Adding new diagram types

Adding new diagram types to the modeling tool consists of two steps:

1. Overriding the abstract *com.nomagic.magicdraw.uml.DiagramDescriptor* class
2. Registering a new diagram type

Step #1. Override the abstract *com.nomagic.magicdraw.uml.DiagramDescriptor* class

Override the abstract class *DiagramDescriptor* and implement the following abstract methods:

- *getDiagramTypeId()* – this method must return a diagram type id, which is used to identify the diagram
- *getSingularDiagramTypeHumanName()* – the method returns the diagram type human name
- *getPluralDiagramTypeHumanName()* – the method returns the diagram type human name in a plural form
- *getSuperType()* – this method must return a super type (super diagram) of this diagram type
- *isCreatable()* – returns the flag indicating if this diagram type is creatable
- *getSVGIcon()* – returns a large icon for this type of the diagram (see Adding New Functionality)
- *getSmallIconURL()* – returns a small icon URL for this type of the diagram (see Adding New Functionality)
- *getDiagramActions()* – returns the manager of actions used in the diagram
- *getDiagramToolbarConfigurator()* – returns the action manager configurator which configures the described diagram toolbar (see Adding New Functionality)
- *getDiagramShortcutsConfigurator()* – returns the action manager configurator which configures described diagram shortcuts (see Adding New Functionality)
- *getDiagramContextConfigurator()* – returns the action manager configurator which configures described diagram shortcut menu actions (see Adding New Functionality).

The diagram descriptor example

For the full source code, see the Open API examples in *<program installation directory>\openapi\examples.*

```
/**
* Descriptor of a specific diagram.
*/
public class SpecificDiagramDescriptor extends DiagramDescriptor
{
     public static final String SPECIFIC_DIAGRAM = "Specific Diagram";

    /**
    * Let this diagram type be a sub type of a class diagram type.
    */
    public String getSuperType()
    {
        return DiagramType.UML_CLASS_DIAGRAM;
    }

    /**
    * This is a creatable diagram.
    */
    public boolean isCreatable()
    {
        return true;
    }

    /**
    * Actions used in this diagram.
    */
    public MDActionsManager getDiagramActions()
    {
        return SpecificDiagramActions.ACTIONS;
    }

    /**
```

```
        * A configurator for a diagram toolbar.
        */
        public AMConfigurator getDiagramToolbarConfigurator()
        {
            return new SpecificDiagramToolbarConfigurator();
        }

        /**
        * A configurator for diagram shortcuts.
        */
        public AMConfigurator getDiagramShortcutsConfigurator()
        {
            return new ClassDiagramShortcutsConfigurator();
        }

        /**
        * A configurator for a diagram shortcut menu.
        */
        public DiagramContextAMConfigurator getDiagramContextConfigurator()
        {
            return new BaseDiagramContextAMConfigurator();
        }

        /**
        * Id of the diagram type.
        */
        public String getDiagramTypeId()
        {
            return SPECIFIC_DIAGRAM;
        }

        /**
        * A diagram type human name.
        */
        public String getSingularDiagramTypeHumanName()
        {
            return "Specific Diagram";
        }

        /**
        * A diagram type human name in a plural form.
        */
        public String getPluralDiagramTypeHumanName()
        {
            return "Specific Diagrams";
        }

        /**
        * Resizable svg icon for diagram.
        */
        public ResizableIcon getSVGIcon()
                {
                           ResizableIcon icon = null;
                           try
                         {
                              icon = IconsFactory.getSvgIcon(new File
("icons/specificdiagram.svg").toURI().toURL());
                           }
                           catch(Exception e){e.printStackTrace();}
                           return icon;
                }

        /**
        * URL to a small icon for a diagram.
        */
        public URL getSmallIconURL()
        {
             return getClass().getResource("icons/specificdiagram.svg");
        }
    }
```

Step #2. Register a new diagram type

The new diagram descriptor should be registered in a modeling tool using the *com.nomagic.magicdraw.core.Application.addNewDiagramType(DiagramDescriptor)* method of the modeling tool you are using. This method can be invoked when a plugin is initialized.

The diagram descriptor registration example

For the full source code, see the Open API examples in *<program installation directory>\openapi\examples\specificdiagram.*

```
        class SpecificDiagramPlugin extends Plugin{
        /**
        * Initializing the plugin.
        */
        public void init()
        {
            // Registering a new diagram type
            Application.getInstance().addNewDiagramType(new
SpecificDiagramDescriptor());
        }
        /**
        * Always returns true, because this plugin does not have any close
specific actions.
        */
        public boolean close()
        {
            return true;
        }
        /**
        * Always returns true, because this plugin does not have any
specific suportability conditions.
        */
        public boolean isSupported()
        {
            return true;
        }
    }
```