Papyrus User Guide Series

About the diagram stylesheets, version 0.9.1

This document is part of a series of documents called, Papyrus User Guides Series, and dedicated to assist the usage of Papyrus. The focus of this volume is to describe the process to customize the Papyrus Diagram Appearance, with the support of Stylesheets.

Camille Letavernier, CEA, LIST, Laboratory of model driven engineering for embedded systems

27/09/2012

Table of Contents

HISTORIC OF THE DOCUMENT

Version #	Contributors <Name>, <Affiliation>	Comments
0.9.1	Camille Letavernier, CEA, LIST, Laboratory of model driven engineering for embedded systems	This is the initial version of the document. It is based on the Papyrus version 0.9.1.

INTRODUCTION

The standard UML specification provides a few hints to represent graphical elements. However, it only focuses on the general representation of these elements (e.g. A Class is a rectangle, with optional compartments for its attributes and operations, and its name should be displayed in italic if the Class is abstract).

Nevertheless, for a better readability (and even esthetic), the user has a certain freedom for changing some graphical properties, such as an Element's color. Until now, in Papyrus, this customization could be done by two complement means:

- Appearance tab (Properties view): change the appearance of the selected element(s)
- Preferences page: change the initial appearance of all newly created element(s)

While this allows defining a custom theme for our diagrams, this mechanism suffers from a few limitations:

- It is not possible to create different categories of appearance for the same semantic Element (e.g. "Blue Class" and "Green class"). We'd have to create "Blue classes" and change some of them to "Green" manually.
- If we wanted to change the current theme, we'd have either to change each object's appearance one by one, or to write an automatic transformation, which could be really complicated.
- It is not easy to export a Theme, as they are stored in the Eclipse preferences. It is almost impossible to ship a ready-to-use distribution of Papyrus with a custom theme.

The release of Eclipse 4 Juno this summer gave us the opportunity to rely on their CSS Engine to support Cascading StyleSheets in Papyrus diagrams. Just like in web pages, the CSS format is used to separate the contents of our diagrams from their appearance. It becomes possible to change the appearance of a whole diagram, or even a set of diagrams, independently of the number of represented elements, in a single click.

INSTALLATION

The CSS Stylesheet support is not shipped in the base installation of Papyrus. However, it can be easily installed from the Papyrus discovery site. Select "Help", "Install Papyrus Additional Components", and check "Diagram Stylesheets".

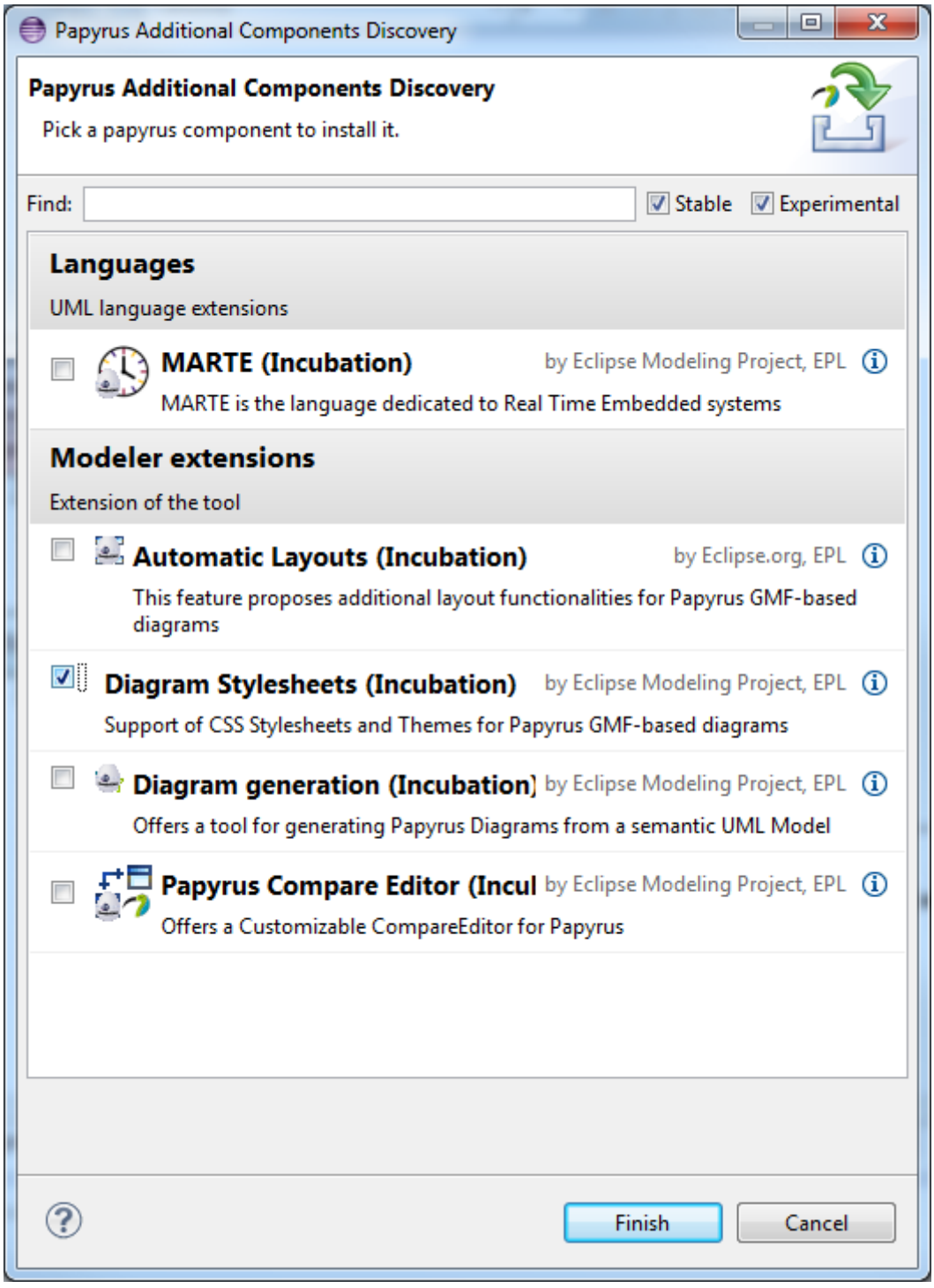


Figure 1 - Installing Stylesheets

Press finish, restart Eclipse, and that's it.

Your diagrams should now look like the following:

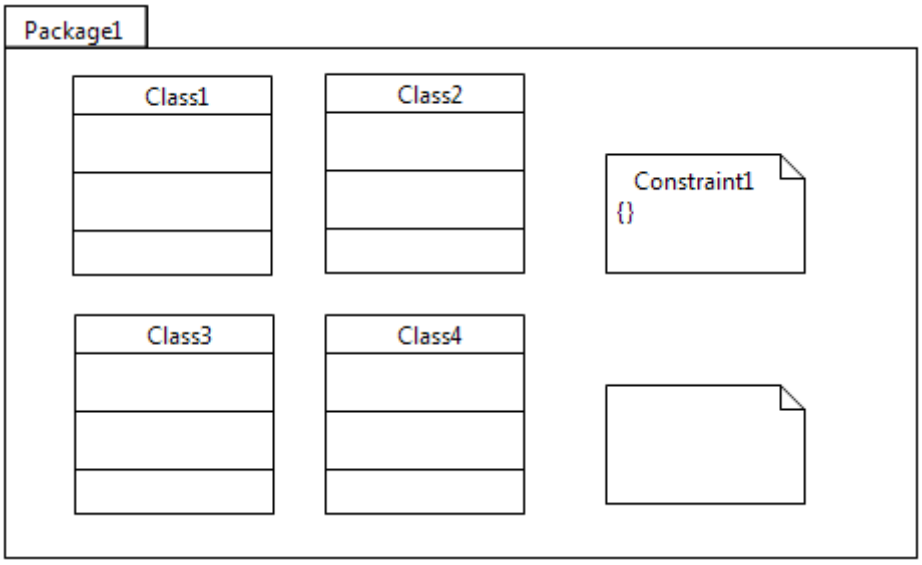


Figure 2 - Without theme

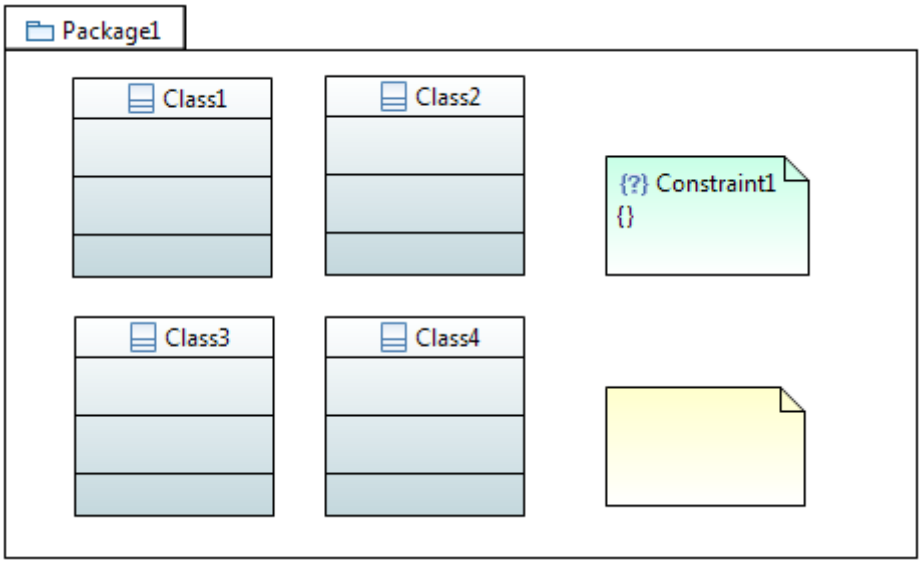


Figure 3 - With CSS Support

Note: If you opened a diagram which has been created without the CSS Support, it may still have a custom appearance, e.g. without Element icons. This might or might not be a problem, as this may conflict with the CSS Theme. You can reset all custom appearances by pressing the "Default style" button in the Style tab of the Diagram's properties view.

USING STYLESHEETS

To define a custom style, the first step is to create a Stylesheet. The creation is really straightforward: a Stylesheet is a text file with the .css extension. You can create such a file using the “New > File” wizard.

Then, you need to associate this file with your diagram. Select a diagram, and then go to the Style tab of the properties view, and add a new “Diagram style sheet”. There isn’t currently any style sheet available from the diagram, so you need to create a new “Stylesheet reference”.

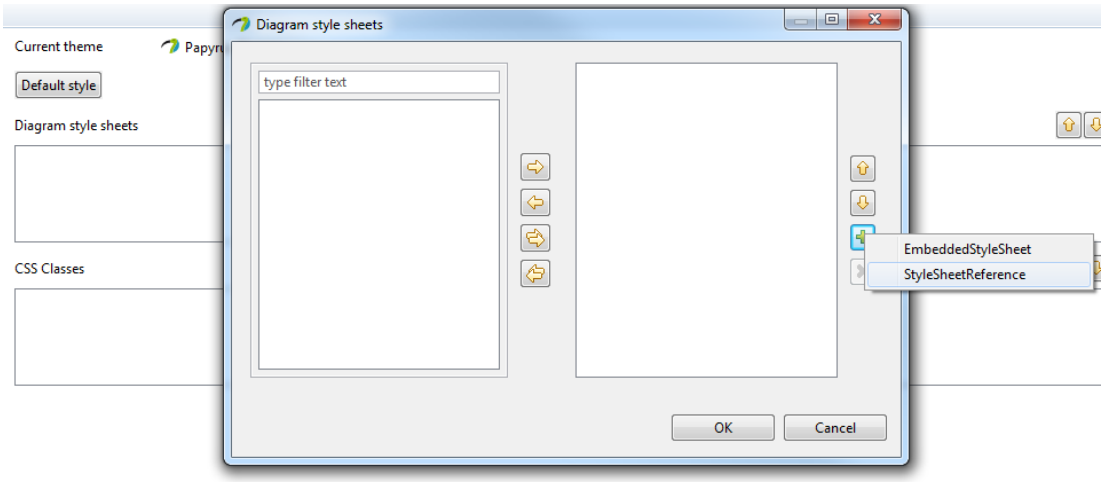


Figure 4 - Import Style sheet

Browse your workspace to find your css file, and then validate. Your stylesheet is now associated to your diagram.

EDITING A STYLESHEET

Papyrus Stylesheets are stored in text files with the ".css" extension (e.g. stylesheet.css).

Manually: The CSS Syntax

Syntax elements

The CSS syntax is a list of rules, which associates selectors and properties.

- The selector: describes the condition which has to be fulfilled for a rule to be applied
- The properties: describes the appearance of the elements matching the rule

Eclipse implements the 2.1 version of the CSS syntax. The available selectors are:

- *: matches all elements
- ElementType: Matches the elements of type "ElementType"
- .myStyle: Matches the elements on which the style "myStyle" is applied
- #myID: Matches the element with the CSS ID "myID". The CSS ID must be unique among a diagram. To avoid confusing the users, and because this selector doesn't make much sense on diagrams, this selector has been disabled in Papyrus.
- [property=value]: Matches the elements which have a property "property" with the value "value".

The pseudo-selectors are currently not used in Papyrus.

These selectors can be combined using one of the following combinators:

- Selector1 Selector2: Descendant selector. Matches the elements which match Selector2 and are contained in an element matching Selector1
- Selector1 > Selector2: Child selector. Matches the elements which match Selector2 and are *directly* contained in an element matching Selector2
- Selector1 + Selector2: Sibling selector. Matches the elements which match Selector2 and are immediately preceded by a Sibling element which match Selector1.
- Selector1, Selector2: Alternative selector. Matches elements matching either Selector1 or Selector2

A rule also contains a list of properties, in the form property:value;

In Papyrus, the selectors apply to semantic elements and properties (i.e. UML Elements), and the properties apply to the appearance properties (i.e. GMF Appearance properties). A few custom properties (Independent from the GMF Appearance model) are also available.

Examples

```
* {
	fillColor: red;
	fontColor: #0000FF;
}
```

Meaning: all elements should be filled in red, and all texts should be blue (RGB Color #0000FF)

```
Class {
	gradient: white vertical;
	elementIcon:true;
	shadow:true;
	qualifiedNameDepth:full;
}
```

Meaning: this style applies to all UML Classes. They will have a white and vertical gradient. The Element icon and the shadow will be displayed, as well as their fully qualified name.

```
ClassDiagram Class>Property {
	fontColor: red;
}
```

Meaning: The properties directly contained in a Class, which is drawn on a ClassDiagram (Either as a root element of the diagram, or in a Package) will be displayed in red.

With the Papyrus tooling.

Papyrus also provides a tool to generate a CSS Rule from an element displayed in the diagram. You need to create a stylesheet before you can edit it with the Create style tool, and it is recommended (Although not required) to associate this style sheet with your current diagram.

To use it, create an element and change its appearance via the “Appearance” tab of the properties view. Then, right click on the element, and select “Format”, “Create a new style”. The dialog box contains three tabs:

- Conditions: The conditions under which the style will be applied (Corresponds to the “CSS Selectors”)
- Properties: The graphical properties to export to the style sheet
- Stylesheet: The style sheet to edit

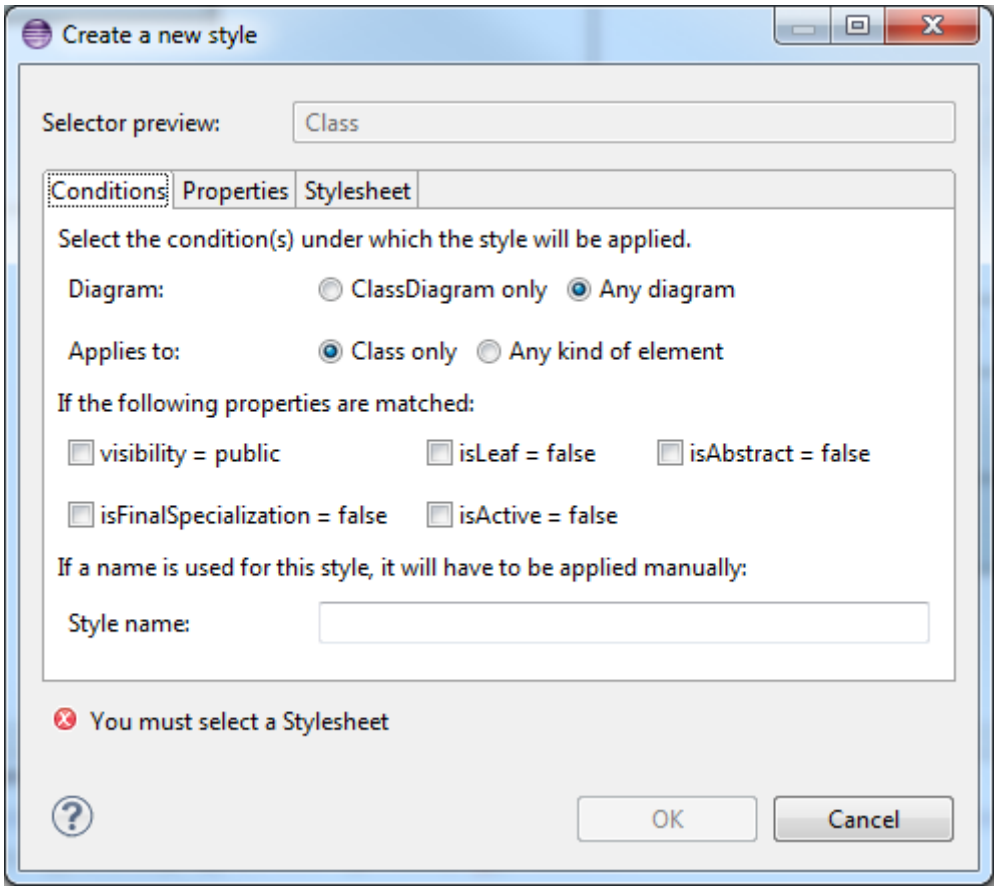


Figure 5 - Create style dialog

In the Stylesheet tab, select your stylesheet:

- If you have already associated the style sheet to your diagram, you can use the "Applies stylesheet" button.
- Otherwise, you need to select "External stylesheet", and browse your workspace to retrieve the file. Note that in this case, as the edited stylesheet might not be applied to your diagram, you won't be able to see the changes in the current diagram.

In the conditions tab, there are three groups of options:

- The kind of element to which the style applies (e.g. the style will be applied to "Class" elements, or to any kind of elements). It can be restricted to a specific diagram.
- You can add restrictions on the current value(s) of the element (e.g. the style will be applied to concrete classes). Only primitive type and enumerated attributes are supported here.
- The style name is used to apply a style manually.

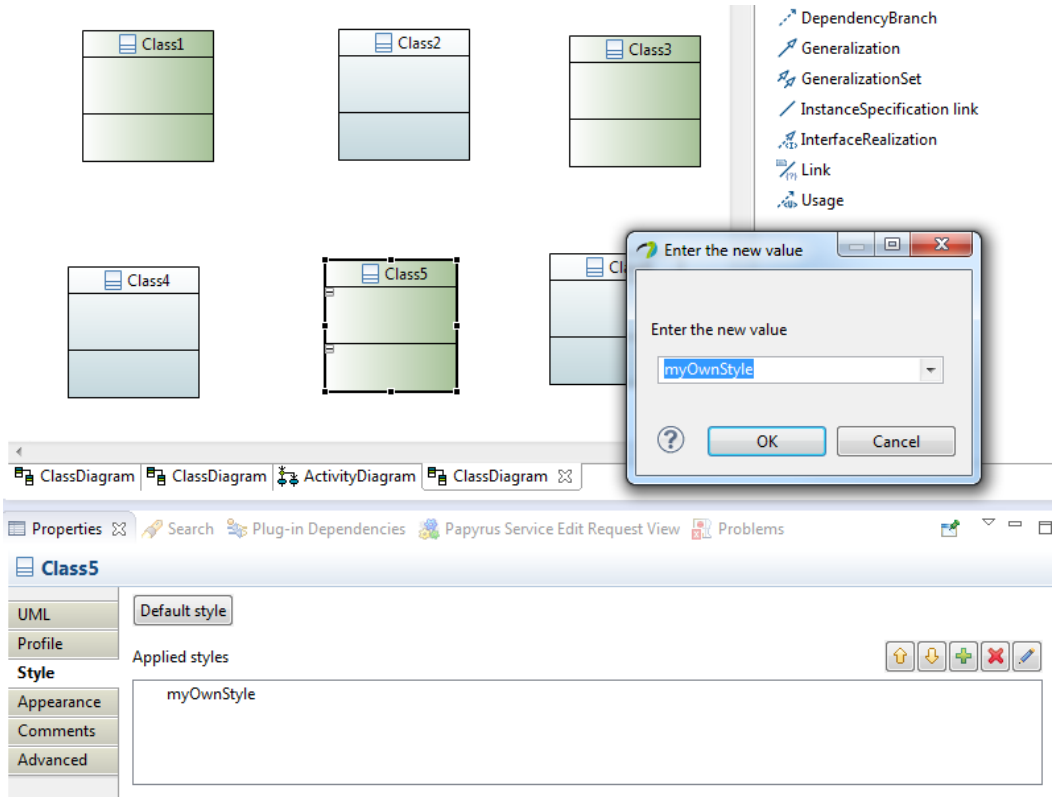
The properties tab contains all the graphical properties that can be applied to the selected object. It is used to choose the graphical properties to apply on the objects matching the conditions.

MANIPULATING STYLES

The computed styles will be applied automatically. For example, if your stylesheet tells that all your abstract classes must be red, the classes color will change automatically according to the value of their "isAbstract" property.

In some other cases, however, you may want to specify manually the style that should be applied to your objects. For this case, you can use the "Applied styles" property in the "Style" tab of the properties view.

```
.myOwnStyle {
	fillColor: #A6C198;
	gradient:white horizontal;
}
```



Once a style is applied on an object, the object will be refreshed each time the style definition changes. For example, changing the style "myOwnStyle" to the following declaration:

```
.myOwnStyle {
	fillColor: #E3A49C;
	gradient: #FFFFFF horizontal;
}
```

All the green classes will become red.

Deploying a CSS Theme

Once you have defined your style sheet, you may want to export it to create a new theme (Or to extend an existing theme). To do so, Papyrus provides an extension point: org.eclipse.papyrus.infra.gmfdiag.css.theme.

This extension point needs two entries: a Theme Definition (ID, Label and icon of the Theme), and Theme contributions (A set of style sheets which will compose the Theme). More than one plug-in can contribute to the same Theme, which makes it possible to extend a Theme.

The Theme can be modified from the Papyrus preferences page “CSS Theme”. Only one theme can be activated at the same time. A Theme applies to all Papyrus Diagrams from the workspace.

LIMITATIONS & ISSUES

There are currently a few identified limitations to the Stylesheet framework, as well as a few minor bugs.

Selectors

Currently, it is not possible to handle complex selectors in the Papyrus stylesheets.

Selector inheritance

It is not possible to apply a style to all descendant of a given Metaclass. For example, a Behavior will not inherit styles from a Class, although a Behavior actually is a Class.

Profile/Stereotype support

It is not possible to restrict a selector to an applied stereotype. There is currently no way to specify that a style should only apply to classes on which the *SysML::Blocks::Block* stereotype is applied.

Labels and compartments

It is currently not possible to customize labels from the Stylesheets (e.g. show/hide type for all properties). It is not possible to show/hide or collapse compartments either (e.g. hide the nested classifier compartment in a Class).

Events support

Selectors based on user events are not yet implemented. For example, the CSS specification defines the *:hover* pseudo-selector to represent an element hovered by the mouse, which is not yet supported in Papyrus.

Stylesheets

Model stylesheets

Currently, stylesheets can only be applied to a Diagram. It is not yet possible to apply a stylesheet on a Model.

Workspace stylesheets

The only way to define a workspace theme is through the use of a plug-in which defines a new extension. It is not yet possible to create a new local theme from the preferences page.

Bugs

The following bugs have been identified:

372322: [Diagram - Refresh] The refresh action is not correctly binded to F5
https://bugs.eclipse.org/bugs/show_bug.cgi?id=372322

386574: [CSS - Refresh] Only the active diagram is refresh when the stylesheets change
https://bugs.eclipse.org/bugs/show_bug.cgi?id=386574

390534: [CSS - Appearance] Undoing graphical changes results in unexpected behavior
https://bugs.eclipse.org/bugs/show_bug.cgi?id=390534