Implementing a Visual Paradigm plug-in

Configuring development environment

The plugin API is in **%Visual-Paradigm-Install-Dir%/lib/openapi.jar**. In order to program a plugin, developer must import the jar into the development classpaths.

Beginning of plugin.xml

plugin.xml is the base of a plugin, to develop a plugin, you should start from writing the plugin.xml. The basic directory structure is "Visual-Paradigm-Install-DIR/plugins/YOUR_PLUGIN_ID/plugin.xml"

For improving the variability of the plugin.xml, a properties file (plugin.properties) can be used for storing the value of the xml. Developer can assign the value of the attributes in xml starts with '%', then the value will be read from the properties file. For example

In plugin.xml:

In plugin.properties: plugin.name=sample.plugin

Sample on XML:

Table shows the description of elements in the plugin.xml.

Element	Attribute	Description
plugin		The root element of plugin.xml, specify the basic information of the plugin (id, name, provider, etc...)
plugin	**class**	The class of the plugin, required to implements **com.vp.plugin.VPPlugin**.
runtime		The element specified the runtime environment of the plugin.
library (1..*)		Specify the .jar or directory as the classpaths required on the

Element	Attribute	Description
		plugin. Such as the classes of the plugin and some libraries the plugin required.
library (1..*)	**Path**	The path of the .jar or directory.
library (1..*)	**relativePath** (optional, default: **true**)	Specify whether the path is relative path.

Description on Code:

VPPlugin (com.vp.plugin.VPPlugin)

This class must be implemented and ref on

The following is the sample code:

```
package sample.plugin;
public class SamplePlugin implements com.vp.plugin.VPPlugin {

	// make sure there is a constructor without any parameters

	public void loaded(com.vp.plugin.VPPluginInfo info) {

		// called when the plugin is loaded
		// developer can get the current plugin's id and the
		// current plugin directory (default: %Visual-Paradigm%/plugins)of Visual-
		Paradigm from the VPPluginInfo.


	}
	public void unloaded() {

		// called when the plugin is unloaded (when Visual Paradigm will be exited)


	}


}
```

Implementing custom action

There are 2 main components for an Action: Action and Action Controller. Action represents the outlook, Action Controller responses to work as function call. In order to create custom action, developer needs to define the Action on xml, and implement the Action Controller on code.

Sample on XML:

Table shows the description of elements in the above XML.

Element	Attribute	Description
actionSets		It is a collection of ActionSet. There 2 kinds of ActionSet: **actionSet** and **cont extSensi tiveActionSet**. **actionSet** is a set of actions which will be shown on menu/toolbar or diagram toolbar. **contextSensitiveActionSet** is set of actions which will be shown on popup menu.

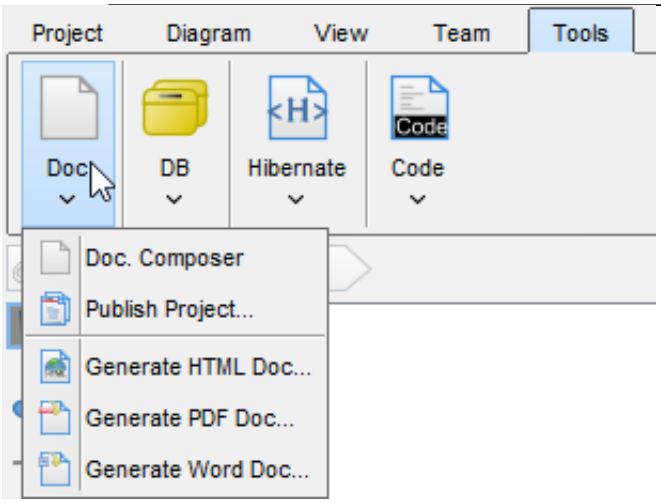
There are differences on xml definition and code implementation of the 3 kinds of Actions (menu/toolbar, popup menu, diagram toolbar).

Custom action on menu/Toolbar

Developer can define the menu, menu item, toolbar, toolbar button and etc... on the plugin.xml. In order to trigger the menu item and toolbar button's function call, Action Controller is required to be implemented and added into the Action. The Action Controller class on menu/toolbar actions is com.vp.plugin.action.VPActionController.

There are 2 important attributes used on menu, action and separator: **menuPath** and **toolbarPath**.

menuPath is the path specified where is the item placed on menu, toolbarPath is the path specified where is the item placed on toolbar. The path is formed by a set of 'name'. The 'name' is similar with the caption of the menu items (caption in English, ignores the "..." and remind the 'space'). '/' is used as delimiter of the path. '#' is used to represent the front of the menu. Here, 4 examples will be given:



Below is the menupaths required for implementing the menus shown in the above images.

Menu	"label" in XML	"menupath" in XML	Remarks
1	Tools	Tools	After the Tools menu
2	Tools/Document	Tools/Document	Under the Tools menu, after the Document menu
3	Tools/Document/#	Tools/Document/#	Under the Tools menu, and under the Document menu, place on the front
4	Tools/Document/Generate HTML Document	Tools/Document/Generate HTML Document	Under the Tools menu, and under the Document menu, after the Generate HTML Document menu item

Sample on XML:

on of elements in the above XML.

	It is a collection of ActionSet. There are 2 kinds of ActionSet: **actionSet** and **contextSensitiveActionSet**. **actionSet** is a set of actions which will be shown on menu/toolbar or diagram toolbar. **contextSensitiveActionSet** is set of actions which will be shown on popup menu.
	Specify a toolbar, contains the location information of the toolbar.
orientation [north | east |south | west]	Specify which side will be the toolbar placed on.
index [(number) | last |new]	Based on the orientation, where the toolbar will be placed. e.g. the orientation is "north" and there are 2 rows toolbars already. If the index is "0", then the toolbar will be placed on the first row's last position. If the index is "last", the toolbar will be placed on the last row, last position. If the index is "new", the toolbar will be placed on the third row (new row).
	Specify a menu or a pull down button on menu bar or toolbar. It contains the outlook information of the menu.
	Specify a menu item or button on menu bar or toolbar. It contains the outlook information of the menu item.
actionType [generalAction | shapeAction | connectorAction] (optional, default: generalAction)	There are 3 types: generalAction, shapeAction and connectorAction. As the action on menu/toolbar, generalAction should be assigned.
	Specify the Action Controller for the action (the parent node in the xml).
	The class name of the Action Controller. For the action on menu/toolbar, it is required to implement **com.vp .plugin.action.VPActionController**.
	Specify a separator on menu bar or toolbar.

.plugin.action.VPActionController)

the function call when the action is clicked. One Action Controller class refers to multi Actions is allowed.

n.actions;
ntroller implements com.vp.plugin.action.VPActionController {

here is an constructor without any parameters

formAction(com.vp.plugin.action.VPAction action) {

d when the button is clicked, the parameter action represents the Action which be clicked.
loper also can set the properties of the action

date(com.vp.plugin.action.VPAction action) {

the actions located on menu bar only. Only function when running the plug-in in Classic UI
n the parent menu is selected, this will be called,
loper can set the properties of the action before it is shown (e.g. enable/disable the menu item)

p menu (context sensitive)

nu, menu item and separator on the popup menu shown on the diagram. The popup menu on diagram is
l on what diagram element or diagram is selected. In order to make the menu item trigger the function
red to be implemented. For popup menu, **com.vp.plugin.action.VPContextActionController** is the
o implement.

lbar, **menuPath** is used to specify the location of the action (menu/menu item on popup menu).

nts in the above XML.

Attribute	
	It is a collection of menu, action, separator on the popup menu of the plugin. The child elements should be ordered if they have the relationship on the position (e.g. developer prefers Action1 is placed into Menu1, then please define the Menu1 on the xml first
	It is a collection of the model of diagram element of diagram types which the **contextSensitiveActionSet** is considering.
all [true | false] (optional, default: **false**)	Specify whether all the types of the models, diagram elements and diagrams will be considered by this actionSet.
	Specify the model, diagram element or diagram type will be considered by this ActionSet. (This will be ignored if the **contextType's** attribute **'all'** is assigned 'true'.
type	It is type of the element. Such as "Class", "Actor", "ClassDiagram", "Attribute", etc...
	Specify the model, diagram element or diagram type will not be considered by this ActionSet. (This will be ignored if the **contextType's** attribute **'all'** is assigned 'false'.
	It is type of the element. Such as "Class", "Actor", "ClassDiagram", "Attribute", etc...
	Specify the Action Controller for the action (the parent node in the xml)
class	The class name of the Action Controller. For the action on popup menu, it is required to implement **com.vp .plugin.action.VPContextActionController**.

p.plugin.action.VPContextActionController)

tion call when the action is clicked. One Action Controller class refers to multi Actions is allowed.

s;
vent;

ntroller implements com.vp.plugin.action.VPContextActionController {

an constructor without any parameters

tion(

action.VPAction action,
action.VPContext context,

the button is clicked

action.VPAction action,
action.VPContext context

pup menu is selected, this will be called,
an set the properties of the action before it is shown (e.g. enable/disable the menu item)

VPContext)

n Controller when the popup menu is shown or action is triggered. It is what the user selected on the
ment or/and diagram.

m elements, when user right-click on the diagram element or the diagram, a popup menu will be shown.
ment or diagram. However, the diagram element must be contained by diagram, then if popup menu is
ntext must contain both diagram element and diagram. And the diagram element always represents for a
contains model, diagram element and diagram as same time. However, sometime, the popup menu is
n an attribute of a class, because there is no diagram element for the attribute, the class's diagram
ext).

ape and connector)

onnect on the specified diagram. But it is not allowed to develop a custom model. ActionSet and Action
ram element.

Attribute	Description
actionType[generalActio (optional, default: genera	
editorToolbarPath	
shapeType	
captionStyle [center | no	

Attribute	Description
controllerClass	
shapeType	
connectorStyle [oblique oblique) **fromArrowHeadStyle**(o	
toArrowHeadStyle(optic **fromArrowHeadSize**[ve extraLarge | jumbo | colo **toArrowHeadSize**[veryS extraLarge | jumbo | colo **dashes** (optional)	
lineWeight (optional)	

er {

of the shape.

apeInfo) {

e models in Visual Paradigm. The base class of the model is **com.vp.plugin.model.IModelElement**.
a model type to access all the model type, please refer to the

tory) to create the model. Or based on a parent model (**com.vp.plugin.model.IModelElementParent**)

es the functions to create all the models.

e the child into it. If the parent class is more specified, it may support a more details function to create
eration() to create an operation into it.

mentFactory.MODEL_TYPE_CLASS);

s model"
e actor cannot be the child of class model

ntFactory.MODEL_TYPE_OPERATION);
ODEL_TYPE_ACTOR);

ugin.action.VPContext) from ActionController to retrieve the models.

agram elements. It provides function (**modelElementIterator()**) for the developer to iterate the models.

menu's action controller (**com.vp.plugin.action.VPContextActionController**). Context may contain a

r the diagram

nd().getModelElement();

properties. For setting and getting the model's property, cast the **IModelElement** into its sub-class is
ext. Developer checks whether the model is a **IClass**, then developer casts

{

et to be the previous model's super class
eateGeneralization();

nce().createJavaOperationCodeDetail());

);

e diagrams or diagram elements in Visual Paradigm. The base class of the diagram is
m.vp.plugin.diagram.DiagramElement. All diagrams are contained in the project
s. The diagram elements can be contained by the diagrams.

. DiagramManager can be accessed by ApplicationManager.instance().getDiagramManager().

anager();

AGRAM_TYPE_CLASS_DIAGRAM);

odel is contained by a package
ement(diagram, packageModel1);
ent(diagram, classModel1);

TYPE_CLASS);

Use a diagram (**com.vp.plugin.diagram.IDiagramUIModel**) to retrieve the contained diagram elements.
he diagram and/or diagram element.

agram elements. It provides function (**diagramIterator()**)for the developer to iterate the diagrams.

menu's action controller (**com.vp.plugin.action.VPContextActionController**). Context may contain a

ementIter.next();

ck the
ector()

c...).

se there is a difference between shape and connector,
hem.

ramUIModel.delete() and **IDiagramElement.delete()**.

Swing (e.g. Eclipse, Visual Studio). That may make the process to be hung on if using the Swing dialog
show the dialog with Swing technology.

dialog as same as show dialog by JOptionPane. Besides that, **Viewmanager** supports developer to
nting an interface (**com.vp.plugin.view.IDialogHandler**).

arent component. To get the component in Visual Paradigm, just call **ViewManager.getRootFrame()**.

the message on Message Pane. The parameter **msg** is the content of the message, **msgTabId** is the id

ort User's Guide. >=)", "sample.plugin");

ptionPane.showMessageDialog(...)). **ViewrManager**provides the functions which simulate the
the dialog. The signature of the functions are same with the JOptionPane. Developer need not feel

log's content pane. But in plugin, developer is required to implement an

mented.

og (similar to the content pane).

still not shown out). Developer can set the outlook of the dialog on **prepare()**, such as title, bounds and
lt. If developer don't want change the location, there is no necessary to call the **setLocation()** function.

g is shown, such as checking something before user to input data on the dialog.

Developer may not allow the user to close the dialog (e.g. failed on validation check), then please return

bel(" = ");

putField2);
utton);

));
));

alog(
nish this test.",

sageDialog(

sageDialog(