990516TTA: READ THIS FIRST:

about this doc

- this is an internal version of the frameworks for java getting started guide.
- comments appreciated! i could especially use some good **step-by-step** examples. contact:

  terry taylor, TTA@PMSMICADO.COM, 02242-871-308
- doc created with adobe framemaker.
- the goal of this doc is to present a step-by-step intro into frameworks for java that demonstrates the basic functionality. the idea is that someone can learn frameworks with minimual assistance if he has:
  - computer with NT, DB2 and IBM VisualAge Java
  - the frameworks cd rom
  - this doc
- this doc will eventually reference a detailed programming guide (or something like that). there will NOT be separate docs for afw, pfw, obf.
- the format of this doc attempts to mimic ibm smalltalk manuals.
- these notes (underline) will not appear in the final pdf doc.

versions

V02 990518TTA:

- small corrections to installation.
- some more text written for tutorial.

V01 990516TTA:

- source copied from fw_gs_v07.
- formats, text, etc changed as needed. follows installation and getting started steps as written by LHE.
- the installation is written for installing from ntsrv1. this will be changed eventually.
- text for smalltalk version marked with strike through (for example: ~~text from smalltalk manual~~).

pics and drawings

- screenshots: imported eps files. bmps taken with hypersnap. bmp plaziert in illustrator. colors changed to cmyk. reduced 50%. saved as eps.
- drawings: imported eps files. created in illustrator.
- 3d diagrams. drawn in dimensions. saved as .dim. imported into illustrator?

colors used

- black underlined: comment only.
- blue or green: regular text (for pdf).
- red: example text (for pdf and ex_obf.txt)
- pink: example text (for ex_obf.txt only)

start of doc

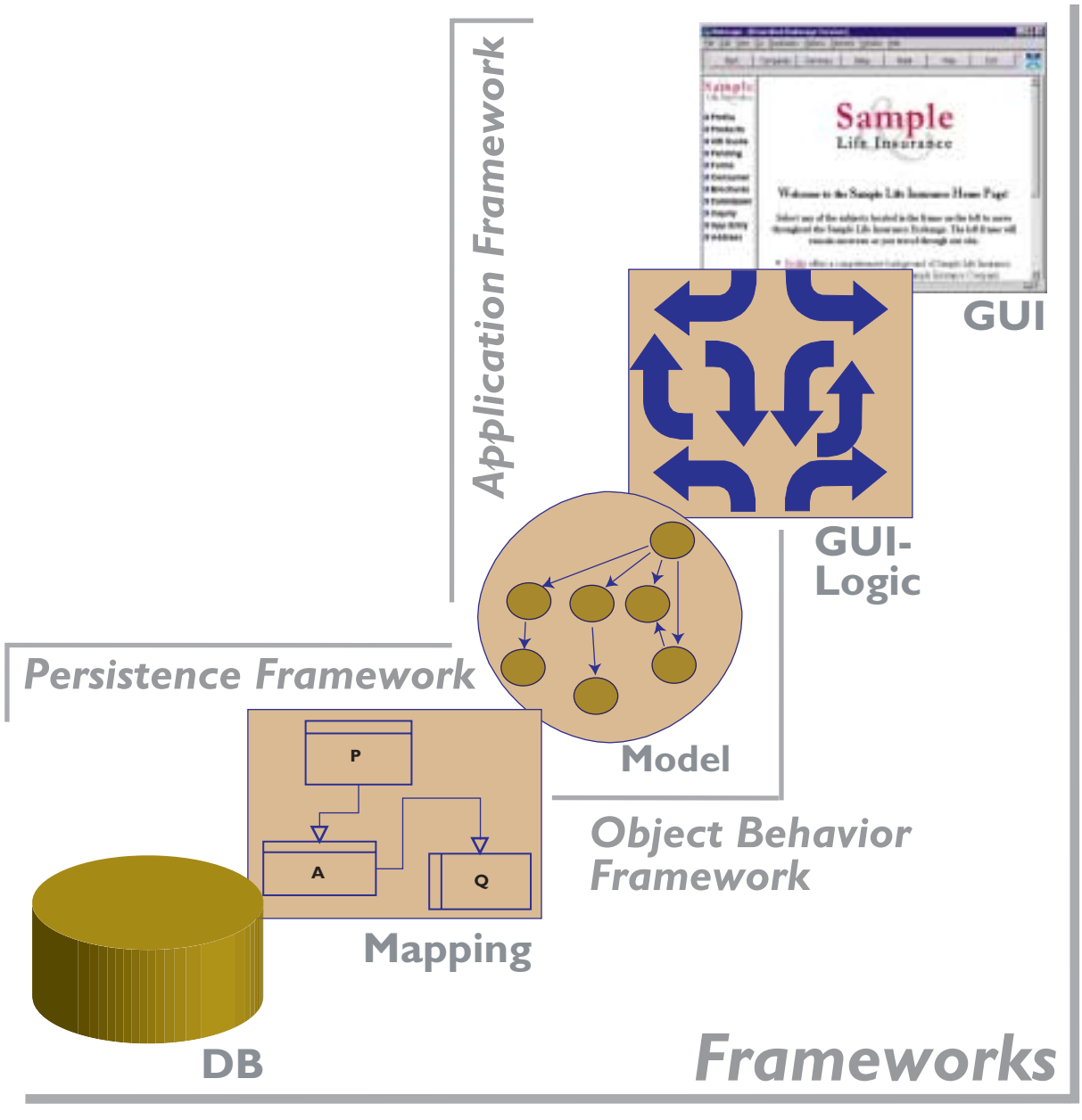
PMS MICADO

Frameworks *for Java*

Getting Started

For Frameworks Release 3.4 Version 1.1

990515TTA??: what kind of image for java? "db logic" changed to "mapping" (RST requrest) image303



Copyright and trademarks

990512TTA: updated

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

990512TTA: other trademarks??

ENVY is a registered trademark of the Object Technology International corporation.

Visual Age for Java and *OS/2* are registered trademarks of the International Business Machines Corp.

Windows and *Windows/NT* are registered trademarks of the Microsoft Corp.

Table of Contents

Documentation Overview

990512TTA: last change.

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)

990418TTA: some changes made.

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.

Content

This manual provides the following:

- Step-by-step tutorial that demonstrates the capabilities of Frameworks.
- API (Application Programming Interface) reference.
- Troubleshooting guide.
- Frequently-asked questions (FAQ) reference.
- Glossary of Framework terms.

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.
- **'API' (page 63).** Lists the Frameworks API messages used in the step-by-step tutorial.
- **'Trouble Shooting Guide' (page 65).** Provides answers to frequently-asked questions about topics covered in the step-by-step tutorial.
- **'Glossary' (page 73).** Includes any special terms used throughout this manual.
- **'List Of Figures' (page 85)**.
- **'Index' (page 89)**.

Recommendations for completing the step-by-step tutorial

990418TTA: added.

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 which doc?? 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

990423TTA: updated.

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

Other sources of information

WWW sites??

Installing Frameworks

990518TTA: some corrections added.

990514TTA: added text about changing nls specification for dpb and ome.

990512TTA: description for installation from ntsrv1 at micado. i will write for an installation from a cd later.

990514TTA: There will eventually be an automated installation procedure. However, the present installation procedure is described. I had trouble with the installation; therefore, the installation procedure described here may have some errors. If you find an error in this description, please contact me.

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 (2.0 Rollup 2 is available via notes on micado10 / archive / Java archiv / Visual Age 2.0 Rollup 2. Download the file rollup2_vaj20_win.zip, unzip, and follow the instructions in readme.txt).
- PMS Micado **Frameworks for Java R3.4 V1.1**.

Installation Overview (via the server)

990512TTA: internal installation using server files... the installation from an installation CD i will write about later.

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**.104

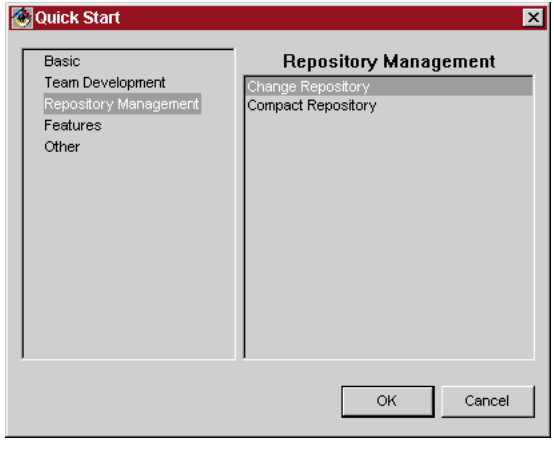


Figure 1: Change repository dialog

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**.103

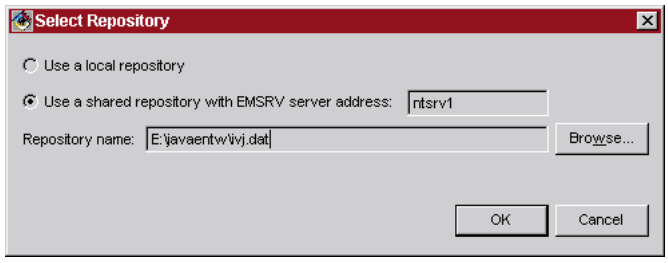


Figure 2: Respository specification

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**.101

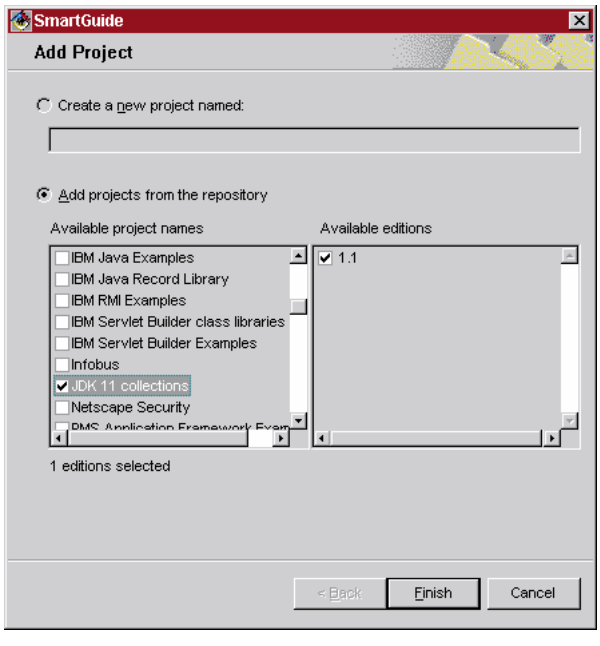


Figure 3: Add project dialog

15. Click **Finish**.

Add IBM IDE Utility class libraries

990518TTA: "IDE" added.

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:102

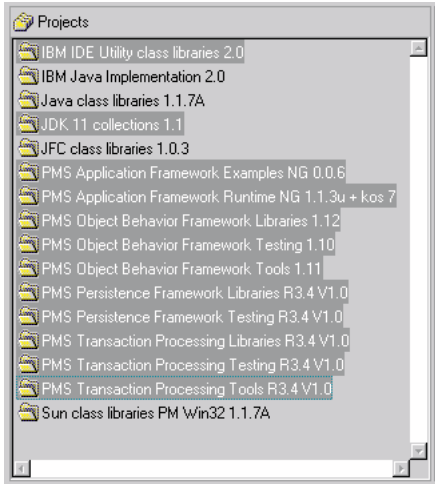


Figure 4: Added projects

Copying the tools to your IBM Java Tools directory

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

990518TTA: "MICOFw" added

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.105

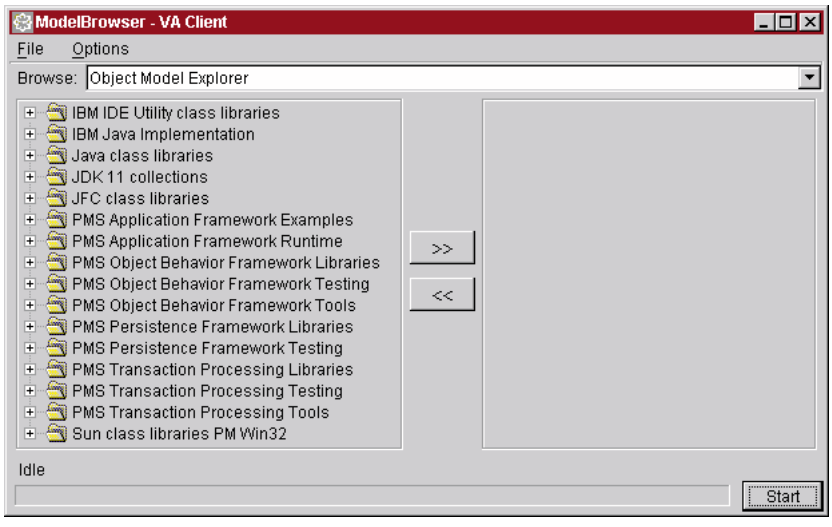


Figure 5: Model Browser - VA Client dialog

26. In the **Model Browser - VA Client**: Select **Options / Set Browser**. The **Options** dialog opens:106

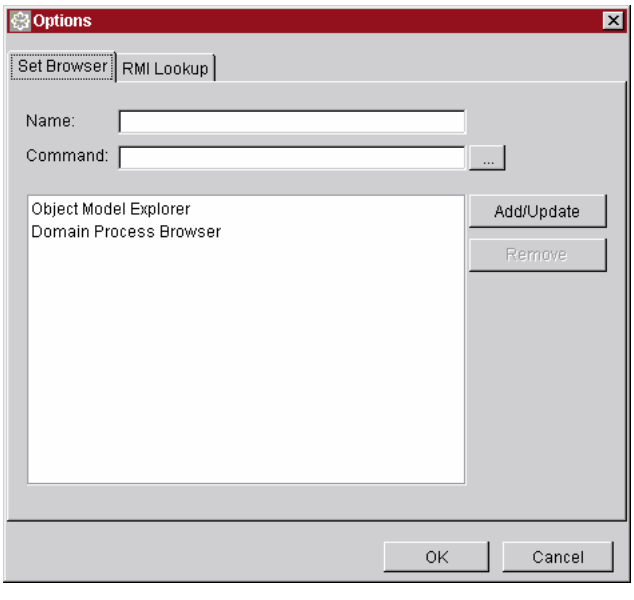


Figure 6: Model Browser options dialog

27. Click on **Object Model Explorer**.
28. Click on the **directory button** for **Command**. The **open File...** dialog appears:107

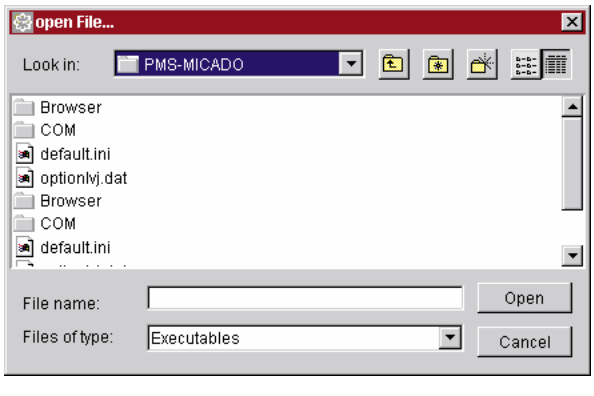


Figure 7: Open file dialog

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:108

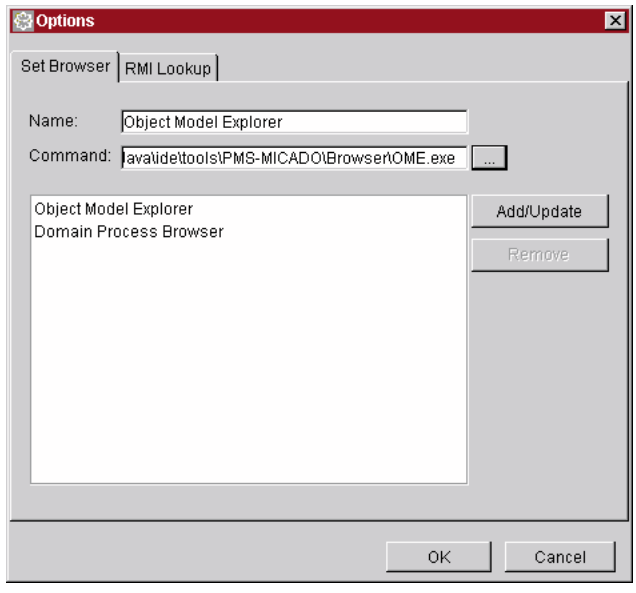


Figure 8: Name and Command information for the OME

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.109

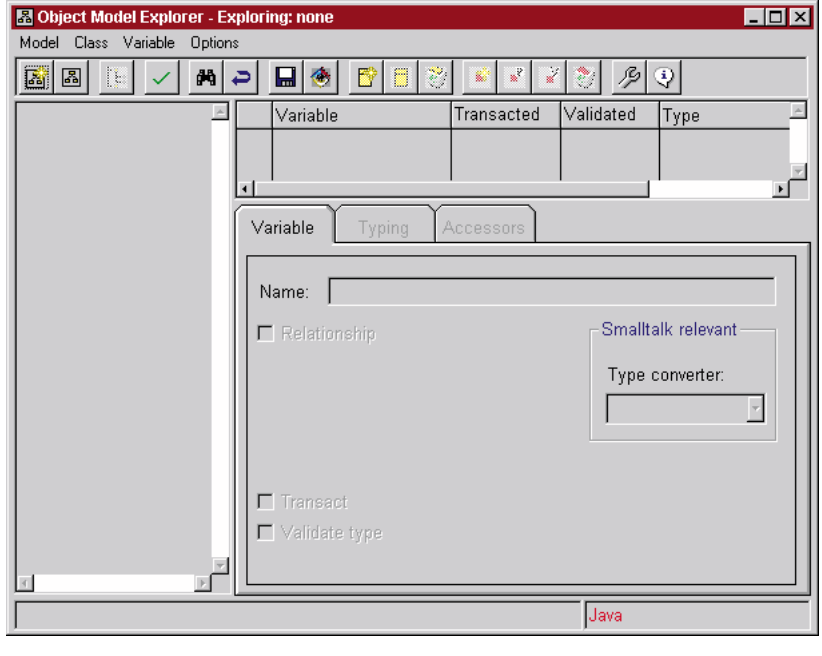


Figure 9: OME dialog

35. Select **Options / Preferences**. The **Options** dialog appears:110

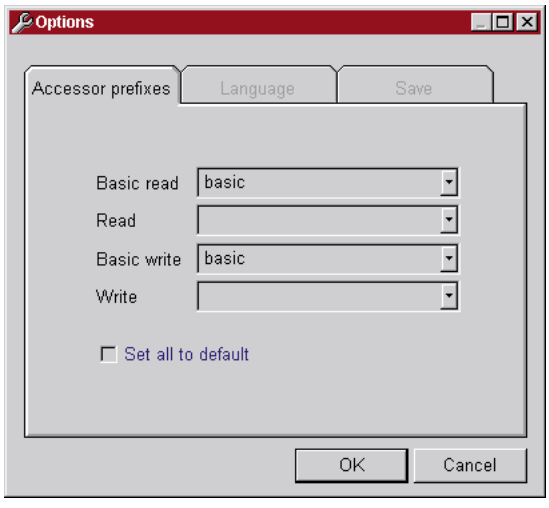


Figure 10: OME Preferences dialog

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**.111

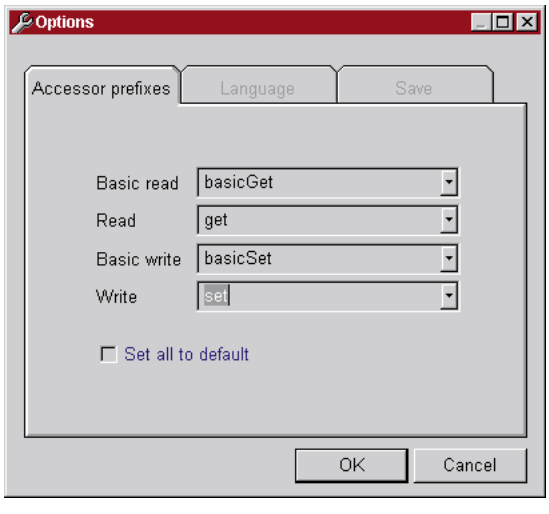


Figure 11: Accessor prefix settings

37. Select **OK**.

Tutorial

990518TTA: installation written from LHE doc.

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. 112

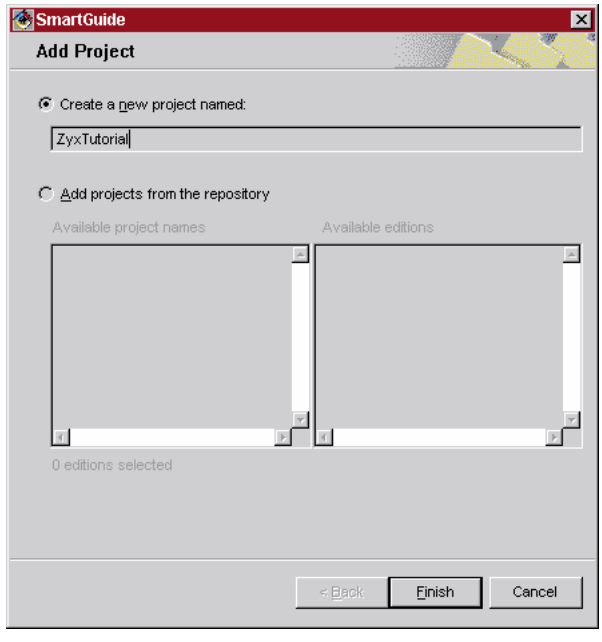


Figure 12: Add Project Smart Guide

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.113

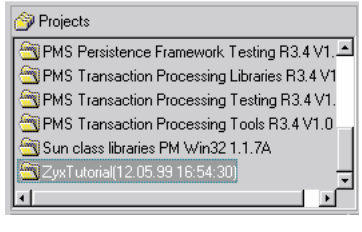


Figure 13: ZyxTutorial in the project list

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.114

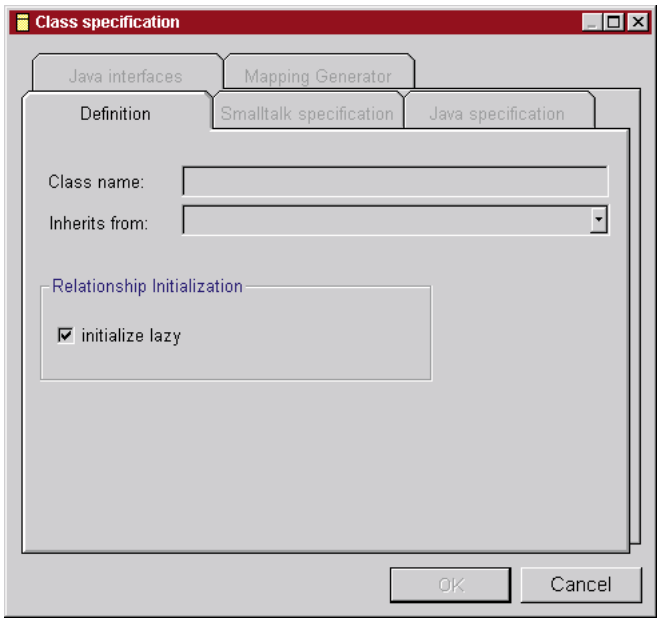


Figure 14: Class specification dialog

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

3.7. From the **Inherits from:** drop-down list: Select **Domain Object**.115

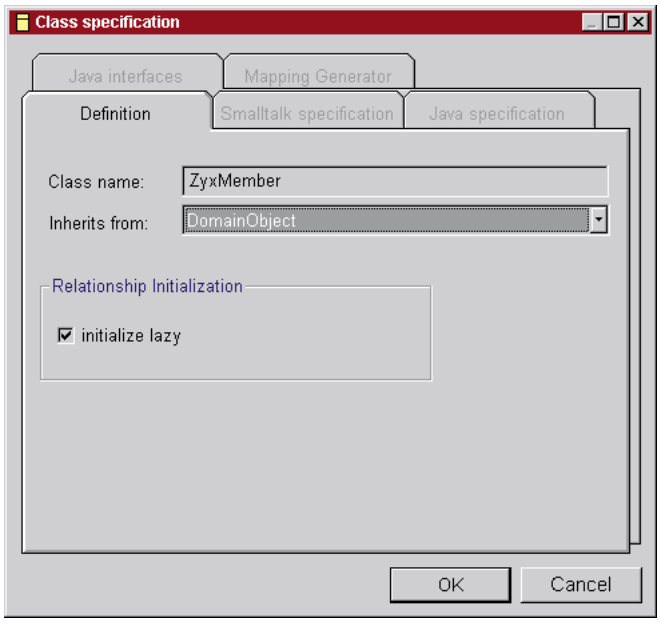


Figure 15: Class specification for ZyxMember

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

3.9. In the field **Package name:**: Enter **zyx.tutorial**.116

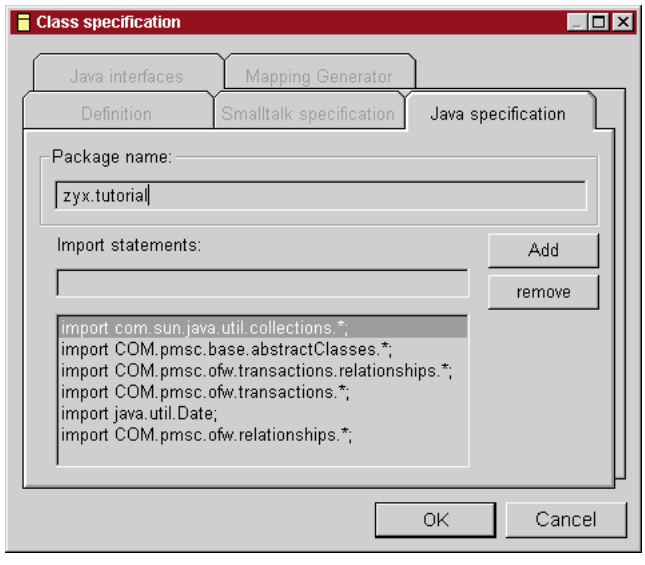


Figure 16: Java specification for ZyxMember

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**.117??

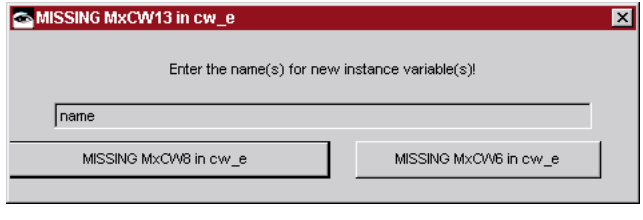


Figure 17: OME Add New Variable dialog

3.14. Click **OK**. The variable name appears in the OME.118

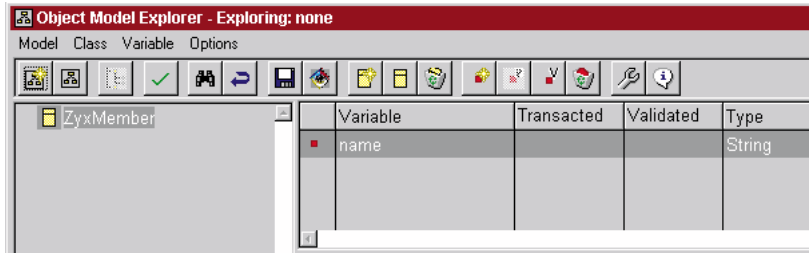


Figure 18: Variable <name> in OME

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**.119

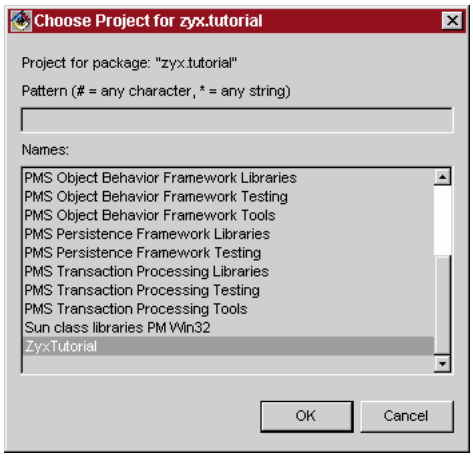


Figure 19: Dialog <Choose Project for zyx.tutorial>

3.17. Select **OK**. The class has been saved to VA.122

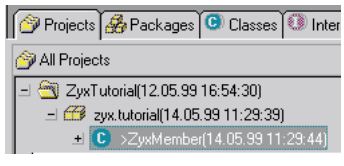


Figure 20: Class ZyxMember in the Workbench projects list

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**.120

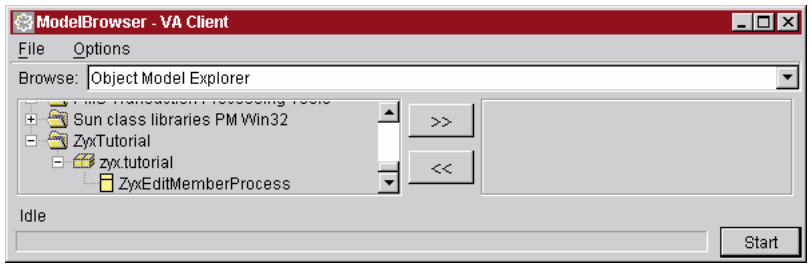


Figure 21: Model Browser dialog

3.29. Select **ZyxMember**.

3.30. Click on **>>**.121

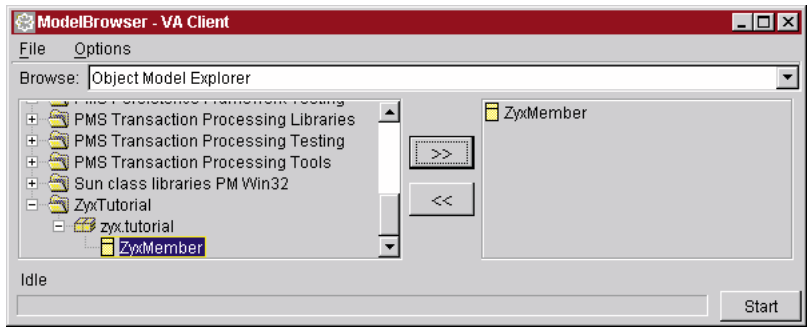


Figure 22: ZyxMember in the Model Browser class list

Open OME

3.31. In the **Model Browser**: Click **Start**. The **OME** dialog appears with **ZyxMember** in the class list.123

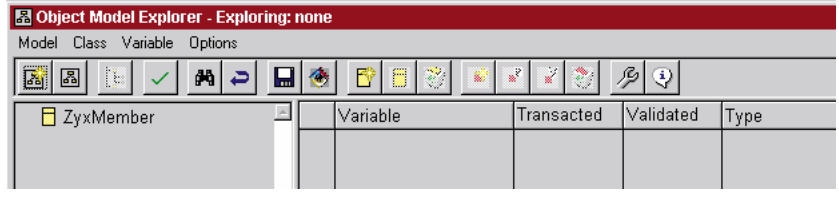


Figure 23: OME with ZyxMember

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**.125

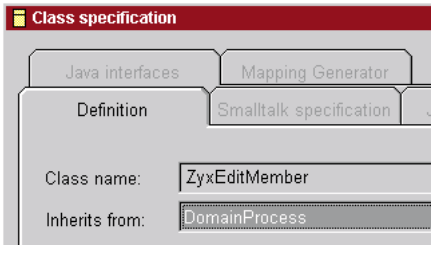


Figure 24: Class specification for ZyxEditMember

4.5. Select the tab **Java specification**. Note that the package name is already set.126

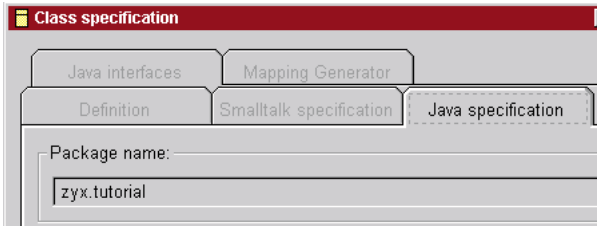


Figure 25: Java specification for ZyxEditMember

4.6. Click **OK**. **ZyxEditMember** appears in the **OME**.127

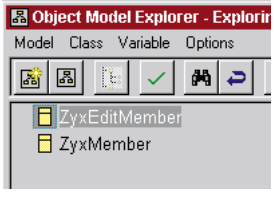


Figure 26: ZyxEditMember in the OME

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 **>>**.128

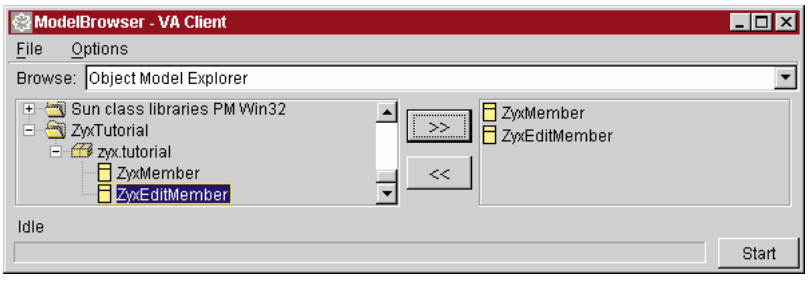


Figure 27: ZyxEditMember in the Model Browser class list

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:129

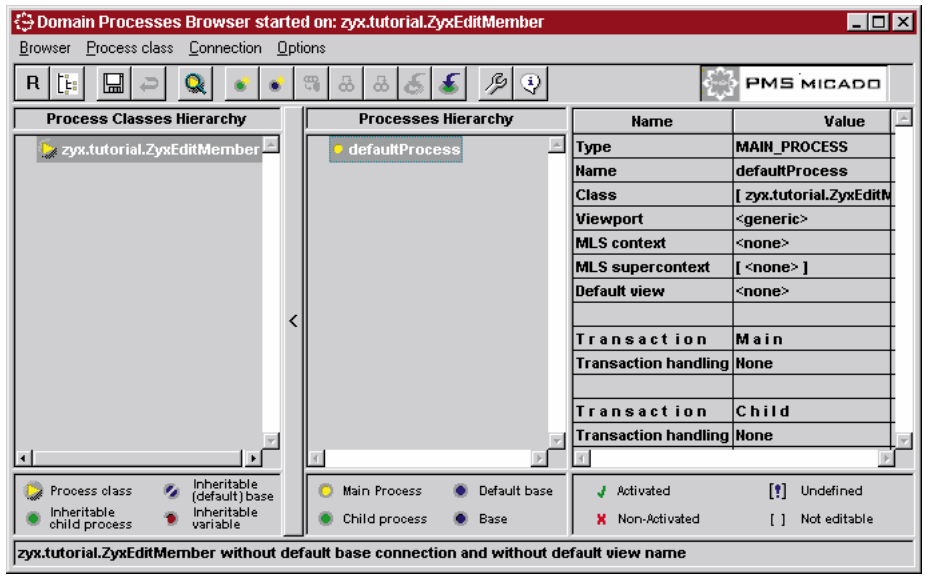


Figure 28: Domain Processes Browser on ZyxEditMember

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**.130

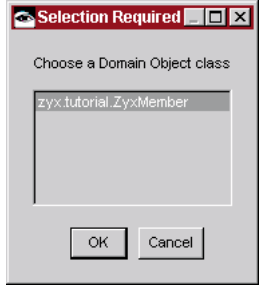


Figure 29: Dialog for selecting ZyxMember as the domain object class for ZyxEditMember

4.19. Click **OK**. Note that the DP ZyxEditMember now has a connection to ZyxMember and that the connection is named **newDefaultBaseConnection**:131

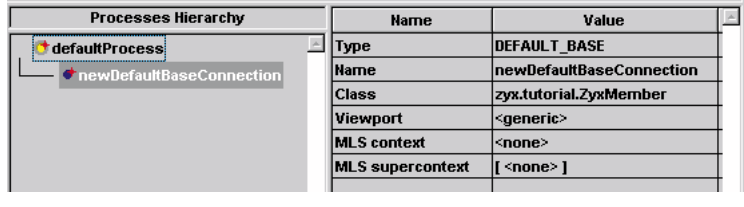


Figure 30: Base connection from ZyxEditMember to ZyxMember in DPB

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**.132

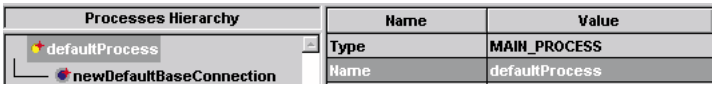


Figure 31: Default name for ZyxEditMember process connection in DPB

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.134

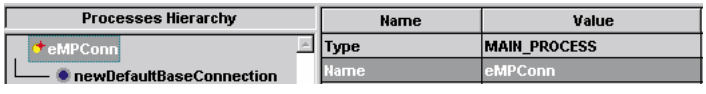


Figure 32: Renaming ZyxEditMember connection to eMPConn in DPB

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**.135

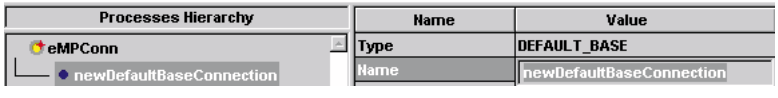


Figure 33: Default name for ZyxMember base connection in DPB

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.136

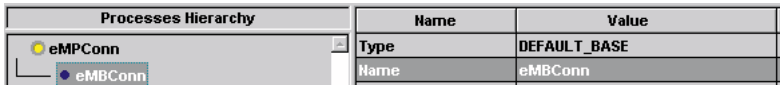


Figure 34: Renaming ZyxMember connection to eMBConn in DPB

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.137

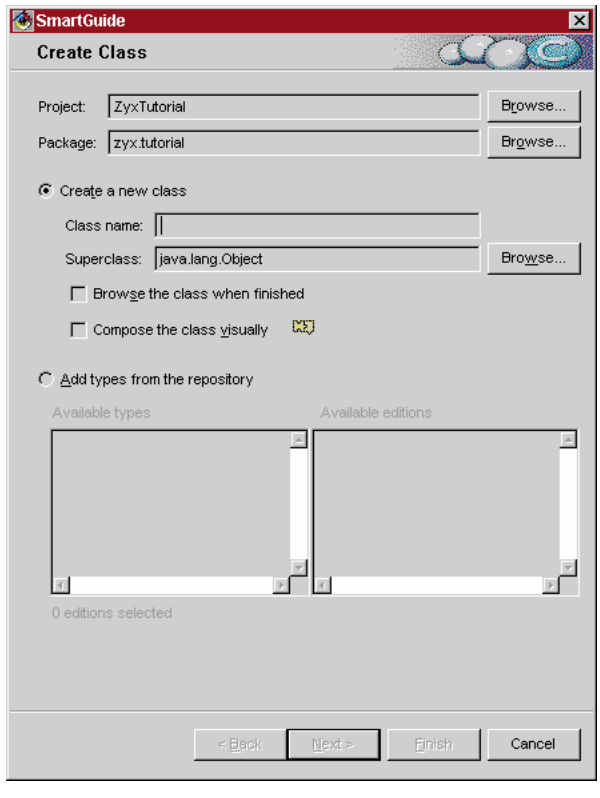


Figure 35: Smart Guide Create Class

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**.138

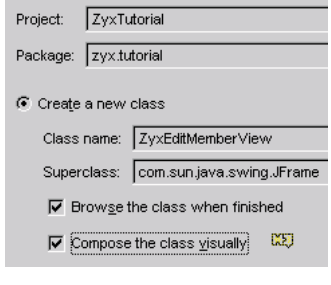


Figure 36: Create Class settings for ZyxEditMemberView

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**.139

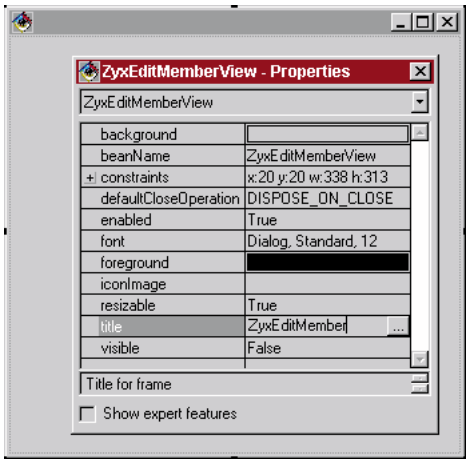


Figure 37: Change title property of ZyxEditMemberView

5.12. Add a **JLabel** bean to the view.140a

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.140b

Figure 39: Selecting JTextField from the parts palette

141

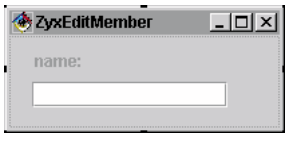


Figure 40: ZyxEditMemberView with label and text field

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 **>>**.142

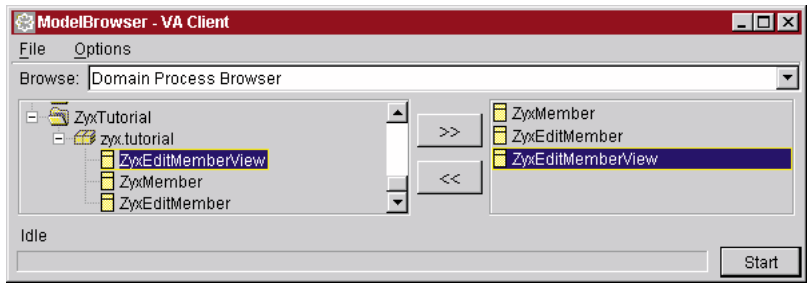


Figure 41: ZyxEditMemberView in the Model Browser class list

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**.143

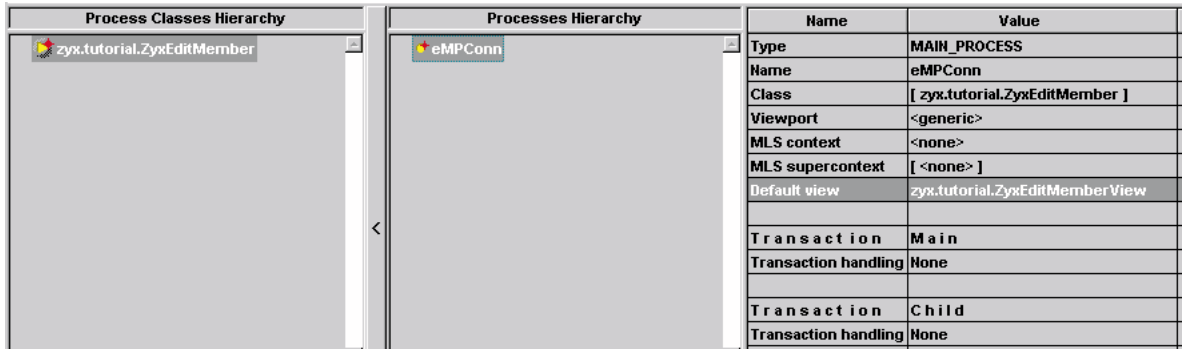


Figure 42: Assigning ZyxEditMemberView to ZyxEditMember in DPB

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**.144

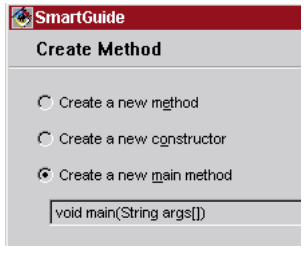


Figure 43: Smart Guide Create Method

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.145??

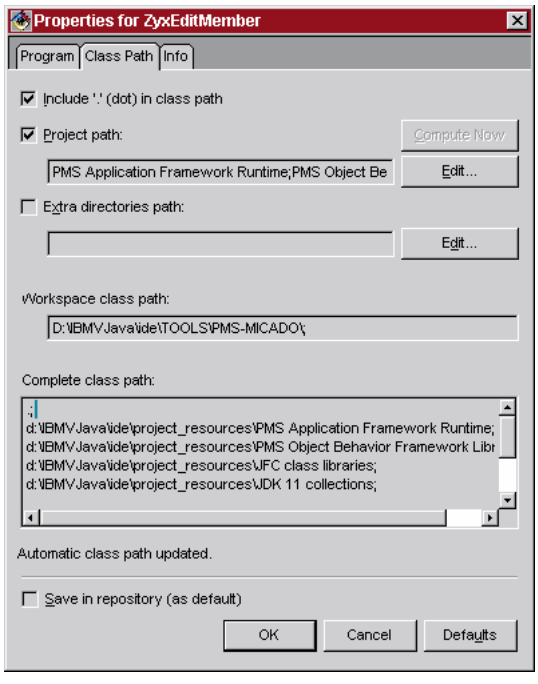


Figure 44: Properties for ZyxEditMember tab Class path

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:146

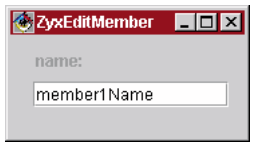


Figure 45: Display of DO ZyxMember attributes in View ZyxEditMemberView

6. Multiple non-transacted views

990514TTA??!!: KOS: when i first tried this it didnt work. i then changed to transacted and still didnt work. no transaction were generated. then i tried again and magically it started. ask me about details.

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):147

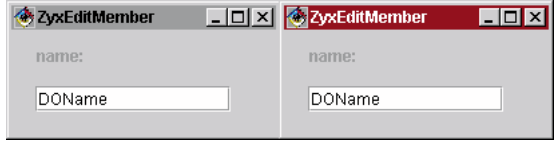


Figure 46: 2 views of same object.

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.148

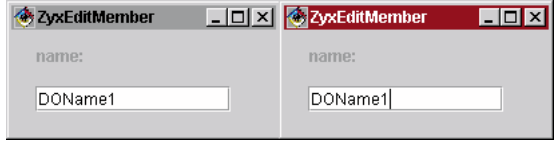


Figure 47: Non-transacted changes in a view are reflected in other views

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.149

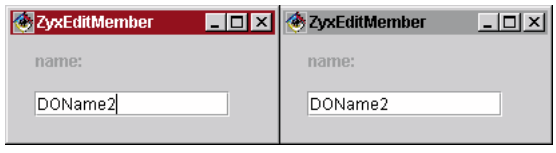


Figure 48: Non-transacted changes to the same object can be made in multiple view dialogs

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.151

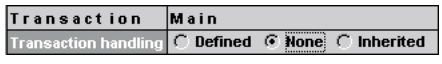


Figure 49: Setting the Transaction Main for ZyxEditMember

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:150

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**:152

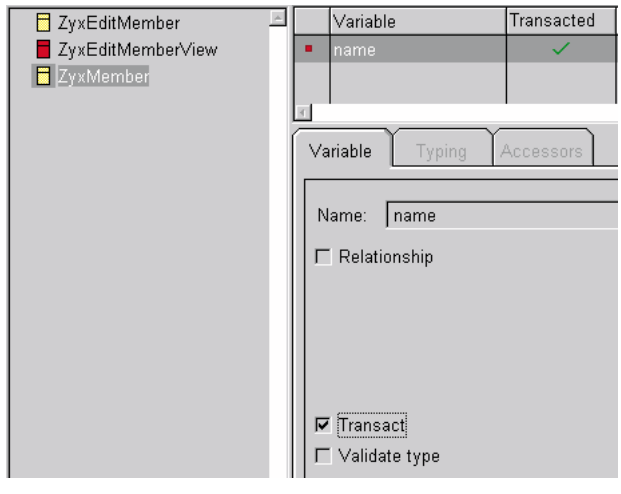


Figure 51: Specifying ZyxMember>>name as transacted in OME

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:153

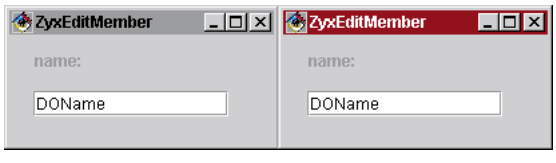


Figure 52: 2 views of same object.

154

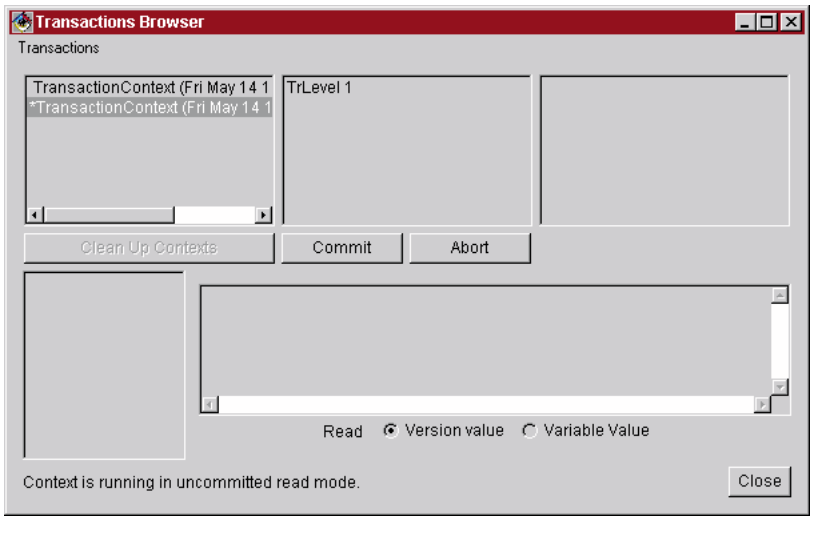


Figure 53: Transaction Browser dialog

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.155

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.156

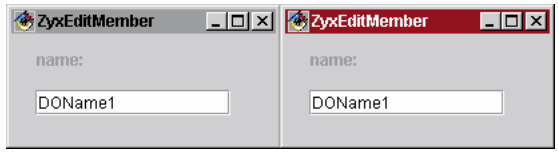


Figure 55: The second view of an object is updated after transacted changes in the first view

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.157

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.158

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.159

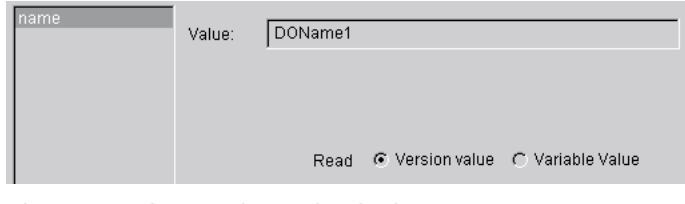


Figure 57: The Version Value in the TB

7.25. Select ***Variable value***. The displayed variable value is **'DOName'**.160

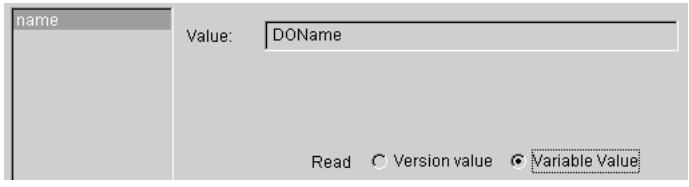


Figure 58: The Variable Value in the TB

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:161

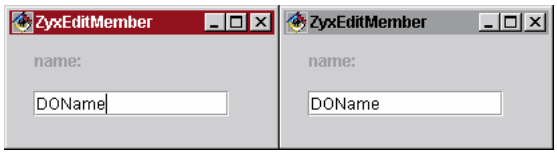


Figure 59: Aborted changes as reflected in the views

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**.162

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.163

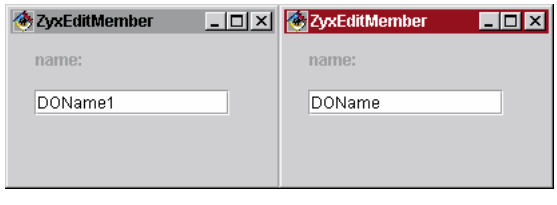


Figure 61: Isolated transacted changes in multiple views

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

990515TTA: eventually add what happens in TB

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.164

Figure 62: JButton bean in the parts palette

165

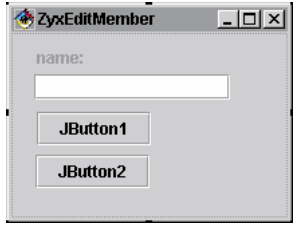


Figure 63: ZyxEditMemberView with 2 buttons

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**.166

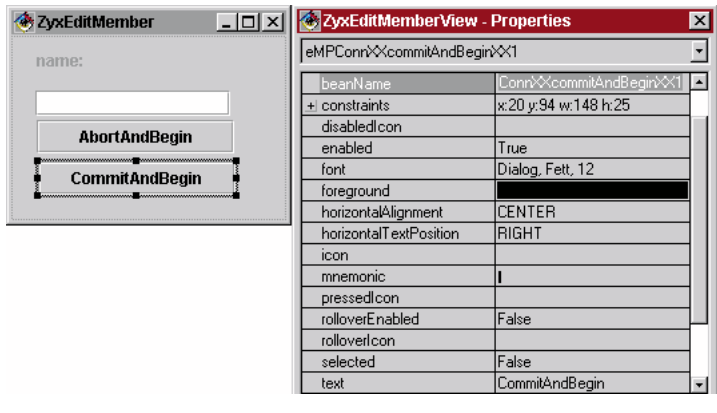


Figure 64: CommitAndBegin button settings

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**.167

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

990515TTA: eventually add what happens in TB

The context that is assigned to the dialog can be aborted/committed from the dialog with the ...AndCloseView() 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**.168

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**.169

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

990518TTA: changes from LHE. a lot of the section from "adding tooltips" was moved here.

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**.170??

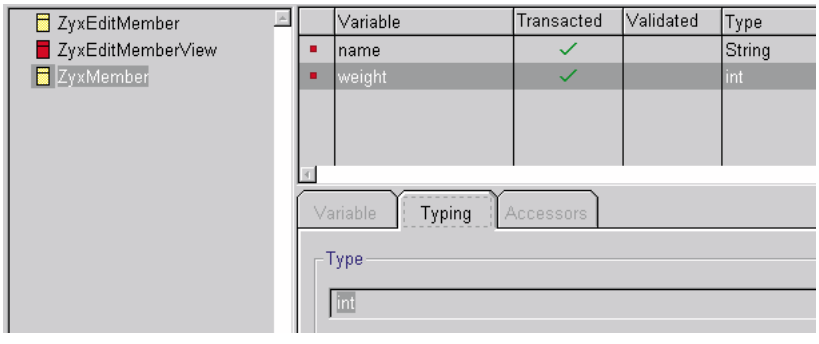


Figure 68: Variable ZyxMember>>weight in OME

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**.171

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**.183??

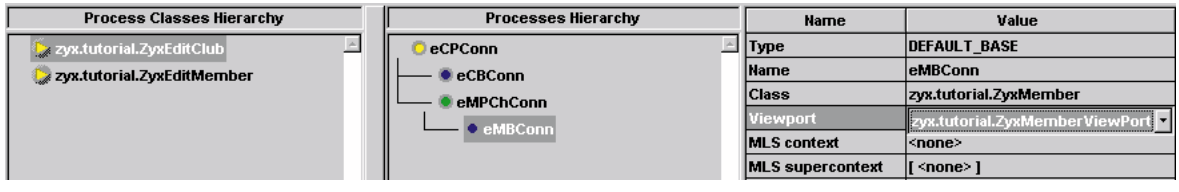


Figure 70: Assigning ZyxMemberViewPort as the ViewPort for the base connection to ZyxMember

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:197

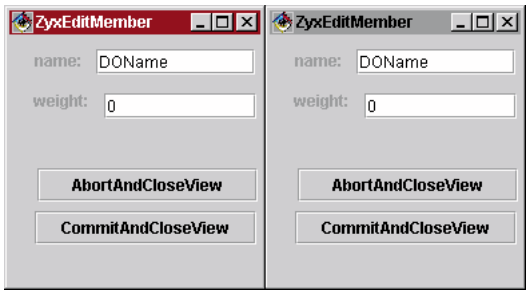


Figure 71: 2 ZyxEditMember dialogs with weight field

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):198

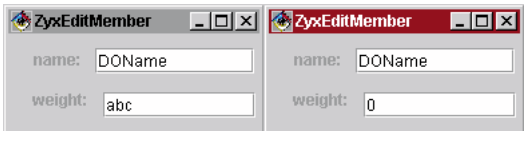


Figure 72: Value of invalid type in LEFT dialog not displayed in RIGHT dialog

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:199

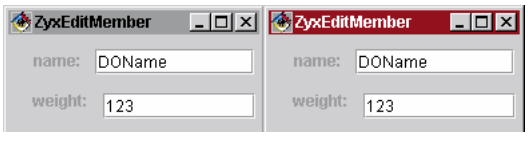


Figure 73: Value of valid type in LEFT dialog is displayed in RIGHT dialog

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**.172

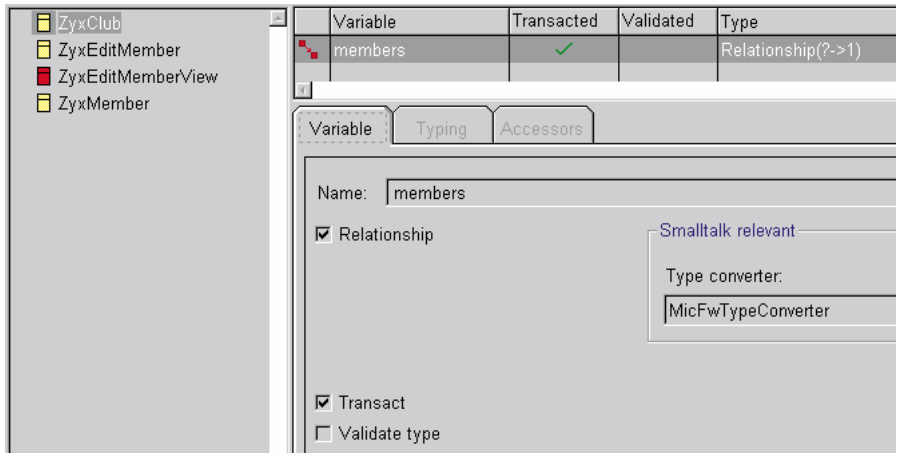


Figure 74: Specifying a relationship in OME

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**.173

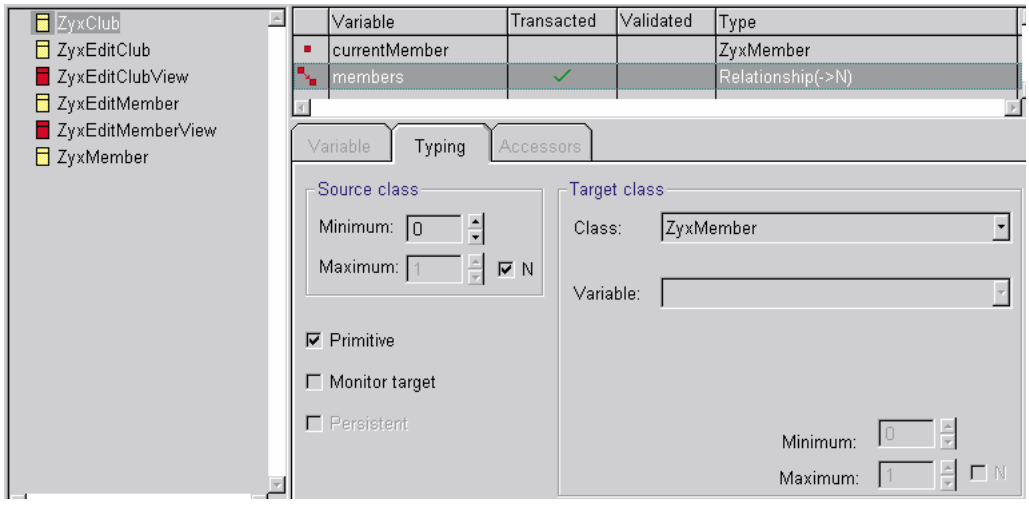


Figure 75: Specifying target class and cardinality in OME

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**.174

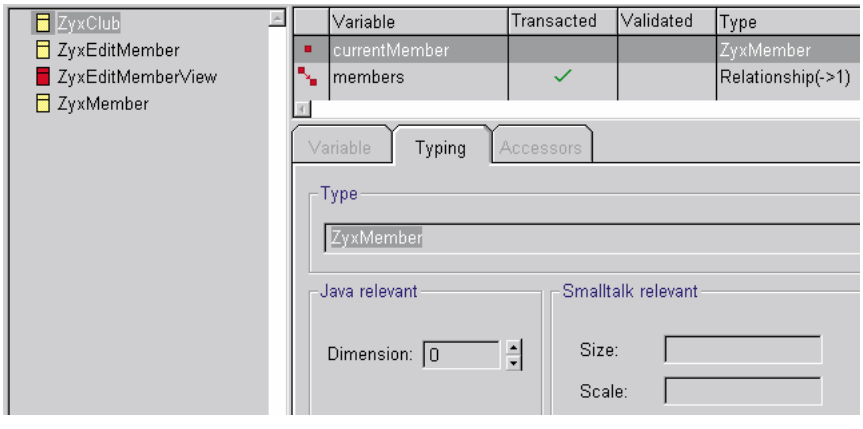


Figure 76: Typing ZyxClub>>currentMember in OME

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

990515TTA??: change this section for ZyxEditMember. also, move registration of ZyxClub to previous section.

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 **>>**.175

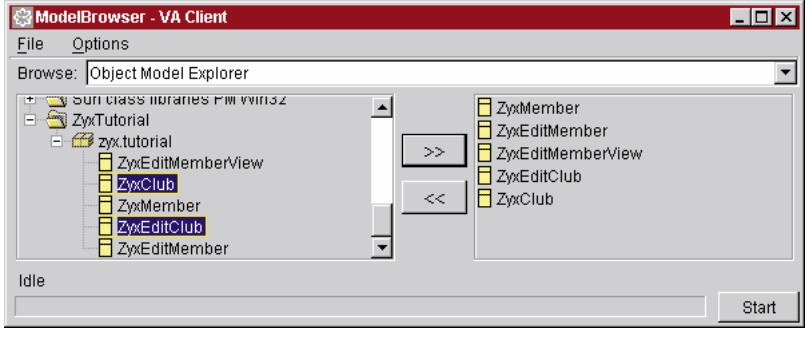


Figure 77: ZyxEditClub, ZyxClub in Model Browser class list

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.176

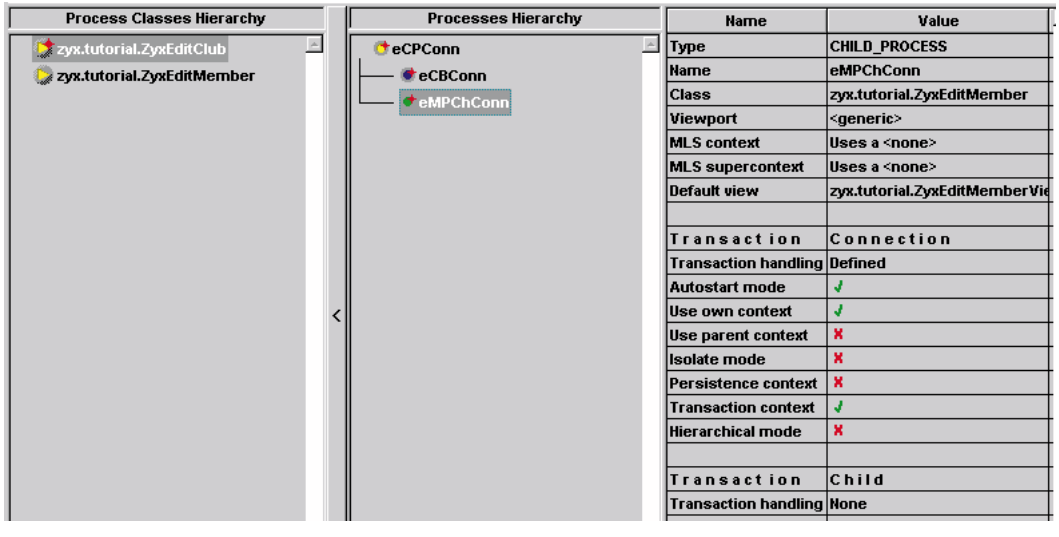


Figure 78: Renaming connections in DPB

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 **ZyxEditClubView - 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**.177

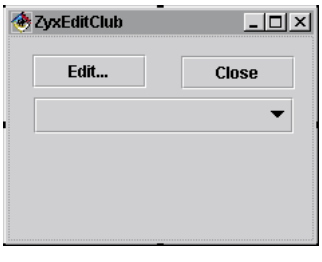


Figure 79: ZyxEditClub Close button

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 **>>**.178

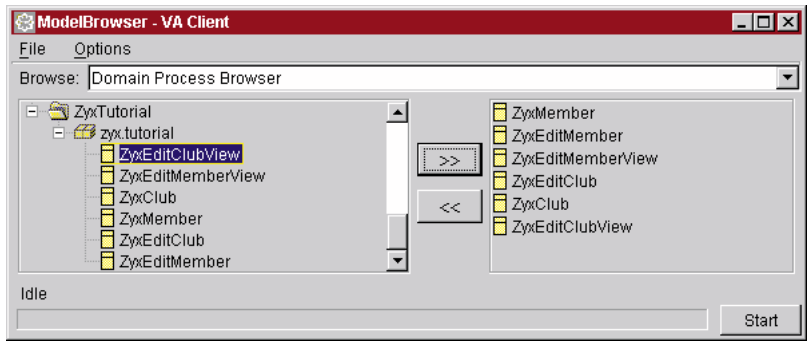


Figure 80: ZyxEditClubView in Model Browser Class list

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

990516TTA: eventually add the TB here, showing how the trlevels and contexts change as changes are made.

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:179

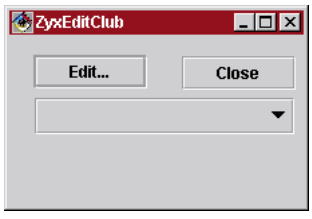


Figure 81: ZyxEditClubView dialog

16.34. Select a member1 from the drop-down list:180

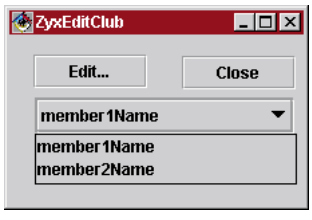


Figure 82: Selecting a member from the ZyxEditClubView drop-down list

16.35. Click **Edit...**. The **ZyxEditMemberView** appears.181

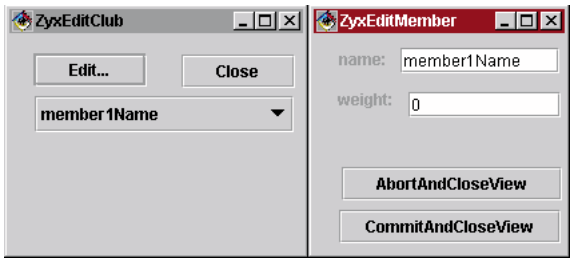


Figure 83: ZyxEditMemberView as a child of ZyxEditClubView

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

16. Transacted changes in child process ZyxEditMember

990516TTA old smalltalk version.

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**.379

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:427

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:428

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:429

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

990516TTA old smalltalk version.

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**.378

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:438

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**.182

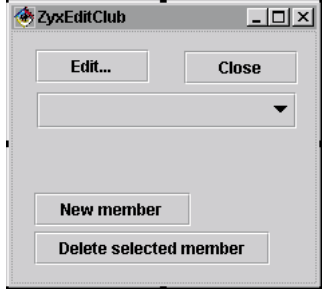


Figure 90: Add and delete buttons in ZyxEditClubView

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)

990518TTA: much of this section moved to validating input section

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:184

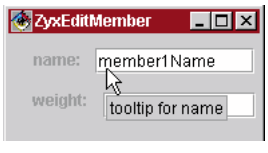


Figure 91: Tooltip text in the dialog

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**.185

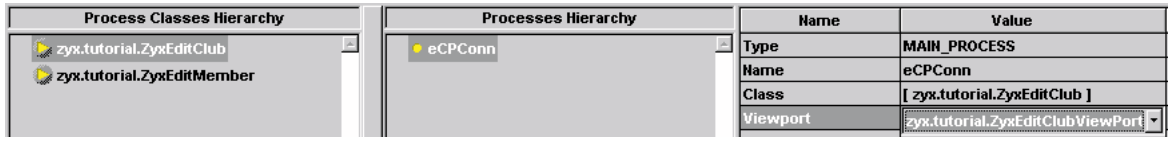


Figure 92: Assigning ZyxEditClubViewPort as the ViewPort for the process connection to ZyxEditClub

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:186

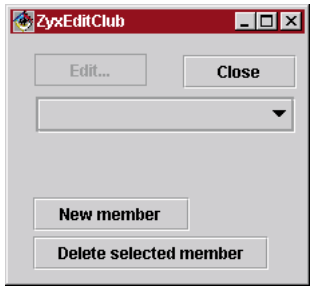


Figure 93: ZyxEditClubView with disabled Edit button

20.19. 990516TTA?? button is not enabled. if a breakpoint is set in getEditEnabled(), the breakpoint is reached when view initialized... however, not reached after a member is selected from the list.Select a mem-

ber. Note that the button is now enabled.187??

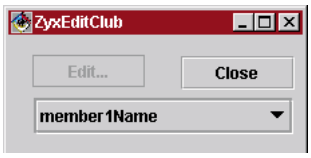


Figure 94: ZyxEditClubView Edit... button enabled after member selected from list

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.188

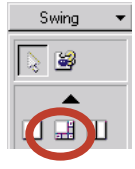


Figure 95: JScrollPane bean in the parts palette

20.22. Add a **JList** bean within the **JScrollPane**.190

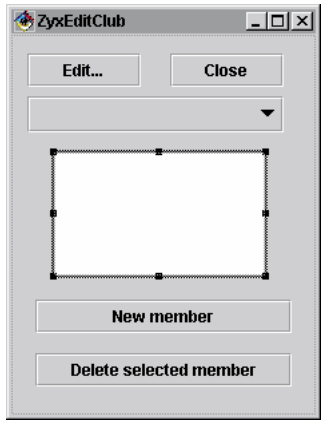


Figure 96: JList bean inside the JScrollPane bean

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.192

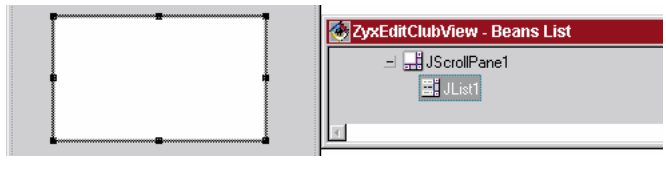


Figure 97: JList properties

193

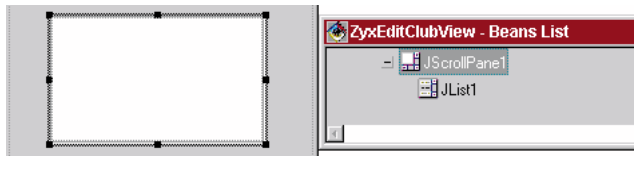


Figure 98: JScrollBar properties

20.23. Change the **JList** property **beanName** to **eCPConnXXmembersXXeCPConnXXselectedMembersXX**.194

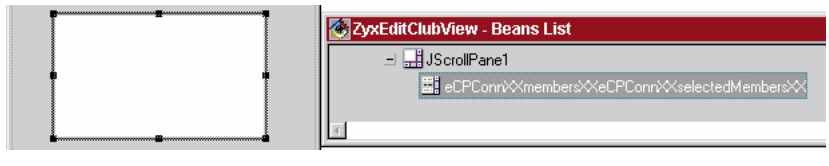


Figure 99: JList beanName

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.195

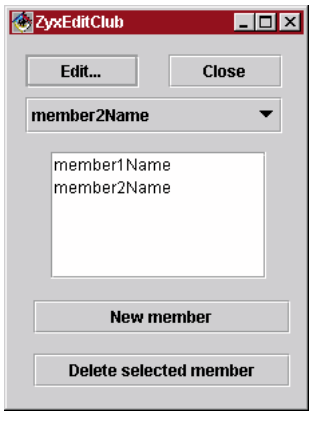


Figure 100: ZyxEditClubView with new ScrollPane/List

20.31. Choose a member from either the drop-down list or the list. Note that the member is selected in the other list.196

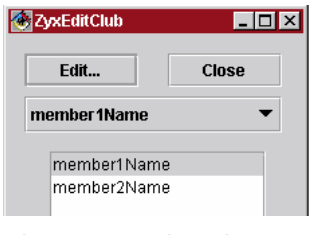


Figure 101: Choosing a member in one list selects the member in the other list

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

990518TTA: i wrote this example by myself, so i'm not sure if correct

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**.200

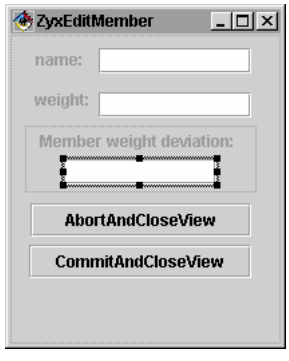


Figure 102: ZyxEditMemberView with the JPanel and contents

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:201

Figure 103: Weight deviation displayed if weight deviant

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

26.11. 990518TTA??: this shoud work (i think), but not implemented yetClick in any other field. Note that the weight deviation information is NOT displayed:202??

Figure 104: Weight deviation NOT displayed if weight NOT deviant

Appendix A

API

990512TTA: this will be a reformatted version of the auto-matically created html files.

Appendix B

Trouble Shooting Guide

990512TTA: modified for java. this is an INTERNAL version of the TS appendix.

- a SINGLE file is used as the source for all 5 docs (fwj, fw, afw, pfw, obf).
- the J. F, A, P, O columns below indicates which doc an entry belongs in (Fw java, Fw, Afw, Pfw, Obf).
- this table will be converted to text when a RELEASE VERSION is created. a table is used because 1 file is used for all docs.

This section recommends trouble-shooting approaches to solving common problems when using the Frameworks.

J	F	A	P	O	area of trouble	trouble	solution
J	F	A	P	O	Application	**TS 1. Prerequisite that I want to assign to an application is not available from the list of prerequisites**	The required configuration map must be loaded. For example, in order to assign MicFwPersistenceOdbc as a prerequisite for an application, the configuration map micPersistenceOdbc Runtime and micPersistenceOdbc Development must be loaded.
J	F	A	P	O	Application	**TS 2. Prerequisite that I want to assign to an application is not available from the list of prerequisites**	The required configuration map must be loaded. For example, in order to assign MicFwPersistenceOdbc as a prerequisite for an application, the configuration map micPersistenceOdbc Runtime and micPersistenceOdbc Development must be loaded.
J	F	A	P	O	DB2	**TS 3. DB2 hangs up when trying to display the Beispielinhalt**	The POM may still be connected to the database. Open the STOPF for the POM (from the **System Transcript** select **micFrameworks / Browse Persistence Manager**, select the POM) and disconnect the POM from the database (**Manager / Disconnect from database**).
J	F	A	P	O	Domain Object (DO)	**TS 4. DO doesnt appear in the Tear-off pop-up window for the DM**	The DP must first be assigned to the DM by double-clicking on the DM and entering the DP class in the dialog. AND the DO must be specified in a base connection for the DP (using the DPB).
J	F	A	P	O	Domain Process (DP)	**TS 5. DP doesnt appear in the Tear-off pop-up window for the DM**	The DP must first be assigned to the DM by double-clicking on the DM and entering the DP class in the dialog.
J	F	A	P	O	Domain Processes Browser (DPB)	**TS 6. Can't save a change that I just made in the DPB... The Save option is greyed out**	Click anywhere else in the DPB (so that the focus changes). The Save option should appear.
J	F	A	P	O	Domain Processes Browser (DPB)	**TS 7. A newly created DP, DO, Viewport, View, etc. does not appear in a drop-down list in the DPB**	Try closing and then reopening the DPB.

J	F	A	P	O	area of trouble	trouble	solution
J	F	A	P	O	Envy debugger messages	**TS 8. "MicFwTransactionContext is not running"**	If a transaction context is aborted within the TB, then the dialog that the context was assigned to cannot be closed. Normally contexts should be aborted or committed by clicking a button in the dialog.
J	F	A	P	O	Envy debugger messages	**TS 9. "Invalid view. A default view is not set for a process"**	A default view was not set for the process. This can sometimes happen because the a change in the DPB was not saved. To solve this problem: • Close the DPB. • Open the DPB. • Reassign the view to the process. • Save the changes.
J	F	A	P	O	Object Net Browser (ONB)	**TS 10. The changes I made in the ONB did not take effect**	Always select **Class / Save** after making any changes in the ONB. NO changes are automatically saved.
J	F	A	P	O	Transactions	**TS 11. Changes to variable are not transacted**	If a variable has not been specified as a transacted variable (by checking the checkbox "Transacted" in the ONB) then changes to the varible will not be transacted.
J	F	A	P	O	Transaction Browser (TB)	**TS 12. MicFwTechnicalTransactionContext's are displayed in the TB.**	MicFwTechnicalTransactionContext is a context that is created when you are using certain Frameworks tools (such as the Domain Processes Browser). These contexts are for internal use only. Do not alter these contexts (ie, commit, abort, etc.) in any way.
J	F	A	P	O	Visual Age Organizer	**TS 13. A menu item is not shown**	VA Organizer has an option to display Full or Basic menus. To display Full Menus: From the VA Organizer: Select **Options / Full Menus**.

Appendix C

Frequently-asked questions

990512TTA: modified for java.

- a SINGLE file is used as the source for all 5 docs (fwj, fw, afw, pfw, obf).
- the J, F, A, P, O columns below indicate which doc an entry belongs in (Fw java, Fw, Afw, Pfw, Obf).
- this table will be converted to text when a RELEASE VERSION is created. a table is used because 1 file is used for all docs.

990510TTA: small errors fixed.

This section provides the answers to the most frequently-asked questions about Frameworks.

J	F	A	P	O	area of use	question	answer
		A			———	**FAQ 1. How Do I Connect a Container Widget, Like a Form or a Notebook Page to the Framework?**	The accessor name of such widgets has 4 parts. The first two parts are used to get the content of the container or set Subaspects like "visible" or "enabled". So, if you want to set the visible state of a Form, you have to use the second part of the forms accessor name. If you write a method to set the content of such widget, this method must return a *MicFwMetaPartCollection*, which contains *MicFwMetaParts* (see 'Metapart Collections' (page 153)). Now for the last two parts of the accessor name: They are used to connect the widgets inside the container to a special Base or Process Connection. It's important to understand, that all the Connection Names must be seen relativ to the surrounding container. If you connect a Form A to a child process A and connect a Form B inside Form A to a child process B, process B must be a child process of process A. A Form should be reusable and because of this, widgets inside the Form shouldn't access Connections directly. You should always try to use "defaultBase", "defaultProcess" or a capitalized dummy name like "VOID". As mentioned above, you can use the third and fourth part of the accessor name of the surrounding Form to exchange the "VOID" names with the Connection Names you want to..

		A			———	**FAQ 2. How Do I Get Informed When a Notebook Page is About to be Left or About to be Entered?**	You should use
							`>> enterInteractionForAspect:comingFromAspect:of:`
							and / or
							`>> leaveInteractionForAspect:goingToAspect:of:`
							These methods can also be used to prevent the change. (See 'Dynamic Notebooks' (page 157)).

		A			———	**FAQ 3. Notebook Labels - What is Important!**	The label of a Notebook Page is set by the Subaspect "tabLabel"

J	F	A	P	O	area of use	question	answer
		A			———	**FAQ 4. Dynamic Notebook Pages; Before You Use them...**	If you want to use a dynamic Notebook Pages, you should first study 'Dynamic Notebooks' (page 157), which explains how the Framework inserts dynamic Notebook Pages before, between or after static Notebook Pages and how the attributes positionHint and tabType of *MicFwMetaPart* work.
		A			———	**FAQ 5. Why Should You Avoid Adding MicFwMetaPartCollection While You Work with Dynamically Exchanged Forms?**	You should work with the visible Subaspect and have only one visible at a time! This approach is supported by the new Framework behavior, which is to have model access only for visible widgets.
		A			———	**FAQ 6. Combo Box Widget; Before You Use it...**	We recommend strongly that you first read the explanation of our 7 part accessor name first (See 'Combo Box' (page 95)).
		A			———	**FAQ 7. Transactions; Before You Use them...**	(We recommend strongly that you first read the object behavior documentation and learn to understand how the "default" and "main" – transaction handling page inside the Domain Processes Browser is used. Don't forget to check if the initialization of your objects takes place inside or outside a transaction. Study the hooks, the Domain Process offers during its own initalization and while closing (See 'Transaction Behavior' (page 70)).
		A			———	**FAQ 8. Hierachical List; Before You Use them...**	If you want to use a hierachical list, you should first understand, what the content of such list consists of and how to write the *~Children* and *~HasChildren* methods to get the children of the list objects.

Appendix D

Glossary

990512TTA: column added for fw java. i would assume that many of the glossary terms will be similar.
990512TTA: this is an INTERNAL version of the GLOSSARY.

- a SINGLE file is used as the source for all 5 docs (fwj, fw, afw, pfw, obf).
- the J, F, A, P, O columns below indicates which doc an entry belongs in (Fw Java, Fw, Afw, Pfw, Obf).
- the table in the RELEASE version will be reformatted.

990505TTA: terms with no definition defined as kommentar.

This glossary defines terms that are presented throughout this manual.

For more general terms relating to Smalltalk, databases, and OOP: Refer to the following sources of information:

- free online dictionary of computing at http://wombat.doc.ic.ac.uk/foldoc/index.html

J	F	A	P	O	term and definition
					Base Connection:
					Child process:
					Connections Browser:
					DM: Domain Manager
					DO: Domain Object
					Domain Manager (DM):
					Domain Object (DO):
					Domain Process (DP):
					Domain Processes Browser (DPB):
					DP: Domain Process.
					F1 Help:
					Hover Help:
					Name part:
					Naming convention:
					partName:
					Persistent Object Manager (POM):
					Persistent Object:
					POM class:
					POM Generator:
					Process Hierarchy:
					Quick form:
					Schema:
					Tear off (a part):
					validation (of type):
					ViewPort:
J	F	A	P	O	**->1 relationship:** In ->1 relationships, a source instance variable references 0..1 target objects (nil or 1 object). No target instance variable refers to the source object. An example of a ->1 relationship is the relationship between Person (source) and PersonName (target). Person instance variable name references 1 PersonName object (a person has only 1 name). No PersonName instance variable references the Person object (a PersonName object never needs to know which Person object is referencing it).

J	F	A	P	O	term and definition
J	F	A	P	O	**->N relationship:** 990217TTA updated In ->N relationships, a source instance variable references min...max target objects (where min...max is specified by the cardinality). No target instance variable refers to the source object. An example of a ->N relationship would be the relationship between Person (source) and PersonName (target), with the assumption that a person can have more than 1 name. Person instance variable name references min...max PersonName objects, where min = 1 and max is unspecified. No PersonName instance variable references the Person object.
J	F	A	P	O	**1<->1 relationship:** In 1<->1 relationships, a source instance variable references 0..1 (signified by the "1" on the RIGHT of "1<->1") target object. The target instance variable references the same 1 (signified by the "1" on the LEFT of "1<->1") source object or nil. The relationship is not primitive, because the target object should have a reference to the source object. An example of a 1<->1 relationship would be the relationship between Customer (source) and Portfolio (target). Customer instance variable portfolio references 1 Portfolio object. Portfolio instance variable customer references the 1 Customer object. The relationship is not primitive, because a Portfolio object should know which object is its Customer.
J	F	A	P	O	**1<->N relationship:** In 1<->N relationships, a source instance variable references N (where N is any number with the range max...min specified by the cardinality of the relationship) target objects. There is an instance variable in all of the referenced target objects that reference the 1 source object. An example of a 1<->N relationship would be the relationship between Employee (source) and Customer (target). Employee instance variable ownedCustomers would reference N (cardinality min <= N <= cardinality max) Customer objects. The instance variable ownerEmployee in each referenced Customer object would reference the 1 Employee object.
J	F	A	P	O	**Abort a context:** A context is aborted when all version objects within the context are dereferenced (ie, none of the changes that were transacted while the context was active will actually be implemented) and the context ceases to exist. See: abortcontext.
J	F	A	P	O	**Abstract Control::** An Abstract Control simplifies external access to the Framework, as such access can only take place via real Control. There are only a small number of genuinely different Abstract Controls (see command example in the MVC chapter). The actual access attempts are handled with real Control via an appropriate Adapter. Focus change, issuing commands and transfer of data - including a range of state information like validation or authorization - will be done in this abstract layer. The real Controls are completely decoupled from all Domain Process actions and Domain Model data; all technical details of external interfaces (GUI, DDE, ...) are completely hidden in the Abstract Control implementation.

J	F	A	P	O	term and definition
J	F	A	P	O	**Abstract Event:** Abstract Window Events are the objects that really perform the communication between the view system and the model world, whereas Abstract Windows are merely containers for Abstract Events which additionally may provide some services for them. Abstract events can be divided into two functional types: Abstract events which propagate changes in the model world to the view and Abstract Events which propagate changes or requests from the user interface to the model world.
J	F	A	P	O	**Abstract Value:** An Abstract Value is a container object that is used by Viewports to keep and propagate information about a Viewport Aspect to and from the view. It does not only contain a value for the content of a control, but also different kinds of state information like whether or not the control should be enabled, what the (background) color is, which context help text is displayed, and so on.
J	F	A	P	O	**Abstract View:** When a real View is created, an Abstract View begins to exist in its shadow. The lifetime of the Abstract View is exactly determined by the lifetime of the real View. The Abstract View's purpose is to manage the Abstract Controls corresponding one-to-one to the real Controls on the view. Moreover, the Abstract View performs coordination between abstract and platform layer when opening, activating and closing the view. It communicates with the real Platform View using a Platform Adapter.
J	F	A	P	O	**Accessor Generator:** The accessor generator is the object that creates the OBF accessors for a variable.
J	F	A	P	O	**Active Context:** Only 1 context can be active at anytime. While a context is active, any changes to any transacted variables will be recorded in object versions. This variable will be locked by the context, which means that no changes may be made to the variable while a sibling context or parent context is active as long as this context exists.
J	F	A	P	O	**Adapter:** A Platform Adapter system is introduced which does the translation of protocols and overcomes the architectural differences between the host Smalltalk system architecture and the Application Framework architecture. This makes the core part of the framework itself portable.
J	F	A	P	O	**Archiver:** See: Code Archiver.
J	F	A	P	O	**Authorization:** The access to individual objects controlled on the model level. Authorization works both for accessing attributes of Domain Objects and for executing Aspects of business processes.
J	F	A	P	O	**Broker:** To allow subsystems or specific requests to them to be exchanged with an own implementation, Application Framework uses Broker classes that offer a thin public interface with the internal knowledge how to delegate the call to the subsystem. A common Broker concept enables the developers to modify request algorithms or subsystem behavior quite easy.
J	F	A	P	O	**Cardinality:** The cardinality of a relationship determines the required min and max number of target objects referenced by the source variable. In 2-way relationships, there are 2 cardinalities. The second cardinality determines the requried min and max number of source objects referenced by the target variable.
J	F	A	P	O	**CB:** Connections Browser.

J	F	A	P	O	term and definition
J	F	A	P	O	**Child context:** A child context can change any variable locked by its parent. A variable, having been changed by the child context, is now locked by the child context. The parent context cannot change a variable locked by the child. See: parent Context.
J	F	A	P	O	**Code Generator:** See Accessor Generator.
J	F	A	P	O	**Commit a context:** 990217TTA updated Committing a context has the same effect as committing all transaction levels (TrLevels) in the context.
J	F	A	P	O	**Committed target:** 990217TTA updated In a VersionObject: A getter message to a source object will return the committed target object referenced by the variable if no context is active OR if [the active context read mode is isolate AND the source object's variable is not locked by the active context AND the active context is not a child context of the context with the variable lock]. See: transacted Target.
J	F	A	P	O	**Concurrent contexts:** 990217TTA updated 2 contexts are concurrent if there is no parent-child relationship between them. Such contexts can also be referred to as "sibling" contexts. A sibling context may not change a variable locked by another sibling context.990217TTA DIRK: as i understand, a "grandchild" and a "child"of a parent context are also sibling contexts??
J	F	A	P	O	**Concurrent transactions:** See Concurrent Contexts.
J	F	A	P	O	**Connector:** A Connector is used to connect objects (Domain Objects and Domain Processes) to an external view. The Connector decouples view and model objects and provides access to connected Domain Objects and Domain Processes on a low level. In this respect, it forms a bridge between the objects, decoupling also Domain Objects from Domain Processes and thus making them more combinable and exchangeable.
J	F	A	P	O	**Context:** See Transaction Context.
J	F	A	P	O	**DDL:** A language enabling the structure and instances of a database to be defined in a human- and machine-readable form.
J	F	A	P	O	**Default Base Connection:** The Default Base Connection will be used to hold the Domain Object if no explicit Base Connection is defined for this process.
J	F	A	P	O	**Delegation Model:** An concept used to decouple subsystems from each other. Application programmers implement subclasses of Viewport to isolate the model from the interaction subsystem.
J	F	A	P	O	**Domain Model:** A Domain Model is a design phrase for real world concepts like Customer, Policy or Address. Its instances are called Domain Objects.
J	F	A	P	O	**Domain Object:** Domain objects describe that part of the MVC architecture which corresponds to business terms. The behavior and structure are primarily defined in this respect. The appropriate tool (object net browser) from the Object Behavior Framework is used for this purpose, and mapping to the database is described with STOPF from Persistence Framework . The Domain Object also has open interfaces for connecting authorization and validation.
J	F	A	P	O	**Domain Process:** A Domain Process is a design phrase for processing and workflow-oriented tasks and control flow. Its instances are also called Domain Processes. Due to their controlling-oriented nature, they take also responsibility for managing a Transaction Context if appropriate.

J	F	A	P	O	term and definition
J	F	A	P	O	**Domain Processes Browser:** A tool, supplied with the Application Framework to browse the exiting Domain Processes
J	F	A	P	O	**DPB:** Domain Processes Browser.
J	F	A	P	O	**Extended Description (MicFwExtendedDescription):** The object that completely describes the typing of all variables in a class. The class method createExtendedDescription creates the MicFwExtendedDescription object.
J	F	A	P	O	**Framework Logger:** The tracing facility of the Framework's. Events and messages will be routed through this logger.
J	F	A	P	O	**Framework:** A Framework is a software architecture for certain tasks, which components can be easily reused by application developers. It provides the system with a basic structure in being a collection of cooperating and conceptual concise classes and methods, which are designed to support a task-oriented work progress in application development.
J	F	A	P	O	**Inactive Context:** A non-active context. See: Active Context.
J	F	A	P	O	**InstanceVariableDescription (MicFwInstanceVariableDescription):** The object that completely describes the typing of a single instance variable in a class. The class method createExtendedDescription creates the MicFwInstanceVariableDescription object (which is referenced within the MicFwExtendedDescription object).
J	F	A	P	O	**Isolated Context:** A variable getter sent to a source object that is locked by another context while an isolated context is active will return not the committed target of the transaction object in the other context, but rather the uncommitted target. See: uncommitedread context.
J	F	A	P	O	**M<->N relationship:** In M<->N relationships, a source instance variable references M (where M is any number with the range max...min specified by the cardinality LEFT of the relationship) target objects. There is an instance variable in all of the referenced target objects that reference the N (where N is any number with the range max...min specified by the cardinality RIGHT of the relationship) objects of the source object class, including the current source object. An example of an M<->N relationship would be the relationship between Person (source) and Address (target). Person instance variable addresses would reference M (cardinality min <= M <= cardinality max) Address objects. Address instance variable persons would reference N (cardinality min <= N <= cardinality max) Person objects, including the current Person object.
J	F	A	P	O	**Mapper:** To maintain registration of loosely coupled elements within the Framework, Mappers are used to set, get and remove associations between unique, constant names and their corresponding classes, which themselves may change or may be reimplemented in your project. Mappers are implemented as singletons and can be reached through an easy to use interface as they extend Object with one method per Mapper.
J	F	A	P	O	**MicFwTransactionContext:**
J	F	A	P	O	**Model View Connector:** Is the object that holds the child-process- and Base-Connections for one process.

J	F	A	P	O	term and definition
J	F	A	P	O	**Model View Controller:** The MVC (concept of a Model View Controller) defines an architecture intended to yield a strict decoupling of Domain Model Aspects, flow control and external views - mostly displayed graphically for user interaction.
J	F	A	P	O	**Nested transaction levels:** The transaction levels in a context are nested if more than 1 level exists.
J	F	A	P	O	**OBF (Object Behavior Framework):** 990217TTA updated OBF is a library of classes that when added to your Smalltalk development environment provides a Framework for defining the requirements and restrictions for the behavior of objects.
J	F	A	P	O	**Object Net Browser:** A visual OBF tool that dispayes both the object net of a class and information about the instance variables of the class (key, validated, transacted, persistent, type, relationship). The visual OBF tools Type Editor and Relationship Editor are opened from the Object Net Browser.
J	F	A	P	O	**Object Net:** An Object Net consists of objects and its relationships between them.
J	F	A	P	O	**ObjectVersion (version object):** An ObjectVersion is created if a transaction context is active and a target object is assigned to a transacted variable of a source object. The ObjectVersion is assigned to the highest TrLevel in the active context. If an ObjectVersion already exists in this TrLevel, it is replaced. The ObjectVersion references the committed target object (committed) and the uncommitted target object (uncommitted) referenced by the variable.
J	F	A	P	O	**ONB:** Object Net Browser.
J	F	A	P	O	**Packaging:** The process of creating a runtime executable.
J	F	A	P	O	**Parent context:** A parent context is a context object that created its child context when it responded to the newTransactionContext message. The parent context can have any number of child contexts. A parent context can also be a child context. If parent context B has a parent context A and B also has a child context C, then A and C also have a parent child relationship. See: Child Context.
J	F	A	P	O	**POM:** Persistent Object Manager.
J	F	A	P	O	**Primitive relationship:** 990217TTA updated A 1-way relationship. See: 1-way relationship.
J	F	A	P	O	**Relationship Editor:** A visual OBF tool for establishing relationships between 2 classes. See: object Net Browser.
J	F	A	P	O	**Relationship:** In Smalltalk, relationships between objects are simply represented as object pointers. No distinction is made between complex and simple (scalar) data types since all are full-scale objects and there is conceptually no difference between a relationship and an "embedded" value. Beyond this, the PMS MICADO Object Behavior Framework provides an elaborated relationship concept maintaining referential integrity between objects. Those relationships may even be mapped to (relational) databases with the help of the PMS MICADO Persistence Framework
J	F	A	P	O	**RelationshipDescription (MicFw~):** Object which contains complete information about a relationship.

J	F	A	P	O	term and definition
J	F	A	P	O	**Running context:** 990217TTA updated A context that has a TrLevel1 (it may have higher levels also).
J	F	A	P	O	**Sibling contexts:** See Concurrent Contexts.
J	F	A	P	O	**STOPF / ODBC:** Smalltalk Object Persistence Framework
J	F	A	P	O	**Transacted variable:** A variable that has been marked as transacted in the Object Net Browser. NOTE:Changes to a variable will not be transacted even if a transaction context is active if the variable is not marked as transacted.
J	F	A	P	O	**Transaction Browser:** 990217TTA updated A Visual OBF tool for displaying and manipulating (aborting / committing) transactions.
J	F	A	P	O	**Transaction Context:** 990217TTA updated A logical unit that can contain transaction levels and that can be related to other contexts as a parent, sibling, or child.
J	F	A	P	O	**Transaction Level:** A subunit of a transaction context. A transaction level has a single object version for each variable of each object that was assigned a new target object while the the transaction level was the highest level in its context and its context was active.
J	F	A	P	O	**Transaction Manager:** The transaction manager is a single instance (singleton) of MicFwTransactionManager and manages all running contexts.
J	F	A	P	O	**Transaction:** A defined state of an object that runs under transactional control will be stored. If the interaction on this object fails by some reason, this object can be restored to this state, if the interaction succeeds, the persistent state of this object will be modified to this new state and the old state will be dropped.
J	F	A	P	O	**TrLevel1, 2, ...:** Transaction level 1, 2, ...
J	F	A	P	O	**Type Converter:** A type converter is assigned to a typed variable (including variables in a relationship). When an object is assigned to the variable that is not of the type specified for the variable, the type converter will be used to attempt to create the correct type of object containing the information in the original object and assign this correct type of object to the variable.
J	F	A	P	O	**Type Editor:** 990217TTA updated A visual tool for setting a variable type as a simple type (Integer, Date, etc.). The type converter and size / scale for the variable type can also be selected. The Type Editor is opened from the Object Net Browser.
J	F	A	P	O	**TypeDescription (MicFw~):** Object which contains complete information about the typing of an instance variable.
J	F	A	P	O	**Typing:** The term typing defines the assignment of types (normally basic classes like Integer or String) to instance variables. Since Smalltalk is an untyped language, the PMS MICADO Object Behavior and Persistence Frameworks introduce typing in order to support storing of objects into (typed) relational databases.
J	F	A	P	O	**UML (Unified Markup Language):** 990217TTA updated Graphical notation for OO analysis and design.

J	F	A	P	O	term and definition
J	F	A	P	O	**uncommitedRead context:** 990217TTA updated: If an ObjectVersion exists for aSourceObject>>aSOVariable (ie, aSOVariable is transacted and anUncommittedTargetObject was assigned to aSOVariable while aContext1 was active): If aContext2 is an active uncommitedRead context and getter message sOVariable is sent to aSourceObject, the object returned will be anUncommittedTargetObject. See: Isolated context. Note: The message is spelled "uncommitedRead".
J	F	A	P	O	**Uncommitted target:** 990217TTA updated In a VersionObject: A getter message to a source object will return the uncommitted target object referenced by the variable if [the active context read mode is uncommitedReadisolate AND the active context does not have a lock on the source object variable AND the active context is not a child context of the context that has the lock] OR if the source object's variable is locked by the active context. See: Committed Target.
J	F	A	P	O	**Validation of ExtendedDescription:** 990217TTA updated Validation of an ExtendedDescription involves checking the Object Net for errors.
J	F	A	P	O	**Validation:** Like the authorization service, validation ties in at the model layer and is used for checking value-based access to and from attributes based on certain rules.
J	F	A	P	O	**VariableDescription (MicFw~):** Object which contains complete information about an instance variable.
J	F	A	P	O	**Viewport:** In order to separate view-related state handling from Domain Objects and Domain Processes, Application Framework implements a Viewport as a Delegation Model concept. Functionality for transportation and the filtering of data and states concerning an object net from a certain object's or view's direction is delegated from the Domain Models to them, leading to a more lightweight and view independent kind of Domain Object. Viewports "learn" the dependencies between their Aspects and the corresponding Model Aspects while running the application. This read trace frees the programmer from dealing with changed-events and allows Application Framework to update both sides automatically and in a generic way. A Dispatcher is called Viewport if Domain Objects are concerned.

List Of Tables

List Of Figures

Index

R

S

T

U

V

W

Z