com.ibm.btt.base.BTTServerOperation;
/**
* Class Generated by BTT Tool
* Created since: 2011/11/08 11:03:37
*/
public class sampleOperation extends BTTServerOperation {
```

```
 /**
* <!-- begin-user-doc -->
* <!-- end-user-doc -->
*/
public void execute() throws Exception {
 }
     }
```

To identify the events that will be generated by the operation, you should add the EVENT annotation before the execute() method, as shown in the following example for events ‘ok’ and ‘error’ (red highlighted statement)

```
package com.ibm.btt.application.op;
import com.ibm.btt.base.BTTServerOperation;
/**
* Class Generated by BTT Tool
* Created since: 2011/11/08 11:03:37
*/
public class sampleOperation extends BTTServerOperation {
 /**
* <!-- begin-user-doc -->
* <!-- end-user-doc -->
*/
@EVENT({"ok","error"})
public void execute() throws Exception {
 }
     }
```

- if no event annotation has been added to the operation code, the technical developers should tell you (if you are not implementing the operation) what are the events that can be expected as a result of the operation execution and then they can be added manually as part of the operation state definition. The Add and Remove icons are both enabled and the events added using this panel appear with a blue arrow.

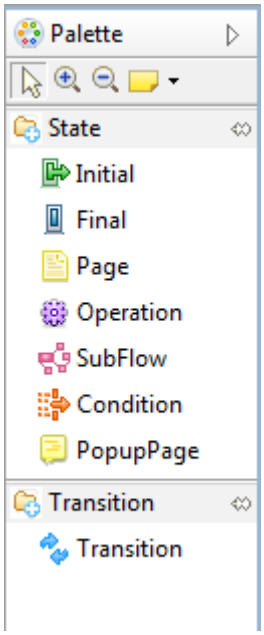


The Operation state is represented in the statechart diagram by the following node:

Client State and PopupPage State

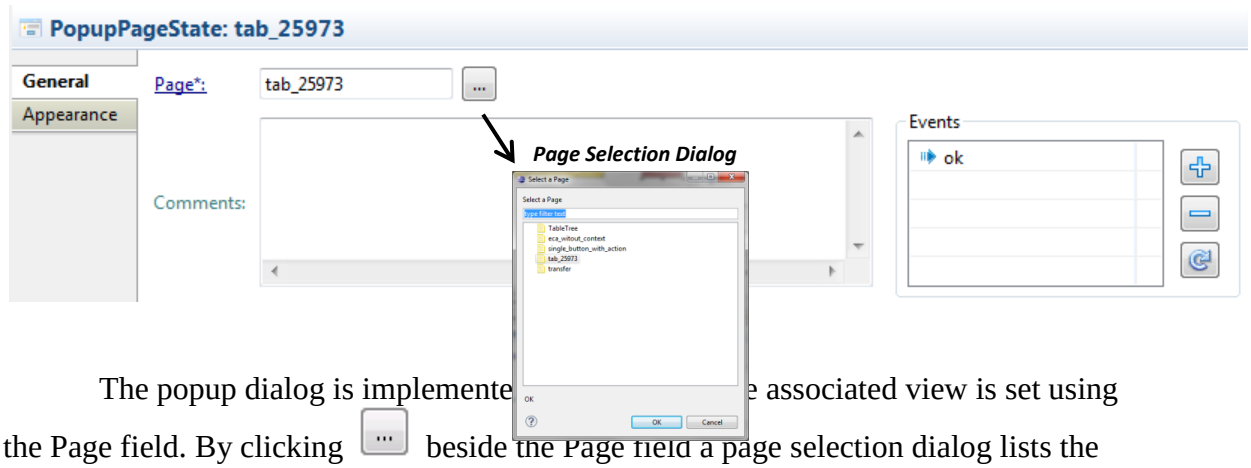
A client state allows the execution of client side operation from a transaction flow. The toolkit comes with an implementation of a client state that allows to show a popup

dialog in the client application side (see **PopupPage** in the Figure below), as well as with an extension mechanism to add any other project level client invocation behaviors (i.e. device control).



The popup dialog appears in front of the page shown previously as part of the flow execution and it can be moved to another position if required.

The following figure provides an example of adding a PopupPage state to a flow:



The popup dialog is implemente e associated view is set using the Page field. By clicking ... beside the Page field a page selection dialog lists the available views for selection. The events defined in the selected view are used by the flow processor to navigate to the next flow state; these events are automatically listed in the Events panel once the view has been selected. The popup dialog has access to the view context data and the flow context data so it can be used to defined the data mappings for

transitions; it also inherits any validation (single field or cross fields validations) and ECA rules defined in the view.

SubFlow

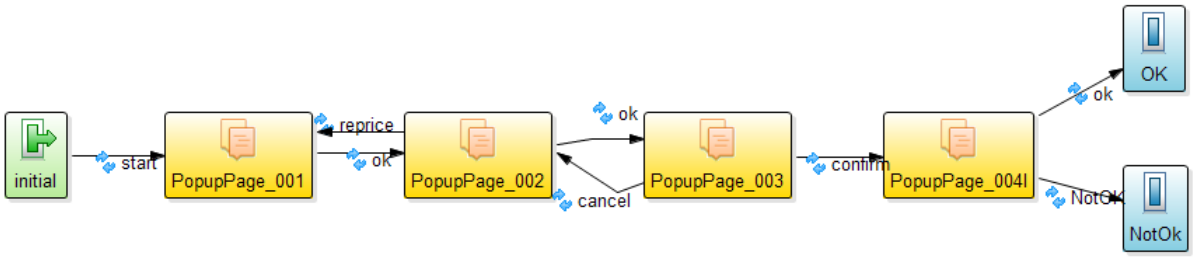
A SubFlow state is used to call another flow as part of a flow execution.

The flow processor can transition to a SubFlow state that has its own separately defined flow processor. A sub-flow has its own context, which is isolated from the context of the parent flow. You can define data mappers from the parent to the child context, thus enabling separate namespaces and the ability to reuse the sub-flows from different parent flows. When the navigation of this sub-flow is complete, the specified data mapper copies the reply data to the parent context and the control of the flow returns to the parent flow.

The events associated by default to the SubFlow state are the exit events defined in the sub-flow and cannot be deleted. The list of events can be refreshed by using the  icon.

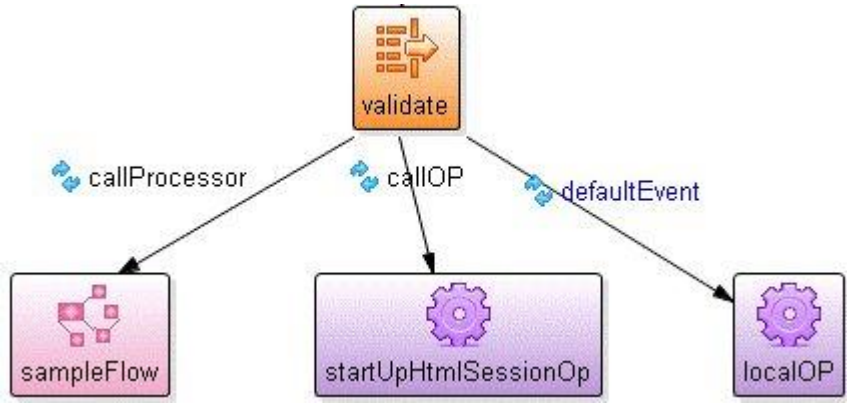
The sub-flow state is represented in the statechart diagram by the following node:

A sub-flow composed by a sequence of popup states is supported, as shown in the Figure below. This type of sub-flows are an example of common flows that may be reused by different transactions.



Condition

A condition state is an evaluation that the flow processor makes before performing an action. The events returned by the condition state determine what the next state in the flow execution process is. A condition state evaluates its conditions sequentially until a given condition is true. Then it fires the associated event to trigger the following state. Following Figure provides an example of condition state:



The condition state has associated a default event that is fired when no condition evaluates to true.

6.5.2 Transitions

A transition is the passing from one state to another. A transition occurs when a specified event occurs and the process satisfies specified conditions. A transition has an owning state and it identifies the actions the process is to execute and may identify the target state that the process will be in once it executes the action. If the transition does not identify a target state (the target state is the same as the owning state) then it is an internal transition: a validation or mapping that needs to be done as part of the owning state execution.

A transition can have more than one defined action. For each of these actions, the process evaluates a set of guard conditions that determines whether the flow processor follows its normal flow or performs an alternative flow. The processor executes the transition's actions in the order in which they appear in the transition definition.

A transition can take place only if departure events are defined to enable a state to move to another state. Every state in a transaction flow must have a departure event for the flow to complete successfully.

6.5.3 Drawing the flow

To draw a flow in the Flow editor, do the following steps:

1. Create a BTT flow and open it either double-clicking on it or selecting the Open option from the context menu.
2. In the Transaction editor, click the Flow tab (opened by default).
3. In the Palette pane, there are State and Transition widgets. You have to drop the states into the editor area by left-clicking in one of them, moving the cursor to the selected position and left-clicking again, and then connect the states using transitions.

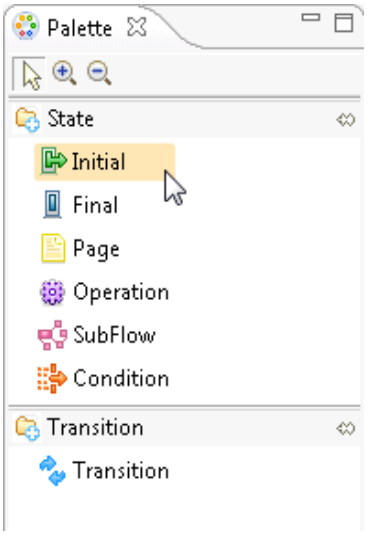
Note: Another convenient way to drag and draw states (only applicable to Page state, Operation state, SubFlow state), is to drag and drop the referenced file from the BTT Perspective Project Explorer to the editor area.

The following sections give you more details about how to build a flow: how to create and configure the different states and how to create and configure a transition.

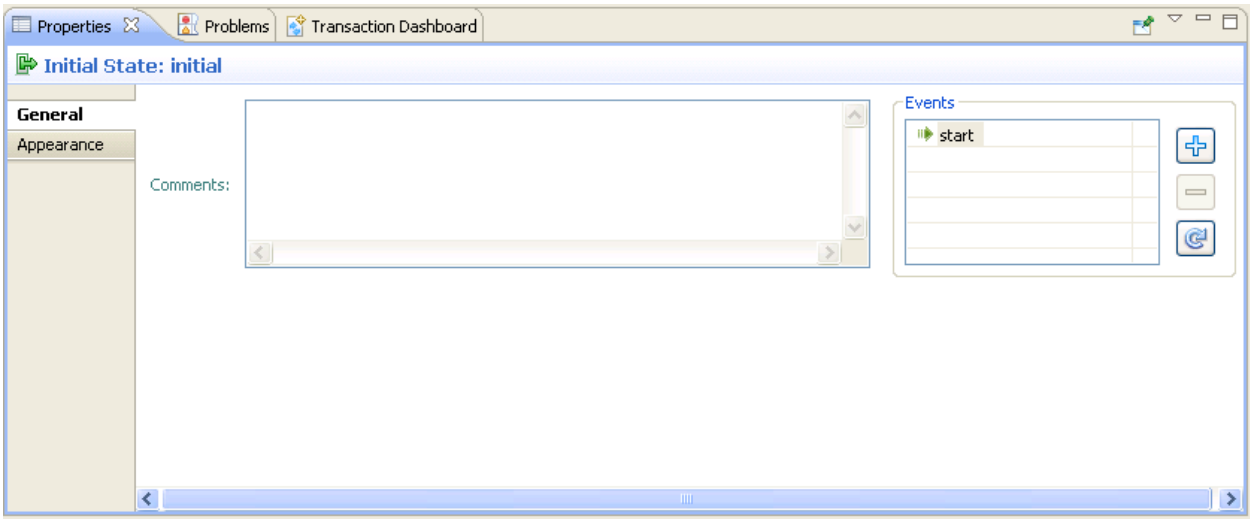
Creating the initial state

To create an initial state for a transaction, do the following steps:

- In the Palette pane, click Initial.



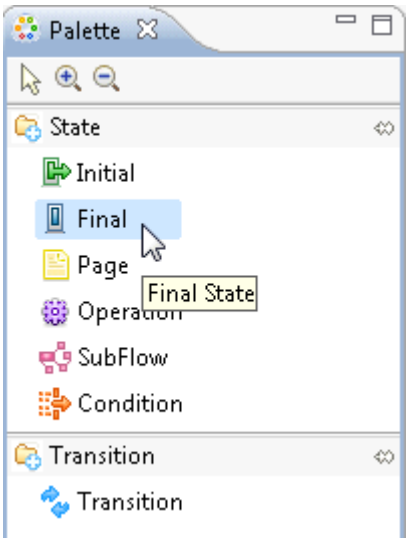
- In the Flow editor, click on the area at which you want to place the Initial state. If you want to change the location of the Initial state icon in the flow, drag and drop the Initial state icon to the desired location.
- By default, the ‘start’ event is generated in the state properties, as shown in the figure below



Creating a final state

To create a final state and add an exit event, do the following steps:

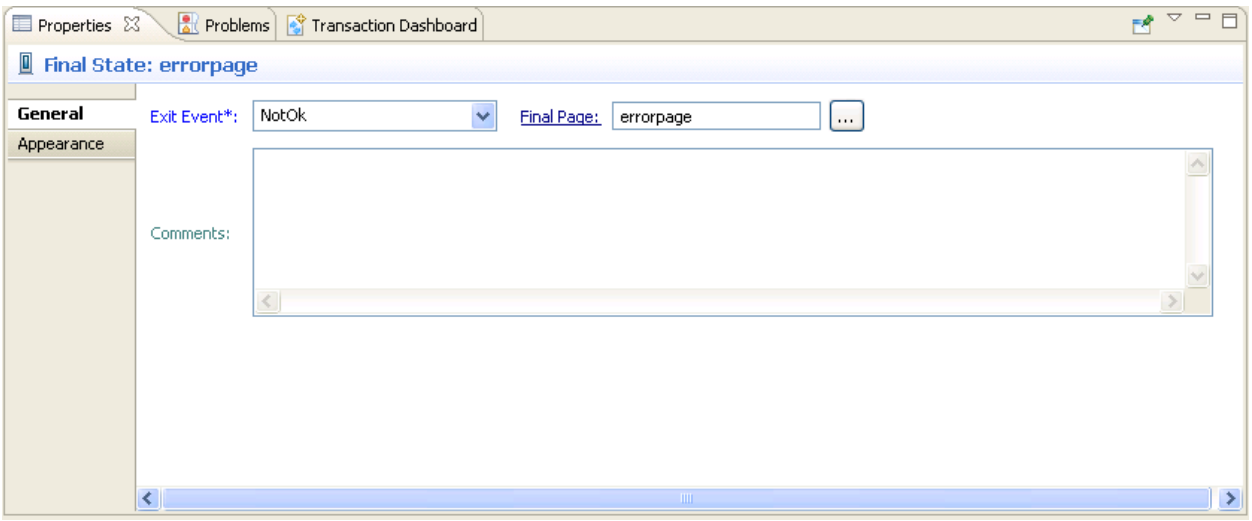
1. In the Palette pane, click Final.



2. In the Flow editor, click on the area at which you want to place the Final state icon. If you want to change the location of the Final state icon in the Flow tab, drag and drop the Final state icon to the desired location.

3. Click the Final state icon in the Flow tab.

4. In the Properties tab of the Final state, click the General tab.



5. In the Exit Event drop-down list of the General tab, select the exit event.
    - If the Final state has been reached because all the specified conditions of the transaction flow have been satisfied, select OK.
    - If the Final state has been reached because some of the specified conditions of the transaction flow have not been satisfied, select NotOK.

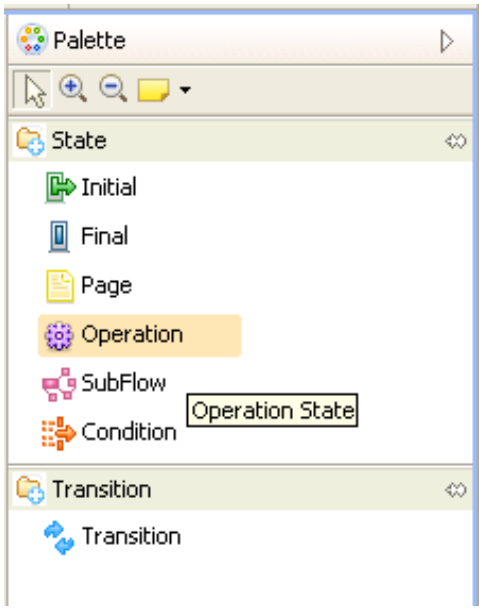
- Besides OK and NotOK default events, user can also enter other exit event by directly entering the event name.

6. Optional: If you want to add a page to the Final state, select the Browse icon  of the Final Page field in the General tab. You can edit the page by double-clicking in the Final Page label and open the selected page with the XUI editor.

Creating an operation state

To create an operation state, do the following steps:

1. In the Palette pane, click Operation.



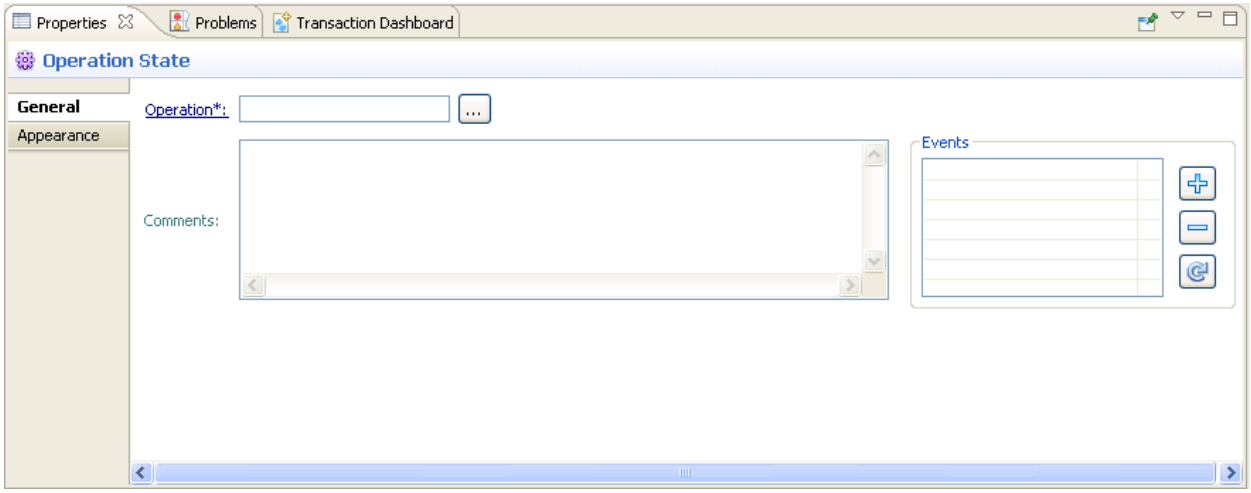
2. In the Flow editor, click the area on which you want to place the Operation state icon. If you want to change the location of the Operation state icon in the Flow tab, drag and drop the Operation state icon to the desired location.

   Alternatively, if the operation is already available, select the Operation from your BTT Project in the Project Explorer and drag and drop it to the desired location.

3. Bind an operation to the Operation state.

   a. Click the operation state icon in the flow.

   b. In the Properties tab of the Operation state, click the General tab.



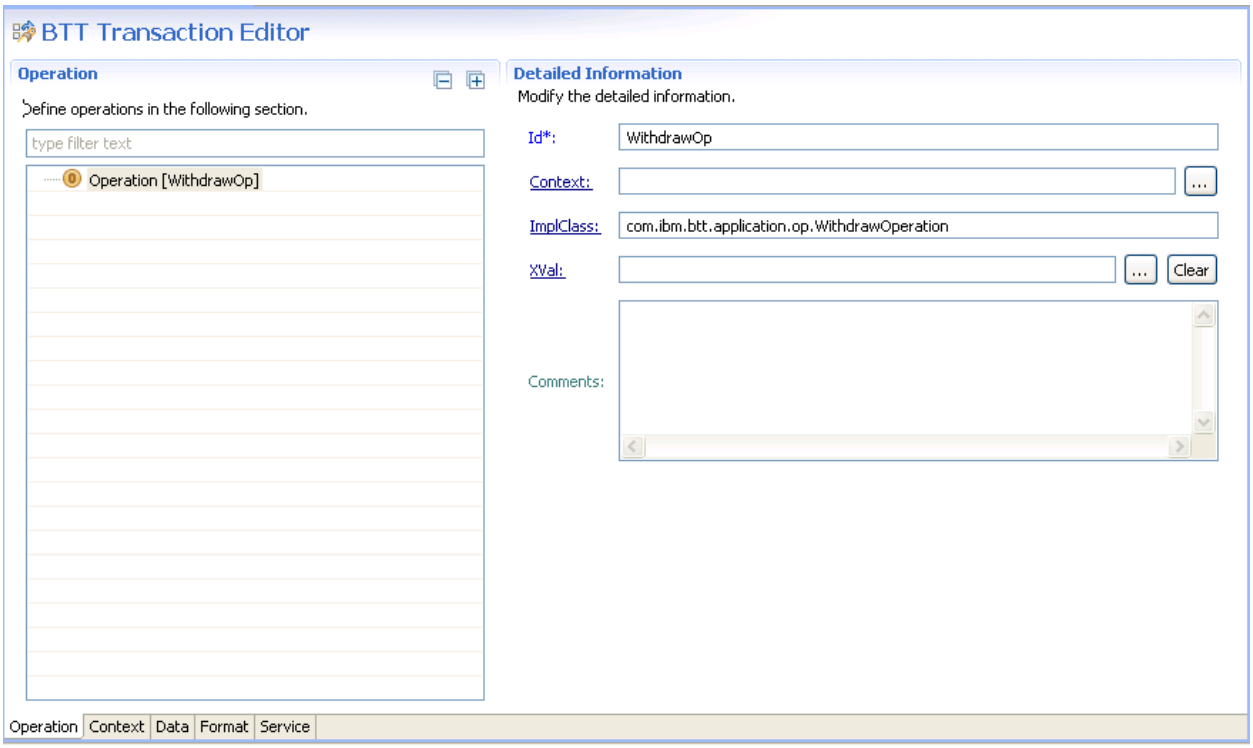
c. In the Operation field of the General tab, click the Browse icon  . The Select Operation window opens.

d. In the Select Operation window, select an operation for the Operation state, and then click OK.

Note: The operations have normally been already defined by the technical developers when defining the flow. If not, you will have to go first to the Operation tab of the Transaction Editor to create the operations that will participate in your flow. To understand how to create a new operation, go to section Creating an operation in the BTT Perspective

In the Events panel of the General tab, click the Add icon  to add events to the operation. The events will appear automatically if the EVENT annotation has been used in the Java code of the operation, as commented in section Operation State

4. If the events are not automatically populated or you have to implement the operation, edit the related Java file by clicking on the Operation label in the General panel. The Operation editor will appear and you can click on the ImplClass label to open the Java code editor.



Shown below is an example of sample code from a .java file:

```
OperationRepliedEvent event = new OperationRepliedEvent(this);
Hashtable ht = new Hashtable();
event.setParameters(ht);
if (balance >= debt)
{
    ht.put(com.ibm.btt.automaton.ext.DSEOperationState.EXIT_EVENT_NAME,"ok");
} else
{
    ht.put(com.ibm.btt.automaton.ext.DSEOperationState.EXIT_EVENT_NAME,"error");
}
fireHandleOperationRepliedEvent(event);
```

In the example above, "`ok`" and "`error`" are the names of the events that should be defined in the Operation state.

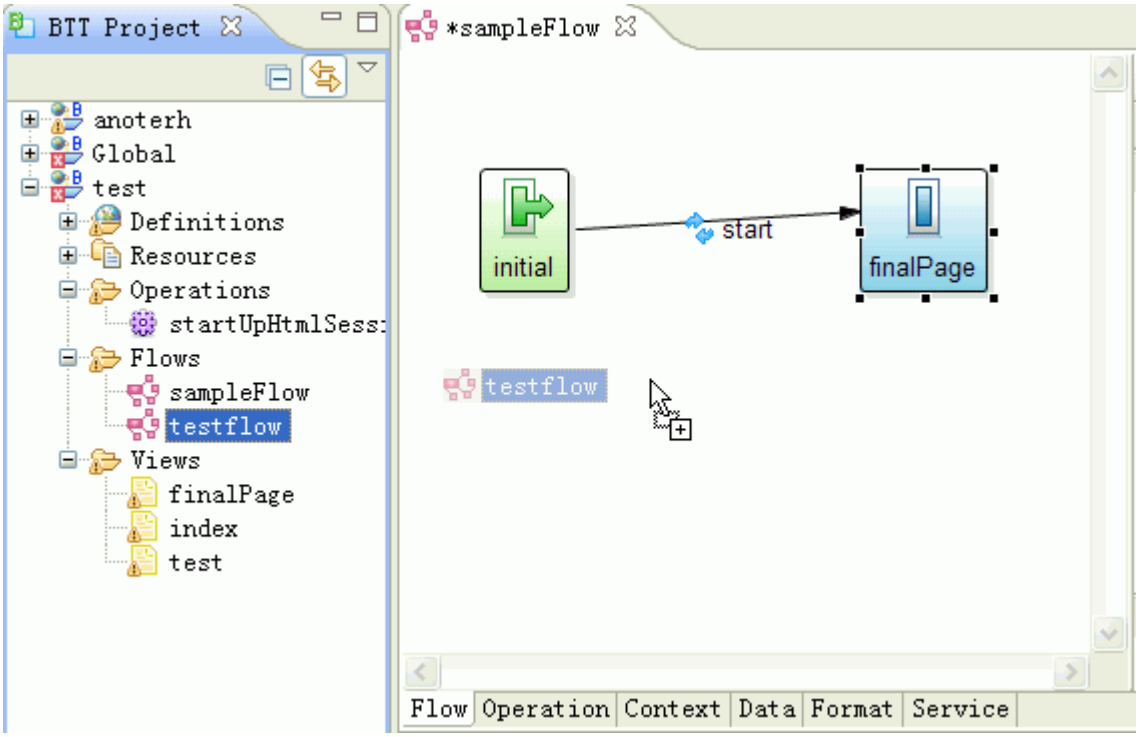
Creating a sub-flow state

There are two ways to create a SubFlow state:

- You can drag an existing flow from the Project Explorer pane to the graphical editing area in the Transaction editor (Flow editor) as shown in *Figure 1*. This can only be done for flows belonging to the same project. The properties and events that have been configured in the flow are copied to the SubFlow state,

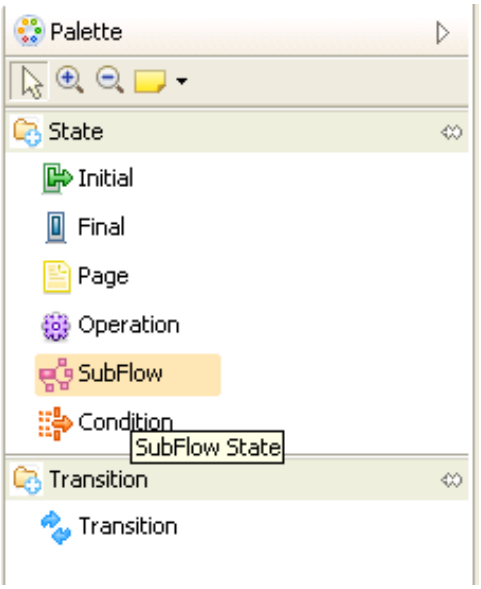
and the name of the transaction flow is displayed on the label of the SubFlow state.

Figure 1. Creating a SubFlow state by dragging a flow to the Flow editor



- You can drag and drop a SubFlow state from the Palette to the Flow editor. In this case you should do the following steps:

  1. In the Palette pane, click SubFlow.



  2. In the Flow tab of the Transaction editor, click the area on which you want to place the SubFlow state icon. If you want to change the location of the

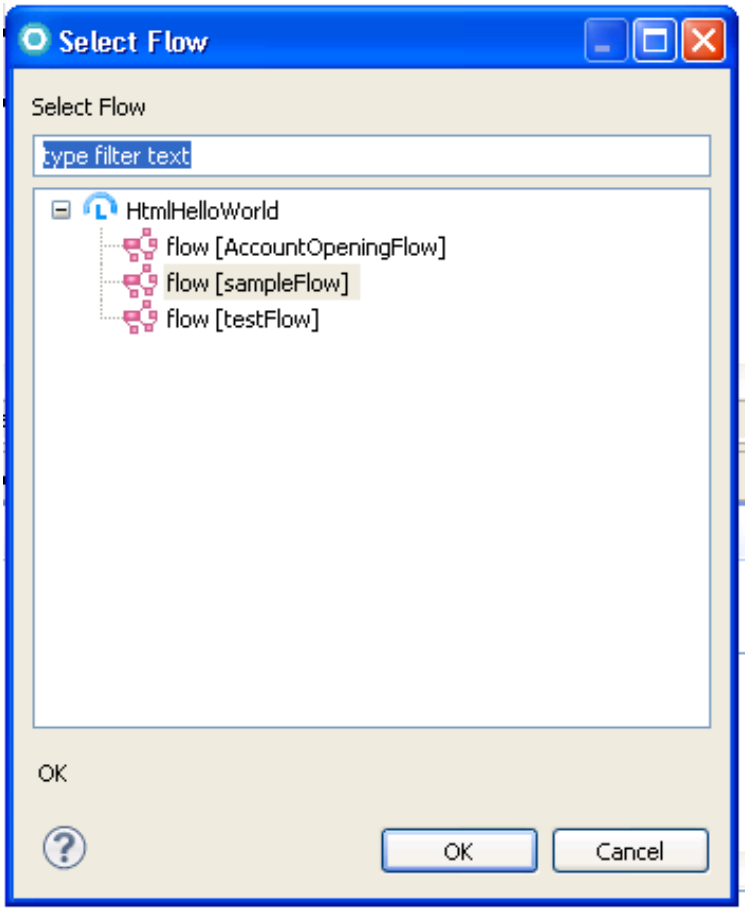
SubFlow state icon in the Flow tab, drag and drop the SubFlow state icon to the desired location.

3. Bind a flow to the SubFlow state:

    a. Click the SubFlow state icon in the Flow tab.

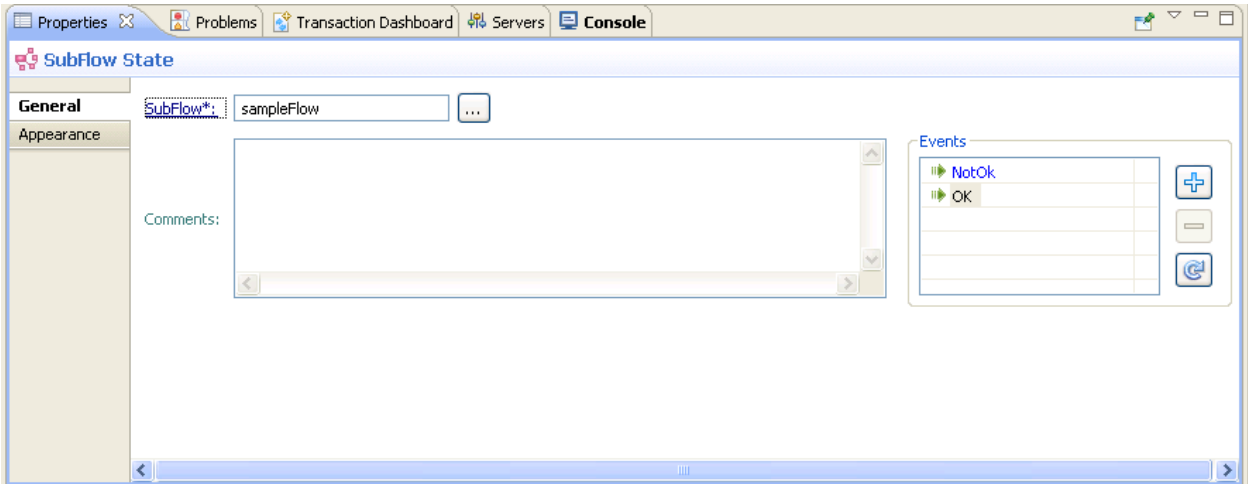
    b. In the Properties tab of the SubFlow state, click the General tab.

    c. In the Subflow field of the General tab, click the Browse icon  . The Select Flow window opens.



    d. In the Select Flow window, select the flow that you want to define as the sub-flow of the transaction flow. Click OK.

4. The exit events of the sub-flow are automatically added to the Events panel of the General tab.

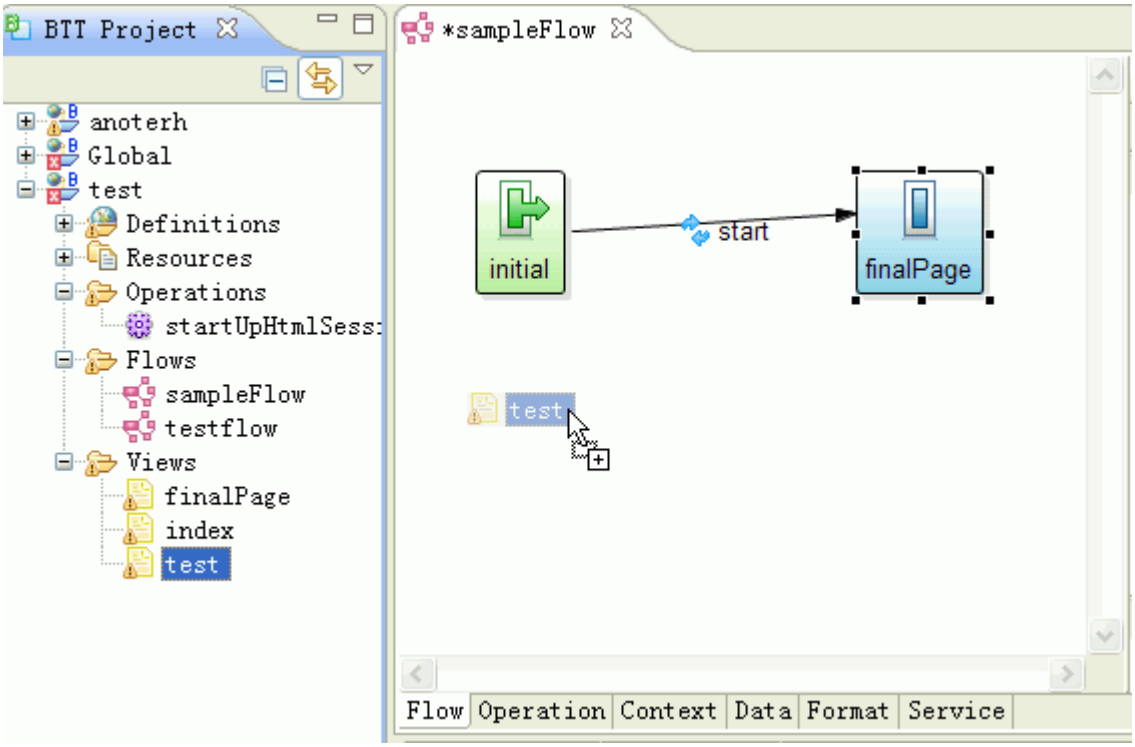


Creating a page state

There are two ways to create a Page state:

- you can drag an existing view from the Project Explorer pane to the Flow tab of the Transaction editor (Flow editor) as shown in *Figure 1*. All the properties and events of the XUI file are copied to the Page state, and the name of the view is displayed on the label of the Page state.

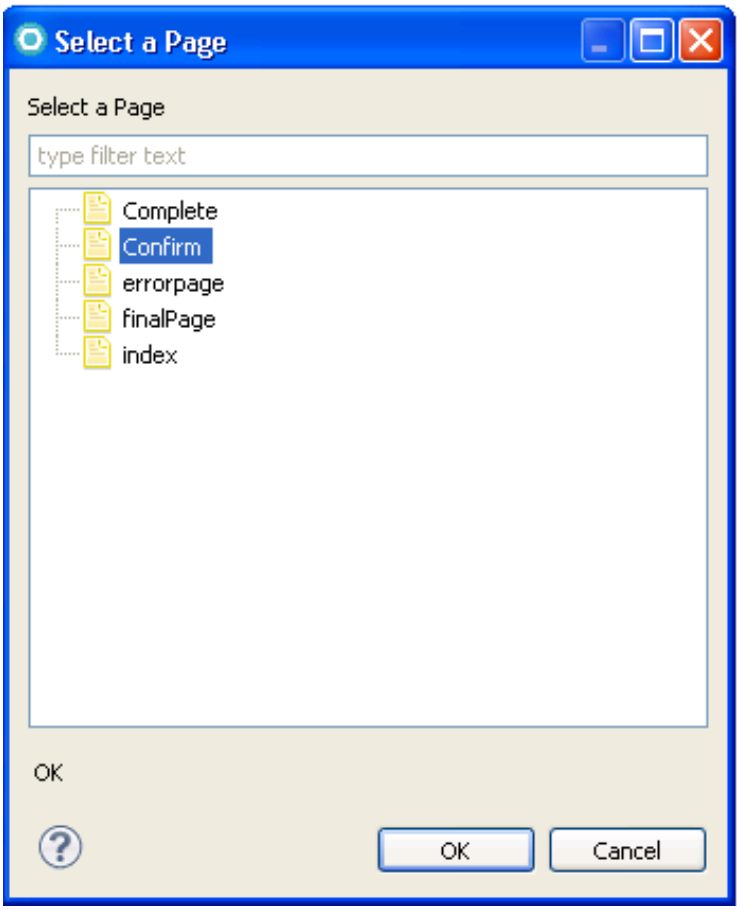
  Figure 1. Creating a Page state by dragging a view into the Flow editor



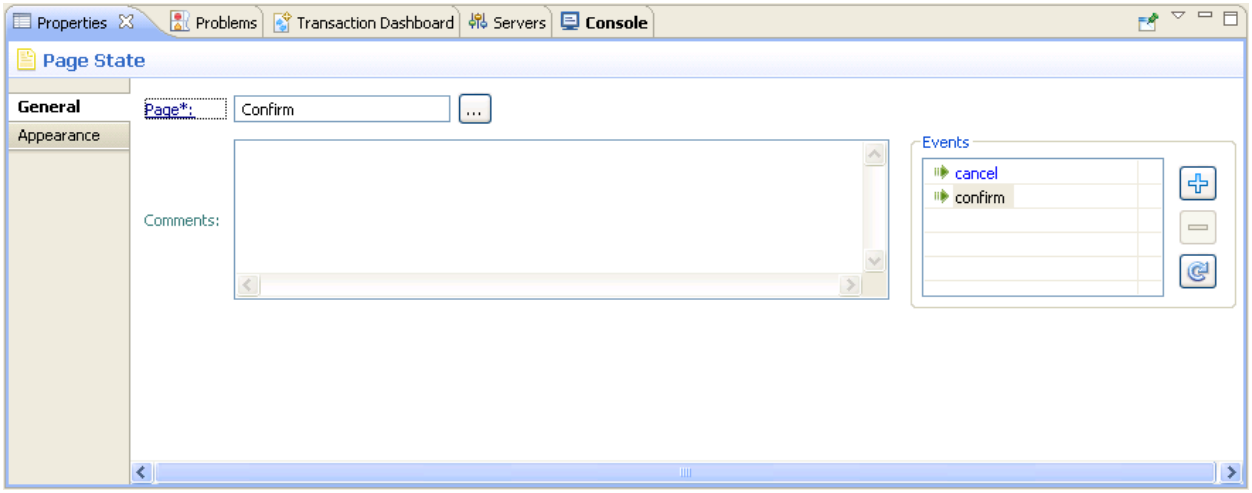
- You can drag and drop a Page state from the Palette to the Flow editor. In this case you should do the following steps:

  1. In the Palette pane, click Page.

2. In the Flow tab of the Transaction editor, click the area on which you want to place the Page state icon. If you want to change the location of the Page state icon in the Flow tab, drag and drop the Page state icon to the desired location.
3. Bind an XUI page to the Page state.
    a. Click the Page state icon in the Flow tab.
    b. In the Properties tab of the Page state, click the General tab.
    c. In the Page field of the General tab, click the Browse icon  . The Select Page window displays.



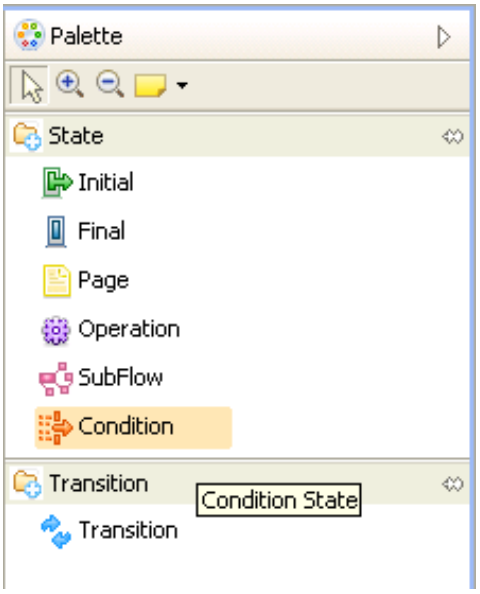
    d. In the Select Page window the views in your BTT project are prompted. Select the view that you want to be displayed in the browser for the page state that you are currently configuring. Click OK.
4. The widget events of the view are automatically added to the Events panel of the General tab, so that the next states in the transaction flow can be navigated to. You can also click the Add icon  to add other events to the Page state if needed.



Creating a condition state

To create an operation state, do the following steps:

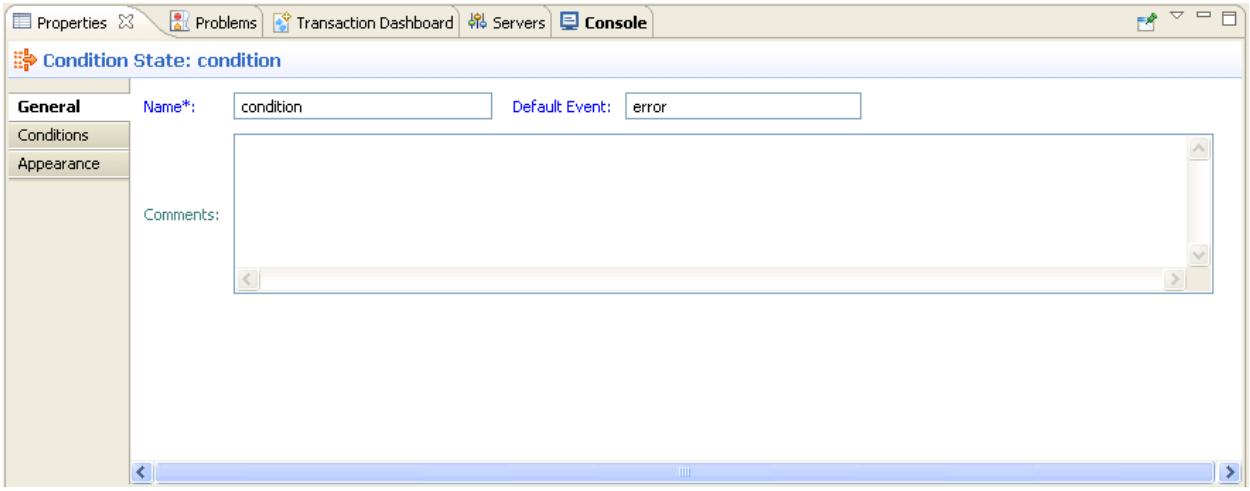
1. In the Palette pane, click Condition.



2. In the Flow editor, click on the area at which you want to place the Condition state icon. If you want to change the location of the Condition state icon in the Flow tab, drag and drop the Condition state icon to the desired location.

3. Click the Condition state icon in the Flow tab.

4. In the Properties tab of the Condition state, click the General tab. Set the Name for the Condition state and the Default Event, the event that is fired when no condition in the Condition state is satisfied.



5. In the Conditions tab, add conditions for the condition state. A condition can be defined either as a function or as an expression. For information on how to create and define conditions, refer to the following sections:

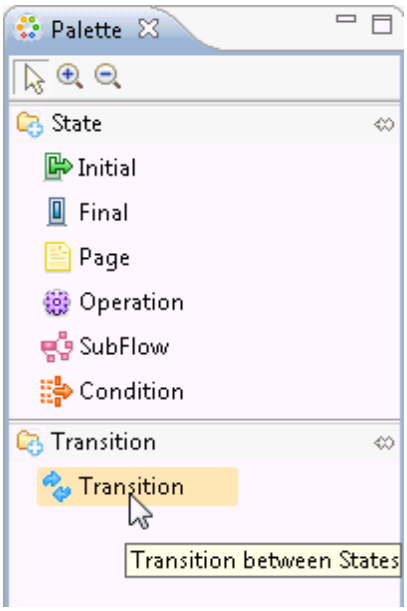
    - Defining conditions using a Global Function
      This section describes how to define a condition using BTT global functions.

    - Defining conditions using a Expression
      This section describes how to create define a condition using an expression.

Connecting states using transitions

To define a transition, do the following steps:

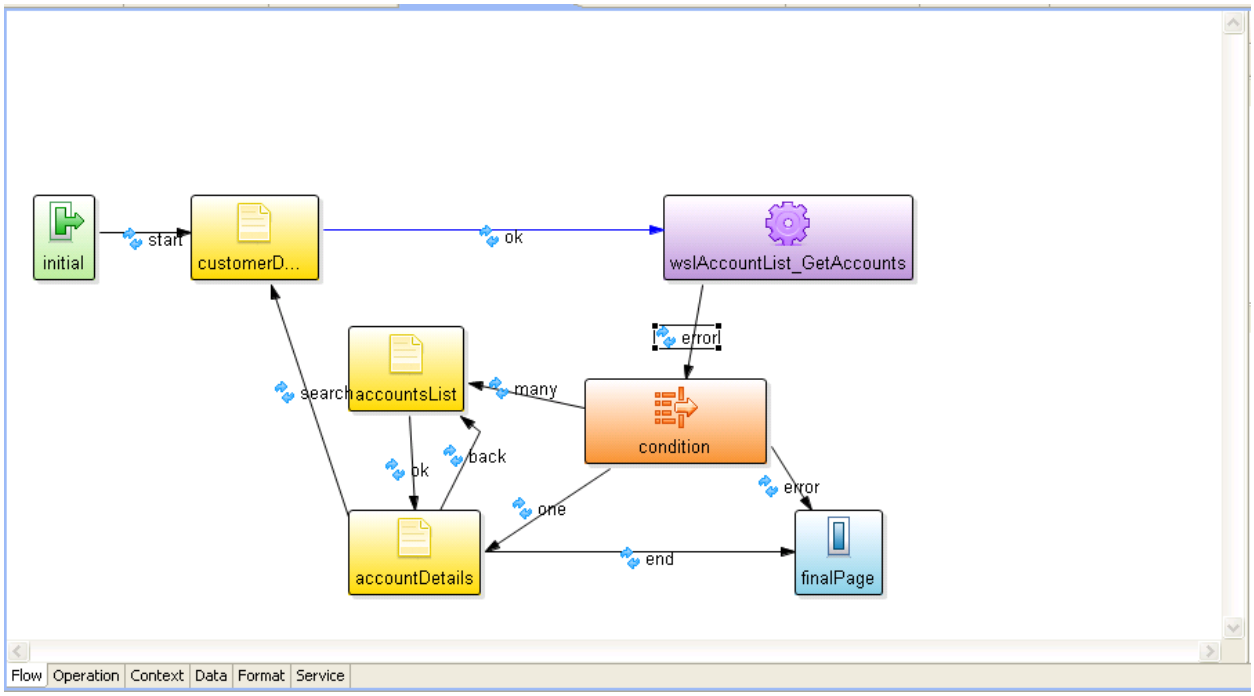
1. In the Palette pane, click Transition.



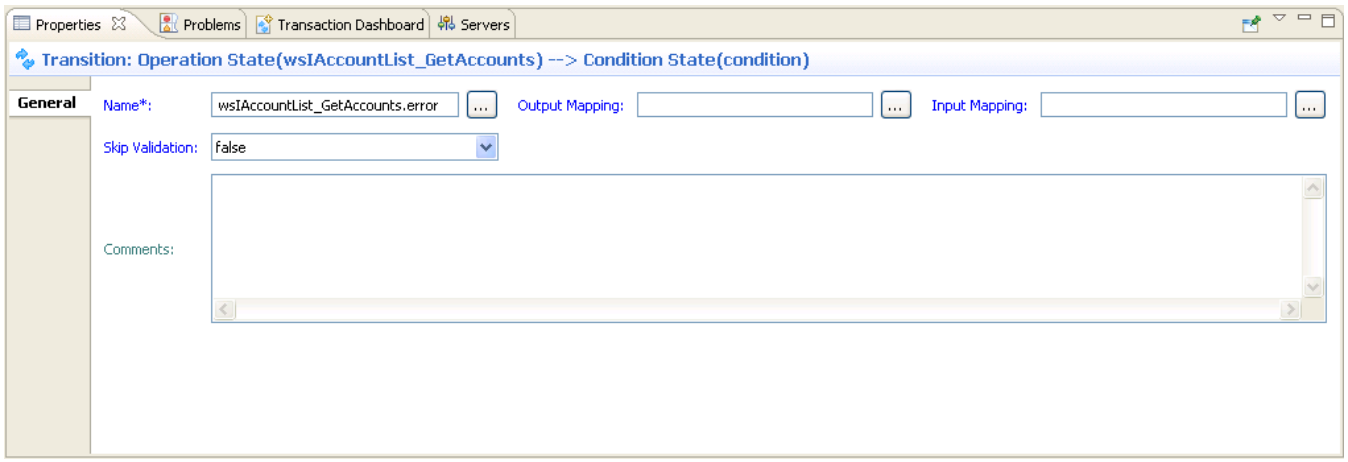
2. In the Flow tab, click and drag from one state to the next state in the flow.
3. Add an event to the transition:

   Note: If only one event is specified for a state, the event that enables a transition to take place is specified automatically; however, if more than one event has been specified for a state, the first one in alphabetical order is selected by default but you can change the event to the one that is required for the state to transition to the next state.

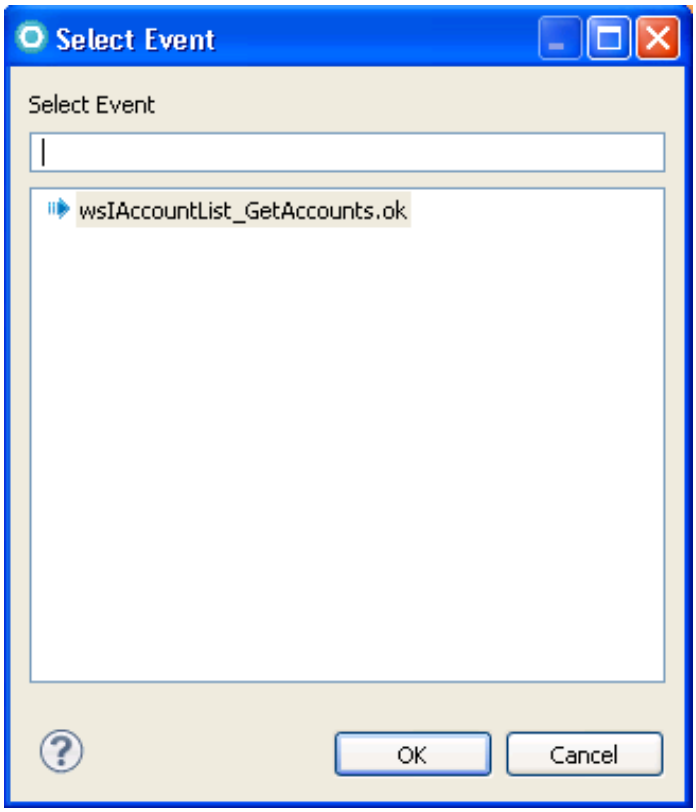
   a. Click the transition icon of the transition in the Flow tab.



   b. The Transition properties panel opens



c. In the General tab of the transition properties, click the Browse icon  of the Name field. The Select Events window displays.



d. In the Select Events window, select the event that you want to add to the transition, and then click OK.

4. Add the data mapping to the transition. Refer to section Setting data mappings for transitions for a detailed explanation on how data mappings are defined and how they work.

6.6 Managing the flow context

A context is an object that defines and encapsulates data for a functional or business entity.

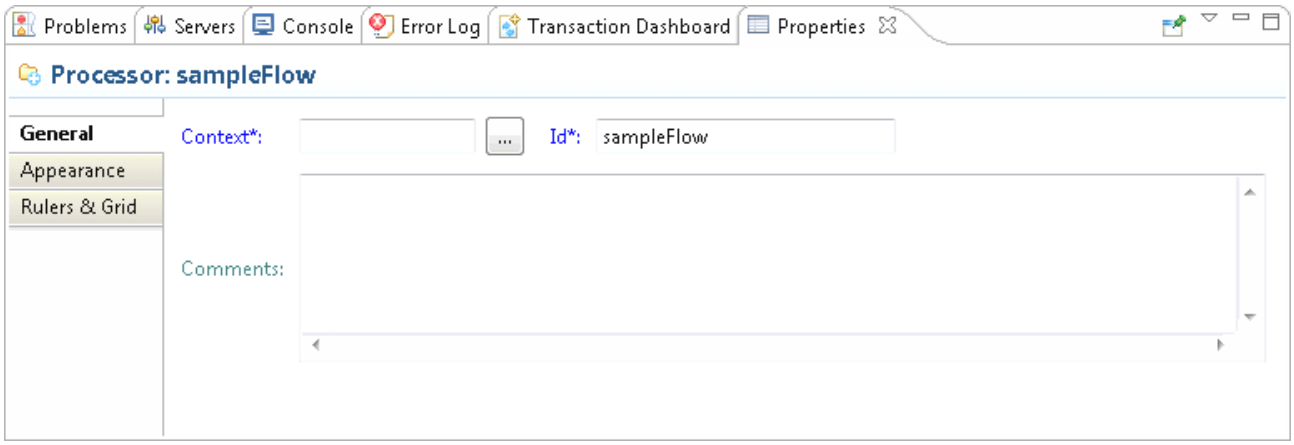
In the presentation layer of BTT, contexts are used to define and encapsulate data at the branch level (a branch context), for workstations (workstation context), for users (user context), and for customer sessions (customer session context).

A context is bound to a transaction, either an operation or a flow. The data of a context that has been bound to a flow is shared across the entire transaction flow . As part of the flow definition, you can select the context that you want to bind to a transaction flow by following the next steps:

1. In the Transaction editor, click the Flow tab.

2. Click in the empty space of the Flow tab to open the Properties tab for the transaction flow.

3. In the Properties tab, click the General tab.



4. In the General tab, click the Browse icon  to select a context. The Select Context window opens.

5. In the Select Context window, select the context that you want to bind to the transaction flow, and then click OK.

In a flow, the different states (operations, pages and sub-flows) may have their own context. The flow context can be used as an intermediate container to pass data from one state context to the other. This data mapping is done as part of the transition definition and is described in detail in next section.

6.7 Setting data mapping for transitions

In situations where you want to manipulate data from flow context to transiting state context and vice versa, you can define the map format to the transition. The format can be from a context to a context, and it can also be from a complex expression to a context data. You can define the data maps by using the data map tool. After creating the data maps the corresponding formatters are created and displayed in the Format editor (Format tab in the Transaction editor).

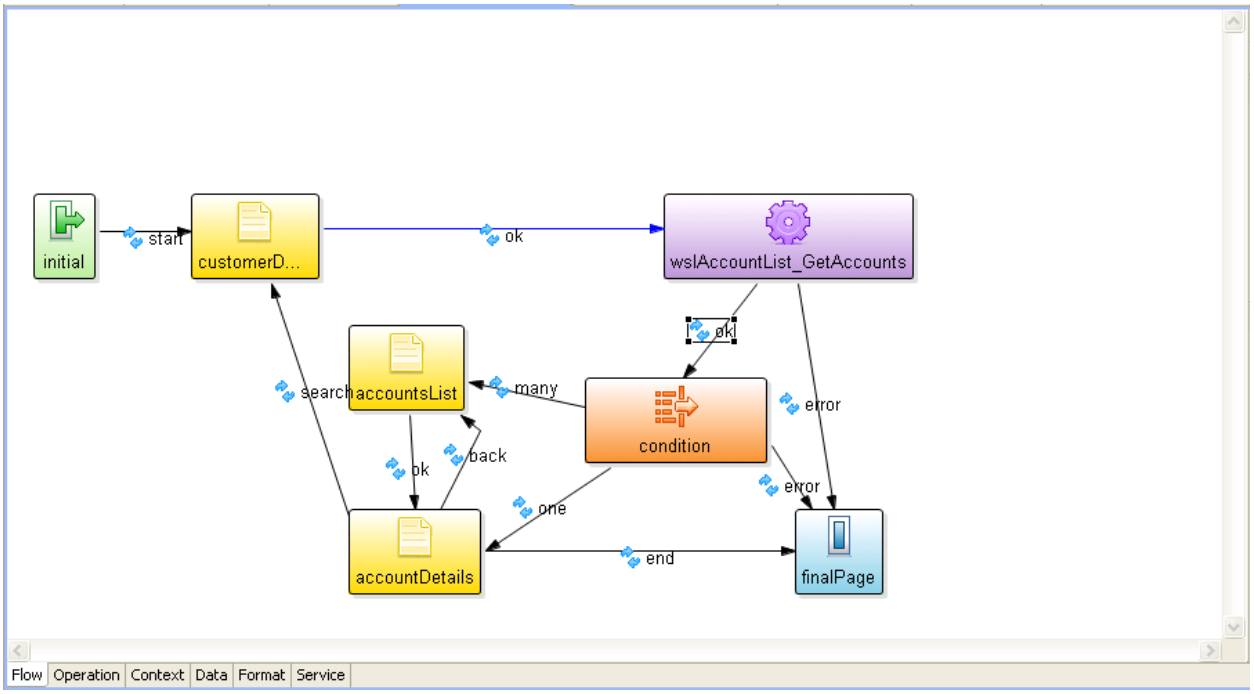
For one transition, there are two types of map format:

- **Output data format**: map from an expression to the flow context. The expression can be a constant, expression, function or source state context.
- **Input data format**: map from an expression to the target state context. The expression can be a constant, expression, function or flow context

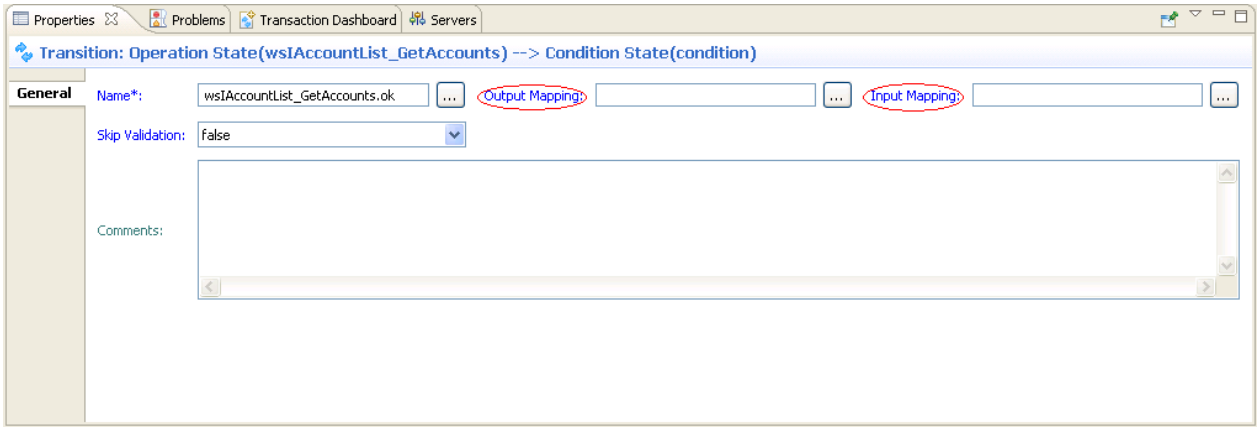
6.7.1 Adding data mapping

To add a data mapping to a transition, do the following steps:

1. In the Flow editor, click the transition to which you want to add the data mapping.



2. The Transition properties panel opens. Two fields are used to define the data mapping: the **Output Mapping** field and **the Input Mapping** field.

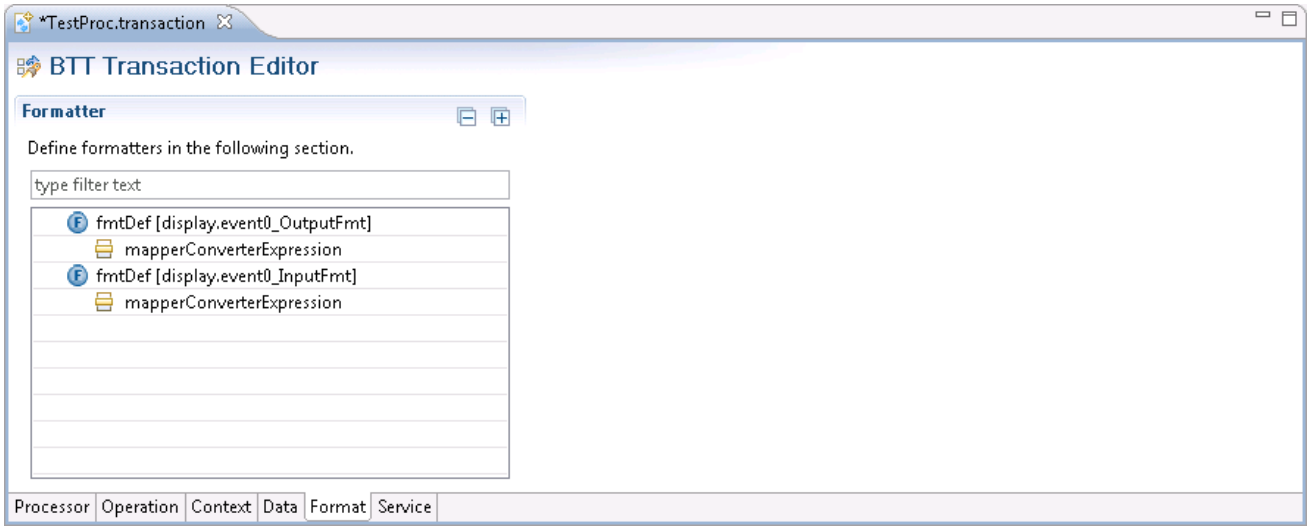


If you are defining a mapping for the transition source state output data, the data resulting from the source state execution, enter a name in the **Output Mapping** field and click the Browse icon  .

If you are defining a mapping to provide the right input data to the transition target state, click the Browse icon  in the **Input Mapping** field. The Data Mapping window opens.

The Data Mapping window opens. The entered data mapping name will be the name of the data formatter automatically generated after the mapping definition. Create the data mappings following the indications described in the next section, Defining data mappings and then click OK.

3. As result of the data mappings definition, the data formatters that are going to be used to perform the mapping during the transaction execution are automatically created. They can be displayed in the Format editor (Format tab of the Transaction editor)



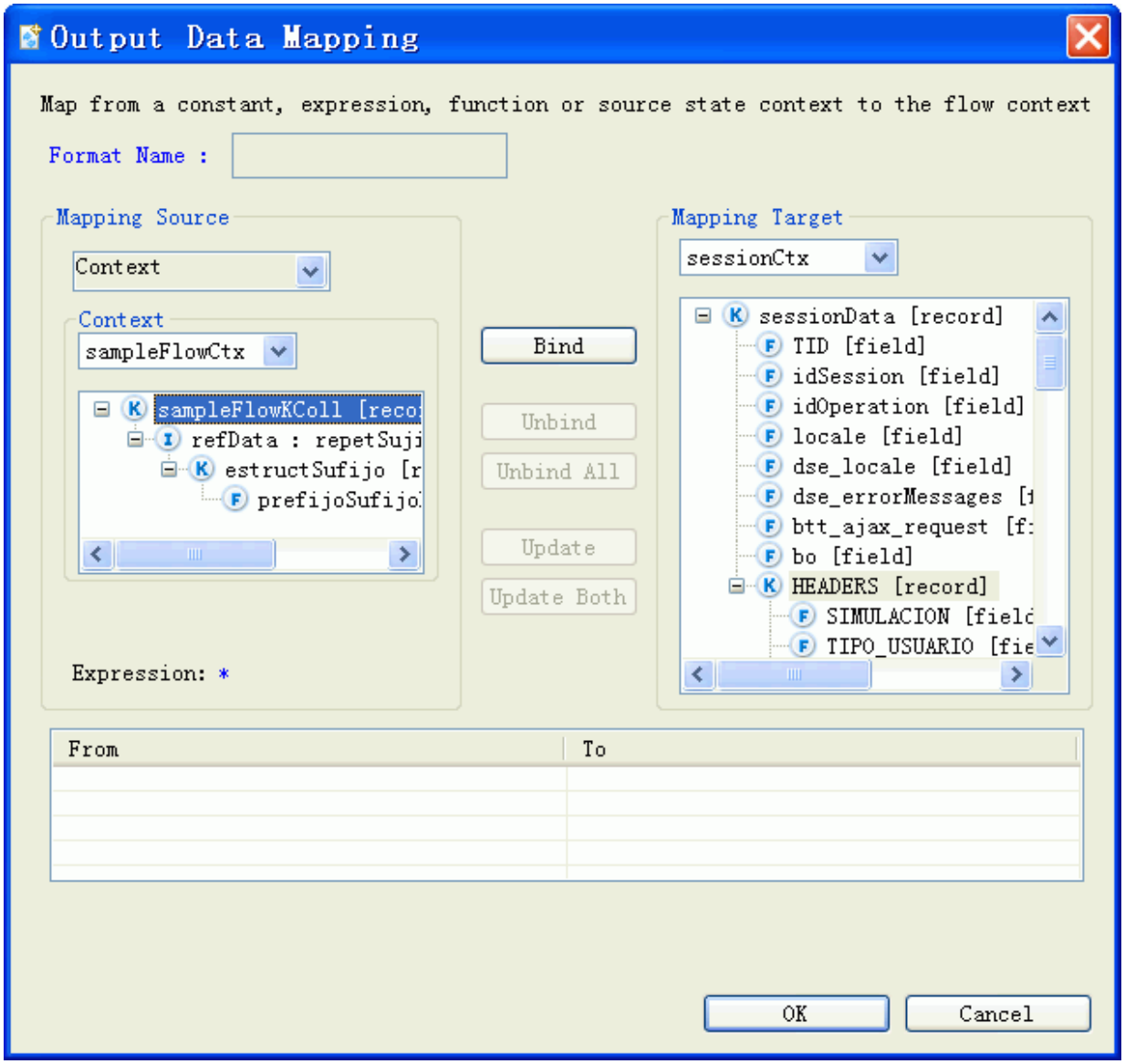
6.7.2 Defining data mappings

This section describes how to define data mappings by using the Data Mapping window.

The toolkit provides the following two data mapping windows for you to define data mappings:

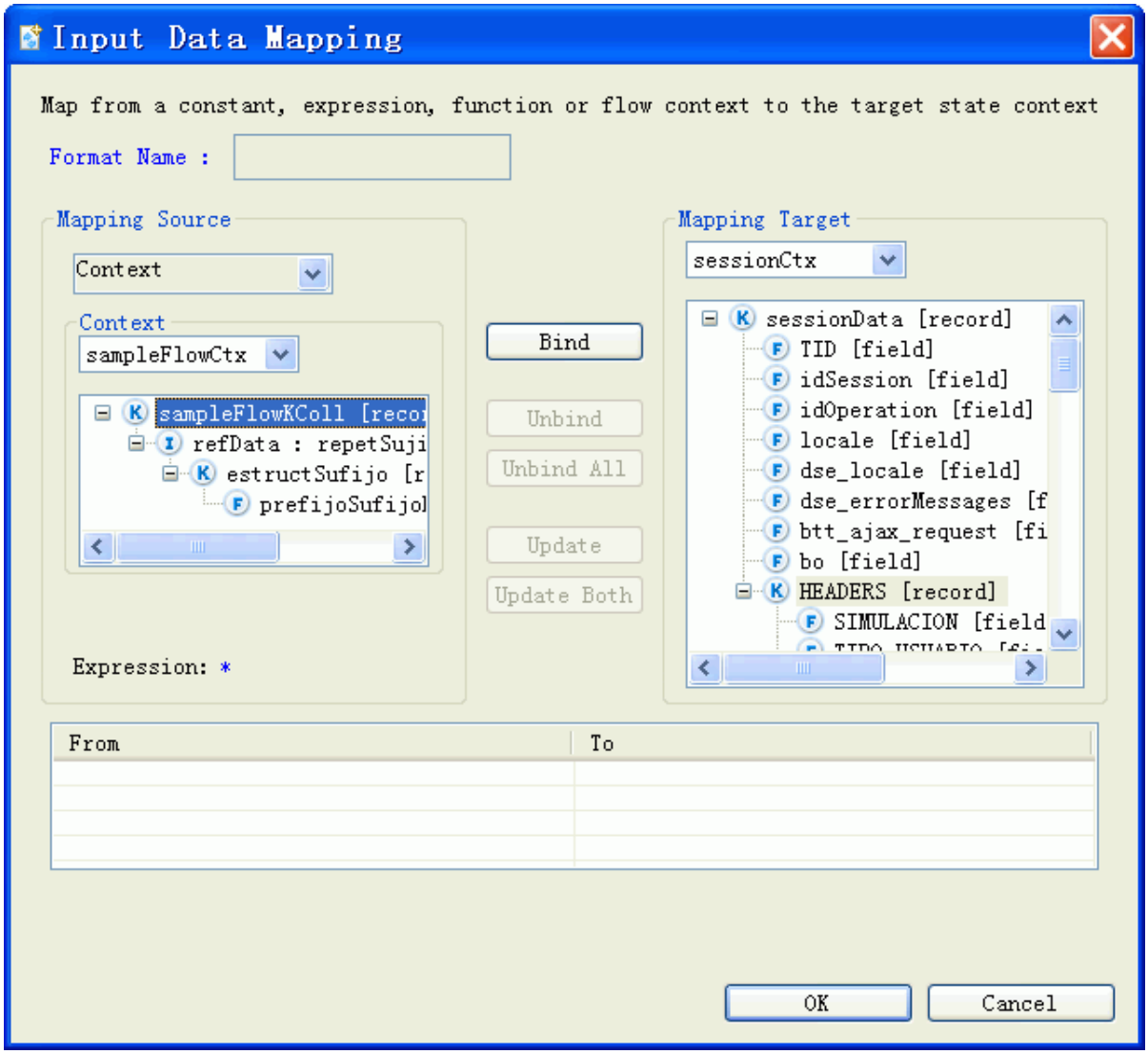
- The Output Data Mapping window opens when you click the Browse icon ... of the Output Mapping field in the Properties tab of the transition. Use the Output Data Mapping window to map a constant, expression, global function, or the context of the source state to the flow context. *Figure 1* shows the Output Data Mapping window. The **Mapping Source** panel of the window contains the data that refers to the **source state** and the **Mapping Target** panel of the window contains the data that refers to the **flow context**.

Figure 1. The Output Data Mapping window



- The Input Data Mapping window opens when you click the Browse icon  of the Input Mapping field in the Properties tab of the transition. Use the Input Data Mapping window to map a constant, expression, global function, or the flow context to the context of the target state. *Figure 2* shows the Input Data Mapping window. The **Mapping Source** panel of the window contains the data that refers to the **flow context** while the **Mapping Target** panel of the window contains the data that refers to the **target state.**

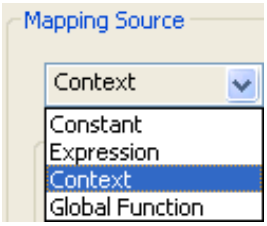
Figure 2. The Input Data Mapping window



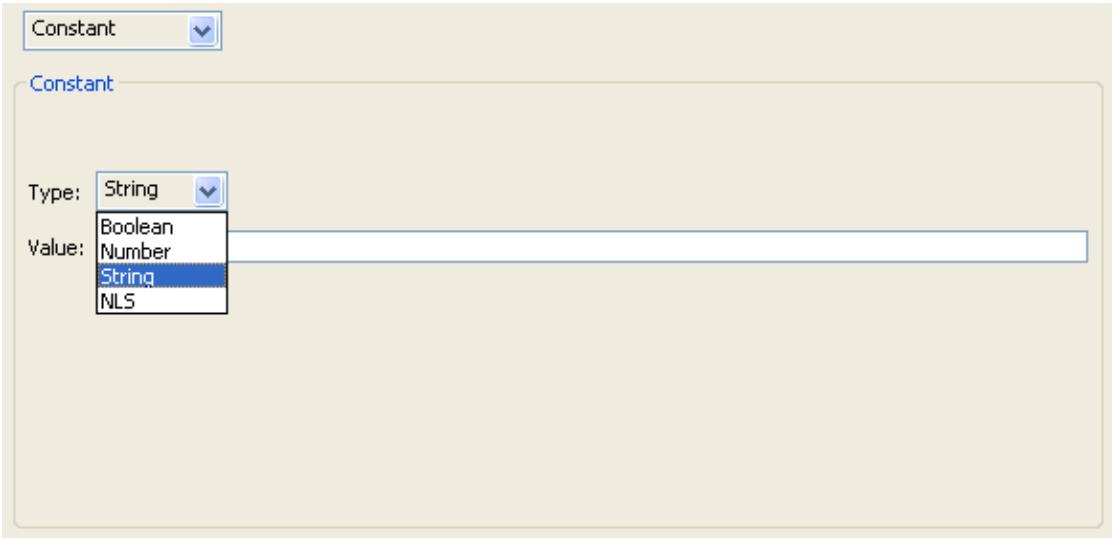
The procedures for using the Input Data Mapping window and the Output Data Mapping window are the same. One transition can have maximum one Output Data Mapping and one Input Data Mapping, although as part of either the Output or Input Data Mapping you can define as many data mappings as you want; the processor, when changing from the source state to the target state, will call the different defined formatters sequentially. To create a new Input or Output Data Mapping and associated to the selected transition, enter the name in the corresponding field and click the Browse icon  .

Mapping Source definition

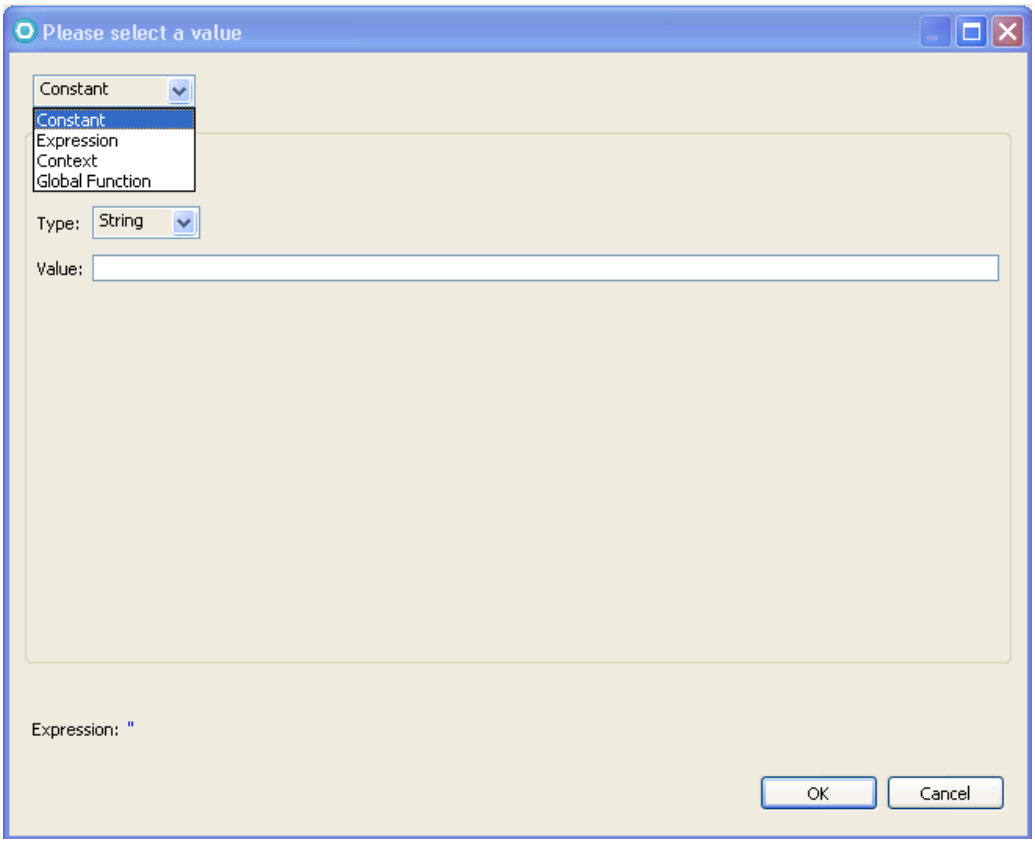
In the **Mapping Source** panel of the Data Mapping window, use the top drop-down list to specify whether the data that is being mapped from is a constant, expression, global function, or from a context.



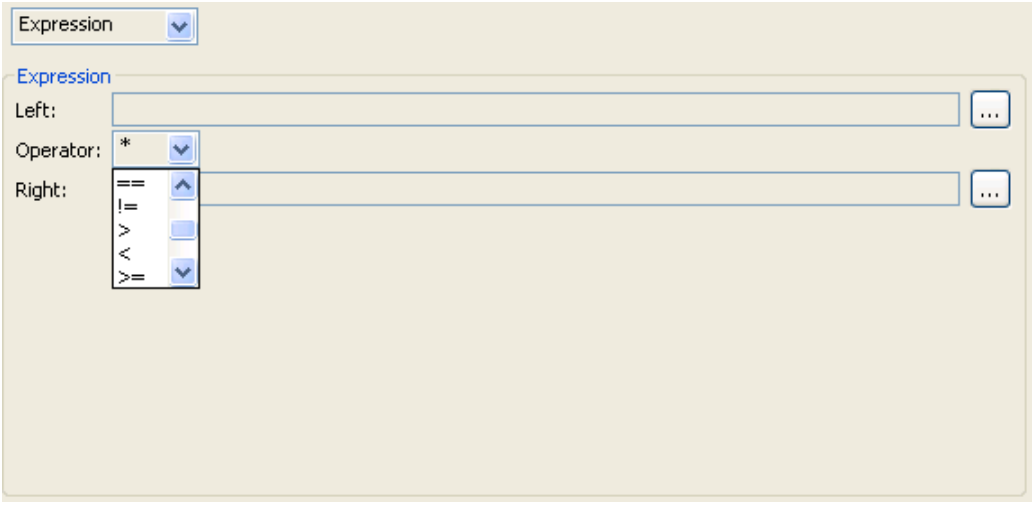
- If you select **Constant**, the Constant panel displays under the drop-down list in which you must specify the constant. String, Number, Boolean and NLS types are supported.



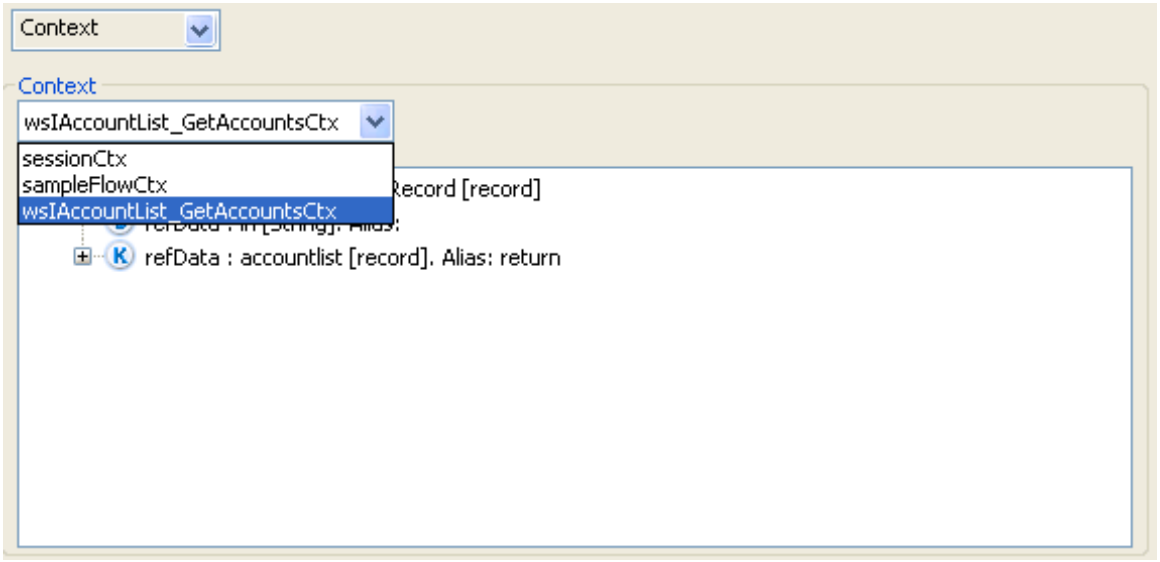
- If you select **Expression**, the Expression panel displays under the drop-down list in which you must specify the expression. You can define expressions with one or two operands. These operands can be constants, expressions, functions or context values. When you click the Browse  icon to create a new operand in either the Left or Right field, you open the expression editor.

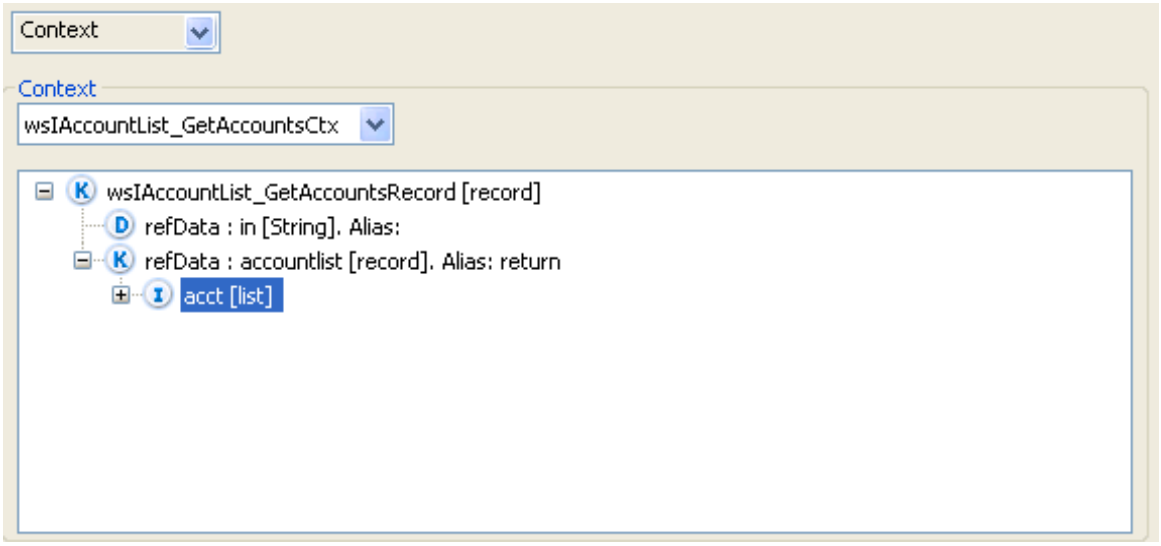


Note that this is the same wizard that the one embedded in the Mapping Source panel. You can then define your operand and click OK to go back to the expression definition editor. An expression supports arithmetic, comparison and boolean operators, as shown in the figure below.



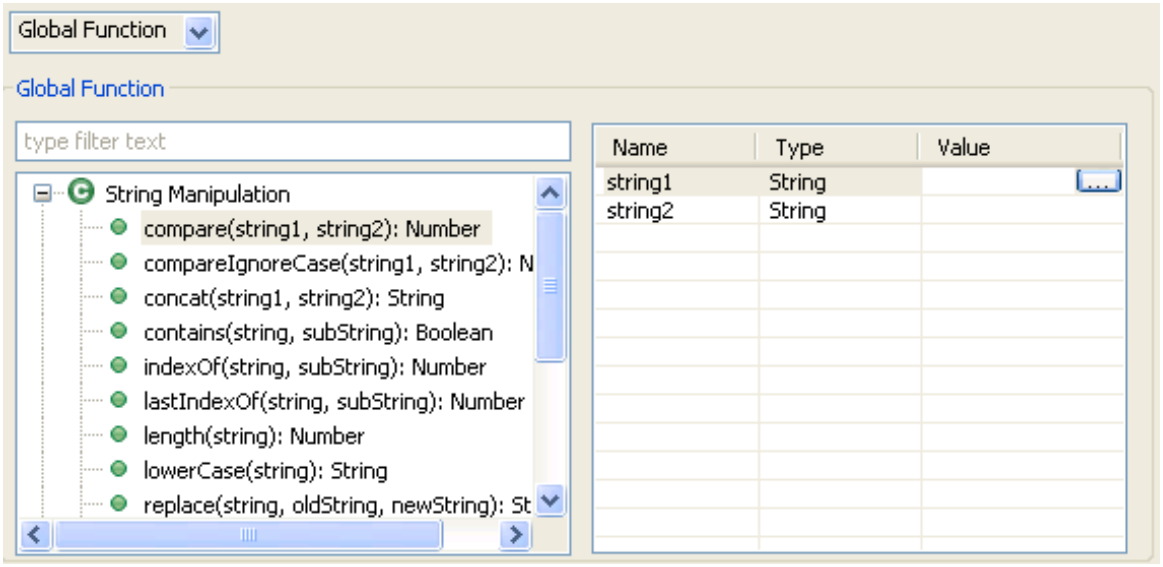
- If you select **Context**, the Context panel displays under the drop-down list in which you must select the Context and the data element in this Context that requires mapping.



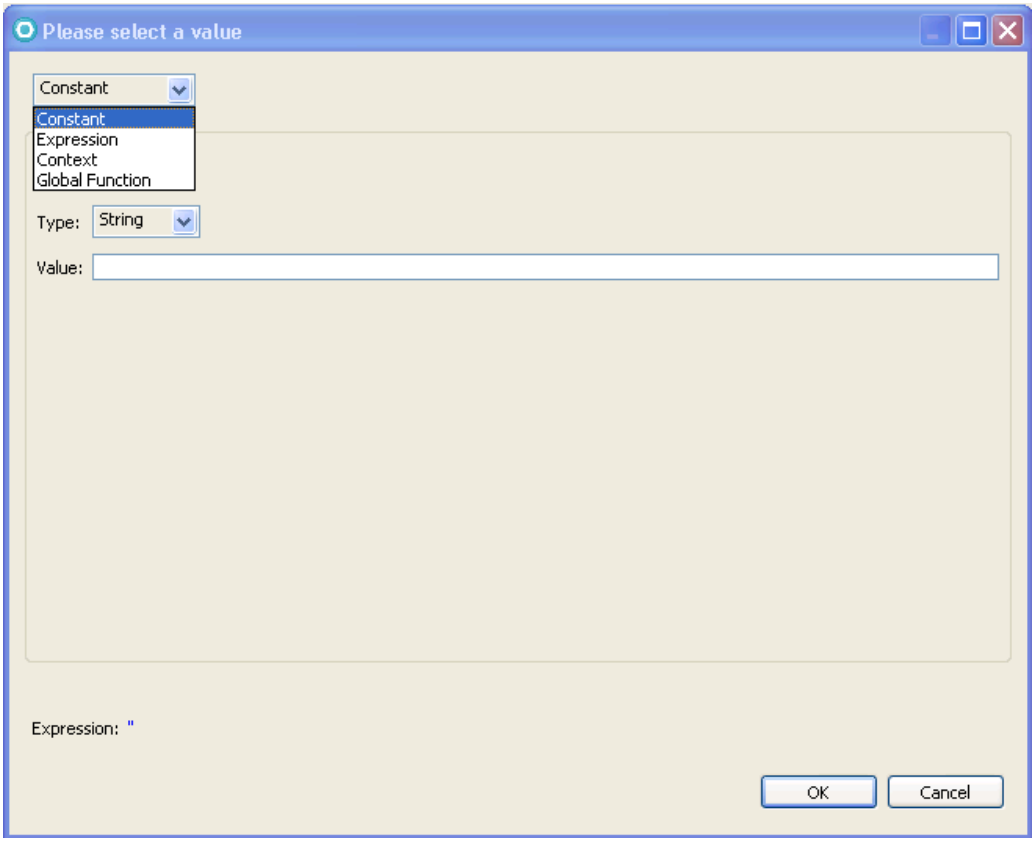


The contexts available for selection depend on the type of state that is the source of the transition. Refer to section Context mappings for a detailed list of contexts associated to a flow state.

- If you select **Global Function**, the Global Function panel displays under the drop-down list in which you must select the function that request mapping. The result of the function execution is the data that will be mapped to the target data.



Each function has a set of parameters that must be specified. These parameters can be constants, expressions, functions or context values. The values can be set using the expression editor, as shown in the figure below, and it is opened when clicking in the Browse icon  in the Value column of the function parameters table.

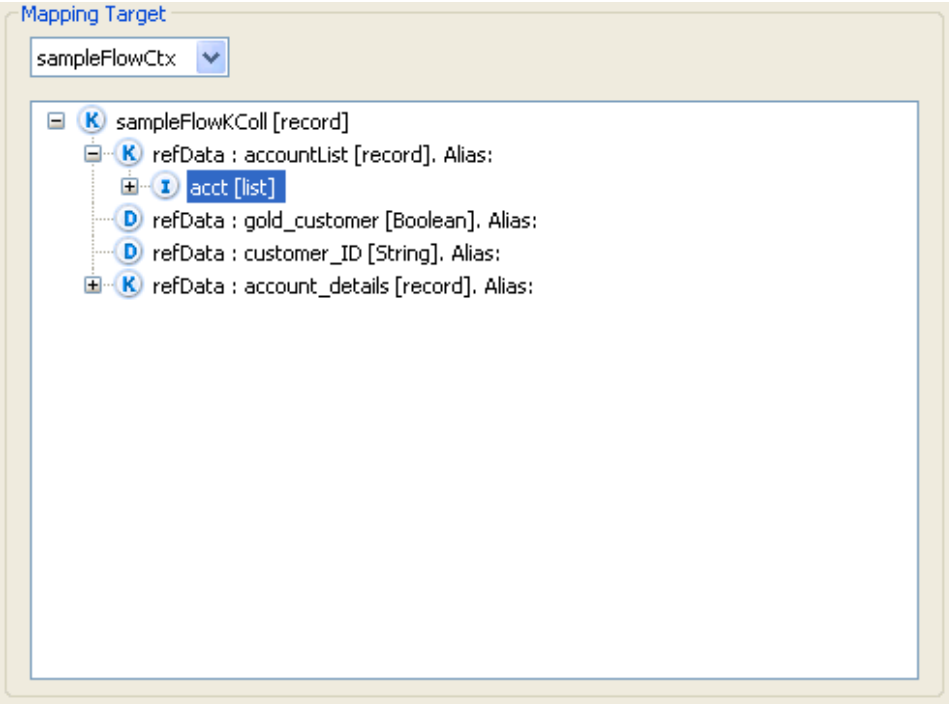


In all cases, the final expression is shown at the bottom of the Mapping Source panel, in the Expression field.

Mapping Target definition

In the **Mapping Target** panel of the Data Mapping window, specify the data from one of the visible contexts that the data formatter is mapping the source data to. Refer to section Context mappings for a detailed list of contexts associated to a flow state.

If you want to specify more than one data element at a time, click Shift and select the data elements.



Context mappings

If you are mapping contexts, the items in the Context panels of the Data Mapping windows will be displayed as described in the following list:

- The Mapping Context panel of the Input Data Mapping window and the Mapping Target panel of the Output Data mapping window displays the following items:

  - Local context (if it exists)
  - Session context (if it exists) and all of the parent contexts of the session context. The session context is the context with type=”session”. - Root context (the context with type=”root”)

- The right Context panel of the Input Data Mapping window and the left panel of the Output Data mapping window displays the following items:

  - For a SubFlow state:

- Sub-flow context
- Local context (if it exists)
- Session context (if it exists) and all its parent hierarchy. Session context is the context with type=”session”.
- Root context (the context with type=”root”)

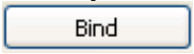
- For a target operation state:

  - Operation context (in case it exists)
  - Local context (if it exists)
  - Session context (if it exists) and all its parent hierarchy. Session context is the context with type=”session”
  - Root context (the context with type=”root”)

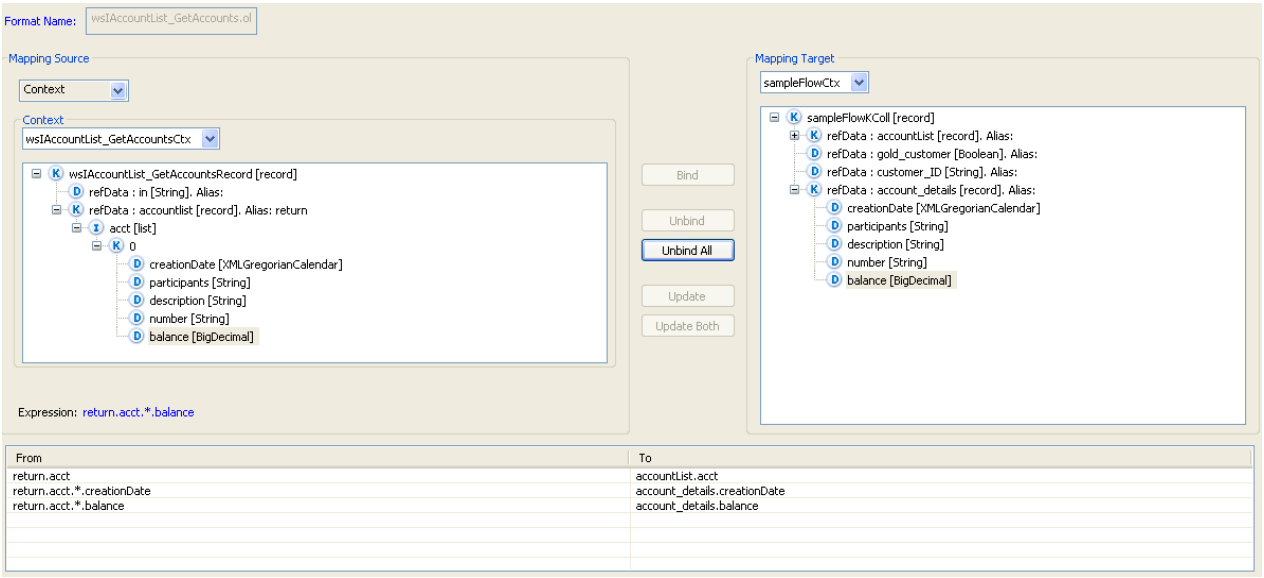
- For all other target states:

  - Local context (if it exists)
  - Session context (if it exists) and all its parent hierarchy. Session context the context with type=”session”
  - Root context (the context with type=”root”)

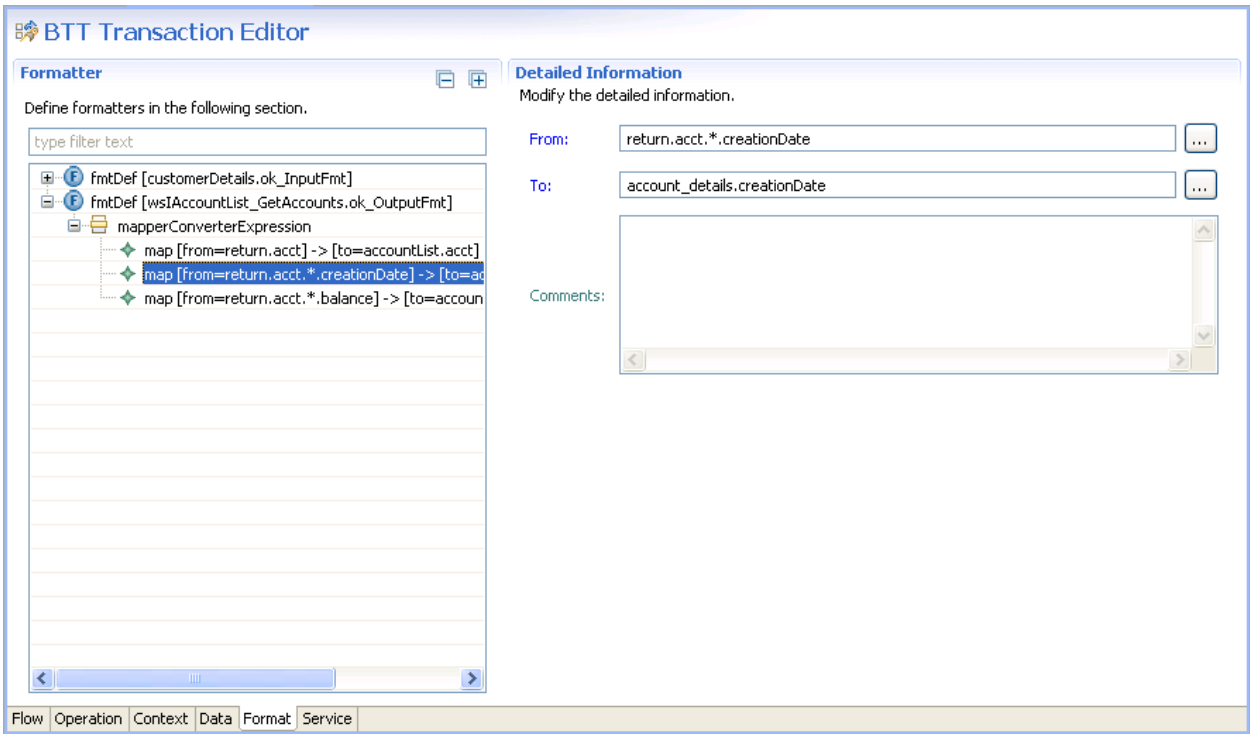
Binding the source and target data

Finally you have to bind the source data to the target data. Once you have selected the source and target data, click the Bind button 

. You can define as many bindings as you may require by repeatedly selecting the source and target data and clicking the Bind button. All defined bindings for a specific transaction appear in the table at the bottom of the Mapping panel, as shown in the *Figure* below.



The data formatter that will take care of executing the mapping at runtime is automatically generated and its definition can be found in the Format tab of the Transaction editor.



6.7.3 Updating data mapping

To modify a data mapping, do the following steps:

1. In the Flow tab, click the transition to which the data mapping that you want to modify belongs. Information on the transition displays in the Properties tab
2. If you want to modify an output data mapping, enter the name of the output data mapping that you want to modify in the Output Mapping field, and then click the Browse icon  . If you want to modify an input data mapping, enter the name of the input data mapping that you want to modify in the Input Mapping field, and then click the Browse icon  .

   Note:
   - The name of the input data mapping or the output data mapping is the id of the corresponding formatter and can be copied from the Format tab panel.
   - If you enter a new name in the Output Mapping field or in the Input Mapping field, a new data mapping is created and associated to the transition .

3. In the Data Mapping window, modify the data mapping.

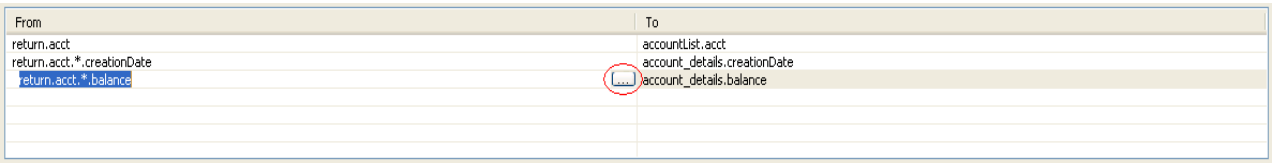
- If you are adding new data mappings to the data formatter, define the data mappings, and then click Bind. Click OK.

  Note: For more information on mapping data in the Data Mapping window, see the Defining data mappings section.

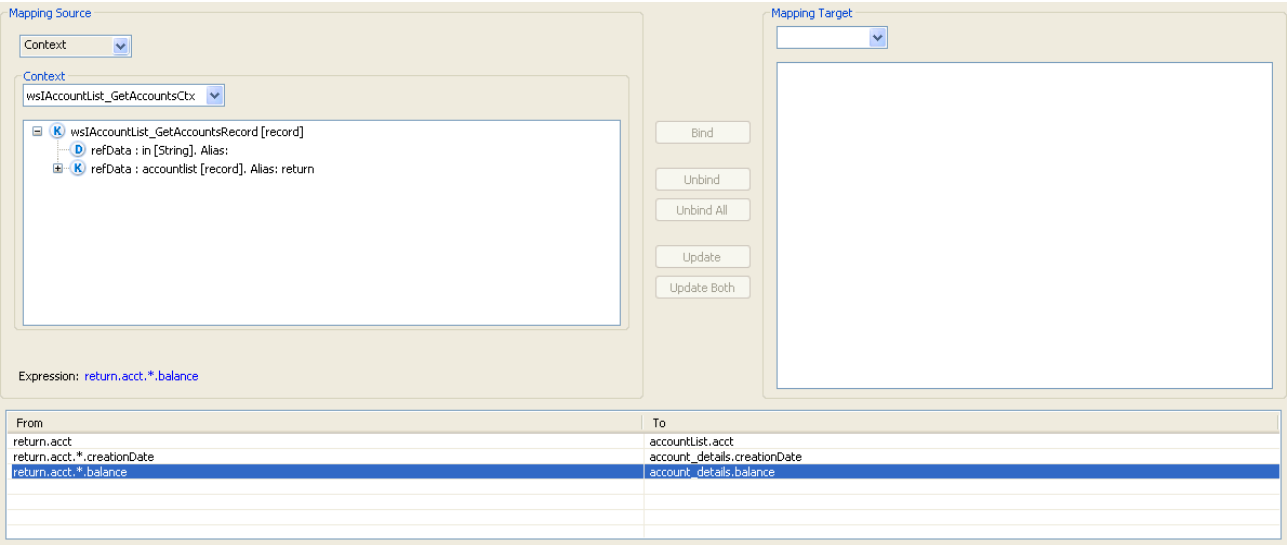
- If you are removing data mappings, select the data mappings that you want to remove in the table that is in the Data Mapping window, and then click Unbind. Click OK

- If what you want to do is to modify an existing data mapping, then:

  1. In the table in the Data Mapping window, double-click the cell that contains the data that you want to update, and then click the Browse icon  .

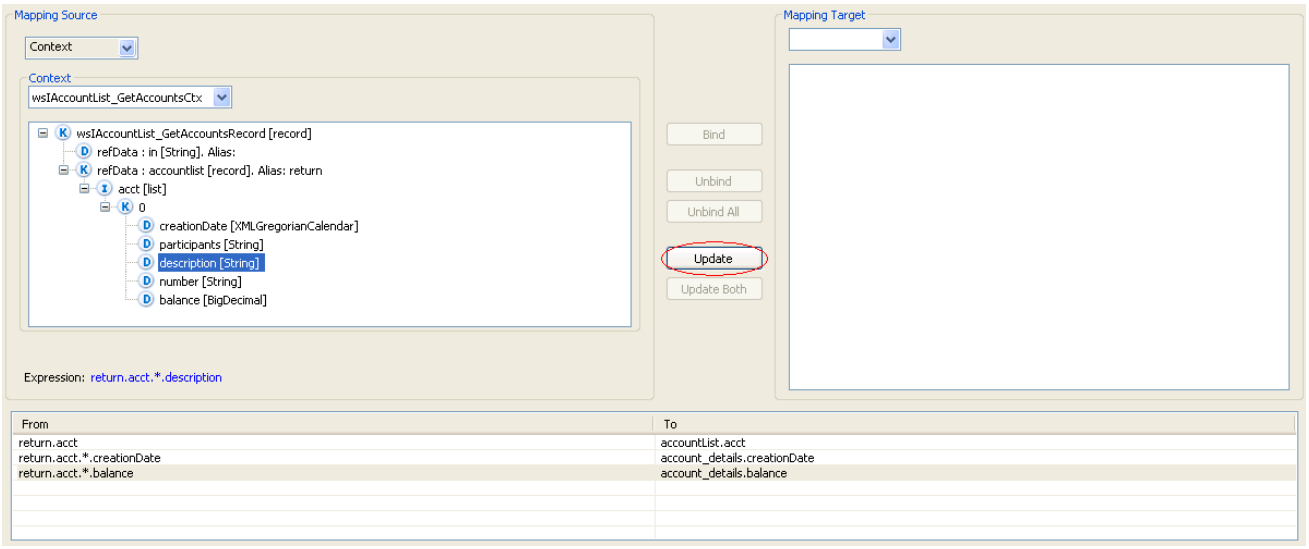


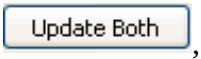
  2. A new Data Mapping window opens. Only the panel corresponding to the selected cell in the table, either source or target data, is available for a new data selection, as shown in the Figure below.

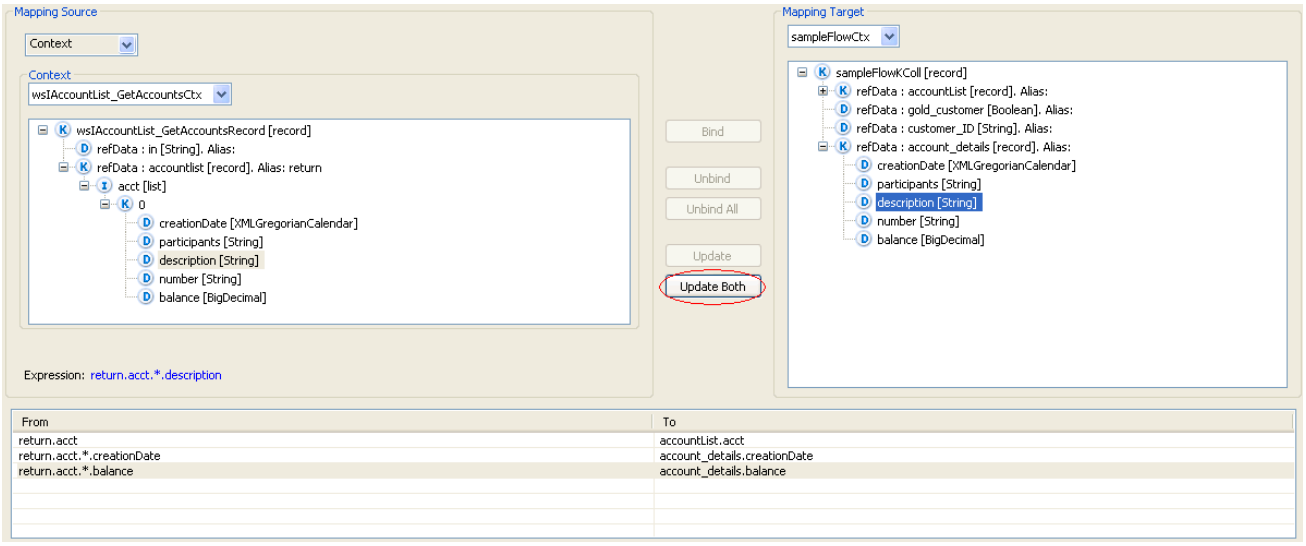


  3. Specify the new value, and then click the Update button 

. Click OK if this is the only data in this mapping you want to update



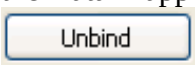
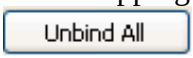
4. If you want to also update the other data element in the same mapping (From and To columns of the same data mapping), you can now, before clicking the OK button, double-click on the other element and click the Browse icon  to select the new value. Once the value is selected, click the Update Both button  , that will be now enabled as shown in the Figure below. Now click OK.



In any of the previous cases, the data mapping and the corresponding data formatter is modified. To see the modified data formatter, see the Formatter panel in the Format tab of the Transaction editor.

6.7.4 Removing data mapping

To remove a data mapping from a transition, do the following steps:

1. In the Flow tab, click the transition to which the data mapping that you want to remove belongs. Information on the transition displays in the Properties tab.
2. Select the output data mapping or input data mapping you want to delete. The name of the data mapping is the id of the corresponding formatter and can be copied from the Format tab panel.
3. If you want to delete an output data mapping, enter the name of the output data mapping that you want to delete in the Output Mapping field, and then click the Browse icon  . If you want to delete an input data mapping, enter the name of the input data mapping that you want to delete in the Input Mapping field, and then click the Browse icon  .
4. In the Data Mapping window, remove the data mapping:
   - If you want to remove a data mapping, select the data mapping in the table of the Data Mapping window that you want to remove, and then click Unbind 
   - If you want to remove all the data mappings that are defined for a data formatter, click Unbind All  . In this case, the data formatter still exists and will be called to format data but it will do nothing.
   - If you want to completely remove the reference to the defined Output Mapping or Input Mapping for a transition, delete the name of the data mapping from either the Output Mapping field or the Input Mapping Field and click OK. The data mapping is removed from the transition and is no longer used to format data for the transition.

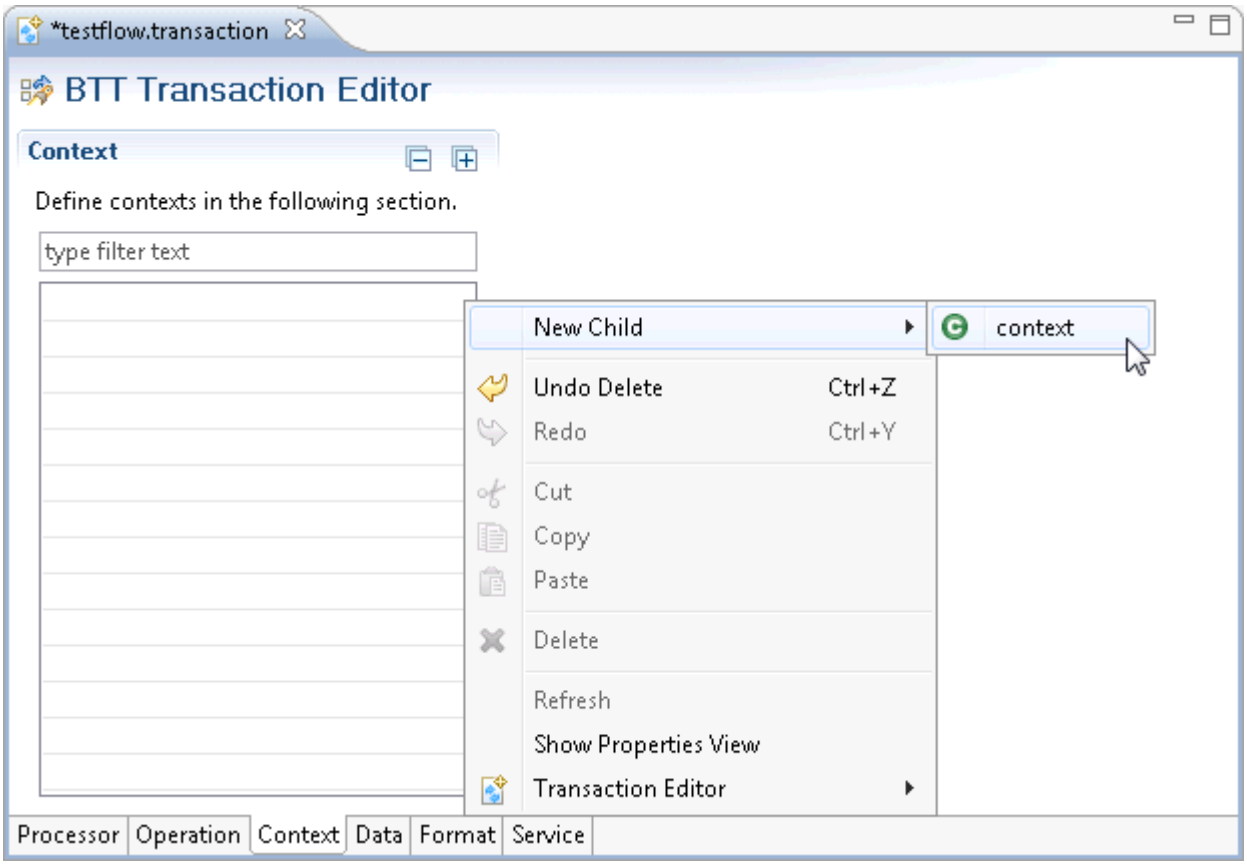
   Note: Removing a data mapping from a transition does not delete the data formatter. After you remove the data formatter from a transition, the data formatter still exists and can be used for another transition. The data formatter can still be viewed in the Format tab of the Transaction editor and can be deleted from there if not needed anymore.

6.8 Defining Context in the Transaction editor

A context defines and encapsulates data that belongs to a functional or business organization entity. To define a new context, do the following steps:, you have the option to define a new context by doing the following steps:

1. When you are defining either a flow or an operation using the Transaction editor, click the Context tab. If you are creating a common context, double-click on Definitions > **Common Contexts** on the Project Explorer or right-click on this option and select **Open**.

2. Right-click in the Context panel of the Context tab, and then click New Child > context.



1. In the Detailed Information panel, enter the required data (see the *Figure* below):

    a. Id: identifies the context instance

    b. Type: used to group the contexts or to identify a concrete context instance that must exist in the context hierarchy, as the root context or the session context. For operation and flow contexts, set the Type to op.

    c. AddToDynamicKColl: determines what happens when the context's keyed collection is dynamic and the toolkit needs to add a data element to the keyed collection.

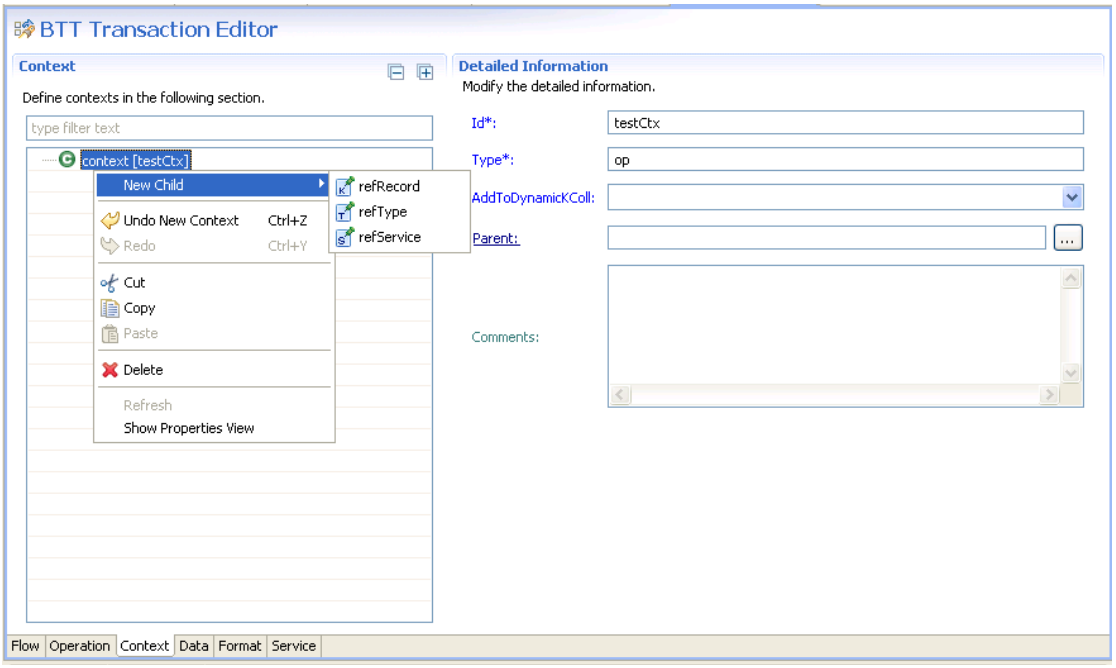
    When this attribute is false (the default value), the setValueAt method, when invoked, searches the context tree for the first element that matches the key. If the method finds the element, it modifies the value. If it does not find the element, it dynamically creates the element in the current context.

    When this attribute is true, one of the following happens:

    - If the element does not exist in the keyed collection of the context, the toolkit adds a DataField to the keyed collection. The toolkit obtains the DataField's name from the key parameter of the setValueAt method. The toolkit obtains the DataField's value from the value parameter of same method.

- If the element already exists in the keyed collection of the context, the toolkit overwrites its value without checking whether the element exists in contexts higher in the context tree.

d. Parent: identifies the parent context. You can select one from the list shown when clicking on the Browse button. If not set, the parent context is set by default to the session context if it is a flow context; if it is an operation context, the parent context is defaulted to the corresponding flow context. To allow flows and operation reuse, it is recommended to leave the Parent context blank.



2. Add a refRecord or a refType to the context as a child element (see figure above):

   Note: Every context must contain either a refRecord or a refType as a child element.

   a. Right-click the context that you are defining and then click New Child > refRecord or New Child > refType depending on the child element that you want to add to the context.

   b. In the Detailed Information panel of the child element of the context, click the Browse icon ... to select a Keyed Collection or a Type for the child element. For a detail on the properties you can define for any of these elements, refer to section Context external definitions

3. Optionally, you can add a refService to the context as a child element (see figure above):

   a. Right-click the context that you are defining and then click New Child > refService.

b. In the Detailed Information panel of the child element of the context, click the Browse icon  to select a Service for the child element. For a detail on the properties you can define for this type of element, refer to section Context external definitions

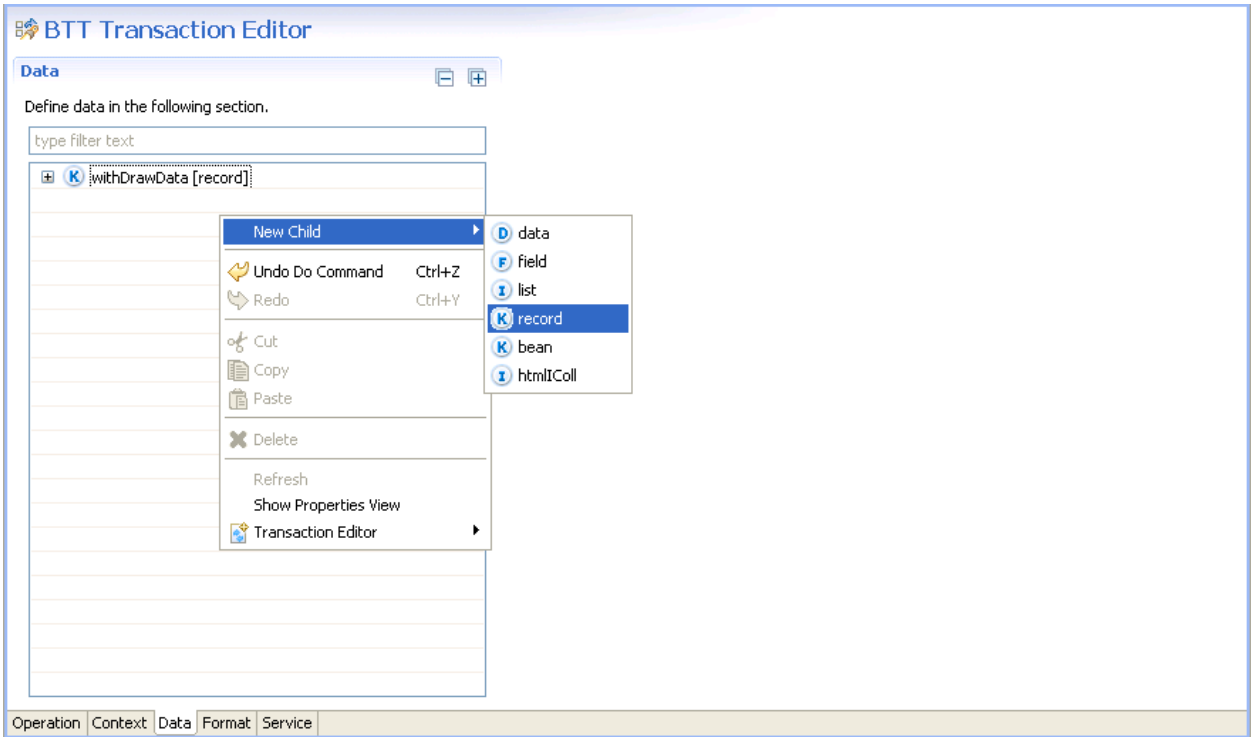
6.9 Defining Data in the Transaction editor

Data for a transaction (BTT Flow or BTT Operation) is defined in the Transaction editor. Every transaction file must have a record data type as the root data container. All other data elements are added to this record as child data elements.

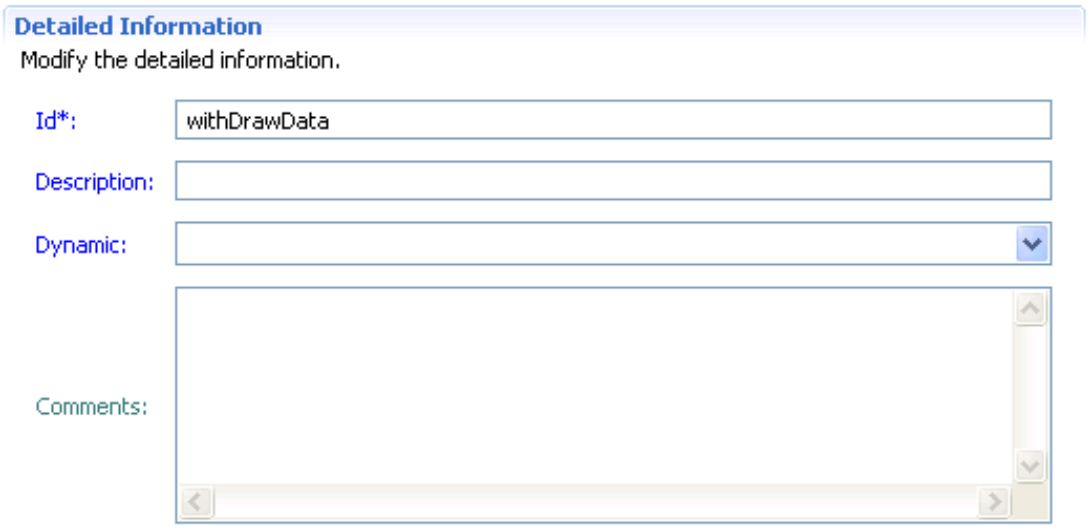
To define data for a transaction file, do the following steps:

1. When you are defining either a flow or an operation using the Transaction editor, click the Data tab. If you are creating common data, double-click on Definitions > **Common Data** on the Project Explorer or right-click on this option and select **Open**.

2. Right-click in the Data panel, and then click New Child > record. The record is displayed in the Data panel.

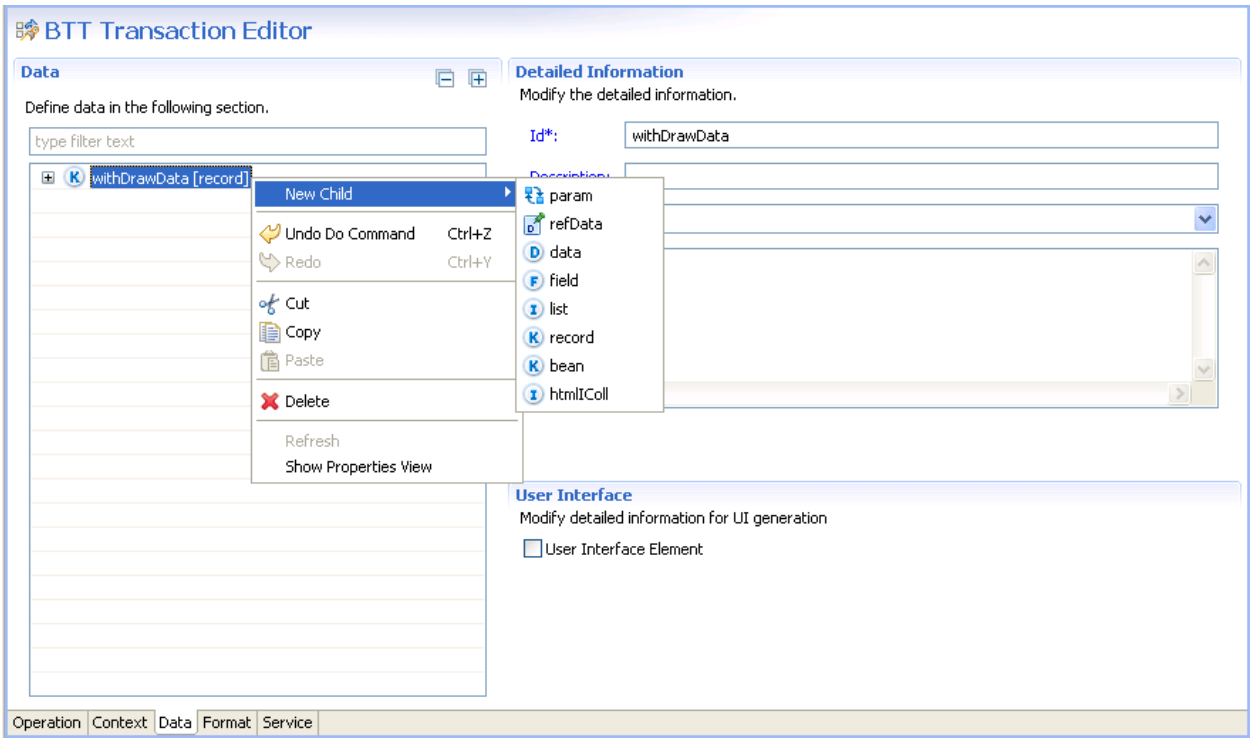


3. Enter the properties of the record you are creating in the Detailed Information panel, as shown in the figure below:



For a detailed explanation of the properties of a record data element, refer to section Data element external definition (Table 2-<kColl> tag attributes).

4. To add child data elements to a record, right-click the record in the Data panel, and then click New Child. All possible data types are prompted.



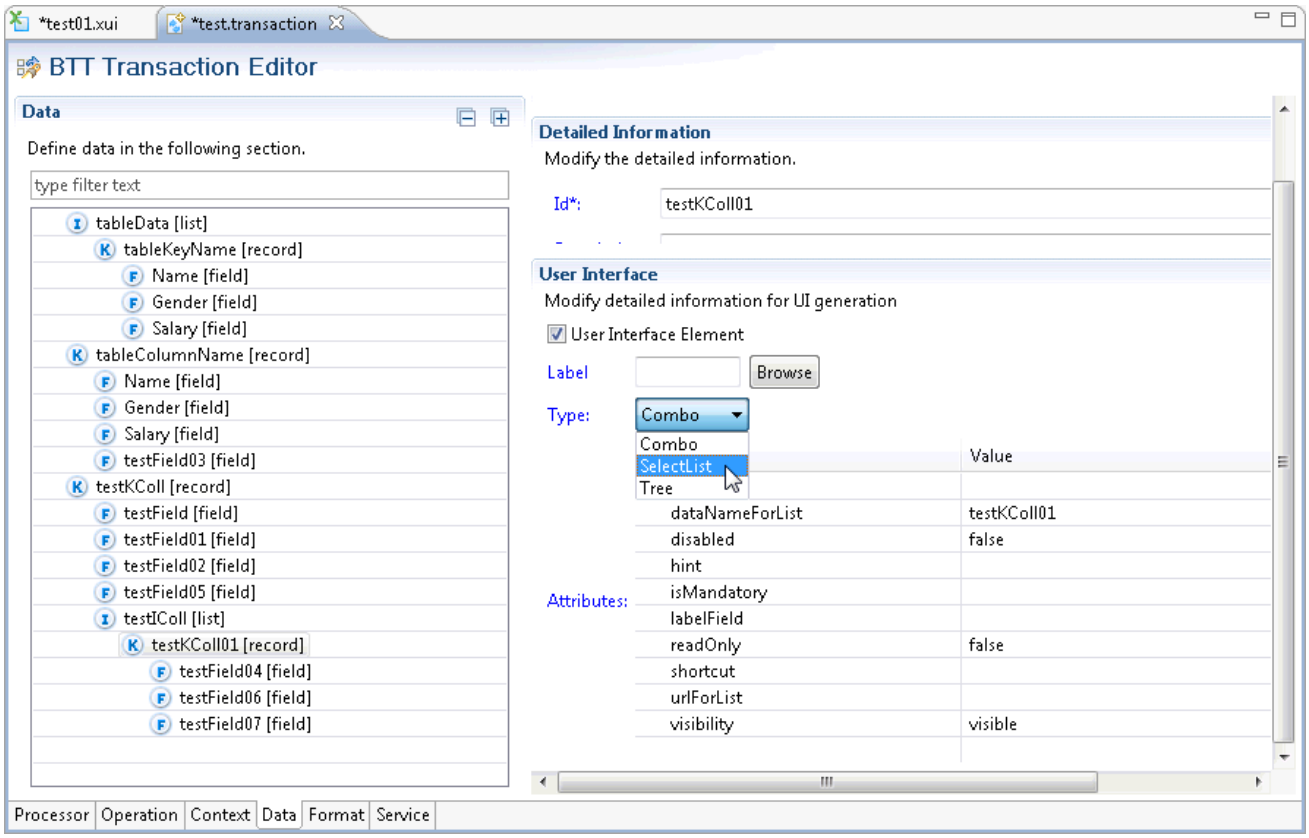
For any data element added to the root record, you have to provide the data prompted in the Detailed Information panel. For a detailed explanation of the properties of each of the possible data elements, refer to section Data element external definition

5. Data elements are associated to specific UI widgets by default; for example, a field data element is associated to a Text widget by default, and an iColl data

element is associated to a Table widget by default. You can change and specify the UI widget that is associated to a data element by using the Type list in the User Interface Panel of the Transaction editor as shown in *Figure 1*, or by using the "View generation from data" window as described in section Generating a view from the transaction context data.

Once the widget selection is made, you are able to set all the properties for the widget in the Attributes table. The Label associated to the widget can also be changed using the Label field.

Figure 1. Using the Data tab of the Transaction editor to modify the data element associated to a UI widget.

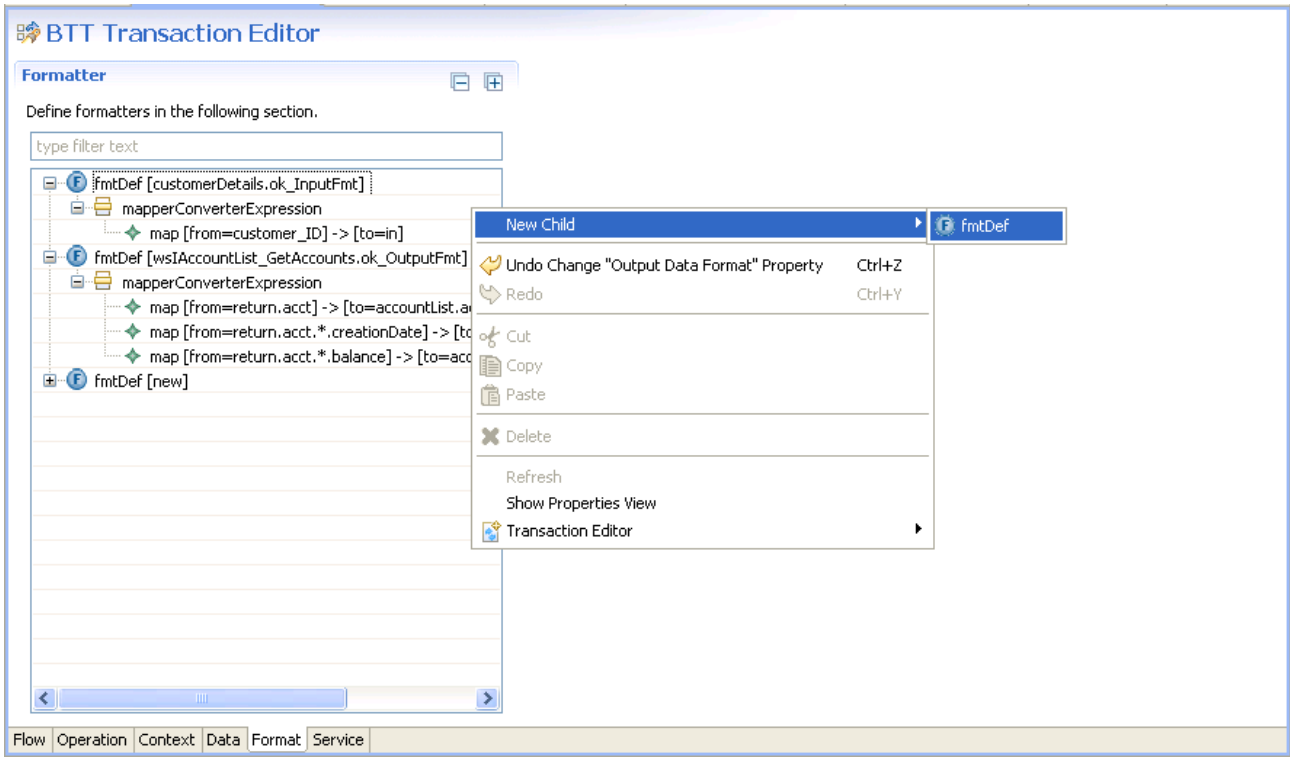


6.10 Defining Formatters in the Transaction editor

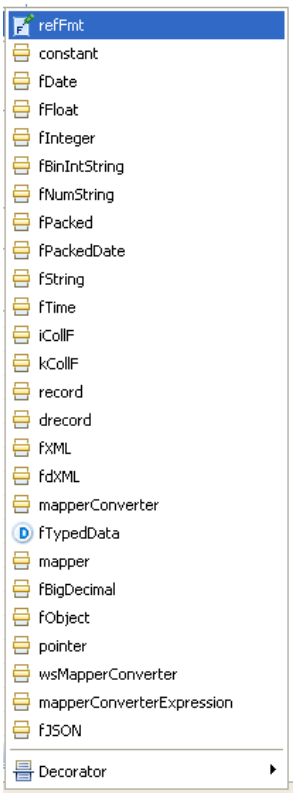
Formatters are what the toolkit uses to exchange data between toolkit entities such as operations, contexts, and services. Each formatter converts a specific data item into a string representation of the data item and parses a string into a specific data item. When you are defining either a flow or an operation using the Transaction editor, you have the option to define new formatters. Some of them are automatically generated by the tool when adding data mapping to flow transitions. To create a new formatter, do the following steps:

1. When you are defining transactions (flows or operations) using the Transaction editor, click the Format tab. If you are creating a common formatter, double-click on Definitions > **Common Formats** on the Project Explorer or right-click on this option and select **Open**.

2. Right-click in a blank area of the Formatter panel of the Format tab, and then click New Child > fmtDef



3. In the Detailed Information panel, enter the id of the new formatter and start adding children to it by right-clicking in the Formatter panel with the new formatter selected. The possible children are the following:



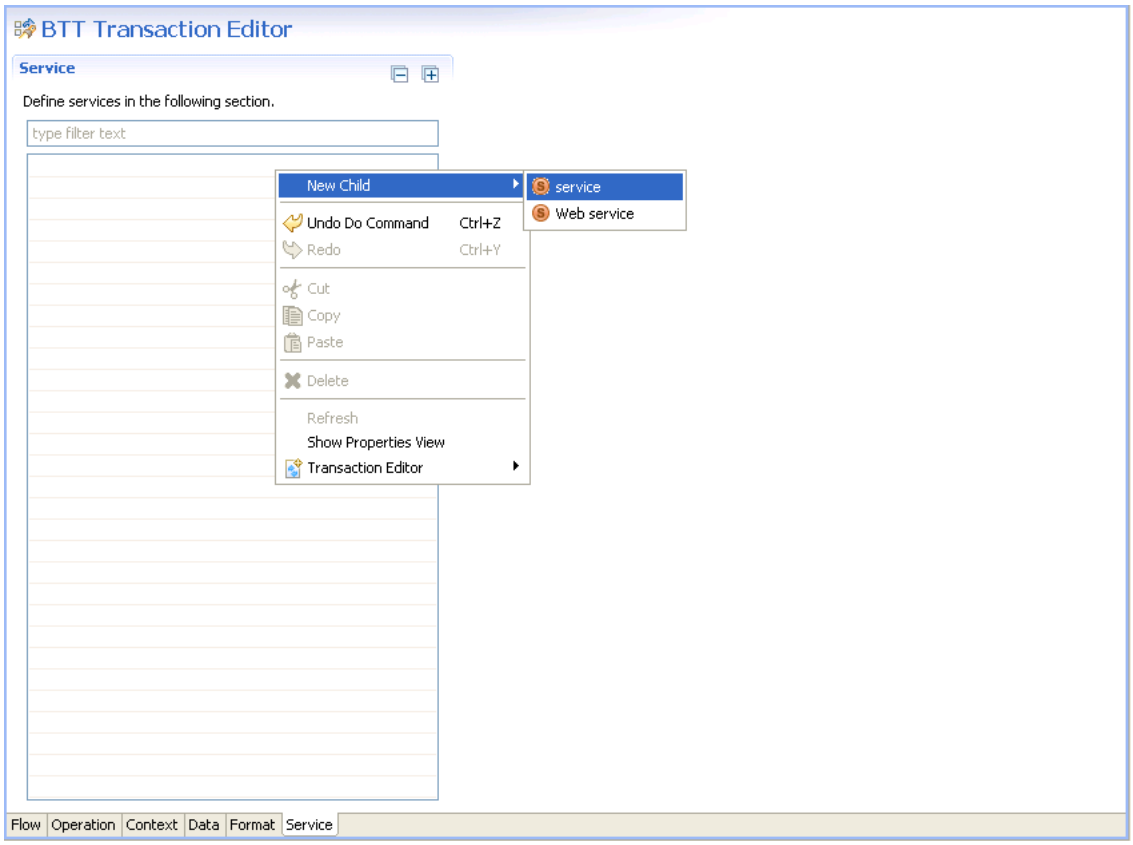
4. For a full description on how formatters work, the formatters that are provided by default and how to configure them, go to section, refer to section Formatters. In the subsections Formatter external definitions and Decorator external definitions, there are tables per each default formatter and decorator describing all their properties in detail.
5. Once the formatter has been created, you can use the formatter simulator that allows you to simulate the Formatter unformat process. It helps to speed up the development process by ensuring that the formatter definition is valid. Refer to section Formatter Simulator for a full explanation on how to set up and run this tool.

6.11 Defining Services in the Transaction editor

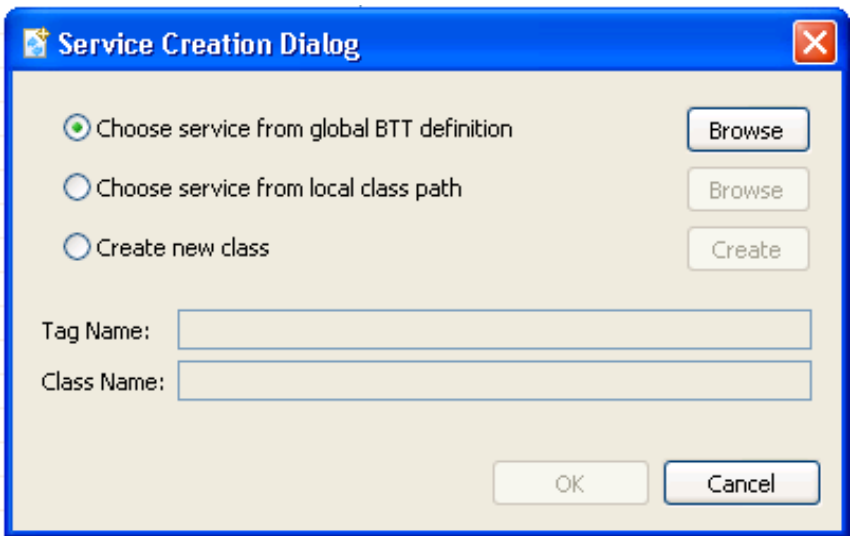
BTT provides a set of service objects that enable an application to complete an operation. These services include host communications, journaling, store-and-forward for offline operations, financial devices for input and output operations, and more. You can use the Transaction editor to add the services that will be required during the transaction execution.

1. When you are defining flows or operations using the Transaction editor, click the Service tab. If you are creating a common service, double-click on Definitions > **Common Services** on the Project Explorer or right-click on this option and select **Open**.

2. Right-click in a blank area of the Formatter panel of the Format tab, and then click New Child. You can then select between **service** or **Web service**.



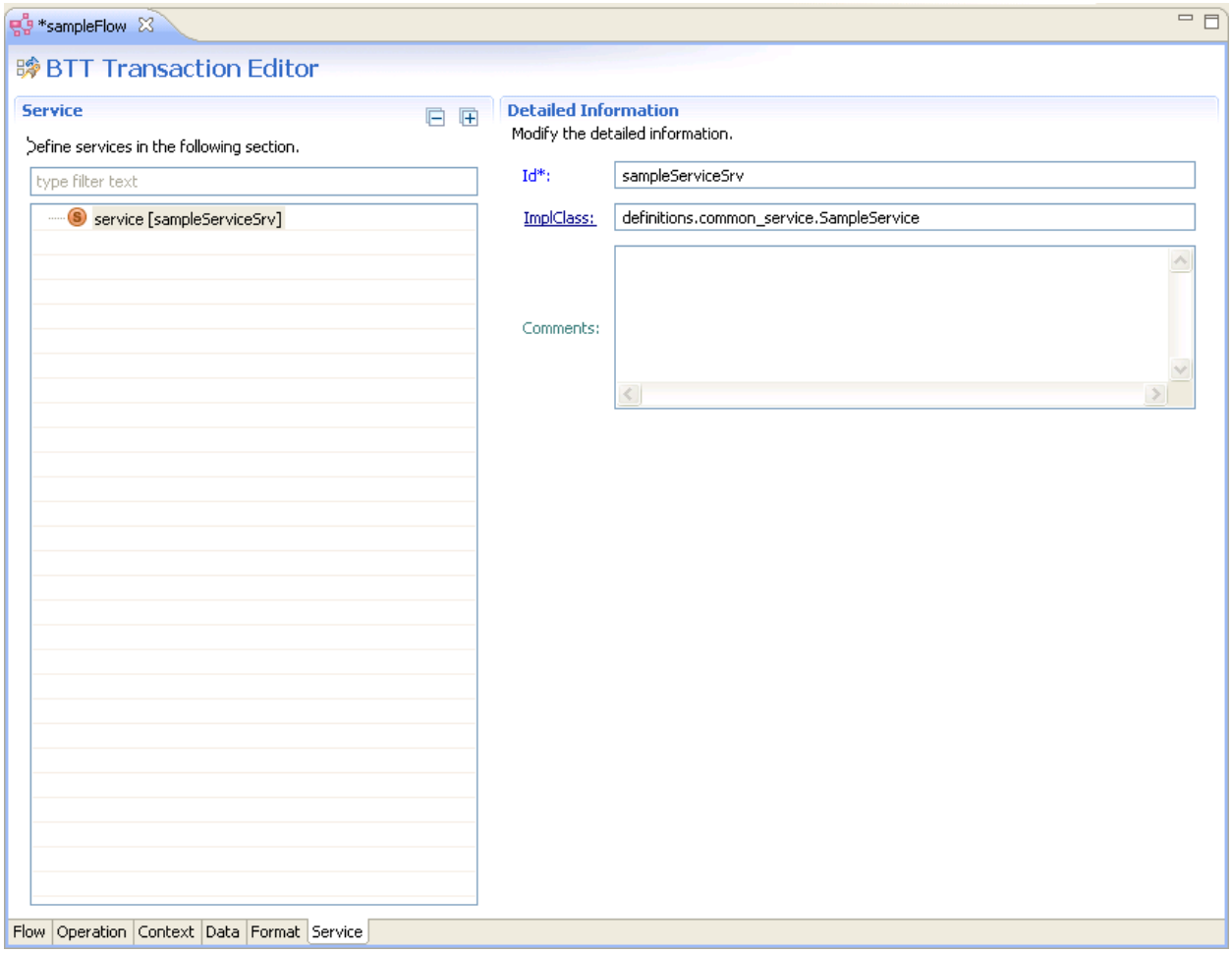
3. If you select the **service** option, the Service Creation Dialog appears. You can then choose an existing service definition either from the global BTT definition or from the local class path, or you can create a new service, as shown in the Figure below:



- If you want to create a fully new service, in the Service Creation Dialog window, select Create new class, and then click Create. The Java Class page displays.

- In the Java Class page, select the package in which you want to store the Java file.
- In the Name field of the Java Class page, enter a name for the Java file. Click Finish.
- In the Service Creation Dialog window, click OK.
- Alternatively, you may want to reuse an existing service implementation class. You can then either choose a service from the global BTT definition (the services defined in the btt.xml file. The contents of this file can be reloaded by right-clicking in the table under the Services label and then clicking Transaction Editor > Reload BTT Global Settings) or choose a service from local classpath (all Java classes implementing a service that are accessible from the current BTT project)

In all cases, the implementation class field is prefilled with either the selected class or with the new implementation class.



If you are creating a new service, you can start adding your service code to the Java template generated by default by clicking in the ImplClass label. The defaulted Java code for a 'SampleService' service is the following:

```
package definitions.common_service;
```

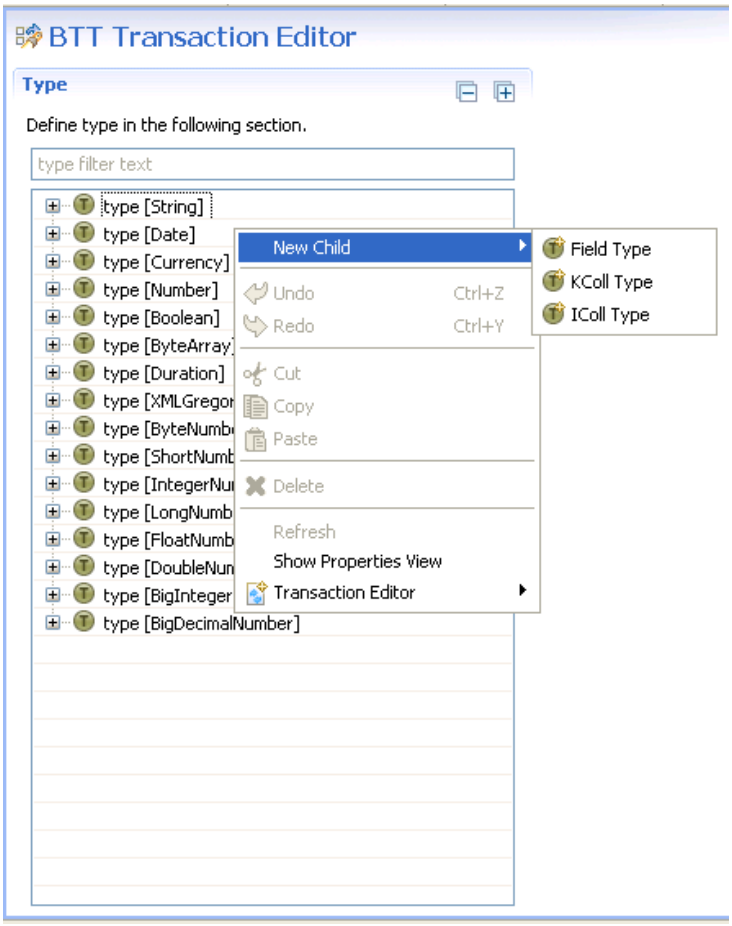
```
import com.ibm.btt.base.Service;
/**
 * Class Generated by BTT Tool
 * Created since: 2011/11/16 11:25:45
 */
public class SampleService extends Service {

      }
```

4. If you select the **Web service** option, then the **New Web services Connector** wizard opens. For instructions on how to proceed, go to section Create a Web services connector.

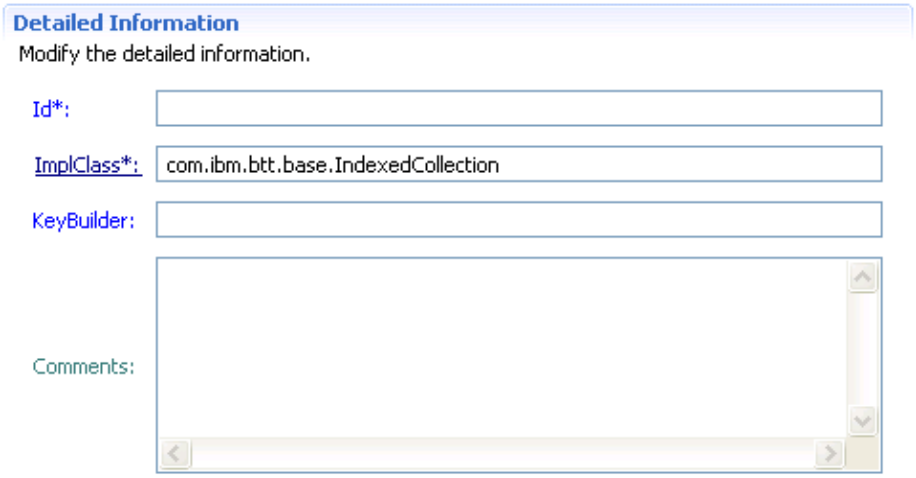
6.12 Defining Types in the Transaction editor

Using the BTT Transaction editor you can edit existing types and also define new types of data for the data elements that are part of your transaction. To add a new data type, do the following steps:

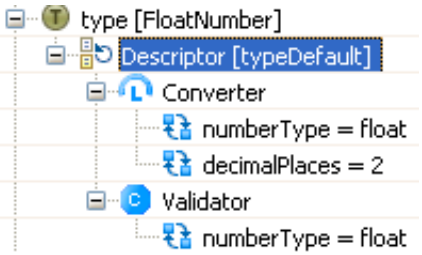
1. Go to the Transaction editor Types wizard (when in the BTT Perspective, select from the Project Explorer the Definitions > **Common Types** entry and either double-click or right-click and select the **Open** option)
2. Right-click in the Type panel, and then click New Child. You can select if you want to create a simple field type, a keyed collection or an indexed collection.



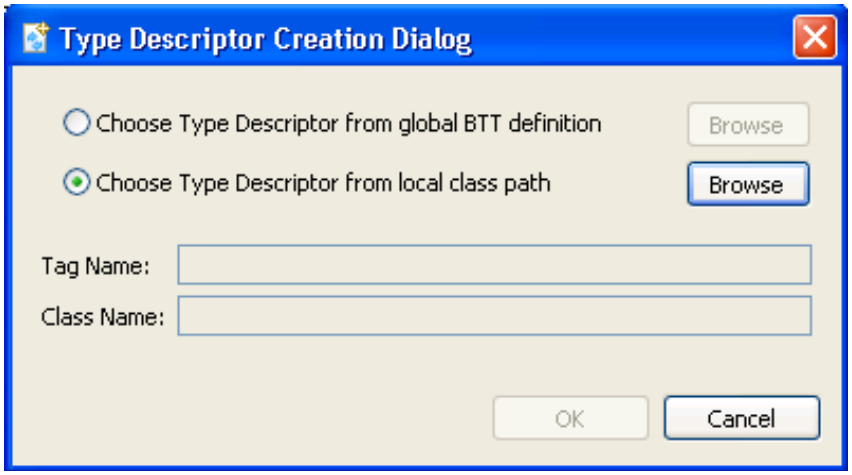
The implementation class will be automatically set to com.ibm.btt.base.DataField, com.ibm.btt.base.KeyedCollection or com.ibm.btt.base.IndexedCollection respectively.



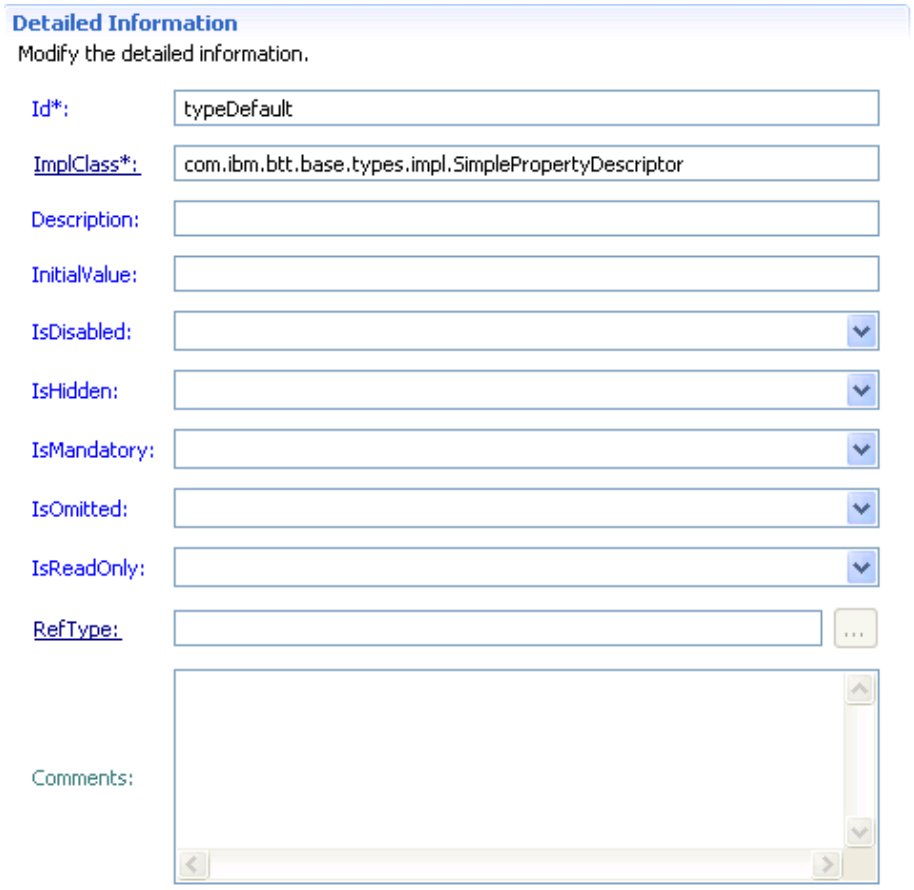
3. A type contains information about the business object it represents and the associated typed data must conform to the object's business rules. Then the type must identify how the toolkit displays the business object it represents and also what validation must occur when the toolkit changes an attribute value. To hold this information, the types have one or more property descriptors, that contain one validator and one or several converters.



The property descriptor must have been created before they can be added as a child of a type in the Transaction Editor. For a complete description on the property descriptors and how to implement them, go to Section Property Descriptors. When adding a descriptor, you will have to select an existing implementation accessible from your BTT Project:



Once selected, you will be prompted with the Detailed Information panel where you can set your specific type values.



To add either a converter or a validator, right-click on the type descriptor and select New Child. Similarly to the property descriptor, the converter and validator should be implemented before being able to select it as part of a type definition. The implementation class also determines the input parameters required for the conversion or validation; the values of these parameters are set as part of the type definition. For a complete description on the converters and validators and how to implement them, go to Sections Converters and Validators.

6.13 Flow Unit Test

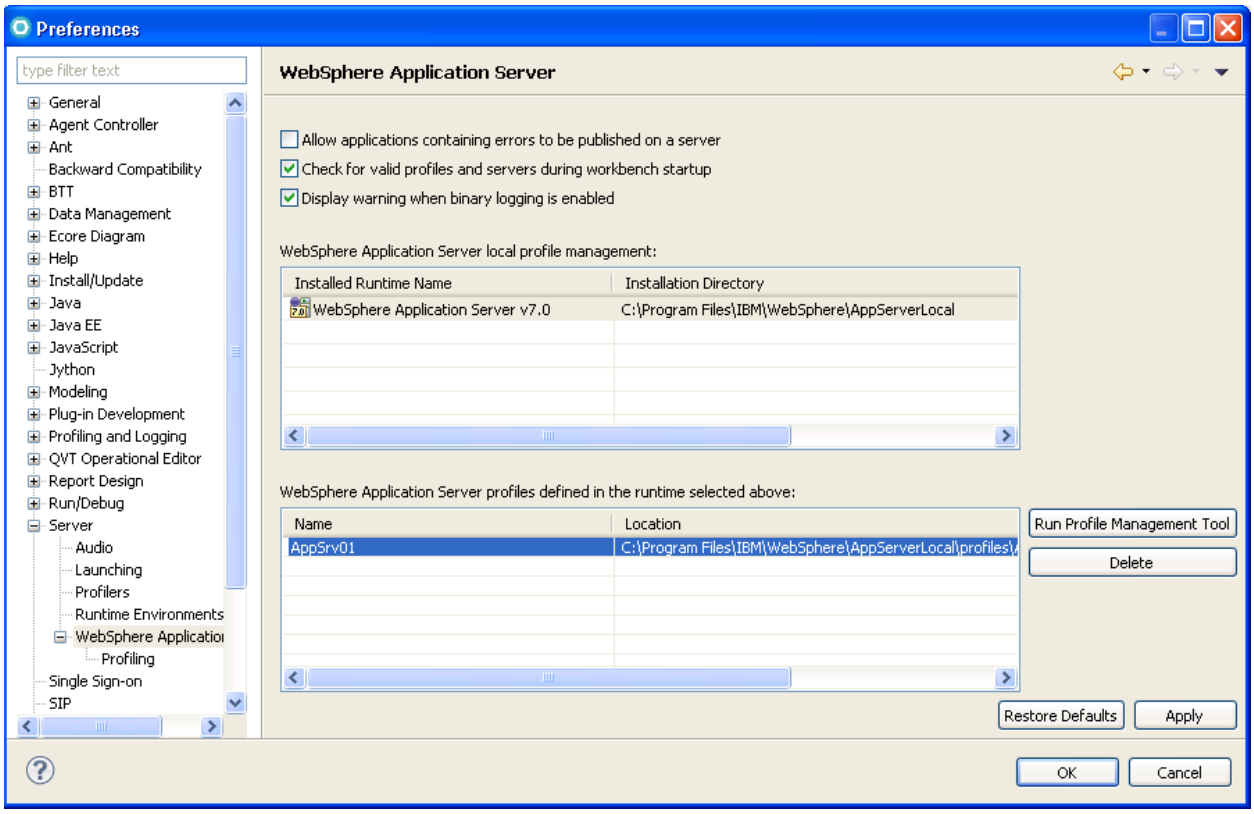
Once you have created your flow, you may want to test it in the development environment before proceeding with the deployment.

6.13.1 Environment Configuration

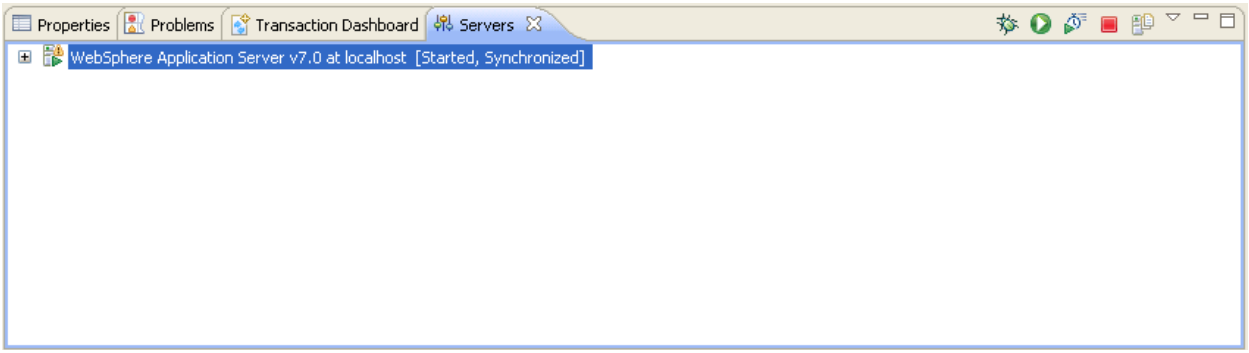
In order to run your flow in the development environment, you should enable in your workspace the WebSphere Application Server v7.0 runtime environment. To do this, do the following steps:

- **Create a server**. Most probably the server is already available in your environment as it can be added during the RAD installation as an option. You can check for the server availability and its default configuration by going to Windows

> Preferences and selecting the Server menu option. You should find a WebSphere Application Server option that opens the server configuration wizard, as shown in the Figure below:



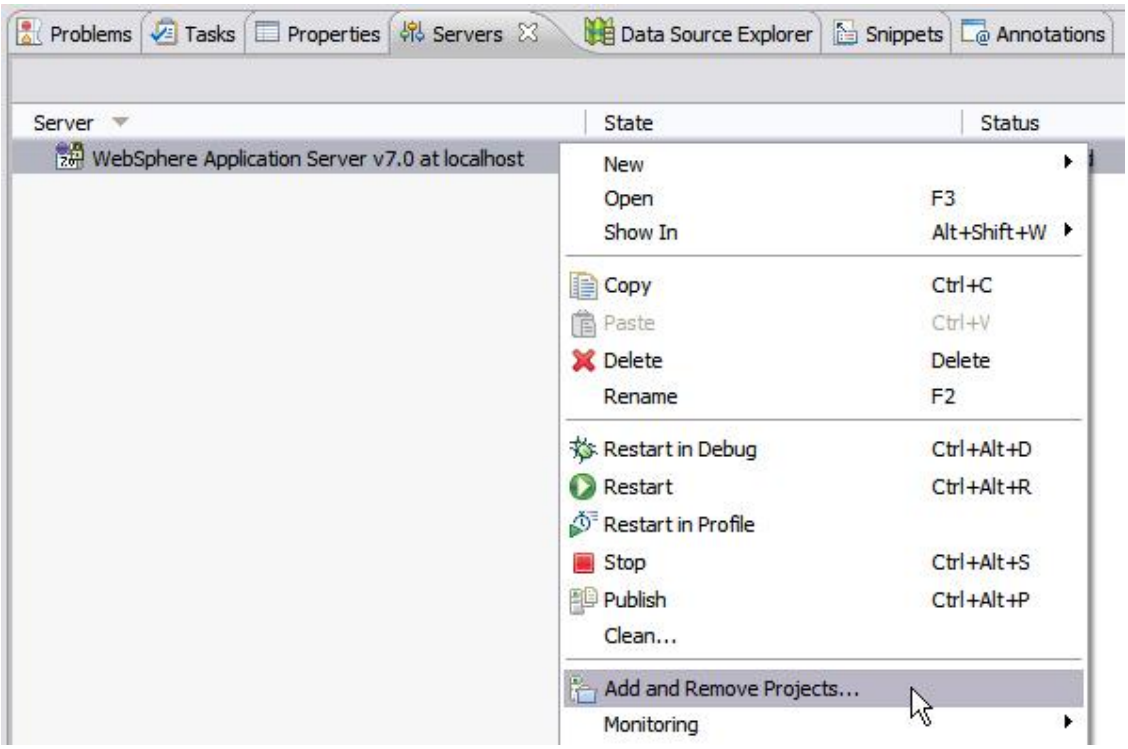
You can also open the Servers view, by going to Window > Show View > Other … > Server > Servers, to check if the server is displayed there.



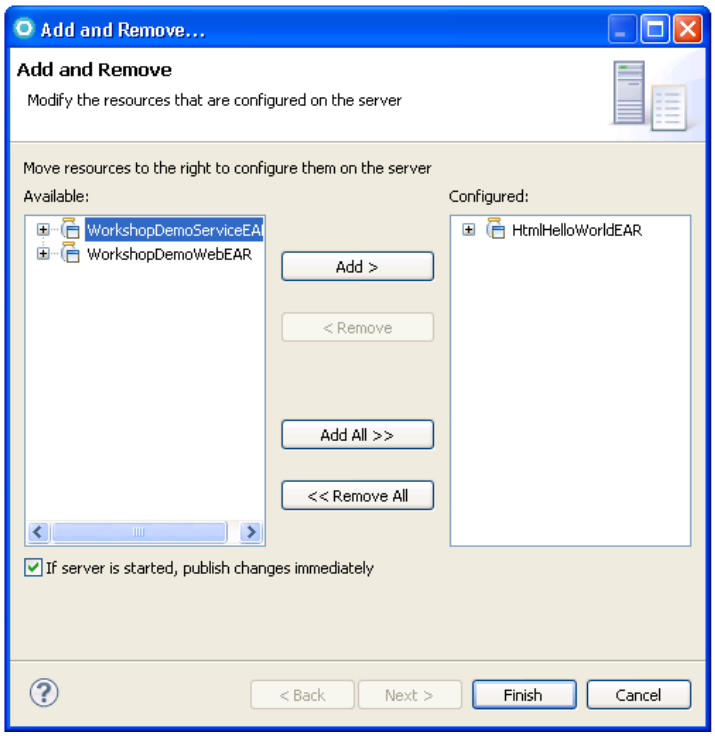
If the server has not yet been created, you can go to Create a server to proceed with the server creation.

- **Add your project to the server**

1. Right-click a server in the servers view and select Add and Remove Projects.



2. Select a project from the Available projects column and click Add. The project is added to the Configured projects column.

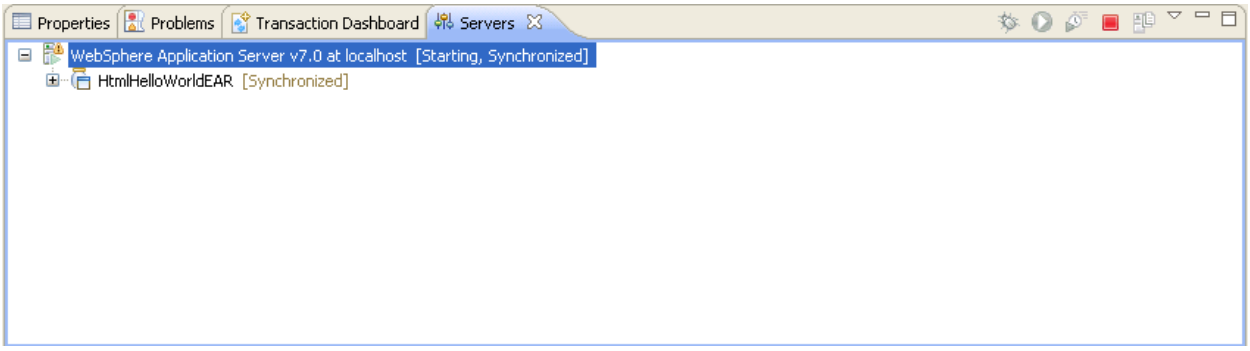


By checking the check box as shown in the figure above, if the server state was `Started` when adding the project, the project is added to the server and published immediately. If the server state was `Stopped` when adding the project, the next time the server is started, the project will be published.

You can decide not to publish the project automatically. You should then publish it after the server is started by:

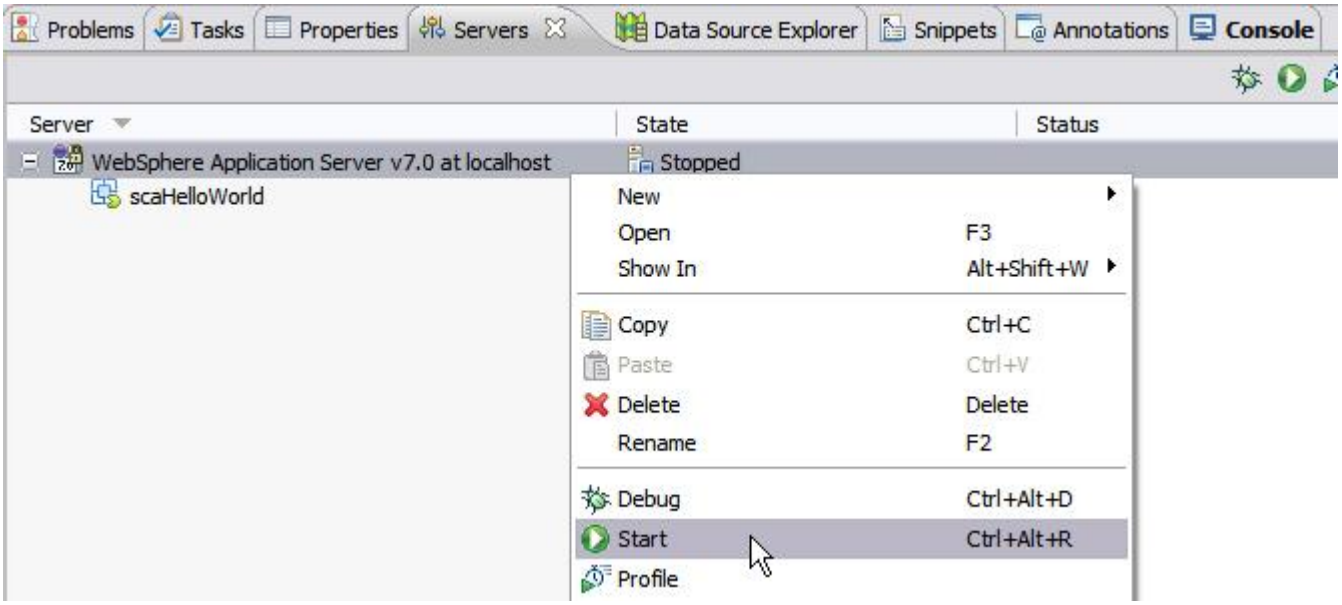
- right clicking in the WebSphere Application Server entry in the Servers view and selecting the Publish option and
- right clicking on the project and selecting the Start option

A project can be tested when it has been published and the server has been started.

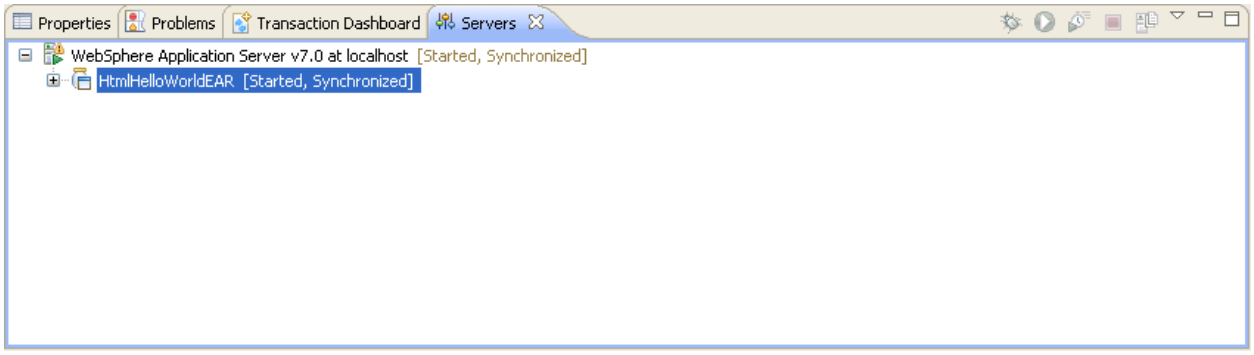


• **Starting the server**. To test a project deployed to the server runtime environment, the server must be started.

If the server is in `Stopped` state, right-click a server in the Servers view and select Start.



When completed, the state changes to `Started`. The state of the published projects should be also `Started` as shown in the Figure below:

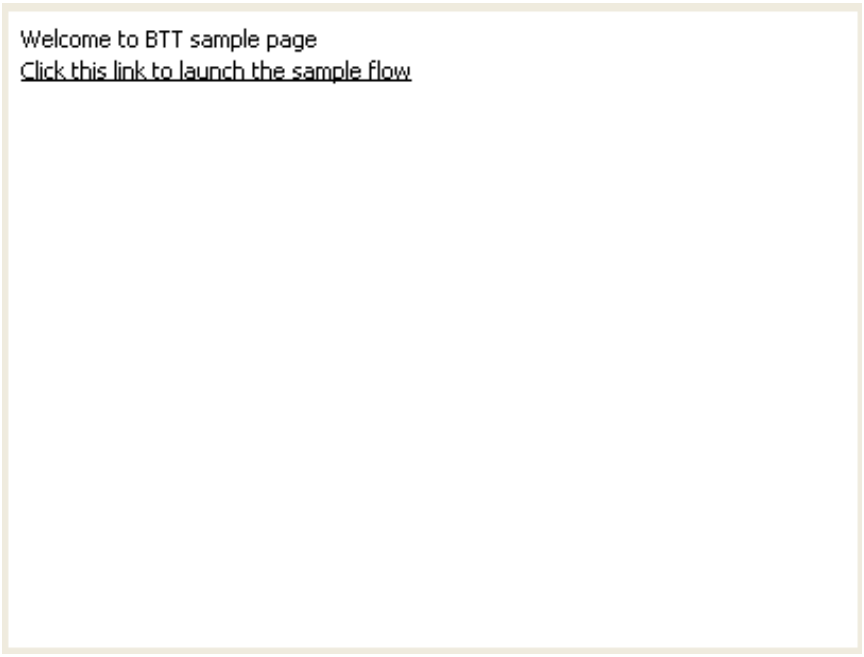


Now you are ready to run your project.

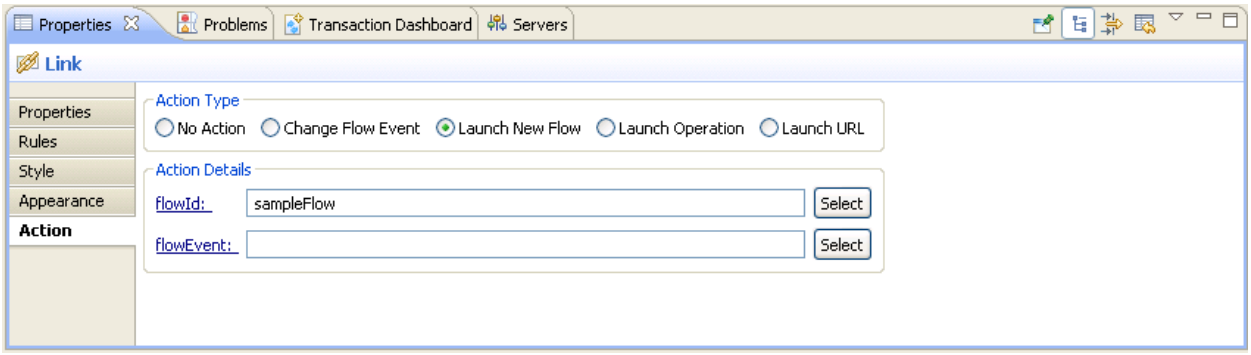
6.13.2 Testing the flows

If your project does not yet have a welcome view from were to launch the different transactions that are being developed, you can use the default **index** view that is generated by default when you create a new BTT project and that simulates the welcome page of your application.

The **index** view contains only a Label and a Link that is an example of how to launch, from this view, a flow.

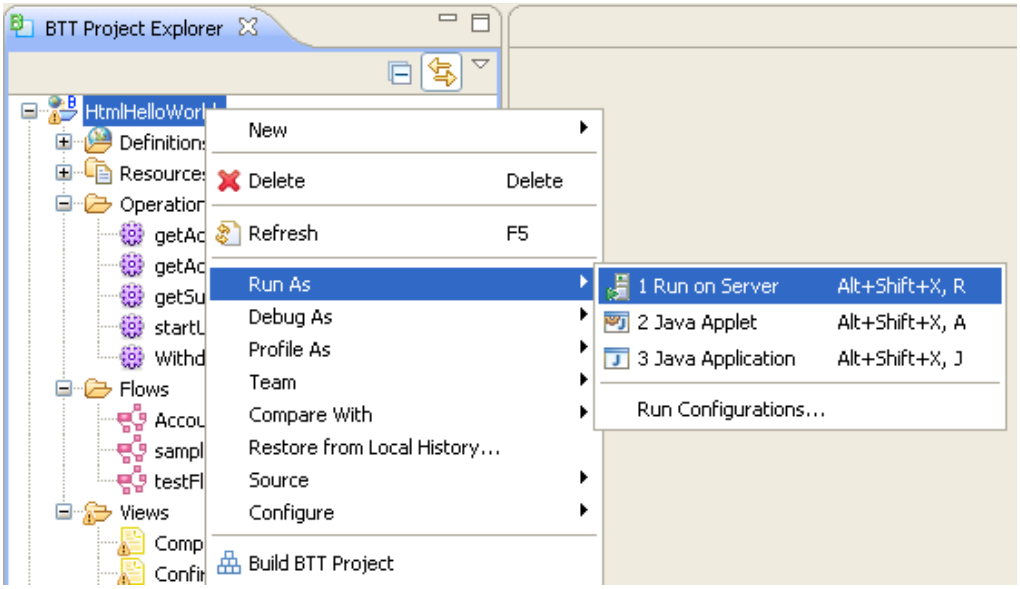


Having a look to the Link properties in the View editor, Action tab, you see the following:

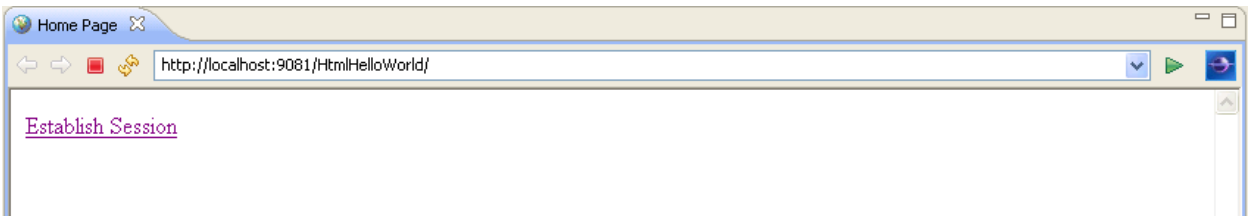


You can use this Link to test any of your project flows by changing the flowId selection or you can add new Links with the same configuration (Action=Launch New Flow) to launch all your flows from this index view. You can change the Link text property to better identify the flow you are going to launch.

You can then run your project by right-clicking in the project name in the Project Explorer and selecting Run As > Run on Server.

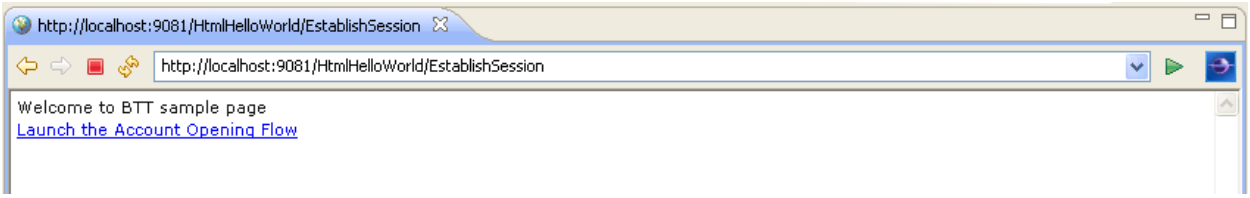


Select the WebSphere Application Server v7.0 as runtime server and click Finish. A browser will start inside the development environment with the following content:



Note: The view is displayed in the web browser that has been configured by the technical developers to be used by default when web pages are opened.

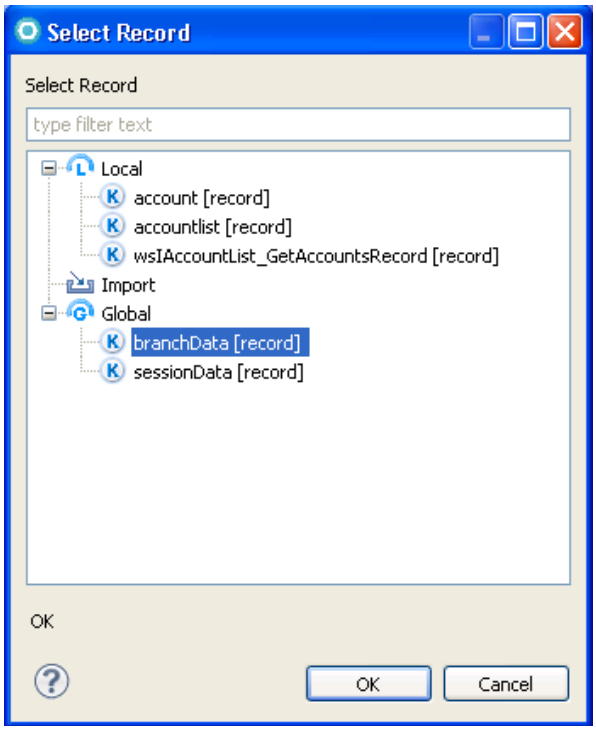
If you click on the Establish Session link, the next page that opens is the index page:



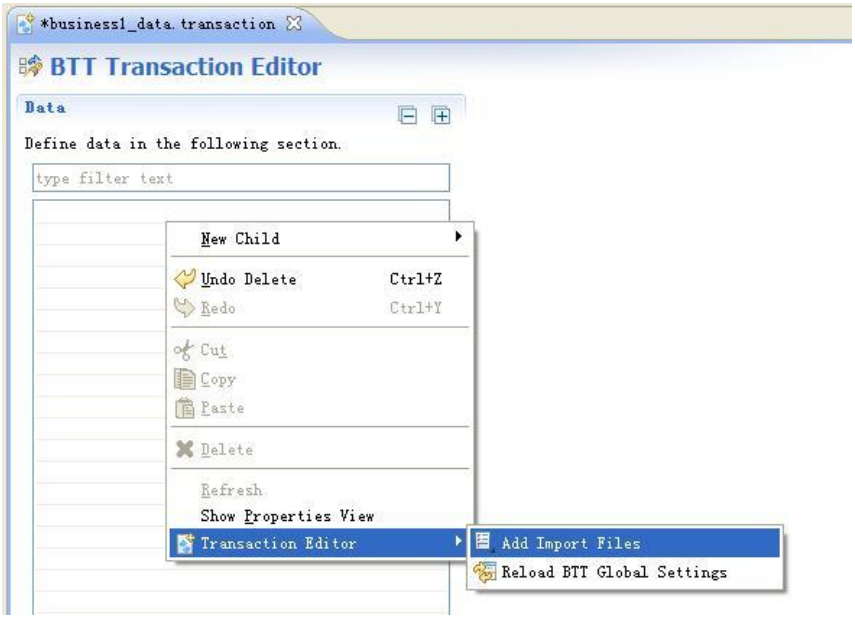
You can also copy the URL (*http://localhost:9081/HtmlHelloWorld/EstablishSession* in the sample image) and open it in any browser outside the development environment. By clicking in the links you have created, the different flows will start. Use the debugging and tracing capabilities provided by BTT to test the flows and detect and correct possible errors. Go to next section to understand how tracing capabilities are configured and used in the development environment for functional developers.

7. Importing Definitions

While defining your transactions, you need to refer to other BTT elements. These BTT elements can be available locally, globally (as part of the global definitions for your project) or can be imported from another project or folder.



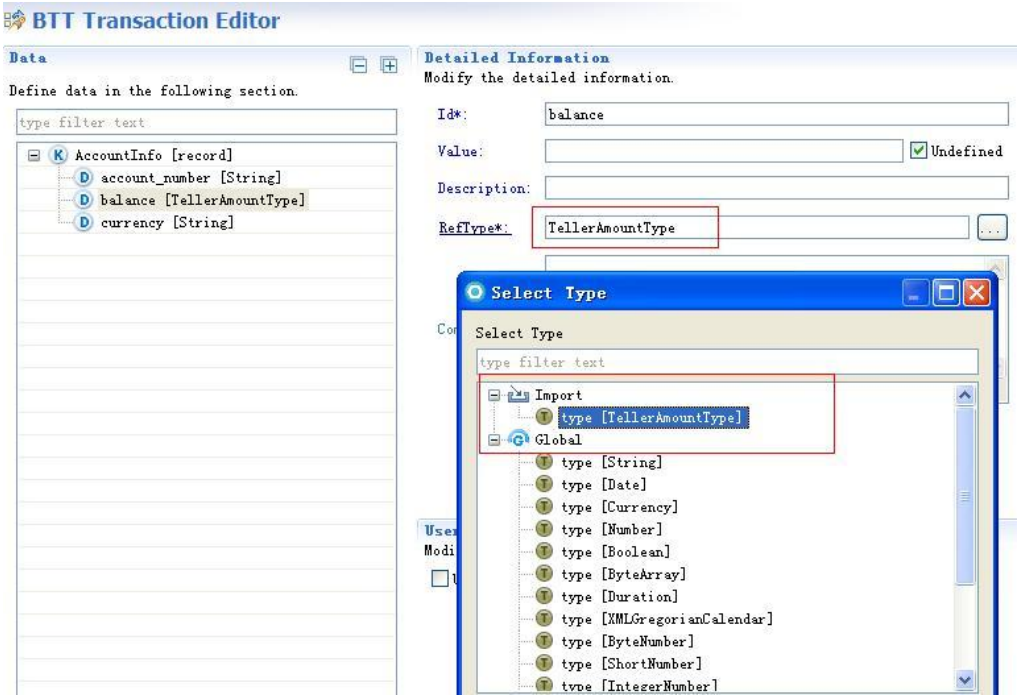
The BTT runtime and tooling support to import BTT definitions from other projects or locations. From the Transaction Editor, while editing any BTT element (types, data, contexts, formatters, services, operations or flows), you can right-click in the elements table with no element selected, and select the Transaction Editor > Add Import Files option.



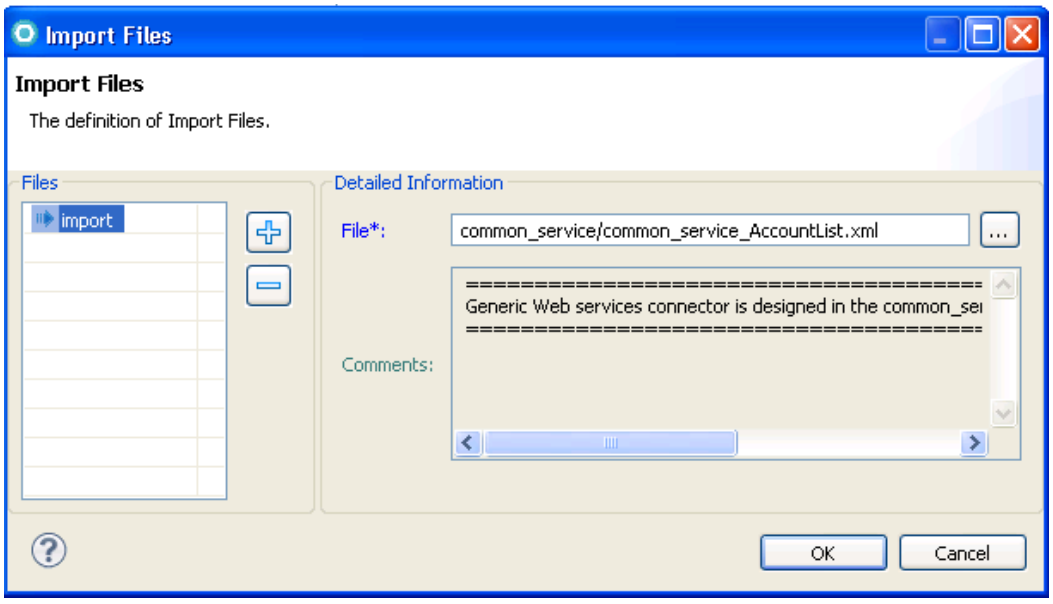
If you are defining any type of global data (the Transaction editor is launched from any of the common folders inside the Definitions folder in your BTT project), the imported files must contain definitions for the same category of elements, being types, data, contexts, formatters, services or operations. For self-defined transactions (operations or flows) you can import files containing any type of elements. All files that have been already imported to your common definitions or to your transaction definitions are listed in the Import Files panel that opens when selecting the Add Import Files option, so you can add new ones or delete existing ones when required.

Note: to be able to import definitions from an existing project in the workspace, a previous configuration must be done by a technical developer that should identify, in the btt.xml file, which projects are available in the Select File panel for importing for a specific project.

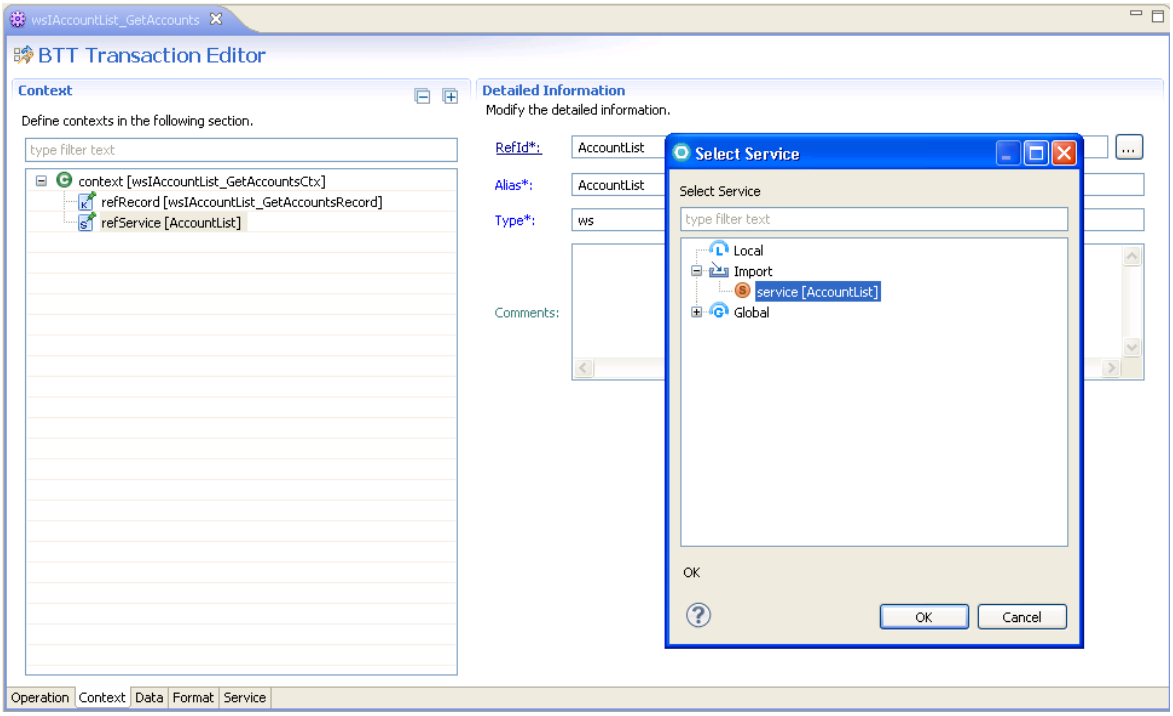
After the definitions are imported, the imported elements can be referenced during the definition of any other BTT component. These imported elements are available in the Import, so you know that the element is not defined in your project. See the Figure below:



An example of imported services can be found when you create an operation from a web service definition file. The generated BTT service, the web service connector, is defined as a common service to be reused by different operations, each one calling a different web service function, and is imported into the project self-defined operation definition file. Opening the operation with the Transaction Editor, you can check for the imported files as shown in the Figure below:



Going to the Context tab in the Transaction editor, and double-clicking on the service definition (refService child) to prompt the Detailed Information, you can confirm that the referenced service has been imported from another project or folder by clicking on the ... button. The service is listed in the Import folder, as expected.



The usage of local, global or imported BTT elements during the transaction design should be transparent to the functional developers. Which BTT elements should be available for selection is part of the development environment configuration done by the technical developers.

The multi-project environment configuration may add some remote BTT elements and resources to the Import folder, as well as to the Global folder. For a details explanation on

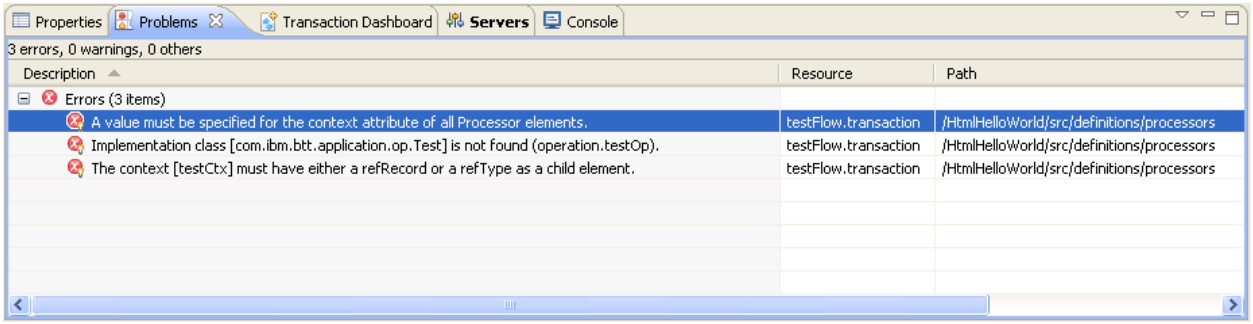
how the multi-project support provided by the toolkit works and how it affects to the development on the BTT Perspective, go to section Understanding and working in a multi-project environment.

8. Detecting errors and following the transaction execution. Using execution traces.

8.1 Managing editing errors

The RAD Problems view allows you to understand the errors when you are editing any of the BTT resources in the development environment. To open the Problems view, go to Windows > Show View > Other … > General > Problems. The messages are self explanatory and indicate where the error is (resource name and path) as well as enough information to resolve it. The errors can be corrected by using the BTT editing tools.

Figure 1. Example of errors during a flow creation



All editing error should be corrected before proceeding with the functional test of any of the BTT resources. You can double-click on the error description to locate the problem in the corresponding resource.

8.2 Server Functional Traces

The Server Functional Trace is intended for functional developers, and it records execution and error information when code is being tested during the development phase. Traces are shown in the development environment Console. To open the Console view, go to Windows > Show View > Other … > General > Console.

The default configuration for the **Server Functional Trace** is the following:

enableTechnicalTraces parameter

Specifies whether a technical trace is enabled. The default value is “false”. The value of true gives additional information that may be used by the technical developers to diagnose an application problem.

traceRequester

Indicates which is the information we want to trace. The possible attributes for this parameter, their possible values and the default values are listed in the following table

Table 1. Attributes of the traceRequester parameter	
Attribute	**Description**
id	Possible values are: #FUNC- Relate to transaction traces for flows and operations. #FUNCDATA- Relate to page post context data information (the data available in the transaction context when the page is about to be sent for processing) These traces are enabled or disableds by setting the *trace* attribute.
trace	Indicates whether to generate the related trace. The default value is "yes".
functional	Indicates whether to throw exception dump to functional developers. The default value is "true".
traceLevel	Indicates the trace level. Possible values are "FATAL, ERROR, WARN, INFO, DEBUG". The default value is "INFO".

This configuration can be changed by the technical developers and additional information will then appear in the execution traces. This section describes what is generated using the default configuration.

Tracing messages are generated for the following components:

- **Flow processor**
  - FLOW START/END. The values set for an specific flow start/end message are the **flow ID** and the **initial or final state label**

```
INFO:    => Starting flow 'tableScenario1Flow' at initial state
'initial'
INFO:    <= Ending flow 'tableScenario1Flow' at final state
'final'
```

  - FLOW CHANGE STATE. The values set for a flow change state message are the **state ID** and either the **flow** or **sub-flow ID**.

```
INFO:    Entering state 'GetTableDataOp' in flow
'tableScenario1Flow'
```

  - TRANSITION TRIGGER. The values set for an specific transition trigger message are the **event ID,** the **source and destination states,** and the **flow** or **sub-flow ID**

```
INFO:    Transition caused by event 'start', from state 'initial'
to 'GetTableDataOp' in flow 'tableScenario1Flow'
```

  - CONDITION. There are two messages associated to a condition: one that indicates that a condition has been evaluated and the other that is shown when no condition evaluates to true and then the default event is

triggered. The values set for a specific condition evaluation message are the **condition expression**, the **evaluation result** and the **associated event.**

```
INFO:   Condition
[BTTCollectionFunctions.tableSizeaccountList.acct)>1] evaluates to
true = => event 'many'
```

When the default event is triggered, the message contains the **default event**

```
INFO:   All conditions are false = => triggering default event
'error'
```

- DATA MAPPING. Several messages are generated for each mapping rule rule:
  - Mapping start/end. The **type of mapping**, input or output, the **from context** and the **to context** are set in the message

```
INFO:    => Begin output mapping from context 'tableCtx' to
context 'tableCtx'
INFO:    <= End of mapping
```

  - Mapping progress. Different messages are shown depending on the type of source and target data. The values that may be set as part of the message are the **source expression or element**, the **target element** and the **mapped value**

```
INFO:    Mapping from '*' to '*'
```

- **Operation execution**
  - OPERATION EXECUTION START/END. The value set for an specific operation start/end message is the **operation ID**

```
INFO:    => About to execute operation 'GetTableDataOp'
INFO:    <= Operation 'GetTableDataOp' executed
```

If the operation is a web service call, then the web service request data and the web service response data is also traced

- **Page rendering**
  - PAGE RENDER. The values set in the page render message is the **Page ID**

```
INFO:    Rendering page 'Pagination/tablePagination01'
```

- PAGE REQUEST DATA. Several messages can be generated as a result of a page request:
  - There is no application data as part of the request

```
INFO:    Processing page request – no application data
```

- The page request generated some application data. This application data is then shown in the trace

```
INFO:     => Page request data follows
INFO:     => Contents of context 'tableCtx' follows:
INFO:     <= End of context data
INFO:     <= End of page request data
```

Between the start and end context data message, the context data (keyed collections, indexed collections and fields) is shown. For the indexed collections, it is specified the name and the number of elements and only the first element is added to the trace.

```
INFO:       IndexedCollection 'testIndexCollection' has 5 elements.
INFO:      Showing first element only:
```

All entries related to a BTT component are preceded by an entry with the date and time when the action occurs and the identifier of the BTT trace generator, **FuntionalTraceHelper**:

```
2011-8-3 16:30:22 FunctionalTraceHelper
```

Figure 1 shows an example of the server functional trace of the execution of a flow.

Figure 1. A trace of the execution of a flow.

```
2011-8-3 16:30:22 FunctionalTraceHel
INFO:    Processing page request - no application data
2011-8-3 16:30:23 FunctionalTraceHel
INFO:    => Starting flow 'tableScenario1Flow' at initial state
'initial'
2011-8-3 16:30:23 HtmlRequestHandler
INFO: no error page specified
2011-8-3 16:30:23 FunctionalTraceHel
INFO:     Transition caused by event 'start', from state 'initial' to
'GetTableDataOp' in flow 'tableScenario1Flow'
2011-8-3 16:30:23 FunctionalTraceHel
INFO:     Entering state 'GetTableDataOp' in flow 'tableScenario1Flow'
2011-8-3 16:30:23 FunctionalTraceHel
INFO:    State 'GetTableDataOp' is an operation state, launching
operation 'GetTableDataOp'
2011-8-3 16:30:23 FunctionalTraceHel
INFO:    => About to execute operation 'GetTableDataOp'
2011-8-3 16:30:23 FunctionalTraceHel
INFO:    <= Operation 'GetTableDataOp' executed
2011-8-3 16:30:23 FunctionalTraceHel
INFO:    => Begin output mapping from context 'tableCtx' to context
'tableCtx'
2011-8-3 16:30:23 FunctionalTraceHel
INFO:    Mapping from '*' to '*'
2011-8-3 16:30:23 FunctionalTraceHel
INFO:    <= End of mapping
2011-8-3 16:30:23 FunctionalTraceHel
INFO:     Transition caused by event 'ok', from state 'GetTableDataOp'
to 'tablePagination01' in flow 'tableScenario1Flow'
2011-8-3 16:30:23 FunctionalTraceHel
INFO:     Entering state 'tablePagination01' in flow
'tableScenario1Flow'
```

```
2011-8-3 16:30:23 FunctionalTraceHel
INFO:    State 'tablePagination01' is a page state, showing page
'Pagination/tablePagination01'
2011-8-3 16:30:23 FunctionalTraceHel
INFO:    Rendering page 'Pagination/tablePagination01'
```

8.3 Client Functional Traces

BTT not only provides server traces capabilities but the tooling also generates the transaction pages in debug mode, so functional traces are shown in the web browser when the pages open, are filled with data and are send for request. The **Client Functional Trace** records pages execution and error information basically related with cross validation and ECA rules and can be used while the code is being tested during the development phase. These traces are shown in the web browser that has been configured by the technical developers to be used by default when web pages are opened. Examples of debuggers are Firebug for Firefox or Firebug Lite for IE.

Technical developers have to enable the tracing capabilities of the specific web browser to allow functional developers to use them. As an example, Firebug is an open source extension for Mozilla Firefox that assists you with debugging, editing, profiling, logging, and monitoring HTML pages, DOM, CSS, and JavaScript. When you use Firebug to debug your Web pages, the Console view and editor in the development environment and the Firebug tools can be synchronized.

Tracing messages that are generated are the following:

- **Rule execution errors**
    - ERROR. The error detailed description is appended to the message.

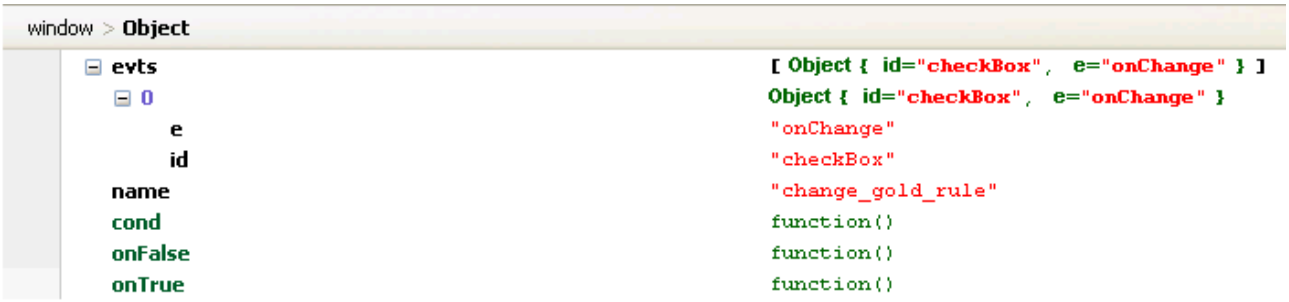
      ```
      An error occurred while ECA condition is executing.
      An error occurred while ECA on true state is executing.
      An error occurred while ECA on false state is executing.
      ```

- **Rule execution monitoring**
    - RULE START/END. Indicate the beginning or end of the rule execution. Any other rule monitoring message appears between the start and end messages.

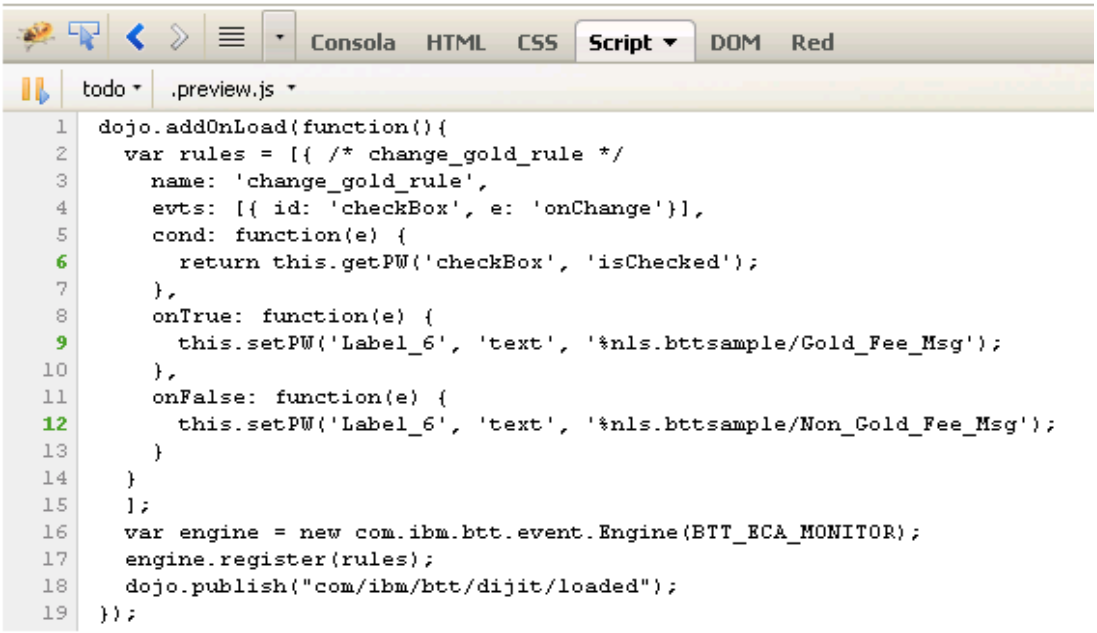
      The values that are set as part of these messages for a specific rule are the **rule ID**, the **event** that started the rule execution and the **condition** to be evaluated.

```
 Start Rule : [change_gold_rule] Object { name="change_gold_rule",
evts=[1], cond=function(), more...} for event checkBox.onChange
 End Rule : [change_gold_rule] Object { name="change_gold_rule",
evts=[1], cond=function(), more...} for event checkBox.onChange
```

To get more detail, you can click on the `more …` label of the message and you will get the following panel.



By clicking on the `function()` label, you can go the Script panel and see the details on the function implementation.



- CONDITIONS. The message shows the **result** of evaluating a **condition**.

```
Evaluated Condition : Condition = function() , Result=true
```

You can click on the `function()` label to deep dive on the condition implementation details in the Script panel.

- ACTIONS. There are several messages associated with a rule action:
  - Call function action. The **function ID**, the call **arguments** and the **value** returned after the function execution, if any, are set in the message.

```
CallFuntion Action : function_ID, args= arguments,
return=value
```

```
CallFunction Action : account_details.numberText.focus ,
args= []
```

- Set property action. The **property name** and the given **value** are set in the message

```
SetProperty Action :
Label_6.text=%nls.bttsample/Gold_Fee_Msg
```

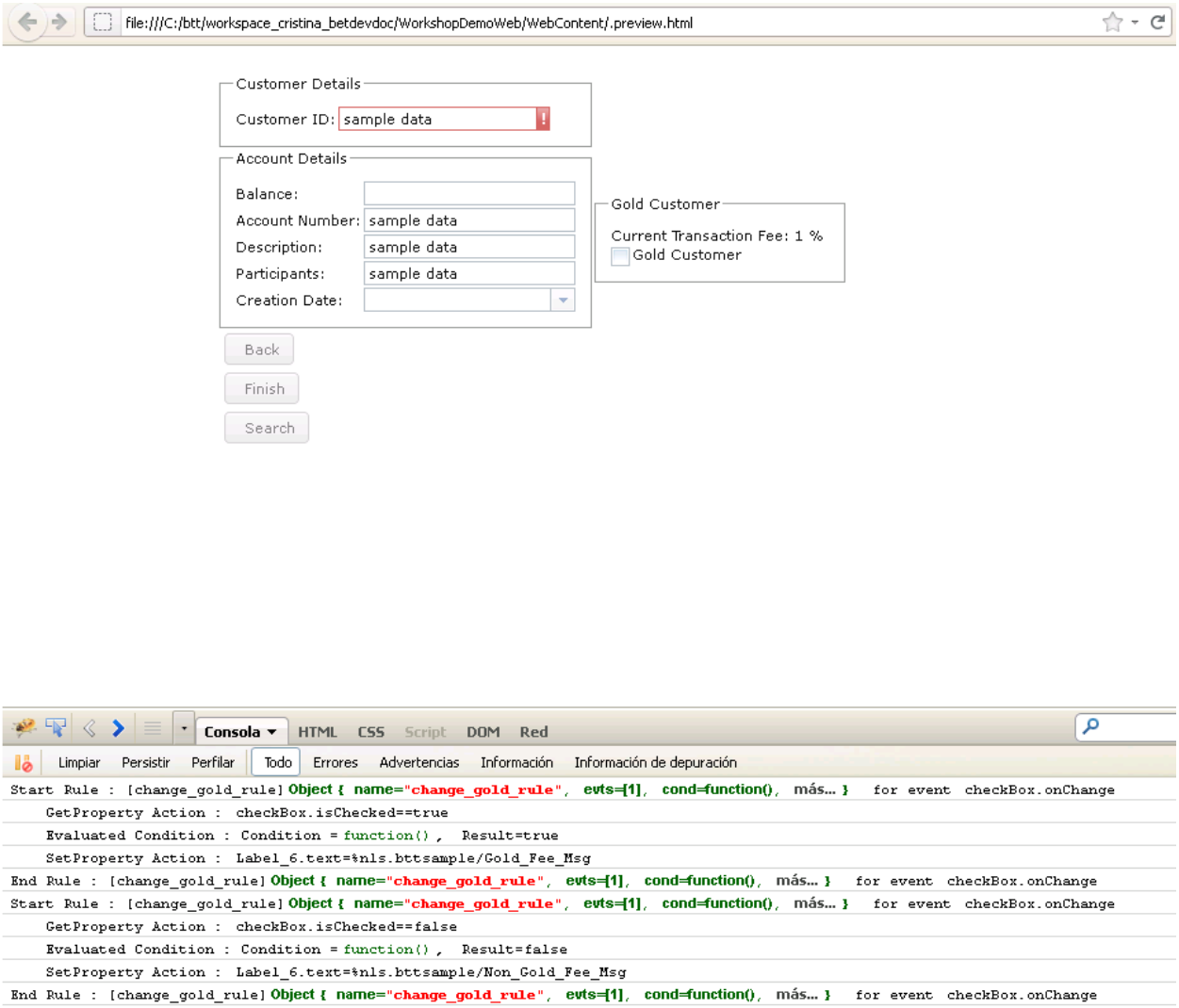
- Get property action. The **property name** and the returned **value** are set in the message

```
GetProperty Action : checkBox.isChecked==true
```

To identify the BTT contributions to the web browser console from any other, you should check for the label **btt.js** at the end of the line.

Figure 1 shows an example of the client functional trace of the execution of a view.

Figure 1. A Firebug trace of a view execution.



9. Understanding and working in a multi-project environment

A banking application possibly has multiple channels, such as teller banking, internet private banking, corporate banking, etc. These applications may be required to share some common business logic or resources such as operations and flows, NLS files or image files. At development time, the applications are organized as multiple projects in the development environment.

BTT supports reference of resources in a project from other projects. For example, developers can choose NLS definition from one project for a widget of another project. With the BTT multi-project support increases the maintainability of the application code and flexibility of project management as well as decreasing the code redundancy. Also it increases the runtime flexibility by hot deployment capabilities. For example, when part of the shared sub-flow business logic is changed or the web resource is changed, you only need to re-deploy the shared EAR, the base business-specific application EAR does not need to be restarted.

9.1 The types of projects in a multi-project environment

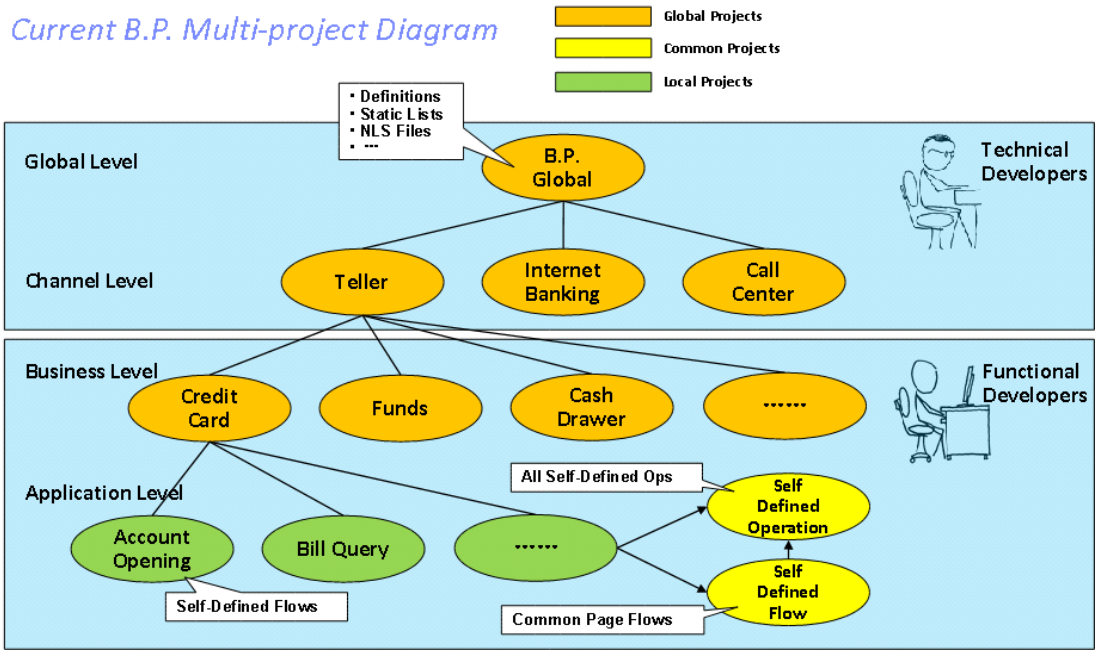
- **Global Web project for Multi-project resources**: this type of project contains web resources which are shared hierarchically with other global, common and local projects. In Global Web projects, there are the following web resources types:
    - NLS
    - CSS
    - Static Lists (for combos and select lists)
    - Images
- **Global Java project**: this type of project contains components which are shared hierarchically with other global, common and local projects. In Global Java projects, there are the following web resources types:
    - BTT Global Definitions XML files (btt.xml, common data, common types, common contexts, common services, common operations and common formatters)
    - Java classes of user extension global definitions, for example, converters or validators of types.
- **Common Java project for operations**: this type of project contains self-defined operations either generated by the operation from a web service generation wizard or created using the Transaction editor. These operations can be reused by other common and local projects.
- **Common Web project for flows**: this type of project contains self-defined sub-flows which are shared by local projects.

- **Local project**: this type of project is a BTT application web project that does not share any component with other projects. A local project has business specific logic implemented by page flows and operations that cannot be shared; however it can use shared components from global and common projects in a multi-project environment. Then, typically, a local project contains what can contain a traditional standalone BTT project in addition to the references to multi-project definitions (externally defined components).

9.2 Designing a multi-project structure

To develop application based on multiple projects, there should be a senior technical developer that design the multiple projects structure and create the project at first.

The following is a recommended projects structure diagram:



The above diagram structures global, local and common projects in a 4 level hierarchy: global bank level, channel level, business group level and local application level, and also indicates who is the responsible of the development at each level.

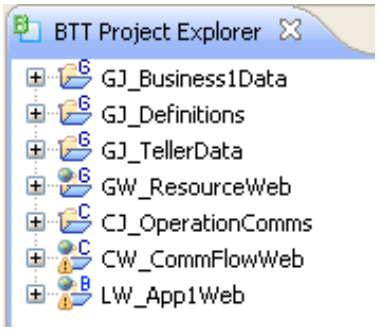
The functional developers are responsible of implementing specific business processes at the application level that will share common line of business data, formatters, services and transactions. The common projects contain back-end access operations for each business line as well as general flows which are transversal to the business; both operation and flows will be reused by the locally implemented self-defined flows.

Once the multi-project structure has been created by the technical developers, the functional developers will be able to use the shared components in any transaction definition. There are two ways you have access to the shared components:

- the fist one is through the different creation or editing wizards; each time you refer from a wizard to an operation, a sub-flow, a context, data, types, formatters, services

or any other resource, all the components in projects that have been configured as shareable will be shown in the selection list together with the local components

- the second one is through the Project Explorer. On a multi-project environment, the BTT Project Explorer shows all BTT Projects you have access to, independently of the nature (local BTT project, global or common)



The following section shows how the definition of different shared components impact the editing of local components in the BTT Perspective.

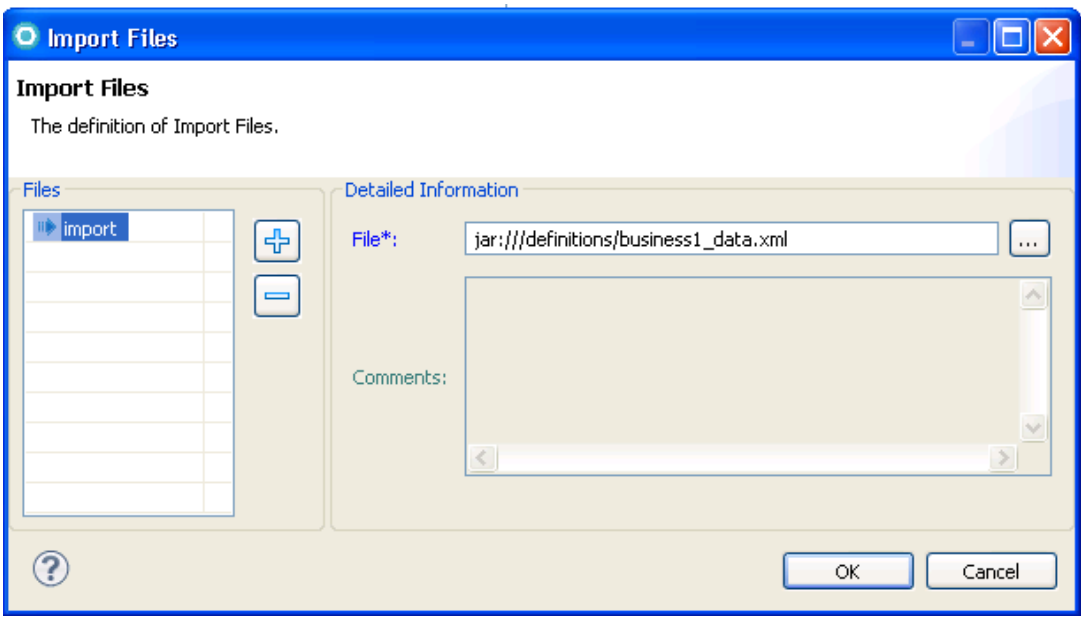
9.3 Multi-project in the BTT Perspective

The following samples show how the proper configuration of a multi-project environment provides the functional developers with access to global and common projects components and how these components can be identified during the transaction development process.

There are two ways to share different BTT components and resources on a multi-project environment:

- importing the definitions into the global definition files. In this case, these components appear in the Project Explorer in the common folders and there is no difference between those that are defined in the current project or those that are defined in a different project. In the reference selection panel, when setting a value for a refId field, these components appear in the Global folder.

```
<?xml version="1.0" encoding="UTF-8"?>
<data.xml>

  <import file="jar:///definitions/business1_data.xml"/>

</data.xml>
```



- setting in the btt.xml files which are the projects that are going to share their content as part of a multi-project environment.

The BTT components in these projects (self-defined transactions elements and global web resources), when being referenced, are internally identified by adding a prefix to their given Id. In some cases (remote self defined operations and flows, global operations) the prefix is also shown in the BTT perspective, so the functional developer can easily understand that they are defined in another project and identify the project in the workspace. The prefix that identifies each of the projects is set in the btt.xml file and is configured by the technical developer. The following example shows the definition in the btt.xml file of the projects that share their components with the local project being defined and the id associated to each one of them, that is used as prefix (the project **CW_CommFlowWeb** uses prefix **CommFlow**).

```
            <kColl id="remoteProjectURL">
                  <field id="CommFlow"

value="http://localhost:9080/CW_CommFlowWeb/"
                        description="CW_CommFlowWeb" />
                  <field id="GW_ResourceWeb"

value="http://localhost:9080/GW_ResourceWeb/"
                        description="GW_ResourceWeb" />
                  <field id="GJ_Definitions" value=" "
                        description="GJ_Definitions">
                  </field>
                  <field id="GJ_Business1Data" value=" "
                        description="GJ_Business1Data">
                  </field>
                  <field id="GJ_TellerData" value=" "
                        description="GJ_TellerData">
                  </field>
                  <field id="CJ_OperationComms" value=" "
                        description="CJ_OperationComms">
                  </field>
            </kColl>
```

The general understanding is that the multi-project environment configuration should be as much transparent as possible for the functional developer, which should have access to the appropriate BTT components and resources without knowing their origin.

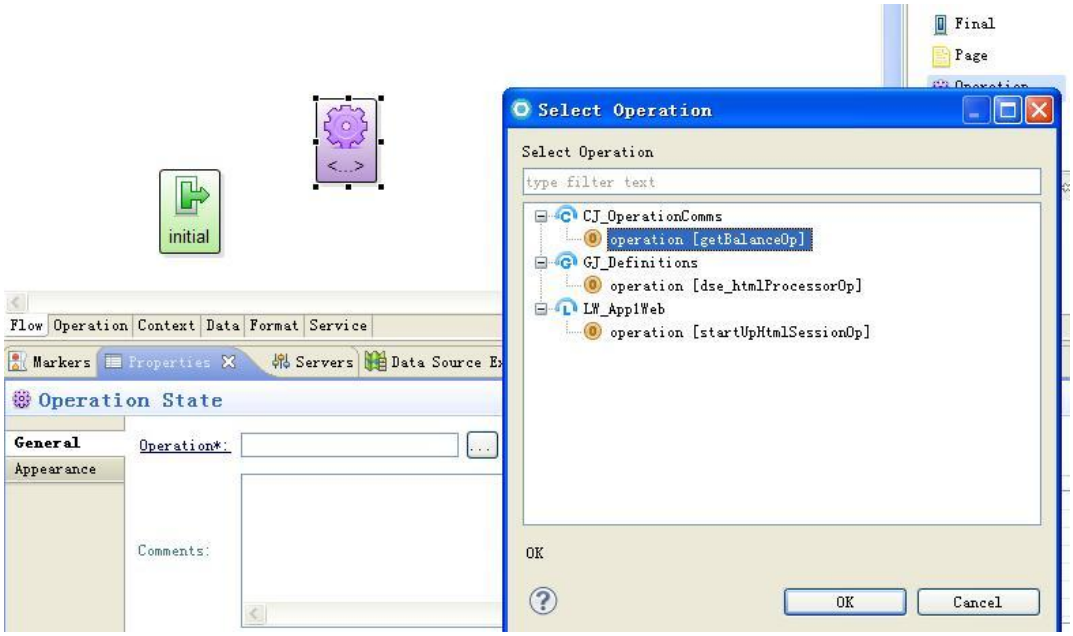
In the following sections there are some samples that show how a functional developers manages the different BTT components and resources on a multi-project environment. In order be able to understand the multi-project configuration that is used during the explanation, a prefix has been added to the project names (this is not part of the product. The tooling allows you to identify the projects by their nature: Global, Common or Local):

- Global Web projects start with prefix GW
- Global Java projects start with prefix GJ
- Common Java projects start with prefix CJ
- Common Web projects start with prefix CW
- Local projects start with prefix LW

9.3.1 Adding self-defined operations and flows

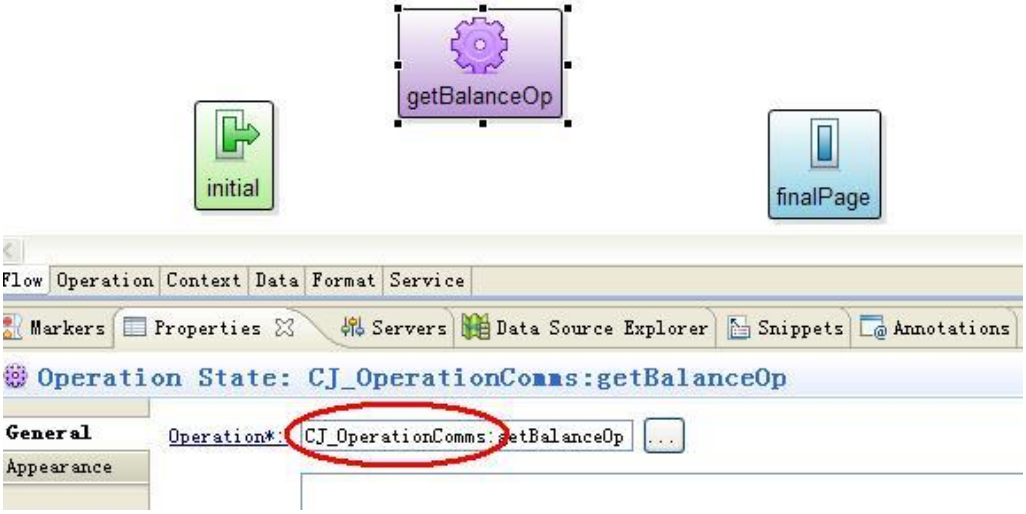
Adding an self-defined remote operation to a flow using the Flow editor

In a multi-project environment, when you want to assign an operation to an operation state during a flow design using the Flow editor, the operation selection dialog shows not only the local operations but also the global and common operation defined as accessible from your environment, as shown in the Figure below:



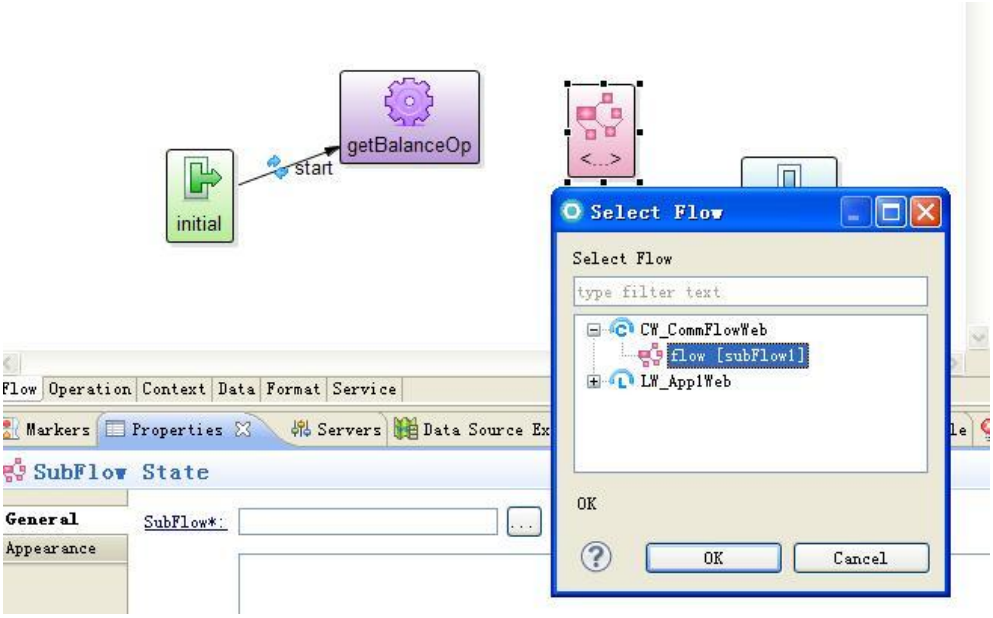
If you select an operation that belongs to a common project that has been configured as part of the multi-project environment, its name is prefixed with the name of the common

java project that actually contains the self-defined operation. This is to indicate that this operation is a common shared operation defined in another project.

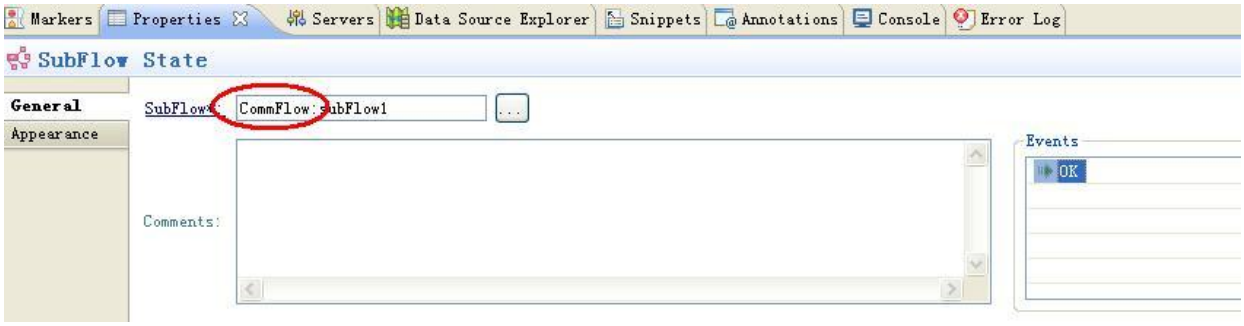


Adding a self-defined remote sub-flow to a flow using the Flow editor

In a multi-project environment, when you want to assign a sub-flow to the SubFlow state in a flow design process, the sub-flow selection dialog shows and not only the local flows are listed but also the common flows defined as accessible from your environment, as shown in the Figure below:



If you select a sub-flow belonging to a common project, its name is prefixed with the name of the common java project that actually contains the sub-flow definition. This is to indicate that this flow is a shared flow defined in another project.



9.3.2 Adding components from a Global or Common Java Project

Most commonly global components that are reused in other projects are data and types.

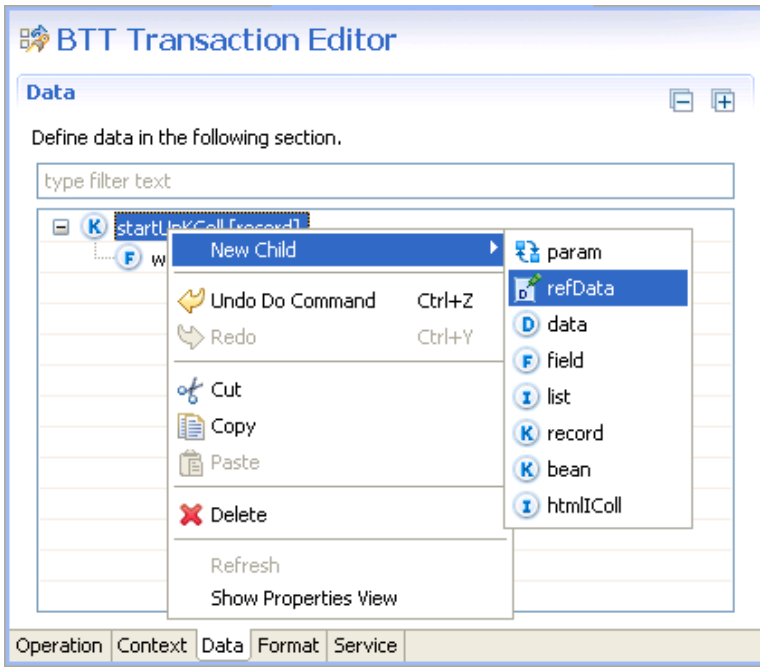
In some places during a transaction definition you need to add a reference to a specific BTT element already defined in the workspace; for instance, when adding data to a context, when adding a service to a context, when adding a formatter to an operation, adding a type to a data…

In the selection panel, the components are normally organized in three different folders: Local, Global and Import. In a multi-project environment, all components that are available to the local project but are defined in another global java project are listed under the Global directory.

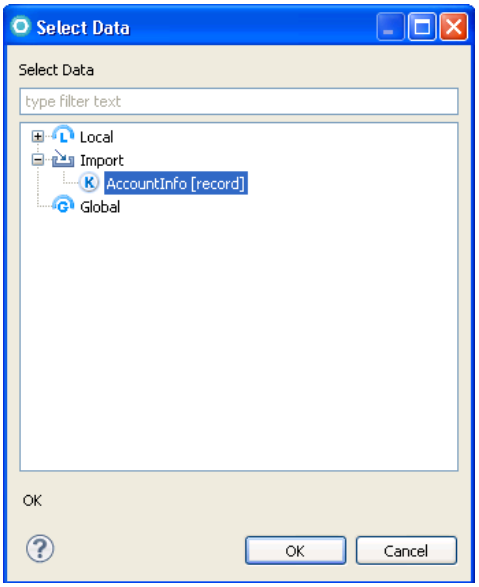
In some other cases, you may want to make a reference to a remote context, for instance when adding a context to a view, an operation or a flow. In this case the selection dialog shows all visible contexts from the local project, either from a global project or from a common project. See the examples below.

Adding remote data to an operation context

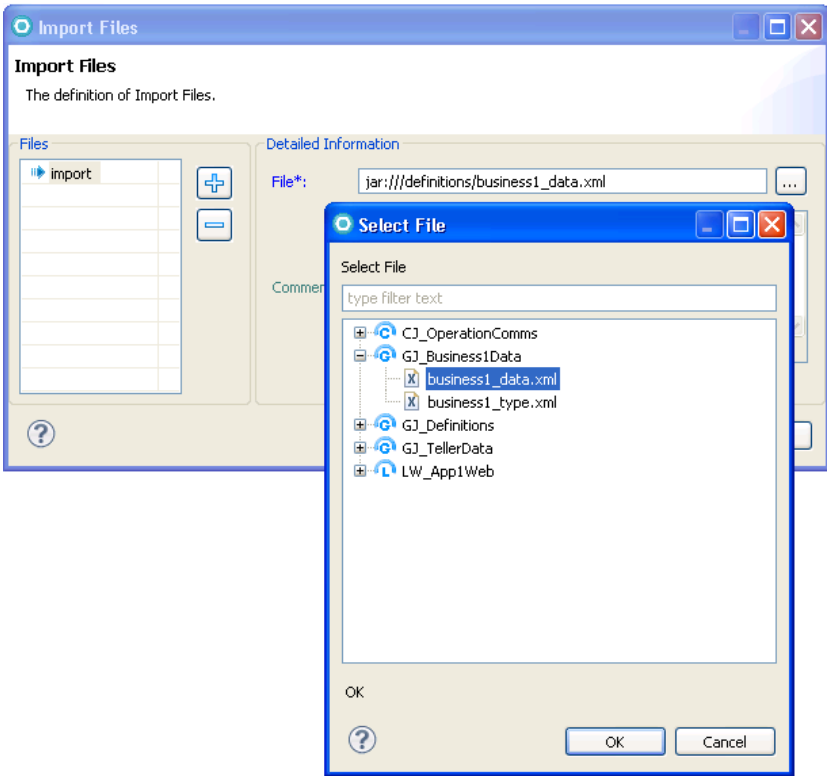
As part of an operation definition, you need to define the context data that the operation is going to manage. In the BTT Transaction editor, you can go to the Data tab to start defining the data, inside the record associated to the operation context. As you want to add a data defined in another project, you select the refData option.



The Detailed information panel opens and a RefId is requested. Clicking on the icon  the list of available data is shown. The resources coming from global web projects can be, depending on the multi-project configuration, in the Import or Global directory.

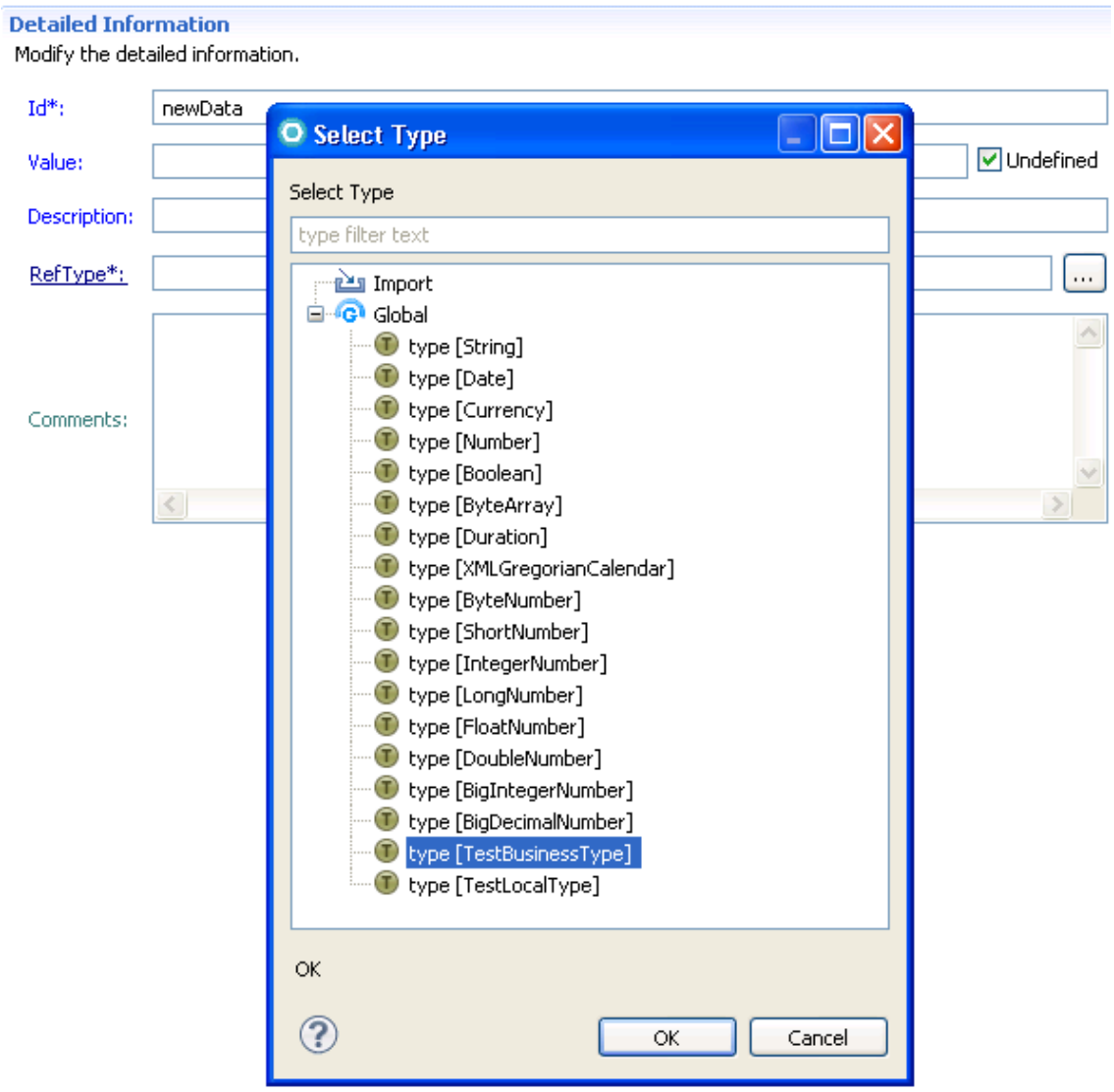


In our example, the AccountInfo record is defined in a global java project but it has been imported as part of the multi-project configuration and then is visible from our local project in the Import folder. In this case, you don't know what the project that actually holds the data definition is. If the element is in the Import folder and you want to know who is defining this data, you can right-click on the blank area in the Data table and select the context menu option Transaction Editor > Add Import Files. Selecting one of the import entries and editing it, you can check the origin project of the specific data are using.



Adding local data of a type defined on a global java project

You may want to add some data to your local project data dictionary but reuse some of the already defined types in a global java project. You go to your local project Definitions > Data Dictionary folder and right-click on the Data table to add a new data. You select the NewChild > Data option and are prompted with the Detailed Information panel were the RefType for the new data must be set. Clicking on the ... icon you get the list of possible types to select, including those defined in the global java projects shared in your environment.



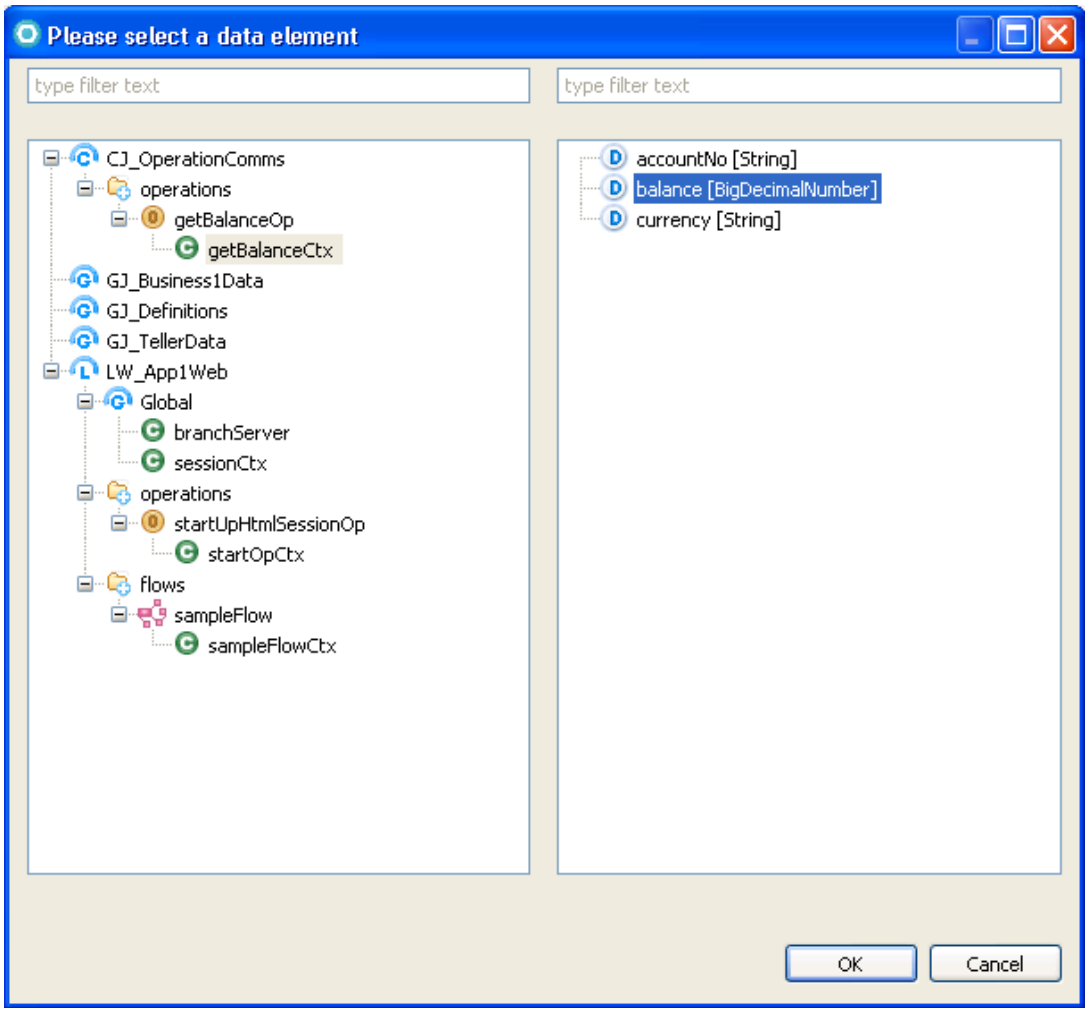
In our sample, the remote type is `TestBusinessType` and is defined in project `GJ_Business1Data` . From the functional developer perspective, there is no difference between local or remote resources. In this case that the type definition does not come from an import, the only way to check for the project that contains the definition is to navigate through all project definitions in your BTT perspective, although in most of the cases this is not needed.

Adding a data element associated to a widget

In the XUI editor, as part of the view definition , you may want to map the view widgets to the context data elements. Select a widget, go to Properties view, and click the Browse button for the dataName property. In the data selection dialog, the tooling lists the imported data and types. In general, the data name is selected from current flow context and its hierarchy in the local project (when imported from a global project, the context appears in a Global directory); alternatively the data name may be selected from a self-defined operation in a common project.

In the example, a reusable operation has been defined in a common project and the view widget (selected Label in the Figure below) is going to be bound to the balance data element that is part of the operation context; the view is going to show the data resulting

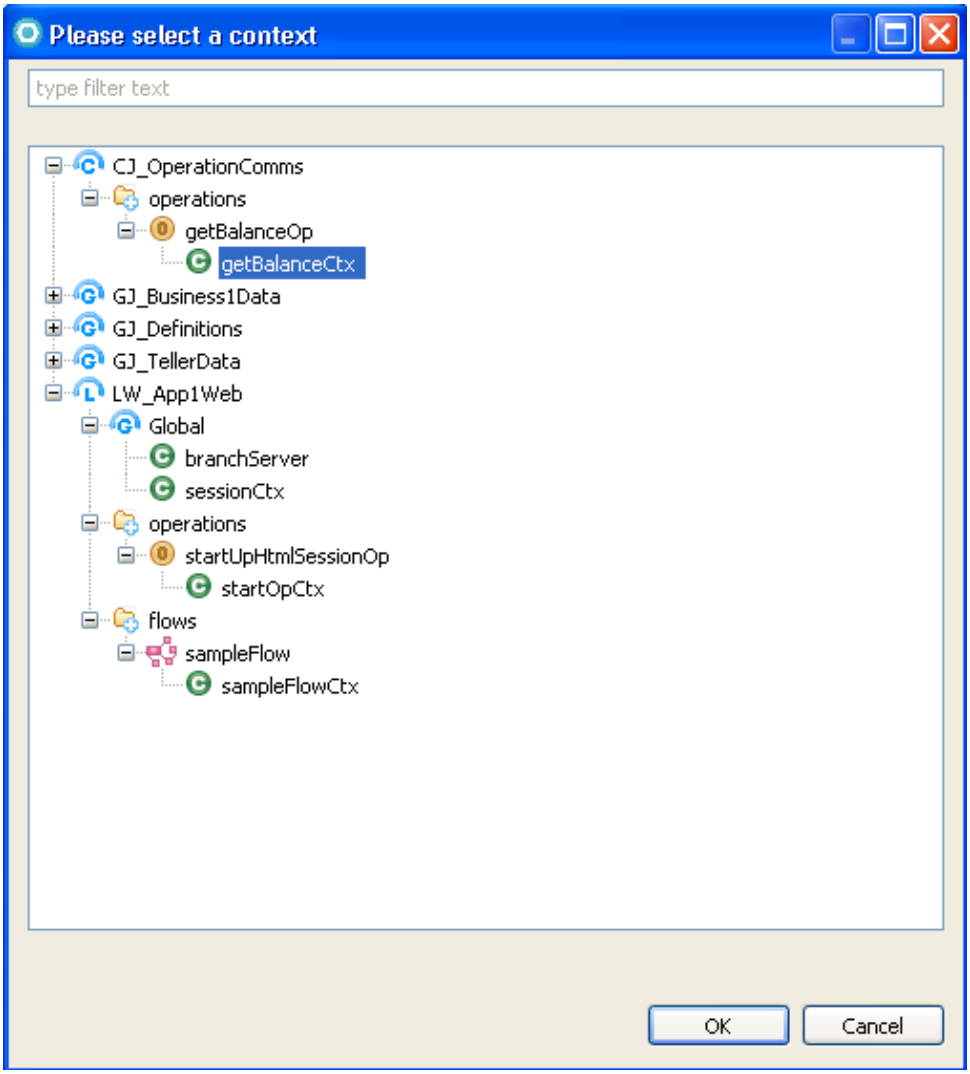
from the operation execution.



Adding a remote context to a view

In the XUI editor, as part of the view definition, you may want to set the view context. You click on the gray area of the XUI editor, go to the Properties view and click the

Select Context button; the context selection dialog is pop up. The context can then be selected from a project hierarchy containing all projects visible from the current project, either from the local project or from a global or common project. In our example, as the view is showing the results of the operation execution, the view context should be set to the self-defined operation context defined in the common project, as shown in the Figure.

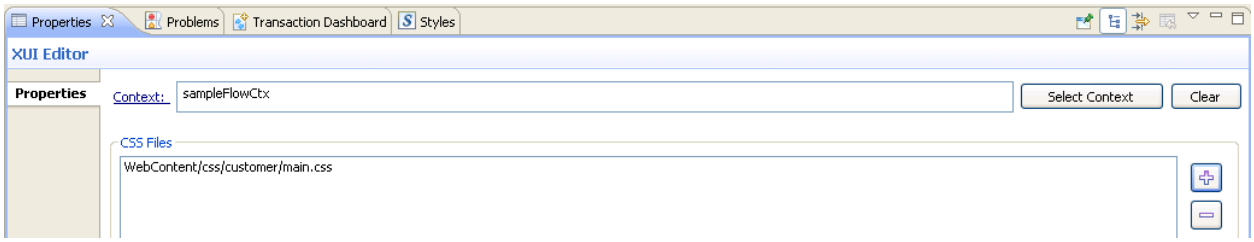


9.3.3 Adding resources from a Global Web Project

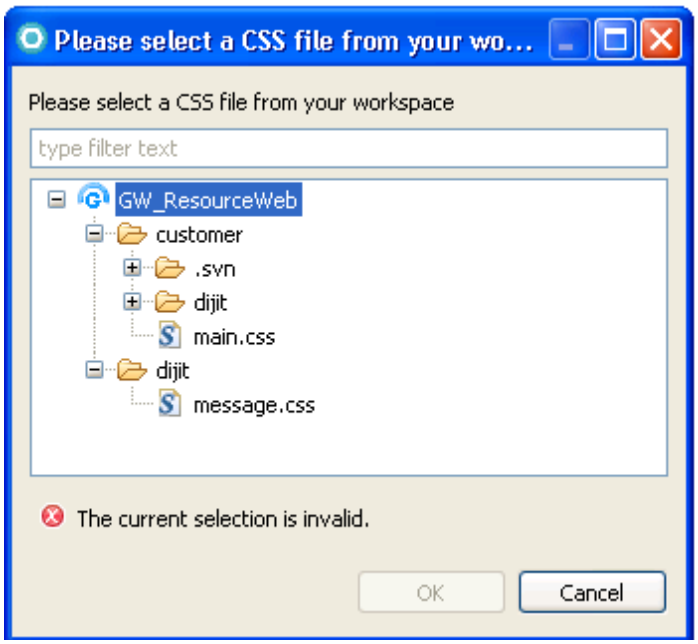
When editing a view in a multi-project environment you can use resources that are defined in other projects. In following sections an example is used to illustrate how the remote resources can be referenced during a view design using the XUI editor.

Adding a remote CSS style to a view in the XUI editor

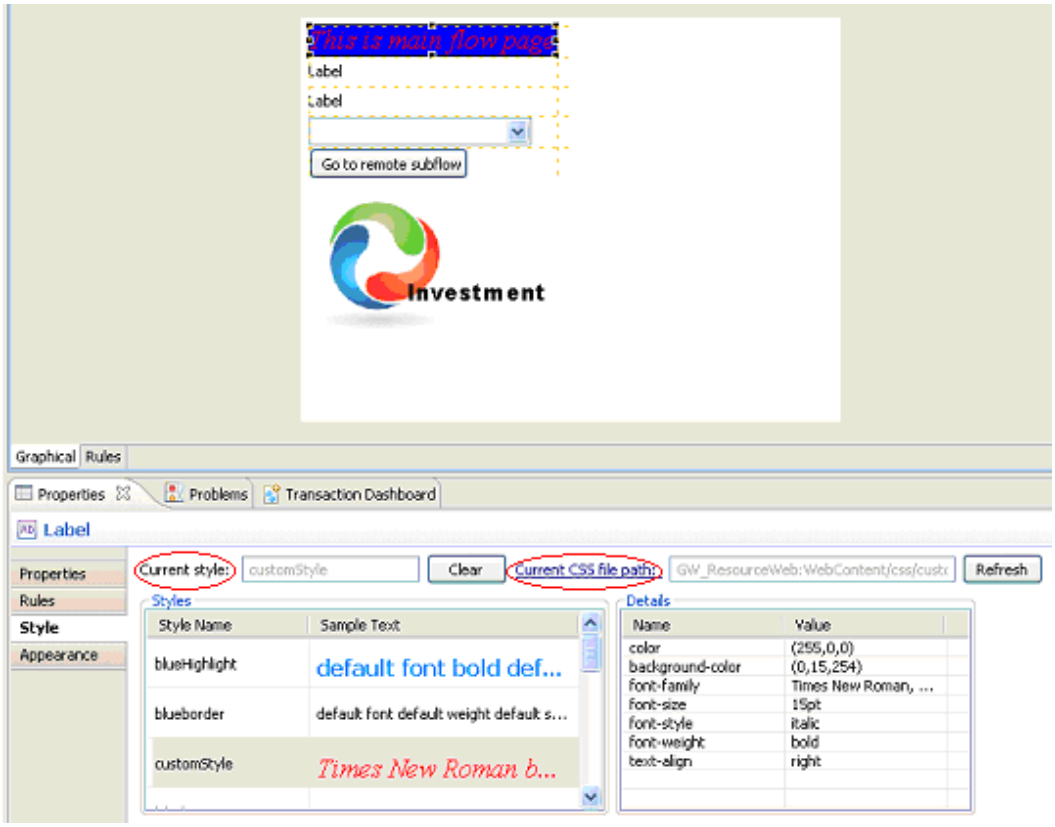
To set the style associated to a view, you open the view in the XUI editor and go to the CSS file table in the Properties panel.



If you click on the  icon, you get the list of all available style files. In our sample, the style files are available in a global web project.



To check for the style associated to a view widget or container, you select the widget or container, and the Properties panel for the selection opens. You can then go to the Style tab to view the current style associated to the widget.



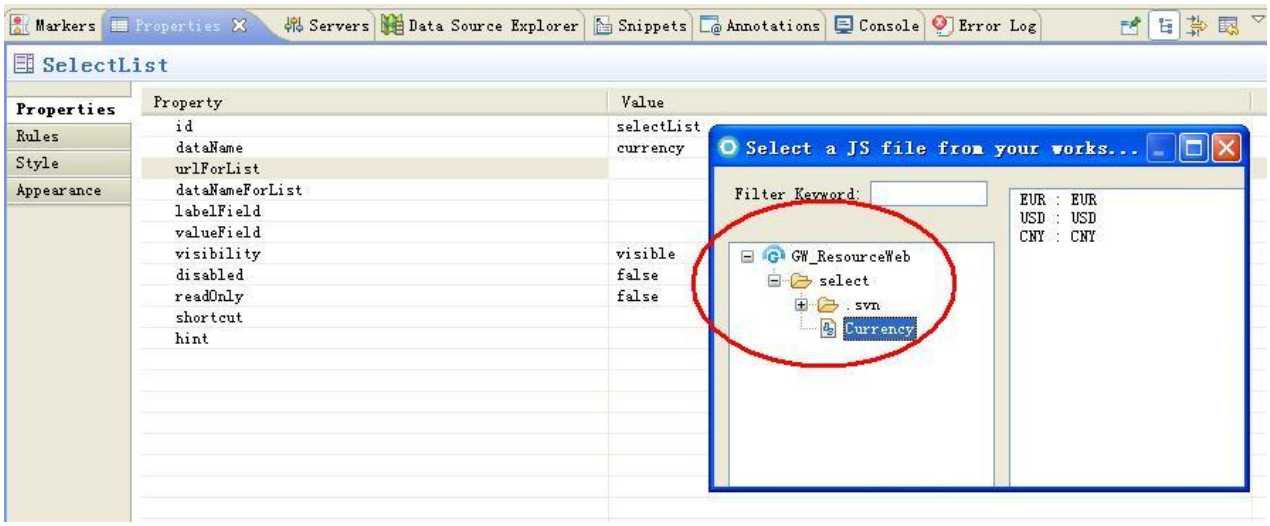
In the Styles table, you have all available styles, including the ones defined in other projects that have been shared with yours in the multi-project configuration. As an example, the current style selected in the sample is defined in project GW_ResourceWeb, a global web project. The name of the project that contains the resource definition can be obtained from the Current CSS file path field, in the Properties > Style tab. In the example, the path points to:

GW_ResourceWeb: WebContent/css/customer/dijit/main.css

The path is automatically set when the style is selected.

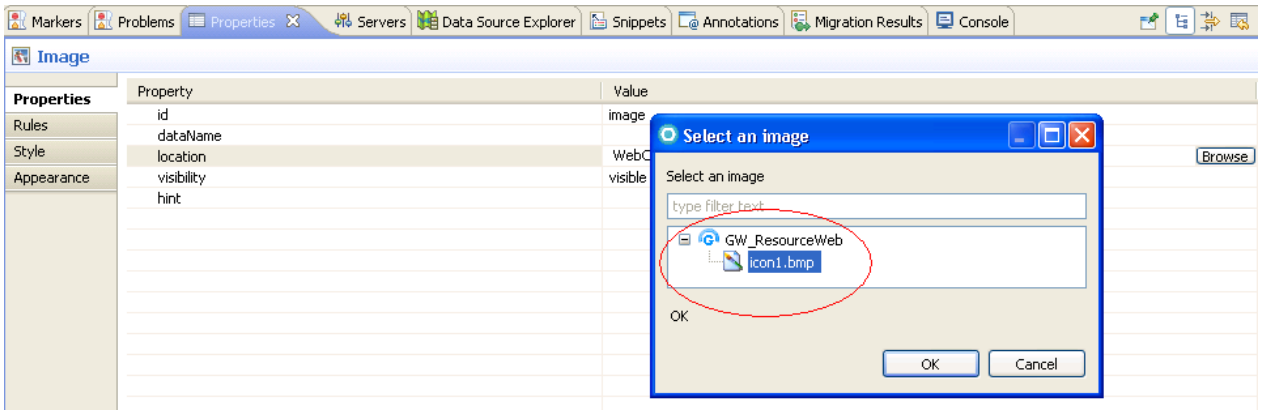
Choosing the list of values associated to a selection widget

For the selection widgets (for instance a Combo or SelectList) you can set the list of possible values by selecting a remote list (a static file with a list of values) available in a global web project. Clicking on the property urlForList the selection panel opens and the local and shared resources are listed.



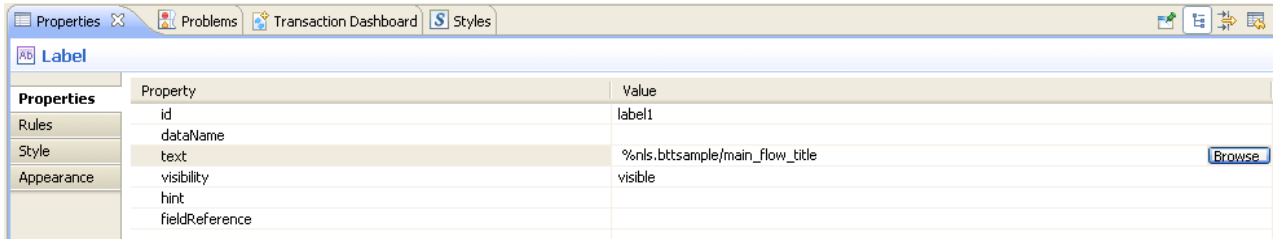
Selecting icons for an Image widget in a view

When an Image widget is added to the view, you have to set the location of the icon that contains the image in the Properties panel. If the multi-project support is configured, this icon can be located in a global web project, as shown in the Figure below.

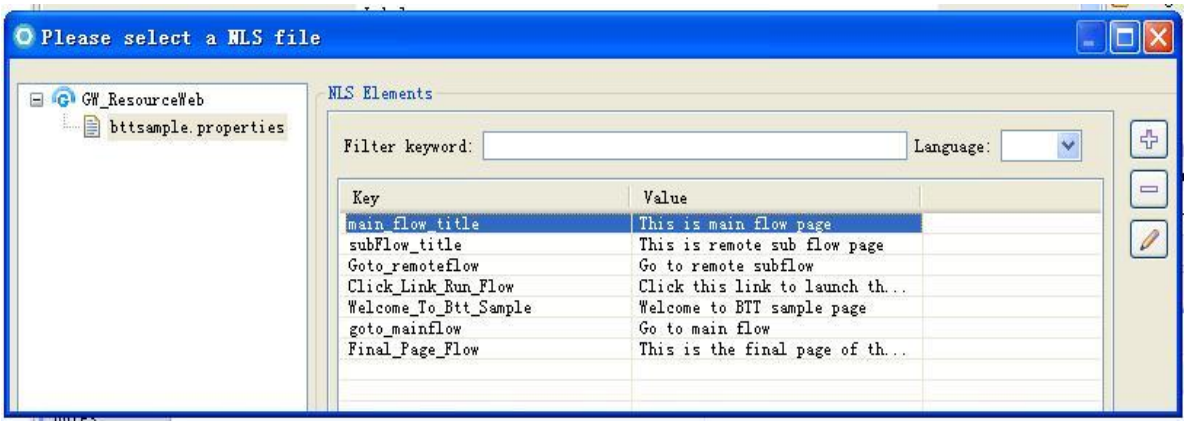


Referring to an NLS properties file when editing a view

When adding to a view a widget that contains a text (for instance, a Label widget), the tooling allows to select the properties file that is going to contain the associated text, as part of the NLS enablement.



The available properties files are listed and in a multi-project support environment, those defined in global web projects are also shown, as in the example.

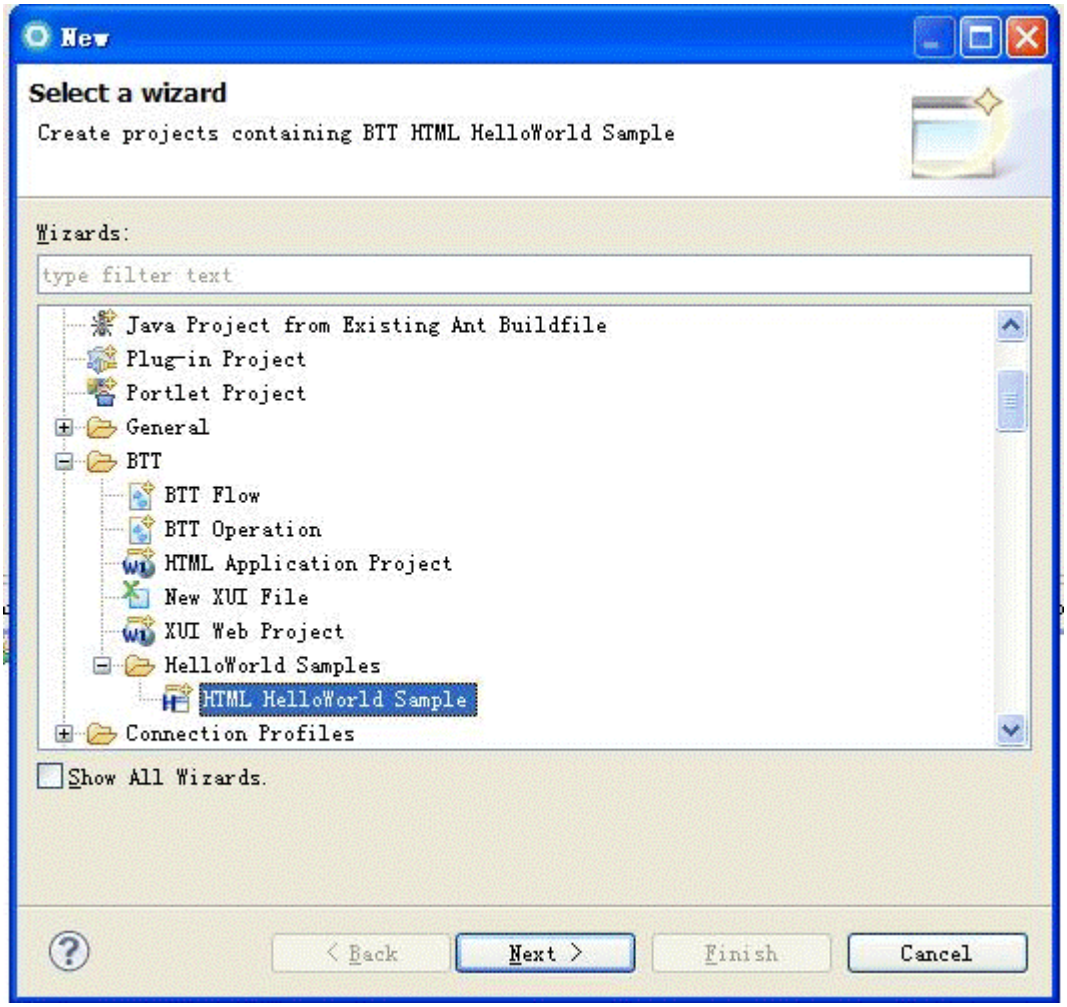


When you add new NLS item, the content will be added to NLS file of the global project. In this case, you need to build the global web project before the application deployment to generate the NLS related runtime artifacts. To build the project, please right on it and select the context menu option ‘Build BTT Project’.

10. Creating multichannel applications using the BTT perspective

10.1 HTML Hello World sample

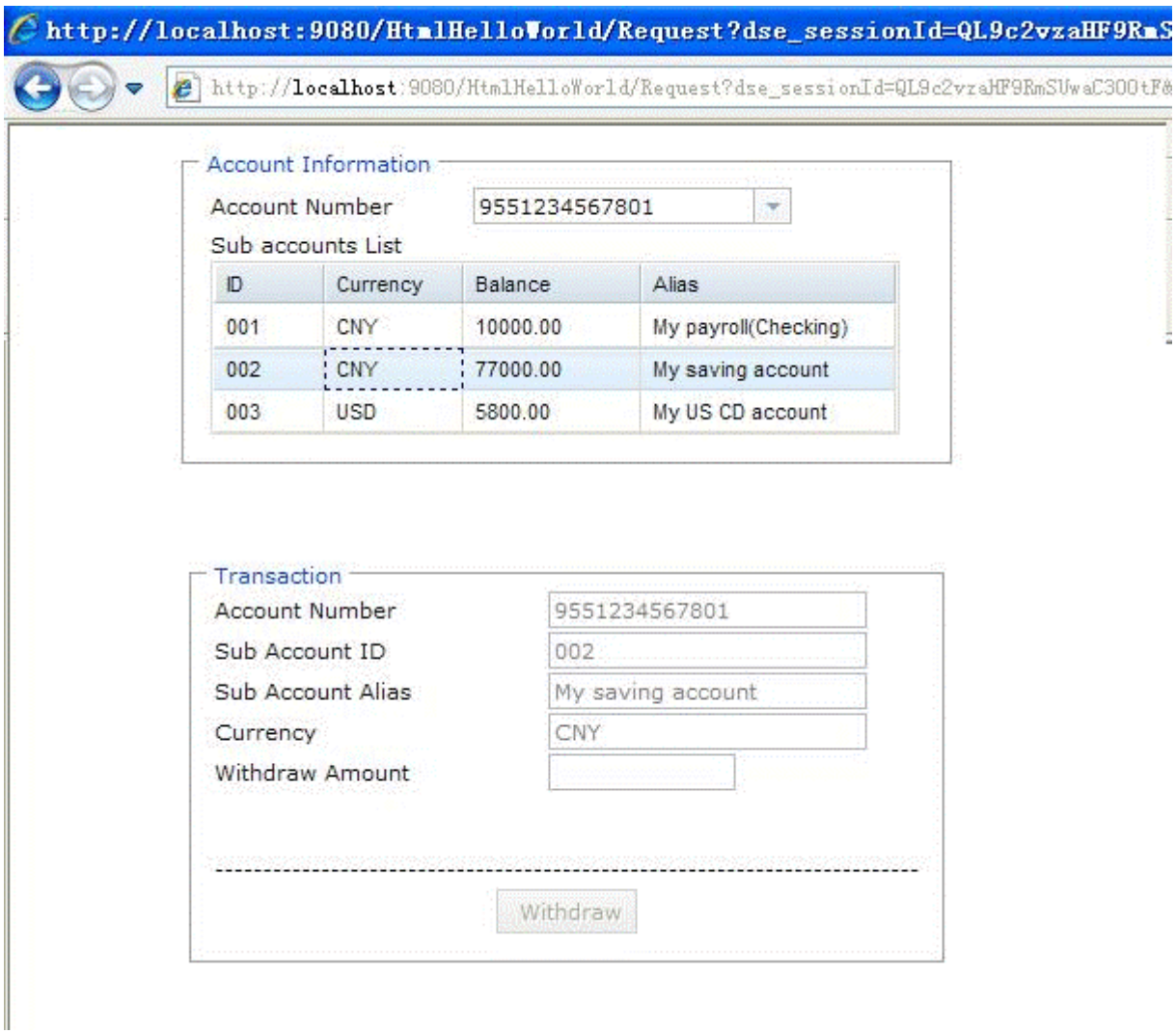
This section describes how a functional developer will implement a simple application using the BTT Perspective. This application has been designed and implemented to show the most important steps in a WebSphere® Multichannel Bank Transformation Toolkit HTML Channel project and is fully described in section Creating multichannel applications. In this section only the tasks that must be performed to develop an application by using the BTT Perspective are described. For clarity, a summary of the application description and design have been also added to this document. The complete sample implementation can be installed in the RAD environment. In the Rational® Application Developer workspace, click Menu > File > New > Others > HTML HelloWorld Sample. Then click Next, and click Finish.



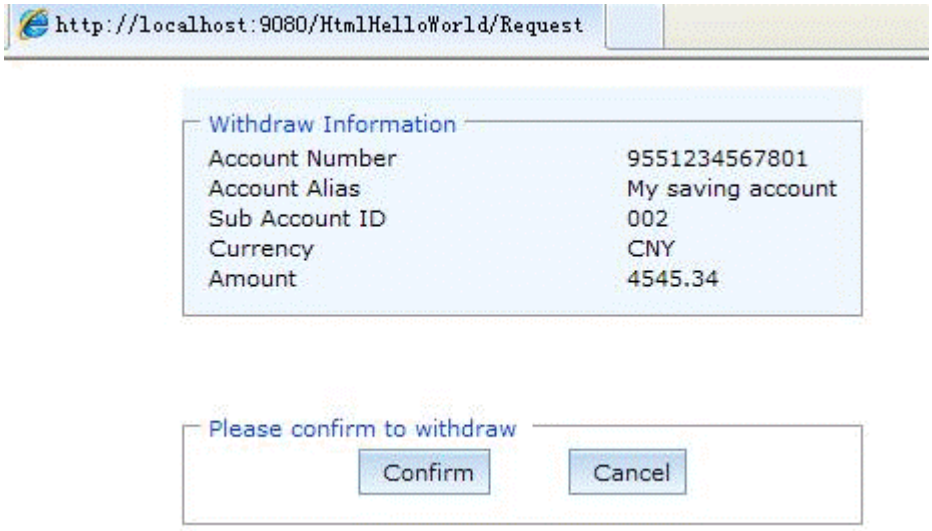
The Html HelloWorld Sample is imported into the workspace.

The application that we are building is a withdrawal transaction that allows the final user to select an account from a list of accounts, retrieve some account information, enter a

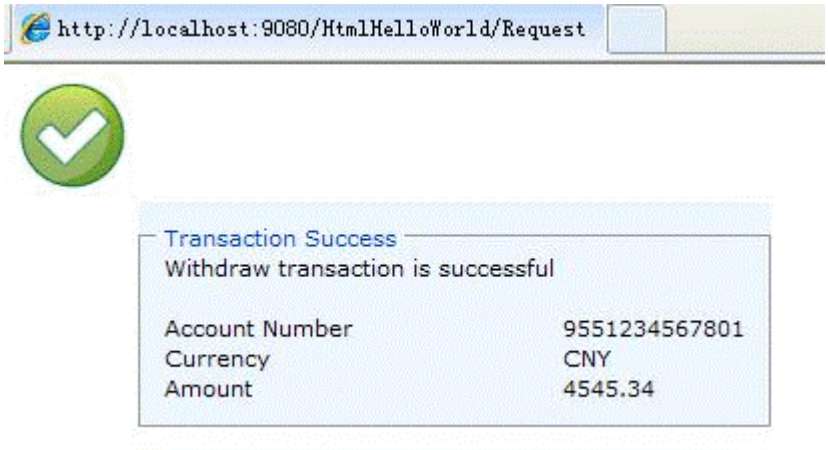
withdrawal amount and execute the transaction. It will have a main page that is displayed below



You can change the account number in the selection box, and then the sub account table will be refreshed. You need to select a sub account to withdraw; then the account information will be filled into the Transaction group. Once you enter a valid withdrawal amount, the Withdraw button will be enable for you to submit the transaction.
Clicking the Withdrawal button the next page is shown. The transaction information is displayed with a Confirm and Cancel button.



If you click Cancel, the previous main page is displayed. If you click Confirm, the result of the transaction is displayed



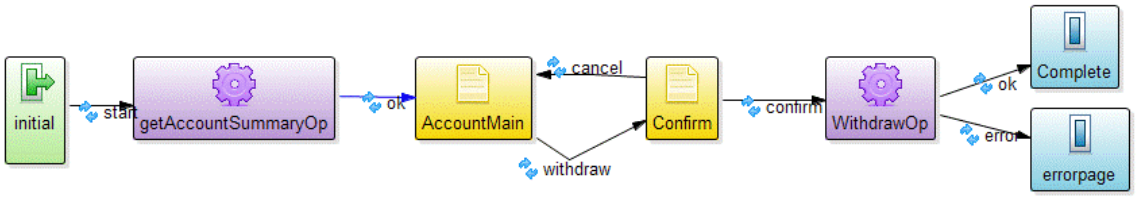
10.1.1 Designing the application

To create a transaction, you must first design the business flow of the transaction; we need to break down the transaction into operations and views, and the transition between them. Then we will design the data context of the flow and operations and the data map between the operations and flow context. At the same time, we will design the UI layout and data binding between the UI widgets and data context of the flow and the operations.

Designing a transaction flow

The transaction flow is designed according to the business scenario and requirement. *Figure 1* shows the withdrawal flow for the bank withdrawal sample we are implementing.

Figure 1. A sample transaction flow for a bank withdrawal.



In the withdrawal transaction flow, there are two operations and four pages, and they are linked by transition events. *Table 1* provides a description of the different states that have been included in the withdrawal transaction flow.

Table 1. Flow states of the withdrawal transaction flow in the HTML HelloWorld sample

Initial	Page	Mandatory initial flow state indicating the start of the transaction execution.
getAccountSummaryOp	Operation	Get account list of the user.
WithdrawOp	Operation	Performs the withdrawal by accessing the backend system.
AccountMain	Page	Main page of the withdrawal transaction.
Confirm	Page	Confirmation page for withdrawal.
Complete	Page	Result page after transaction successful execution.
errorPage	Page	Result page after transaction error.

Designing the context data

When defining the transaction flow and operations, you must define the data and services that will be managed by them. For the example scenario of a withdrawal, the flow context is going to keep all the required data to move between flow states such as sub account detailed information, selected sub account id, and withdrawal amount for the transaction. The data in the flow context can be shared across pages and operations in the transaction flow. The operations contain also a operation level context data keeping the data that will be used during the operation execution, such as the account information for the getAccountSummaryOp operation. The operations can use data from another flow state by mapping input data from the flow context and can share its data after execution with the other flow states by mapping the output data to the flow context.

Designing a view and binding the context to it

Finally we define the layout of the views and the interaction between different view components (field validation, cross fields validation, ECA rules,…). We also need to specify how the transaction data is bound to the view elements.

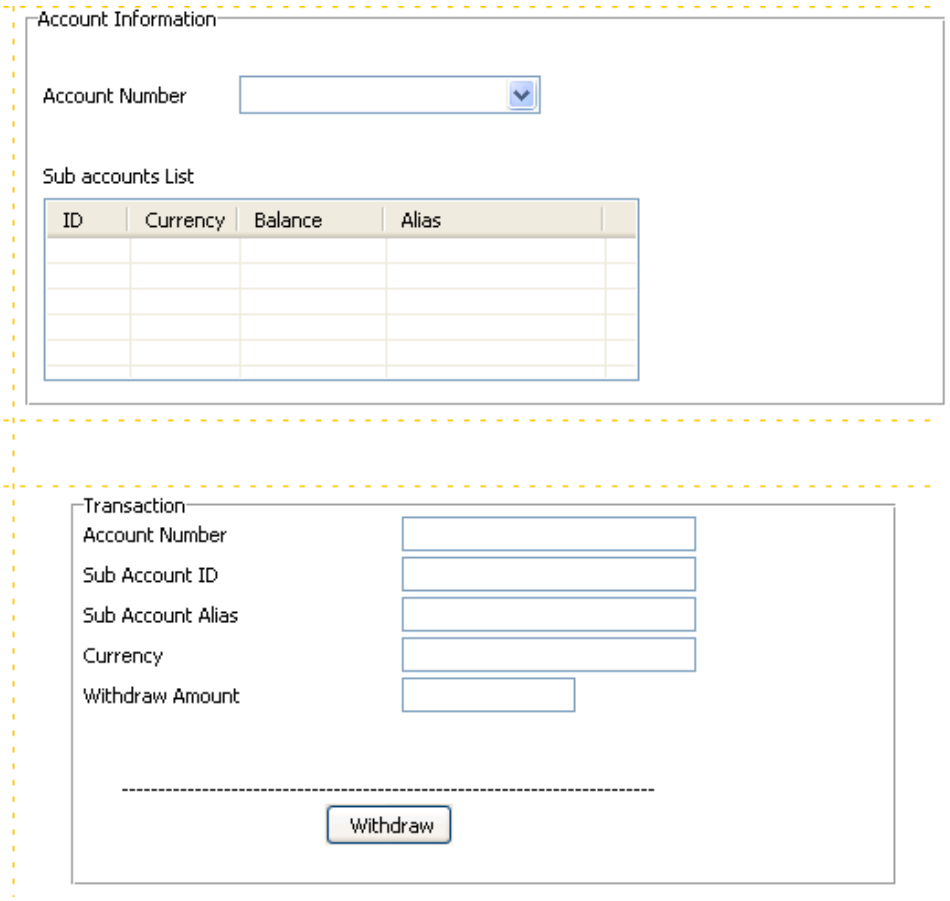
Figure 1 shows the example AccountMain view, main page of the withdrawal transaction, that is created with the WebSphere® Multichannel Bank Transformation Toolkit XUI editor.

The main account list is displayed in a SelectList widget, and the sub account detail information is showed in a Table widget. In the Transaction group shown in *Figure 1*, text widgets are used to display data from the sub account that has been selected for withdrawal. These text fields are filled with data automatically by applying an ECA rule and are refreshed when the user selects another account number.

For each widget on the view, the "dataName" property is used to bind the widget to a data element in the any of the available data contexts. For the main view Account Number widget and text fields in the Transaction group, the data is bound to the flow context. For the sub account list table, the data is bind with the operation context of an operation named "GetSubAccountDetailsOp ". This is an internal operation that is used to keep the view data, as the views do not have an associated data context.

During the application design phase, the context data and the UI might be designed and created at the same time.

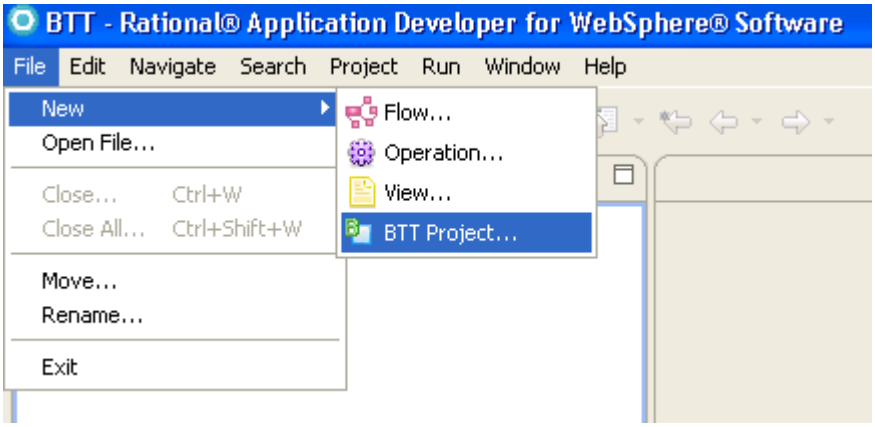
Figure 1. The AccountMain view



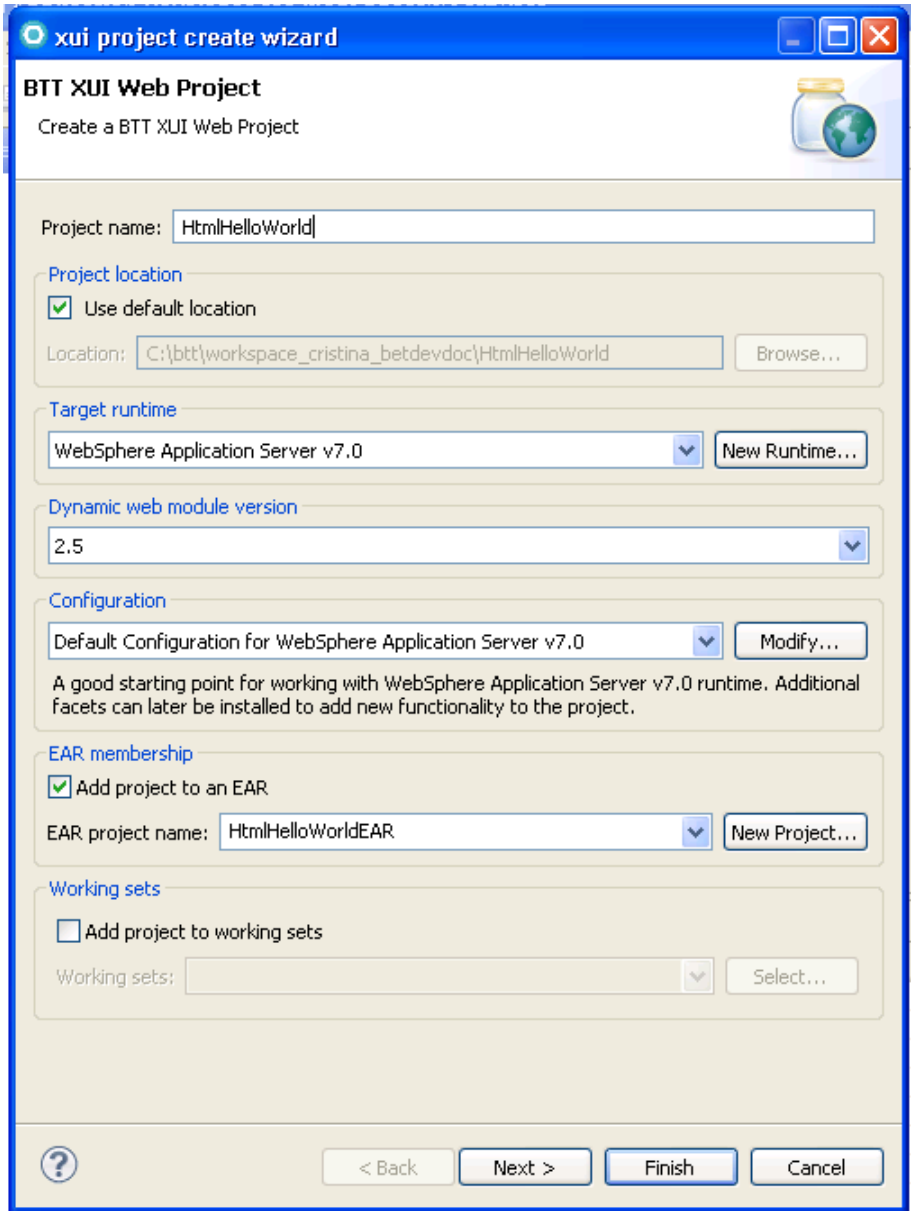
10.1.2 Developing the application using the BTT perspective

Creating the BTT project

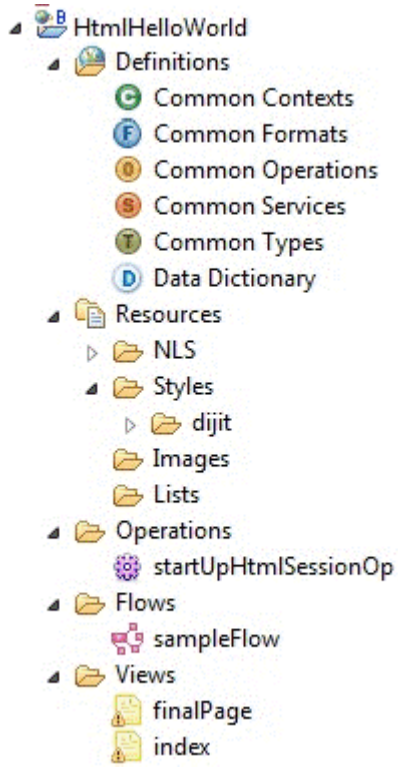
The first step of the developing the WebSphere® Multichannel Bank Transformation Toolkit HTML application is to create an XUI web project. In Rational® Application Developer **BTT perspective**, click Menu > File > New > BTT Project…



Enter the Project name, HtmlHelloWorld.



After creating the BTT Project, in the **BTT Perspective** you will see the project structure like following diagram. WebSphere Multichannel Bank Transformation Toolkit tooling generates all the base-required files of WebSphere Multichannel Bank Transformation Toolkit framework like definitions, resources and required tooling configurations.



Creating context data

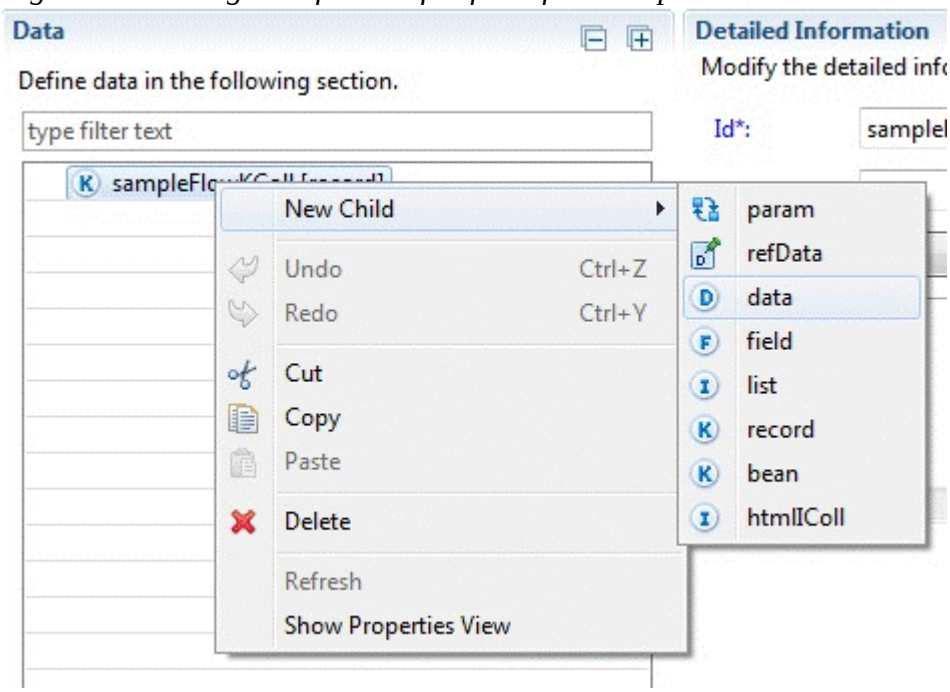
Now you can create the context data of the transaction according to the design.

A project data can either be created in the global common data dictionary, so it can be reused by the different transactions in the project or transaction specific data can be created in a self-defined flow or self-defined operation.

It is normally recommended to use as much as possible the global common data dictionary, although for this sample we will create the data in either the flow or the operation.

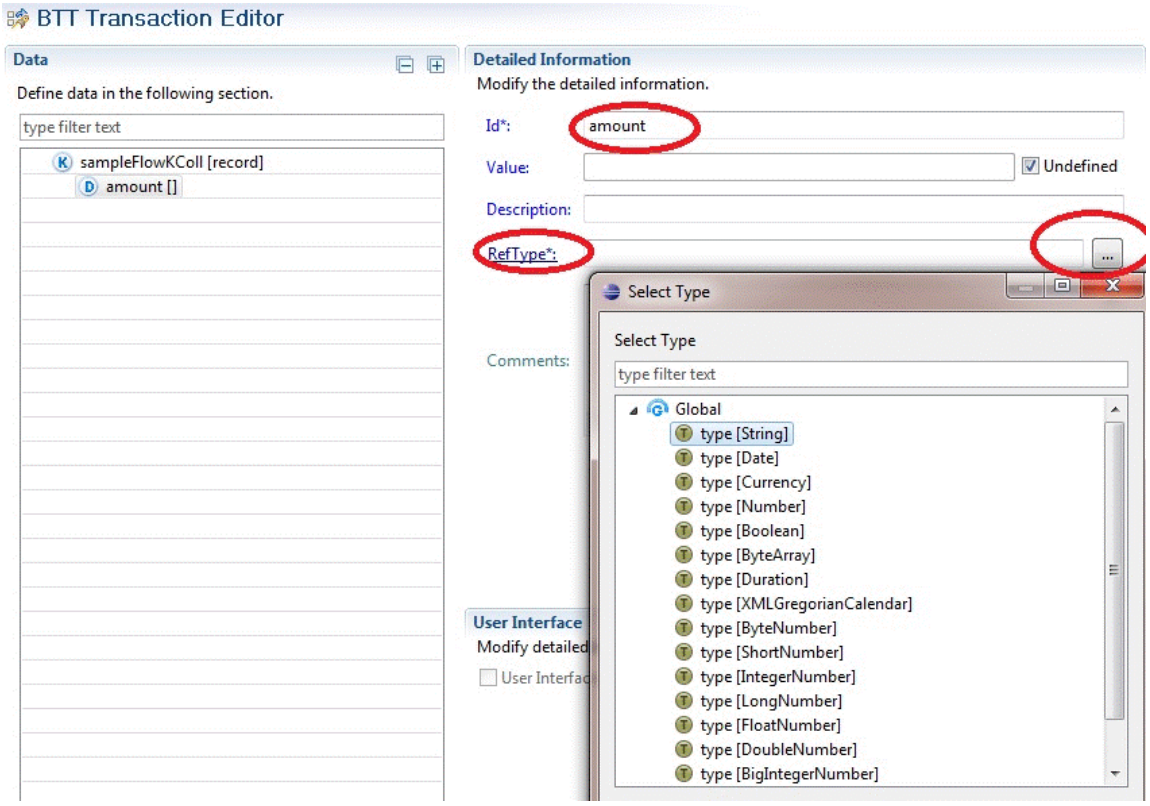
When the BTT Project is created, a **sampleFlow** flow has been by default created for you. The flow will have already an associated context, **sampleFlowCtx**, and a defined keyed collection, **sampleFlowKColl**, that is the root data for the context. Open the sampleFlow with the Transaction editor, double-clicking on it, and then go to the Data tab. For example, we will show how to add the **amount** element. We can add new data element by right click on the **sampleFlowKColl**, and then click New Child > data as shown in *Figure 1*.

Figure 1. Creating data for a self-defined flow or operation



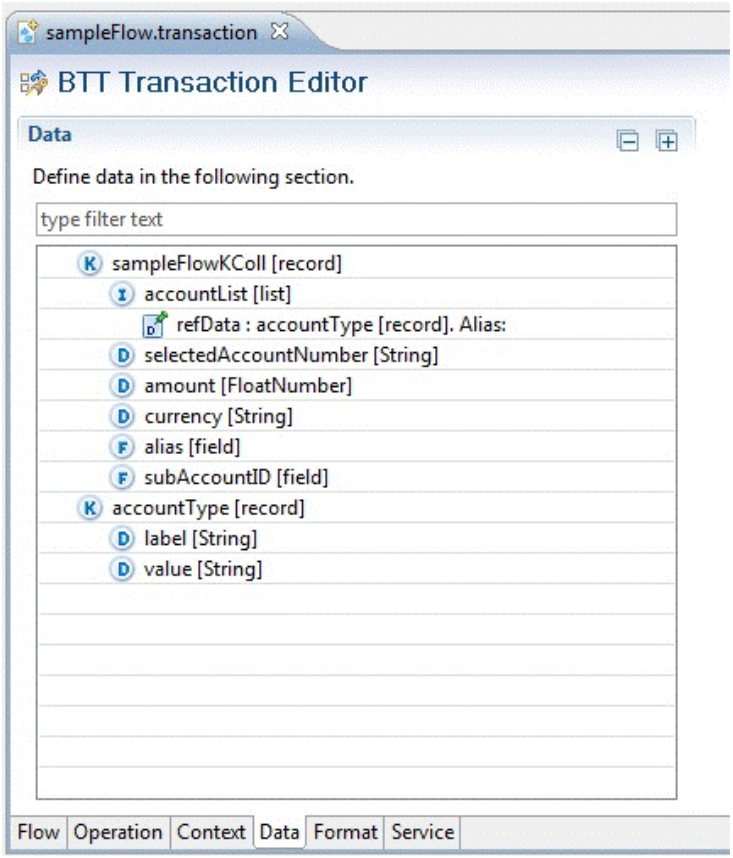
In the Detailed Information panel, enter the data element id and select the type of the data. For the amount, the type must be FloatNumber.

Figure 2. Selecting the type for a new data element



Use the same procedure to define all the flow related data:

Figure 3. Withdrawal flow context data

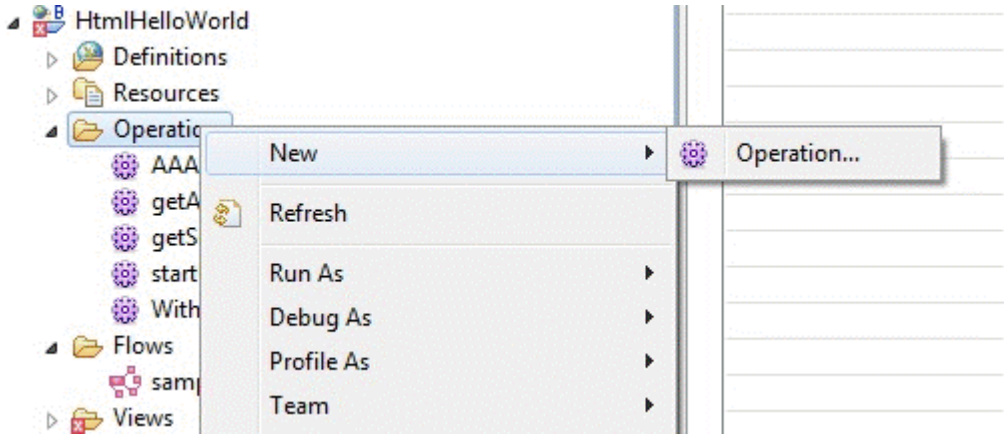


For the operations, the same procedure can be followed once the operation has been created. The next section describes, using a sample, how to create an operation that will be part of your transaction flow.

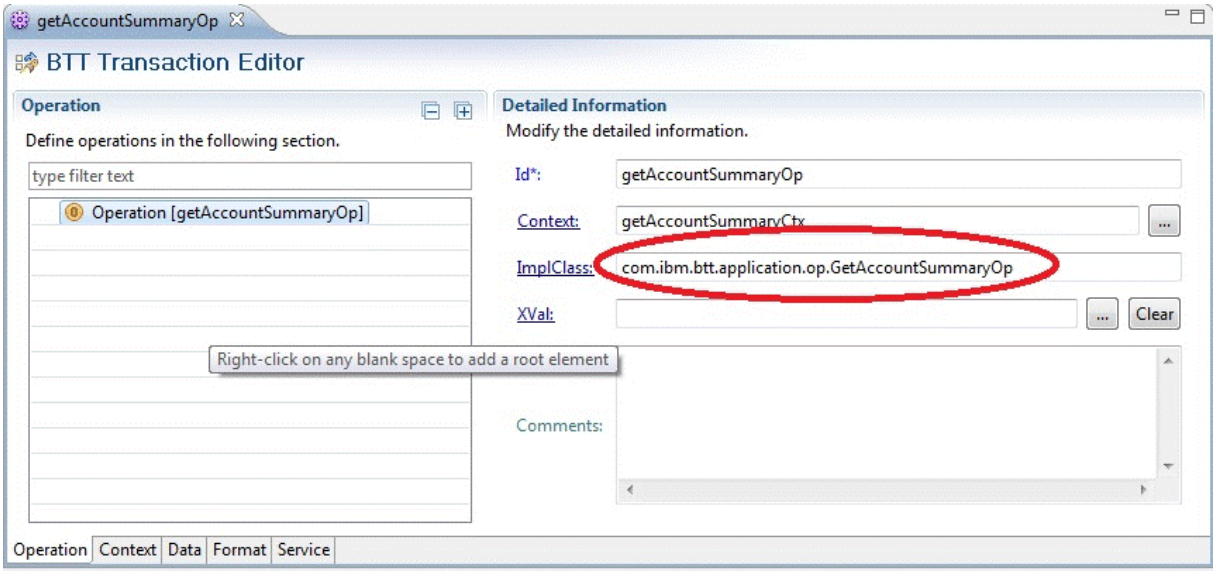
Creating transactions operations

The following steps show how to add the transaction operation getAccountSummaryOp, that simulates a backend call to retrieve the account details:

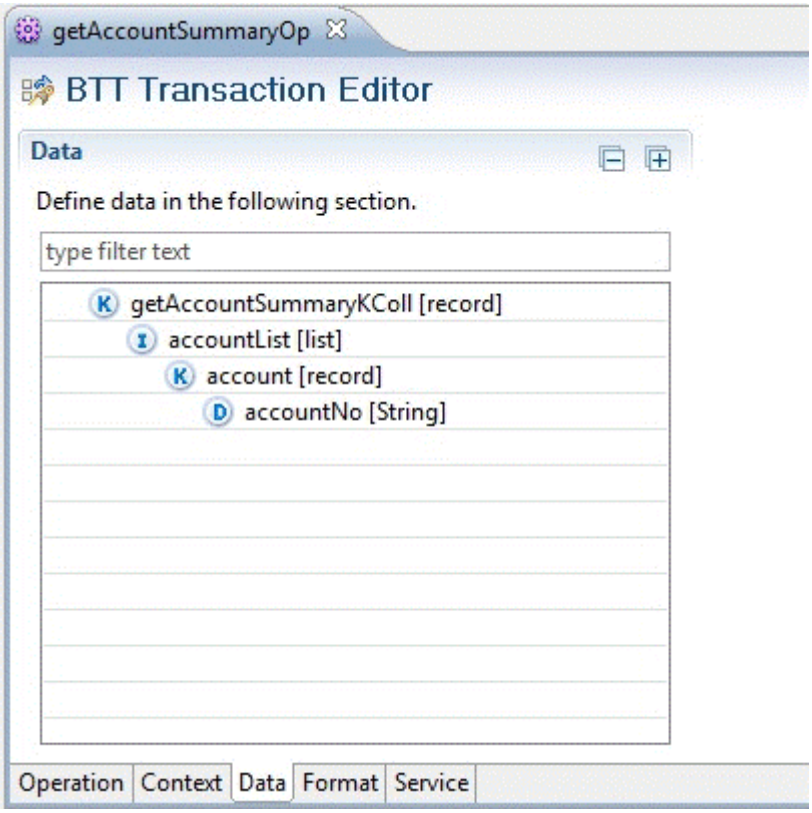
1. In the **BTT perspective** in Rational® Application Developer, right-click the Operations folder of the project, and then click New > Operation to create a new operation.



2. In the Operation tab of the Transaction editor, add a new operation and set a name for the operation implementation class. If the operation is going to connect to a backend that is a Web Service provider, then WebSphere® Multichannel Bank Transformation Toolkit also has tooling to generate the WebService calling operation automatically from WSDL, as seen before in this documentation. In the HTML Hello World sample, we have simulated the call by coding the data that should be obtained from backend calling. As the operation will be used in the flow, the implementation will have to generate an event that will be used to change from the operation state to another flow state during the flow execution process. For details on the implementation, refer to the imported code.



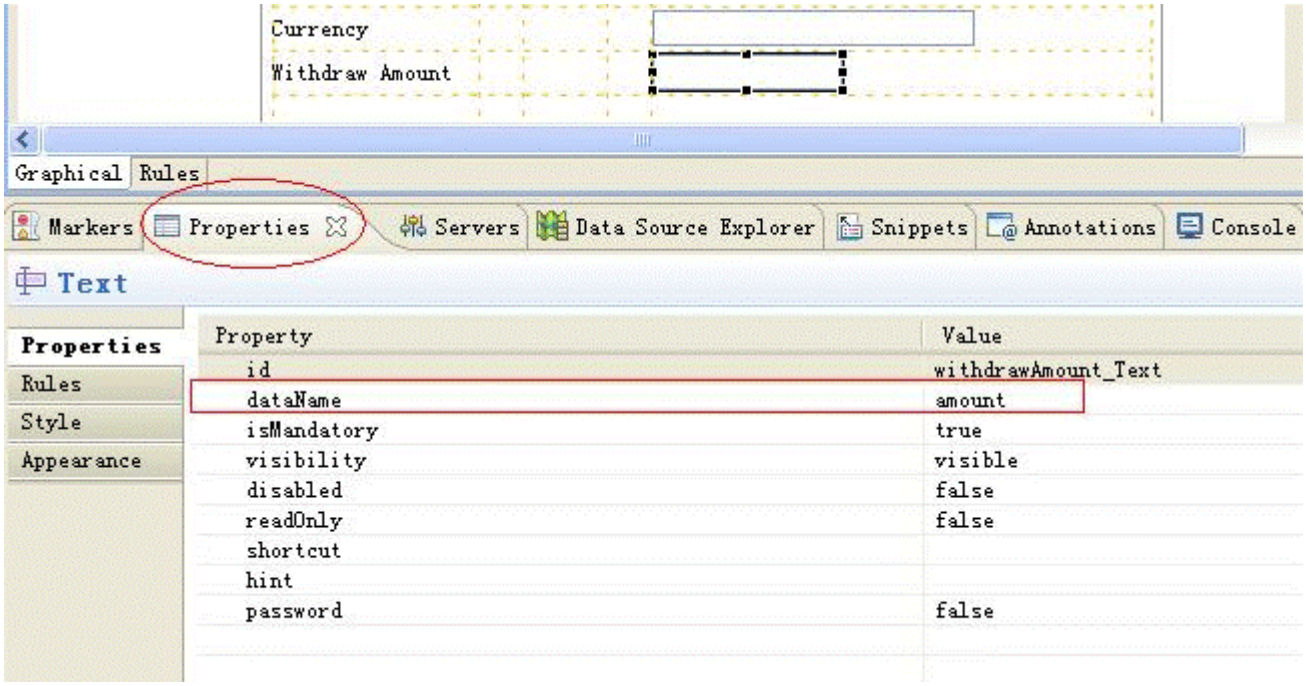
3. In the Data tab of the Transaction editor, add the data of the operation:



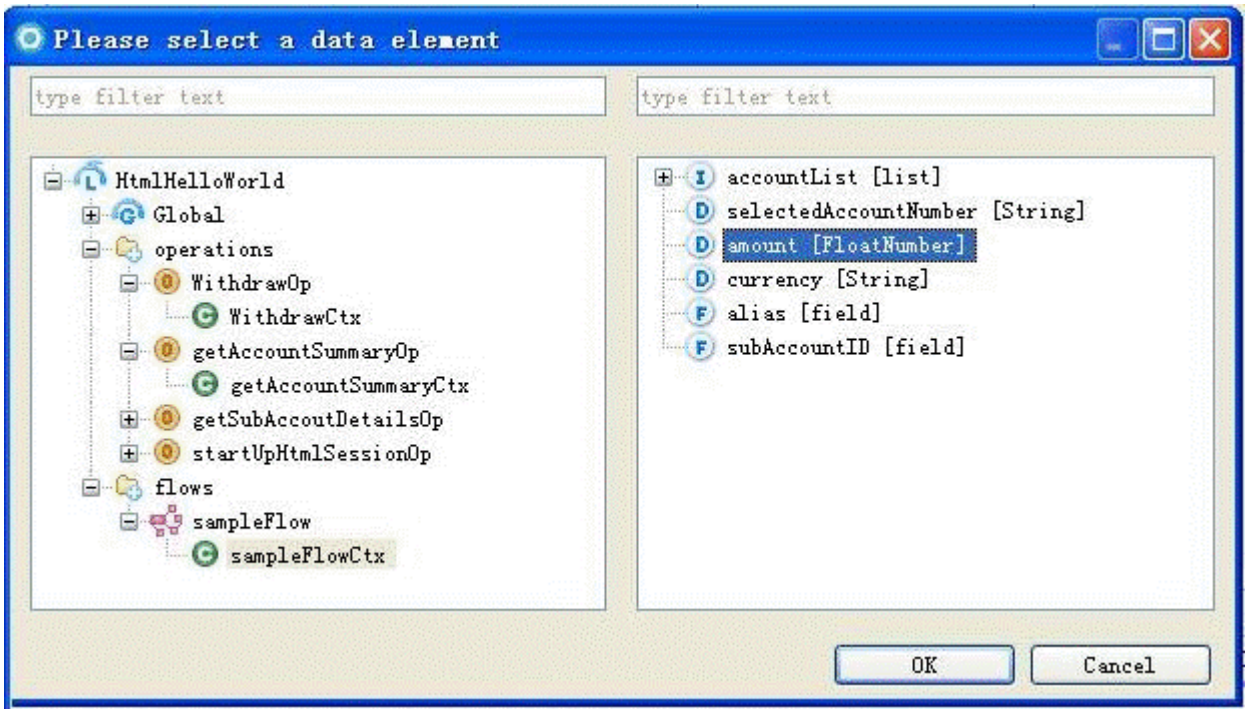
Creating a view

BTT tooling provides a XUI editor to development the UI visually, as described in the previous sections. For details on how to use the XUI editor to create a view, refer to section Modeling views with the XUI editor. In this section we will show some important definitions that should be done as part of the view construction.

An important thing to do once the view layout has been created is to bind the widgets to the data elements in the transaction context. The following screen shot is the example to set the data name property of text widget for amount.

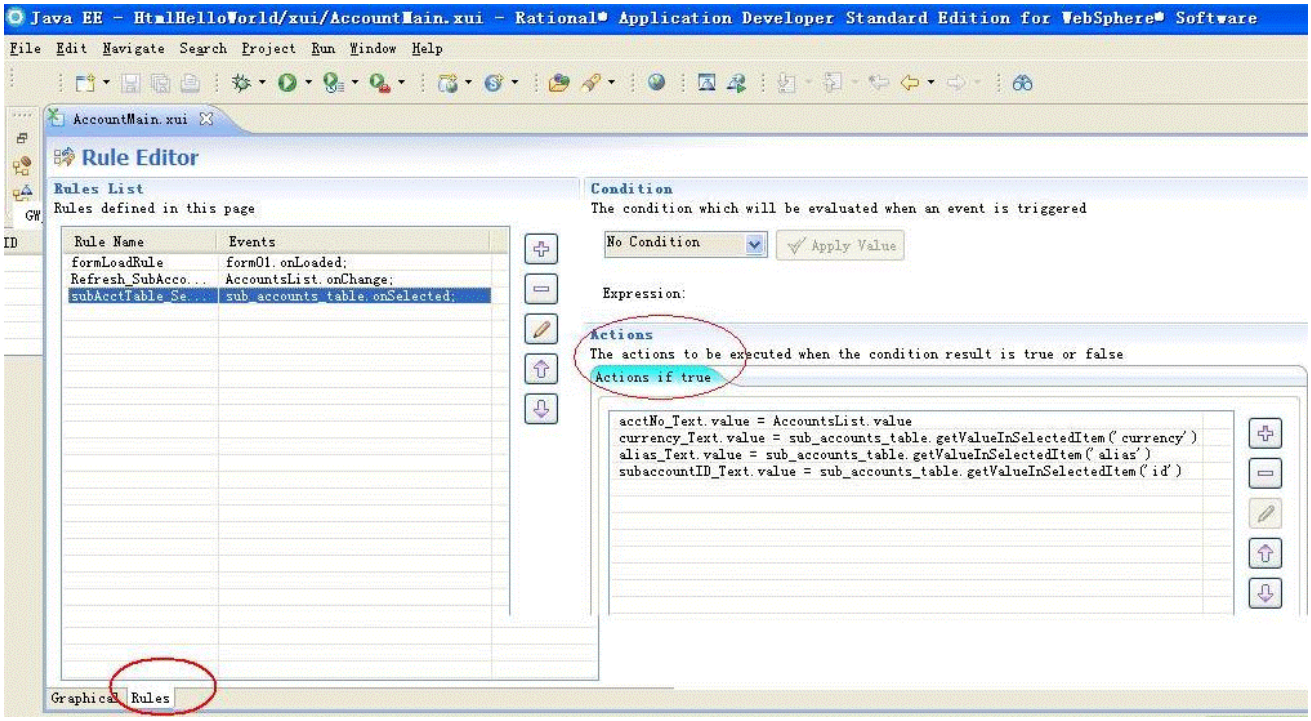


The data name can be selected from a dialog popup when clicking beside the dataName properties value.

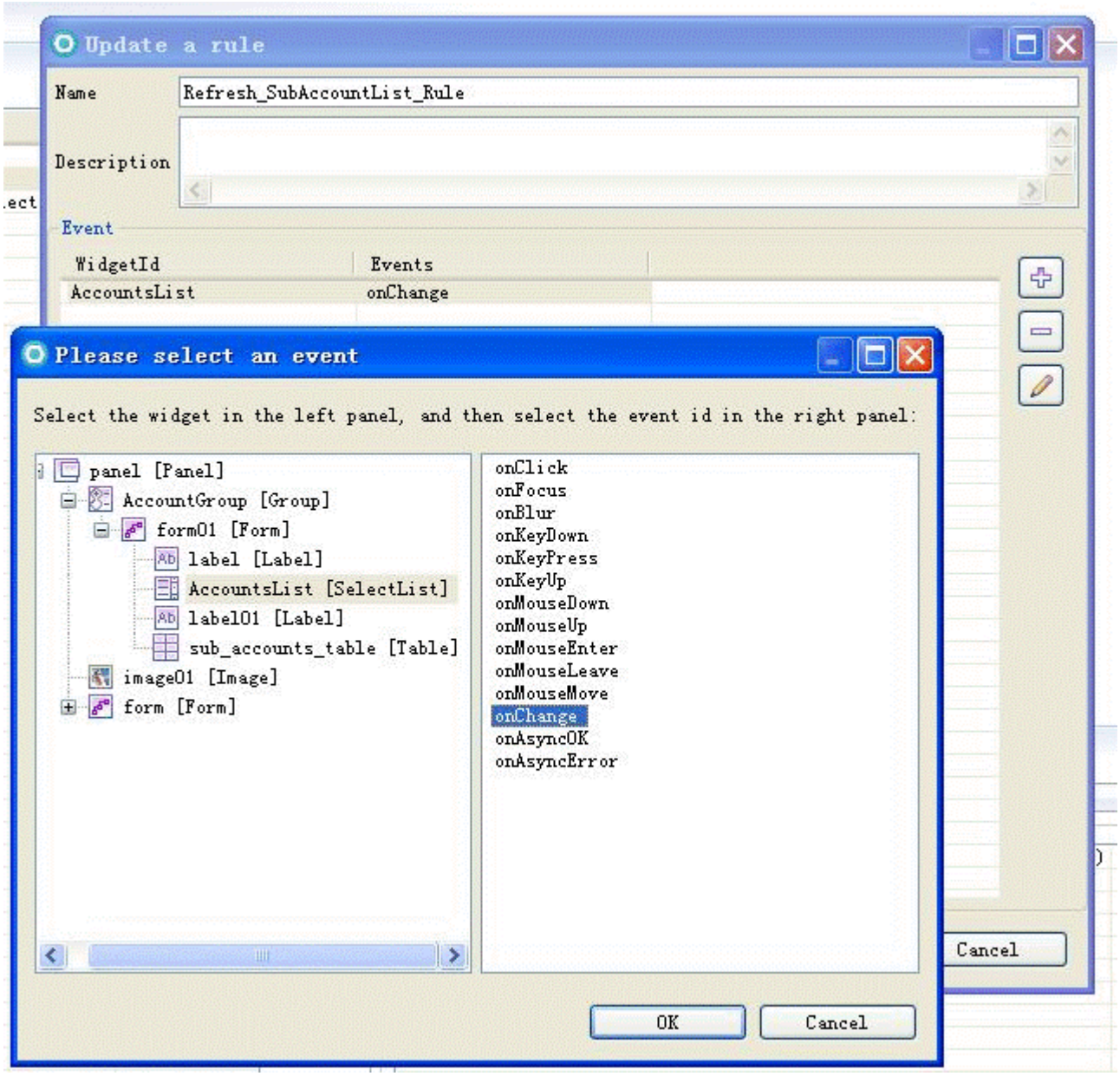


BTT can also supports the definition of some rules that allow triggering some actions on the view based on an event and on some conditions. These ECA rules (events, conditions and actions) cab be defined easily and visually using the tooling.

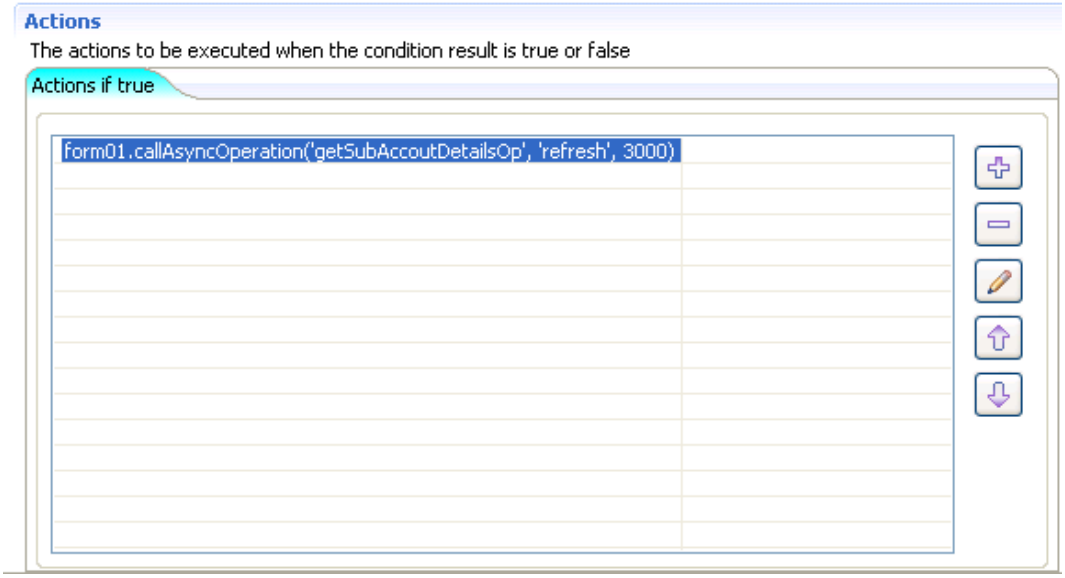
In the sample, we define three ECA rules for the AccountMain view. You can get the details of all the ECA rules defined for the view by moving from the Graphical view to the Rules views ("Rules" tab) of the XUI editor as shown in the Figure below.



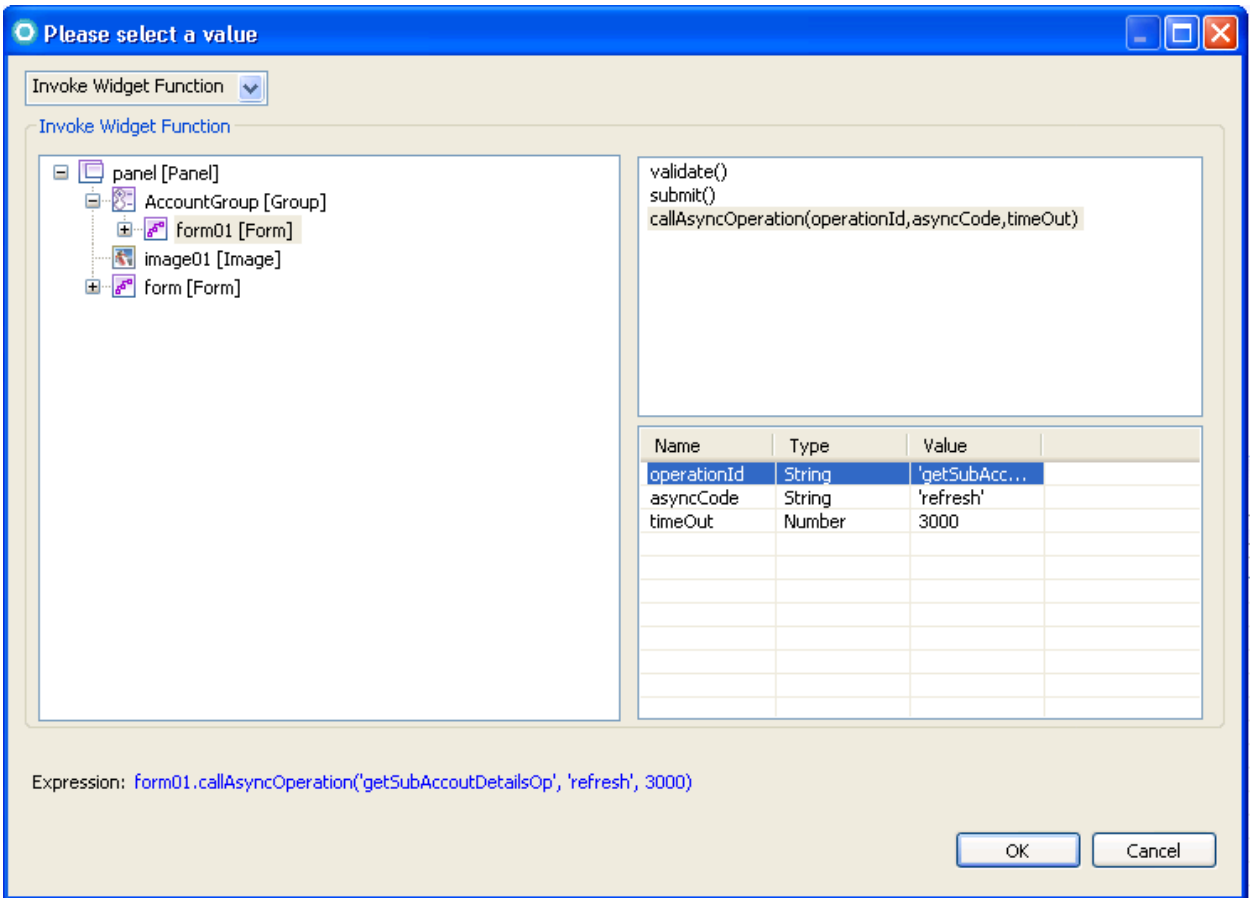
For instance, the **Refresh_SubAccountList_Rule** is the responsible of detecting that a sub account has been selected in the table and then refresh the new sub account in the Transaction group. If we open the rule definition, the event that launches the rule execution is a change in the AccountsList table. This is defined as shown in the Figure below:



Then we need to identify the condition and the resulting action. In this case, there is no condition. The action will be a call to an internal operation that will get the values from the selected sub account and will update the view fields with those values:



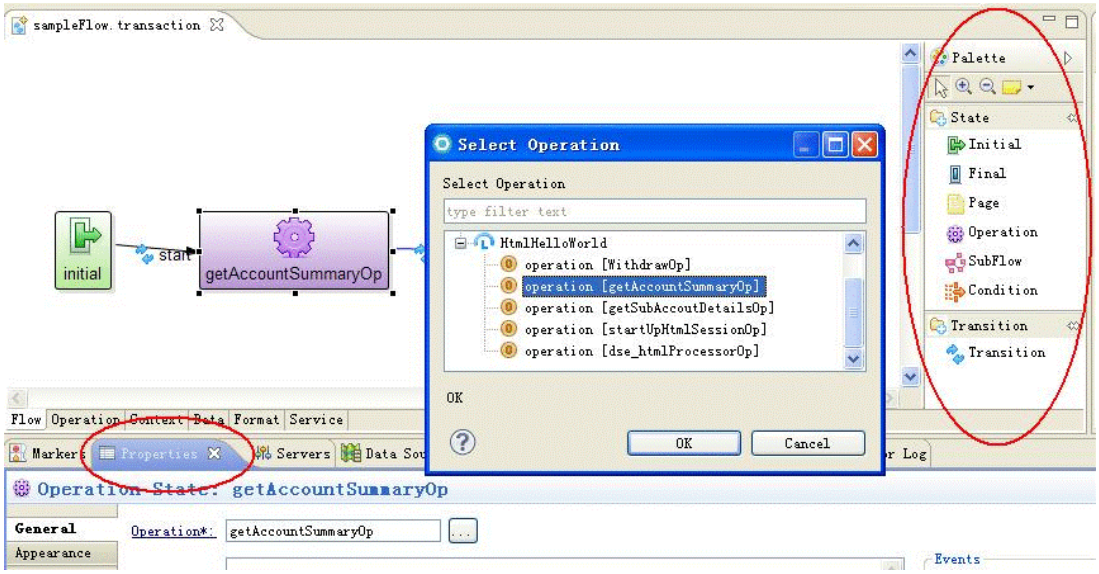
This action has been set by clicking on the  icon, selecting the corresponding Form widget function and setting the required parameters.



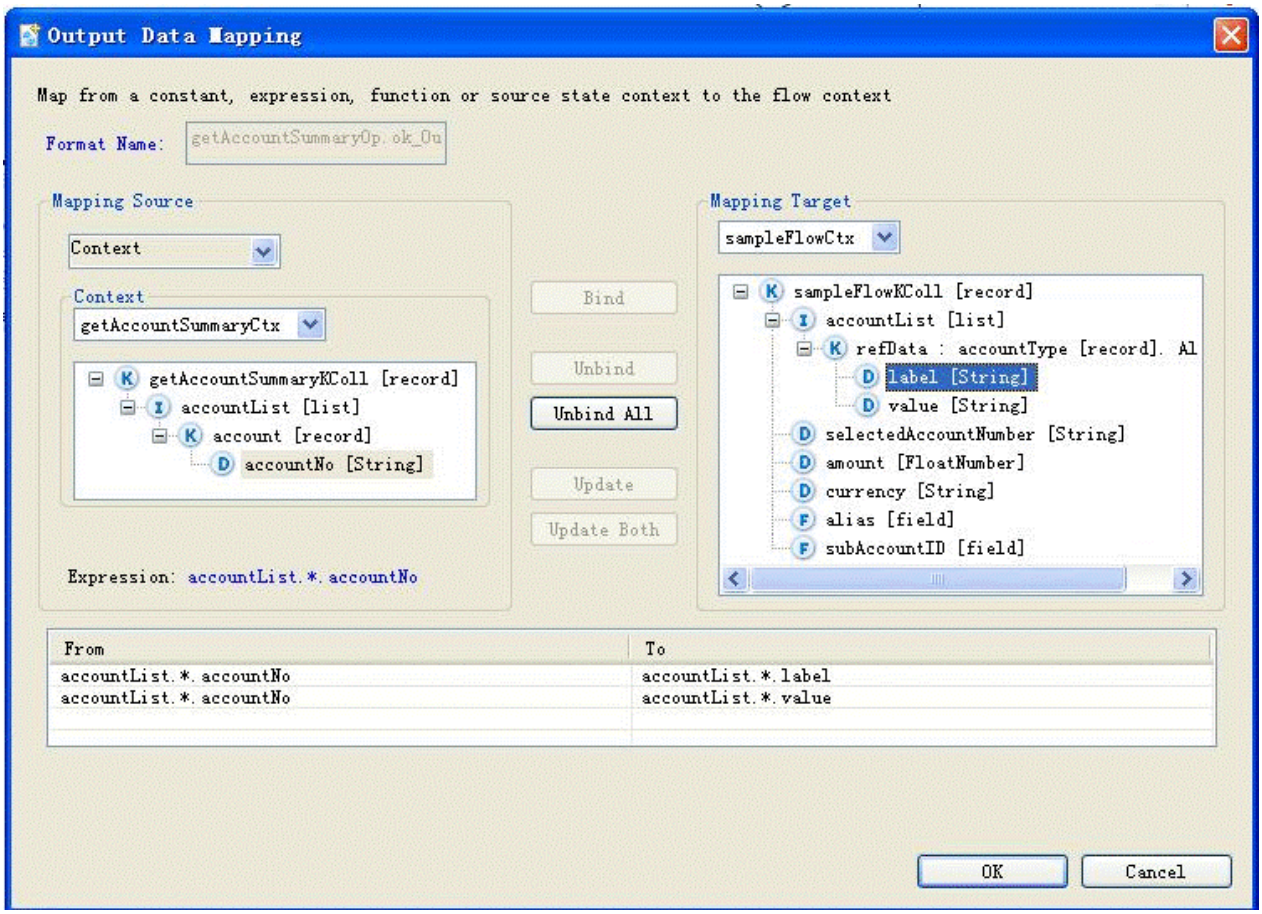
Assembling the transaction flow

After developer has created all the required context data, operations, views for a transaction, then they can be assembled into transaction flow. So it is a bottom-up process of development. In the sample, we use the sampleFlow which is created by default for BTT Project.

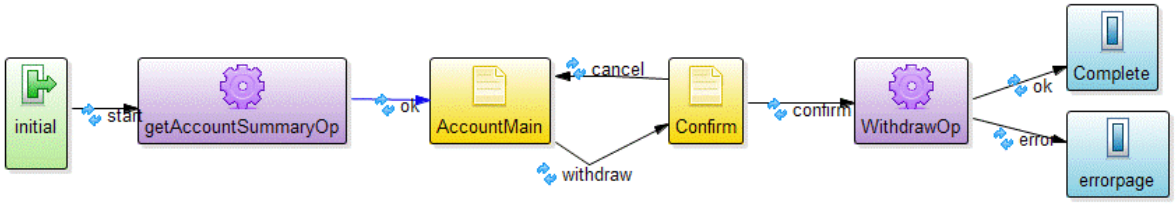
Open the flow editor. We can drop different flow states from palette. For example, we drop an operation state into editor, and select the operation that is going to be executed in this state as part of the state properties definition:



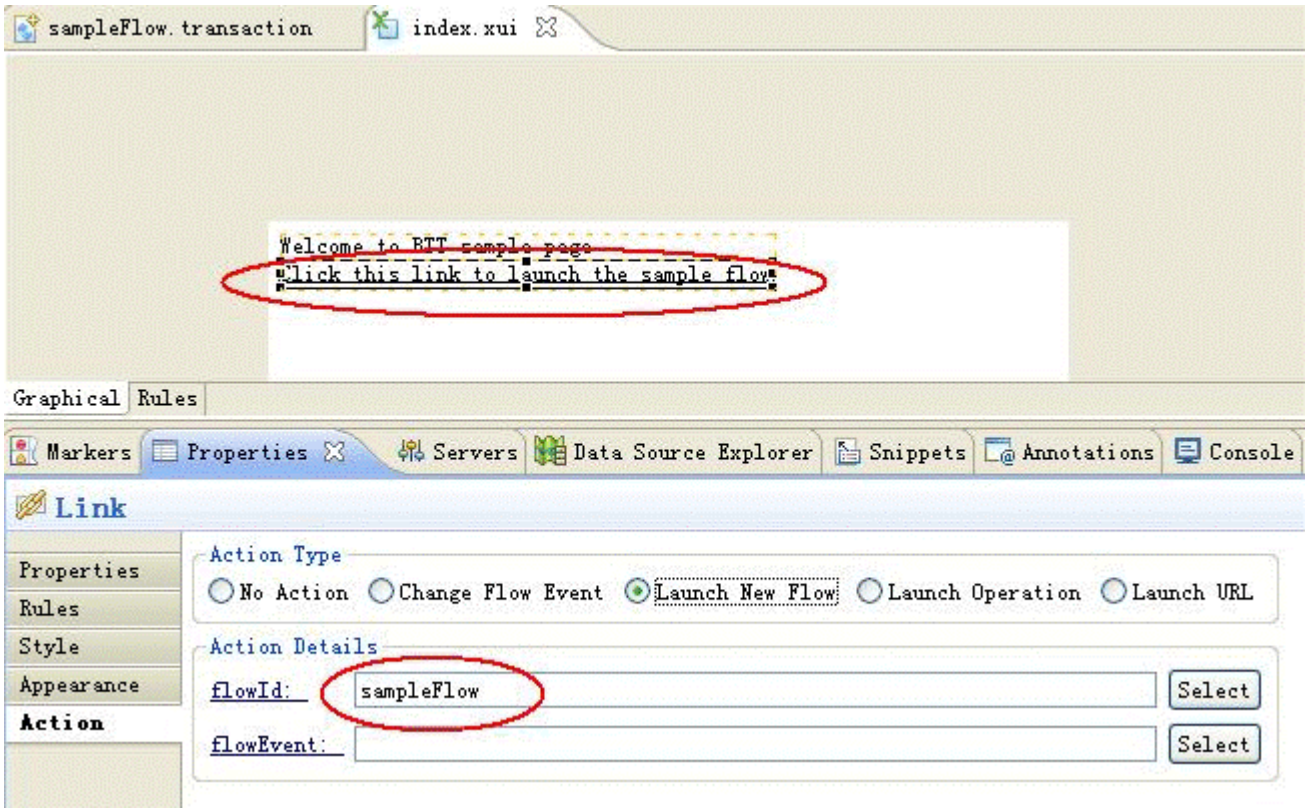
After we have added all the flow states, we need to line them up using transitions. One state will be jump to other state by the transition event. In the transition properties, we can also define the input/output map between the operation context and the flow context. The following is an example of data map definition to map the output result of getAccountSummaryOp to the flow context.



The resulting flow should be the one we had previously designed:



To be able to run the defined flow when we run the sample, we may want, for testing purposes, to configure the index view to add a new link to our flow. The index view is also added by default to your BTT Project when created and is associated to the initial state in the flow.



You can then test your sample following the indications in section Testing the flows.

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Lab Director
IBM China Software Development Lab

Diamond Building, ZhongGuanCun Software Park, Dongbeiwang West Road No.8, ShangDi, Haidian District, Beijing 100193 P. R. China

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.
The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java is a trademark of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.