

# **Intrex Professional**

# **Intrex Compact**

RELEASE 5



**Editors**

# Contents


<b>1. Internal and external Editors.....</b>	<b>4</b>
<b>2. Stylesheet Editor .....</b>	<b>5</b>
<b>3. JavaScript Editor .....</b>	<b>6</b>
3.1. Enter Script.....	6
3.2. Libraries .....	7
3.3. Setting Markers.....	10
<b>4. Groovy Script Editor .....</b>	<b>11</b>
4.1. Data Transfer – Groovy Script Data Source .....	11
4.2. Functions for Data Transfer.....	11
<b>5. Velocity Editor .....</b>	<b>12</b>
<b>6. XML Editor .....</b>	<b>13</b>
<b>7. Filter Editor .....</b>	<b>13</b>



## Copyright






This document is in all its parts protected by copyright. All rights are reserved, especially the right of translation, presentation, reproduction, and duplication. Regardless of the accuracy of the text, images, and programs neither author, publisher or translator may take judicial or other responsibility for possible errors and their consequences.

The conventional names, trade names, product designations, etc. reproduced in this work may, even without special designation, be trademarks, and as such may be subject to legal regulations.

## Writing Conventions

In this handbook, text passages will be displayed in *italics* when they refer to settings in the displayed dialogs. Menu items that are available in context menus can, in addition, always be selected from the main menu. Main menu items will not be described if they are not available in the context menu. A description of the general main menu items can be found in the  *Portals* handbook. Programming code in the text will be displayed in the Courier font. Context menus can be opened by clicking with the right mouse button on the described element.

In the following, `<intrexx>` refers to your Intrexx installation path; under Windows, for example, this is usually  `c:\intrexx\`. On Linux, the normal install path is  `/opt/intrexx/`. The following symbols will be used for designation of special kinds of information:

-  Information
-  References to an Intrexx handbook
-  Directories
-  URLs
-  Click on buttons

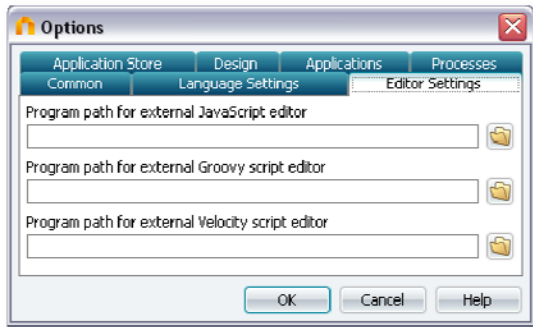
## Background Knowledge


To understand this document, basic knowledge of the *Design*, *Applications*, and *Processes* modules are required. You can find supplemental information in the handbooks of the same names.

## 1. Internal and external Editors

In Intrexx, there exist internal editors for use in the integration of JavaScript, Groovy, and Velocity into your portal. With them, you can compose and administer your own code comfortably and easily. Alternatively, you have the ability to integrate external editors if you wish as well.


This ability exists in a central, module-spanning location: in the editor settings, which you can reach via the menu item *Extras / Options*.



Click on  *Browse* here in order to enter the program path to your external editor.

If a file (such as the JavaScript file for the application) is opened in an external editor, the script button cannot be used as long as the external editor is running.

The external editor writes first to a temporary file, which will be applied to the application only after the editor is closed.

-  Please do not change the temporary file name. Use the "Save" function in the external editor.

Here are some examples for the setup of an external editor:

### **Notepad++**

```
<Program path>\notepad++.exe -multiInst ${file}
```

### **Editplus**

```
<Program path>\editplus.exe ${file}
```

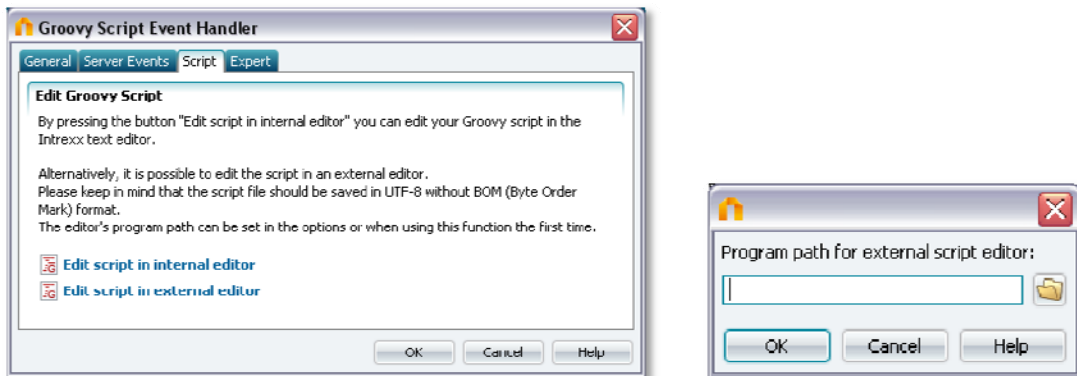
### **gedit**

```
/usr/bin/gedit --new-window ${file}
```

Entering the parameter `${file}` is usually not required; however, it can be entered if the external editor expects or supports additional parameters after the file name.

In addition to the possibility of entering editor program paths as defaults in the options, external editors can also be set up at the moment in which they are required, such as in a particular location in a process, or for entering JavaScript in the *Applications* module.

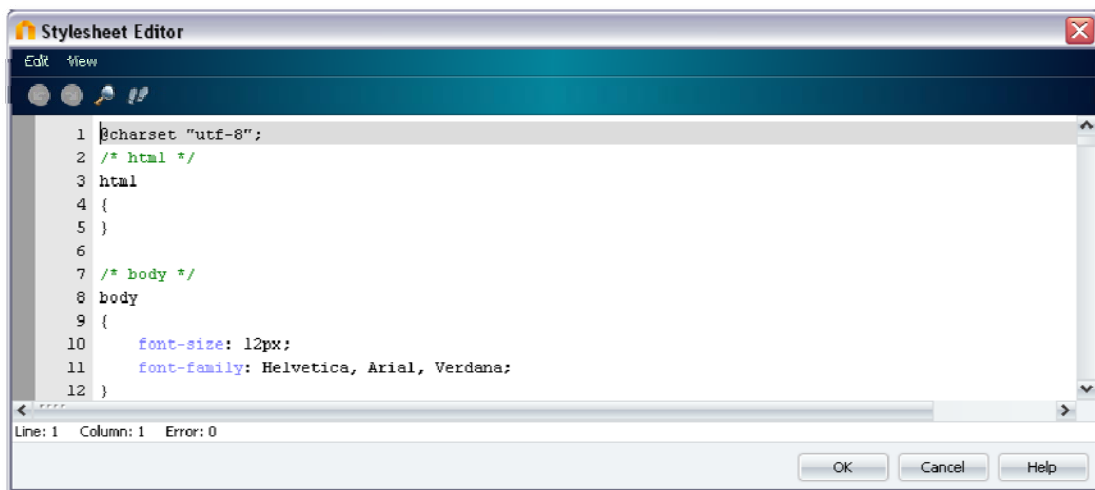
In addition, external editors can be entered in the place and time that they are required, such as at a specific place in a process or for the entry of JavaScript in the *Applications* module.





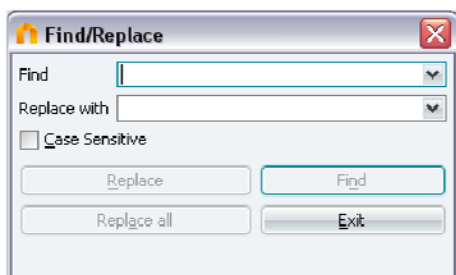
Here as well, simply enter the path to the desired editor.

## 2. Stylesheet Editor

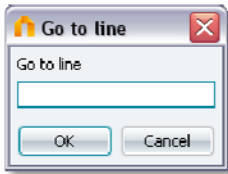
In the *Design* module, you can reach this editor via the menu item *Edit / Edit Stylesheet*.



Here you have the ability to make changes directly to the CSS file underlying the layout. The changes will be applied by clicking on  *OK* and take effect when the layout is published ( *Design*). By selecting the menu item *Edit / Search*, a dialog will be opened, in which you can search and for or search and replace specific terms.

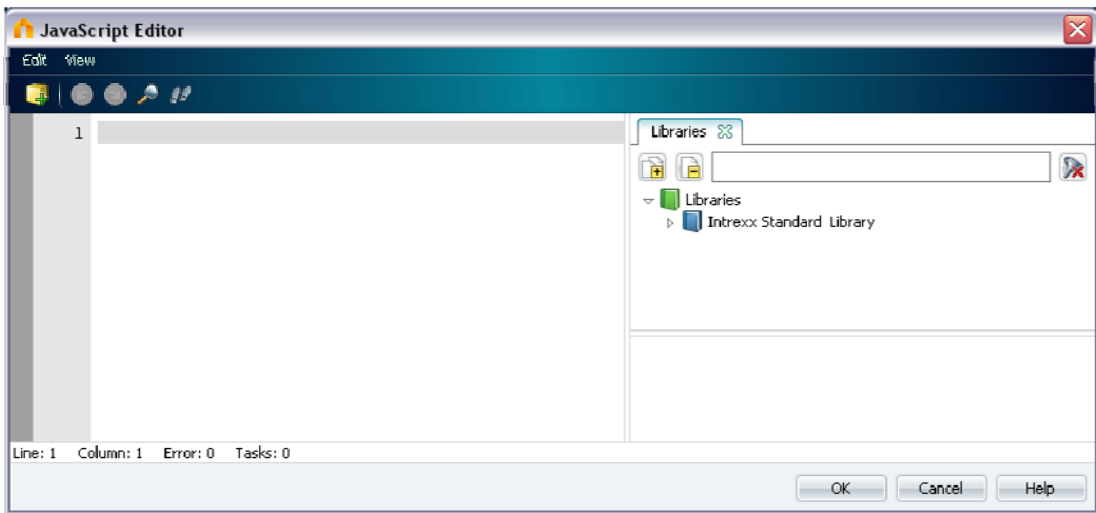


With *Edit / Go to line*, you can jump directly to lines in the stylesheet. Simply enter the desired line number here.



### 3. JavaScript Editor

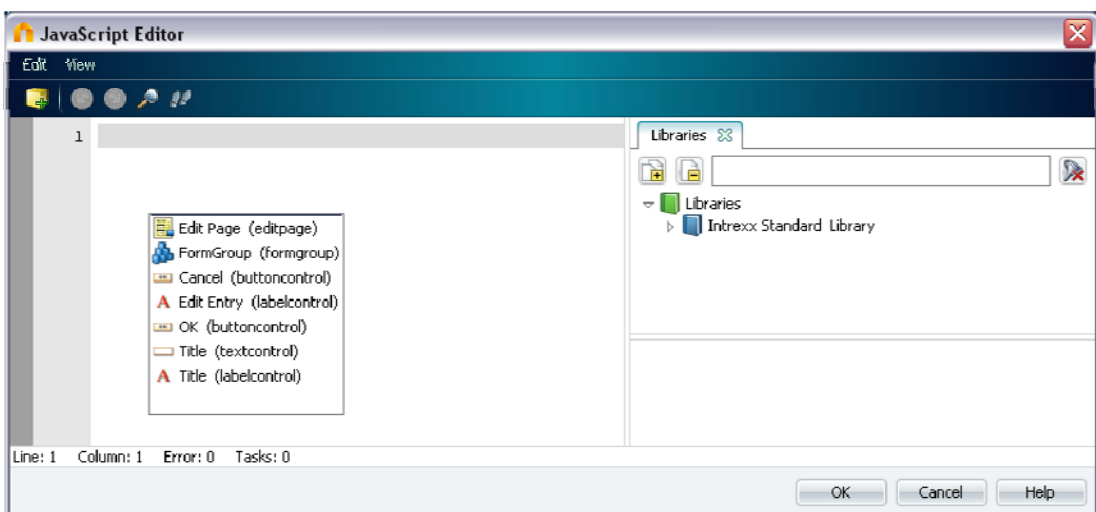
In the *Applications* module, you can find this editor via the *Script* tab of the properties dialog of many application elements ( *Applications*).



Here you can expand the functionality of applications however you wish with JavaScript.

#### 3.1. Enter Script

Elements that are to be read out or controlled via a script can be referenced quite easily. With the menu item *Edit / Insert*, a list will be opened of all edit and view elements that are located on the current page.



Select the element that is to be used in the script. The following programming code will then be automatically inserted into the script editor:

```
getElement("GUID of element")
```

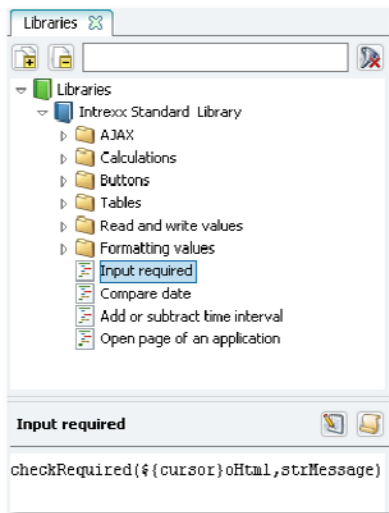
Create your references to the HTML object with the syntax



```
var NameOfElement = getElement("GUID of element");
```


As with editing stylesheets in the script editor, you also have the ability here to search for and to replace specific terms, and jump to specific lines in the script, via the *Edit* menu.


Via the *View* menu, you can reach the syntax check here, which will draw your attention to incomplete bracket pairs and other places in the script where an error is likely. The libraries can also be shown or hidden here. Via the *View / Full Screen* menu or the key *F11* the editor can be maximized.

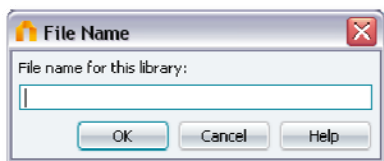
## 3.2. Libraries



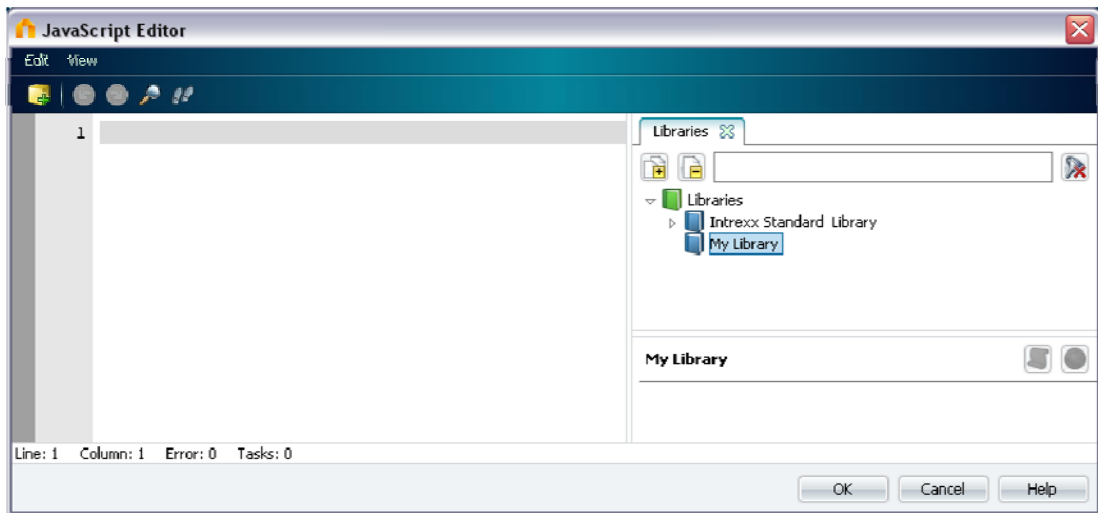
In the right-hand area of the editor you can find the libraries. From the   symbol bar, you can show or hide the entire tree, so that you can view or not view the subordinate levels. In the search field, you can search for specific keywords, and then remove this filter again when it is no longer needed.

The  *Intrexx standard library* contains a collection of useful functions that you can insert quite easily into the script that you enter into the editor.

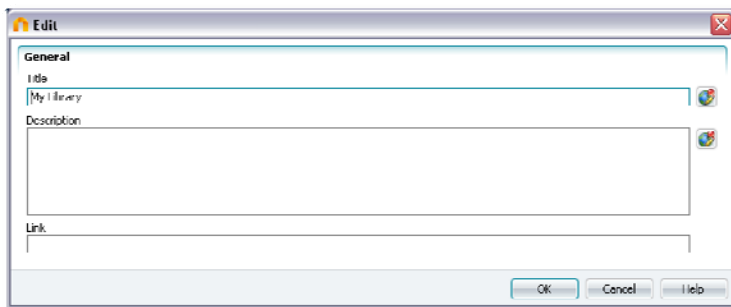
In addition to the *Intrexx standard library*, you can create additional, individual libraries with the functions that you require frequently. You can reach this feature via the context menu item *New / Add library* for the  *Libraries* entry.




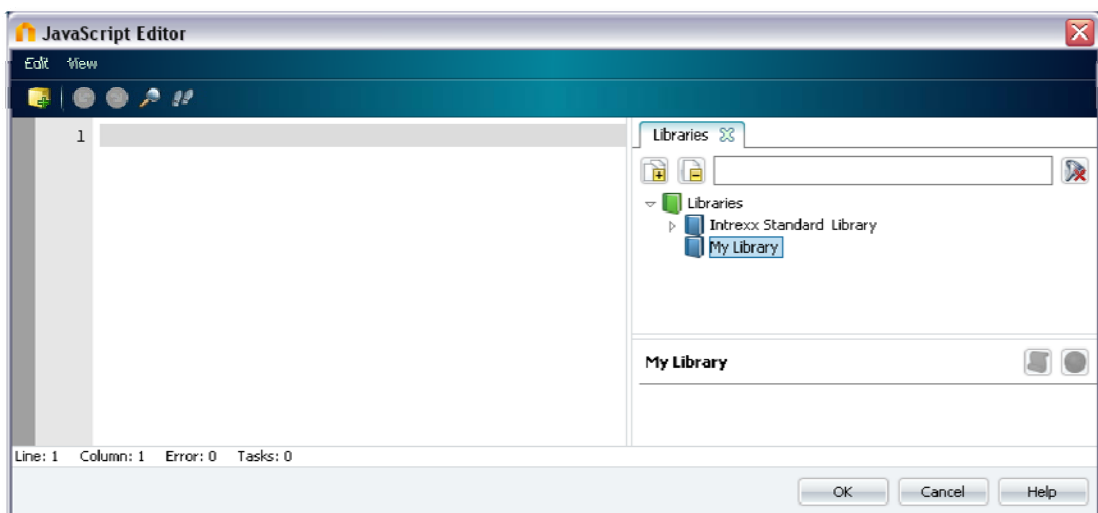
Enter a name for the library here.




Via the context menu item *Edit*, a dialog will open in which you can change the properties of the new library.

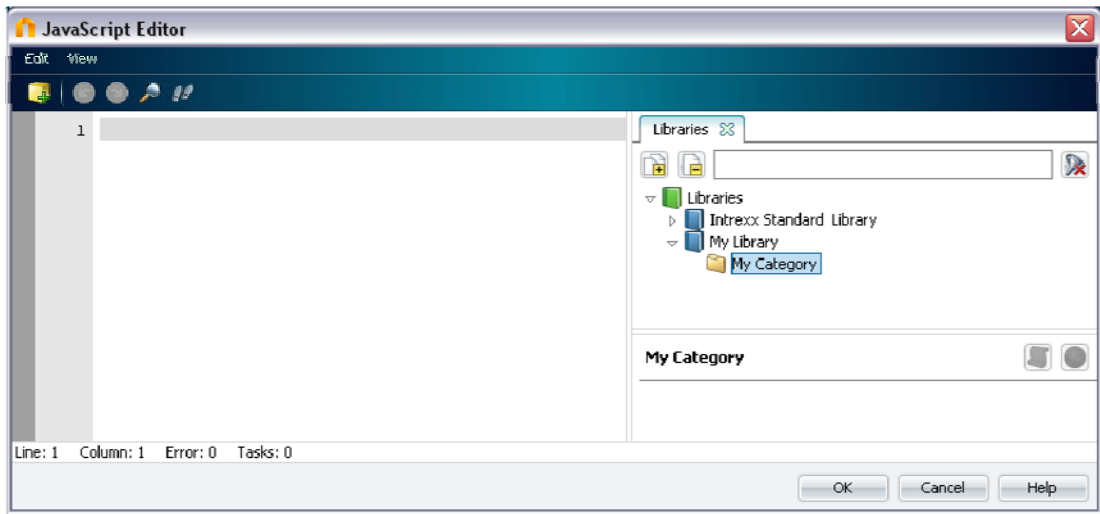


The title and description of the library can be changed here. In the *Link* field, you have the ability to enter a URL to a website, on which you have entered information on the library. Click  *OK* to continue.



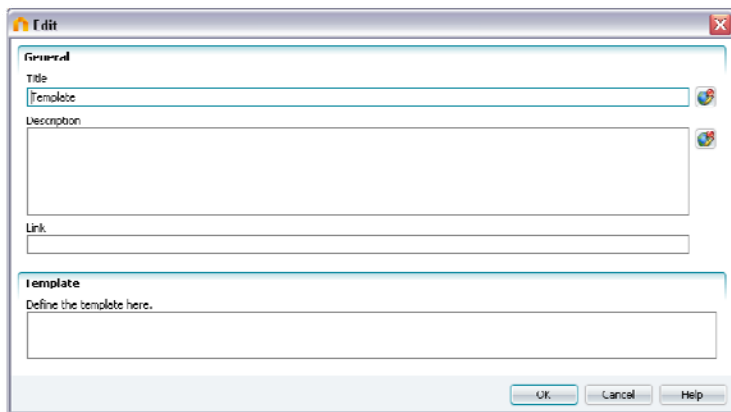
If you have entered a URL to a page of your choice, the  globe symbol will be shown in the preview window in the lower right. Clicking on this symbol will open the page in your browser.


Via the context menu item *New / Insert category*, categories can be created for the organization of your library.

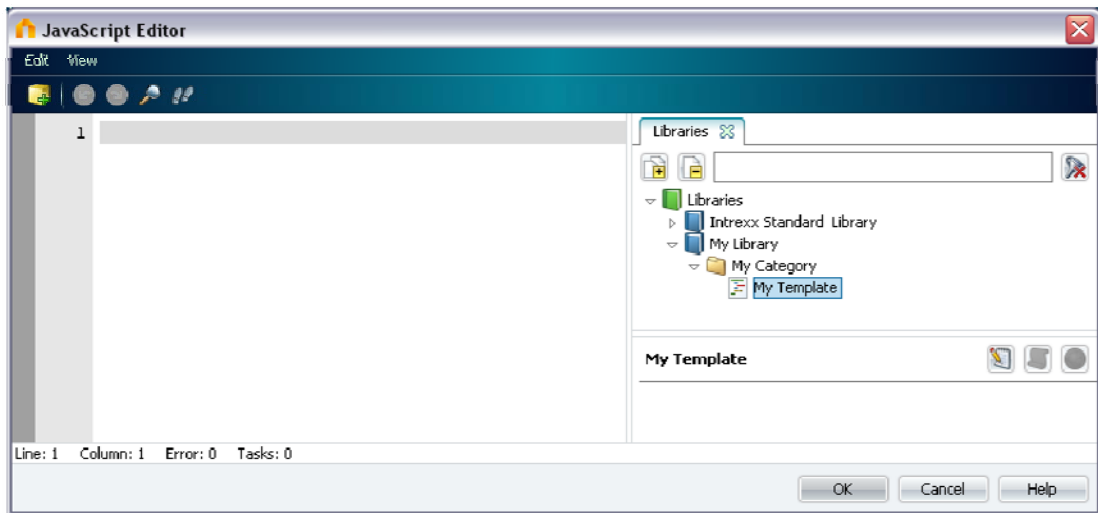


As with libraries, for a category you can also *Edit* the title, description, and a link to the website of your choice via the context menu

Functions that you use frequently can be administered here in the so-called *Templates*. You can create a template via the context menu of a category with *New / Add template*.

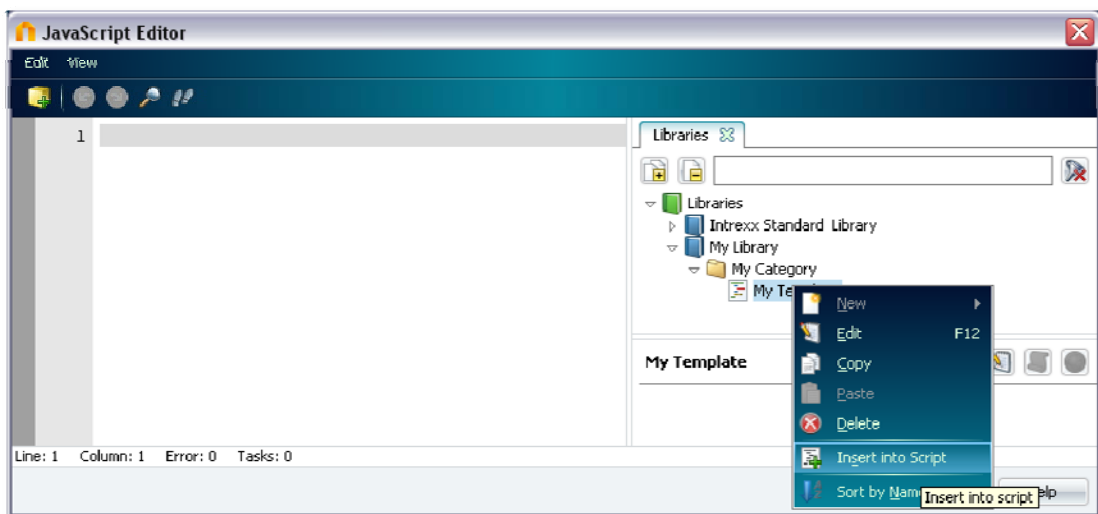


Here you can change the *Title* and enter a description. In the *Template* area, code can now be entered. What you enter here can be inserted later wherever you wish into the script that you composed in the main area of the editor. Click on  *OK* to save the template.



If you highlight the template in the tree structure, the script contained in it will be shown in the preview window in the lower right. From there you can open the dialog in which the script was composed by selecting *Edit*. *Description* will show the description of the template.

The script from the template can be inserted into the existing script in the left-hand area of the editor at the current cursor position via the context menu item *Paste into script*.



When your script is complete, it can be saved by clicking *OK*. All changes will take effect when the application is published.

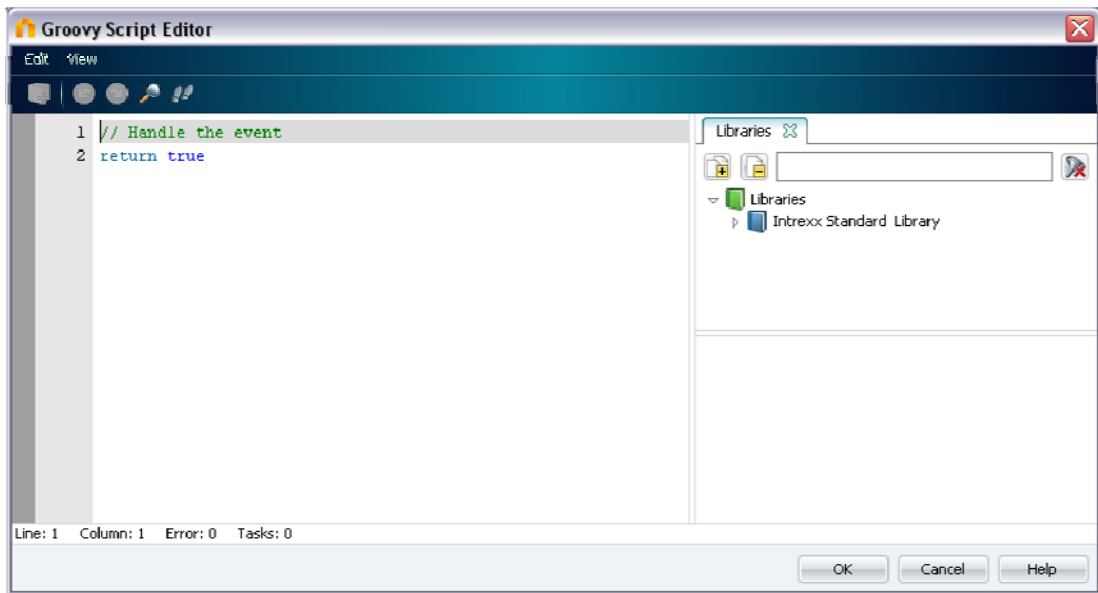
- User-defined libraries can be exported via the context menu of the corresponding library into a directory available to the server of your choice. This allows individually created libraries, for example, to be implemented in other portals as well.

### 3.3. Setting Markers

When opening an application or process, JavaScript and Groovy will be searched for comments that contain the character string `//FIXME` or `//TODO`. Hits will be noted in the messages area with this icon. With a double click on a hit the corresponding internal editor will be opened. Here again the comment will be marked with this icon.

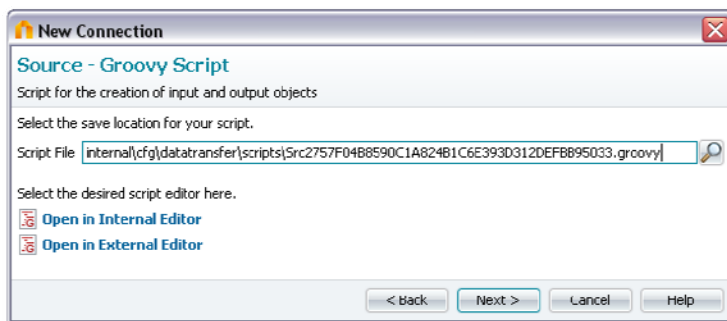
## 4. Groovy Script Editor

You can find the Groovy Script Editor in the *Integration* module in a data transfer, if Groovy script is selected as a data source or target (🔗 *Integration*). Additionally, it can be reached from the *Processes* module (🔗 *Processes*).



As with the use of the JavaScript editor, libraries, categories, and templates are available in the right-hand area of the screen. The Intrexx standard library offers a multitude of objects and functions that you can insert into your existing script here as well.

### 4.1. Data Transfer – Groovy Script Data Source



First, select the place where the *Script File* will be saved. The directory 📁 *internal\cfg\datatransfer\scripts\* will be automatically suggested for the save location of the new script file. In addition, Intrexx will generate a unique file name, consisting of the character string *Src* and a GUID with the ending *.groovy*. If you want to use a different location to save the script, please note that the location must be reachable by the server. In addition, the selected file name must be unique in the portal.

### 4.2. Functions for Data Transfer

If you open the Groovy Script Editor when setting up a data transfer in the *Integration* module, the following script will be entered as a preset, if Groovy is selected as source or target:

```

import de.uplanet.lucy.server.datatrans.IDataSet
import de.uplanet.lucy.server.datatrans.IDataObject
import de.uplanet.lucy.server.datatrans.table.DefaultDataRecord
/**
 * Initialize the data source here.
 */
void open()
{
}
/**
 * Called, when the data source is no longer needed. Cleanup resources here.
 */
void close()
{
}
/**
 * Returns the data set, from which the data will be read when importing from this
data source.
 * @return data set
 */
IDataSet getDataSet()
{
    return new DataSet()
}
/**
 * This class is the data source specific data set implementation.
 */
class DataSet implements IDataset
{
    int i = 0;

    /**
     * This method is called to retrieve a data object (for instance some kind of
data record).
     * When no data object is available (anymore), return null here.
     * @return the next data object or null
     */
    IDataObject next()
    {
        if (i > 0)
            return null;

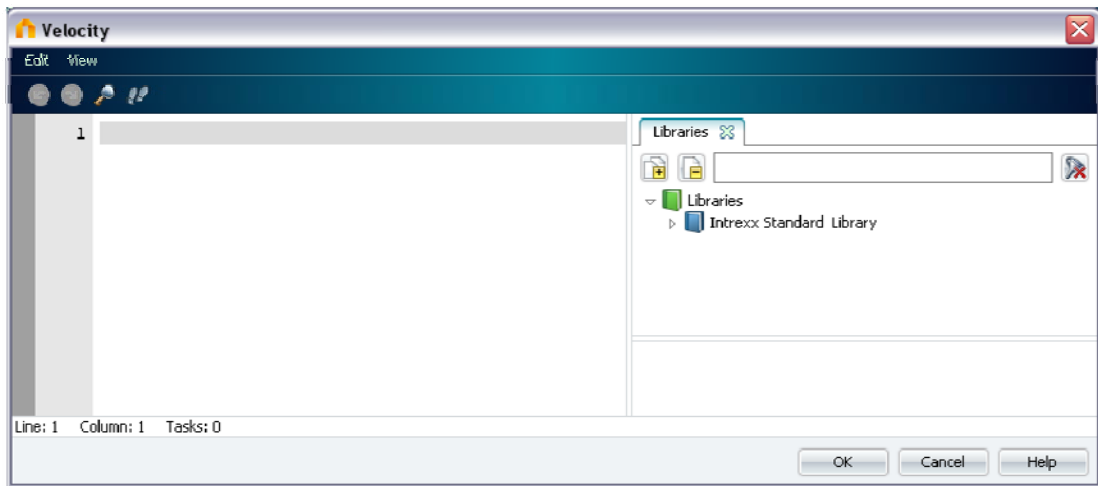
        i++;

        // Sample construction of a default data record
        def l_record = new DefaultDataRecord(null,
["firstField":java.lang.String,"secondField":java.lang.String])
        l_record.setValue("firstField", 1)
        l_record.setValue("secondField", "SampleText")
        return l_record
    }
    /**
     * Closes the data set.
     */
    void close()
    {
    }
}

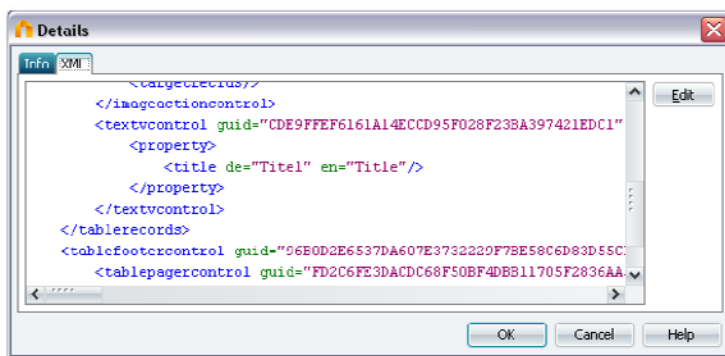
```

## 5. Velocity Editor

The Velocity Editor can be reached from the *Processes* module. If you define an eMail action, you can compose messages depending on the context.



## 6. XML Editor



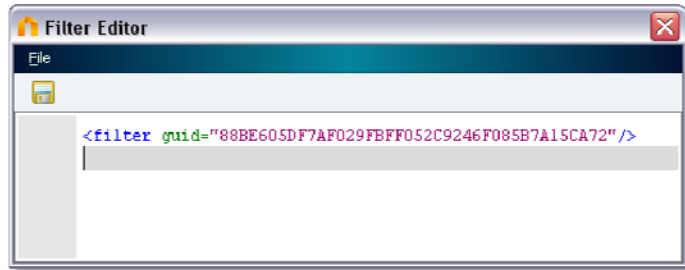
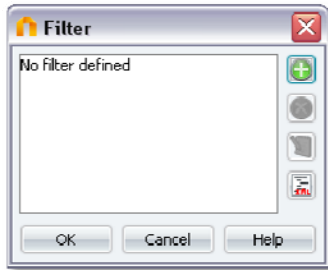
In the *Applications* module, you will find the *XML* tab in the *Details* dialog, assuming expert options have been activated (🔧 *Expert*). The XML of the corresponding element will be shown here. By clicking 🖋️ *Edit*, its attributes can be changed.


After clicking 📄 *OK*, the changes will be applied to the element. The changes will be published to the XML, and thereby taken into effect for the user in the browser, only once the application is saved.

- 📘 Portals can, under certain circumstances, be made nonfunctional if incorrect settings are entered in the XML. Please note that United Planet cannot offer support if this occurs.

## 7. Filter Editor

In the *Applications* module, 🗑️ filters can also be directly edited in the XML, if expert options are active (🔧 *Expert*). You can find filters, for example, in the properties dialog of the application elements table or selection list (🔧 *Applications*). Click on 📄 *Expert* in the filter dialog in order to open the filter editor.



The source text of the filter will be shown in the filter editor and can be further edited. The changes will be applied by clicking on  *Save filter and close*. They will take effect only after the application is published.