MAPFISH PRINT 3 WORKSHOP

GETTING STARTED

camptocamp

INNOVATIVE SOLUTIONS
BY OPEN SOURCE EXPERTS

About Camptocamp

Company specialized in Open Source, pioneering in software development:

- Geographical Information System (**GIS**)
- Business Management (**ERP**)
- Server Infrastructure (**IT Automation and Orchestration**)

Presence in three countries :

- Switzerland (Camptocamp SA)
- France (Camptocamp France SAS)
- Germany (Linuxland GmbH)

Geospatial department:

- Contributor to famous projects as OpenLayers, GeoNetwork, MapServer, QGIS, etc.
- Editor of two main products geOrchestra and GeoMapfish
- Development of tailor-made solutions based on QGIS, GeoMapfish, geOrchestra, etc.

INTRODUCTION

Agenda of this chapter

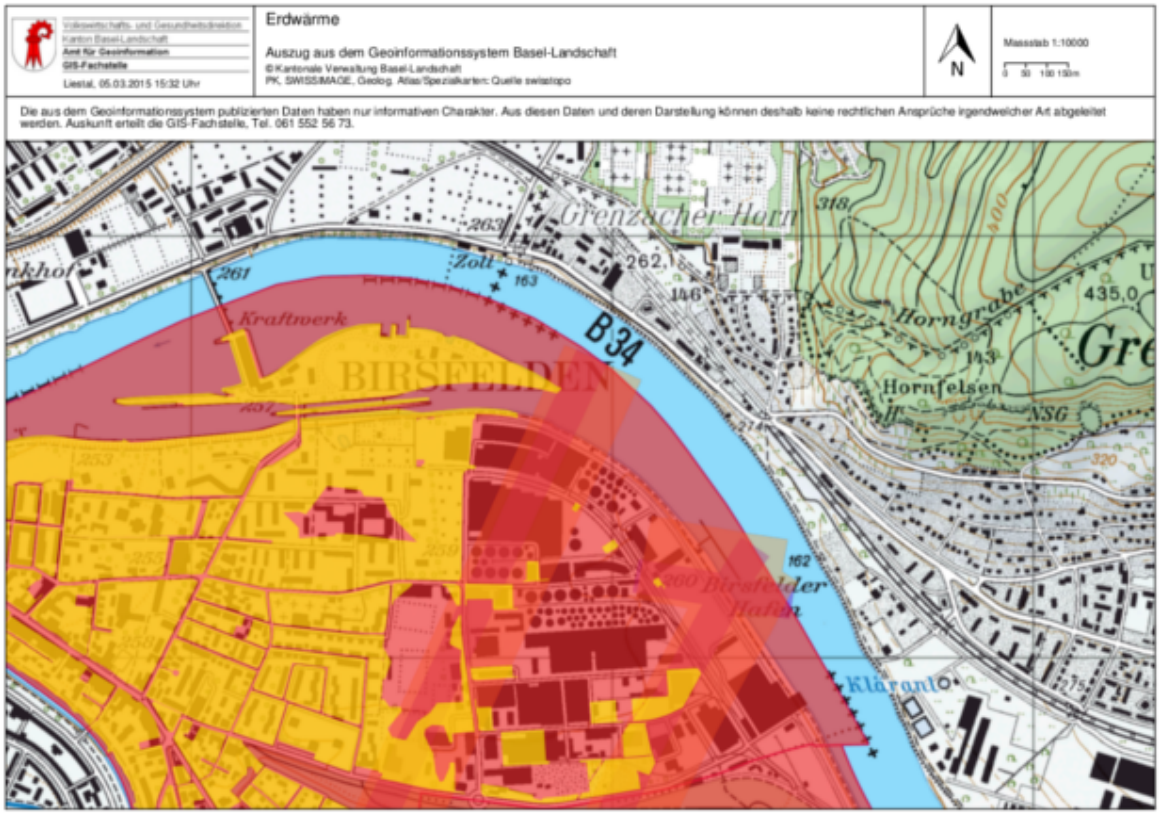
Goal: Get an idea on what MapFish Print is and how a report is generated.

- What is MapFish Print?
- Architecture
- Print process
- Basic configuration
- Resources

What is MapFish Print?

- Goal of the project: Create reports that contain maps and map related components
- A Java library and web-application

Examples



Kanton Basel-Landschaft
Amt für Geoinformation
GIS-Fachstelle
Liestal, 07.03.2015 12:20 Uhr

Auszug aus de
© Kantonale Verw
PK, SWISSIMAGE

Die aus dem Geoinformationssystem publizierten Daten haben
keine rechtlichen Ansprüche irgendwelcher Art abgeleitet werde

Zugangsinfo	Einschränkung	Strasse			
Gasthof Schlüssel, WC-Treppe	Während den Öffnungszeiten	Hauptstrasse			
Parterre Gemeindeverwaltung links neben den Liften	nur während den Öffnungszeiten derGemeindeverwaltung	Hauptstrasse			
Meier & Co. AG; AED hängt an Aussenwand links neben dem Eingang	keine	Baselstrasse	5	4144	Arlesheim
	nachts geschlossen	Domplatz	12	4144	Arlesheim
	Bei geöffneter Anlage	Landskronstrasse	41	4147	Aesch
	0900-1900 Mai -	Dornacherstrasse	85	4147	Aesch

Features

- Components: map, overview-map, scalebar, north-arrow, legend
- Other elements: tables, diagrams, graphics
- Supported geo-data: GeoJSON, GML, GeoTIFF, WMS, WMTS, XYZ/OSM
- Vector styling: SLD, JSON style
- Layouting via JasperReports
- Multi-page support (e.g. with multiple maps)
- Highly customizable
- Integrates with external datasources (e.g. databases)

Architecture

GeoTools

Mapping

JasperReports

Layout

spring

Plugin Framework

mapfish

MapFish Print

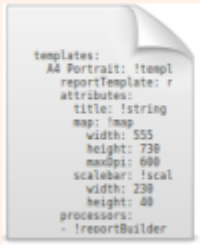
Web API / Security / Widgets

Print process: Configuration

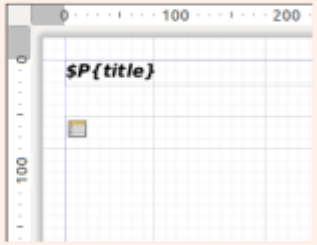
Client

Server

print-app: demo



config.yaml



report.jrxml

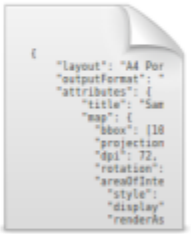
Configuration

Print process: Creating a print job

Client

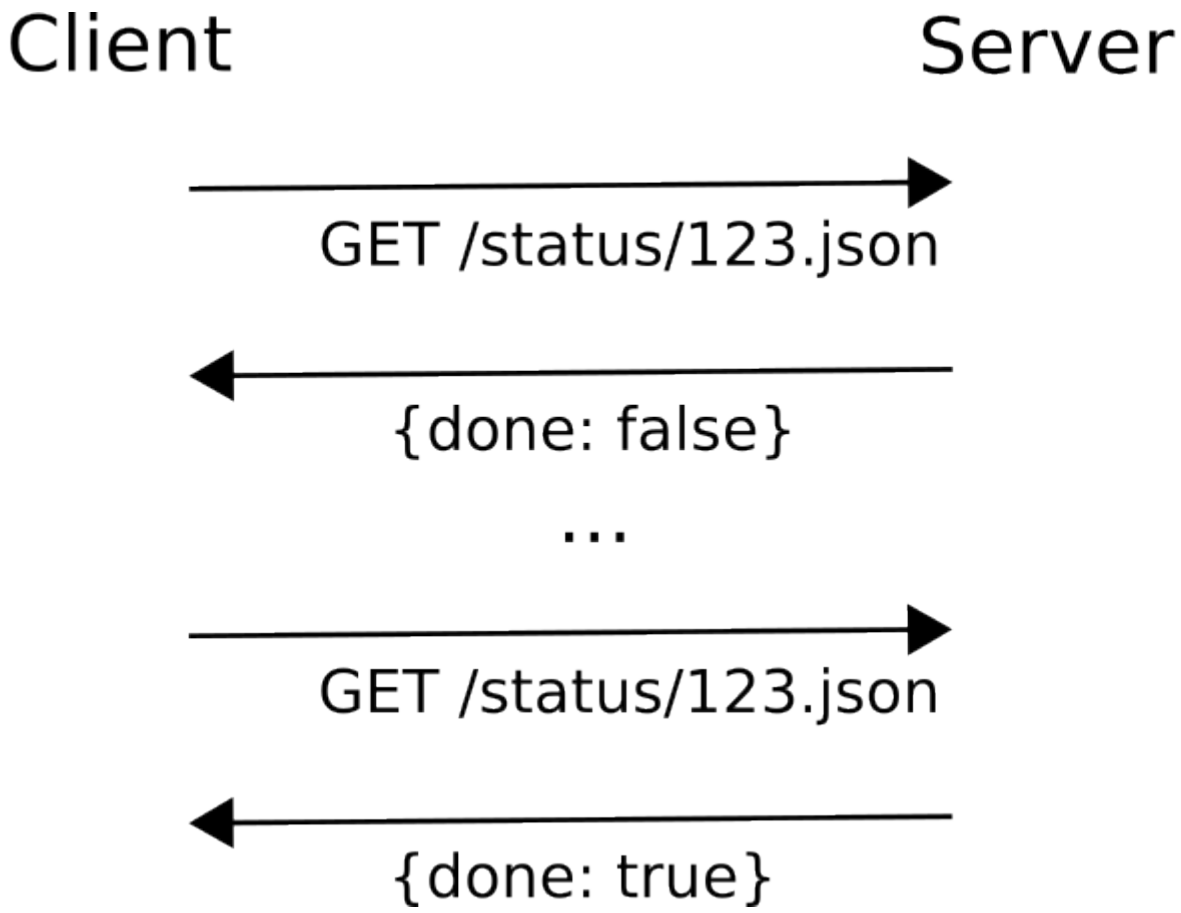
Server

POST /demo/report.pdf

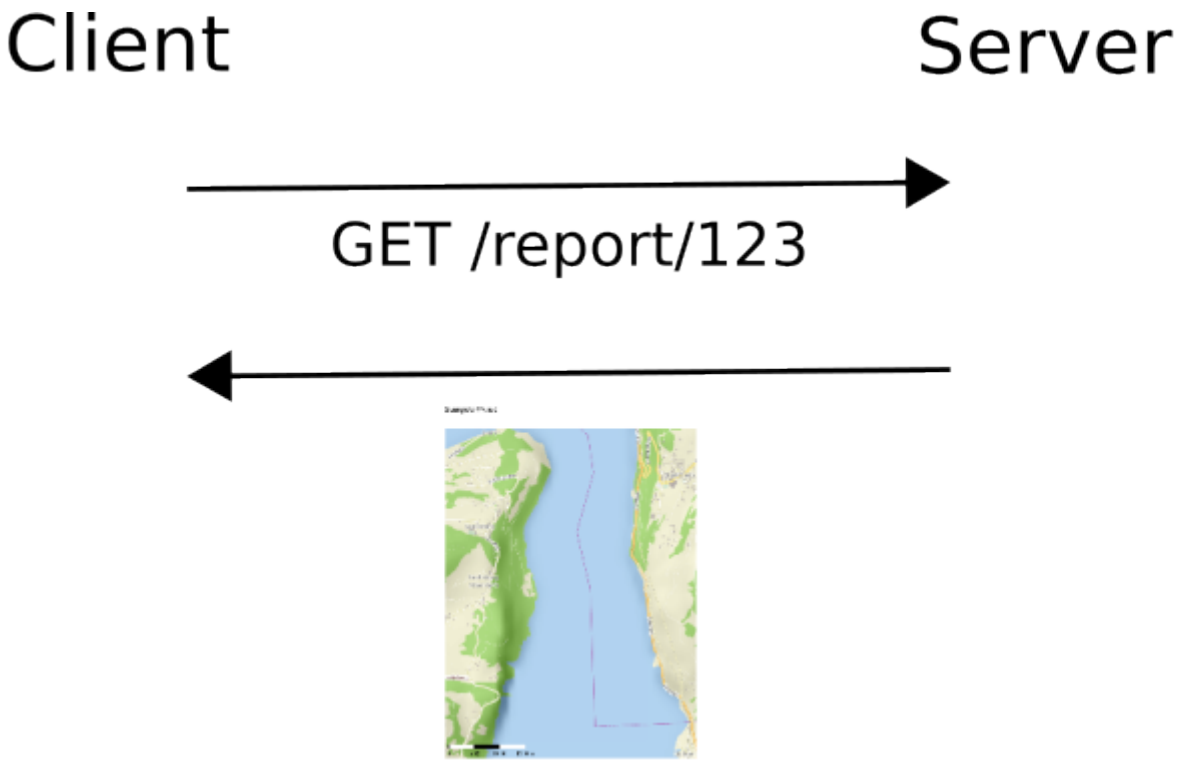


{ref: "123"}

Print process: Polling the status of a print job



Print process: Getting the created report



Basic configuration (config.yaml)

```yaml
templates:
  A4 Portrait: !template
    reportTemplate: report.jrxml

    attributes:
      title: !string {}
      map: !map
        width: 555
        height: 730
        maxDpi: 300
      scalebar: !scalebar
        width: 230
        height: 40

    processors:
    - !reportBuilder
      directory: '.'
    - !createMap
      inputMapper: {map: map}
      outputMapper: {mapSubReport: mapSubReport}
    - !createScalebar {}
```

Basic configuration (template file)

Jaspersoft Studio

Reposi | Project

MyReports

Outline

- report
  - Styles
  - Parameters
  - Fields
  - Sort Fields
  - Variables
  - Scriptlets
  - Title
    - Subreport
    - Subreport
    - $P{title}
  - Page Header

report.jrxml

Main Report | 60%

$P{title}

Title

Design | Source | Preview

Palette

Elements
- Note
- Text Field
- Static Text

Tools
- Page Number
- Total Pages
- Current Date

Propertie | Problems

Sub...ort | Search Property

Appearance | Subreport

Location
- x 0 px | y 51 px
- Position Type: Fix Relative To Top

Size
- w 555 px | h 730 px
- Stretch Type: No Stretch

Color

Writable | Insert | 1 : 1

Basic configuration (JSON request)

```
{
  "layout": "A4 Portrait",
  "outputFormat": "pdf",
  "attributes": {
    "title": "Sample Print",
    "map": {
      "projection": "EPSG:900913",
      "dpi": 72,
      "rotation": 0,
      "center": [957352, 5936844],
      "scale": 25000,
      "layers": [
        {
          "type": "osm",
          "baseURL": "http://tile.osm.ch/osm-swiss-style",
          "imageExtension": "png"
        }
      ]
    }
  }
}
```

Ressources

- Documentation
- Code: GitHub repository
- Example configurations
- Mailing list
- Template creation: Getting Started with Jaspersoft Studio

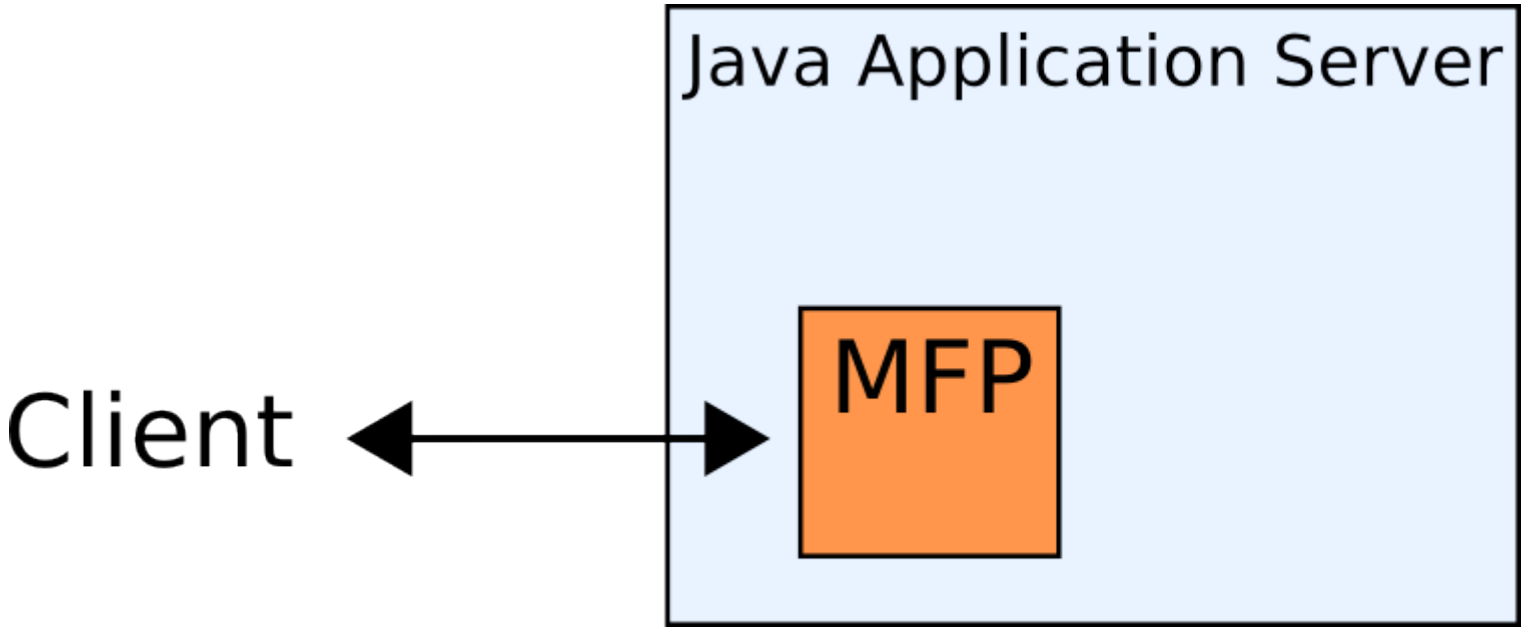
INSTALLATION

Agenda of this chapter

Goal: Setting up MapFish Print and generating a first report.

- MFP as web-application
- MFP as command-line application

Mapfish Print as web-application



Exercise 2.1: Installation

- **Setup Tomcat and deploy MapFish Print**
- **Open http://localhost:8080/print/ in a browser and print a report using the example configuration `simple`.**

Customizing the WAR

- Print configurations: `/print-apps`
- Log configuration: `/WEB-INF/classes/logback.xml`
- Configuration parameters: `/WEB-INF/classes/mapfish-spring.properties`

Exercise 2.2: Customizing the WAR

- Create a custom WAR using the print configuration `00-workshop` with the help of the batch script `update-war.sh`.
- Create a report for the print configuration `00-workshop`.
- To change the report title, add an attribute `"title": "World Map"` in the request.

Using the MFP command-line-interface

For testing

```
$ print-cli/bin/print -config config.yaml -spec requestData.json -output test.pdf
```

Exercise 2.3: Using the CLI application

- **Create a report for the print configuration `00-workshop` with the CLI application.**
- **Change the extent of the printed map by setting a different center and scale.**

YAML CONFIGURATION

Agenda of this chapter

Goal: Understanding the basic structure and elements of the configuration.

- Structure
- Attributes
- Processors
- Overview of available attributes/processors
- Other configuration elements

Basic structure of a configuration file

```yaml
pdfConfig: !pdfConfig
  author: "..."
  subject: "..."

templates:
  A4 Portrait: !template
    reportTemplate: report.jrxml

    attributes:
      title: !string {}
      map: !map
        width: 555
        height: 730
        maxDpi: 300
      scalebar: !scalebar
        width: 230
        height: 40

    processors:
    - !reportBuilder
      directory: '.'
    - !createMap
      inputMapper: {map: map}
      outputMapper: {mapSubReport: mapSubReport}
    - !createScalebar {}
```

Template attributes

```
attributes:
  title: !string
    default: "Countries"
  description: !string {}
  showHeader: !boolean {}
  map: !map
    width: 555
    height: 730
    maxDpi: 300
  scalebar: !scalebar
    width: 230
    height: 40
```

Template processors

```yaml
processors:
- !reportBuilder
  directory: '.'
- !createMap
  inputMapper: {map: map}
  outputMapper: {mapSubReport: map}
- !createScalebar {}
```

Available processors

- createMap
- createScalebar
- createNorthArrow
- createOverviewMap
- prepareLegend
- prepareTable
- createDataSource
- ...

Documentation: Processors

Other configuration elements

- pdfConfig: PDF Metadata

Documentation: pdfConfig

```
pdfConfig: !pdfConfig
  author: "..."
  subject: "..."
```

- proxy: Proxy for requests

Documentation: proxy

```
proxies:
  - !proxy
    scheme: http
    host: proxy.host.com
    port: 8888
    username: username
    password: xyzpassword
    matchers:
      - !localMatch
        reject: true
      - !dnsMatch
        host: www.camptocamp.com
        reject: true
      - !acceptAll {}
```

Proxy all requests except localhost and www.camptocamp.com

Configure HTTP requests

- Restrict access only to certain addresses
- Add or forward headers (e.g. authentication headers)
- Convert URLs (e.g. from `http://domain.com/tiles` to `http://localhost:1234`)

Example

```
- !configureHttpRequests
  httpProcessors:
    # change myhost.com urls to localhost
    - !mapUri
      mapping:
        (http)://myhost.com(.*) : "$1://localhost$2"
    # forward headers
    - !forwardHeaders
      headers: [Cookie, Referrer]
    # only allow localhost requests, any other requests
    # will be rejected
    - !restrictUris
      matchers: [!localMatch {}]
```

Documentation: configureHttpRequests

Exercise 3.1: YAML Configuration

- Make a copy of the print configuration `00-workshop` and for example name the folder `01-exercise-config`.
- Add a new attribute `description` (string) in `config.yaml` without a default value. Also set the attribute in the print request (`requestData.json`).
- Configure a scalebar processor, including a scalebar attribute.
- For now, do not show the description attribute and the scalebar in the JasperReports template. But test if the configuration is correct by generating a report (for example with the CLI application).

REPORT TEMPLATES

Agenda of this chapter

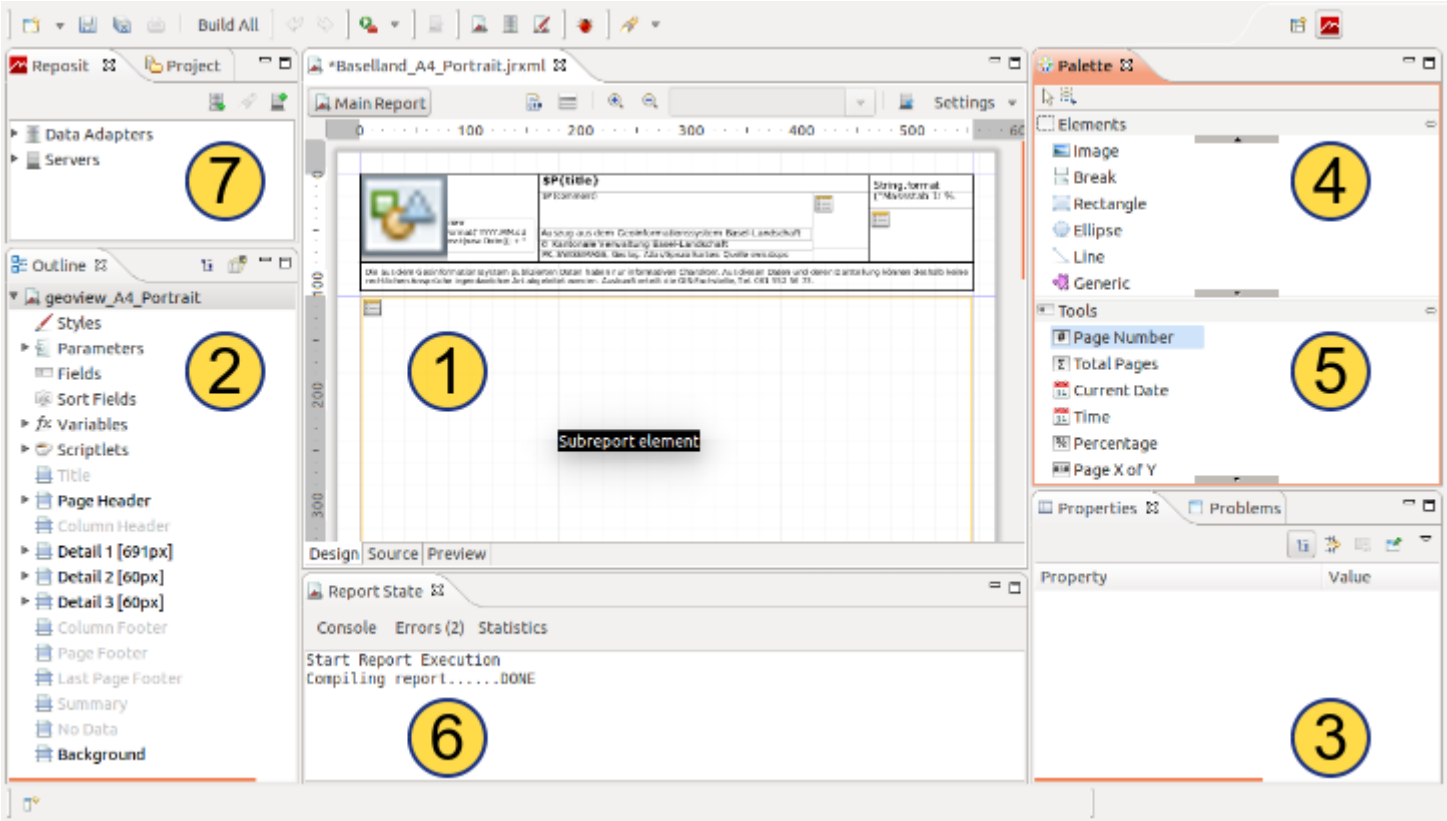
Goal: Learn how the YAML configuration and the templates are connected and how to use JasperSoft Studio to edit the templates.

- Structure, concepts
- Static text
- Images
- Text fields with expressions (date, page)
- Text fields with attribute values
- How is the map included?

Report Structure: JRXML files

- Basically an XML
- Called "template" of the print service
- Skeleton which will be filled by data from a request
- Can be edited in a simple text editor or easier with JasperSoft Studio (WYSIWYG editor)

JasperSoft Studio



- Editor / Source / Preview (1)
- Template outline (2)
- Properties of selected items (3)
- Elements available for insertion in the report (4)
- Shortcuts to pre-configured fields (5)
- Status (6)
- (7)

Static Text Elements

Used to show static text defined at design time.

```
<staticText>
  <reportElement x="12" y="443" width="68" height="20" uuid="..."/>
  <textElement>
    <font fontName="Arial" size="12"/>
  </textElement>
  <text><![CDATA[Some static text]]></text>
</staticText>
```

Can be used for:

- labels
- descriptions
- copyright information

Text Fields with Expressions

Text fields leave you with more options than the static text element especially in terms of:

- size of the field
- what is contained in the field

These elements can be used for:

- dates
- page numbers

```
"Lausanne, " + new
SimpleDateFormat("dd.MM.YYYY
HH:mm").format(new Date()) + " Uhr"
```

→

```
Lausanne, 25.6.2016 12:43 Uhr
```

Text Fields with Expressions

```
<textField>
  <reportElement x="592" y="440" width="208" height="30" uuid="..."/>
  <textElement>
    <font fontName="Arial" size="12"/>
  </textElement>
  <textFieldExpression>
    <![CDATA[
      "Created: " +
      new SimpleDateFormat("dd.MM.YYYY HH:mm").format(
        new Date())
    ]]>
  </textFieldExpression>
</textField>
```

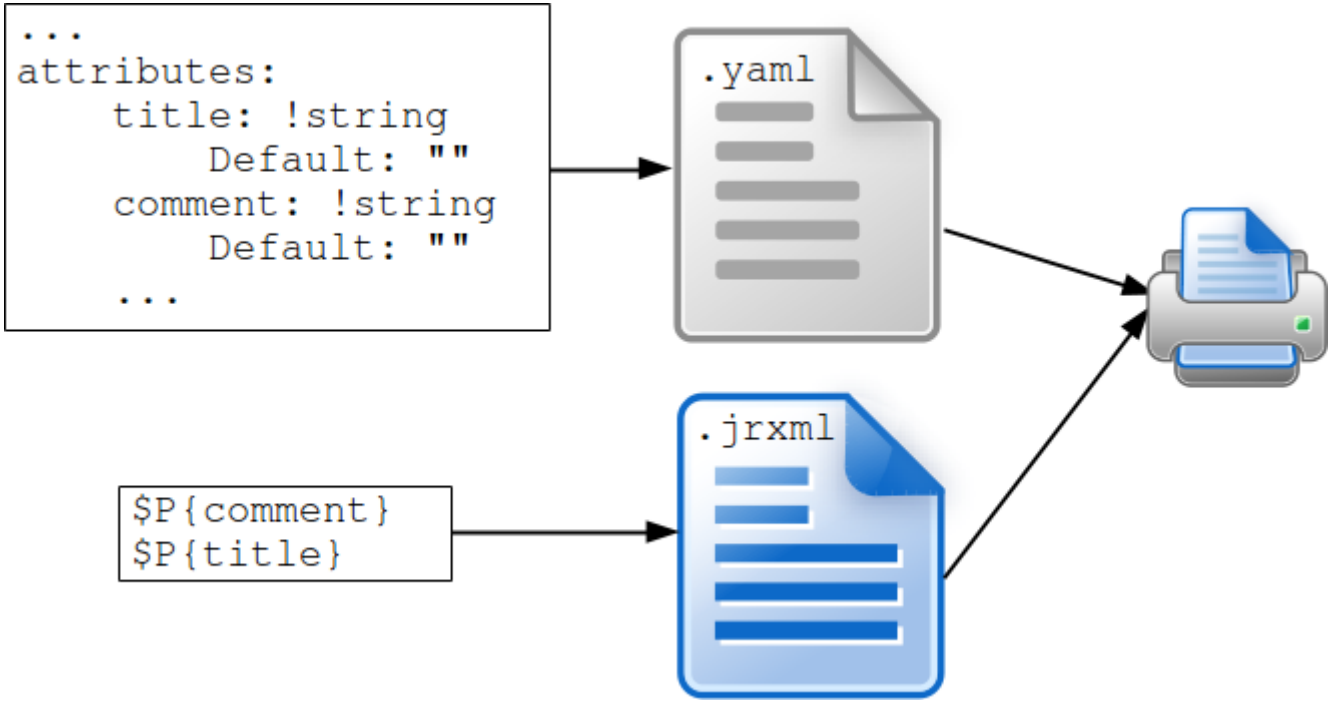
Text Fields with Attribute Values

Attribute values from the `config.yaml` can be added as attributes in the report.

This can be used to:

- Add comments
- Add custom titles
- Show/hide elements (e.g. `showFooter`).

The attribute used in the text field needs to be defined in the `config.yaml` and included as a parameter in the `.jrxml` file.



Text Fields with Attribute Values

```
<parameter name="title" class="java.lang.String"/>
...
<textField>
  <reportElement x="0" y="1" width="800" height="50" uuid="..."/>
  <textElement textAlignment="Center">
    <font size="36"/>
  </textElement>
  <textFieldExpression><![CDATA[$P{title}]]></textFieldExpression>
</textField>
```

Images

Most commonly images are used to insert raster images (such as GIF, PNG and JPEG).

This can be used to add:

- logos
- all kind of graphics

The images must be inside the folder of the print configuration.

```
<image hAlign="Left" vAlign="Middle">
  <reportElement x="90" y="424" width="183" height="50" uuid="..."/>
  <imageExpression><![CDATA["logo.png"]]></imageExpression>
</image>
```

Including a Map

- A map is included as a sub-report element (report in a report).
- The sub-report is created by the `!createMap` processor.
- The map properties are defined in the `config.yaml`.
- The layers shown on the map are defined in the print request.

Including a Map

The parameter `map` contains the path to the sub-report.

```
<parameter name="map" class="java.lang.String"/>
...
<subreport>
  <reportElement x="0" y="94" width="800" height="330" uuid="...">
    <property name="local_mesure_unitwidth" value="pixel"/>
    <property name="com.jaspersoft.studio.unit.width" value="px"/>
    <property name="local_mesure_unitheight" value="pixel"/>
    <property name="com.jaspersoft.studio.unit.height" value="px"/>
  </reportElement>
  <subreportExpression><![CDATA[$P{map}]]></subreportExpression>
</subreport>
```

The size of the sub-report should match the size of the map configured in `config.yaml`.

Resources

Documentation: Getting Started with Jaspersoft Studio

Jaspersoft Community

Exercise 4.1: Installing JasperSoft Studio

- **Download the latest version of JasperSoft Studio from here:**
http://community.jaspersoft.com/project/jaspersoft-studio/releases
- **Extract the file.**
- **To start JasperSoft Studio open a shell and type**

```
cd folder/containing/TIBCOJaspersoftStudio-X.X.X.final
./runubuntu.sh
```

Exercise 4.2: Templates

This exercise will be based on the exercise from chapter 3 (configuration). Either continue with your own configuration or start with the solution of that exercise located at `configurations/01-exercise-config-solution`. Copy the configuration to a new folder `02-exercise-template`.

- **Add a static text field below the map with the text "Copyright: ...".**
- **Add a text field showing the date and time.**
- **Add a logo to the report (copy the file `02-exercise-template-solution/logo.png` into your configuration folder).**
- **Show the value of the attribute `description` (which was defined in the previous exercise in the `config.yaml`).**
- **Show the sub-report of the scalebar that was configured in the previous exercise. Use `scalebarSubReport` as parameter name and the size 230x40px.**

REFERENCING GEO-DATA IN A PRINT REQUEST

Agenda of this chapter

Goal: Learn how geo-data can be referenced in a request.

- Supported geo-data
- Styling vector data

Geo-data in a print request

```
{
  "layout": "A4 Portrait",
  "outputFormat": "pdf",
  "attributes": {
    "title": "Sample Print",
    "map": {
      "projection": "EPSG:3857",
      "dpi": 72,
      "rotation": 0,
      "center": [957352, 5936844],
      "scale": 25000,
      "layers": [
        {
          "type": "osm",
          "baseURL": "http://tile.osm.ch/osm-swiss-style",
          "imageExtension": "png"
        }
      ]
    }
  }
}
```

Supported layer types

- GeoJSON
- GML/WFS
- GeoTIFF
- WMS and tiled WMS
- WMTS
- XYZ/OSM

Documentation: Layers

Example WMS

```
"map": {
  "projection": "EPSG:21781",
  "dpi": 180,
  "rotation": 0,
  "center": [615928, 174957],
  "scale": 1000000,
  "layers": [
    {
      "type": "WMS",
      "layers": ["ch.swisstopo.pixelkarte-farbe-pk1000.noscale"],
      "baseURL": "http://wms.geo.admin.ch/",
      "imageFormat": "image/jpeg",
      "version": "1.1.1",
      "customParams": {
      }
    }
  ]
}
```

Example GeoJSON

```
"map": {
  "longitudeFirst": true,
  "center": [5, 45],
  "scale": 100000000,
  "projection": "EPSG:4326",
  "dpi": 72,
  "rotation": 0,
  "layers": [
    {
      "type": "geojson",
      "geoJson": "file://countries.geojson",
      "style": {
        "version": "2",
        "*": {
          "symbolizers": [
            {
              "type" : "polygon",
              "fillColor": "#5E7F99",
              "fillOpacity": 1,
              "strokeColor": "#CC1D18",
              "strokeOpacity": 1,
              "strokeWidth": 1
            }
          ]
        }
      }
    }
  ]
}
```

Vector styling

- With SLD styles
- Mapfish JSON Style Version 1 (similar to OpenLayers 2 styles)
- Mapfish JSON Style Version 2 (similar to SLD)

Documentation: Styles

SLD styles

style.sld

```
<?xml version="1.0" encoding="UTF-8"?>
<StyledLayerDescriptor ...>
  <NamedLayer>
    <Name>countries_style</Name>
    <UserStyle>
      <FeatureTypeStyle>
        <Rule>
          <PolygonSymbolizer>
            <Stroke>
              <CssParameter name="stroke">#CC1D18</CssParameter>
              <CssParameter name="stroke-width">1</CssParameter>
            </Stroke>
            <Fill>
              <CssParameter name="fill">#5E7F99</CssParameter>
            </Fill>
          </PolygonSymbolizer>
        </Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

In request:

```
  "style": "file://style.sld"
```

Documentation: SLD Reference

SLD Cookbook

Version 2 JSON Styles

```
"style": {
  "version": "2",
  "*": {
    "symbolizers": [
      {
        "type" : "polygon",
        "fillColor": "#5E7F99",
        "fillOpacity": 1,
        "strokeColor": "#CC1D18",
        "strokeOpacity": 1,
        "strokeWidth": 1
      },
      {
        "type": "text",
        "label": "[name]"
      }
    ]
  }
}
```

Exercice 5.1: Geo-data

- **Using one of the previous configurations, print a map in the projection 'epsg:3857' at a location and scale of your choice using an `osm` layer with the URL `http://tile.thunderforest.com/cycle/`. Note that the center coordinate must be given in the projection of the map (you can use http://epsg.io/3857 to get a coordinate in the right projection).**

- **Add a red point as `geojson` layer.**

Static overlay and background layers

Static layers defined in `config.yaml`

```yaml
attributes:
  map: !map
    ...
  overlayLayers: !staticLayers
    default:
      layers:
        - type: "grid"
          numberOfLines: [10, 10]
          labelColor: rgba(0,0,0,0)
          haloColor: rgba(0,0,0,0)
processors:
- !addOverlayLayers
  inputMapper:
      overlayLayers: staticLayers
      map: map
- !createMap {}
```

Example

Questions & Next Steps

Still Learning:

- Tables
- Legends
- Data-sources (e.g. arbitrary number of maps in a report)
- Integrate with external data-sources (e.g. databases)
- Integration in UI
- ...