Geomajas Client GWT2
Print plugin documentation

Geomajas Developers and Geosparc

Geomajas Client GWT2 Print plugin documentation

by Geomajas Developers and Geosparc

Version 2.4.2

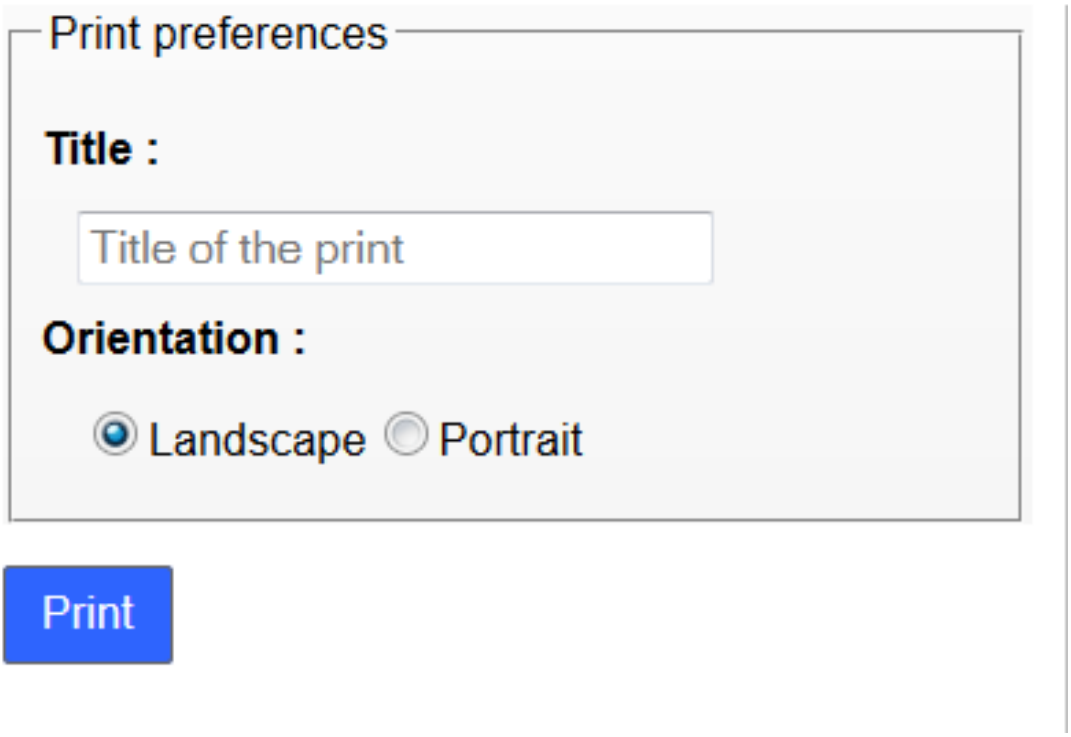
Table of Contents

Chapter 1. Introduction

1. What does this plugin do?

This plugin enables printing the content of a map to pdf. Some extra elements can be added to the print (scale, legend, ...).

The default widget is a panel that allows the user to give a title and an orientation of the map. After clicking the "print" button, a pdf file will be created. By default this pdf file will be shown in a new browser window.



2. Maven configuration

In order to use this plug-in in combination with the client GWT2, the following Maven dependency is required:

```
<dependency>
<groupId>org.geomajas.plugin</groupId>
<artifactId>geomajas-client-gwt2-plugin-print-impl</artifactId>
<version>2.4.2</version>
</dependency>
```

This in turn includes a dependency on the Geomajas server project.

Chapter 2. How-to Print Widget

This chapter shows how to create and use the Print Widget.

1. Default Print Widget

The default Print widget is class `org.geomajas.gwt2.plugin.print.client.widget.PrintWidget`. The following code creates the Print Widget. Map and application specifications are required beforehand.

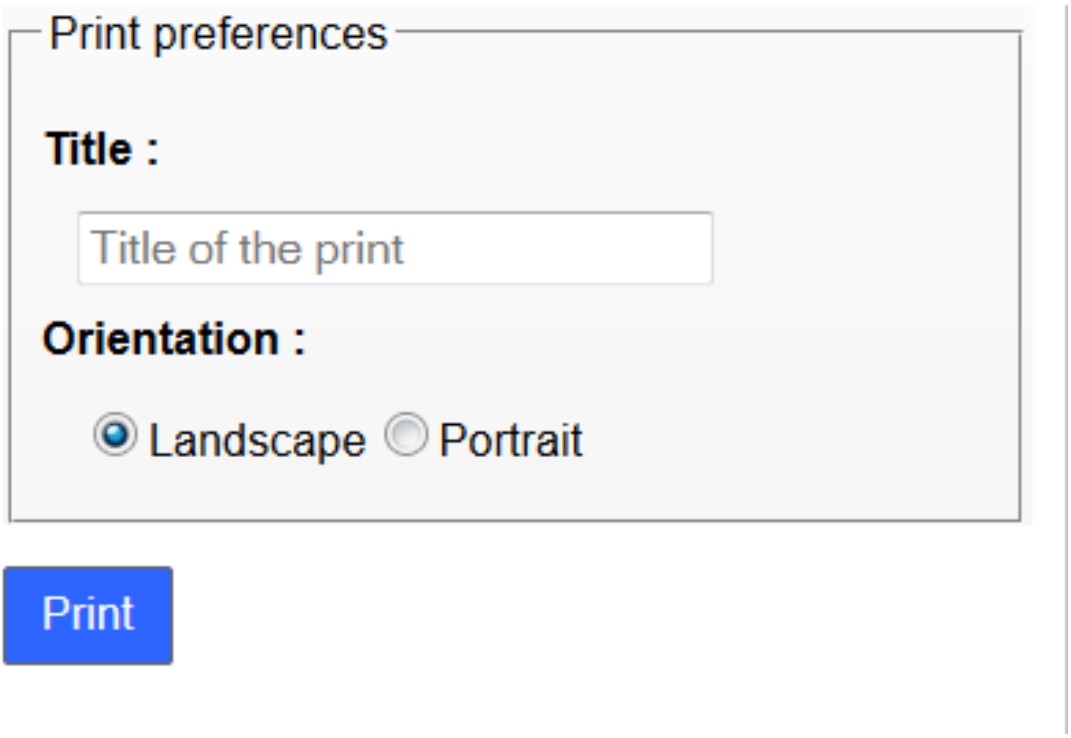
```
// MapPresenter mapPresenter : defines the specific map
// String applicationId : id of the application to build, server-side configura
PrintWidget widget = new PrintWidget(mapPresenter, applicationId);
```

2. Print Widget with custom view.

The view of the print widget can be customized by creating a custom class that extends `PrintWidgetView`:

```
PrintWidgetView customView = new CustomView();
PrintWidget widget = new PrintWidget(getMapPresenter(), applicationId, customVi
```

Some useful views are available in the plugin. The `BasePrintPanel` is the default view, containing title and orientation request.

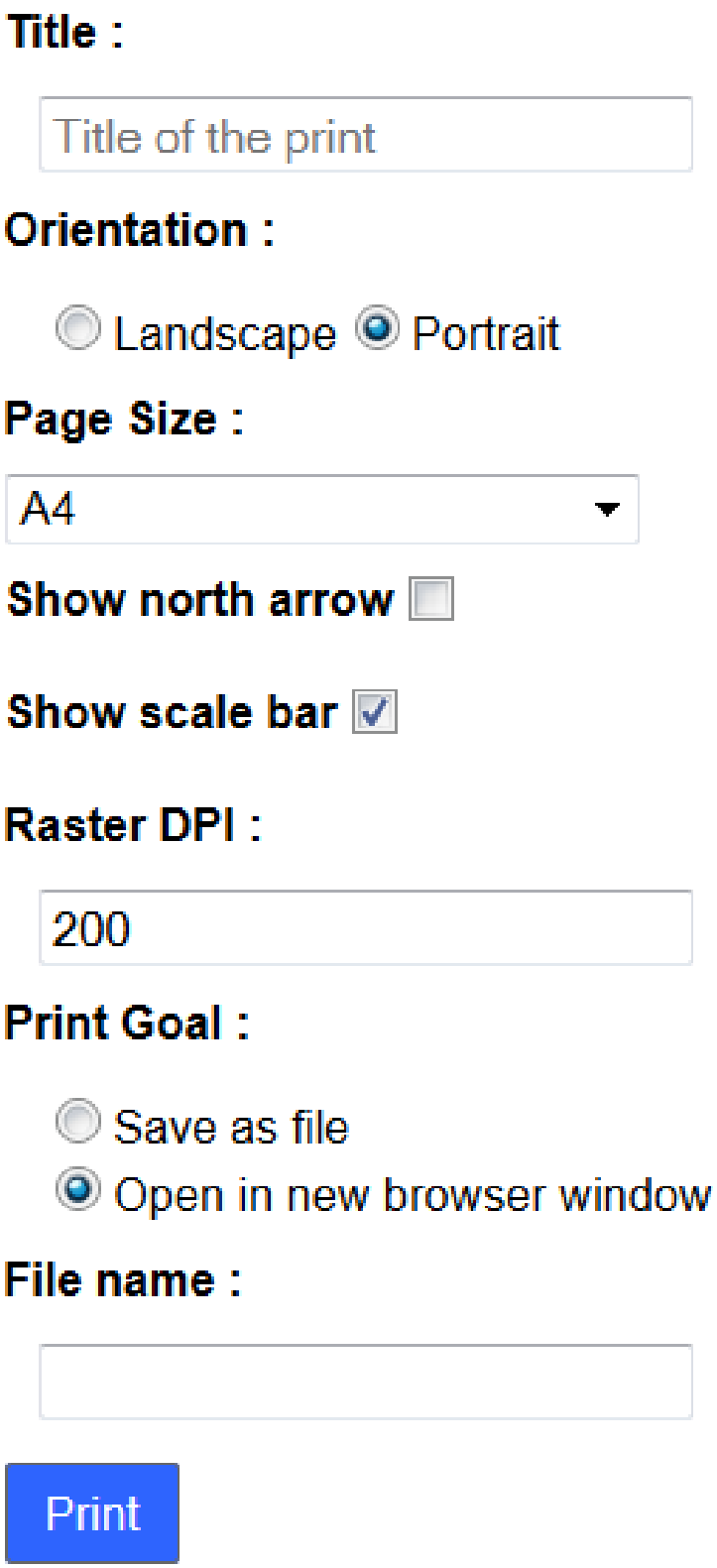


The `OptionsPrintPanel` is a view where the different options can be activated or not. By default, no option is active. The code below activates all options. Make sure to call the method `createViewBasedOnConfiguration`!

```
OptionsPrintPanel optionsPrintPanel = new OptionsPrintPanel();
// activate all options
optionsPrintPanel.getOptionsToShowConfiguration().setShowTitleOption(true);
optionsPrintPanel.getOptionsToShowConfiguration().setShowLandscapeOption(true);
optionsPrintPanel.getOptionsToShowConfiguration().setShowPageSizeOption(true);
```

```
optionsPrintPanel.getOptionsToShowConfiguration().setShowWithArrowOption(true);
optionsPrintPanel.getOptionsToShowConfiguration().setShowWithScaleBarOption(tru
optionsPrintPanel.getOptionsToShowConfiguration().setShowRasterDpiOption(true);
optionsPrintPanel.getOptionsToShowConfiguration().setShowPostPrintActionOption(
optionsPrintPanel.getOptionsToShowConfiguration().setShowFileNameOption(true);
// recreate (redraw) the view.
optionsPrintPanel.createViewBasedOnConfiguration();

PrintWidget printWidget = new PrintWidget(mapPresenter, applicationId, optionsP
```



Both `BasePrintPanel` and `OptionsPrintPanel` extend the abstract class `DefaultDataProviderPrintWidgetView`. This is an implementation of the `PrintWidgetView` that contains a `DefaultPrintRequestDataProvider` implementation.

The `DefaultPrintRequestDataProvider` contains setters and getters fot the default values of data that a gui user does not need to specify, but that a print builder still requires. For example, if an `OptionsPrintPanel` does not show the rasterDpi option, the print builder still need to have a rasterDpi value for creating the print.

The example below shows how you can overwrite the default values of the `DefaultDataProviderPrintWidgetView`

```
DefaultDataProviderPrintWidgetView dataProviderWidgetView = new OptionsPrintPan
// set some default options
optionsPrintPanel.getDefaultPrintRequestDataProvider().getDefaultTemplateBuilde
optionsPrintPanel.getDefaultPrintRequestDataProvider().getDefaultTemplateBuilde
optionsPrintPanel.getDefaultPrintRequestDataProvider().getDefaultTemplateBuilde
optionsPrintPanel.getDefaultPrintRequestDataProvider().setFileName("custom file
optionsPrintPanel.getDefaultPrintRequestDataProvider().setPostPrintAction(Print
// activate some options
...
// create the printWidget
PrintWidget printWidget = new PrintWidget(mapPresenter, applicationId, dataProvi
```

3. Print Widget with custom PrintRequestHandler

When the print action is requested, a call is send to the server. When the pdf file has been created, a response is send back to the client. By default, the pdf is opened in a new browser window. Alternatively a download of the file is started (see full options panel).

The response is handled by a `PrintRequestHandler` instance. This handler can be customized; the widget can contain only one handler at the same time. A custom handler can be registered as follows:

```
PrintWidget widget = new PrintWidget(mapPresenter, applicationId);
PrintRequestHandler customHandler = new CustomPrintRequestHandler();
widget.setPrintRequestHandler(customHandler);
```

The `PrintRequestHandler` contains two methods. `onPrintRequestStarted` will provide with some information just before the request is send to server. `onPrintRequestFinished` will provide with the url and other information when the request has been successfully completed.

4. Print Widget without css (no style)

The print widget can be loaded with empty css classes. This can be achieved by setting the resource bundle factory *before* creating the widget:

```
Print.getInstance().setBundleFactory(new PrintClientBundleFactoryNoStyle());
// create the widget
```

Chapter 3. Gwt2 Print Plugin API

1. Further Configuration

As with most Geomajas plugins, the print plugin has a single class that provides access to all the configurable functional classes within the plugin. In this case:

`org.geomajas.gwt2.plugin.print.client.Print`

This class provides access to the client bundle factory and view factory that are used in the creation of the Print Widget. Both factories can be overwritten by customized classes.

The `Print` class also provides access to a `PrintService`. This service allows to create a print request without the need for a widget of view.

`Print` contains a `PrintUtil` reference.

2. Calling the PrintService without the PrintWidget.

It is possible to use the print plugin without the widget. The information that is necessary to build the pdf can also be provided by encoding. A simple example can be found below:

```
// create a custom TemplateBuilderDataProvider. This is an object that provides
TemplateBuilderDataProvider customDataProvider = new CustomTemplateBuilderDataP
// create a PrintTemplateInfo object. TemplateBuilder can be customized, here w
PrintTemplateInfo printTemplateInfo = Print.getInstance().getPrintUtil().create
// create a print request
PrintRequestInfo printRequestInfo = new PrintRequestInfo();
printRequestInfo.setFileName("customFileName");
printRequestInfo.setPostPrintAction(PrintConfiguration.PostPrintAction.SAVE);
printRequestInfo.setPrintTemplateInfo(printTemplateInfo);
printRequestInfo.setSync(true); // if true, a single HTTP POST request is issue
// call the print method. The PrintRequestHandler can also be customized
Print.getInstance().getPrintService().print(printRequestInfo, new DefaultPrintR
```