

GEF – Connection – Three Traces

Use Full Trace (No Slicing)

Supporting Applications : [Logic, Flow, Shapes]

```
import org.eclipse.gef.requests.CreateConnectionRequest;
import org.eclipse.gef.editpolicies.GraphicalNodeEditPolicy;
import org.eclipse.gef.editpolicies.ConnectionEditPolicy;
import org.eclipse.draw2d.IFigure;
import org.eclipse.jface.action.IAction;
import org.eclipse.gef.EditPartFactory;
import org.eclipse.gef.palette.PaletteDrawer;
import org.eclipse.gef.palette.PaletteRoot;
import org.eclipse.gef.NodeEditPart;
import org.eclipse.gef.ui.actions.RedoRetargetAction;
import org.eclipse.gef.RootEditPart;
import org.eclipse.jface.resource.ImageDescriptor;
import org.eclipse.gef.EditPartViewer;
import org.eclipse.gef.palette.ToolEntry;
import org.eclipse.gef.ui.parts.GraphicalViewerKeyHandler;
import org.eclipse.gef.editparts.AbstractConnectionEditPart;
import org.eclipse.gef.palette.PaletteSeparator;
import org.eclipse.gef.dnd.TemplateTransferDropTargetListener;
import org.eclipse.gef.palette.MarqueeToolEntry;
import org.eclipse.gef.ui.actions.ActionRegistry;
import java.util.EventObject;
import org.eclipse.gef.palette.ConnectionCreationToolEntry;
import java.util.List;
import org.eclipse.gef.ContextMenuProvider;
import org.eclipse.gef.requests.CreationFactory;
import org.eclipse.gef.ui.actions.ActionBarContributor;
import org.eclipse.ui.IEditorInput;
import org.eclipse.gef.editpolicies.ComponentEditPolicy;
import org.eclipse.gef.palette.CombinedTemplateCreationEntry;
import org.eclipse.ui.IEditorPart;
import org.eclipse.gef.dnd.TemplateTransferDragSourceListener;
import org.eclipse.swt.dnd.Transfer;
import org.eclipse.gef.editpolicies.RootComponentEditPolicy;
import org.eclipse.ui.IWorkbenchPart;
import org.eclipse.gef.DefaultEditDomain;
import org.eclipse.ui.part.WorkbenchPart;
import org.eclipse.gef.ui.parts.GraphicalEditor;
import org.eclipse.gef.commands.Command;
import org.eclipse.gef.EditPart;
import org.eclipse.gef.palette.PaletteGroup;
import org.eclipse.gef.editpolicies.ConnectionEndpointEditPolicy;
import org.eclipse.gef.GraphicalViewer;
import org.eclipse.gef.commands.Command && appCommand && command2;
import org.eclipse.gef.ui.actions.UndoRetargetAction;
import org.eclipse.gef.requests.SimpleFactory;

public class AppAbstractConnectionEditPart
implements NodeEditPart
extends AbstractConnectionEditPart {

    public List getModelSourceConnections() {
        Object object = appAbstractConnectionEditPart.getModel(); // REPEATED!
    }

    public void getSourceConnectionAnchor(editPart1 && appAbstractConnectionEditPart) {
        IFigure ifigure = appAbstractConnectionEditPart.getFigure(); // REPEATED!
    }

    public void activate() {
        Object object = appAbstractConnectionEditPart.getModel(); // REPEATED!
    }

    public IFigure createFigure() {
    }

    public void createEditPolicies() {
        AppComponentEditPolicy appComponentEditPolicy = new AppComponentEditPolicy();
        RootComponentEditPolicy rootComponentEditPolicy = new RootComponentEditPolicy();
        AppGraphicalNodeEditPolicy appGraphicalNodeEditPolicy = new AppGraphicalNodeEditPolicy();
        ConnectionEndpointEditPolicy connectionEndpointEditPolicy = new ConnectionEndpointEditPolicy();
        AppConnectionEditPolicy appConnectionEditPolicy = new AppConnectionEditPolicy();
        appAbstractConnectionEditPart.installEditPolicy(appComponentEditPolicy && appGraphicalNodeEditPolicy && rootComponentEditPolicy &&
        connectionEndpointEditPolicy && appConnectionEditPolicy); // REPEATED!
    }

    public void deactivate() {
        Object object = appAbstractConnectionEditPart.getModel(); // REPEATED!
    }

    public void refreshVisuals() {
        IFigure ifigure = appAbstractConnectionEditPart.getFigure(); // REPEATED!
        Object object = appAbstractConnectionEditPart.getModel(); // REPEATED!
    }

    public List getModelChildren() {
        Object object = appAbstractConnectionEditPart.getModel(); // REPEATED!
    }

    public List getModelTargetConnections() {
        Object object = appAbstractConnectionEditPart.getModel(); // REPEATED!
    }

    public getTargetConnectionAnchor() {
        IFigure ifigure = appAbstractConnectionEditPart.getFigure();
    }
}

public class AppCommand
extends Command {

    public void execute() {
        appAbstractConnectionEditPart.refreshTargetConnections();
        appAbstractConnectionEditPart.refreshSourceConnections();
    }

    public boolean canExecute() {
    }
}

public class AppGraphicalNodeEditPolicy
extends GraphicalNodeEditPolicy {

    public Command getConnectionCompleteCommand(CreateConnectionRequest) {
        Command command = CreateConnectionRequest.getStartCommand(); // REPEATED!
        EditPart editPart = appGraphicalNodeEditPolicy.getHost(); // REPEATED!
        Object object = editPart.getModel(); // REPEATED!
    }
}
```

```

    public Command getConnectionCreateCommand(CreateConnectionRequest) {
        AppCommand appCommand = new AppCommand();
        CreateConnectionRequest.setStartCommand(Command && appCommand && command2); // REPEATED!
        EditPart editPart = appGraphicalNodeEditPolicy.getHost(); // REPEATED!
        Object object = editPart.getModel(); // REPEATED!
    }
}

public class AppditPartFactory
implements EditPartFactory {

    public EditPart createEditPart(appAbstractConnectionEditPart && editPart) {
        AppAbstractConnectionEditPart appAbstractConnectionEditPart = new AppAbstractConnectionEditPart();
        appAbstractConnectionEditPart.setModel(Object); // REPEATED!
    }
}

public class AppGraphicalEditor
extends GraphicalEditor {

    public void configureGraphicalViewer() {
        AppditPartFactory appditPartFactory = new AppditPartFactory();
        /*          Cyclic Statements          */
        GraphicalViewerKeyHandler graphicalViewerKeyHandler = new GraphicalViewerKeyHandler(graphicalViewer);
        graphicalViewer.setEditPartFactory(appditPartFactory); // MAY REPEAT!
        graphicalViewer.setRootEditPart(RootEditPart); // MAY REPEAT!
        ActionRegistry actionRegistry = GraphicalEditor.getActionRegistry(); // MAY REPEAT!
        graphicalViewer.setKeyHandler(graphicalViewerKeyHandler); // MAY REPEAT!
        graphicalViewer.setContextMenu(appContextMenuProvider); // MAY REPEAT!
        GraphicalViewer graphicalViewer = GraphicalEditor.getGraphicalViewer(); // MAY REPEAT!
        AppContextMenuProvider appContextMenuProvider = new AppContextMenuProvider(graphicalViewer && actionRegistry);
    }

    public void commandStackChanged(EventObject) {
        WorkbenchPart.firePropertyChange(int);
    }

    public void initializeGraphicalViewer() {
        GraphicalViewer graphicalViewer = GraphicalEditor.getGraphicalViewer(); // REPEATED!
        graphicalViewer.setContents(EditPart):()|(Object):();
        TemplateTransferDropTargetListener templateTransferDropTargetListener = new TemplateTransferDropTargetListener(graphicalViewer);
        graphicalViewer.addDropTargetListener(templateTransferDropTargetListener); // MAY REPEAT!
    }

    public void setInput(IEditorInput) {
    }
}

public class AppActionBarContributor
extends ActionBarContributor {

    public void declareGlobalActionKeys() {
    }

    public void buildActions() {
        RedoRetargetAction redoRetargetAction = new RedoRetargetAction();
        UndoRetargetAction undoRetargetAction = new UndoRetargetAction();
        ActionBarContributor.addAction(undoRetargetAction && redoRetargetAction); // REPEATED!
    }
}

public class AppComponentEditPolicy
extends ComponentEditPolicy {
}

public class AppContextMenuProvider
extends ContextMenuProvider {
}

public class AppConnectionEditPolicy
extends ConnectionEditPolicy {
}

public class SomeClass {

    public void someMethod() {
        MarqueeToolEntry marqueeToolEntry = new MarqueeToolEntry(String)|(String,String)|();
        TemplateTransferDragSourceListener templateTransferDragSourceListener = new TemplateTransferDragSourceListener(EditPartViewer)|(EditPartViewer,Transfer);
        EditPartViewer.addDragSourceListener(templateTransferDragSourceListener);
        IAction iaction = ActionBarContributor.getAction(String); // REPEATED!
        PaletteDrawer paletteDrawer = new PaletteDrawer(String)|(String,ImageDescriptor);
        PaletteGroup paletteGroup = new PaletteGroup(String,List)|(String);
        ConnectionCreationToolEntry connectionCreationToolEntry = new ConnectionCreationToolEntry(String,String,CreationFactory,ImageDescriptor,ImageDescriptor);
        DefaultEditDomain defaultEditDomain = new DefaultEditDomain(IEditorPart);
        GraphicalEditor.setEditDomain(defaultEditDomain);
        PaletteRoot paletteRoot = new PaletteRoot();
        paletteRoot.setDefaultEntry(ToolEntry);
        SimpleFactory simpleFactory = new SimpleFactory(Class);
        CombinedTemplateCreationEntry combinedTemplateCreationEntry = new CombinedTemplateCreationEntry(simpleFactory);
        PaletteSeparator paletteSeparator = new PaletteSeparator()|(String);
    }
}

```