Creating new actions

You can find the code examples in

- *<modeling tool installation directory>\openapi\examples\actiontypes*
- *<modeling tool installation directory>\openapi\examples\simpleconfigurators*
- *<modeling tool installation directory>\openapi\examples\selectionactions*

Step #1: Create a new action class

All actions used in a modeling tool must be subclasses of the *com.nomagic.magicdraw.actions.MDAction* class (see javadoc for more details). The following *MDAction* subclasses that are used for basic purposes are already created in a modeling tool:

- *com.nomagic.magicdraw.ui.browser.actions.DefaultBrowserAction* – an action class, used for a browser action. It enables to access some browser tree and nodes and is recommended to use for performing some actions with the selected browser nodes.
- *com.nomagic.magicdraw.ui.actions.DefaultDiagramAction* – an action class for a diagram action. It enables to access some diagram elements and is recommended to use for performing some actions with the selected diagram elements.
- *com.nomagic.magicdraw.actions.PropertyAction* – an action for changing some element or application property. It can be used for changing properties defined by a user.

You must override at least the *actionPerformed(java.awt.event.ActionEvent)* method and implement in it what this actions is going to do.

Example #1: A simple action

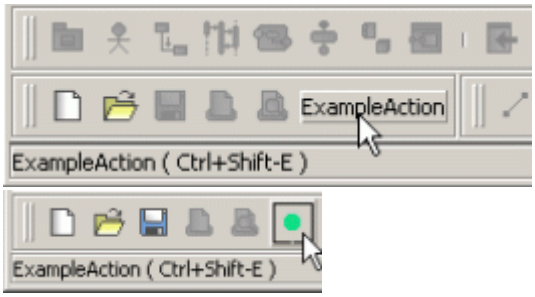
```
    class SimpleAction extends MDAction
    {
        public SimpleAction(String id, String name)
        {
             super(id, name, null, null);
        }
        /**
        * Shows message.
        */
        public void actionPerformed(ActionEvent e)
        {
            JOptionPane.showMessageDialog(MDDialogParentProvider.
getProvider().getDialogParent(), "This is:" + getName());
        }
    }
```

Example #2: An action for a browser

```
    public class BrowserAction extends DefaultBrowserAction
    {
        /**
        * Creates action with name "ExampleAction"
        */
        public BrowserAction()
        {
          super("", "ExampleAction", null, null);
        }
        public void actionPerformed(ActionEvent e)
        {
            Tree tree = getTree();
            String text="Selected elements:";
            for (int i = 0; i < tree.getSelectedNodes().length; i++)
            {
                Node node = tree.getSelectedNodes()[i];
                Object userObject = node.getUserObject();
                if (userObject instanceof BaseElement)
                {
                    BaseElement element = (BaseElement) userObject;
                    text += "\n"+element.getHumanName();
                }
            }
            JOptionPane.showMessageDialog(MDDialogParentProvider.
getProvider().getDialogParent(), text);
        }
    }
```

Step #2: Specify action properties

Property	Function
ID	ID is a unique String for identifying an action. If the action ID is set to *null, a* new ID is generated. An action name can be used as ID. All MagicDraw (or another modeling tool developed by us) default actions IDs are defined in the *com.nomagic.magicdraw.actions.ActionsID* class (for more details, see JavaDoc). You can use these constants for accessing actions. Do not use it as new actions ID.
Name	The action name is visible in all GUI elements.
Shortcut and mnemonic	Every action can have an assigned keyboard shortcut. Shortcuts can be customized from the **Environment Options** dialog box. ⚠ The action must have an ID assigned (not null). In other case, shortcuts cannot be restored after restarting an application.

Icon	Every action can have a small and large icon. A small icon is described as *javax.swing.Action.SMALL_ICON* and can be used in menu items. A large icon is used in toolbar buttons. The action for a toolbar must have a large icon, otherwise it is displayed as a button with an action name.
	 
	
	`// setting an icon. On the toolbar, the button with an icon looks better than with the text.`
	`action.setLargeIcon( new ImageIcon(getClass().getResource ("main_toolbar_icon.gif") ) );`

Description	The action's description that is shown as a tool tip text.

Step #3: Describe Enabling/Disabling logic

There are two ways for controlling the update of the actions state:

1. **Add an action to the predefined actions group.**

   Actions can be added into one of predefined actions groups. All actions of the one group are disabled/enabled together. Conditions for groups enabling/disabling and status updating are predefined and cannot be changed.

   Example:

```
MDAction action = new MDAction("Example","Example",KeyEvent.VK_E,
ActionsGroups.PROJECT_OPENED_RELATED);
```

2. **Implement the *updateState()* method for the action**.

   Here you may describe all conditions when an action must be enabled and disabled. Example of the *updateState()* method for the browser shortcut menu action:

```
        public void updateState()
        {
           //action will be enabled only if there are some selected nodes.
                        Tree tree = getTree();
                        if(tree != null)
                        {
                                    setEnabled(tree.getSelectedNode() !=
null);
                        }
        }
```

If the action is not added to any group, the *updateState()* method for all such actions will be invoked after executing any command and after closing/opening a project or window.

When some actions need to refresh their state, all actions without a group can be updated manually:

```
ActionsStateUpdater.updateActionsState();
```

Step #4: Configure actions

Every action must be added into an *com.nomagic.actions.ActionsCategory* class. *ActionsCategory* is a small group for actions. It can be represented as a separator or submenu (a nested category).

Categories are added into the *com.nomagic.actions.ActionsManager* class, which is some kind of an actions container. One *ActionsManager* represents one GUI element – a menu bar, a shortcut menu, or a toolbar.

The following table explains how classes of a modeling tool maps into GUI elements:

	ActionsManager	Category	Action
Menu	Menu bar	Menu	Menu item
Toolbar	All toolbars	One toolbar	Button
Shortcut Menu	Shortcut menu	Submenu	Menu item

Actions in *ActionsManagers* are configured by many *Configurators*. A *Configurator* is responsible for adding or removing the action into a strictly defined place and position between other actions.

There are three types of configurators:

- *com.nomagic.actions.AMConfigurator* - a configurator for a general purpose. It is used for menus, toolbars, browser, and diagrams shortcuts.
- *com.nomagic.magicdraw.actions.BrowserContextAMConfigurator* - a configurator for configuring managers for a browser shortcut (pop-up) menu. It can access a browser tree and nodes.
- *com.nomagic.magicdraw.actions.DiagramContextAMConfigurator* - a configurator for configuring shortcut menus in a diagram. It can access a diagram, selected diagram elements, and an element that requests a shortcut menu.

Once *ActionsManagers* for the main menu and all toolbars are created and configured, the actions can be only disabled but not removed later.

Shortcut menus are created on every invoking, so *ActionsManagers* are created and configured every time and actions can be added or removed every time.

Example #1: Add some action into a browser's shortcut menu

```
    final DefaultBrowserAction browserAction = ...
    BrowserContextAMConfigurator brCfg = new BrowserContextAMConfigurator()
    {
        // Implement configuration.
        // Add or remove some actions in ActionsManager.
        // A tree is passed as an argument, provides ability to access
nodes.
       public void configure(ActionsManager mngr, Tree browser)
       {
                    // Actions must be added into some category.
                    // So create the new one, or add an action into the
existing category.
                    MDActionsCategory category = new MDActionsCategory("",
"");
                   category.addAction(browserAction);
              // Add a category into the manager.
                   // A category isn't displayed in a shortcut menu.
                   mngr.addCategory(category);
       }
       public int getPriority()
       {
          return AMConfigurator.MEDIUM_PRIORITY;
       }
   };
```

Example #2: add some action into main menu, after creating a new project

```
    // Create an action.
    final MDAction someAction = ...
    AMConfigurator conf = new AMConfigurator()
    {
        public void configure(ActionsManager mngr)
        {
           // Searching for an action after which an insertion should be
done.
           NMAction found= mngr.getActionFor(ActionsID.NEW_PROJECT);

           // The action found, inserting
           if( found != null )
           {
                      // find the category of the "New Project" action.
                      ActionsCategory category = (ActionsCategory)mngr.
getActionParent(found);


                     // Get all actions from this category (menu).
                     List actionsInCategory = category.getActions();


                     //Add the action after the "New Project" action.
                     int indexOfFound = actionsInCategory.indexOf(found);
                     actionsInCategory.add(indexOfFound+1, someAction);

                     // Set all actions.
                     category.setActions(actionsInCategory);
                       }
            }

            public int getPriority()
            {
                return AMConfigurator.MEDIUM_PRIORITY;
            }
           };
```

Step #5: Register Configurator

All configurators are registered in the *com.nomagic.magicdraw.actions.ActionsConfiguratorsManager* class. *ActionsConfiguratorsManager* enables to add or remove many configurators to every configuration predefined by a modeling tool.

All available configurations are accessible in the following way:

```
    ActionsConfiguratorsManager.getInstance().
add<configuration_name>Configurator(configurator);
```

Example: Add new configurator for CONTAINMENT_BROWSER_CONTEXT configuration

```
    //See previous examples, how to create a configurator for browser
actions.
    //Add the configurator into ActionsConfiguratorsManager.
    ActionsConfiguratorsManager.getInstance().
addContainmentBrowserContextConfigurator(brCfg);
```