RADS

Building Plugins

https://github.com/radservice/rad-community/fork

Last Revised on Oct 2021

Prerequisites

1. Basic web application development knowledge.
2. Java web application programming knowledge.
3. Understanding on RADS 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.

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://docs.rads.purwana.net/Introduction+to+Plugin+Architecture

Plugin Abstract Classes and Interface

- **Deadline Plugins**

  extends net.purwana.rads.workflow.model.DefaultDeadlinePlugin
- **Process Participant Plugins**

  extends net.purwana.rads.workflow.model.DefaultParticipantPlugin
- **Process Tool / Post Form Submission Processing Plugins**

  extends net.purwana.rads.plugin.base.DefaultApplicationPlugin
- **Form Field Element Plugins**

  extends net.purwana.rads.apps.form.model.Element

  implements

  net.purwana.rads.apps.form.model.FormBuilderPaletteElement

Plugin Abstract Classes and Interface

- **Form Load Binder Plugins**

  extends net.purwana.rads.apps.form.model.FormBinder implements net.purwana.rads.apps.form.model.FormLoadBinder, net.purwana.rads.apps.form.model.FormLoadElementBinder

- **Form Options Binder Plugins**

  extends net.purwana.rads.apps.form.model.FormBinder implements net.purwana.rads.apps.form.model.FormLoadOptionsBinder

- **Form Store Binder Plugins**

  extends net.purwana.rads.apps.form.model.FormBinder implements net.purwana.rads.apps.form.model.FormStoreBinder, net.purwana.rads.apps.form.model.FormStoreElementBinder

Plugin Abstract Classes and Interface

- **Form Validator Plugins**

  extends net.purwana.rads.apps.form.model.FormValidator
- **Datalist Action Plugins**

  extends net.purwana.rads.apps.datalist.model.DataListActionDefault
- **Datalist Binder Plugins**

  extends net.purwana.rads.apps.datalist.model.DataListBinderDefault
- **Datalist Column Formatter Plugins**

  extends net.purwana.rads.apps.datalist.model.DataListColumnFormatDefault
- **Userview Menu Plugins**

  extends net.purwana.rads.apps.userview.model.UserviewMenu

Plugin Abstract Classes and Interface

- **Userview Permission Plugins**

  extends net.purwana.rads.apps.userview.model.UserviewPermission
- **Userview Theme Plugins**

  extends net.purwana.rads.apps.userview.model.UserviewTheme
- **Audit Trail Plugins**

  extends net.purwana.rads.plugin.base.DefaultAuditTrailPlugin
- **Hash Variable Plugins**

  extends net.purwana.rads.apps.app.model.DefaultHashVariablePlugin
- **Directory Manager Plugins**

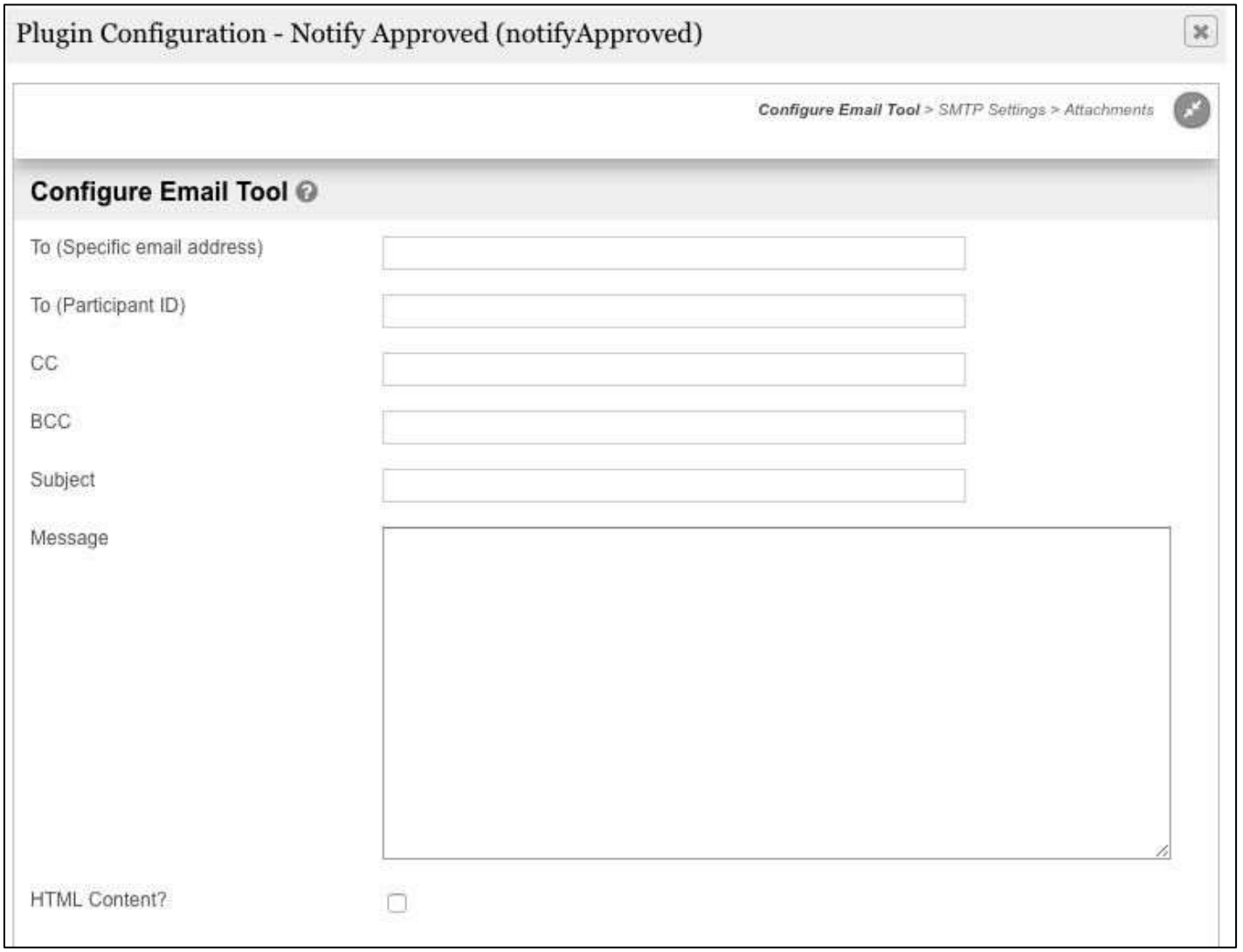
  extends net.purwana.rads.plugin.base.ExtDefaultPlugin implements net.purwana.rads.directory.model.service.DirectoryManagerPlugin, net.purwana.rads.plugin.property.model.PropertyEditable

Introduction to Property Options

- Each plugin uses the Property Options template scheme to provide an interface for **app designers** to configure your plugin.

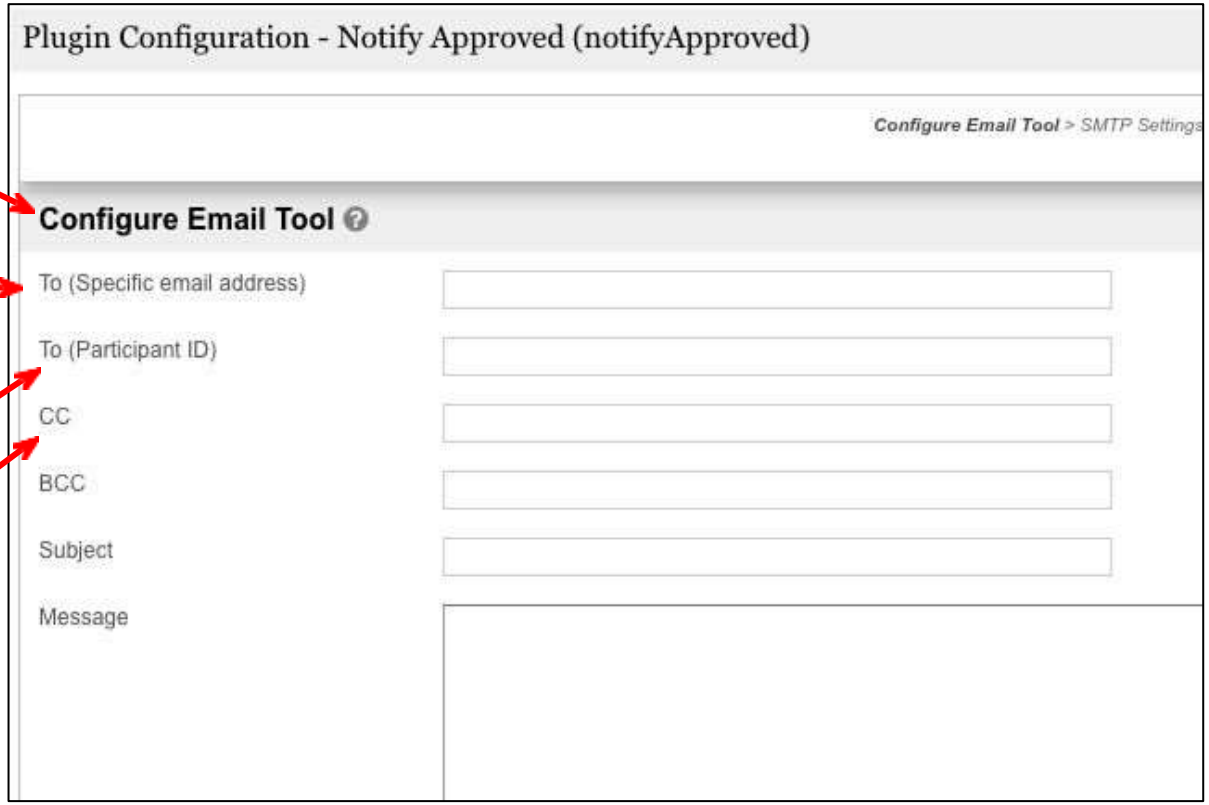
- Reference: https://docs.rads.purwana.net/Plugin+Properties+Options

Property Options - Email Tool



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"
      },
```



https://github.com/radservice/rad-community/blob/master/rad-core/src/main/resources/properties/app/emailTool.json

Property Options

- JSON format

```
[
    {
        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 …
]
```

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

Property Options Types

- From V4
  - Check Box– **CheckBox**
  - Element Select Box – **ElementSelect**
  - Grid – **Grid**
  - Hidden Field– **Hidden**
  - HTML Editor– **HtmlEditor**
  - Multi Select Box– **MultiSelect**
  - Password Field – **Password**
  - Radio Button – **Radio**
  - Readonly Text Field– **Readonly**
  - Select Box– **SelectBox**
  - Text Area – **TextArea**
  - Text Field – **TextField**

Property Options Types

- New in V5
  - Code Editor – **CodeEditor**
  - Combine Grid – **GridCombine**
  - Fixed Row Grid – **GridFixedRow**
  - Header – **Header**
  - Label – **Label**

Property Options Types

- New in 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**

Property Options Types

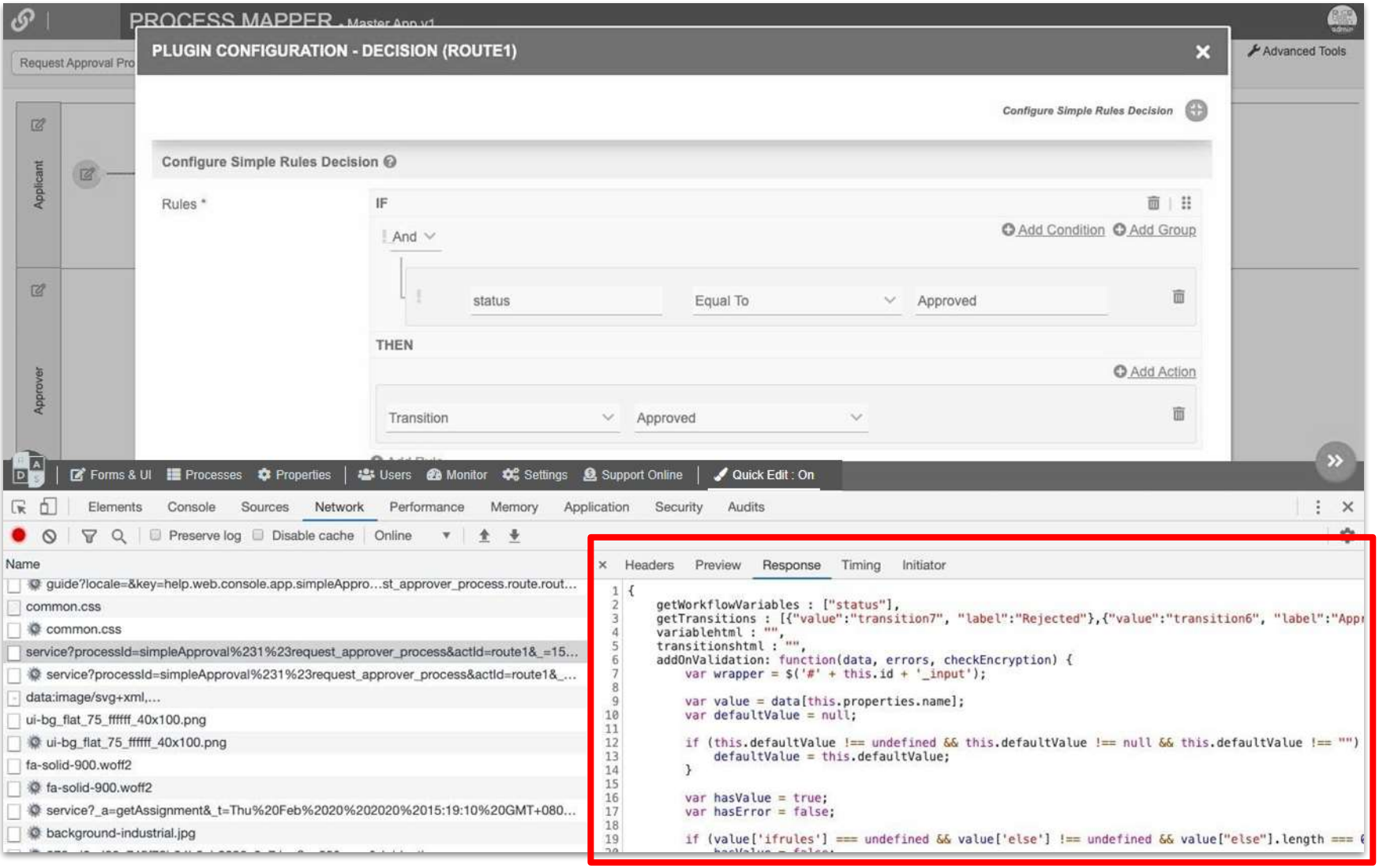
- New in RADS
  - custom - **Custom Scripting**
  - elementmultiselect - **Multiselect in Grid interface**
  - number - **Number**
  - ImageRadio - **Image Radio**
  - Repeater

Please refer to https://docs.rads.purwana.net/Plugin+Properties+Options latest documentation.

RADS - Property Options Types - custom

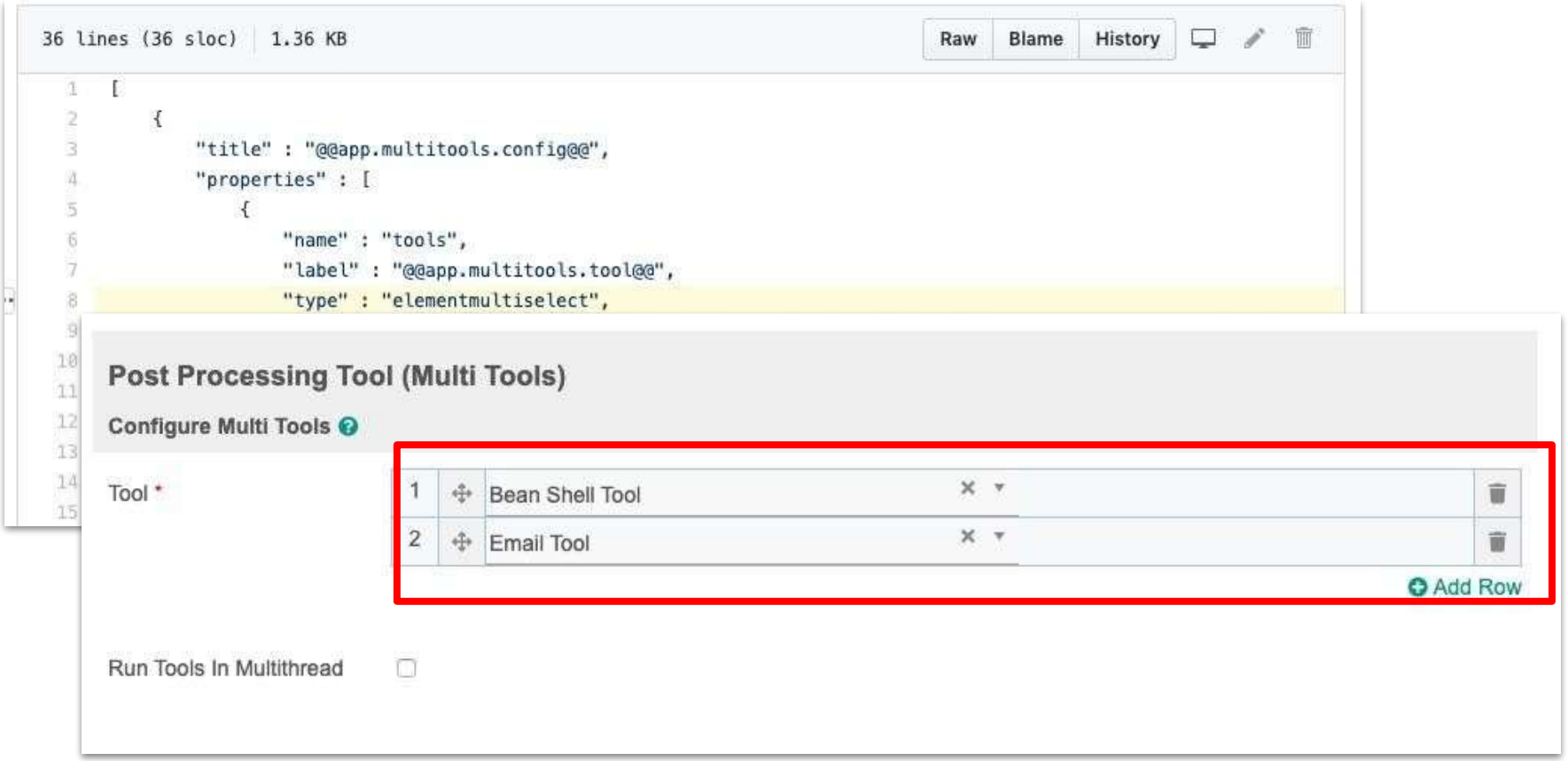
- Custom Scripting - For complex use cases such as the one used in Rules Decision Plugin.
- https://github.com/radservice/rad-community/blob/master/rad-core/src/main/resources/properties/app/rulesDecisionPlugin.json
- Calls line 7 “script_url” to fetch script to load.

RADS - Property Options Types - custom



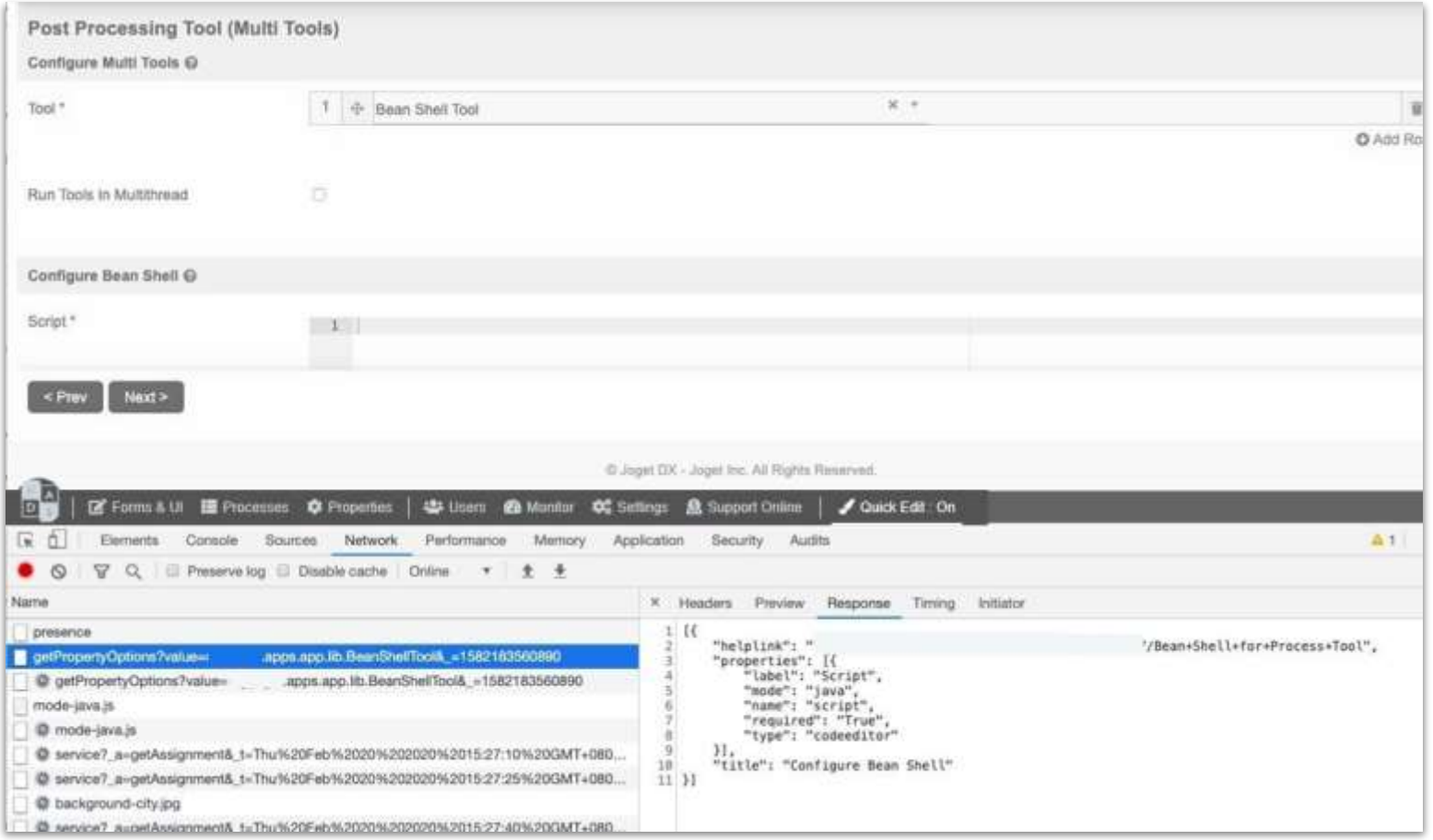
RADS - Property Options Types - elementmultiselect

- Grid-like selections with ability to support external call to dynamically populate more properties.
- https://github.com/radservice/rad-community/blob/master/rad-core/src/main/resources/properties/app/multiTools.json#L8



RADS - Property Options Types - elementmultiselect

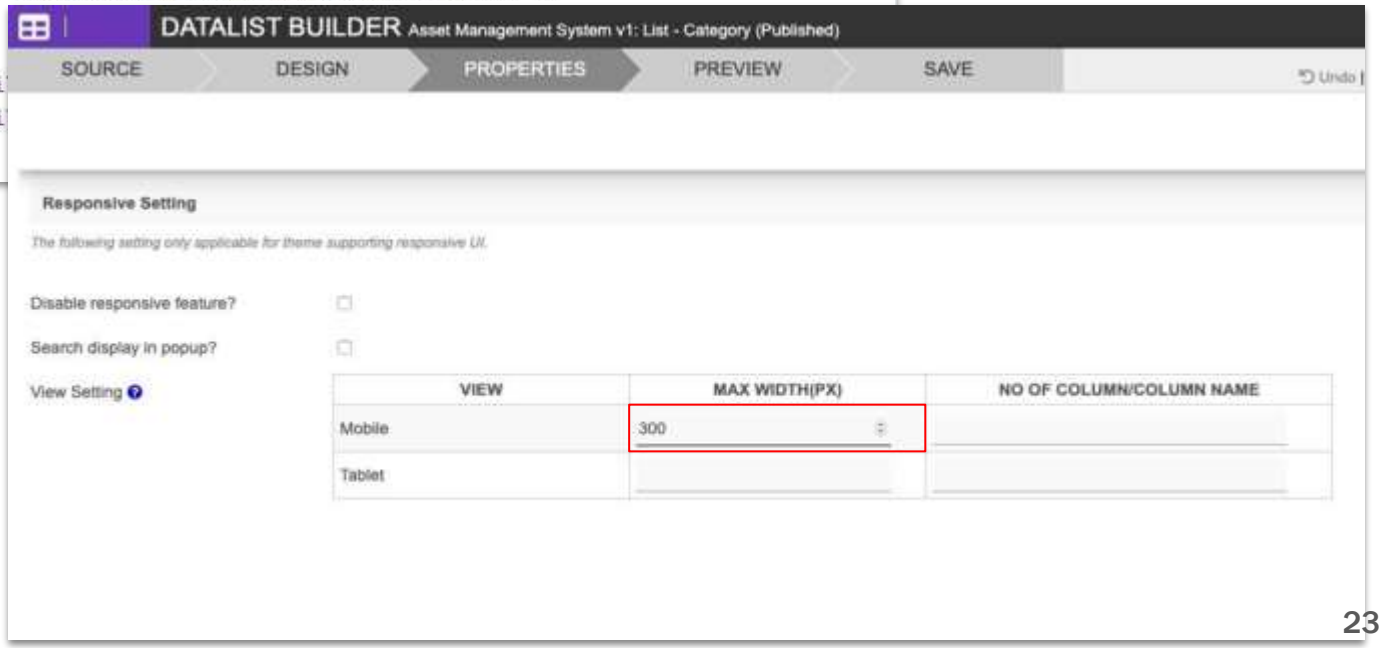
- https://github.com/radservice/rad-community/blob/master/rad-core/src/main/resources/properties/app/multiTools.json#L8



RADS - Property Options Types - Number

- https://github.com/radservice/rad-community/blob/master/rad-consoleweb/src/main/webapp/js/dbuilder.core.js#L1565

```
1559            {label: get_dbuilder_msg('dbuilder.responsiveView'),
1560              name: 'responsiveView',
1561              description : get_dbuilder_msg('dbuilder.responsiveView.desc'),
1562              type: 'gridfixedrow',
1563              columns : [
1564                  {key : 'view', label : get_dbuilder_msg('dbuilder.view')},
1565                  {key : 'breakpoint', label : get_dbuilder_msg('dbuilder.breakpoint'), type : 'number'},
1566                  {key : 'columns', label : get_dbuilder_msg('dbuilder.columns')}
1567              ],
1568              rows: [
1569                  {label : get_dbui
1570                  {label : get_dbui
1571              ]}
```



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
}
```

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=
net.purwana.rads.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");
```

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

Path to resource file in resources folder

Argument used for String.format()

Plugin Class Name

Remove new line char

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 i18n 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

- More at

https://docs.rads.purwana.net/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/rad/web/json/plugin/net.purwana.rads.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/rad/web/json/app/crm/1/plugin/net.purwana.rads.sample.lib.SimpleFormElement/service?say_something=Hello World`

Plugin Web Support

- Implements net.purwana.rads.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://docs.rads.purwana.net/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

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/net.purwana.rads.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");
    }
```

Creating a Maven Project for Your Plugin

- Using RADS subproject “rad-plugin-archetype” to create a Maven project for your plugin.
- **For Windows:**
  1. Create a directory to contain your plugins. (E.g. “C:\RADS-projects”).
  2. In Command Prompt, navigate to the newly created directory.
  3. Run “…**\rad-plugin-archetype\create-plugin.bat" net.purwana.rads.tutorial color_picker-pack 7.0-SNAPSHOT”.**
  4. Key in “7.0-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. “~\RADS-projects”).
  2. In Command Terminal, go to the created directory.
  3. Run “…**\rad-plugin-archetype\create-plugin.bat" net.purwana.rads.tutorial color_picker-pack 7.0-SNAPSHOT.**
  4. Key in “7.0-SNAPSHOT” for version and “y” to confirm all the information.

Creating a Maven Project for Your Plugin

- Windows

```
C:\RADS-projects>C:\rad-community\trunk\rad-plugin-archetype\create-plugin.bat net.purwana.rads.tutorial
color_picker-pack  7.0-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/net/purwana/rads/rad-plugin-archetype/7.0-SNAPSHOT/maven-
metadata.xml  Downloading: http://repo1.maven.org/net/purwana/rads/rad-plugin-archetype/7.0-SNAPSHOT/maven-
metadata.xml
```

Creating a Maven Project for Your Plugin

- Windows

```
[INFO] Using property: groupId = net.purwana.rads.tutorial
[INFO] Using property: artifactId = color_picker-pack
Define value for property 'version':  1.0-SNAPSHOT: 7.0-SNAPSHOT
[INFO] Using property: package =
net.purwana.rads.tutorial Confirm properties
configuration:
groupId:
net.purwana.rads.tutorial
artifactId: color_picker-pack
version: 7.0-SNAPSHOT
package:
net.purwana.rads.tutorial  Y: : Y
[INFO] ____________________________________________________________
[INFO] Using following parameters for creating project from Old (1.x) Archetype: rad-plugin-archetype:7.0-SNAPSHOT [INFO] __
[INFO] Parameter: groupId, Value: net.purwana.rads.tutorial
[INFO] Parameter: packageName, Value:
net.purwana.rads.tutorial [INFO] Parameter: package, Value:
net.purwana.rads.tutorial  [INFO] Parameter: artifactId, Value:
color_picker-pack  [INFO] Parameter: version, Value: 7.0-
SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\RADS-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).

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

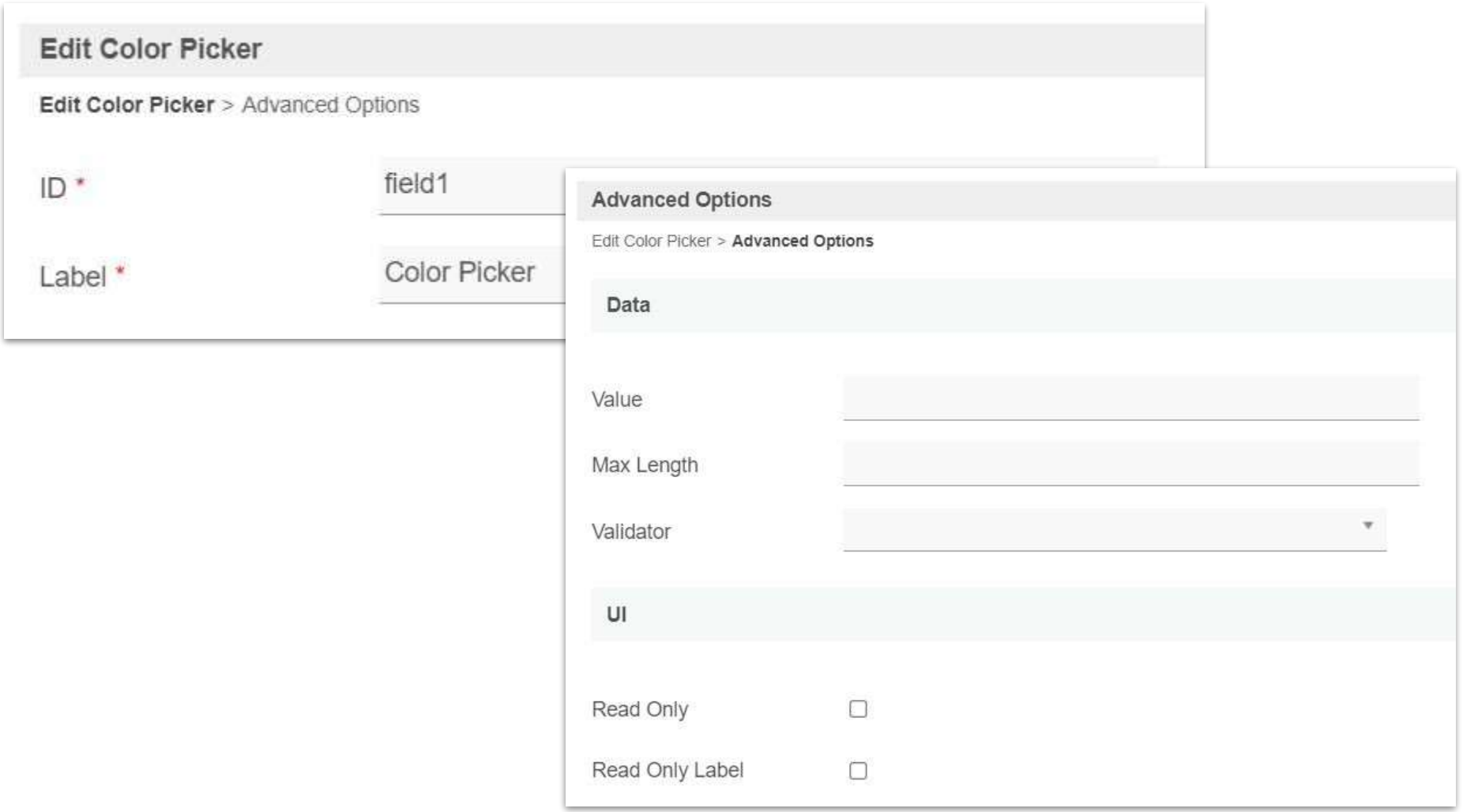
1. Open Activator.java.
2. Add the code below to **start** method.

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

Implementing the Methods

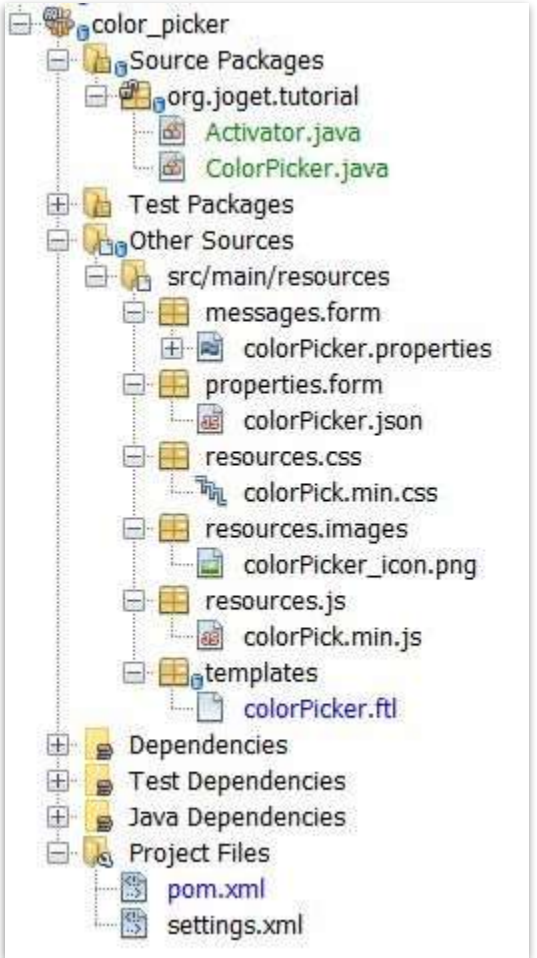
- You may copy the existing implementations of **TextField.**
- Replace where applicable.

Property Options



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/templates/css/**colorPick.min**
- resources/templates/**colorPicker.ftl**

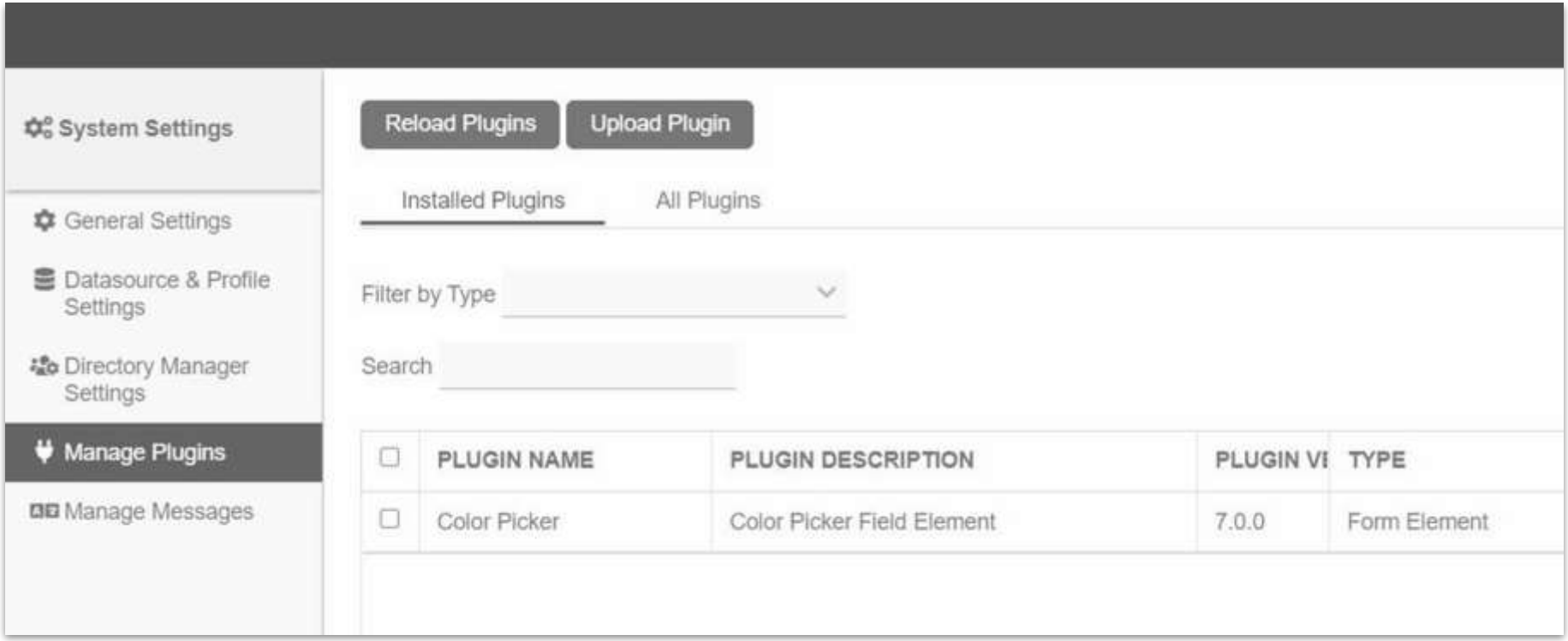


Build and Upload

- 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 RADS.



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 RADS subproject “rad-plugin-archetype” to create a Maven project for your plugin.
- **For Windows:**
  1. Create a directory to contain your plugins. (E.g. “C:\RADS-projects”)
  2. In Command Prompt, navigate to the newly created directory.
  3. Run “…**\rad-plugin-archetype\create-plugin.bat" net.purwana.rads.tutorial color_datalist_formatter-pack 7.0-SNAPSHOT”**
  4. Key in “7.0-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. “~\RADS-projects”)
  2. In Command Terminal, go to the created directory.
  3. Run “…**\rad-plugin-archetype\create-plugin.bat" net.purwana.rads.tutorial color_datalist_formatter-pack 7.0-SNAPSHOT**
  4. Key in “7.0-SNAPSHOT” for version and “y” to confirm all the information

Creating a Maven Project for Your Plugin

- Windows

```
C:\RADS-projects>C:\jw-community\trunk\rad-plugin-archetype\create-plugin.bat
net.purwana.rads.tutorial color_datalist_formatter-pack 7.0-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/net/purwana/rads/rad-plugin-archetype/7.0-SNAPSHOT/maven-
metadata.xml  Downloading: http://repo1.maven.org/net/purwana/rads/rad-plugin-archetype/7.0-SNAPSHOT/maven-
metadata.xml
```

Creating a Maven Project for Your Plugin

- Windows

```
[INFO] Using property: groupId = net.purwana.rads.tutorial
[INFO] Using property: artifactId = color_datalist_formatter-pack
Define value for property 'version':  1.0-SNAPSHOT: 7.0-SNAPSHOT
[INFO] Using property: package = net.purwana.rads.tutorial
Confirm properties configuration:
groupId: net.purwana.rads.tutorial
artifactId: color_datalist_formatter-pack
version: 7.0-SNAPSHOT
package:
 net.purwana.rads.tutorial
 Y: : Y
[INFO] ----------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: rad-plugin-archetype:7.0-SNAPSHOT
[INFO] ----------------------------------------------------------
[INFO] Parameter: groupId, Value:
net.purwana.rads.tutorial  [INFO] Parameter:
packageName, Value: net.purwana.rads.tutorial [INFO]
Parameter: package, Value: net.purwana.rads.tutorial
[INFO] Parameter: artifactId, Value: color_datalist_formatter-pack
 [INFO] Parameter: version, Value: 7.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\RADS-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.cla
ss.getName(), new ColorDatalistFormatter(), null));
```

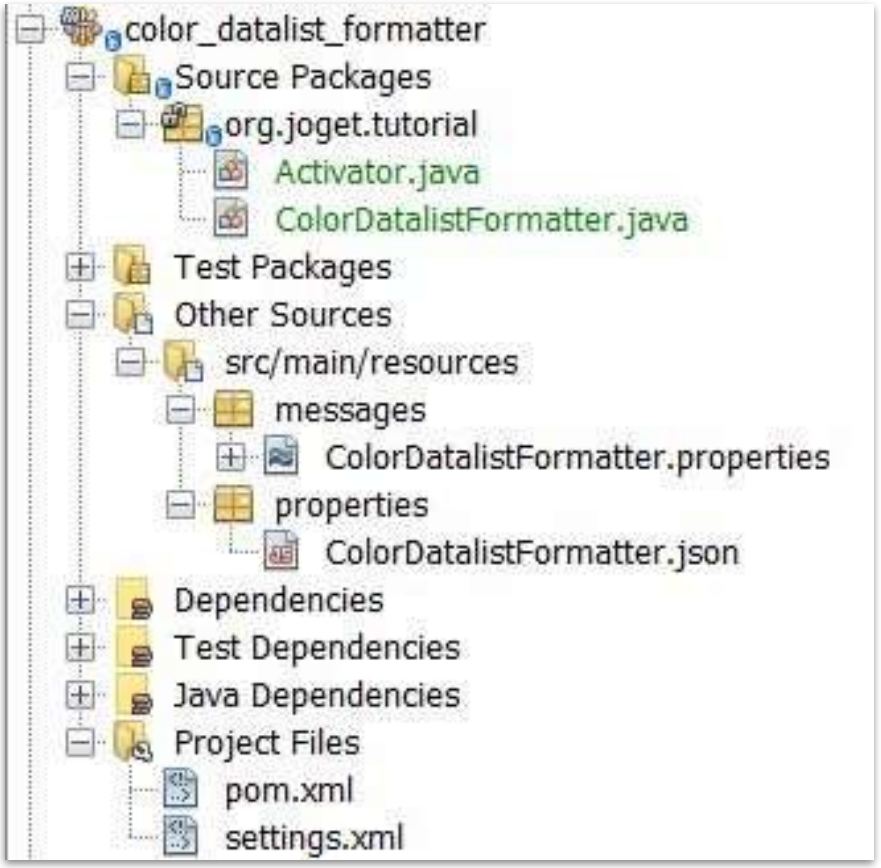
Implementing the Methods

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

Property Options

Creating the Supporting Files

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

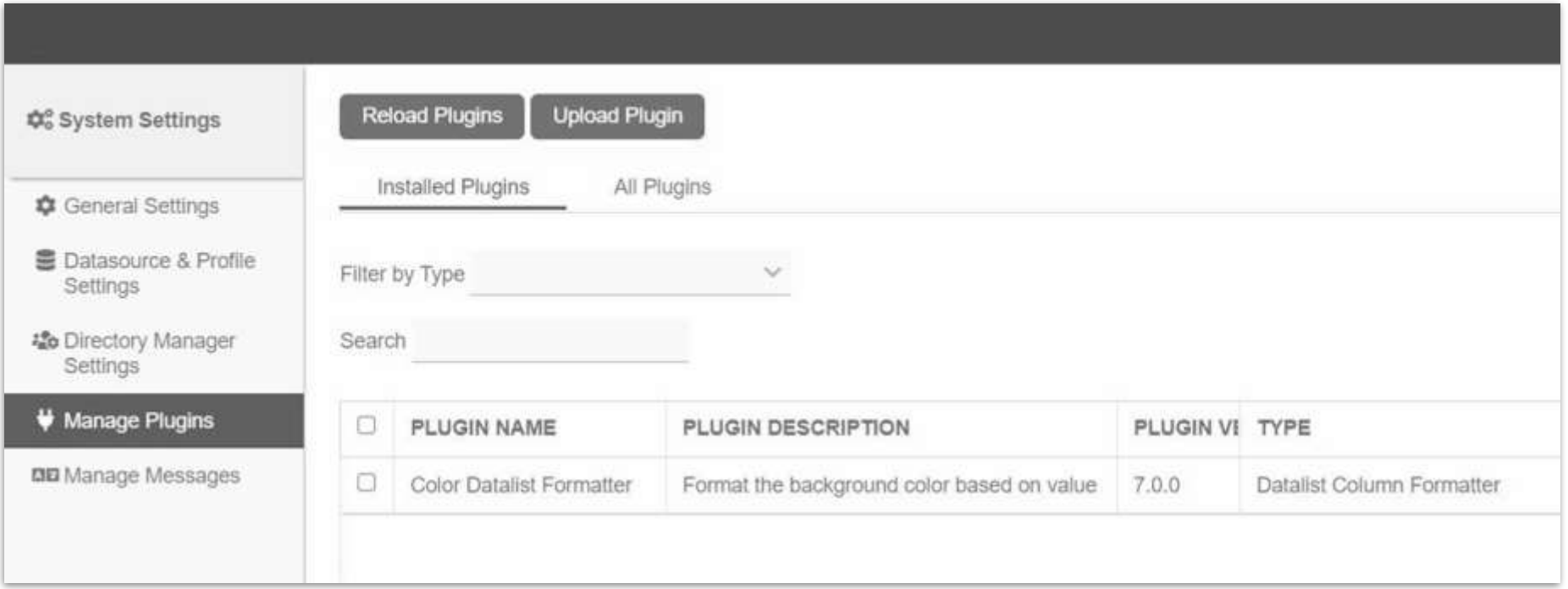


Build and Upload

- 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 RADS.



Build and Upload

- Test out the datalist formatter in actual datalist.
- 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 Datalist Formatter plugin.

Chapter 4

Creating a Custom HTML Userview Menu Plugin

What Are We Going To Build?

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

Creating a Userview 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/net.purwana.rads.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 RADS subproject “rad-plugin-archetype” to create a Maven project for your plugin.
- **For Windows:**
  1. Create a directory to contain your plugins. (E.g. “C:\RADS-projects”)
  2. In Command Prompt, navigate to the newly created directory.
  3. Run “…**\rad-plugin-archetype\create-plugin.bat" net.purwana.rads.tutorial classic_html_menu”**
  4. Key in “7.0-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. “~\RADS-projects”)
  2. In Command Terminal, go to the created directory.
  3. Run “…**\rad-plugin-archetype\create-plugin.bat" net.purwana.rads.tutorial classic_html_menu**
  4. Key in “7.0-SNAPSHOT” for version and “y” to confirm all the information

Creating a Maven Project for Your Plugin

- Windows

```
C:\RADS-projects>C:\jw-community\trunk\rad-plugin-archetype\create-plugin.bat
net.purwana.rads.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/net/purwana/rads/rad-plugin-archetype/7.0-SNAPSHOT/maven-
metadata.xml  Downloading: http://repo1.maven.org/net/purwana/rads/rad-plugin-archetype/7.0-SNAPSHOT/maven-
metadata.xml
```

Creating a Maven Project for Your Plugin

- Windows

```
[INFO] Using property: groupId = net.purwana.rads.tutorial
[INFO] Using property: artifactId = classic_html_menu
Define value for property 'version':  1.0-SNAPSHOT: 7.0-SNAPSHOT
[INFO] Using property: package =
net.purwana.rads.tutorial Confirm properties
configuration:
groupId:
net.purwana.rads.tutorial
artifactId: classic_html_menu
version: 7.0-SNAPSHOT
package:
net.purwana.rads.tutorial
 Y: : Y
[INFO] ____________________________________________________________
[INFO] Using following parameters for creating project from Old (1.x) Archetype: rad-plugin-archetype:7.0-SNAPSHOT [INFO]
____________________________________________________________
[INFO] Parameter: groupId, Value: net.purwana.rads.tutorial
[INFO] Parameter: packageName, Value:
net.purwana.rads.tutorial [INFO] Parameter: package, Value:
net.purwana.rads.tutorial  [INFO] Parameter: artifactId, Value:
classic_html_menu  [INFO] Parameter: version, Value: 7.0-
SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\RADS-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

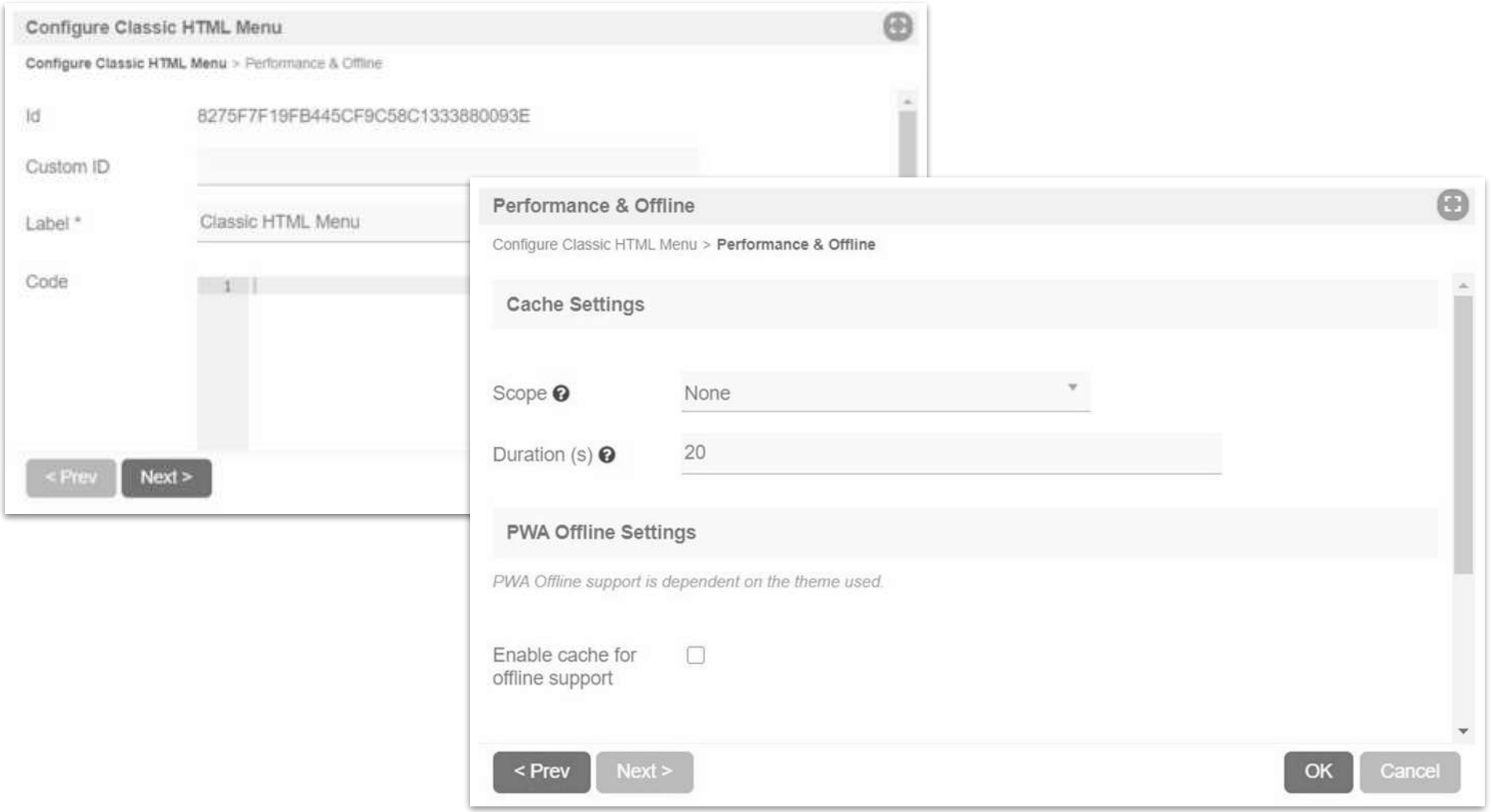
1. Open Activator.java
2. Add the code below to **start** method

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

Implementing the Methods

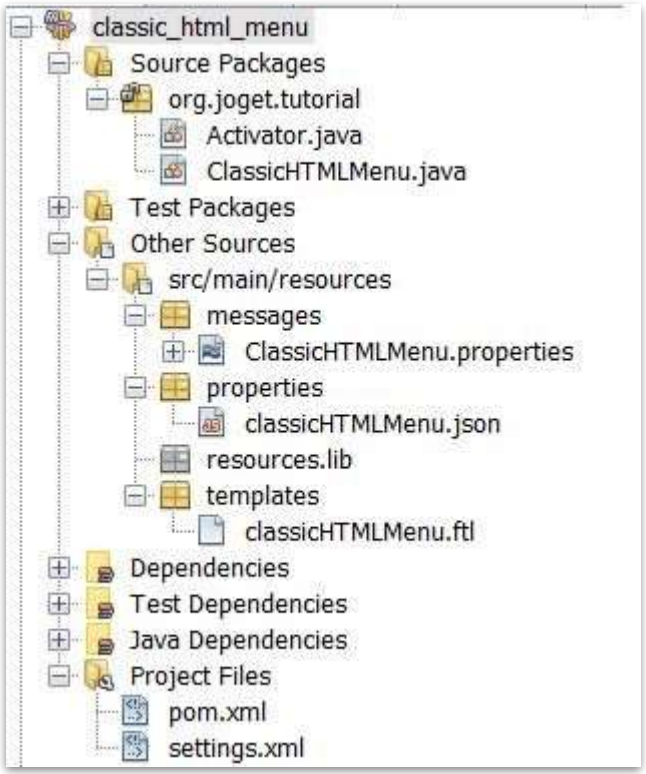
- You may copy the existing implementations of **Userview Menu.**
- Replace where applicable.

Property Options



Creating the Supporting Files

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

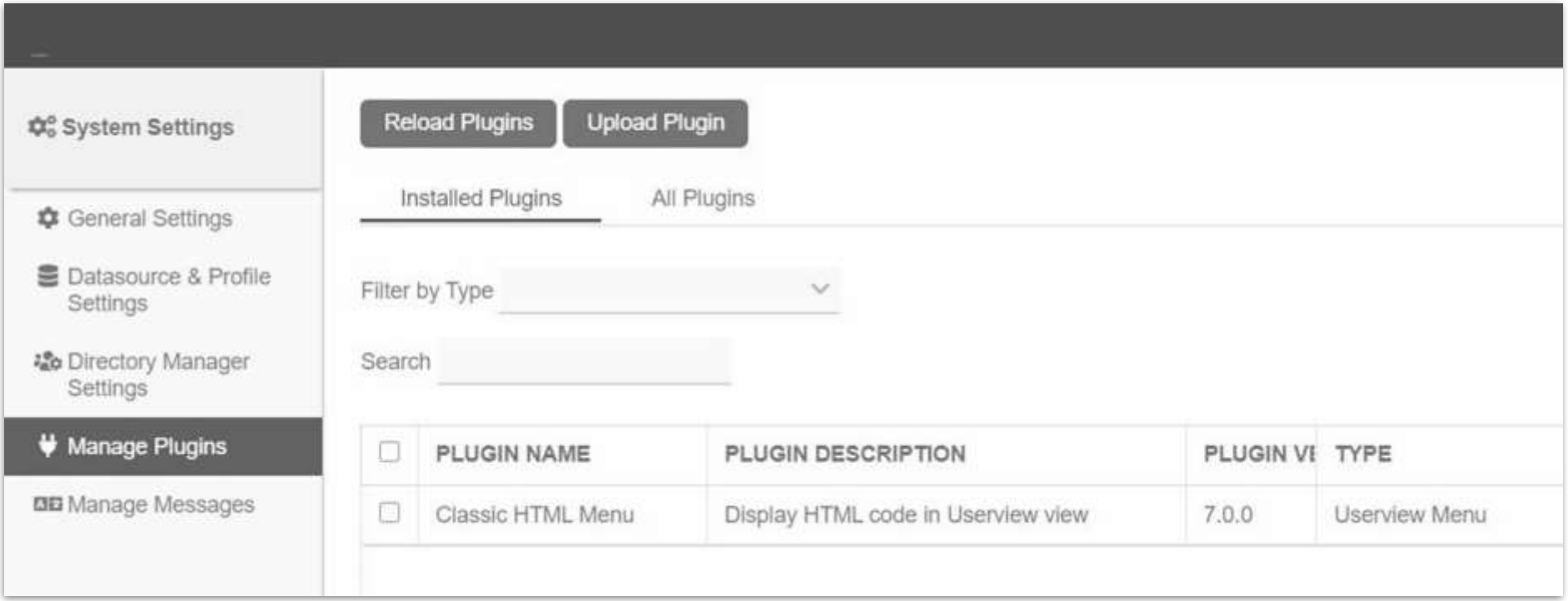


Build And Upload

- 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 RADS.



Build And Upload

- Test out the userview 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 Userview 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://docs.rads.purwana.net/Build+Plugin+Source+Code+using+Docker

  to continue.

- Reference:
  https://hub.docker.com/r/rads1945/docker-maven-rads

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 Datalist Formatter Plugin
4. Creating a Custom html Userview Menu plugin
5. Generate & Build Plugin via Docker

Recommended Further Learning

- Study on how other plugin types are implemented.
  - Check out the RADS Marketplace at https://marketplace.rads.purwana.net/
  - Check out the RADS Knowledge Base – Developer Guide https://docs.rads.purwana.net/Developing+Plugins

Stay Connected With RADS

- rads.purwana.net
- https://github.com/radservice/rad-community