Creating a pattern

Here is an example of a simple pattern for adding a method into the target classifier. The pattern allows for editing a method name in the **Pattern Wizard**. The pattern will be added as a plugin into your modeling tool.

The example consists of 5 steps:

Step #1. Creating a pattern properties class

The *MyPattern* class has only two main properties for displaying the not editable target classifier name and for entering the method name.

```
    package com.nomagic.magicdraw.plugins.impl.examples.mypatterns;

    import com.nomagic.magicdraw.plugins.impl.patterns.
AbstractPatternProperties;
    import com.nomagic.magicdraw.properties.StringProperty;
    import com.nomagic.magicdraw.properties.PropertyManager;

    public class MyPatternProperties extends AbstractPatternProperties
    {
        public static final String TARGET_CLASS = "TARGET_CLASS";
        public static final String METHOD_NAME = " METHOD_NAME ";

        /**
        * Add two properties into main properties manager.
        */
        protected void configurePropertyManager()
        {
            StringProperty p = new StringProperty(TARGET_CLASS,
getTarget().getTargetClassifier().getName());
            p.setResourceProvider(MyResourceProvider.getInstance());
            p.setEditable(false);
            PropertyManager properties = getPropertyManager();
            properties.addProperty(p);

            p = new StringProperty(METHOD_NAME, "method");
            p.setResourceProvider(MyResourceProvider.getInstance());
            properties.addProperty(p);
        }
    }
```

MyPattern does not have extended properties, so we do not override the *com.nomagic.magicdraw.patterns.AbstractPatternProperties.configureExtendedPropertyManager()* method.

Names of your modeling tool properties can be translated into other programming languages, so they are not hard coded inside the properties. To get a property name from the property ID, *com.nomagic.magicdraw.properties.PropertyResourceProvider* is used.

Let's write a simple *PropertyResourceProvider* for our pattern's properties.

```
package com.nomagic.magicdraw.plugins.impl.examples.mypatterns;
    import com.nomagic.magicdraw.properties.PropertyResourceProvider;
public class MyResourceProvider implements PropertyResourceProvider
{
    /**
    * An instance of this provider.
    */
    private static MyResourceProvider mInstance;

    /**
    * Returns a shared instance of this provider.
    * @return a shared instance.
    */
    public static MyResourceProvider getInstance()
            {
        if(mInstance == null)
        {
            mInstance = new MyResourceProvider();
        }
        return mInstance;
           }

    /**
    * Returns resource for given key.
    * @param key a resource key.
    * @return the resource for given key.
    */
    public String getString(String key)
    {
        if(key.equals(MyPatternProperties.METHOD_NAME))
        {
            return "Method Name";
        }
        else
        if(key.equals(MyPatternProperties.TARGET_CLASS))
        {
            return "Target Class";
        }
        return null;
    }
}
```

Step #2. Creating a pattern panels class

API provides the default implementation of *com.nomagic.magicdraw.patterns.AbstractPanelContainer* for patterns without extended properties (just with the main properties panel). This is the *com.nomagic.magicdraw.patterns.impl.common.PatternPropertiesPanel* class. *MyPattern* will use this class. Other patterns may extend *AbstractPanelContainer* and implement/override corresponding methods.

Step #3. Creating a pattern class

Now we have *MyPattern* properties and panels. We just need the pattern implementation:

```
    package com.nomagic.magicdraw.plugins.impl.examples.mypatterns;

    import com.nomagic.magicdraw.core.Application;
    import com.nomagic.magicdraw.openapi.uml.ReadOnlyElementException;
    import com.nomagic.magicdraw.plugins.impl.patterns.AbstractPattern;
    import com.nomagic.magicdraw.plugins.impl.patterns.
AbstractPatternProperties;
    import com.nomagic.magicdraw.plugins.impl.patterns.PatternHelper;
    import com.nomagic.magicdraw.plugins.impl.patterns.Target;
    import com.nomagic.magicdraw.properties.PropertyManager;
    import com.nomagic.uml2.ext.magicdraw.classes.mdkernel.Operation;
    import com.nomagic.uml2.impl.ElementsFactory;

    public class MyPattern extends AbstractPattern
    {
        /**
        * Returns the categorized name of the pattern.
        * @return name of the pattern {"Common", "MyPattern"}.
        */
        public String[] getCategorizedName()
        {
            return new String[]{"Common", "My Pattern"};
        }

        /**
        * Applies a design pattern to the target, using properties, passed
as an argument.
        * @param target Target, the pattern is applied for.
        * @param prop the pattern properties.
        */
        public void applyPattern(Target target, AbstractPatternProperties
prop) throws ReadOnlyElementException
        {
            PropertyManager propManager = prop.getPropertyManager();
            String methodName = propManager.getProperty
(MyPatternProperties.METHOD_NAME).getValue().toString();
            ElementsFactory ef = Application.getInstance().getProject().
getElementsFactory();
            Operation op = ef.createOperationInstance(); op.setName
(methodName);
            PatternHelper.addDistinctOperation(target.
getTargetClassifier(), op);
        }
    }
```

Step #4. Creating *Description.html*

Every pattern may provide an *html* file with a pattern description. We will provide *Description.html* one:

```
    <h2>MyPatern</h2>
    <h3>Intent</h3>
```

Add a method into a classifier.

The *description.html* file must be in the same package as the *MyPattern* class.

Step #5. Creating plugins

Let's create the class *MyPatternPlugin*.

```
    package com.nomagic.magicdraw.plugins.impl.examples.mypatterns;

    import com.nomagic.magicdraw.plugins.Plugin;
    import com.nomagic.magicdraw.plugins.impl.patterns.impl.common.
PatternPropertiesPanel;
    import com.nomagic.magicdraw.plugins.impl.patterns.PatternInfo;
    import com.nomagic.magicdraw.plugins.impl.patterns.PatternsManager;

    public class MyPatternPlugin extends Plugin
    {
        /**
        * Init this plugin.
        * Register MyPlugin in PluginsManager here.
        */
        public void init()
        {
            PatternsManager.getInstance().addPattern(new PatternInfo(new
MyPattern(),
                                                              new
PatternPropertiesPanel(), new MyPatternProperties(), "Description.html"));
        }

        /**
        * Close this plugin always.
        * @return false always
        */
        public boolean close()
        {
            return true;
        }

        /**
        * @see com.nomagic.magicdraw.plugins.Plugin#isSupported()
        */
        public boolean isSupported()
        {
             return true;
        }
    }
```

The pattern must be registered in the *com.nomagic.magicdraw.patterns.PatternsManager* with the *addPattern(PatternInfo)* method. Also we need to provide *plugin.xml* for this plugin.

```
    <?xml version="1.0" encoding="UTF-8"?>
    <plugin
        id="mypatterns.MyPattern"
        name="My Pattern"
        version="1.0"
        provider-name="No Magic"
        class="com.nomagic.magicdraw.plugins.impl.examples.mypatterns.
MyPatternPlugin">

        <requires>
             <api version="1.0"/>
             <required-plugin id="com.nomagic.magicdraw.plugins.impl.
patterns"/>
        </requires>

        <runtime>
             <library name="mypatterns.jar"/>
        </runtime>
    </plugin>
```

Compile all classes and bundle them into the *mypatterns.jar* file. Also add *Description.html* into this *jar* file. Create the subfolder *mypatterns* in <*your modeling tool install*>/*plugins* and copy *mypatterns.jar* and *plugin.xml* into it.

For more details how to deploy plugin, see Plugins.