Joget

10:46 AM
Faster, Simpler Digital Transformation
81
Monthly Course Enrollments
78
Monthly Course Completions
3,291,922
8
TOTAL DEMOS
Overview
Sign-ups
Enquiries
Demos
March 2022

Joget DX

Building Plugins

http://facebook.com/jogetworkflow
http://twitter.com/jogetworkflow

Last Revised on Jan 2023

Prerequisites

- Basic web application development knowledge
- Java web application programming knowledge
- Understanding on Joget plugin architecture and plugin types

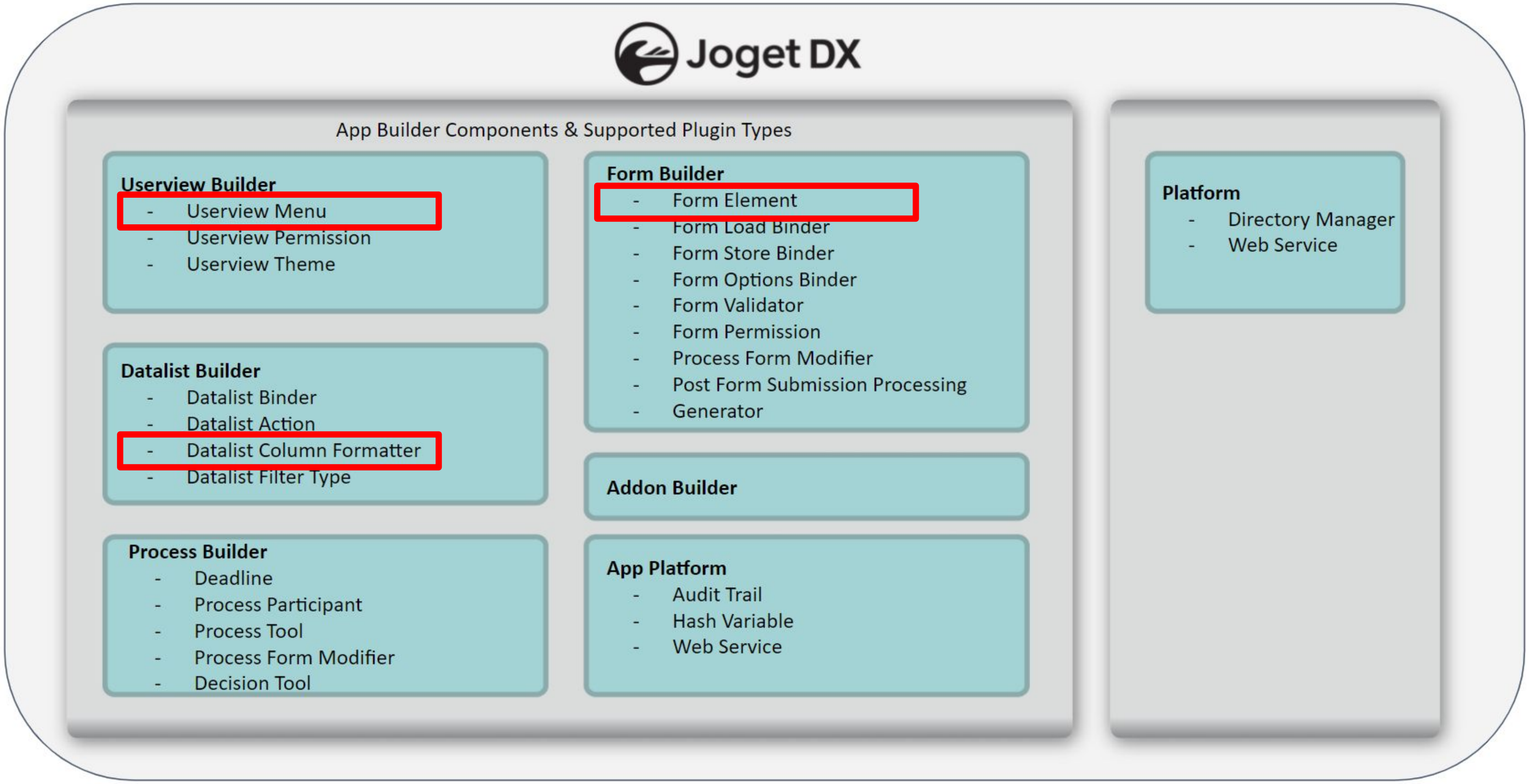
Content

Chapter 1 Introduction

Introduction

- In this module, we will be learning on how to create:
  - Process Tool / Post Form Submission Processing plugin
  - Userview plugin
  - Form Field Element plugin

Plugin Types



Joget DX

App Builder Components & Supported Plugin Types

Userview Builder
- Userview Menu
- Userview Permission
- Userview Theme

Datalist Builder
- Datalist Binder
- Datalist Action
- Datalist Column Formatter
- Datalist Filter Type

Process Builder
- Deadline
- Process Participant
- Process Tool
- Process Form Modifier
- Decision Tool

Form Builder
- Form Element
- Form Load Binder
- Form Store Binder
- Form Options Binder
- Form Validator
- Form Permission
- Process Form Modifier
- Post Form Submission Processing
- Generator

Addon Builder

App Platform
- Audit Trail
- Hash Variable
- Web Service

Platform
- Directory Manager
- Web Service

Each Plugin Is Different

- Before you embark on your journey to build a new plugin, be sure to:
  - Each plugin may be implemented and configured (very) differently
  - Check out existing implementations of such plugin type
  - Extend the necessary classes for each implementation

Reference: https://dev.joget.org/community/display/DX8/Introduction+to+Plugin+Architecture

Plugin Abstract Classes and Interface

- **Deadline Plugins**

  extends org.joget.workflow.model.DefaultDeadlinePlugin

- **Process Participant Plugins**

  extends org.joget.workflow.model.DefaultParticipantPlugin

- **Process Tool / Post Form Submission Processing Plugins**

  extends org.joget.plugin.base.DefaultApplicationPlugin

- **Form Field Element Plugins**

  extends org.joget.apps.form.model.Element
  implements org.joget.apps.form.model.FormBuilderPaletteElement

Plugin Abstract Classes and Interface

- **Form Load Binder Plugins**

  extends org.joget.apps.form.model.FormBinder

  implements org.joget.apps.form.model.FormLoadBinder, org.joget.apps.form.model.FormLoadElementBinder

- **Form Options Binder Plugins**

  extends org.joget.apps.form.model.FormBinder

  implements org.joget.apps.form.model.FormLoadOptionsBinder

- **Form Store Binder Plugins**

  extends org.joget.apps.form.model.FormBinder

  implements org.joget.apps.form.model.FormStoreBinder, org.joget.apps.form.model.FormStoreElementBinder

Plugin Abstract Classes and Interface

- **Form Validator Plugins**

  extends org.joget.apps.form.model.FormValidator
- **List Action Plugins**

  extends org.joget.apps.datalist.model.DataListActionDefault
- **List Binder Plugins**

  extends org.joget.apps.datalist.model.DataListBinderDefault
- **List Column Formatter Plugins**

  extends org.joget.apps.datalist.model.DataListColumnFormatDefault
- **UI Menu Plugins**

  extends org.joget.apps.userview.model.UserviewMenu

Joget

Plugin Abstract Classes and Interface

- **UI Permission Plugins**

  extends org.joget.apps.userview.model.UserviewPermission

- **UI Theme Plugins**

  extends org.joget.apps.userview.model.UserviewTheme

- **Audit Trail Plugins**

  extends org.joget.plugin.base.DefaultAuditTrailPlugin

- **Hash Variable Plugins**

  extends org.joget.apps.app.model.DefaultHashVariablePlugin

- **Directory Manager Plugins**

  extends org.joget.plugin.base.ExtDefaultPlugin

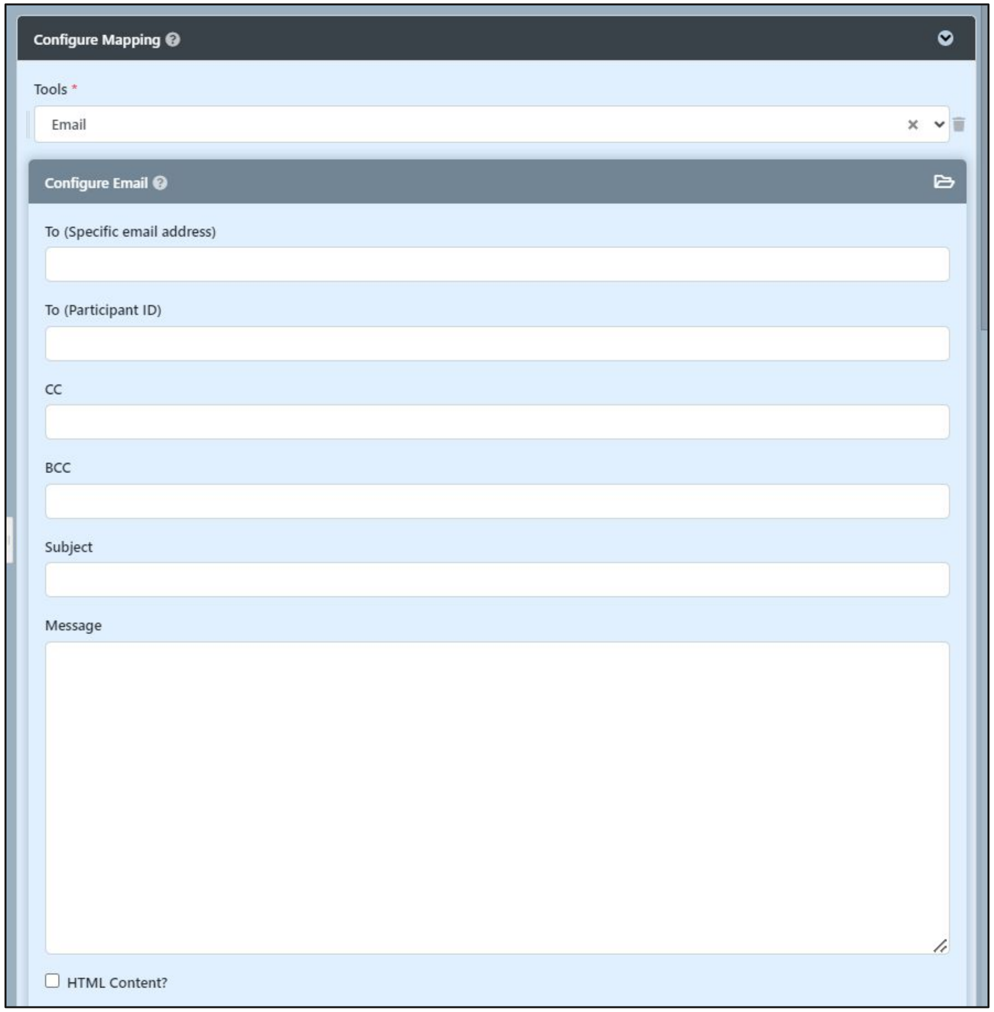
  implements org.joget.directory.model.service.DirectoryManagerPlugin, org.joget.plugin.property.model.PropertyEditable

Introduction to Property Options

- Each plugin uses the Property Options template scheme to provide an interface for **App Designers** to configure your plugin

Reference:
http://dev.joget.org/community/display/DX8/Plugin+Properties+Options

Property Options – Email Tool


Configure Mapping

Tools *

Email

Configure Email

To (Specific email address)

To (Participant ID)

CC

BCC

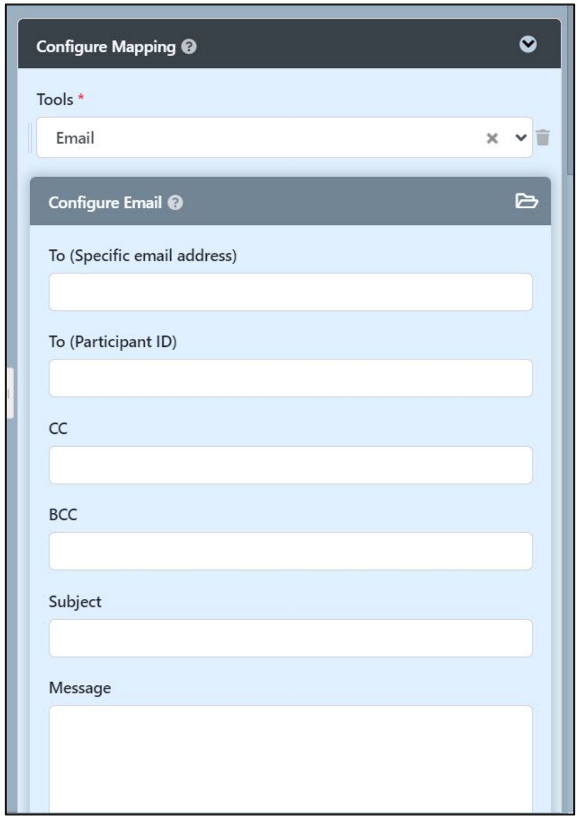
Subject

Message

☐ HTML Content?

Property Options – Email Tool

```
[
  {
    "title": "@@app.emailtool.config@@",
    "properties": [
      {
        "name": "toSpecific",
       "label": "@@app.emailtool.toEmail@@",
        "type": "textfield"
      },
      {
        "name": "toParticipantId",
        "label": "@@app.emailtool.toPid@@",
        "type": "textfield"
      },
      {
        "name": "cc",
        "label": "@@app.emailtool.cc@@",
        "type": "textfield"
      },
```


Configure Mapping

Tools *

Email

Configure Email

To (Specific email address)

To (Participant ID)

CC

BCC

Subject

Message

https://github.com/jogetworkflow/jw-community/blob/8.0-SNAPSHOT/wflow-core/src/main/resources/properties/app/emailTool.json

Property Options

- JSON format

TIP: Just search online for "JSON Beautifier" to properly indent and format your JSON definition.

```
[
    {
        title : 'Page Title',
        properties : [
            {
                name : 'Property Name',
                label : 'Property Label',
                description : 'Property Description', //optional, default is NULL
                type : 'Property Type',
                value : 'Property Value', //optional, default is null
                required : 'Mandatory or Not', //optional, 'true' or 'false', default is 'false'
                //… more attributes …
            }, //… more fields …
        ],
        validators : [  //optional
            //… properties custom validators …
        ],
        buttons : [  //optional
            //… custom properties page buttons …
        ]
    }, //… more properties page …
]
```

Property Options Types

- From V4
  - checkbox - **Check Box**
  - elementSelect - **Element Select Box**
  - grid – **Grid**
  - hidden – **Hidden Field**
  - htmleditor– **HTML Editor**
  - multiselect– **Multi Select Box**
  - password– **Password Field**
  - radio – **Radio Button**
  - readonly – **Readonly Text Field**
  - selectbox – **Select Box**
  - textarea – **Text Area**
  - textfield – **Text Field**

Property Options Types

- From V5
  - codeeditor – **Code Editor**
  - gridcombine – **Combine Grid**
  - gridfixedrow – **Fixed Row Grid**
  - header – **Header**
  - label – **Label**

Property Options Types

- From V6
  - file – **File**
  - image – **Image**
  - color **- Color Picker**
  - sortableselect - **Sortable Select**
  - autocomplete - **Auto Complete**
  - New Types to be used inside **Grid**
    - truefalse - **True/False**
    - autocomplete - **Autocomplete**

Reference to latest documentation:

https://dev.joget.org/community/display/KBv6/Plugin+Properties+Options

Property Options Types

- From DX 7
  - custom - **Custom Scripting**
  - elementmultiselect - **Multiselect in Grid interface**
  - number - **Number**

Reference to latest documentation:

https://dev.joget.org/community/display/DX7/Plugin+Properties+Options

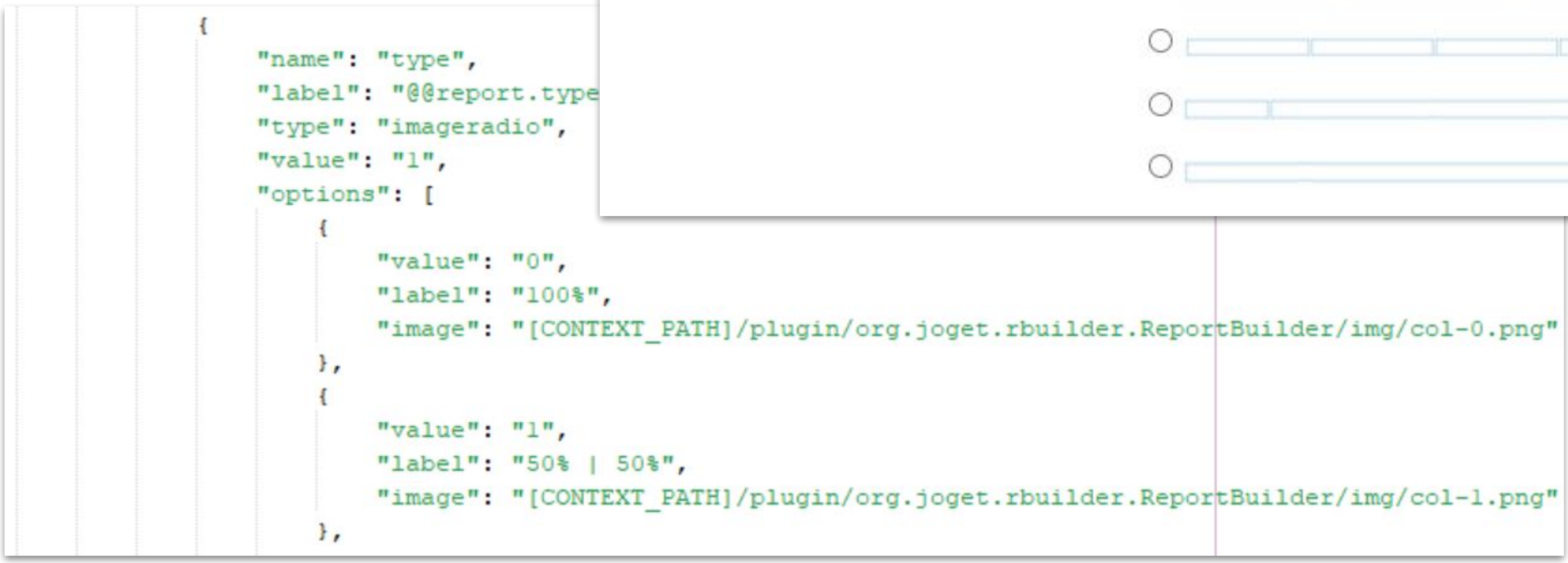
Property Options Types

- New in DX 8
  - imageradio - **Image Radio**
  - repeater - **Repeater**
  - icon-textfield - **Icon Text Field**
  - iconbuttons - **Icon Buttons**
  - colorscheme - **Color Scheme**
  - cssstyle - **CSS Style**

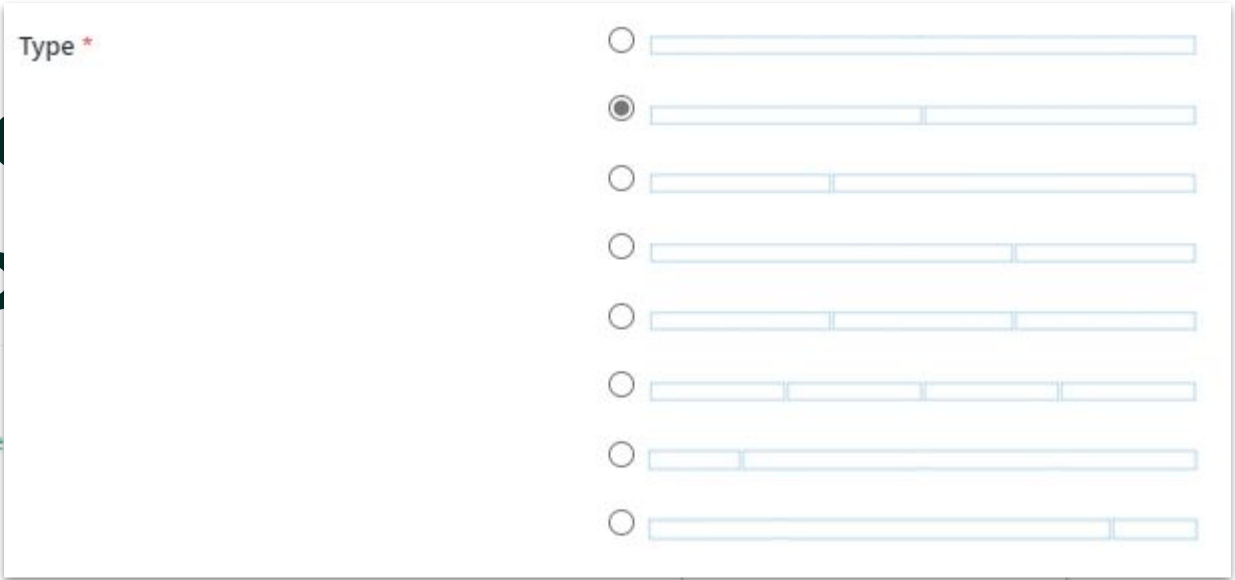
Reference to latest documentation:

https://dev.joget.org/community/display/DX8/Plugin+Properties+Options

DX 8 – Prop
imageradio

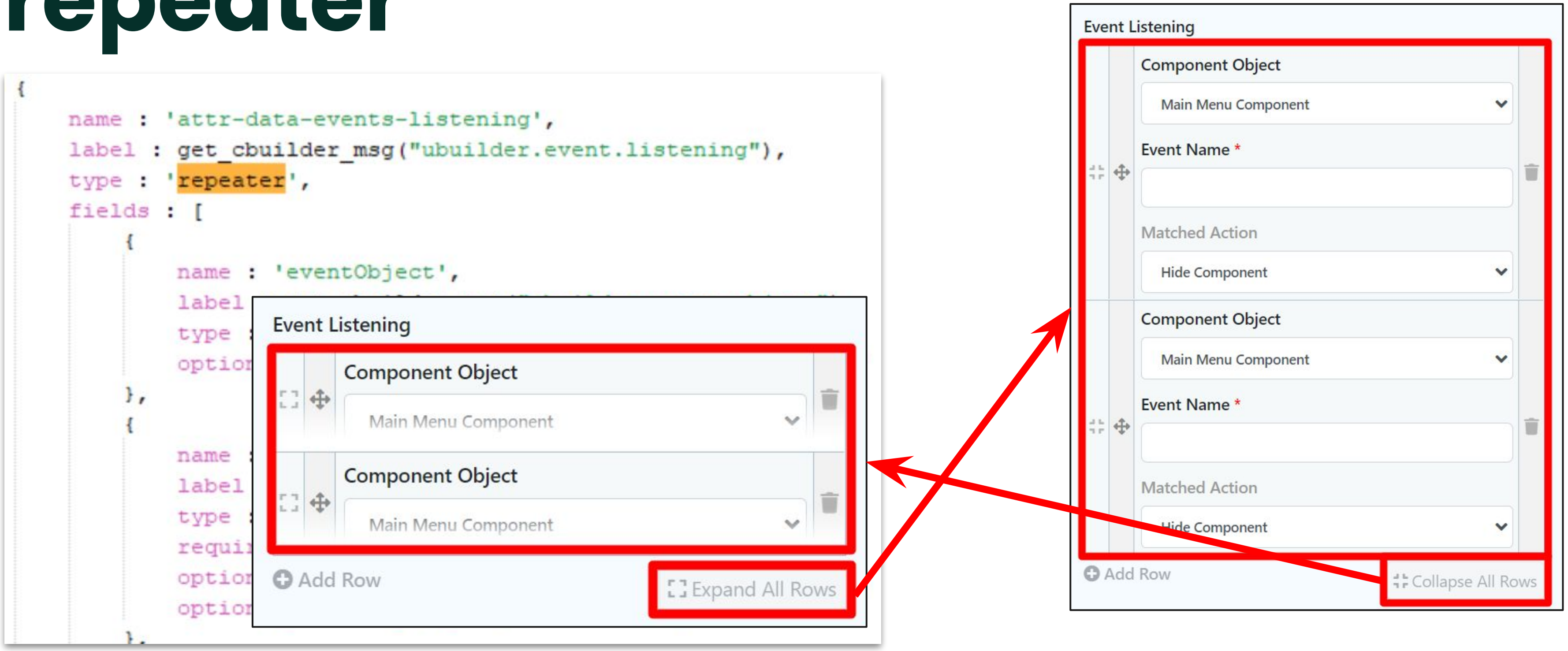


```
{
    "name": "type",
    "label": "@@report.type
    "type": "imageradio",
    "value": "1",
    "options": [
        {
            "value": "0",
            "label": "100%",
            "image": "[CONTEXT_PATH]/plugin/org.joget.rbuilder.ReportBuilder/img/col-0.png"
        },
        {
            "value": "1",
            "label": "50% | 50%",
            "image": "[CONTEXT_PATH]/plugin/org.joget.rbuilder.ReportBuilder/img/col-1.png"
        },
```



https://github.com/jogetworkflow/jw-community/blob/8.0-SNAPSHOT/wflow-consoleweb/src/main/webapp/js/jquery/jquery.propertyeditor.js#L2937

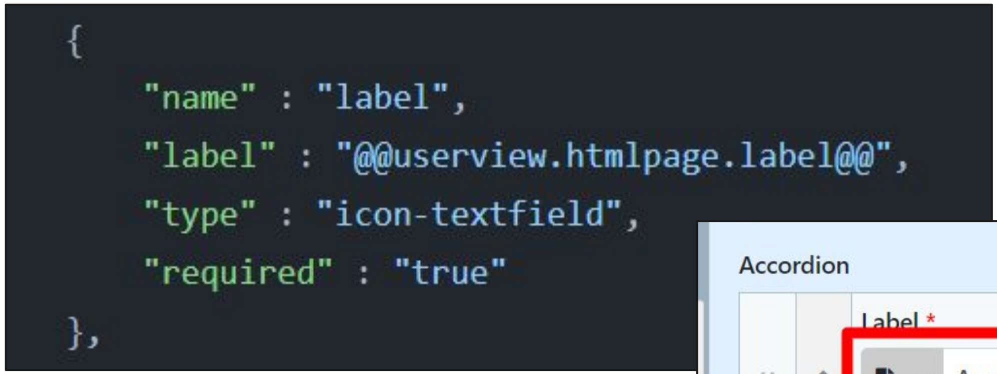
DX 8 – Property Options Types – repeater



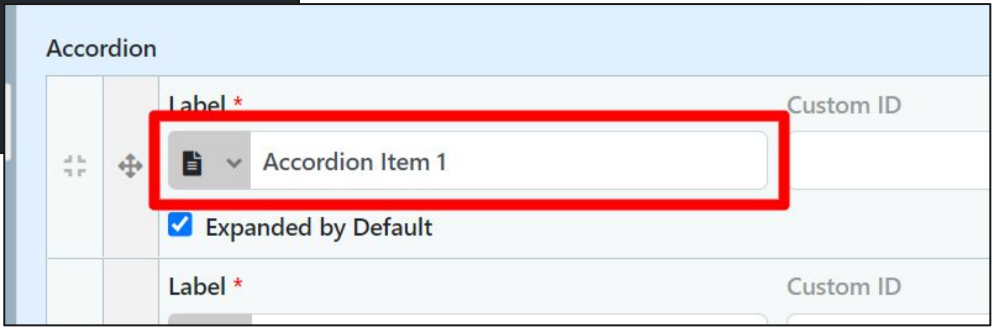
```
{
    name : 'attr-data-events-listening',
    label : get_cbuilder_msg("ubuilder.event.listening"),
    type : 'repeater',
    fields : [
        {
            name : 'eventObject',
            label
            type
            optio
        },
        {
            name
            label
            type
            requi
            optio
            optio
        },
```

https://github.com/jogetworkflow/jw-community/blob/8.0-SNAPSHOT/wflow-consoleweb/src/main/webapp/js/ubuilder.core.js#L1276

DX 8 – Property Options Types – icon-textfield

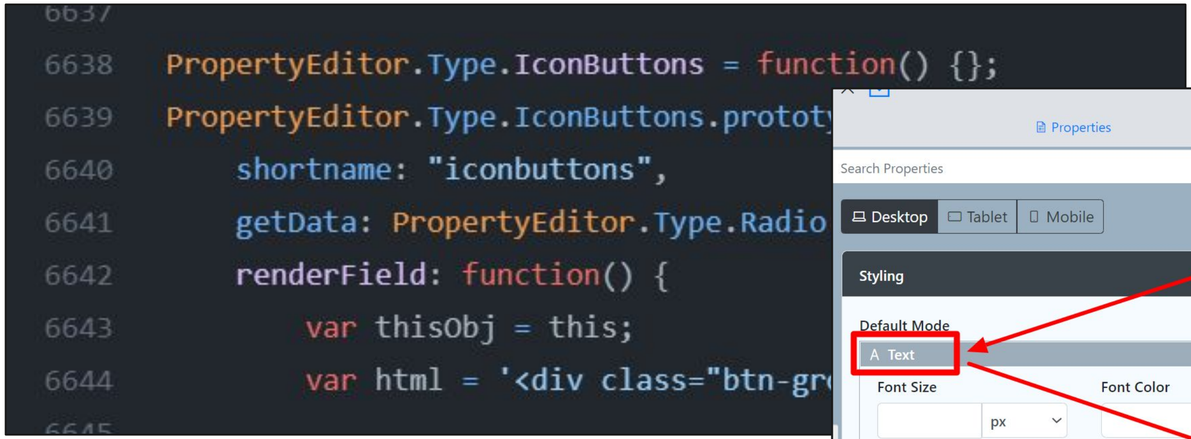


```
{
    "name" : "label",
    "label" : "@@userview.htmlpage.label@@",
    "type" : "icon-textfield",
    "required" : "true"
},
```

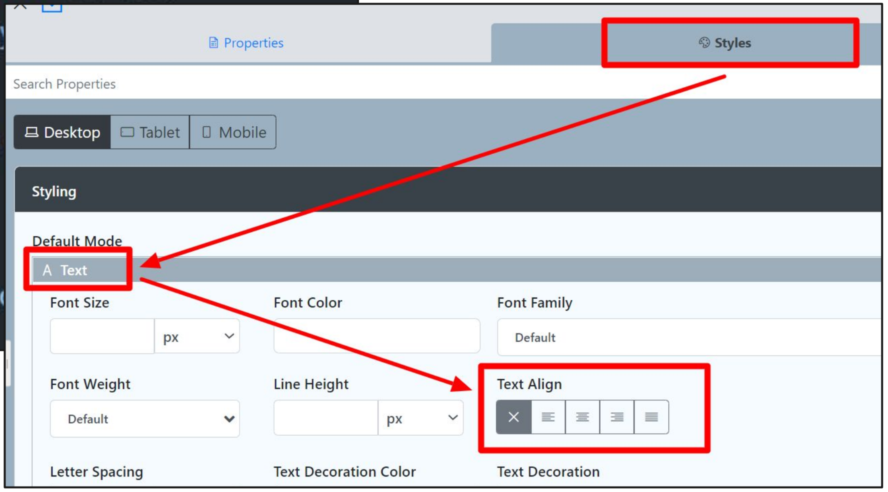


https://github.com/jogetworkflow/jw-community/blob/8.0-SNAPSHOT/wflow-core/src/main/resources/properties/userview/AccordionComponent.json

DX 8 – Property Options Types – iconbuttons

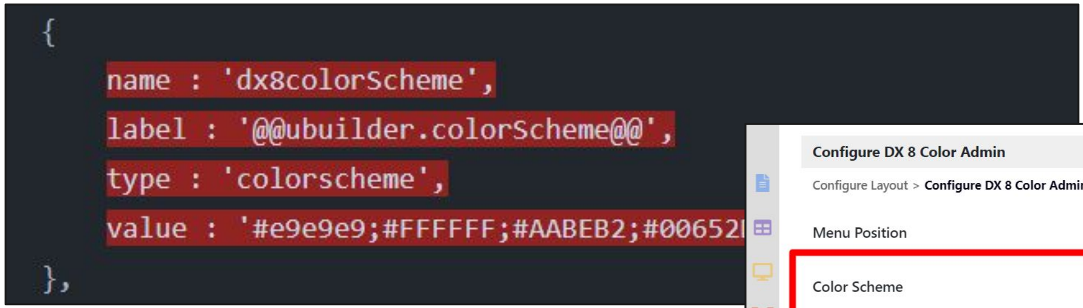


```
PropertyEditor.Type.IconButtons = function() {};
PropertyEditor.Type.IconButtons.prototy
    shortname: "iconbuttons",
    getData: PropertyEditor.Type.Radio
    renderField: function() {
        var thisObj = this;
        var html = '<div class="btn-gr
```

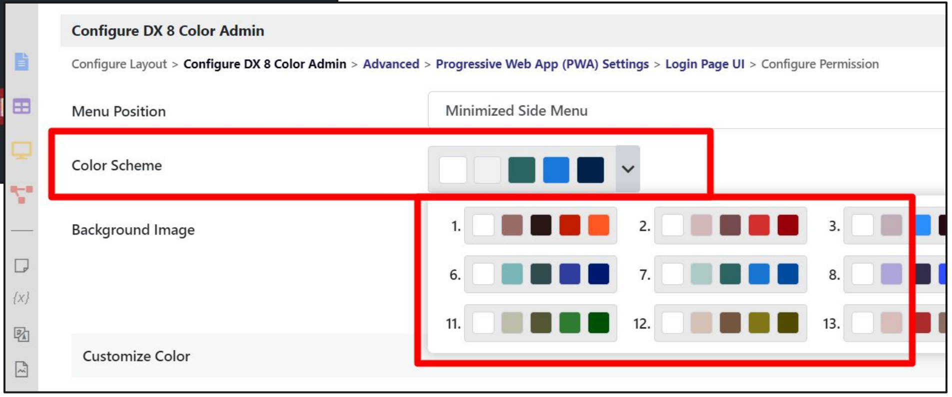


https://github.com/jogetworkflow/jw-community/blob/8.0-SNAPSHOT/wflow-consoleweb/src/main/webapp/js/jquery/jquery.propertyeditor.js#L2937

DX 8 – Property Options Types – colorscheme



```
{
    name : 'dx8colorScheme',
    label : '@@ubuilder.colorScheme@@',
    type : 'colorscheme',
    value : '#e9e9e9;#FFFFFF;#AABEB2;#00652
},
```



Configure DX 8 Color Admin

Configure Layout > **Configure DX 8 Color Admin** > Advanced > Progressive Web App (PWA) Settings > Login Page UI > Configure Permission

Menu Position: Minimized Side Menu

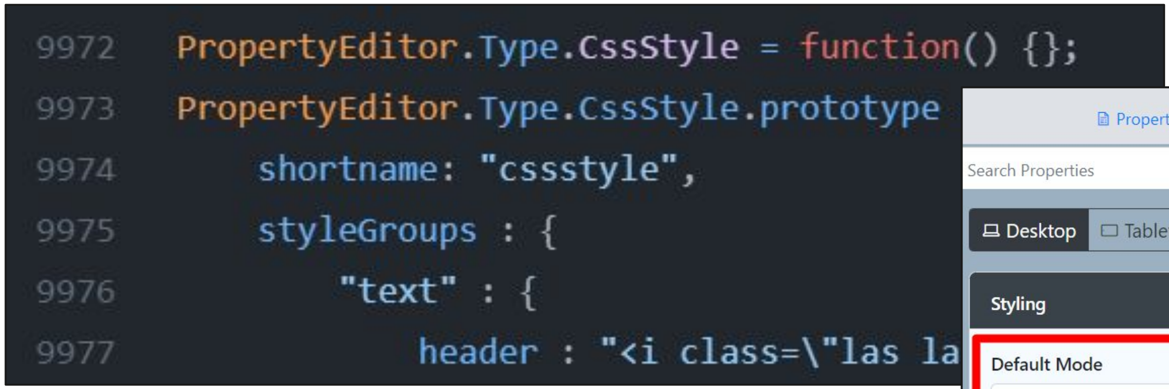
Color Scheme

Background Image

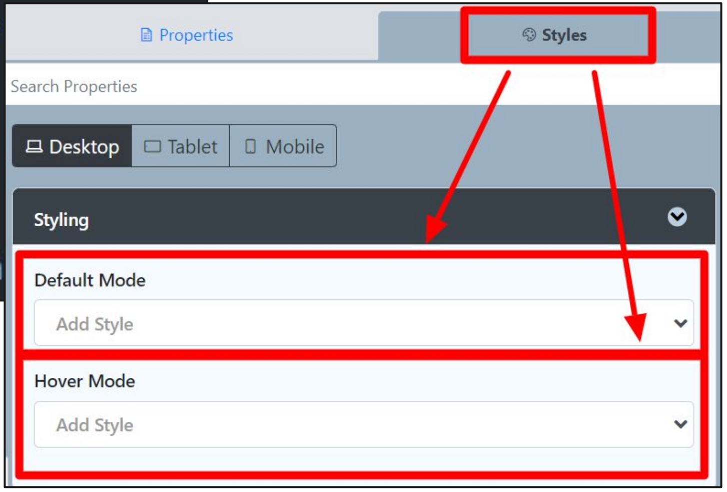
Customize Color

https://github.com/jogetworkflow/jw-community/blob/8.0-SNAPSHOT/wflow-core/src/main/resources/properties/userview/dx8ColorAdminTheme.json

DX 8 – Property Options Types – cssstyle



```
PropertyEditor.Type.CssStyle = function() {};
PropertyEditor.Type.CssStyle.prototype
    shortname: "cssstyle",
    styleGroups : {
        "text" : {
            header : "<i class=\"las la
```



https://github.com/jogetworkflow/jw-community/blob/8.0-SNAPSHOT/wflow-consoleweb/src/main/webapp/js/jquery/jquery.propertyeditor.js#L2937

Common Attributes for All Property Options Type except Hidden Field and Grid

```
{
    name : ‘Property Name’,
    label : ‘Property Label’,
    description : ‘Property Description’, //optional, default is NULL
    type : ‘readonly’,
    value : ‘Property Value’, //optional , default is empty string
    required : ‘true’, //optional, boolean value, default is false
}
```

Extra Attributes for Text Field, Password Field, Text Area and HTML Editor

```
{
    size : ‘50’, //optional , integer value, default is NULL, only for
text field and password field
    maxlength : ‘50’, //optional, integer value, default is NULL, only
for text field and password field
    rows : ‘50’, //optional, integer value, default is NULL, only for
text area and html editor
    cols : ‘50’, //optional, integer value, default is NULL , only for
text area and html editor
    regex_validation : ‘^[a-zA-Z0-9_]+$’, //optional, default is NULL
    validation_message : ‘Error!!’ //optional, default is NULL
}
```

Joget

Extra Attributes for Checkbox, Radio Button, Select Box and Multi Select Box

```
{
    size : ‘10’, //optional, integer value, default is 4, only for multi
select box
    options : [ //is optional to use this attribute or options_ajax
        {value: ‘value1’, label : ‘Value 1’},
        {value: ‘value2’, label : ‘Value 2’},
        {value: ‘value3’, label : ‘Value 3’}
    ],
    options_ajax_on_change : ‘property1’, //optional, value of this
property name will passed over to load options from ajax, only for select
box and multi select box
    options_ajax : ‘URL to load options JSON’ //optional, URL return JSON
Array of a set of Objects that have value & label attribute
}
```

Attributes for Hidden Field

```
{
    name : ‘Property Name’,
    type : ‘hidden’,
    value : ‘Property Value’
}
```

Attributes for Grid

```
{
    name : ‘Property Name’,
    label : ‘Property Label’,
    description : ‘Property Description’, //optional, default is NULL
    type : ‘grid’,
    columns : [ // 2 type of column, with and without options attribute
         {key : ‘col1’, label : ‘Col 1’},
         {key : ‘col2’, label : ‘Col 2’,
             options:[
                 {value :‘option1’, label : ‘Option 1’},
                 {value :‘option2’, label : ‘Option 2’}
             ]
         },
    ]
    value : [ //optional, default is NULL
         {col1 : ‘abc’, col2 : ‘option1’},
         {col1 : ‘def’, col2 : ‘option2’}
    ],
    required : ‘true’, //optional, boolean value, default is false
}
```

Extra Attributes for Element Select Field

```
{
    options_ajax_on_change : ‘property1’, //optional, value of this property name will passover to load options from ajax
    options_ajax : ‘[CONTEXT_PATH]/web/property/json/getElements?classname=org.joget.apps.form.model.FormLoadElementBinder’, //Load plugin list based on class name given
    url : ‘[CONTEXT_PATH]/web/property/json/getPropertyOptions’ //Load plugin properties
}
```

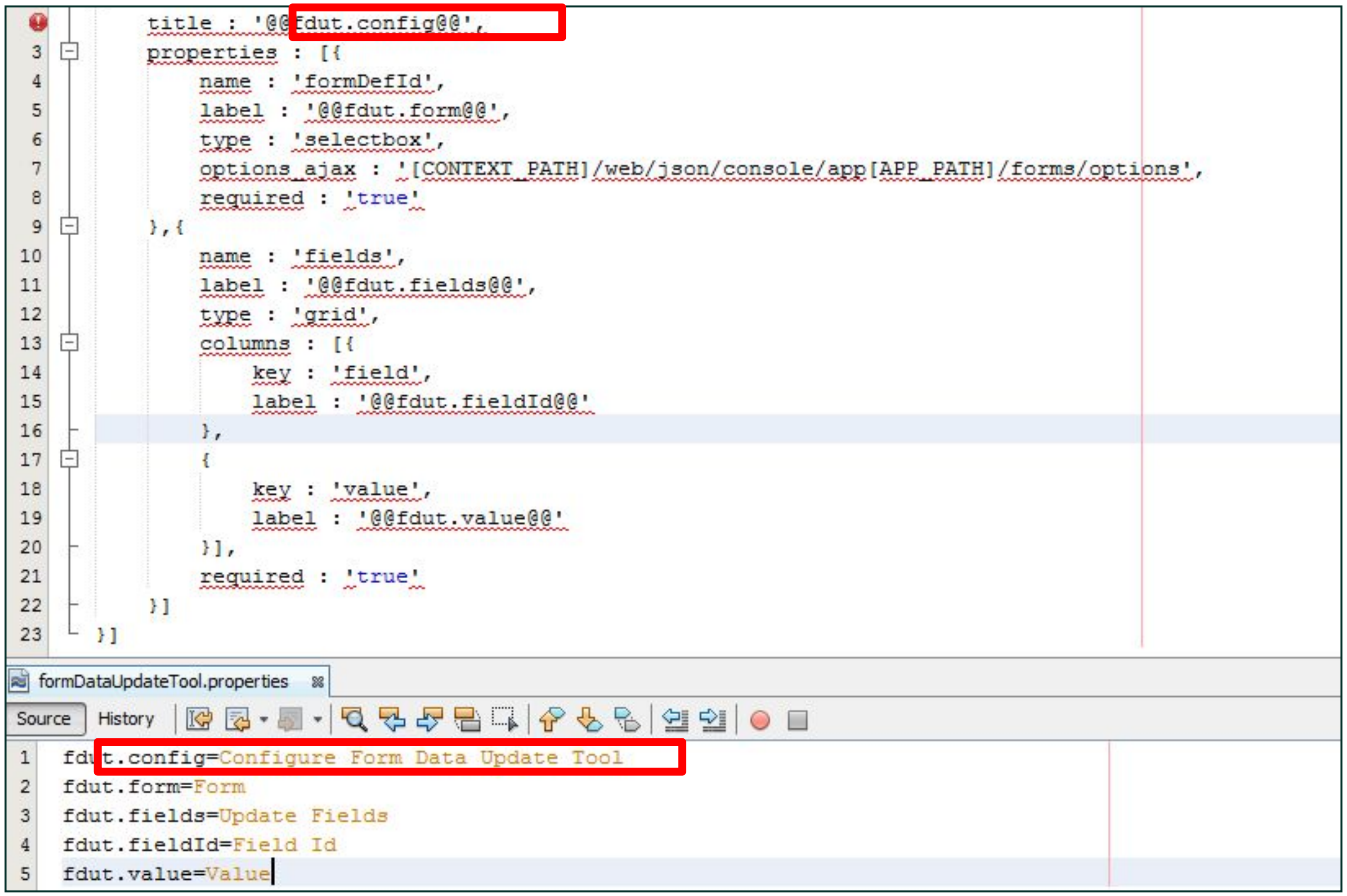
Property Validator Types

- Currently only one validator type supported - AJAX

```
{
    type : ‘AJAX’,
    url : ‘URL to validate properties page value’ ,
    // All properties in the same page will send to this url to validate,
URL return a JSON Object with status (success or fail) & message (JSONArray
of String) attribute
    default_error_message : ‘Error in this page!!’ //optional, default is
null
}
```

Internationalization Support

- You may localise your plugin



```
    title : '@@fdut.config@@',
    properties : [{
        name : 'formDefId',
        label : '@@fdut.form@@',
        type : 'selectbox',
        options_ajax : '[CONTEXT_PATH]/web/json/console/app[APP_PATH]/forms/options',
        required : 'true'
    },{
        name : 'fields',
        label : '@@fdut.fields@@',
        type : 'grid',
        columns : [{
            key : 'field',
            label : '@@fdut.fieldId@@'
        },
        {
            key : 'value',
            label : '@@fdut.value@@'
        }],
        required : 'true'
    }]
}]
```

formDataUpdateTool.properties

Source History

```
fdut.config=Configure Form Data Update Tool
fdut.form=Form
fdut.fields=Update Fields
fdut.fieldId=Field Id
fdut.value=Value
```

Internationalization Support

1. In the **getPropertyOptions()** method of your plugin, make reference to the message key properties file
   Example:

```
return AppUtil.readPluginResource(getClass().getName(),
"/properties/formDataUpdateTool.json", null, true,
"/messages/formDataUpdateTool");
```

2. Create the corresponding file (e.g. **formDataUpdateTool.properties**) in “src/main/resources/messages” folder.

Common Plugin Utility Methods

- Get a bean from application context

```
DirectoryManager dm = (DirectoryManager)
AppUtil.getApplicationContext().getBean(“directoryManager”);
```

Bean Name

- Read a resource file as String from resources folder

```
String resource = AppUtil.readPluginResource(getClass().getName(),
"/properties/plugin.json", new String[]{“value 1”, “value 2”}, true,
"message/pluginResouseBundle");
```

Plugin Class Name

Path to resource file in resources folder

Remove new line char

Path to message bundle in resources folder, without .properties file extension

Argument used for String.format()

Common Plugin Utility Methods

- Processing Hash Variable to a String

```
content = AppUtil.processHashVariable(content, workflowAssignment,
StringUtil.TYPE_REGEX, replaceMap);
```

String escape type,
TYPE_REGEX,
TYPE_JSON or null

A map contains Strings to be replace

Content to be process

Workflow Assignment
Object

- Get a il8n message from message bundle

```
String message = AppPluginUtil.getMessage(“Label”, getClass().getName(),
"message/pluginResouseBundle")
```

Path to message bundle in resources
folder, without .properties file
extension

Message Key

Plugin Class Name

Common Plugin Utility Methods

- Get parameter name of Form Element

```
String param = FormUtil.getElementParameterName(this);
```

- Get value of Form Element

```
String value = FormUtil.getElementPropertyValue(this, formData);
```

Common Plugin Utility Methods

Read more at:
https://dev.joget.org/community/display/DX8/Utility+and+Service+Methods

Plugin Web Support

- Provides an interface that enables you to implement your Web Service in a plugin
- How to invoke/call?
  - URL Pattern
    - **{Context Path}/web/json/plugin/{Plugin Class Name}/service**
    - Example: `http://localhost:8080/jw/web/json/plugin/org.joget.sample.lib.SimpleFormElement/service?say_something=Hello World`
  - URL Pattern
    - **{Context Path}/web/json/app/{App Id}/{App Version}/plugin/{Plugin Class Name}/service**
    - Example: `http://localhost:8080/jw/web/json/app/crm/1/plugin/org.joget.sample.lib.SimpleFormElement/service?say_something=Hello World`

Plugin Web Support

- Implements org.joget.plugin.base.PluginWebSupport
- Implement webService method
- Example:

```
public void webService(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
     //get request parameter
    String param = request.getParameter("param");

    //print json response
    JSONArray jsonArray = new JSONArray();
    Map<String, String> option1 = new HashMap<String, String>();
    option1.put("value", "value1"); option1.put("label", "Value 1");
    jsonArray.put(option1);

    Map<String, String> option2 = new HashMap<String, String>();
    option2.put("value", "value2"); option2.put("label", "Value 2");
    jsonArray.put(option2);

    jsonArray.write(response.getWriter());
}
```

Plugin Web Support

- Add the following dependency into your **pom.xml** file

```
<dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jsp-api</artifactId>
            <version>2.0</version>
</dependency>
```

Read more at https://dev.joget.org/community/display/DX8/Web+Service+Plugin

Chapter 2
Creating a Form Field Element Plugin

What Are We Going To Build?

- We are going to build a Form Field Element that works very similarly like a Text Field but…
  - By incorporating the use of a jQuery plugin that allows users to pick a color from a pallette.

Reference:
https://github.com/philzet/ColorPick.js

Joget

Creating a Form Field Element Plugin

- Study the existing implementation of other existing form field elements
  - Open “TextField.java”, locate relevant files that made up of the plugin

```
@Override
    public String renderTemplate(FormData formData, Map dataModel) {
        String template = "textField.ftl";
```

```
@Override
    public String getFormBuilderIcon() {
        return
"/plugin/org.joget.apps.form.lib.TextField/images/textField_icon.gif";
    }
```

```
@Override
    public String getPropertyOptions() {
        ...
        return AppUtil.readPluginResource(getClass().getName(),
"/properties/form/textField.json", new Object[]{encryption}, true,
"message/form/TextField");
    }
```

Joget®

Creating a Maven Project for Your Plugin

- Using Joget subproject “wflow-plugin-archetype” to create a Maven project for your plugin.
- **For Windows:**
  1. Create a directory to contain your plugins. (E.g. “C:\joget-projects”).
  2. In Command Prompt, navigate to the newly created directory.
  3. Run “…**\wflow-plugin-archetype\create-plugin.bat" org.joget.tutorial color_picker-pack 8.0-SNAPSHOT”.**
  4. Key in “8.0-SNAPSHOT” for version and “Y” to confirm all the information.
- **For Linux:**
  1. Create a directory at home directory to contain your plugins. (E.g. “~\joget-projects”).
  2. In Command Terminal, go to the created directory.
  3. Run “…**\wflow-plugin-archetype\create-plugin.bat" org.joget.tutorial color_picker-pack 8.0-SNAPSHOT.**
  4. Key in “8.0-SNAPSHOT” for version and “Y” to confirm all the information.

Creating a Maven Project for Your Plugin

- Windows

```
C:\joget-projects>C:\jw-community\trunk\wflow-plugin-archetype\create-plugin.bat org.joget.tutorial color_picker-pack
8.0-SNAPSHOT
[INFO] Scanning for projects...
[INFO]
[INFO] Using the builder org.apache.maven.lifecycle.internal.builder.singlethreaded.SingleThreadedBuilder with a thread
count of 1
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom <<<
[INFO]
[INFO] --- maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Interactive mode
[WARNING] Archetype not found in any catalog. Falling back to central repository (http://repo1.maven.org/maven2).
[WARNING] Use -DarchetypeRepository=<your repository> if archetype's repository is elsewhere.
Downloading: http://repo1.maven.org/maven2/org/joget/wflow-plugin-archetype/8.0-SNAPSHOT/maven-metadata.xml
Downloading: http://repo1.maven.org/org/joget/wflow-plugin-archetype/8.0-SNAPSHOT/maven-metadata.xml
```

Creating a Maven Project for Your Plugin

- Windows

```
[INFO] Using property: groupId = org.joget.tutorial
[INFO] Using property: artifactId = color_picker-pack
Define value for property 'version':  1.0-SNAPSHOT: 8.0-SNAPSHOT
[INFO] Using property: package = org.joget.tutorial
Confirm properties configuration:
groupId: org.joget.tutorial
artifactId: color_picker-pack
version: 8.0-SNAPSHOT
package: org.joget.tutorial
 Y: : Y
[INFO] ------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: wflow-plugin-archetype:8.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: org.joget.tutorial
[INFO] Parameter: packageName, Value: org.joget.tutorial
[INFO] Parameter: package, Value: org.joget.tutorial
[INFO] Parameter: artifactId, Value: color_picker-pack
 [INFO] Parameter: version, Value: 8.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\joget-projects\color_picker-pack
[INFO] ---------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

What is Inside The Maven Project

- Open the newly created project in your IDE (i.e. Netbeans).

Joget

What is Inside The Maven Project

- pom.xml
  - POM stands for "Project Object Model“, an XML representation of a Maven project
  - Used to manage your plugin dependencies jar
- Activator.java
  - Bundle Activator for OSGi framework
  - The activator class is the bundle's hook to the lifecycle layer for management
  - Used to register your plugin class in **start** method

Creating a Plugin Class

1. In your Maven project, create a plugin class call “ColorPicker”
2. Extends **Element** implements **FormBuilderPaletteElement**
3. Import required class file
4. Implement all abstract methods

Register Your Plugin Class

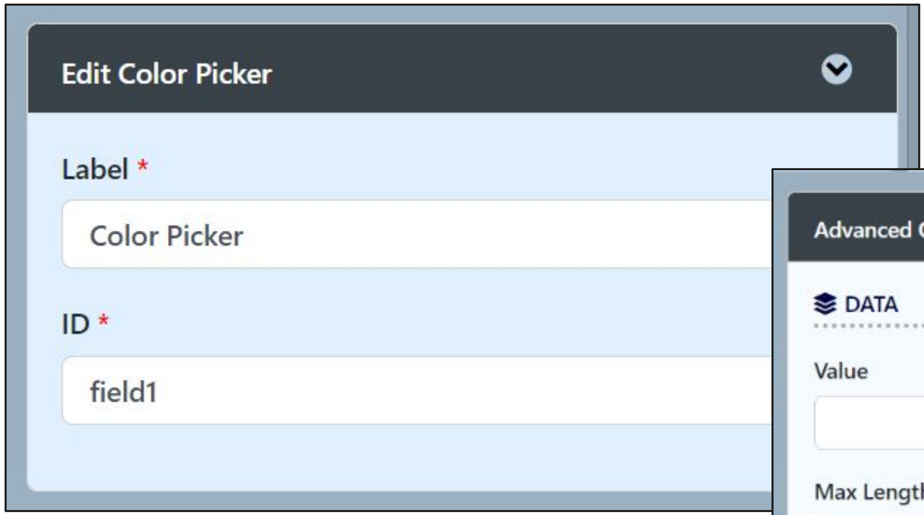
1. Open Activator.java
2. Add the code below to **start** method

```
//Register plugin here
registrationList.add(context.registerService(ColorPicker.class.getName()
, new ColorPicker(), null));
```

Implementing the Methods

- You may copy the existing implementations of **TextField**
- Replace where applicable

Property Options



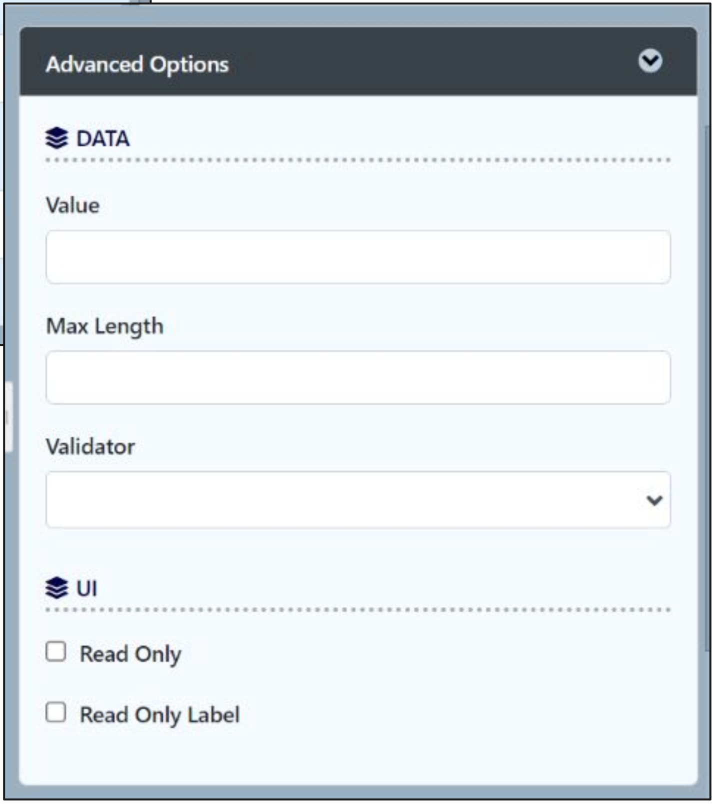
Edit Color Picker

Label *

Color Picker

ID *

field1



Advanced Options

DATA

Value

Max Length

Validator

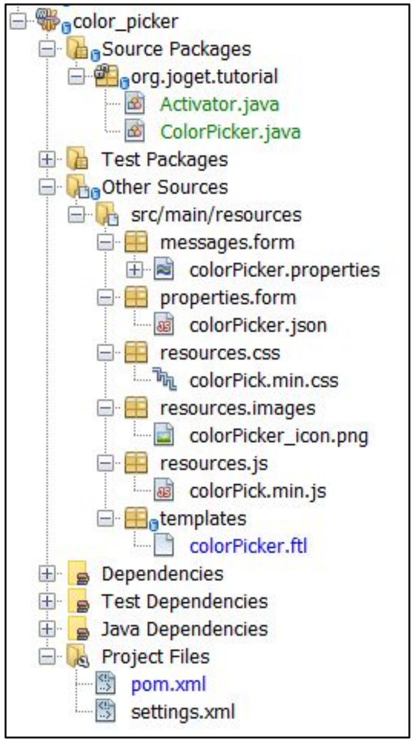
UI

☐ Read Only

☐ Read Only Label

Creating the Supporting Files

- resources/messages/form/**colorPicker.properties**
- resources/properties/form/**colorPicker.json**
- resources/resources/images/**colorPicker_icon.gif**
- resources/resources/js/**colorPick.min.js**
- resources/resources/css/**colorPick.min**
- resources/templates/**colorPicker.ftl**



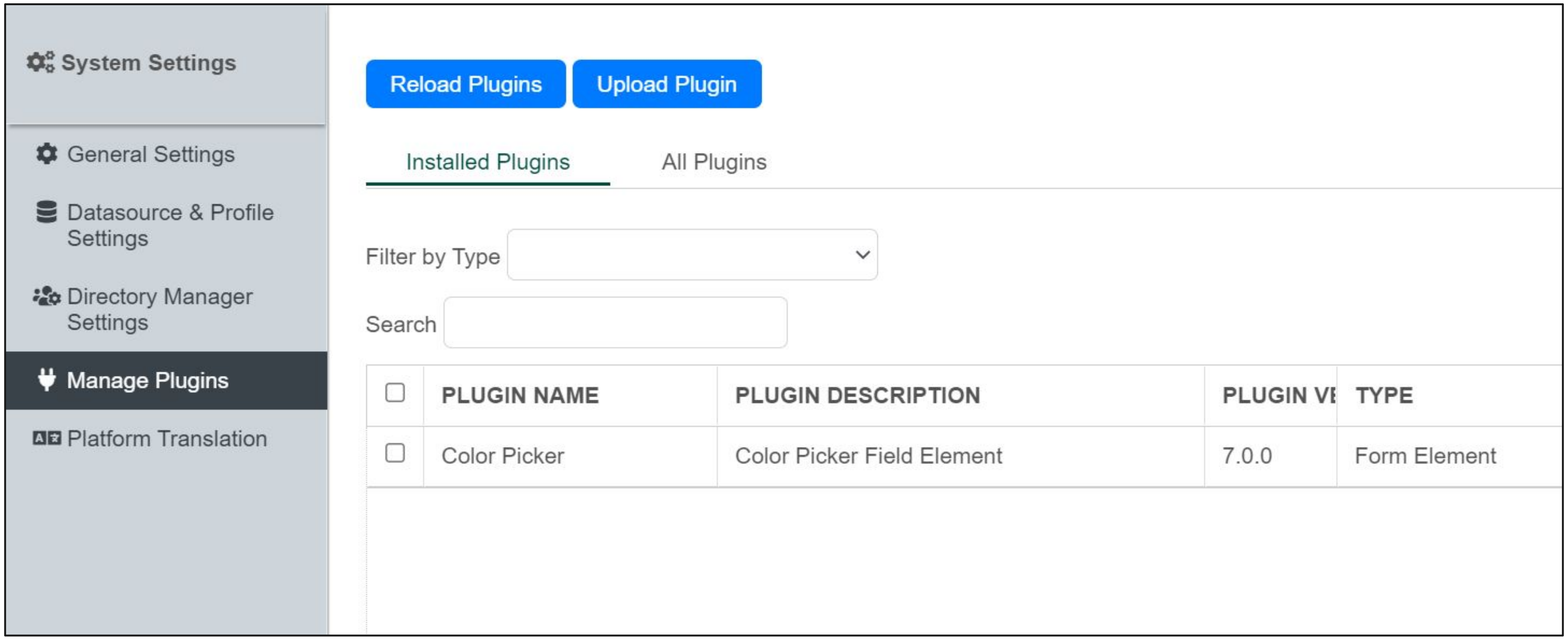
```
color_picker
  Source Packages
    org.joget.tutorial
      Activator.java
      ColorPicker.java
  Test Packages
  Other Sources
    src/main/resources
      messages.form
        colorPicker.properties
      properties.form
        colorPicker.json
      resources.css
        colorPick.min.css
      resources.images
        colorPicker_icon.png
      resources.js
        colorPick.min.js
      templates
        colorPicker.ftl
  Dependencies
  Test Dependencies
  Java Dependencies
  Project Files
    pom.xml
    settings.xml
```

Build and Upload

- When you are ready, build the project by running the

```
mvn clean install
```

- Obtain the .jar file generated and upload into Joget



	PLUGIN NAME	PLUGIN DESCRIPTION	PLUGIN VE	TYPE
☐	Color Picker	Color Picker Field Element	7.0.0	Form Element

Build and Upload

- Test out the form element field in actual Form
- Repeat as many times as needed too until you achieve what you want

Materials

- You may obtain project source code of the sample new theme showcased in this chapter from the file **17-2-color_picker-pack.zip**

Chapter Review

- Be able to create a Form Field Element plugin

Chapter 3
Creating a Datalist Formatter Plugin

What Are We Going To Build?

- Previous chapter we developed a color picker
- We are going to build a **Datalist Formatter** that displays the background color based on the value in color picker

Creating a Datalist Formatter Plugin

- Study the existing implementation of other existing Datalist Formatter elements.
  - Open “DefaultFormatter.java”, locate relevant files that made up of the plugin.

```
@Override
    public String getPropertyOptions() {
        AppDefinition appDef = AppUtil.getCurrentAppDefinition();
        String appId = appDef.getId();
        String appVersion = appDef.getVersion().toString();
        Object[] arguments = new Object[]{appId, appVersion};
        String json = AppUtil.readPluginResource(getClass().getName(),
"/properties/datalist/defaultFormatter.json", arguments, true,
"message/datalist/defaultFormatter");
        return json;
    }
```

Creating a Maven Project for Your Plugin

- Using Joget subproject “wflow-plugin-archetype” to create a Maven project for your plugin.
- **For Windows:**
  1. Create a directory to contain your plugins. (E.g. “C:\joget-projects”)
  2. In Command Prompt, navigate to the newly created directory
  3. Run “… **\wflow-plugin-archetype\create-plugin.bat" org.joget.tutorial color_datalist_formatter-pack 8.0-SNAPSHOT”**
  4. Key in “8.0-SNAPSHOT” for version and “Y” to confirm all the information
- **For Linux:**
  1. Create a directory at home directory to contain your plugins (E.g. “~\joget-projects”)
  2. In Command Terminal, go to the created directory
  3. Run “… **\wflow-plugin-archetype\create-plugin.bat" org.joget.tutorial color_datalist_formatter-pack 8.0-SNAPSHOT**
  4. Key in “8.0-SNAPSHOT” for version and “y” to confirm all the information

Creating a Maven Project for Your Plugin

- Windows

```
C:\joget-projects>C:\jw-community\trunk\wflow-plugin-archetype\create-plugin.bat org.joget.tutorial
color_datalist_formatter-pack 8.0-SNAPSHOT
[INFO] Scanning for projects...
[INFO]
[INFO] Using the builder org.apache.maven.lifecycle.internal.builder.singlethreaded.SingleThreadedBuilder with a thread
count of 1
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom <<<
[INFO]
[INFO] --- maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Interactive mode
[WARNING] Archetype not found in any catalog. Falling back to central repository (http://repo1.maven.org/maven2).
[WARNING] Use -DarchetypeRepository=<your repository> if archetype's repository is elsewhere.
Downloading: http://repo1.maven.org/maven2/org/joget/wflow-plugin-archetype/8.0-SNAPSHOT/maven-metadata.xml
Downloading: http://repo1.maven.org/org/joget/wflow-plugin-archetype/8.0-SNAPSHOT/maven-metadata.xml
```

Creating a Maven Project for Your Plugin

- Windows

```
[INFO] Using property: groupId = org.joget.tutorial
[INFO] Using property: artifactId = color_datalist_formatter-pack
Define value for property 'version':  1.0-SNAPSHOT: 8.0-SNAPSHOT
[INFO] Using property: package = org.joget.tutorial
Confirm properties configuration:
groupId: org.joget.tutorial
artifactId: color_datalist_formatter-pack
version: 8.0-SNAPSHOT
package: org.joget.tutorial
 Y: : Y
[INFO] ------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: wflow-plugin-archetype:8.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: org.joget.tutorial
[INFO] Parameter: packageName, Value: org.joget.tutorial
[INFO] Parameter: package, Value: org.joget.tutorial
[INFO] Parameter: artifactId, Value: color_datalist_formatter-pack
 [INFO] Parameter: version, Value: 8.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\joget-projects\color_datalist_formatter-pack
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

What Is Inside the Maven Project

- Open the newly created project in your IDE (i.e. Netbeans)

What Is Inside the Maven Project

- pom.xml
  - POM stands for "Project Object Model“, an XML representation of a Maven project
  - Used to manage your plugin dependencies jar
- Activator.java
  - Bundle Activator for OSGi framework
  - The activator class is the bundle's hook to the lifecycle layer for management
  - Used to register your plugin class in **start** method

Creating a Plugin Class

1. In your Maven project, create a plugin class call “ColorDatalistFormatter”
2. Extends **DataListColumnFormatDefault**
3. Import required class file
4. Implement all abstract methods

Register Your Plugin Class

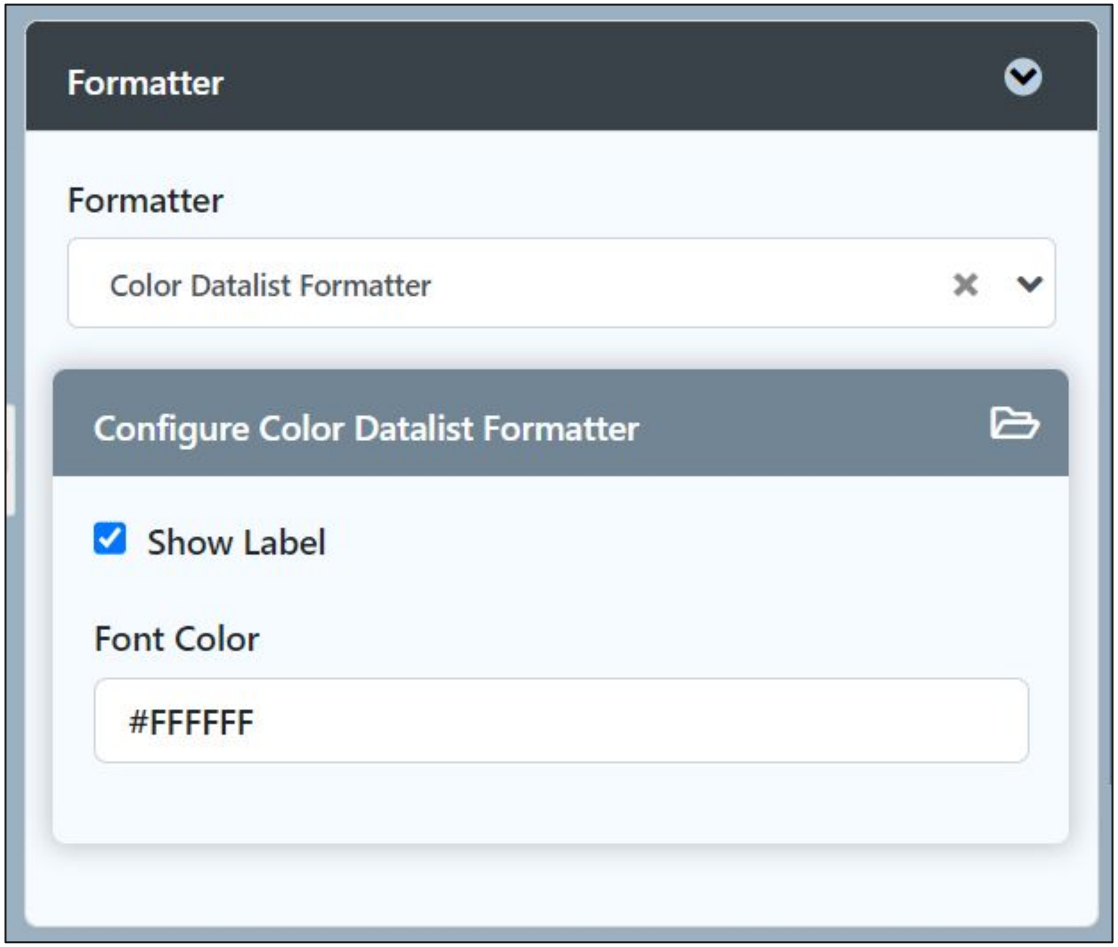
1. Open Activator.java
2. Add the code below to **start** method

```
//Register plugin here
registrationList.add(context.registerService(ColorDatalistFormatter.
class.getName(), new ColorDatalistFormatter(), null));
```

Implementing the Methods

- You may copy the existing implementations of **Default Formatter**
- Replace where applicable

Property Options



Formatter

Formatter: Color Datalist Formatter

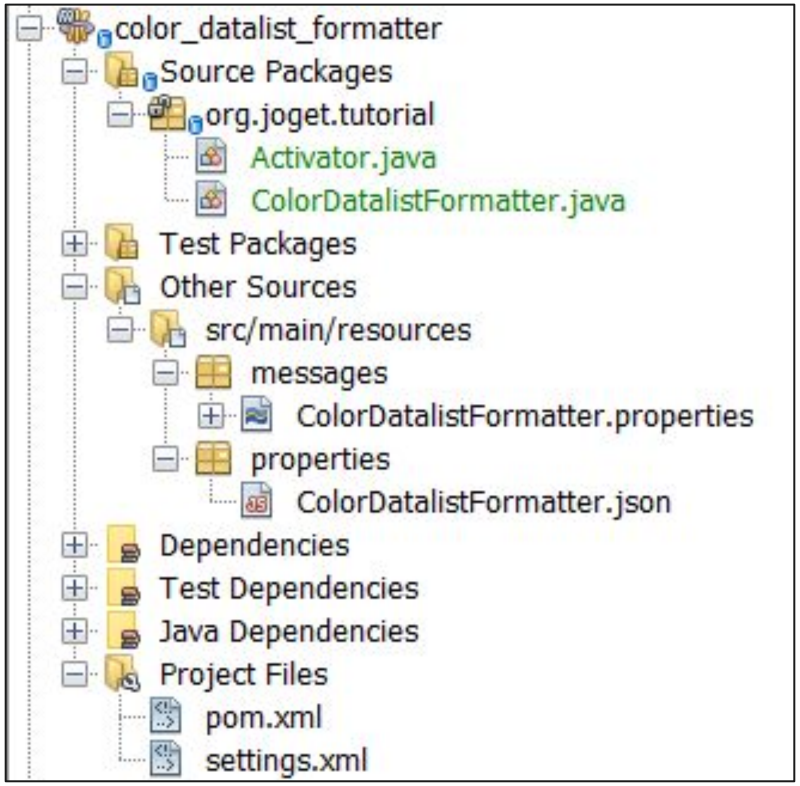
Configure Color Datalist Formatter

[✓] Show Label

Font Color: #FFFFFF

Creating the Supporting Files

- `resources/messages/form/`**`ColorDatalistFormatter.properties`**
- `resources/properties/form/`**`ColorDatalistFormatter.json`**



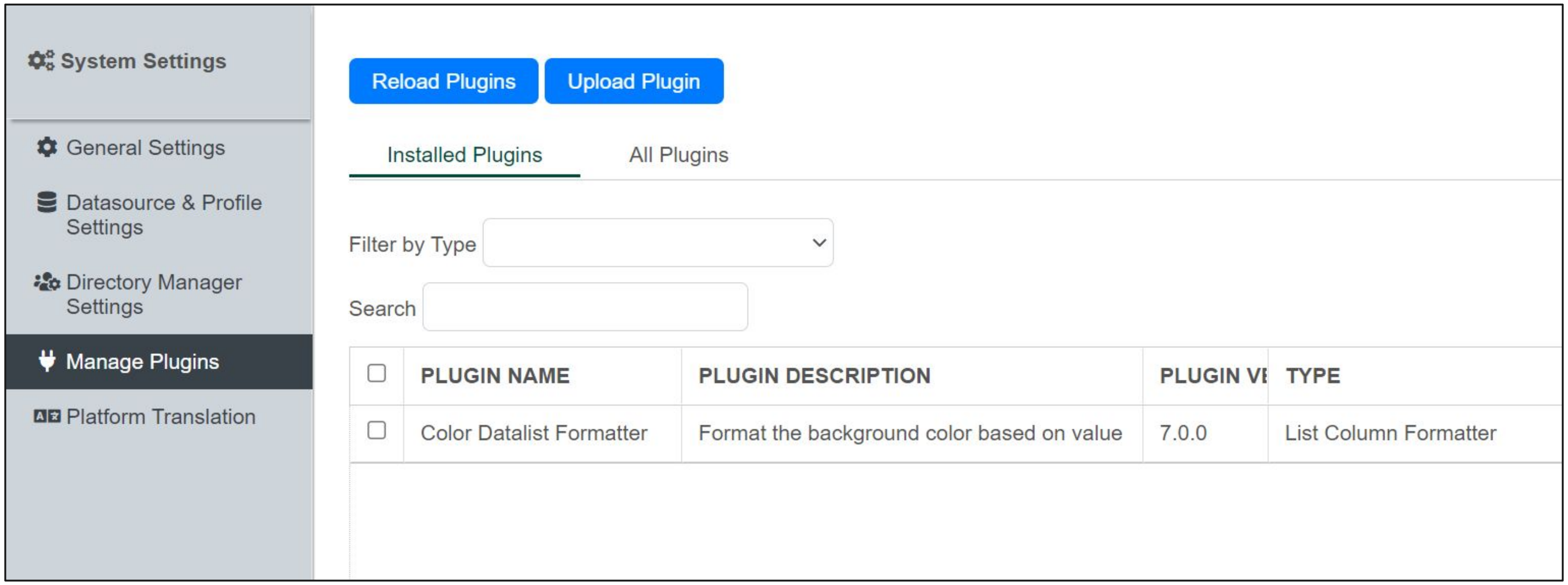
```
color_datalist_formatter
  Source Packages
    org.joget.tutorial
      Activator.java
      ColorDatalistFormatter.java
  Test Packages
  Other Sources
    src/main/resources
      messages
        ColorDatalistFormatter.properties
      properties
        ColorDatalistFormatter.json
  Dependencies
  Test Dependencies
  Java Dependencies
  Project Files
    pom.xml
    settings.xml
```

Build and Upload

- When you are ready, build the project by running the following command in your project path

```
mvn clean install
```

- Obtain the .jar file generated and upload into Joget



	PLUGIN NAME	PLUGIN DESCRIPTION	PLUGIN VE	TYPE
☐	Color Datalist Formatter	Format the background color based on value	7.0.0	List Column Formatter

Build and Upload

- Test out the List formatter in actual List
- Repeat as many times as needed too until you achieve what you want

Materials

- You may obtain project source code of the sample new theme showcased in this chapter from the file: **17-3-color_datalist_formatter-pack.zip**

Chapter Review

- Be able to create a List Formatter plugin

Chapter 4 Creating a Custom HTML UI Menu Plugin

What Are We Going To Build?

- We are going to build a UI Menu that allows users to input HTML scripts similar to Custom HTML in Form Builder

Joget

Creating a UI Menu Plugin

- Study the existing implementation of other existing form field elements
  - Open “UserProfileMenu.java”, locate relevant files that made up of the plugin

```
@Override
   public String getRenderPage() {
      ..
       String content = pluginManager.getPluginFreeMarkerTemplate(model, getClass().getName(),
"/templates/userProfile.ftl", null);
       return content;
      }
```

```
@Override
   public String getIcon() {
       return "/plugin/org.joget.plugin.enterprise.UserProfileMenu/images/grid_icon.gif"  ;
   }
```

```
public String getPropertyOptions() {
       return AppUtil.readPluginResource(getClass().getName(),
"/properties/userview/userProfileMenu.json" , null, true, "message/userview/userProfileMenu");
   }
```

Creating a Maven Project for Your Plugin

- Using Joget subproject “wflow-plugin-archetype” to create a Maven project for your plugin.
- **For Windows:**
  1. Create a directory to contain your plugins. (E.g. “C:\joget-projects”)
  2. In Command Prompt, navigate to the newly created directory.
  3. Run “…**\wflow-plugin-archetype\create-plugin.bat" org.joget.tutorial classic_html_menu”**
  4. Key in “8.0-SNAPSHOT” for version and “Y” to confirm all the information
- **For Linux:**
  1. Create a directory at home directory to contain your plugins. (E.g. “~\joget-projects”)
  2. In Command Terminal, go to the created directory.
  3. Run “…**\wflow-plugin-archetype\create-plugin.bat" org.joget.tutorial classic_html_menu**
  4. Key in “8.0-SNAPSHOT” for version and “Y” to confirm all the information

Creating a Maven Project for Your Plugin

- Windows

```
C:\joget-projects>C:\jw-community\trunk\wflow-plugin-archetype\create-plugin.bat org.joget.tutorial classic_html_menu
[INFO] Scanning for projects...
[INFO]
[INFO] Using the builder org.apache.maven.lifecycle.internal.builder.singlethreaded.SingleThreadedBuilder with a thread
count of 1
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom <<<
[INFO]
[INFO] --- maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Interactive mode
[WARNING] Archetype not found in any catalog. Falling back to central repository (http://repo1.maven.org/maven2).
[WARNING] Use -DarchetypeRepository=<your repository> if archetype's repository is elsewhere.
Downloading: http://repo1.maven.org/maven2/org/joget/wflow-plugin-archetype/8.0-SNAPSHOT/maven-metadata.xml
Downloading: http://repo1.maven.org/org/joget/wflow-plugin-archetype/8.0-SNAPSHOT/maven-metadata.xml
```

Creating a Maven Project for Your Plugin

- Windows

```
[INFO] Using property: groupId = org.joget.tutorial
[INFO] Using property: artifactId = classic_html_menu
Define value for property 'version':  1.0-SNAPSHOT: 8.0-SNAPSHOT
[INFO] Using property: package = org.joget.tutorial
Confirm properties configuration:
groupId: org.joget.tutorial
artifactId: classic_html_menu
version: 8.0-SNAPSHOT
package: org.joget.tutorial
 Y: : Y
[INFO] ------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: wflow-plugin-archetype:8.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: org.joget.tutorial
[INFO] Parameter: packageName, Value: org.joget.tutorial
[INFO] Parameter: package, Value: org.joget.tutorial
[INFO] Parameter: artifactId, Value: classic_html_menu
 [INFO] Parameter: version, Value: 8.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\joget-projects\classic_html_menu
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

What Is Inside the Maven Project

- Open the newly created project in your IDE (i.e. Netbeans)

What Is Inside the Maven Project

- pom.xml
  - POM stands for "Project Object Model“, an XML representation of a Maven project
  - Used to manage your plugin dependencies jar
- Activator.java
  - Bundle Activator for OSGi framework
  - The activator class is the bundle's hook to the lifecycle layer for management.
  - Used to register your plugin class in **start** method

Creating a Plugin Class

1. In your Maven project, create a plugin class called as “ClassicHTMLMenu”
2. Extends **UserviewMenu**
3. Import required class file

Register Your Plugin Class

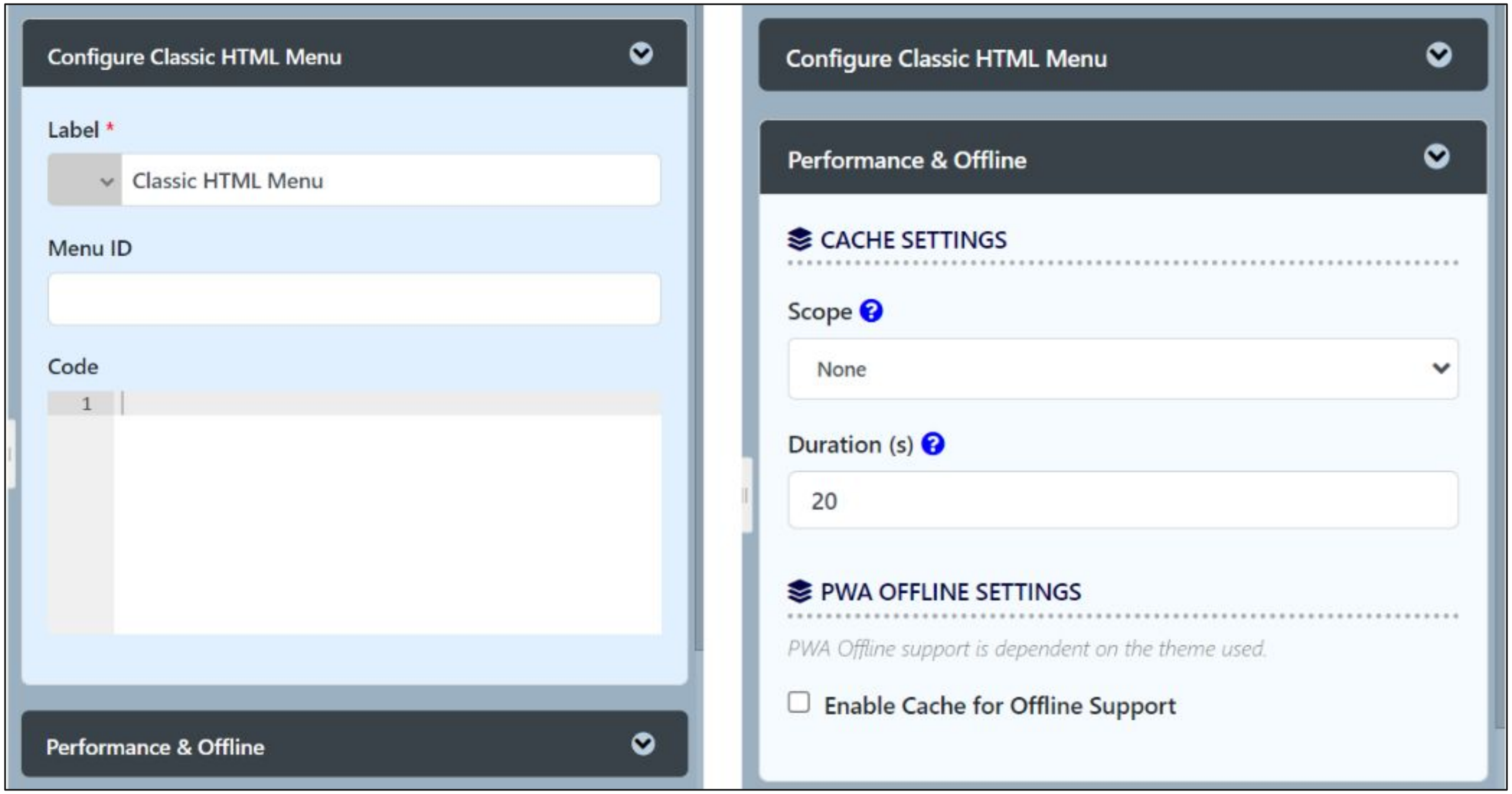
1. Open Activator.java
2. Add the code below to **start** method

```
//Register plugin here
registrationList.add(context.registerService(ClassicHTMLMenu.
class.getName(), new ClassicHTMLMenu(), null));
```

Implementing the Methods

- You may copy the existing implementations of **UI Menu**
- Replace where applicable

Property Options



Configure Classic HTML Menu

Label *

Classic HTML Menu

Menu ID

Code

Performance & Offline

Configure Classic HTML Menu

Performance & Offline

CACHE SETTINGS

Scope: None

Duration (s): 20

PWA OFFLINE SETTINGS

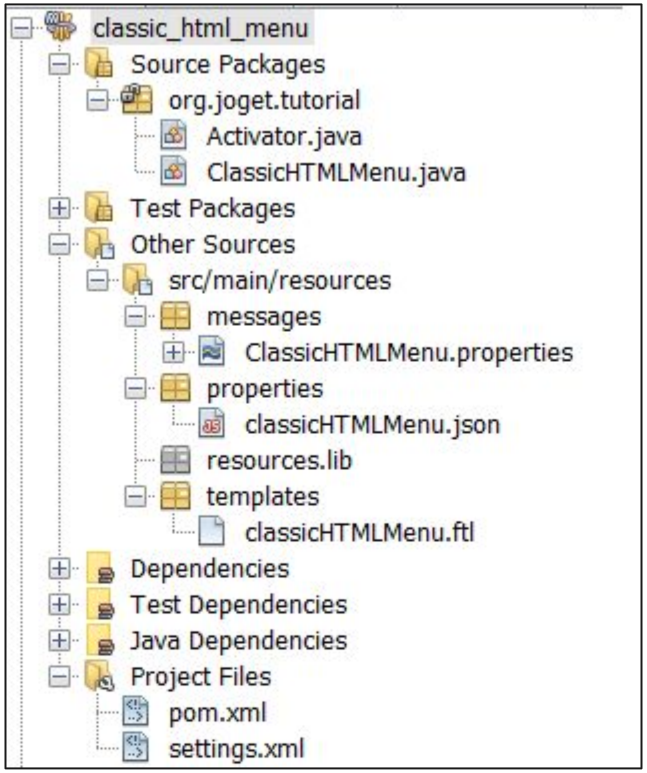
PWA Offline support is dependent on the theme used.

☐ Enable Cache for Offline Support

Joget

Creating the Supporting Files

- resources/messages/**ClassicHTMLMenu.properties**
- resources/properties/**classicHTMLMenu.json**
- resources/templates/**classicHTMLMenu.ftl**



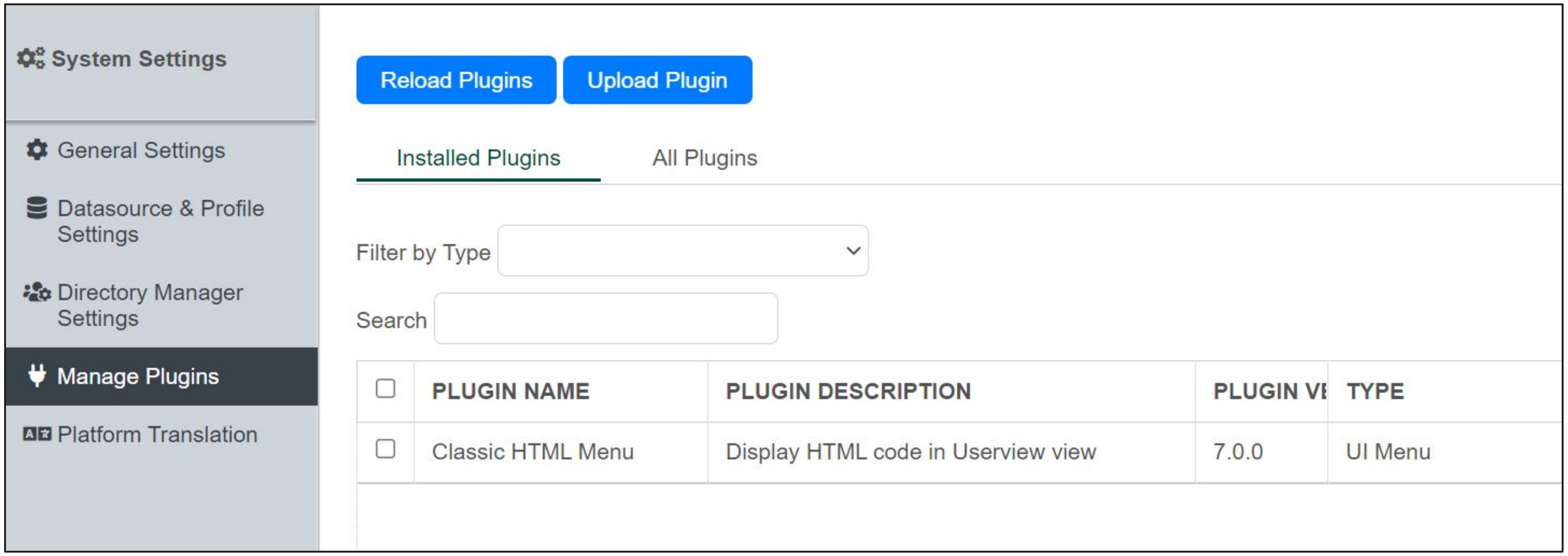
```
classic_html_menu
  Source Packages
    org.joget.tutorial
      Activator.java
      ClassicHTMLMenu.java
  Test Packages
  Other Sources
    src/main/resources
      messages
        ClassicHTMLMenu.properties
      properties
        classicHTMLMenu.json
      resources.lib
      templates
        classicHTMLMenu.ftl
  Dependencies
  Test Dependencies
  Java Dependencies
  Project Files
    pom.xml
    settings.xml
```

Build And Upload

- When you are ready, build the project by running the following command in your project path

```
mvn clean install
```

- Obtain the .jar file generated and upload into Joget



	PLUGIN NAME	PLUGIN DESCRIPTION	PLUGIN VE	TYPE
☐	Classic HTML Menu	Display HTML code in Userview view	7.0.0	UI Menu

Build And Upload

- Test out the UI menu
- Repeat as many times as needed too until you achieve what you want

Materials

- You may obtain project source code of the sample new theme showcased in this chapter from the file **17-4-classic_html_menu.zip**

Chapter Review

- Be able to create a Custom HTML UI Menu plugin

Chapter 5
Generate & Build Plugin via Docker

Building Plugin Using Docker

- Without needing to prepare development environment as described in Module 16 (Java, Maven, Install External Libraries), you can generate and build plugin by using a special docker image prepared

Links

- Please refer to https://dev.joget.org/community/display/DX8/Build+Plugin+Source+Code+using+Docker to continue

- Reference: https://hub.docker.com/r/jogetworkflow/docker-maven-joget

Note: For long term development, still recommended to setup a proper environment as per the KB

Module Review

1. Introduction
2. Creating a Form Field Element Plugin
3. Creating a List Formatter Plugin
4. Creating a Custom HTML UI Menu plugin
5. Generate & Build Plugin via Docker

Joget

Recommended Further Learning

- Study on how other plugin types are implemented
  - Check out the Joget Marketplace at http://marketplace.joget.org/
  - Check out the Joget Knowledge Base – Developer Guide https://dev.joget.org/community/display/DX8/Developing+Plugins

Stay Connected With Joget

- http://www.joget.org
- http://community.joget.org
- http://twitter.com/jogetworkflow
- http://facebook.com/jogetworkflow
- http://youtube.com/jogetworkflow
- http://slideshare.net/joget