**How to combine the editors?**

**Setting the stage**

* Org.kermeta.fsm.gmf/model/gmfgen/Gen Editor Generator/Property ***Same File for Diagram And Model* to true and change diagram file extension to match the domain file extension**
* Launch the app .. voila

**Upgrading the editing domain**

* Change FsmEditor Initialize methode (copy from the zip-file)
* @generated NOT
* Add dependency “org.eclipse.emf.transaction”
* Add import **import** org.eclipse.emf.transaction.TransactionalEditingDomain;
* Change the added code

From: domain.setID("org.example.emfgmf.diagram.EditingDomain");

To: domain.setID("org.kermeta.fsm.gmf.diagram.EditingDomain");

**Extending the editor input**

* Add this Attribute to FsmEditor:

/\*\*

\* The input, wrapped in an FileEditorInputProxy instance.

\*/

**protected** IEditorInput wrappedInput;

* Add this method to FsmEditor:

protected IEditorInput getWrappedInput() {

if (wrappedInput == null) {

if (getEditorInput() instanceof IFileEditorInput) {

wrappedInput = new FileEditorInputProxy((IFileEditorInput)getEditorInput(),

(TransactionalEditingDomain) getEditingDomain());

} else {

wrappedInput = getEditorInput();

}

}

return wrappedInput;

}

* Open org.kermeta.fsm.gmf.editor plug in and add dependency: org.kermeta.fsm.gmf.diagram
* Set all the dependencies in diagram package: gmf.\* on *reexport this dependency*
* To remove the error we see in the getWrappedInput() method 🡪 first bundle, then import
* In org.diagram/part/FsmDocumentProvider replace all FileEditorInput with IFileEditorInput expect the one, after “new” in the “handleElementMoved()” method.

Modify the createEmptyDocument() method and add onther method (without param) to FsmDocumentProvider class:

protected IDocument createEmptyDocument() {

**return createEmptyDocument(null);**

}

**protected IDocument createEmptyDocument(Object input) {**

**DiagramDocument document = new DiagramDocument();**

**if (input instanceof FileEditorInputProxy) {**

**FileEditorInputProxy proxy = (FileEditorInputProxy) input;**

**document.setEditingDomain(proxy.getEditingDomain());**

**} else {**

**document.setEditingDomain(createEditingDomain());**

**}**

**return document;**

**}**

* Don’t forget the **@generated** NOT createEmptyDocument()
* Throw the errors that appears import the requested libraries (FileEditorInput, IFileEditorInput)
* we have to change the call to createEmptyDocument(Object) in IDocument createDocument(Object element) method:

From: IDocument document = createEmptyDocument();

To: IDocument document = createEmptyDocument(element);

* Don’t forget the **@generated** NOT createEmptyDocument(Object element)

**Dissecting the EMF editor**

* Copy the class **TopicmapEditorPart** and remane it to **FsmEditorPart (extern and entern)**
* Replace the **menuAboutToShow** method in **FsmEditorPart** class with this one:

/\* (non-Javadoc)

\* @see org.eclipse.jface.action.IMenuListener#menuAboutToShow(org.eclipse.jface.action.IMenuManager)

\*/

public void menuAboutToShow(IMenuManager manager) {

// pass the request to show the context menu on to the parent editor

((IMenuListener)parentEditor.getEditorSite().getActionBarContributor()).menuAboutToShow(manager);

}

* Copy the six EditorPart classes from the zipfile into fsm/presentation
* Replace all the six packeges errors with: **package** fsm.presentation;
* Replace all the errors with Fsm
* Copy the reverse class into presentation
* Replace the six attributes in fsmEditor class with this code:

/\*\*

\* This is the viewer that shadows the selection in the content outline.

\* The parent relation must be correctly defined for this to work.

\* <!-- begin-user-doc -->

\* <!-- end-user-doc -->

\* **@generated** NOT

\*/

**protected** SelectionEditorPart selectionEditorPart;

/\*\*

\* This inverts the roll of parent and child in the content provider and show parents as a tree.

\* <!-- begin-user-doc -->

\* <!-- end-user-doc -->

\* **@generated** NOT

\*/

**protected** ParentEditorPart parentEditorPart;

/\*\*

\* This shows how a tree view works.

\* <!-- begin-user-doc -->

\* <!-- end-user-doc -->

\* **@generated** NOT

\*/

**protected** TreeEditorPart treeEditorPart;

/\*\*

\* This shows how a list view works.

\* A list viewer doesn't support icons.

\* <!-- begin-user-doc -->

\* <!-- end-user-doc -->

\* **@generated** NOT

\*/

**protected** ListEditorPart listEditorPart;

/\*\*

\* This shows how a table view works.

\* A table can be used as a list with icons.

\* <!-- begin-user-doc -->

\* <!-- end-user-doc -->

\* **@generated** NOT

\*/

**protected** TableEditorPart tableEditorPart;

/\*\*

\* This shows how a tree view with columns works.

\* <!-- begin-user-doc -->

\* <!-- end-user-doc -->

\* **@generated** NOT

\*/

**protected** TableTreeEditorPart tableTreeEditorPart;

* Modify the method createpages() from try to catches by copying it
* Add this attribte to FsmEditor:

/\*\*

\* The graphical diagram editor embedded into this editor.

\*/

**protected** TopicMapDiagramEditor diagramEditor;

* Remove isDirty()

setFocus()

setCurrentViewerPane()

**Selection handling**

* Add attribute to FsmEditor:

/\*\*

\* This selection provider coordinates the selections of the various editor parts.

\*/

**protected** MultiPageSelectionProvider selectionProvider;

* Modify the constructure and don’t forget the @generate NOT

/\*\*

\* This creates a model editor.

\* <!-- begin-user-doc -->

\* <!-- end-user-doc -->

\* **@generated** NOT

\*/

**public** FsmEditor() {

**super**();

initializeEditingDomain();

selectionProvider = **new** MultiPageSelectionProvider(**this**);

selectionProvider.addSelectionChangedListener(**new** ISelectionChangedListener() {

**public** **void** selectionChanged(SelectionChangedEvent event) {

setStatusLineManager(event.getSelection());

}

});

}

* Modify setcurrentViewer() and @generated

/\*\*

\* This makes sure that one content viewer, either for the current page or the outline view, if it has focus,

\* is the current one.

\* <!-- begin-user-doc -->

\* <!-- end-user-doc -->

\* **@generated** NOT

\*/

**public** **void** setCurrentViewer(Viewer viewer) {

// If it is changing...

//

**if** (currentViewer != viewer) {

// Remember it.

//

currentViewer = viewer;

}

}

* Modify handleActivate() and generated

Add this line

selectionProvider.setSelection(selectionProvider.getSelection());

insteadof the line

setSelection(getSelection());

* ModifyhandleContentOutlineSelection()

**Combining the Editors**

* Modify handleContentOutlineSelection .. again
* Remove this code from the diagram plugin.xml file

<extension point="org.eclipse.ui.editors" id="gmf-editor">

<?gmfgen generated="true"?>

<editor

id="fsm.diagram.part.FsmDiagramEditorID"

name="%editorName"

icon="icons/obj16/FsmDiagramFile.gif"

extensions="fsm"

default="true"

class="fsm.diagram.part.FsmDiagramEditor"

matchingStrategy="fsm.diagram.part.FsmMatchingStrategy"

contributorClass="fsm.diagram.part.FsmDiagramActionBarContributor">

</editor>

</extension>

Tip: I think I can regenerate after all of this the diagram. Because of the flag @generate NOT in the diagram/part/FsmDocumentProvider.

Problems after vacation;

1. The xmi:id in the xml file
2. Auto generation
3. In create page I can choose wich parts I want to show or not (by adding and removing the code of the specific part)