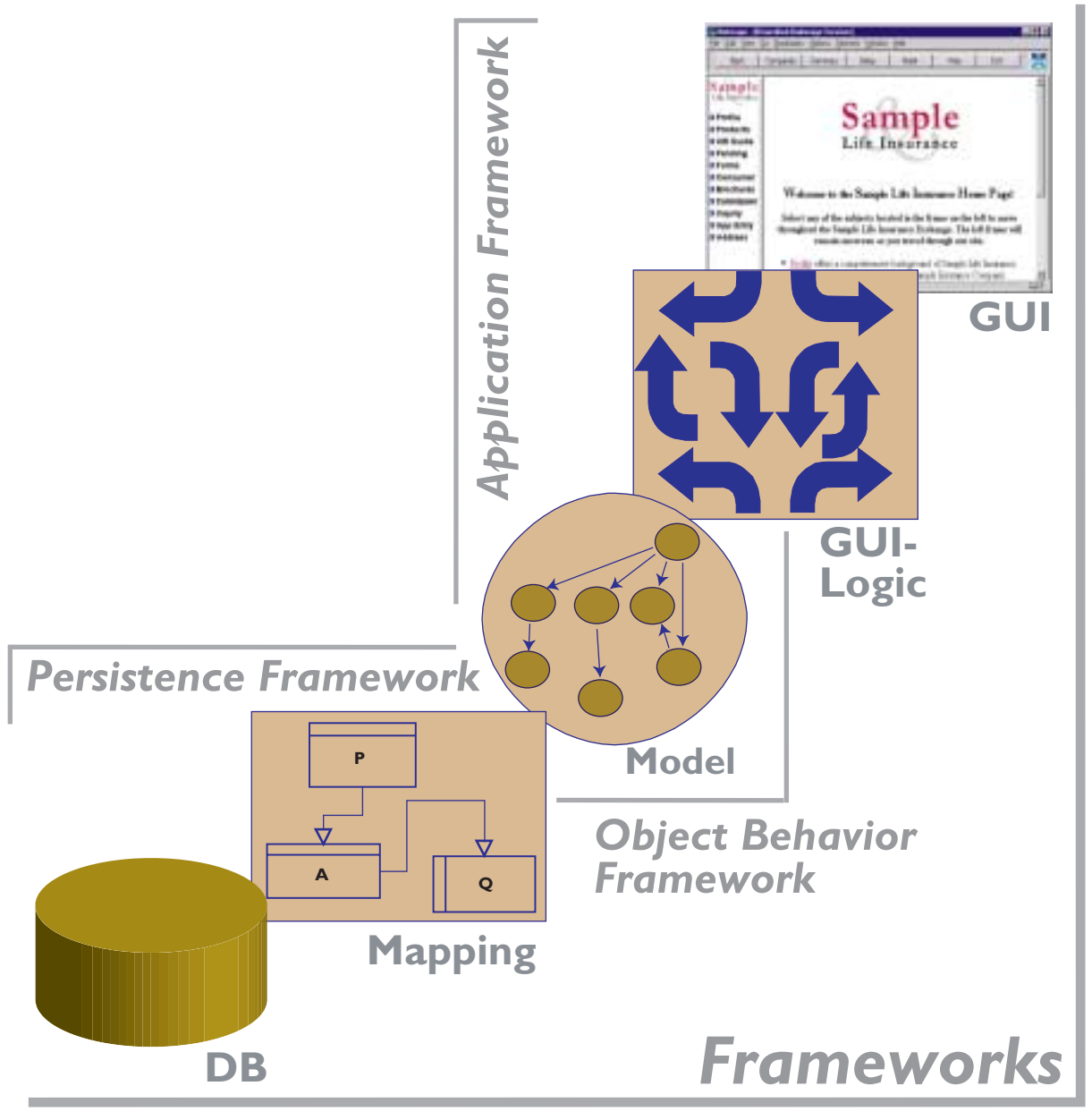
PMS MICADO

Frameworks *for Java*

Getting Started

For Frameworks Release 3.4 Version 1.1



Copyright and trademarks

Copyright

PMS MICADO Frameworks for Java RX.X VX.X

Frameworks for Java Getting Started Manual, May 1999

For more information about PMS Micado Frameworks, please contact:

PMS MICADO SoftwareConsult GmbH
Reutherstr. 1a-c
53773 Hennef / Germany
Tel. : (+49) (0) 22 42 - 871 - 400
FAX : (+49) (0) 22 42 - 871 - 455
Email: info.micado@notes.compuserve.com
Web: www.pmsmicado.com

Trademarks

Table of Contents

Documentation Overview

Congratulations

Congratulations on your purchase of PMS Micado Frameworks. Your purchase reflects your commitment to high-quality, cost-effective and rapid object-oriented software development. Extensive experience in software development for the banking and insurance industries has enabled PMS Micado to develop a suite of tools that can vastly shorten product development cycles. The documentation set from PMS Micado is designed to help you exploit the full potential of these tools in the shortest time possible.

Frameworks Getting Started (this manual)

Intended audience

The intended audience for this manual includes anyone who wishes to use PMS Micado Frameworks to raise the productivity and quality of software development with Visual Age for Java. This manual provides a very detailed step-by-step introduction that allows someone with even minimal Java, Visual Age or database experience to quickly master the basic capabilities of PMS Micado Frameworks.

Sections

This manual contains the following sections:

- **‘Copyright and trademarks’ (page 3)**.
- **‘Table of Contents’ (page 5)**.
- **‘Documentation Overview’ (page 9)**. This section.
- **‘Tutorial’ (page 17).** Provides a set of programming examples that demonstrate the major programming concepts and techniques of Frameworks.
- **‘List Of Figures’ (page 85)**.
- **‘Index’ (page 89)**.

Recommendations for completing the step-by-step tutorial

For those with minimal IBM Visual Age Java or IBM DB2 experience

IBM Visual Age for Java and DB2 must be installed on your computer. Then this manual will guide you step-by-step in installing and using PMS Micado Frameworks. After completing the tutorial you will have a good overview of the basic capabilities of Frameworks.

For those with IBM Visual Age Java and IBM DB2 experience, but with no Frameworks experience

You will be able to quickly complete the step-by-step tutorial. When introducing a Frameworks concept, the manual references the for more detailed information about Frameworks capabilities.

For those with some previous Frameworks experience

You might want to glance at ‘Tutorial overview’ (page 18), which provides an overview of the content of the tutorial. And then go directly to those sections that cover topics that are new to you. However, note that each tutorial chapter assumes that the previous chapter was completed. Therefore, not having completed a chapter, you may have to do some extra work before completing the next chapter.

For those with advanced Frameworks experience

This Getting Started manual is intended as an introduction to Frameworks. Therefore, the tutorial may not be for you. However, your comments or requests for the content of the tutorial are appreciated (comments can be sent to **info.micado@notes.compuserve.com**).

Conventions used in this manual

The following conventions are used in this manual.

- A term being used for the first time is bold and italic. For example: ***Object Network***.
- Dialog names and menu entries are displayed in bold, with a forward slash between nested entries. For example: **System Transcript / Tools / Manage Applications**.
- Java code is fixed-width Courier. For example:

```
^self
```

Reader comments

Any comments you have concerning this manual can be sent to:

info.micado@notes.compuserve.com

Other manuals

From PMS Micado (and referenced throughout this manual):

- .

From IBM:

- IBM Visual Age for Java manuals.

Training from PMS Micado

This manual is provides complete information to help you start using Frameworks as soon as possible. However, it is also recommended to consider enrolling in a training course from PMS Micado that is customized to your specific needs. PMS Micado provides a wide-range of courses covering Java programming, object-oriented analysis, design, methodolgy, and project management.

For more information about training courses, please contact:

PMS MICADO SoftwareConsult GmbH
Reutherstr. 1a-c
53773 Hennef / Germany
Tel. : (+49) (0) 22 42 - 871 - 400
FAX : (+49) (0) 22 42 - 871 - 455
Email: info.micado@notes.compuserve.com
Web: www.pmsmicado.com

Installing Frameworks

System requirements

This tutorial assumes that you have the following:

- **Windows NT**.
- IBM **Visual for Java Version 2.0** with **2.0 Rollup 2** installed.
- PMS Micado **Frameworks for Java R3.4 V1.1**.

Installation Overview (via the server)

Installing Frameworks consists of the following main steps:

- Binding to the repository.
- Adding projects from the repository.
- Copying the tools to your IBM Java Tools directory.
- Setting NLS path in OME.ini and DPB.ini.
- Configure browsers (OME and OMB).

Binding to the repository

If you binded to the respository on ntsrv1 during installation: Skip this section.

1. Start **Visual Age for Java**.
2. From the **Workbench**: Select **File / Quick Start**.
3. Select **Repository Management**.
4. Double-click on **Change Repository**.

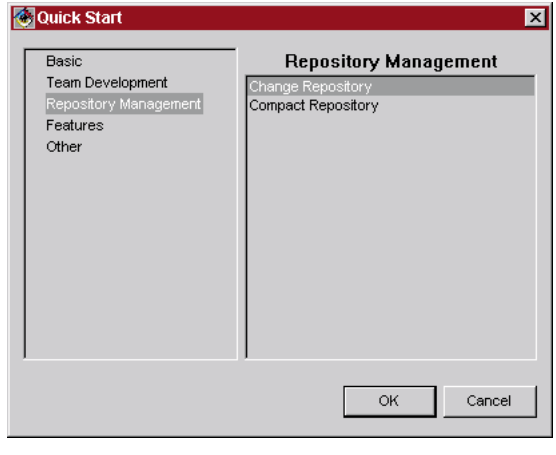


Figure 1: Change repository dialog

5. Select **Admin / Change repository**.
6. Select **Use a shared repository with EMSRV server address:**.
7. Enter **ntsrv1** as the shared repository.
8. In **Repository name:**: Enter **e:\javaentw\ivj.dat**.

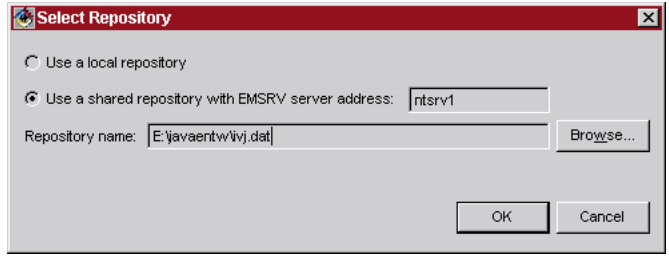


Figure 2: Respository specification

9. Click **OK**.

10. Select the **workspace owner**.
11. Click **OK**. The binding is complete.

Adding projects from the repository

The projects for Frameworks must now be added. The projects should be added in the order described. Otherwise errors can arise.

Add JDK 11 Collections

12. From the **Workbench**: Select **Projects / Add / Projects...**. The **SmartGuide Add Project** appears.
13. Select **Add projects from the repository**.
14. Check the checkbox for **JDK 11 Collections**.

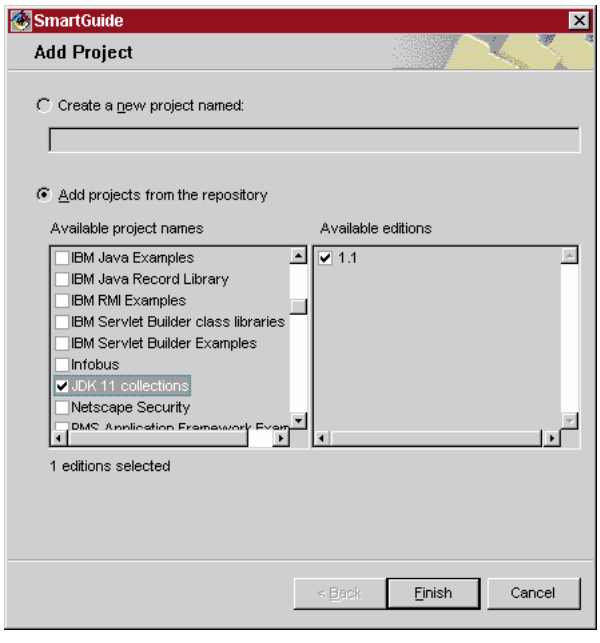


Figure 3: Add project dialog

15. Click **Finish**.

Add IBM IDE Utility class libraries

16. Add **IBM IDE Utility class libraries**.

Add PMS projects

17. Add all projects that begin with **PMS**. NOTE: Ignore the error messages that are generated. These are generated due to the Persistence Framework.

The added projects are now displayed in the Workbench:

Figure 4: Added projects

Copying the tools to your IBM Java Tools directory

The files (class, etc.) must now be copied to your IBM Java tools directory.

18. Copy the directory **\\Ntsrv1\EnvyEntw\MICOfw\Tools\Runtime\Java\PMS-MICADO** to your **IBM Java Tools directory** (typically **C:\IBMJava\ide\tools**).

Set NLS path in OME.ini and DPB.ini

Set OME.ini NLS path

19. Open the file **\IBMVJava\ide\TOOLS\PMS-MICADO\Browser\OME.ini**.

20. Change the nlspath specification to the following (on my computer VAJava was installed on disk D; change this if installed on a different hard disk on your computer):

```
nlspath=D:\IBMVJava\ide\TOOLS\PMS-MICADO\Browser\NLS;
```

21. Close and save the file.

Set DPB.ini NLS path

22. Open the file **\IBMVJava\ide\TOOLS\PMS-MICADO\Browser\DPB.ini**.

23. Change the nlspath specification to the following (on my computer VAJava was installed on disk D; change this if installed on a different hard disk on your computer):

```
nlspath=D:\IBMVJava\ide\TOOLS\PMS-MICADO\Browser\NLS;
```

24. Close and save the file.

Configure browsers

Set OME

The OME must be configured.

25. In the **Workbench**: Select **Workspace / Tools / PMS MICADO / Open ModelBrowser**. The Model Browser - VA Client dialog is opened.

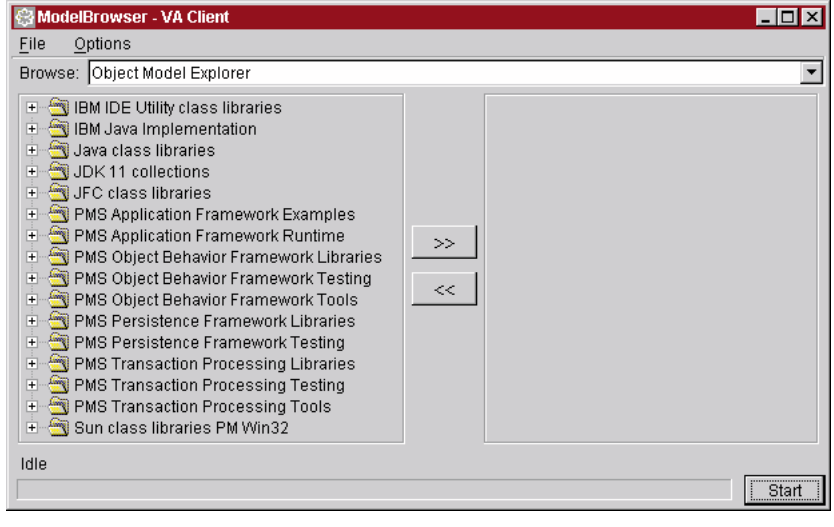


Figure 5: Model Browser - VA Client dialog

26. In the **Model Browser - VA Client**: Select **Options / Set Browser**. The **Options** dialog opens:

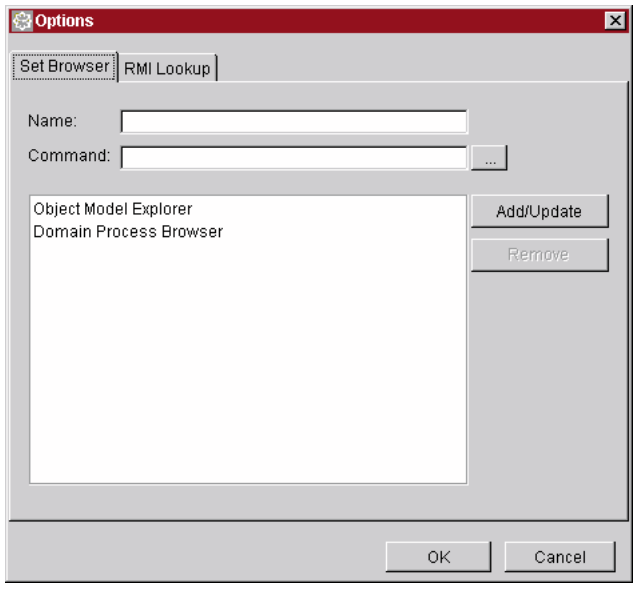


Figure 6: Model Browser options dialog

27. Click on **Object Model Explorer**.
28. Click on the **directory button** for **Command**. The **open File...** dialog appears:

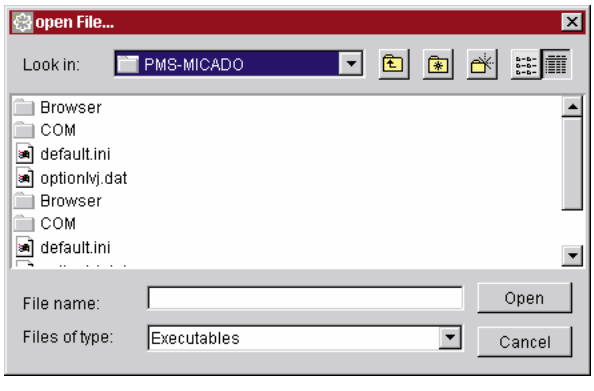


Figure 7: Open file dialog

29. Double-click on directory **Browser**.
30. Double-click on **OME.exe**. The Name and Command information for the OME appear in the Options dialog:

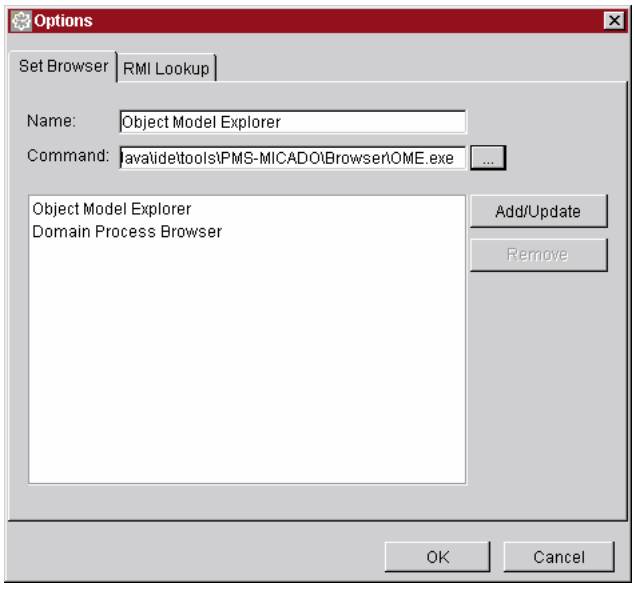


Figure 8: Name and Command information for the OME

Set OMB

31. Set the OMB to **DPB.exe** using the same method as above.
32. Click **OK**. The **Options** dialog closes.

Configure OME

33. In the **ModelBrowser - VA Client** dialog: Select from the **Browse:** drop-down list **Object Model Explorer**.
34. Click **Start**. The **Object Model Explorer - Exploring: none** dialog appears.

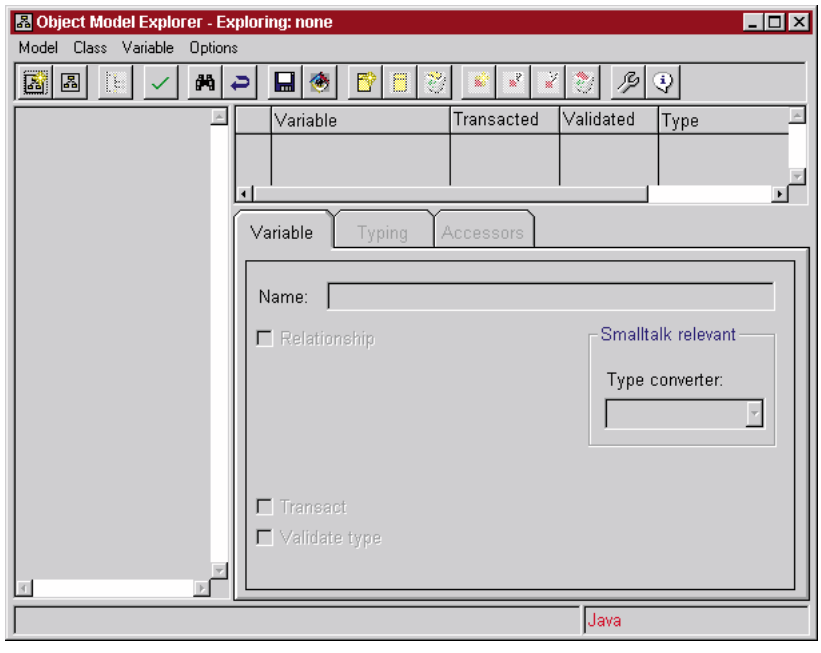


Figure 9: OME dialog

35. Select **Options / Preferences**. The **Options** dialog appears:

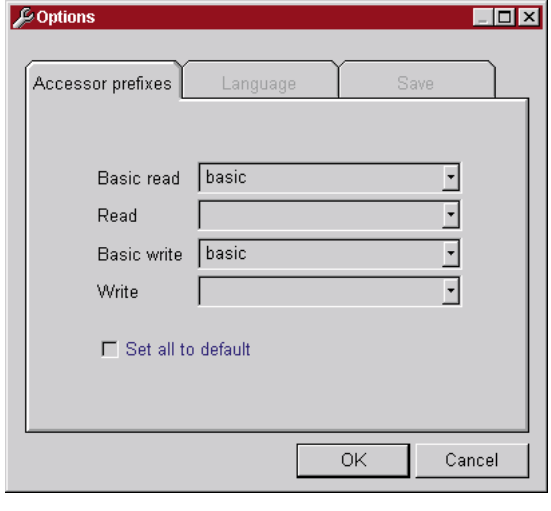


Figure 10: OME Preferences dialog

36. In the **Accessors Prefixes** tab: Make the following changes:
    36.1 Set **Basic read** to **basicGet**.
    36.2 Set **Read** to **get**.
    36.3 Set **Basic write** to **basicSet**.

36.4 Set **Write** to **set**.

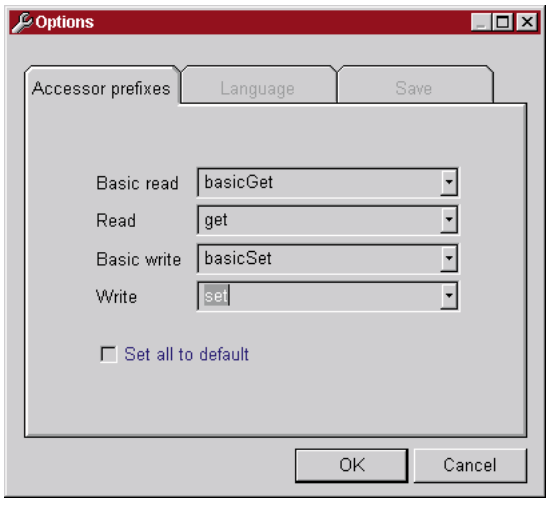


Figure 11: Accessor prefix settings

37. Select **OK**.

Tutorial

1. Tutorial overview

"Example is the school of mankind, and they will learn at no other." Edmund Burke.

This section presents a step-by-step tutorial that walks you through examples that demonstrate the major concepts in Frameworks. As such, the organization of the tutorial is dictated by what you need to do to get your job done as quickly as possible.

The following is an outline of the tutorial. It is highly recommended to work through each step in the tutorial.

Create Project, ZyxMember Domain Object, Domain Process, View

- 'Create ZyxTutorial project' (page 19). Explains how to create a project.
- 'Create ZyxMember (Domain Object)' (page 20). Shows how to create a simple domain object. Domain objects are objects such as Club, Member, Address, etc.
- 'Create ZyxEditMember (Domain Process)' (page 24).
- 'Create ZyxEditMemberView (View)' (page 27).

Transacting changes

- 'Multiple non-transacted views' (page 31).
- 'Multiple views: transacted non-isolated' (page 33).
- 'Multiple views (single object): transacted isolated' (page 37).
- 'Aborting/committing transacted changes with the TB' (page 38).
- 'Aborting/committing transacted changes with abort/commit buttons' (page 39).
- 'Aborting/committing transaction contexts with abort/commit buttons' (page 41).

Validating input type and range

- 'Validate type and range of input' (page 42).

Create ZyxClub Domain Object, Domain Process, View

- 'Create DO ZyxClub' (page 45).
- 'Create DP ZyxEditClub' (page 47).
- 'Create View ZyxEditClubView' (page 50).

Transactions with parent/child processes

- 'Transacted changes in child process ZyxEditMember' (page 53).
- 'Display transacted changes in child process in parent process view' (page 54).

Adding and deleting ZyxMember's

- 'Adding and deleting members' (page 55).

Viewports

- 'Implementing Tooltips (using a ViewPort)' (page 56).
- 'Enabling/disabling buttons' (page 57).
- 'Add JList to ZyxEditClubView' (page 59).

Group controls

- 'Controlling visibility of a GroupControl (with a ViewPort)' (page 61).

2. Create ZyxTutorial project

When you create a class, the class must be assigned to an application. All of the classes created during this tutorial will be assigned to your tutorial application.

Create project ZyxTutorial.

2.1 In **Workspace**: Select **Project / Add / Project...**. The Smart Guide **Add Project** appears.

2.2. Select **Create a new project named:**.

2.3. Enter **ZyxTutorial** as the project name.

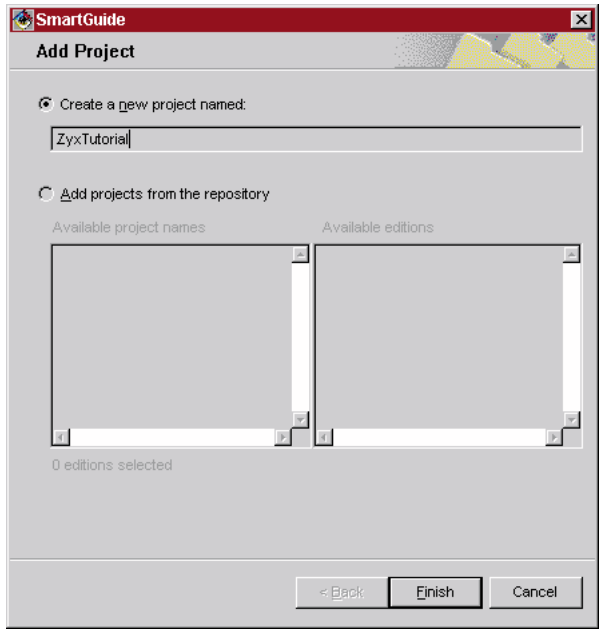


Figure 12: Add Project Smart Guide

NOTE: If the repository is being shared for this tutorial, then "Zyx" must be changed to a unique prefix that is not being used by anyone else. If this is the case, then use the unique prefix throughout the rest of this tutorial.

2.4. Make sure that the checkbox **Subapplication of** is not checked.

2.5. Click **Finish**. The project ZyxTutorial is selected in the list of projects.

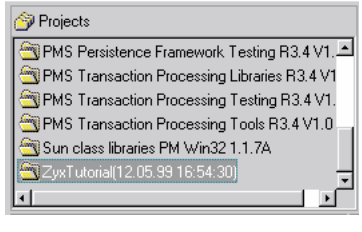


Figure 13: ZyxTutorial in the project list

3. Create ZyxMember (Domain Object)

ZyxMember will be the first ***Domain Object*** (DO) that you create in this tutorial. ZyxMember attributes will reference objects such as a member name, weight, address, etc. For more information about DO's, consult the ***Application Framework User's Guide*** chapter .

Start OME

3.1 In **Workbench**: Select **Workspace / Tools / PMS MICADO Frameworks / Open ModelBrowser**. The **Model Browser - VA Client** dialog appears.

3.2. In the **Browse:** drop-down list: Select **Object Model Explorer**.

3.3. Click **Start**. The OME dialog appears.

Create DomainObject subclass ZyxMember

3.4. Right-click in the left part of the window.

3.5. Select **New Class**. The **Class specification** dialog appears.

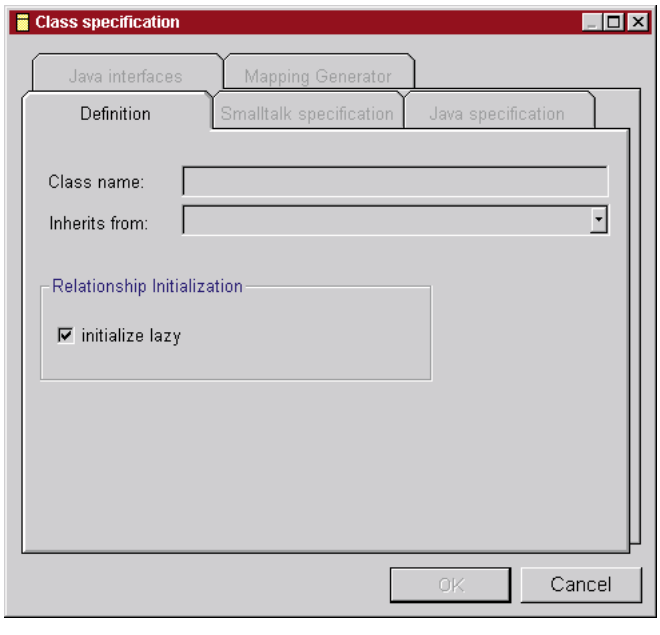


Figure 14: Class specification dialog

3.6. In the **Class Name:** field: Enter **ZyxMember**.

3.7. From the **Inherits from:** drop-down list: Select **Domain Object**.

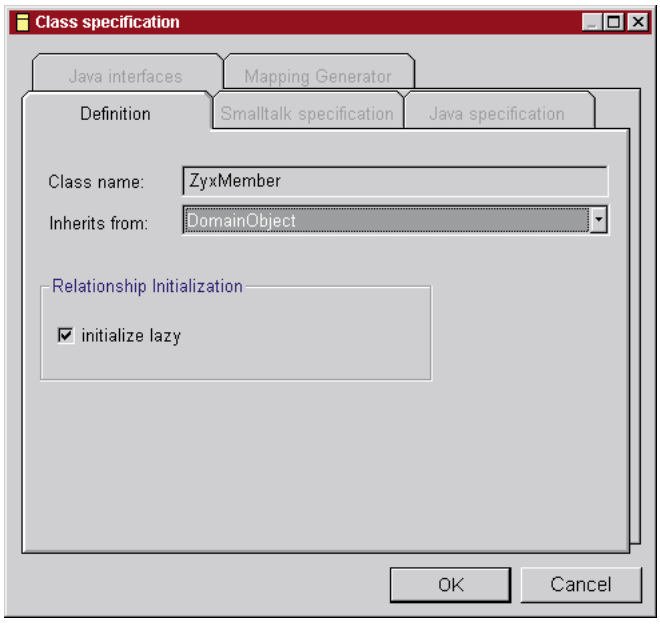


Figure 15: Class specification for ZyxMember

3.8. Select the tab **Java specification**.

3.9. In the field **Package name:**: Enter **zyx.tutorial**.

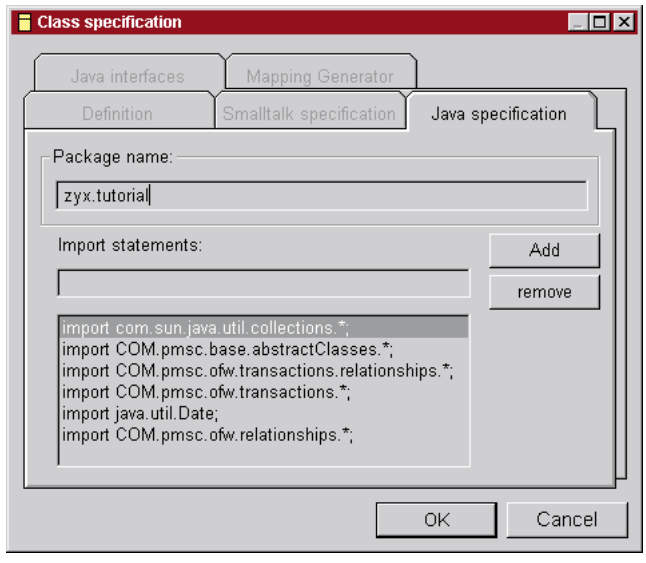


Figure 16: Java specification for ZyxMember

3.10. Click **OK**.

Create ZyxMember>>name

3.11. In the **OME**: Right-click in the **upper-right** window.

3.12. Select **Add variable**. The ?? dialog appears.

3.13. In the field **Enter the name(s) for new instance variable(s)!**: Enter **name**.

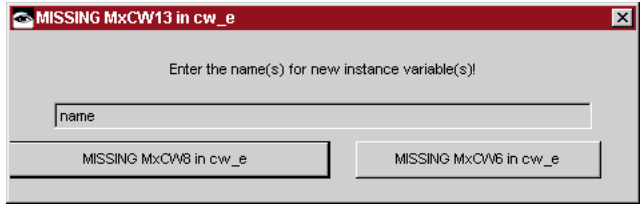


Figure 17: OME Add New Variable dialog

3.14. Click **OK**. The variable name appears in the OME.

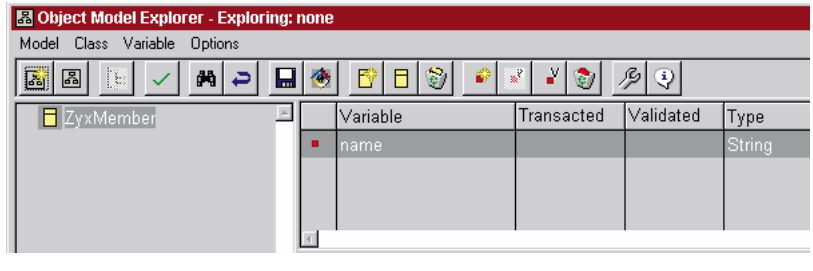


Figure 18: Variable <name> in OME

Save model ZyxMember to VA

3.15. In the **OME**: Select **Model / Save to VA**. The **Choose Project for zyx.tutorial** dialog appears.
NOTE: If the dialog does not appear: The dialog is hidden behind the OME or the Model Browser - VA Client dialogs. Move the other dialogs until the Choose Project for zyx.tutorial dialog is visible.

3.16. In the list **Names**: Select **ZyxTutorial**.

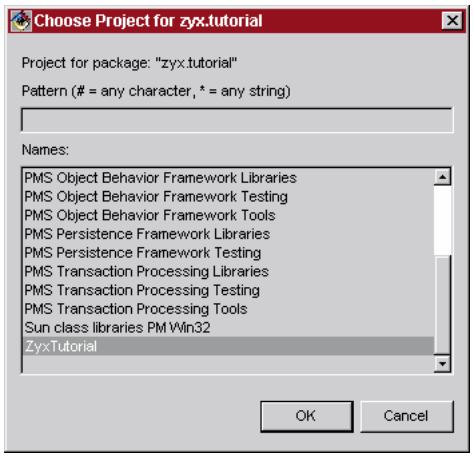


Figure 19: Dialog <Choose Project for zyx.tutorial>

3.17. Select **OK**. The class has been saved to VA.

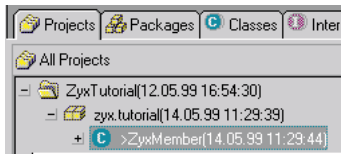


Figure 20: Class ZyxMember in the Workbench projects list

3.18. Close **OME**.

3.19. Save the workspace (from the **Workbench**: **File / Save Workspace**).

Redisplaying ZyxMember in OME

Whenever a class is created in OME, the class must also be added to the Model Browser right window in order to display the class in future OME dialogs.

Attempt to find ZyxMember in OME

3.20. In the **Model Browser**: In the **Browse:** drop-down list: Select **Object Model Explorer**.

3.21. Click **Start**. The OME dialog appears.

3.22. Select **Class / Find class**.

3.23. In response to **Enter the name of the class to find**: Enter **ZyxMember**.

3.24. Click **OK**. The message **No match found** appears.

3.25. Click **OK**.

3.26. Close **OME**.

Add ZyxMember to Model Browser class list

3.27. In the **Model Browser**: Select **File Reload**.

3.28. Expand the tree for **ZyxTutorial**.

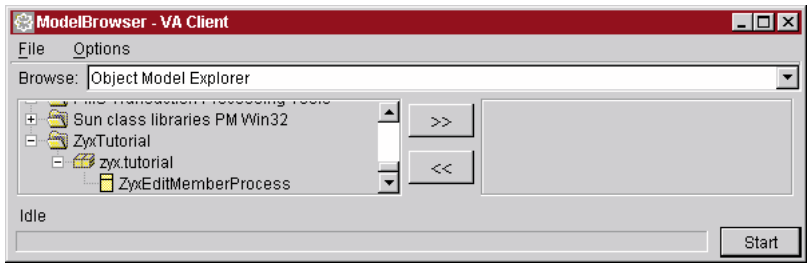


Figure 21: Model Browser dialog

3.29. Select **ZyxMember**.

3.30. Click on **>>**.

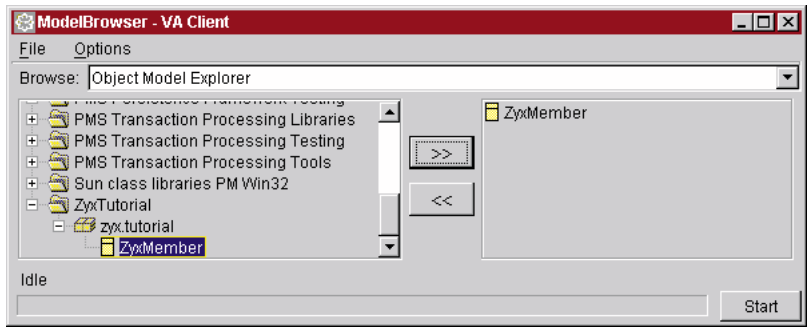


Figure 22: ZyxMember in the Model Browser class list

Open OME

3.31. In the **Model Browser**: Click **Start**. The **OME** dialog appears with **ZyxMember** in the class list.

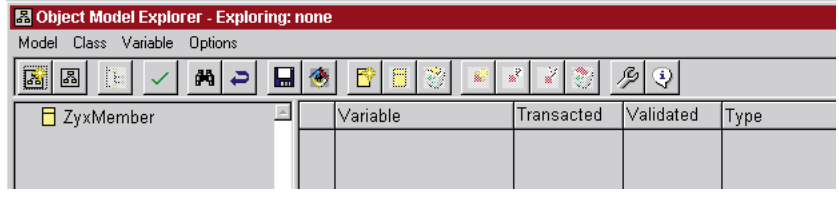


Figure 23: OME with ZyxMember

4. Create ZyxEditMember (Domain Process)

ZyxEditMember will be the first ***Domain Process*** (DP) that you create in this tutorial. ZyxEditMember methods implement the functionality required to edit a member's attributes.

For more information about Domain Process, consult the chapter

Create DomainProcess subclass ZyxMember

4.1 In the **OME**: Right-click in the left part of the window.

4.2. Select **New Class**. The **Class specification** dialog appears.

4.3. In the **Class Name:** field: Enter **ZyxEditMember**.

4.4. From the **Inherits from:** drop-down list: Select **Domain Process**.

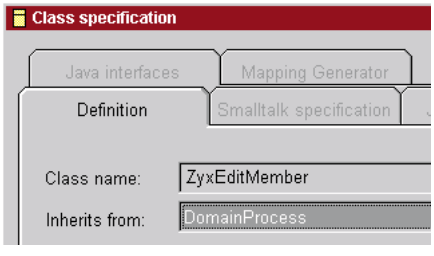


Figure 24: Class specification for ZyxEditMember

4.5. Select the tab **Java specification**. Note that the package name is already set.

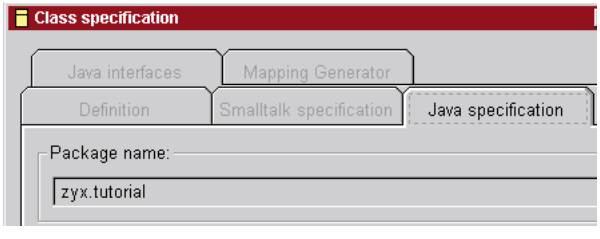


Figure 25: Java specification for ZyxEditMember

4.6. Click **OK**. **ZyxEditMember** appears in the **OME**.

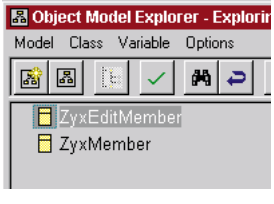


Figure 26: ZyxEditMember in the OME

4.7. In the **OME**: Select **Model / Save to VA**. Note that the **Choose Project for zyx.tutorial** dialog does not appear.

Add ZyxEditMember to Model Browser class list

4.8. In the **Model Browser**: Select **File Reload**.

4.9. Expand the tree for **ZyxTutorial**.

4.10. Select **ZyxEditMember**.

4.11. Click on **>>**.

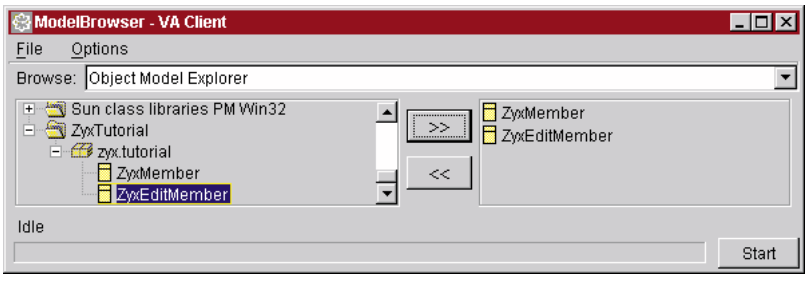


Figure 27: ZyxEditMember in the Model Browser class list

Open Domain Process Browser (DPB) on ZyxEditMember.

The ***Domain Processes Browser*** (DPB) is a Visual tool for setting a variety of parameters for the DP. For more information about DP, consult the .

4.12. In the **Model Browser**: From the **Browse** drop-down list: Select **Domain Processes Browser**.

4.13. Click **Start**.

4.14. In response to **Select a root class**: Click **OK**. The **Selection Required** dialog appears.

4.15. In response to "Choose a class": Double-click on **zyx.tutorial.ZyxEditMember**. The Domain Processes Browser opens:

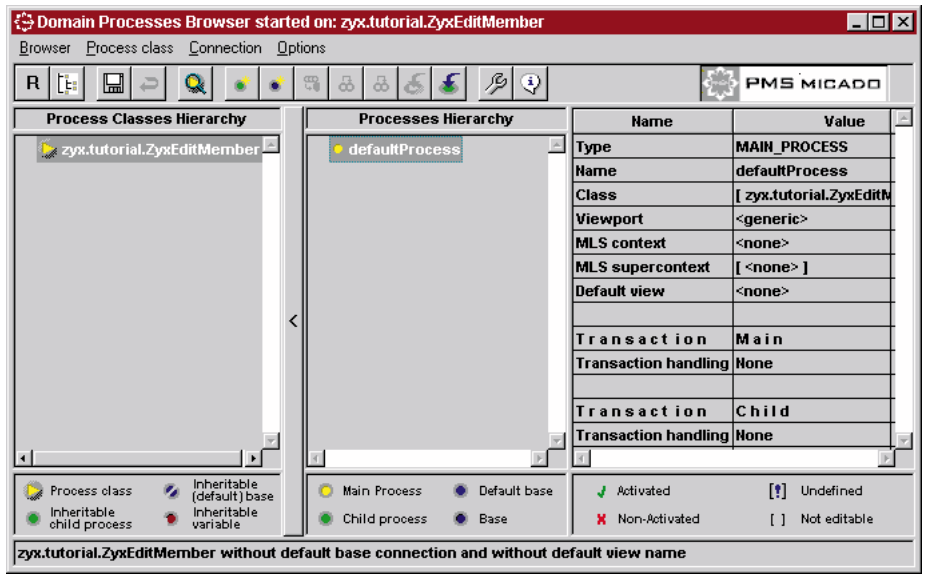


Figure 28: Domain Processes Browser on ZyxEditMember

Adding a Base Connection to a DP

A ***Base Connection*** refers to the connection between the DP class and a DO class (the DO is the "base"; the origins of the term "base" are historical in nature).

4.16. In the **Processes Hierarchy** box: Right-click on **ZyxEditMember**.

4.17. Select **Add Base Connection**.

4.18. In response to **Choose a domain object class**: Select **zyx.tutorial.ZyxMember**.

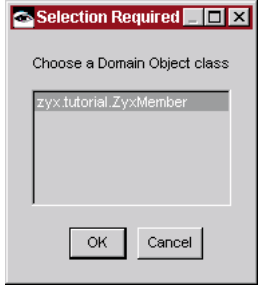


Figure 29: Dialog for selecting ZyxMember as the domain object class for ZyxEditMember

4.19. Click **OK**. Note that the DP ZyxEditMember now has a connection to ZyxMember and that the connection is named **newDefaultBaseConnection**:

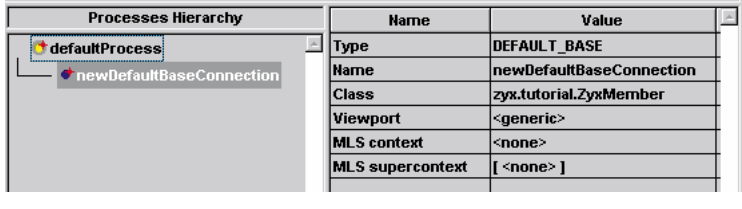


Figure 30: Base connection from ZyxEditMember to ZyxMember in DPB

Note: The **red stars** indicate that unsaved changes have been made.

Change the names of the connections

The names in the column **Processes Hierarchy** are the names of ***connections***. These names will be changed now to make it easier to distinguish the connection designations during the course of the tutorial.

Rename the DP connection

4.20. In the **Process Hierarchy Column**: Select **defaultProcess**.

4.21. In the **Value** column and in the row **Name**: Click on **defaultProcess**.

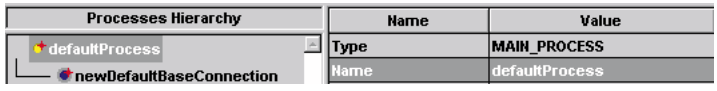


Figure 31: Default name for ZyxEditMember process connection in DPB

4.22. Change the name of the connection to **eMPConn** (edit Member Process Connection). This is the name of the connection to the DP ZyxEditMember.

4.23. Click in a different area of the Domain Processes Browser in order to reflect the change throughout the browser.

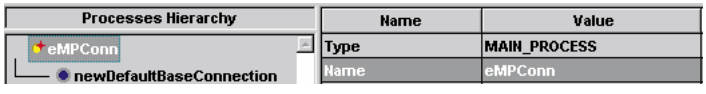


Figure 32: Renaming ZyxEditMember connection to eMPConn in DPB

Rename the DO connection

4.24. Click on **newDefaultBaseConnection** in the Processes Hierarchy column.

4.25. In the **Value** column and in the row **Name**: Click on **newDefaultBaseConnection**.

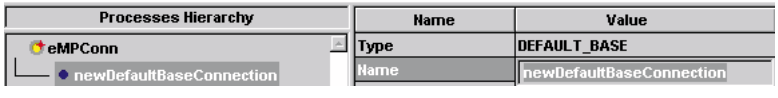


Figure 33: Default name for ZyxMember base connection in DPB

4.26. Change the name of the connection to **eMBConn** (edit Member Base Connection). This is the name of the connection to the DO ZyxMember.

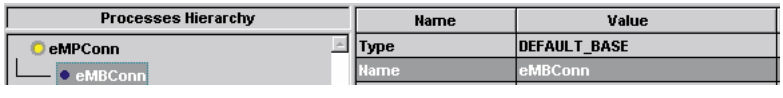


Figure 34: Renaming ZyxMember connection to eMBConn in DPB

4.27. Select **Browser / Save all changes**. Note that the red arrows have disappeared (the changes have been saved).

4.28. Close the **DPB**.

4.29. Save the workspace.

5. Create ZyxEditMemberView (View)

ZyxEditMemberView will be the first ***View*** that you create in this tutorial. ZyxEditMemberView will provide the user interface to the DP ZyxEditMember and DO ZyxMember.

For more information about views, see the ***Application Framework User's Guide*** chapter .

Create com.sun.java.swing.JFrame subclass ZyxEditMemberView.

5.1 In the **Workbench**: In project **ZyxTutorial**: Select package **zyx.tutorial**.

5.2. Right-click on **zyx.tutorial**.

5.3. Select **Add / Class...**. The Smart Guide **Create class** appears.

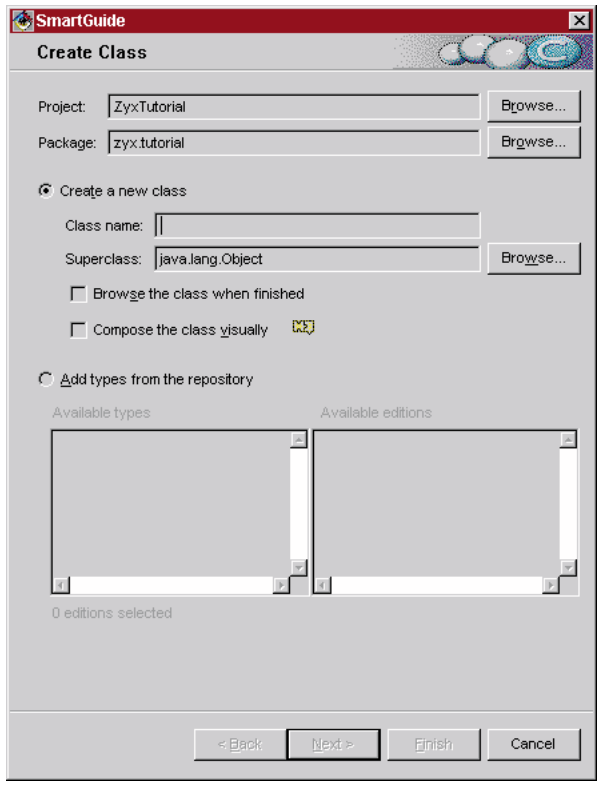


Figure 35: Smart Guide Create Class

5.4. Select **Create a new class**.

5.5. In field **Class name:**: Enter **ZyxEditMemberView**.

5.6. In field **Superclass:**: Enter **com.sun.java.swing.JFrame**.

5.7. Check the checkbox **Browse class when finished**.

5.8. Check the checkbox **Compose class visually**.

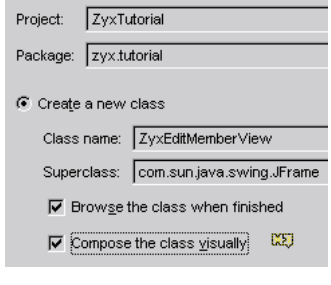


Figure 36: Create Class settings for ZyxEditMemberView

5.9. Click **Finish**. The **Composition Editor** for **ZyxEditMemberView** opens.

5.10. Double-click on the **JFrame** (NOT on the JFrameContentPane inside the JFrame). The **ZyxEditMemberView - Properties** dialog appears.

5.11. Change the **title** property to **ZyxEditMember**.

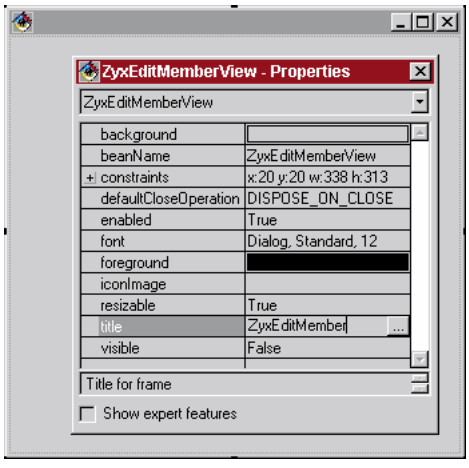


Figure 37: Change title property of ZyxEditMemberView

5.12. Add a **JLabel** bean to the view.

Figure 38: Selecting JLabel from the parts palette

5.13. Change the **JLabel** bean **text** property to **name:**.

5.14. Add a **JTextField** bean to the view.

Figure 39: Selecting JTextField from the parts palette

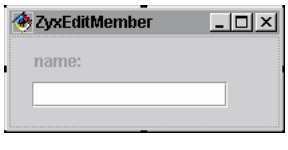


Figure 40: ZyxEditMemberView with label and text field

5.15. Change the **JTextField** bean **beanName** property to **eMBConnXXnameXX**.

5.16. Select **Bean / Save bean**.

Assign View to Process

Add ZyxEditMemberView to Model Browser class list

5.17. In the **Model Browser**: Select **File Reload**.

5.18. Expand the tree for **ZyxTutorial**.

5.19. Select **ZyxEditMemberView**.

5.20. Click on **>>**.

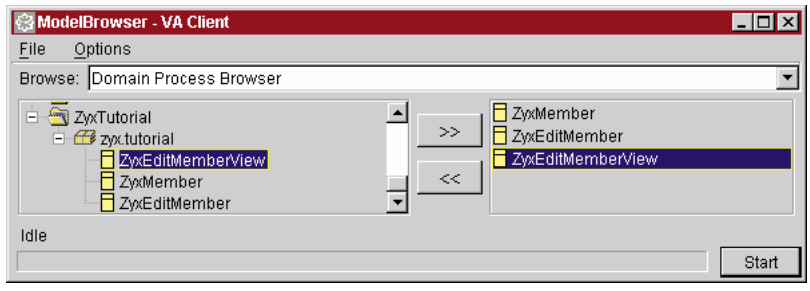


Figure 41: ZyxEditMemberView in the Model Browser class list

Assign ZyxEditMemberView to ZyxEditMember

5.21. Open the **DPB** on **ZyxEditMember**.

5.22. In the **Process Hierarchy Column**: Select **eMPConn**.

5.23. From the drop-down list in column **Value** row **Default view**: Select **zyx.tutorial.ZyxEditMember-View**.

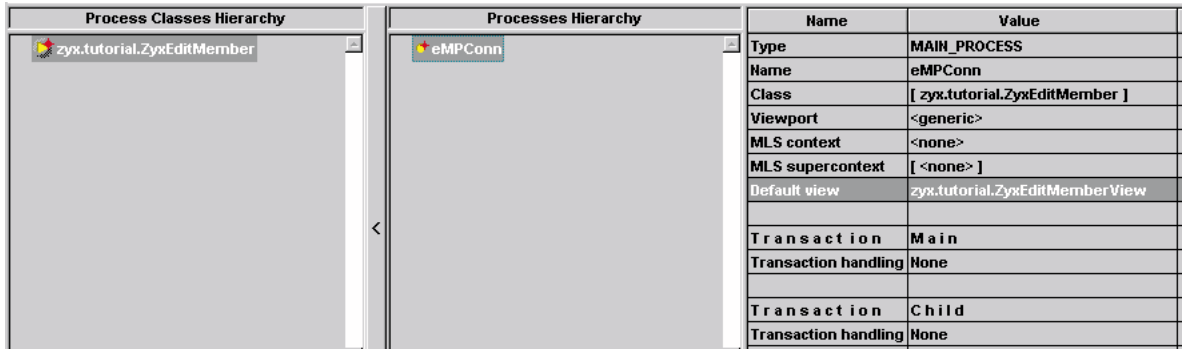


Figure 42: Assigning ZyxEditMemberView to ZyxEditMember in DPB

5.24. Select **Browser / Save all changes**.

5.25. Close the **DPB**.

5.26. Save the workspace.

Test

Add ZyxEditMember.main()

5.27. In the **Workbench**: Right-click on **ZyxEditMember**.

5.28. Select **Add / Method...**. The Smart Guide **Create method** appears.

5.29. Select **Create a new main method**.

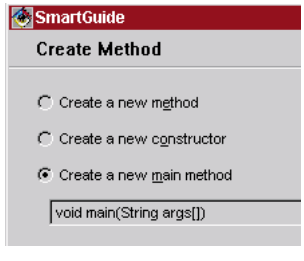


Figure 43: Smart Guide Create Method

5.30. Click **Finish**.

5.31. Select in class **ZyxEditMember** method **main**.

5.32. Add the following code to the method:

```
ZyxMember member = new ZyxMember();
member.setName("member1Name");
ZyxEditMember process = new ZyxEditMember();
process.setEMBConn(member);
process.openView();
```

5.33. Save the method (right-click in the code window and select **Save**).

Compute ZyxEditMember class path

5.34. In the **Workbench**: Right-click on **ZyxEditMember**.

5.35. Select **Run / Check class path...**. The dialog **Properties for ZyxEditMember** tab **Class path** is opened.

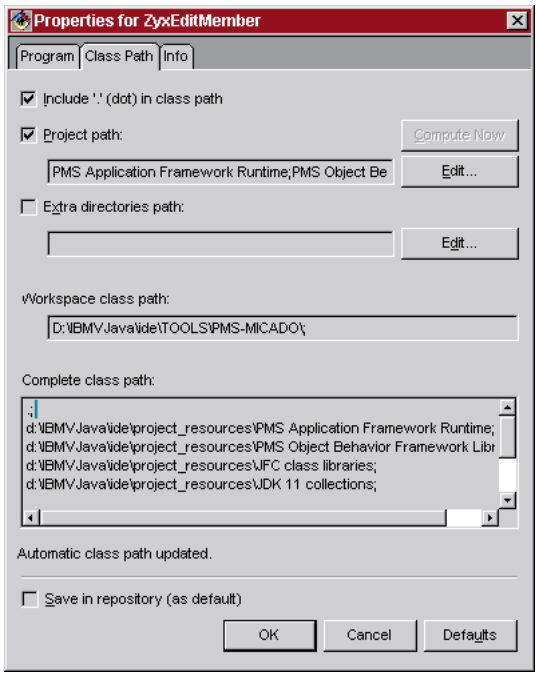


Figure 44: Properties for ZyxEditMember tab Class path

5.36. Click **Compute now**. The class path for the class is computed.

5.37. Click **OK**.

Run ZyxEditMember.main()

5.38. In the **Workbench**: Right-click on **ZyxEditMember**.

5.39. Click the **Run** icon. The following dialog appears:

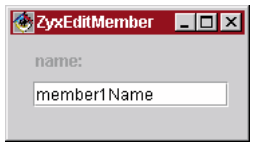


Figure 45: Display of DO ZyxMember attributes in View ZyxEditMemberView

6. Multiple non-transacted views

In this chapter you will create 2 views of the object referenced by an attribute. An object can be assigned to the attribute from either window.

Edit ZyxEditMember.main()

6.1 Change main() to the following:

```
public static void main(String args[]) {
   ZyxMember member = new ZyxMember();
   member.setName("member1Name");
   ZyxEditMember.newOpenOn(member);
   ZyxEditMember.newOpenOn(member);
}
```

Create ZyxEditMember.newOpenOn(ZyxMember) (class method)

6.2. Create the following method:

```
public static ZyxEditMember newOpenOn(ZyxMember member) {
   ZyxEditMember process = new ZyxEditMember();
   process.openOn(member);
   return process;
}
```

Create ZyxEditMember.openOn(ZyxMember)

6.3. Create the following method:

```
public void openOn(ZyxMember member) {
   setEMBConn(member);
   openView();
}
```

6.4. Save the workspace.

Open 2 views

6.5. Run ZyxEditMember main. A single ZyxEditMember dialog appears (the other dialog is behind this dialog).

6.6. Move the displayed ZyxEditMember dialog (with the focus; this was the last dialog to be created) to the RIGHT. The ZyxEditMember dialog that was hidden behind the RIGHT dialog will be the LEFT dialog (the first dialog created):

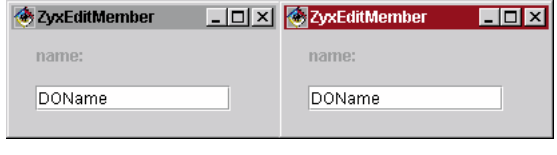


Figure 46: 2 views of same object.

Test non-transacted changes

Both views are of the same object. The changes are not transacted (transacted changes will be discussed in the next section).

6.7. In the **LEFT** dialog: Change the name to **DOName1.** The new String object will be the new object referenced by the ZyxMember object attribute name.

6.8. Click in the **RIGHT** dialog.

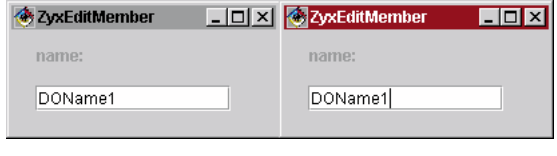


Figure 47: Non-transacted changes in a view are reflected in other views

Clicking in the RIGHT dialog changed the focus, causing the following:

- The String object entered in the text field in the left dialog was assigned to the ZyxMember object attribute **name**.
- The right view (ZyxEditMemberView) is updated (and thus the current ZyxMember object attributes are displayed).

6.9. In the **RIGHT** dialog: Change the name to **DOName2**.

6.10. Click in the **LEFT** dialog.

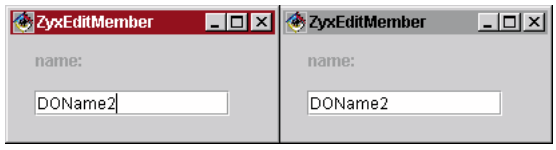


Figure 48: Non-transacted changes to the same object can be made in multiple view dialogs

6.11. Note that the attributes of the ZyxEditMember object can be changed in either dialog.

7. Multiple views: transacted non-isolated

Transacted changes

The problem with the situation in the previous chapter is that an object can be changed from any view, overwriting the changes made in any other views. This undesireable situation will be avoided by transacting changes (demonstrated in this chapter).

For more information about transactions, see the ***Object Behavior Framework User's Guide*** chapter .

Define ZyxEditMember as transacted process

7.1 Open the **DPB** on **ZyxEditMember**.

7.2. In the **Processes Hierarchy** box: Select **eMPConn**.

7.3. For the **Transaction Main**: In the **Value** column in row **Transaction handling**: Click on **None**. 3 radio buttons for defining the transaction properties of the main process are displayed.

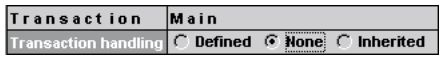


Figure 49: Setting the Transaction Main for ZyxEditMember

7.4. Select radio button **Defined**.

7.5. Click elsewhere in the Processes Browser to enter the change. Note the default setting for the transacted process:

Transaction	Main
Transaction handling	Defined
Autostart mode	✓
Use own context	[✓]
Use parent context	[✗]
Isolate mode	✗
Persistence context	✗
Transaction context	✓
Hierarchical mode	✗

Figure 50: Default settings in the DPB for Transaction Main

Note that the transaction context is non-isolation mode and is enabled .

7.6. **Save all changes**.

7.7. Close the **DPB**.

Specify ZyxMember>>name as transacted

The DP is now specified as being transacted. However, changes made to object (ZyxMember) attributes (name) will not be transacted unless the actual attribute is defined as being transacted with the OME (this causes the necessary methods for supporting transaction behavior to be created for the attribute).

7.8. Open the **OME** on **ZyxMember**.

7.9. Select **name**.

7.10. In the tab **Variable**: Check the checkbox **Transact**:

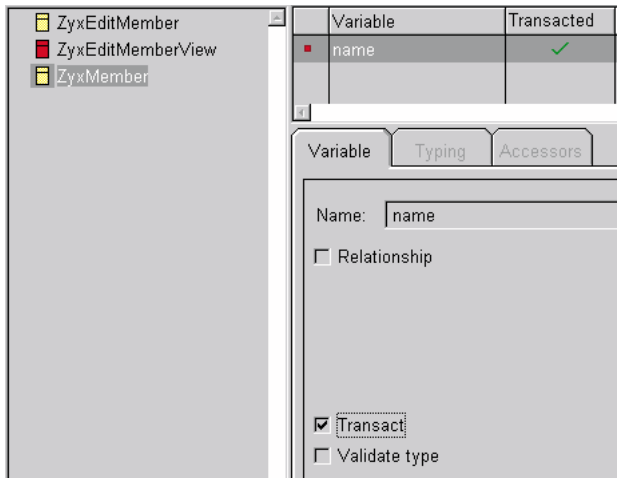


Figure 51: Specifying ZyxMember>>name as transacted in OME

7.11. **Save the changes to VA**.

Modify ZyxEditMember.main() to open the Transaction Browser

7.12. Add a line to main() to display the TB:

```
public static void main(String args[]) {
   ZyxMember member = new ZyxMember();
   member.setName("DOName");
   COM.pmsc.tools.ofw.tabrowser.TransactionsBrowser.openOn(member.getTrans-
actionManager());
   ZyxEditMember.newOpenOn(member);
   ZyxEditMember.newOpenOn(member);
}
```

NOTE the spelling of "TransactionsBrowser" (with an "s") above.

7.13. Save the method.

7.14. Save the workspace.

Test

7.15. Run ZyxEditMember main. The Transaction Browser (TB) and 2 dialogs are displayed:

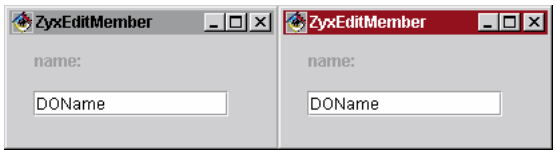


Figure 52: 2 views of same object.

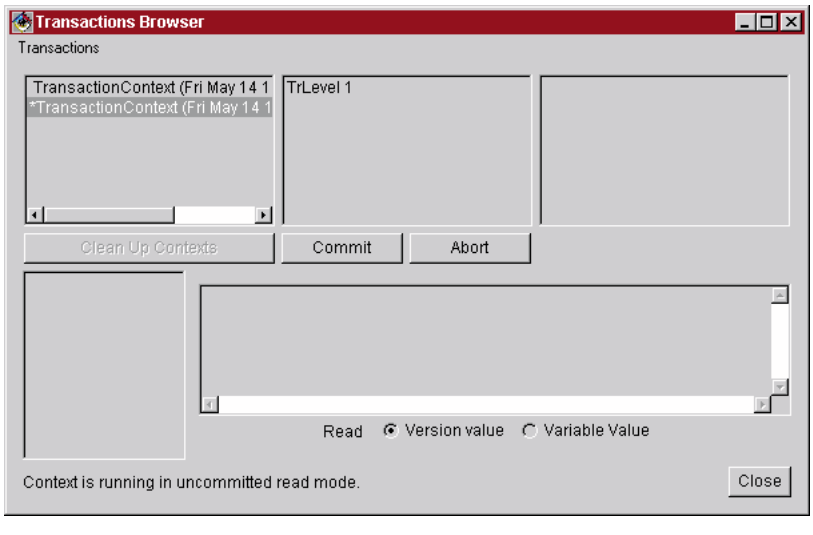


Figure 53: Transaction Browser dialog

For more information about the Transaction Browser, see the ***Object Behavior Framework User's Guide*** chapter .

Active and Inactive Transaction contexts

Note that 2 ***MicFwTransactionContext***'s were created when the dialogs were opened. The ***inactive context*** (the context without the asterisk ("*") in front of it) is assigned to the window that is currently not selected (does not have the focus). The ***active context*** (the context with the asterisk ("*") in front of it) is assigned to the window that is currently selected (has the focus).

7.16. Click on the active **MicFwTransactionContext**.

7.17. Click on **TrLevel1**. Note that no changes have been made in the context yet.

Figure 54: No transacted changes for the object whose attributes havent been changed in the view

TrLevel's (Transaction levels)

A dialog can have subdialogs. In such situations the nested dialogs can belong to the same transaction context, but each dialog has a different transaction level. For now we will be using only a single TrLevel. TrLevels are described in detail in the OBF User's Guide.

Assign a new object to a transacted variable

7.18. In the **LEFT** dialog: Change the name to **DOName1**.

7.19. Click in the **RIGHT** dialog. Note that the change in focus updates both views.

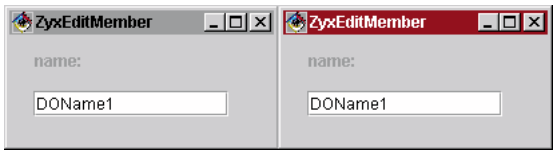


Figure 55: The second view of an object is updated after transacted changes in the first view

Transacted change: View transaction info in TB

7.20. In the Transaction Browser: Click on the **inactive transaction context** (the context without the asterisk "*" in front of it). This is the left window in which the change to the name was entered (the right window was the last window activated, and therefore the transaction context for that window is active).

7.21. Click on **TrLevel1**. Note that TrLevel 1 includes a ZyxMember object. Note: If the ZyxMember object is not displayed, refresh the TB view by clicking on the other Transaction Context.

Figure 56: Transaction info in the TB after an object has been assigned to transacted attribute

7.22. Click on **zyx.tutorial.ZyxMember@xxx**. Note that a **name** attribute appears.

7.23. Select **Read Version Value**.

7.24. Click on **name**. Note that the ***Version value*** **'DOName1'** is displayed. Note: Versioned objects are described in detail in the OBF User's Guide.

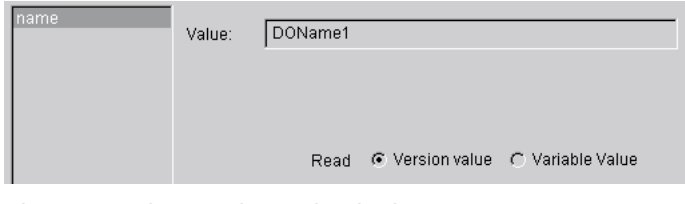


Figure 57: The Version Value in the TB

7.25. Select ***Variable value***. The displayed variable value is **'DOName'**.

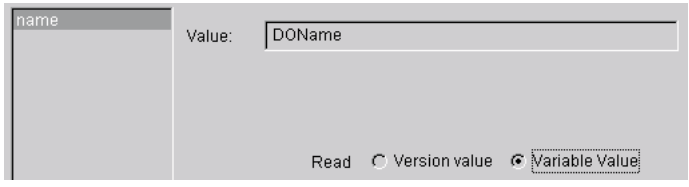


Figure 58: The Variable Value in the TB

What actually occurred when the new object ('DOName1') was assigned to the transacted variable

When a new object was "assigned" to the object attribute in the LEFT dialog, the following occurred:

- An version object was created for and assigned to the attribute. This version object references 2 objects: The original object assigned to the variable (the variable value 'DOName'), and the latest object assigned to the variable while the transaction was active (the version value 'DOName1').
- The version value was displayed in the LEFT dialog.

The version value of the transacted attribute was displayed in the RIGHT dialog when the dialog was updated (clicked on).

Attempt to assign a new object to variable locked by another transaction

7.26. In the **RIGHT** dialog: Change the name to **DOName2**.

7.27. Click in the **LEFT** dialog. A **exception** is thrown.

990514TTA??: what kind of exception?? when i did this only the console popped up with some benign messages.

***Variable locking* by a transaction context**

When a new object is assigned to a transacted variable within an active context, the variable is locked by that context (a context is typically activated when the dialog that the context is assigned to is clicked on). If an attempt is made to assign an object to the variable while a different context is active, a transaction write conflict exception is thrown.

Abort a transaction context

7.28. In the TB: Make sure that TrLevel1 of the the transaction context that contains the transacted changes is selected.

7.29. Click **Abort**. This aborts TrLevel1, which also aborts the context (a context is always aborted if TrLevel1 is aborted), which also aborts the changes that were made in the dialog. Note that the transaction context no longer exists in the TB. Note also that the dialog content has been updated:

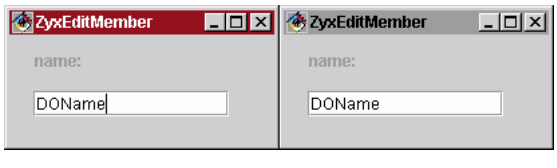


Figure 59: Aborted changes as reflected in the views

Note: Normally transacted changes are aborted or committed by clicking on a dialog button such as "Cancel" or "Accept changes". Adding such buttons to the dialog will be demonstrarted later in this tutorial.

Assign new object to transacted variable (in RIGHT dialog)

7.30. In the RIGHT dialog: Change the name to **'DOName2'**.

7.31. Click on the LEFT dialog. Note that the dialog was updated. When the context for the left window was aborted, the lock on the variable was released. The variable is now locked by the context for the RIGHT dialog.

Close the dialogs

7.32. Close the RIGHT **ZyxEditMember** dialog.

7.33. Note in the **TB** that the transaction context no longer exists.

7.34. Close the LEFT **ZyxEditMember** dialog.

7.35. Close the **TB**.

8. Multiple views (single object): transacted isolated

Isolated transaction contexts

Sometimes you may not want the versioned objects from other dialogs to be displayed in a dialog (ie, you want only the version values to be displayed). This can be accomplished by specifying the Transaction Main for the DP to be in isolatedMode.

Define ZyxEditMember as transacted ISOLATED process

8.1 Open the **DPB** on **ZyxEditMember**.

8.2. In the **Processes Hierarchy** box: Select **eMPConn**.

8.3. For the **Transaction Main**: In the **Value** column in row **Isolate mode**: Click on the **X**. 2 radio buttons "true" and "false" appear.

8.4. Select **true**.

Transaction	Main
Transaction handling	Defined
Autostart mode	√
Use own context	[√]
Use parent context	[x]
Isolate mode	◉ true ○ false
Persistence context	x
Transaction context	√
Hierarchical mode	x

Figure 60: Specifying transaction mode isolate for Transaction Main in OME

8.5. Click elsewhere in DPB to register the change.

8.6. **Save all changes**.

8.7. Close the **DPB**.

8.8. Save the workspace.

Test

8.9. Run ZyxEditMember main.

8.10. In the **LEFT** dialog: Change **name** to **DOName1**.

8.11. Click in the **RIGHT** dialog. The new string object is NOT displayed.

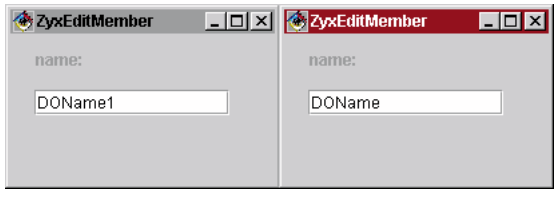


Figure 61: Isolated transacted changes in multiple views

8.12. Close both **ZyxEditMember** dialogs and the TB.

9. Aborting/committing transacted changes with the TB

9.1

10. Aborting/committing transacted changes with abort/commit buttons

ZyxEditMember is a subclass of DomainProcess, which provides standard methods for aborting and committing transacted changes without closing the view (...AndBegin() methods; methods that close the view (...AndCloseView) will be explored later in this tutorial). These methods can be connected to a button in a view using the naming convention.

Add commitAndBegin and abortAndBegin buttons to ZyxEditMemberView

10.1 In the **Composition Editor** for **ZyxEditMemberView**: Add 2 **JButton** beans to the view.

Figure 62: JButton bean in the parts palette

Figure 63: ZyxEditMemberView with 2 buttons

10.2. Set **JButton1** title to **AbortAndBegin**.
10.3. Set **JButton1** name to **eMPConnXXabortAndBeginXX**.
10.4. Set **JButton2** title to **CommitAndBegin**.
10.5. Set **JButton2** name to **eMPConnXXcommitAndBeginXX**.

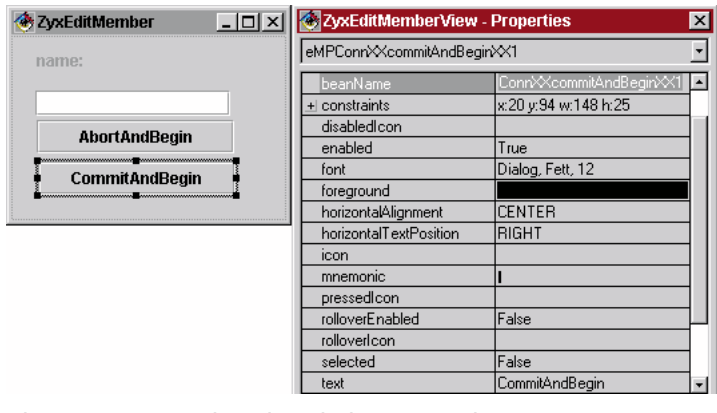


Figure 64: CommitAndBegin button settings

10.6. **Save the bean**.
10.7. Save the workspace.

Test

Isolated transaction

Abort changes

10.8. Run **ZyxEditMember.main()**.
10.9. In the **LEFT** dialog: Change **name** to **DOName1**.
10.10. Click in the **RIGHT** dialog. The new string object is NOT displayed.
10.11. In the **LEFT** dialog: Click **AbortAndBegin**. Note that **name** returns to **DOName**.

Commit changes

10.12. In the **LEFT** dialog: Change **name** to **DOName2**.

10.13. In the **LEFT** dialog: Click **CommitAndBegin**. Note that in the **RIGHT** dialog **name** changes to **DOName2**.

Figure 65: ZyxEditMemberView's with abort and commit buttons

10.14. Close the **ZyxEditMember** and the **TB** dialogs.

Non-Isolated transaction

Set ZyxEditMember transaction mode to non-isolated

The following example will show how the abortion of a change in a non-isolated transaction is reflected in all dialogs.

10.15. In the **DPB**: For **ZyxEditMember** process connection **eMPConn**: Set **Transaction Main** mode **Isolated** to **False** (a red arrow should be displayed).

10.16. Save all changes.

Abort changes

10.17. Run **ZyxEditMember.main()**.

10.18. In the **LEFT** dialog: Change **name** to **DOName3**.

10.19. Click in the **RIGHT** dialog. The new string object IS displayed.

10.20. In the **LEFT** dialog: Click **AbortAndBegin**. Note that **name** returns to **DOName** in **LEFT** and **RIGHT** dialogs.

11. Aborting/committing transaction contexts with abort/commit buttons

The context that is assigned to the dialog can be aborted/committed from the dialog with the ...AndClose-View() methods.

Change the ZyxEditMember view buttons to abortAndCloseView/commitAndCloseView

11.1 In the **Composition Editor** for **ZyxEditMemberView**: Set the **AbortAndBegin** button title to **AbortAndCloseView**.

11.2. Set **AbortAndCloseView** name to **eMPConnXXabortAndCloseViewXX**.

11.3. Set the **CommitAndBegin** button title to **CommitAndCloseView**.

11.4. Set **CommitAndCloseView** name to **eMPConnXXcommitAndCloseViewXX**.

Figure 66: ZyxEditMemberView with ...AndCloseView buttons

11.5. Save the bean.

11.6. Save the workspace.

Test (non-isolated transactions)

11.7. Run **ZyxEditMember.main()**.

11.8. In the **LEFT** dialog: Change **name** to **DOName1**.

11.9. Click in the **RIGHT** dialog. The new string object IS displayed.

11.10. In the **LEFT** dialog: Click **AbortAndCloseView**. The **LEFT** dialog closes and **name** in the RIGHT dialog returns to **DOName**.

Figure 67: After LEFT dialog aborted: Original values displayed in RIGHT dialog

11.11. In the **RIGHT** dialog: Change **name** to **DOName2**. Note that no transaction exception is thrown, since the context for the LEFT dialog no longer exists (and thus has no lock on the variable).

11.12. Click **CommitAndCloseView**. The dialog closes.

12. Validate type and range of input

This chapter will demonstrate how to validate the type and the range of input. However, since currently there is only 1 field in the view and it accepts String input, it is not possible to enter an invalid type. So a second field will be added that expects an Integer.

Validating input also requires that a ***ViewPort*** is also created for DO ZyxMember.

Add ZyxMember.weight

12.1 Open the **OME** on **ZyxMember**.

12.2. Add **transacted** variable **weight**.

12.3. In the tab **Typing**: From the drop-down list **Type**: Select **float**.

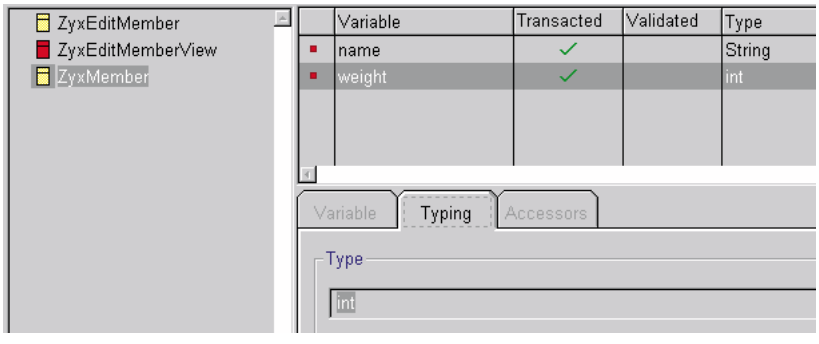


Figure 68: Variable ZyxMember>>weight in OME

12.4. **Save changes to VA**.

12.5. Close **OME**.

In ZyxEditMemberView: Add JLabel and JTextField for weight

12.6. Open **ZyxEditMemberView** in the **Composition Editor**.

12.7. Add a **JLabel** and a **JTextField** bean between the existing text field and the buttons.

12.8. For **JLabel**: Change property **text** to **weight:**.

12.9. For **JTextField**: Change property **beanName** to **eMBConnXXweightXX**.

Figure 69: ZyxEditMemberView with label and text field for weight

12.10. **Save the bean**.

Create ViewPort subclass ZyxMemberViewPort

13.1 In **Workbench**: Right-click on **zyx.tutorial**.

13.2. Select **Add / Class...**.

13.3. In the field **Class name:**: Enter **ZyxMemberViewPort**.

13.4. In the field **Superclass:**: Enter **COM.pmsc.afw.viewPorts.ViewPort**.

13.5. Uncheck the checkbox **Compose the class visually**.

13.6. Click **Finish**.

Register ZyxMemberViewPort in Model Browser

13.7. In the **Model Browser**: Select **File / Reload**.

13.8. Expand the tree for **ZyxTutorial**.

13.9. Select **ZyxMemberViewPort**.

13.10. Click on **>>**.

Assign ZyxMemberViewPort as the ViewPort for eMBConn

A ***Generic viewport*** is assigned to **eMBConn** by default. ZyxMemberViewPort must now be assigned to the connection.

13.11. Open **DPB** on **ZyxEditClub**.

13.12. in **Processes Hierarchy**: Expand the tree for **eCPConn**.

13.13. Expand the tree for **eMPChConn**.

13.14. Select **eMBConn**.

13.15. In column **Value** row **Viewport**: From the drop-down list: Select **zyx.tutorial.ZyxMemberView-Port**.

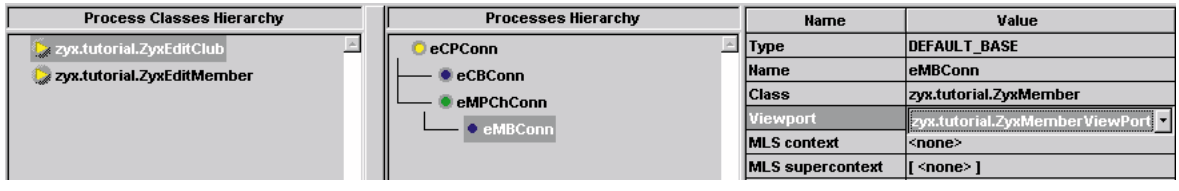


Figure 70: Assigning ZyxMemberViewPort as the ViewPort for the base connection to ZyxMember

13.16. **Save all changes**.

Add ZyxMemberViewPort.getWeight(), .setWeight()

13.17. Add the following line to ZyxMemberViewPort:

```
import java.text.*;
```

13.18. Create **ZyxMemberViewPort.getWeight()**:

```
public String getWeight() {
   float w = ((ZyxMember) getModel()).getWeight();
   return NumberFormat.getInstance().format(w);
}
```

13.19. Create **ZyxMemberViewPort.setWeight()**:

```
public void setWeight(String weight) {
   ZyxMember m = (ZyxMember) getModel();
   try {
      float w = NumberFormat.getInstance().parse(weight).floatValue();
      if (w < 20 || w >200) {
         m.setWeight(m.getWeight());
      }
      else {
         m.setWeight(w);
      }
   }
   catch(ParseException ex) {
      m.setWeight(m.getWeight());
   }
}
```

13.20. Save the workspace.

Test

13.21. Run **ZyxEditMember.main()**. The following dialogs appear:

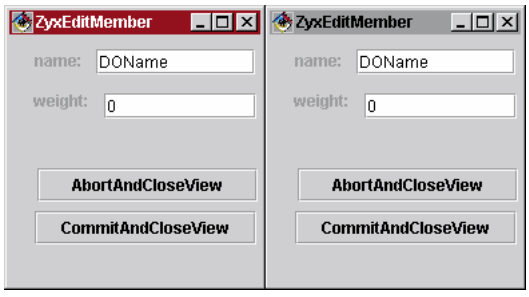


Figure 71: 2 ZyxEditMember dialogs with weight field

Enter value of NOT valid type

13.22. In the **LEFT** dialog: Enter "**abc**".

13.23. Click in the **RIGHT** dialog. Note that the the value of invalid type is displayed in the LEFT dialog; however, the value is NOT displayed in the RIGHT dialog (transaction settings are non-isolated):

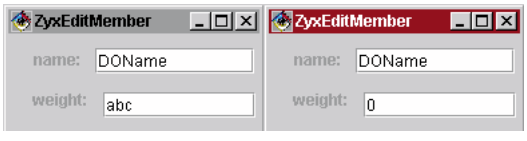


Figure 72: Value of invalid type in LEFT dialog not displayed in RIGHT dialog

Enter value of valid type

13.24. In the **LEFT** dialog: Enter **123**.

13.25. Click in the **RIGHT** dialog. Note that the the value of valid type is displayed in the RIGHT dialog:

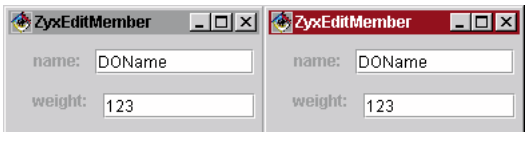


Figure 73: Value of valid type in LEFT dialog is displayed in RIGHT dialog

Enter value out of range (20 ... 200)

13.26. In the **LEFT** dialog: Enter **300**.

13.27. Click in the **RIGHT** dialog. The value of the weight in the LEFT dialog changes to the previous valid value.

13. Create DO ZyxClub

The DO object ZyxClub will have 2 attributes:

- members. members will reference a collection of ZyxMember instances.
- currentMember. currentMember will reference the ZyxMember instance that is currently selected in ZyxEditClubView (to be created later).

Create DomainObject subclass ZyxClub.

14.1 Open the **OME**.

14.2. Click on the button **New class**. The **Class specification** dialog appears.

14.3. In tab **Definition** field **Class name:**: Enter **ZyxClub**.

14.4. In tab **Definition** drop-down list **Inherits from:**: Enter **DomainObject**.

14.5. In tab **Java specification** field **Package name:**: Enter **zyx.tutorial**.

14.6. Click **OK**.

Add ZyxClub>>members

The **OME** will be used to establish a ***relationship*** between ZyxClub>>members and ZyxMember. The relationship will be specified as being a ***primitive relationship***, which means that the members attribute will reference a collection of ZyxMember objects, but the ZyxMember objects will have no attribute that reference the ZyxClub instance.

14.7. In the **OME**: Add instance variable **ZyxClub>>members**.

14.8. In tab **Variable**: Check the checkbox **Relationship**.

14.9. In tab **Variable**: Check the checkbox **Transacted**.

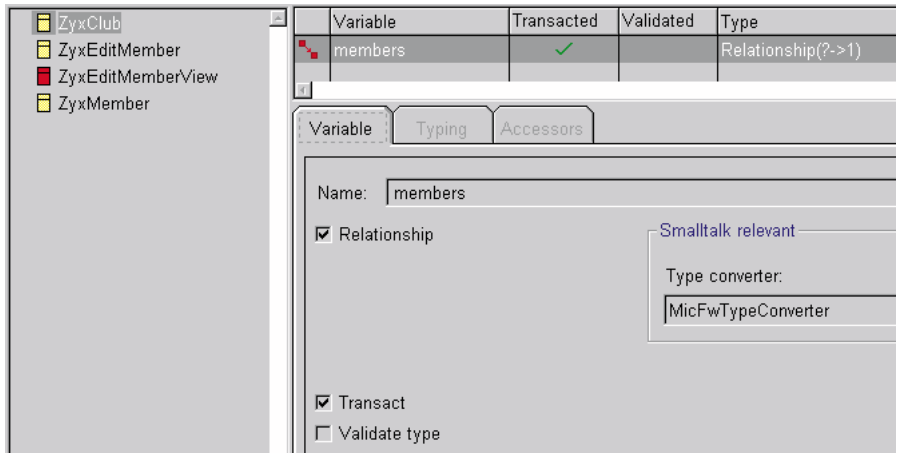


Figure 74: Specifying a relationship in OME

14.10. In tab **Typing**: In group box **Target class**: In drop-down list **Class:**: Select **ZyxMember**.

14.11. In group box **Source class**: Check the checkbox **N**.

14.12. Check the checkbox **Primitive**.

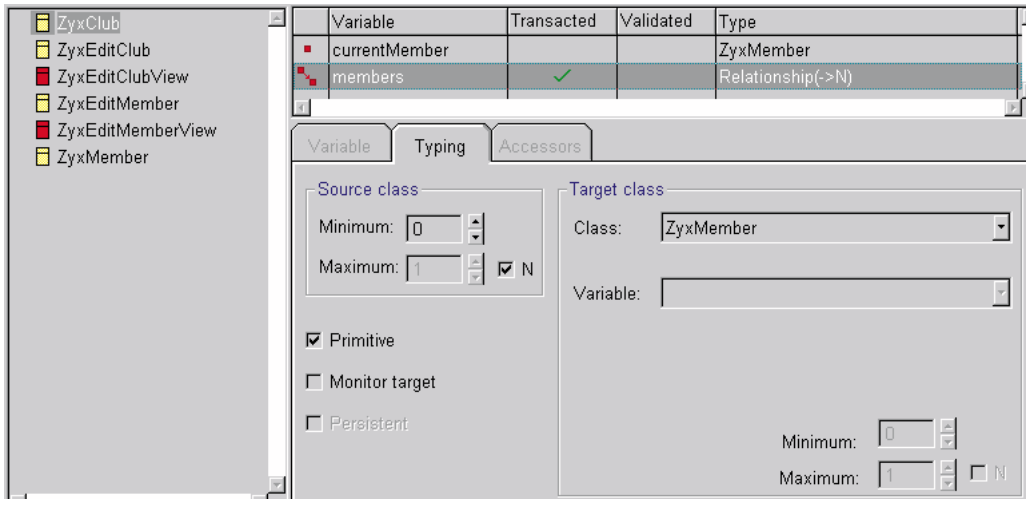


Figure 75: Specifying target class and cardinality in OME

Add ZyxClub>>currentMember

14.13. In the **OME**: Add instance variable **ZyxClub>>currentMember**.

14.14. In tab **Variable**: Check the checkbox **Transacted**.

14.15. In tab **Typing**: In drop-down list **Type**: Select **ZyxMember**.

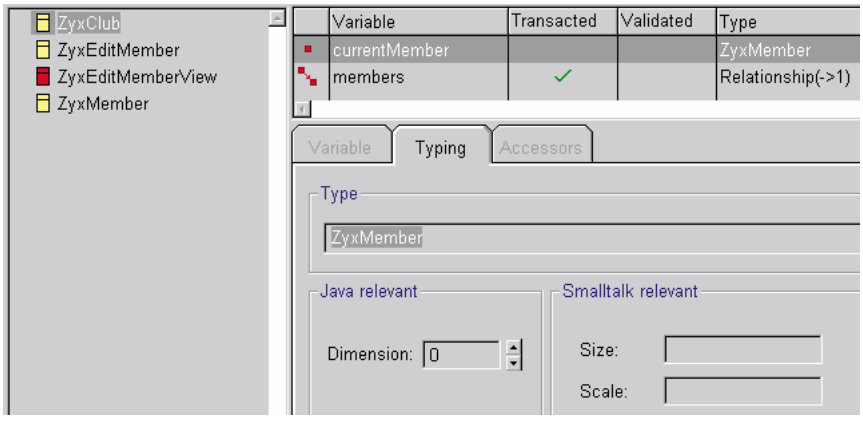


Figure 76: Typing ZyxClub>>currentMember in OME

14.16. **Save to VA**.

14.17. Save the workspace.

14. Create DP ZyxEditClub

ZyxEditClub will be the DP for editting the attributes of the ZyxClub instances. ZyxEditMember will be specified as the ***child process*** of ZyxEditClub, thus allowing ZyxEditMemberView to be opened from the ZyxEditClubView.

Create DomainProcess subclass ZyxEditClub

15.1 In the **OME**: Right-click in the left part of the window.

15.2. Select **New Class**. The **Class specification** dialog appears.

15.3. In field **Class Name:**: Enter **ZyxEditClub**.

15.4. From drop-down list **Inherits from:**: Select **Domain Process**.

15.5. In tab **Java specification**: In field **Package name**: Enter **zyx.tutorial**.

15.6. Click **OK**.

15.7. In **OME**: **Save to VA**.

15.8. Close **OME**.

Add ZyxEditClub, ZyxClub to Model Browser class list

15.9. In the **Model Browser**: Select **File / Reload**.

15.10. Expand the tree for **ZyxTutorial**.

15.11. Select **ZyxEditClub** and **ZyxClub** (using Ctrl key).

15.12. Click on **>>**.

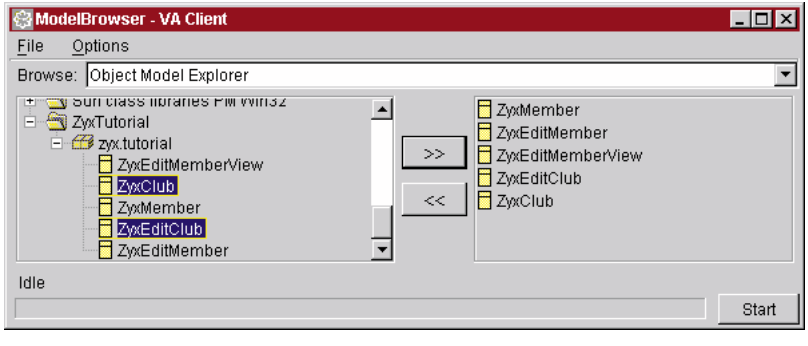


Figure 77: ZyxEditClub, ZyxClub in Model Browser class list

Change transaction settings for ZyxEditClub

15.13. Open **DPB** on **ZyxEditClub**.

Set transaction to Defined

15.14. In the **Process Hierarchy Column**: Select **defaultProcess**.

15.15. Set the **Transaction Main** to **Defined** (use the standard settings for defined).

Rename the DP connection

15.16. In the column **Value** in the row **Name**: Click on **defaultProcess**.

15.17. Change the name of the connection to **eCPConn** (edit Club Process Connection). This is the name of the connection to the DP ZyxEditClub.

15.18. Click in a different area of the Domain Processes Browser in order to reflect the change throughout the browser.

Adding ZyxEditClub base connection to ZyxClub

15.19. In the **Processes Hierarchy** box: Right-click on **eCPConn**.

15.20. Select **Add Base Connection**.

15.21. In response to **Choose a domain object class**: Select **zyx.tutorial.ZyxClub**

15.22. Click **OK**. Note that the DP ZyxEditMember now has a connection to ZyxClub and that the connection is named **newDefaultBaseConnection**.

Rename the DO connection

15.23. Click on **newDefaultBaseConnection** in the Processes Hierarchy column.

15.24. In the column **Value** in the row **Name**: Click on **newDefaultBaseConnection**.

15.25. Change the name of the connection to **eCBConn** (edit Club Base Connection). This is the name of the connection to the DO ZyxClub.

Add ZyxEditMember as child process of ZyxEditClub

15.26. In the **Processes Hierarchy** box: Click on **eCPConn**.

15.27. Right-click.

15.28. Select **Add child connection**.

15.29. In response to **Choose a class**: Select **ZyxEditMember**.

15.30. Click **OK**. ZyxEditMember is added as a child process with a connection named **newChildProcessConnection**. Note that the list in **Process Classes Hierarchy** now includes **zyx.tutorial.ZyxEditMember**.

Set transaction to Defined

15.31. Click on **newChildProcessConnection** in the Processes Hierarchy column.

15.32. Set the **Transaction Main** to **Defined** (use the standard settings for defined).

Change the name of the connection

15.33. Click on **newChildProcessConnection** in the Processes Hierarchy column.

15.34. Change the name of connection **newChildProcessConnection** to **eMPChConn** (edit Member Process Child Connection).

15.35. Click anywhere in the DPB to register the changes.

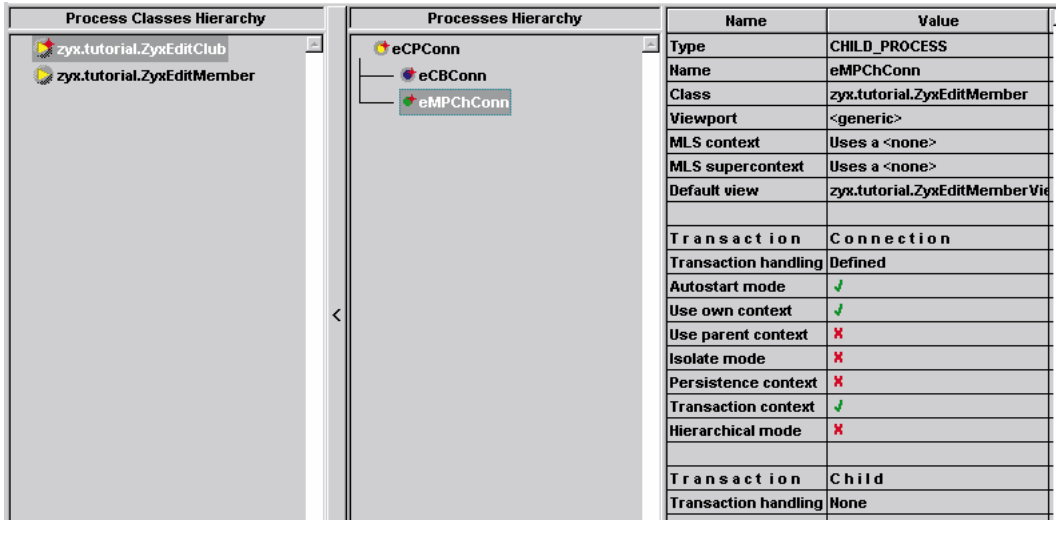


Figure 78: Renaming connections in DPB

15.36. Select **Browser / Save all changes**.

15.37. Close the **DPB**.

15.38. Save the workspace.

Additions to ZyxEditClub

15.39. Add the following to ZyxEditClub:

```
import COM.pmsc.afw.collectionSupport.*;
```

15.40. Add the following variable to ZyxEditClub:

```
private AfwList members;
```

15.41. getMembers()

```
public AfwList getMembers() {
   if (members == null) {
      members = new NRelationshipAfwList (
         getECBConn().getMembers());
   }
   return members;
}
```

15.42. setSelectedMember / getSelectedMember: are the accessors for the member that is currently selected in the process.

```
public void setSelectedMember(ZyxMember member) {
  getECBConn().setCurrentMember(member);
}

public ZyxMember getSelectedMember() {
  return getECBConn().getCurrentMember();
}
```

15.43. openOn: establishes the base connection to ZyxClub and then sends the openView message.

```
public void openOn(ZyxClub club) {
  setECBConn(club);
  openView();
}
```

15.44. edit...

```
public void edit() {
  ZyxMember m = getECBConn().getCurrentMember();
  if (m != null) {
    getEMPChConn().openOn(m);
  }
}
```

Additions to ZyxEditMember

15.45. add this constructor:

```
public ZyxEditMember(IfDomainProcess parentProcess, MicAtom name) {
  super (parentProcess, name);
  }
```

15.46. Save the image.

15. Create View ZyxEditClubView

ZyxEditClubView will be the view for displaying, selecting, adding, deleting, and editting members.

Create com.sun.java.swing.JFrame subclass ZyxEditMemberView.

16.1 In the **Workbench**: In project **ZyxTutorial**: Right-click on package **zyx.tutorial**.

16.2. Select **Add / Class...**. The Smart Guide **Create class** appears.

16.3. Select **Create a new class**.

16.4. In field **Class name:**: Enter **ZyxEditClubView**.

16.5. In field **Superclass:**: Enter **com.sun.java.swing.JFrame**.

16.6. Check the checkbox **Browse class when finished**.

16.7. Check the checkbox **Compose class visually**.

16.8. Click **Finish**. The **Composition Editor** for **ZyxEditClubView** opens.

Change title

16.9. Double-click on the **JFrame** (NOT on the JFrameContentPane inside the JFrame). The **ZyxEdit-ClubView - Properties** dialog appears.

16.10. Change the **title** property to **ZyxEditClub**.

Add Edit button

16.11. Add a **JButton** bean to the view.

16.12. Set the property **text** to **Edit...**.

16.13. Set the property **beanName** to **eCPConnXXeditXX**.

Add drop-down list

16.14. Add a **JComboBox** bean to the view.

16.15. Set the property **beanName** to **eCPConnXXmembersXXeCPConnXXselectedMemberXX**.

16.16. Add a **JButton** bean to the view.

Add Close button

16.17. Set the property **text** to **Close**.

16.18. Set the property **beanName** to **eCPConnXXcloseXX**.

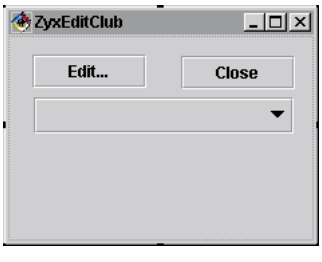


Figure 79: ZyxEditClub Close button

16.19. **Save the bean**.

Add ZyxEditClubView to Model Browser class list

16.20. In the **Model Browser**: Select **File / Reload**.

16.21. Expand the tree for **ZyxTutorial**.

16.22. Select **ZyxEditClubView**.

16.23. Click on **>>**.

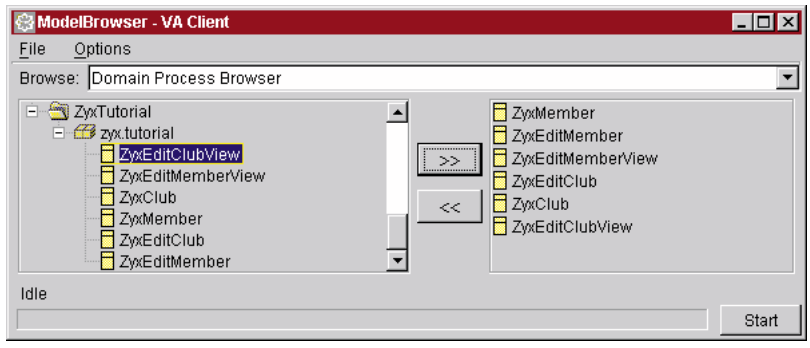


Figure 80: ZyxEditClubView in Model Browser Class list

Assign ZyxEditClubView to ZyxEditClub

16.24. Open the **DPB** on **ZyxEditClub**.

16.25. In the **Process Hierarchy Column**: Select **eCPConn**.

16.26. From the drop-down list in column **Value** row **Default view**: Select **zyx.tutorial.ZyxEditClubView**.

16.27. Select **Browser / Save all changes**.

16.28. Close the **DPB**.

Create ZyxMember>>getAsListEntry

The **getAsListEntry** method must be implemented by a DO when instances of the DO may be displayed in a view with a drop-down list. The object returned by the method is the object that is displayed in the list (typically a string).

16.29. Create the following method:

```
public String getAsListEntry() {
   return getName();
}
```

16.30. Save the workspace.

Test

16.31. Add the following method ZyxEditClub.main():

```
public static void main(String args[]) {
   ZyxEditClub p;
   ZyxClub c;
   ZyxMember m;

   c = new ZyxClub();
   m = new ZyxMember();
   m.setName("member1Name");
   c.getMembers().add(m);
   m = new ZyxMember();
   m.setName("member2Name");
   c.getMembers().add(m);

   p = new ZyxEditClub();
   p.openOn(c);
}
```

16.32. Recompute the **class path** for **ZyxEditClub** (right-click on **ZyxEditClub**; select **Run / Check Class Path...**; select **Compute Now**; click **Yes**; click **OK**).

16.33. Run ZyxEditClub.main(). The following dialog is opened:

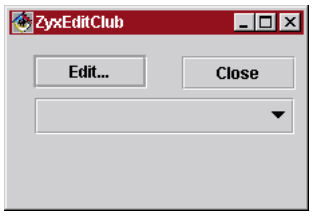


Figure 81: ZyxEditClubView dialog

16.34. Select a member1 from the drop-down list:

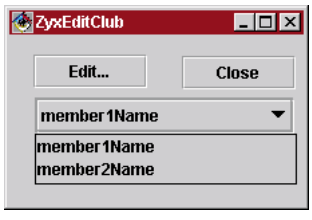


Figure 82: Selecting a member from the ZyxEditClubView drop-down list

16.35. Click **Edit...**. The **ZyxEditMemberView** appears.

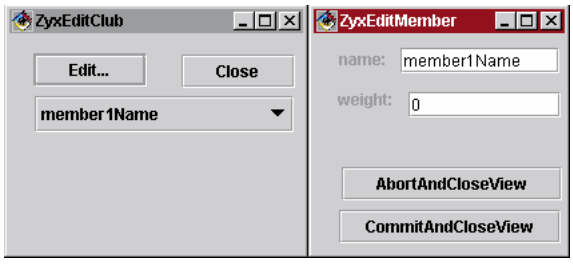


Figure 83: ZyxEditMemberView as a child of ZyxEditClubView

16.36. Test the views. Eventually click **Close** to close the **ZyxEditClubView**.

16. Transacted changes in child process ZyxEditMember

In this chapter the connection to the ZyxEditMember process will have transaction characteristics specified in the DPB. Therefore, the changes made in the child view (ZyxEditMemberView) will be transacted. However, there will still be no transaction settings for the parent process ZyxEditClub. Therefore, the transacted changes in the child view will not be shown in the parent view (the changes will be shown if the changes in the child view are commited).

Define default transaction settings for child process ZyxEditMember connection (eMPChConn)

17.1 In the **DPB**: Define **Transaction Connection Transaction Handling** for **eMPChConn** with the following (default) options enabled:

- **Autostart mode**.
- **Use own context**.
- **Transaction context**.

Figure 84: Defining transaction settings for child connection to ZyxEditMember

17.2. Select **Browser / Save all changes**.

17.3. Save the image.

Test

17.4. In the workspace: Select the code from the previous example and **execute with Execute**. The **ZyxEditClubView** dialog appears.

Abort changes to member name in child view

17.5. From the **Drop-down List**: Select **member1**.

17.6. Click **edit**. The **ZyxEditMemberView** dialog appears.

17.7. Change **name** to **member1new**.

17.8. Click on **ZyxEditClubView** dialog (to change the focus). Note that the dialog is NOT updated with the new name, since transaction handling is not defined for ZyxEditClub:

Figure 85: Transacted changes in child are not reflected in parent

17.9. In the **ZyxEditMemberView** dialog: Click **abortAndCloseView**. Note that the changes are aborted in the **ZyxEditClubView** dialog as well:

Figure 86: Changes aborted in child are also aborted in the parent

Commit changes to member name in child view

17.10. From the **Drop-down List**: Select **member1**.

17.11. Click **Edit**. The **ZyxEditMemberView** dialog appears.

17.12. Change **name** to **member1new**.

17.13. Click **commitAndBegin**. Note that the changes are committed in the **ZyxEditClubView** dialog:

Figure 87: Changes committed in the child are committed in the parent

17.14. Click **commitAndCloseView** to close the ZyxEditMember dialog.

17. Display transacted changes in child process in parent process view

In this chapter, transactions (non-isolated) will be specified for parent process ZyxEditClub. Therefore, when uncommitted changes are made in the child view, the changes will be displayed in the parent view.Note: If the transactions for the parent process are specified as isolated, then the changes in the child will not be shown in the parent until they are committed.

Define default transaction settings for parent process ZyxEditClub (eCPConn)

18.1 In the **DPB**: Define **Transaction Main Transaction Handling** for **eCPConn** with the following (default) options enabled:

- **Autostart mode**.
- **Use own context**.
- **Transaction context**.

Figure 88: Defining transaction settings for ZyxEditClub in DPB

18.2. Select **Browser / Save all changes**.

18.3. Save the image.

Test

18.4. In the workspace: Select the code from the previous example and **execute with Execute**. The **ZyxEditClubView** dialog appears.

Display uncommitted changes in child view in parent view

18.5. From the **Drop-down List**: Select **member2**.

18.6. Click **edit**. The **ZyxEditMemberView** dialog appears.

18.7. Change **name** to **member2new**.

18.8. Click on **ZyxEditClubView** dialog (to change the focus). Note that the dialog IS updated with the new name, since transaction handling is defined for ZyxEditClub:

Figure 89: Uncommitted changes in child are displayed in parent

18. Adding and deleting members

In this chapter the ZyxEditClub methods and view buttons will be added for adding and deleting members.

Create ZyxEditClub.newMember(), ZyxEditClub.deleteMember()

19.1 Create the **newMember()** method:

```
public void newMember() {
   ZyxMember m = new ZyxMember();
   m.setName("newMemberName");
   getECBConn().getMembers().add(m);
}
```

19.2. Create the **deleteMember()** method:

```
public void deleteMember() {
   ZyxMember m = getECBConn().getCurrentMember();
   if (m != null) {
      getECBConn().getMembers().remove(m);
   }
}
```

Add new / delete buttons to ZyxEditClubView

19.3. Open **Composition Editor** on **ZyxEditClubView**.

19.4. Add a **JButton** bean to the view.

19.5. Set the bean property **beanName** to **eCPConnXXnewMemberXX**.

19.6. Set the bean property **text** to **New member**.

19.7. Add a **JButton** bean to the view.

19.8. Set the bean property **beanName** to **eCPConnXXdeleteMemberXX**.

19.9. Set the bean property **text** to **Delete selected member**.

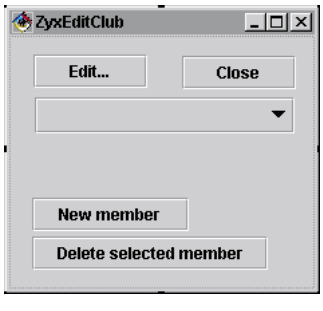


Figure 90: Add and delete buttons in ZyxEditClubView

19.10. **Save the bean**.

19.11. Save the workspace.

Test

19.12. Run **ZyxEditClub.main()**. Use the **Add member** and **Delete selected member** buttons to add and delete members.

19. Implementing Tooltips (using a ViewPort)

In this chapter ***Tooltips*** will be implemented for the ZyxEditMemberView name text field. Tooltips utilize the ***ViewPort*** for DO ZyxMember.

Create method ZyxMemberViewPort.getNameTooltip().

19.13. Create the following method:

```
public String getNameTooltip() {
   return "tooltip for name";
}
```

19.14. Save the method.

19.15. Save the workspace.

Test

19.16. Run **ZyxEditClub.main()**.

19.17. Select a **member**.

19.18. Click **Edit...**. The **ZyxEditMember** dialog appears.

19.19. Move the cursor into the **name text field**. The Tooltip text appears:

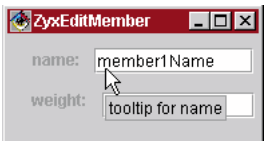


Figure 91: Tooltip text in the dialog

20. Enabling/disabling buttons

In this chapter you will implement enabling/disabling of the **Edit...** button depending on whether or not a member has been selected from the drop-down list.

Create ViewPort subclass ZyxEditClubViewPort

20.1 In **Workbench**: Right-click on **zyx.tutorial**.

20.2. Select **Add / Class...**.

20.3. In the field **Class name:**: Enter **ZyxEditClubViewPort**.

20.4. In the field **Superclass:**: Enter **COM.pmsc.afw.viewPorts.ViewPort**.

20.5. Uncheck the checkbox **Compose the class visually**.

20.6. Click **Finish**.

20.7. Create the method **ZyxEditClubViewPort.getEditEnabled()**.

```
public boolean getEditEnabled() {
  ZyxEditClub p = (ZyxEditClub) getModel();
  return p.getECBConn().getCurrentMember() != null;
}
```

20.8. Save the method.

Register ZyxEditClubViewPort in Model Browser

20.9. In the **Model Browser**: Select **File / Reload**.

20.10. Expand the tree for **ZyxTutorial**.

20.11. Select **ZyxEditClubViewPort**.

20.12. Click on **>>**.

Assign ZyxEditClubViewPort as the ViewPort for eCPConn

20.13. Open **DPB** on **ZyxEditClub**.

20.14. Select **eCPConn**.

20.15. In column **Value** row **Viewport**: From the drop-down list: Select **zyx.tutorial.ZyxEditClubView-Port**.

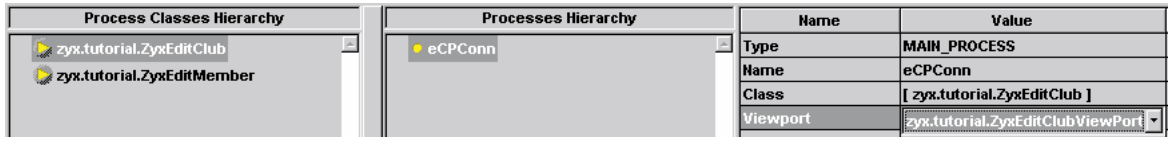


Figure 92: Assigning ZyxEditClubViewPort as the ViewPort for the process connection to ZyxEditClub

20.16. **Save all changes**.

20.17. Save the workspace.

Test

20.18. Run **ZyxEditClub.main()**. Note that the **Edit...** button in **ZyxEditClub** dialog is disabled:

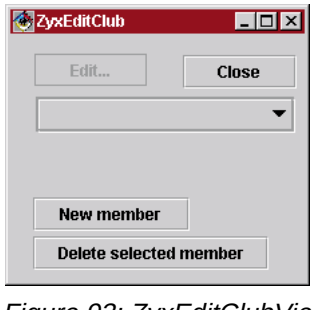


Figure 93: ZyxEditClubView with disabled Edit button

20.19. Select a member. Note that the button is now enabled.

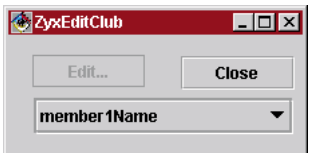


Figure 94: ZyxEditClubView Edit... button enabled after member selected from list

21. Add JList to ZyxEditClubView

In this chapter you will add a JList to ZyxEditClubView. Special methods are required for interfacing with the List.

Add JScrollPane and JList to ZyxEditClubView

20.20. Open **ZyxEditClubView** in the Composition Editor.

20.21. Add a **JScrollPane** to the view.

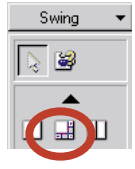


Figure 95: JScrollPane bean in the parts palette

20.22. Add a **JList** bean within the **JScrollPane**.

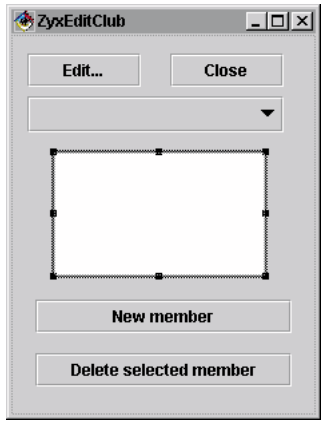


Figure 96: JList bean inside the JScrollPane bean

Note: Click on the **Beans List** icon in toolbar to display the list of beans. The JList and JScrollPane beans can be easily selected by clicking in the beans list. Double-clicking in the beans list will also open the Properties dialog for that bean.

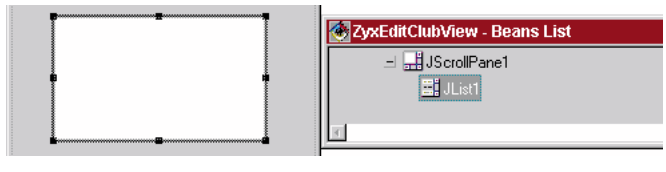


Figure 97: JList properties

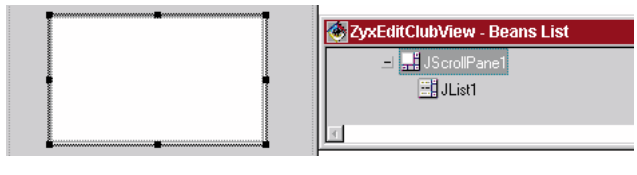


Figure 98: JScrollBar properties

20.23. Change the **JList** property **beanName** to **eCPConnXXmembersXXeCPConnXXselectedMembersXX**.

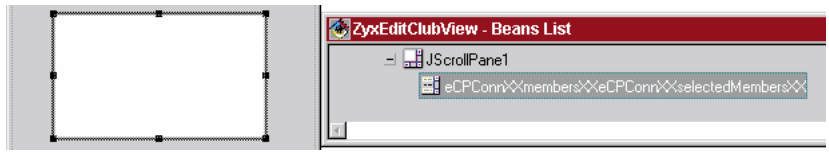


Figure 99: JList beanName

20.24. **Save the bean**.

Changes to ZyxEditClub

20.25. Add the variable definition to **ZyxEditClub**:

```
private AfwList selectedMembers;
```

20.26. Add the method **ZyxEditClub.getSelectedMembers()**:

```
public AfwList getSelectedMembers() {
   if (selectedMembers == null) {
      selectedMembers = new AfwList();
   }
   return selectedMembers;
}
```

20.27. Modify the method **ZyxEditClub.setSelectedMember()**:

```
public void setSelectedMember(ZyxMember member) {
   getECBConn().setCurrentMember(member);
   getSelectedMembers().clear();
   if (member != null) {
      getSelectedMembers().add(member);
   }
}
```

20.28. Add the method **ZyxEditClub.modelsSelected()**:

```
public void modelsSelected(IfSelectionNotification notification) {
   if (notification.getAspectName()
   == MicAtom.forString("selectedMembers")) {
      ZyxMember m = null;
      if (!getSelectedMembers().isEmpty()) {
         m = (ZyxMember) getSelectedMembers().get(0);
      }
      getECBConn().setCurrentMember(m);
   }
}
```

20.29. Save the image.

Test

20.30. Run **ZyxEditClub.main()**. The following dialog appears.

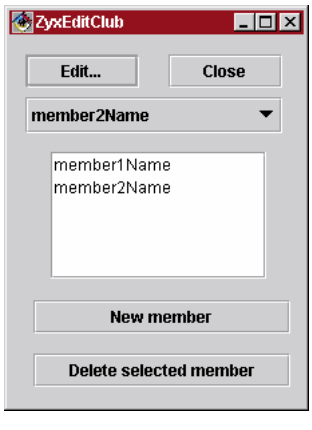


Figure 100: ZyxEditClubView with new ScrollPane/List

20.31. Choose a member from either the drop-down list or the list. Note that the member is selected in the other list.

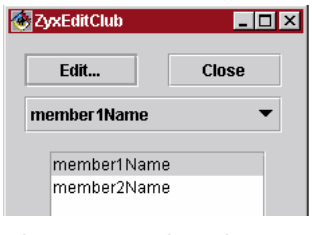


Figure 101: Choosing a member in one list selects the member in the other list

22. Controlling visibility of a GroupControl (with a ViewPort)

In this chapter you will add a **JPanel** to the ZyxEditMemberView that displays the weight deviation (from the "ideal" weight range) of the selected member (for simplicity, it is assumed that all members should have a weight from 60-80 kilos). The JPanel is only displayed if the members weight is <60 or >80 kilos.

Add JPanel to ZyxEditMemberView

21.1 Add a **JPanel** bean to **ZyxEditMemberView**.
22.1 Change the **beanName** to **eMPConnXXweightIsDeviantXX**.
23.1 Add a **JLabel** bean to **ZyxEditMemberView**.
24.1 Change the **text** to **Member weight deviation:**.
25.1 Add a **JTextField** bean to **ZyxEditMemberView**.
26.1 Change the **beanName** to **eMBConnXXweightDeviationXX**.

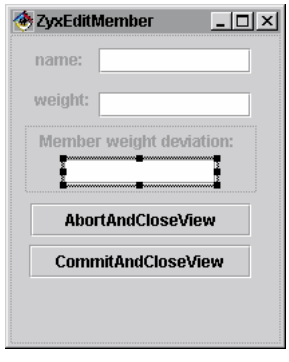


Figure 102: ZyxEditMemberView with the JPanel and contents

26.2. Save the bean.

Create ZyxMember.getWeightDeviation()

26.3. Create the method:

```
public float getWeightDeviation()  {
  if (getWeight() < 60) {
    return (getWeight() - 60);
  }
  if (getWeight() > 80) {
    return(getWeight() - 80);
  }
  return 0;
}
```

Create ZyxEditMember.isWeightDeviant()

26.4. Create the method:

```
public boolean isWeightDeviant() {
  return (getEMBConn().getWeightDeviation() != 0);
}
```

Create ZyxEditMemberViewPort.getWeightIsDeviantVisible()

26.5. Create the method:

```
public boolean getWeightIsDeviantVisible() {
  return !((ZyxEditMember) getModel()).isWeightDeviant();
}
```

26.6. Save the workspace.

Test

26.7. Run **ZyxEditMember.main()**.
26.8. Enter a member **weight** of **50**.

26.9. Click in any other field. Note that the weight deviation information is displayed:

Figure 103: Weight deviation displayed if weight deviant

26.10. Enter a member **weight** of **60**.

26.11. in any other field. Note that the weight deviation information is NOT displayed:

Figure 104: Weight deviation NOT displayed if weight NOT deviant

List Of Tables

List Of Figures

Index

R

S

T

U

V

W

Z