LECTRA FASHION PLM PRODUCT DEVELOPMENT SCREENS CONFIGURATION

Setup Guide

Date of last update: January 2017

Contents

Modifications made to the document since its last publication are highlighted in blue.

CONVENTIONS

Product Development module = Product Developer

Administration and Configuration module = PLM Manager

EXCEPTION

In Lectra Fashion PLM V5R1, some screens are not customizable :

- Composition Set Explorer
- Composition Set Form
- Composition Tab in Product GO

INTRODUCTION

Technology used for screen description has evolved between versions V4R1 and V5R1 of Lectra Fashion PLM.

The screen description files are still XML files but the file structure has changed, as well as the content of the screens.

This document aims to help you identify the XML files to be modified to configure the screens as well as the parts to be duplicated and modified.

The first chapter presents the evolution of the configuration structure.

The second chapter presents the definitions and concepts necessary for reading and understanding the XML files. Some definitions related to the ExtJS technology are included.

The third part, the configuration, deals with adding fields and blocks to the **General Objectives** screens for **Products**, in the **Products** Explorer with the new **Search** panel, and the **Collection Plan** screens.

When you are familiar with this configuration, you can use the concept of inheritance in order to improve the configuration efficiency by reducing block duplication.

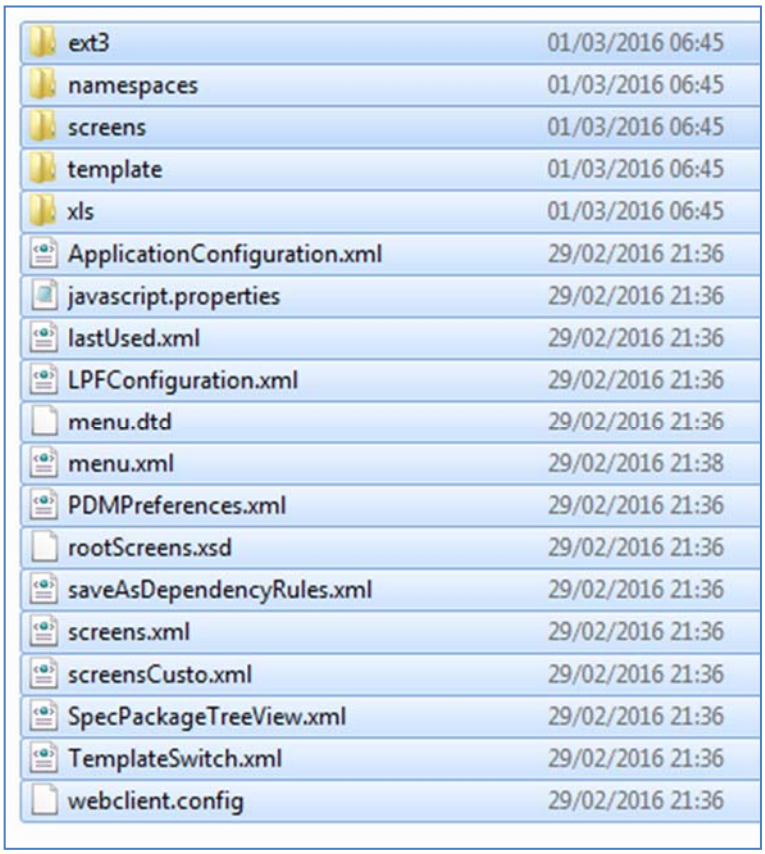
The fourth part will help you reorganize the screen content by modifying the default **layout**. The **layout** is a definition of data organization on a screen: by column, table, etc.

In the appendices, you will find reminders about data creation in the Administration and Configuration module, as well as the technical documentation on the graphical elements used with part of the ExtJS and LPFExt frameworks.

1. NEW CONFIGURATION STRUCTURE

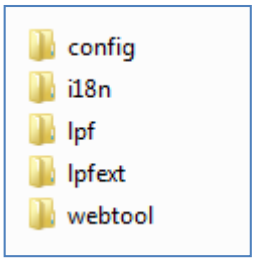
The LectraPLMParam directory, used in production mode for the customization of different files (xml, html template, menu, tree) has been restructured to introduce a 'custom' directory which contains all the customized files.

1.1 Previous structure



ext3	01/03/2016 06:45
namespaces	01/03/2016 06:45
screens	01/03/2016 06:45
template	01/03/2016 06:45
xls	01/03/2016 06:45
ApplicationConfiguration.xml	29/02/2016 21:36
javascript.properties	29/02/2016 21:36
lastUsed.xml	29/02/2016 21:36
LPFConfiguration.xml	29/02/2016 21:36
menu.dtd	29/02/2016 21:36
menu.xml	29/02/2016 21:38
PDMPreferences.xml	29/02/2016 21:36
rootScreens.xsd	29/02/2016 21:36
saveAsDependencyRules.xml	29/02/2016 21:36
screens.xml	29/02/2016 21:36
screensCusto.xml	29/02/2016 21:36
SpecPackageTreeView.xml	29/02/2016 21:36
TemplateSwitch.xml	29/02/2016 21:36
webclient.config	29/02/2016 21:36

1.2 New structure



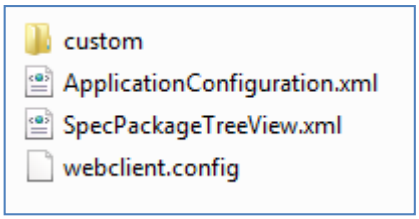
- config
- i18n
- lpf
- lpfext
- webtool

5 directories have been created. Each one contains a type of file (according to the technologies or usage).

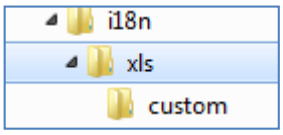
Certain files, which are no longer customized, or have not been customized for a certain period of time, have also been deleted.

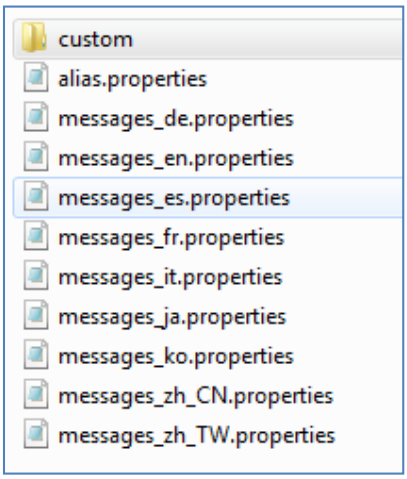
Each of these directories contains a 'custom' directory and this is the only place custom files must be added.

1.3 Config Directory

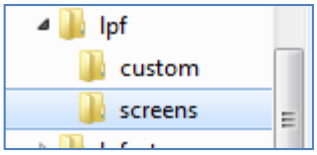


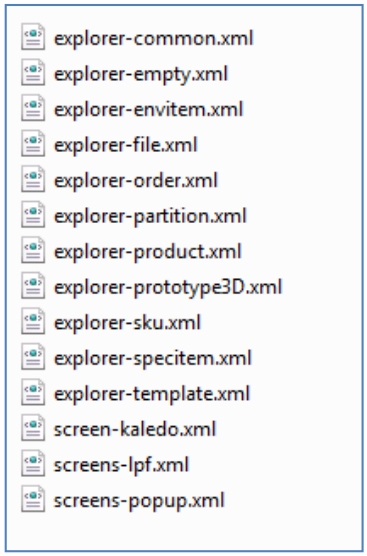
1.4 i18n Directory





1.5 lpf Directory





Inheritance does not exist in this technology, the file to be customized must be completely recopied in the custom directory, then modified according to the required configuration.

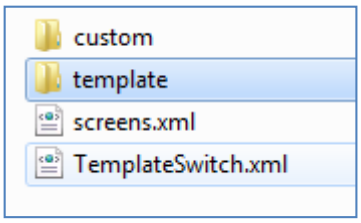
1.6 Ipfext Directory

It contains the standard structure for namespace files (xml).

There are no main changes in this technology.

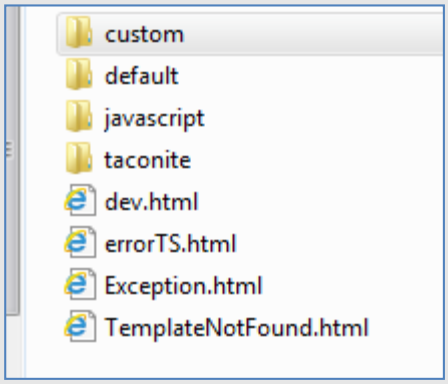
1.7 Webtool Directory

It contains the two Screens.xml and TemplateSwitch.xml files and the template directory.



The template directory contains all the Product Developer html templates and has its own template directory.

This is the only directory specified for customizing or overwriting existing templates:



All the customized html templates for customers must be contained **on the flat** in the same 'custom' directory regardless of the initial hierarchy in the standard. The Product Developer recovers the html templates in the 'custom' directory (if the they are present) as a priority.

1.8 Operation

1.8.1 Previous operation

On the PLM installation and with the startup of JBoss, the LectraPLMParam directory is created if it doesn't exist and its contents are copied from the WebClient WAR.

Each time JBoss is started up, the existence of LectraPLMParam is checked. If it does not exist, the resources are recopied.

1.8.2 New operation

Each time JBoss is started up (after a fresh install or just a Stop/Start), the contents of the LectraPLMParam will be overwritten with new resources contained in the WebClient WAR with the exception of custom directories which will not be modified.

All customization made in the original files will be lost!

2. DEFINITION AND CONCEPTS

2.1 XML files

Screen definition can be done in several XML description files. Unlike previous versions, from V3R2 on, an XML file is created for a screen and even for a part of a screen.

When you look at the source code of a Product Development module screen you can see that the "**screenName**" variable may be entered in the header of the displayed source code: its value indicates the screen name.

2.1.1 The screens.xml file

The input file to be taken into account to know which XML file is to be modified to customize a screen is: **PLM-Fashion\PDM\LectraPLMParam\webtool\screens.xml**. This file references all the screens defined with the previous technology as well as the new one.

Example:

For the **ProductGO** screen, we have the description of the new **General Objectives** tab, as well as its **Description** part and the description of the other tabs that are not yet in LPFExt.

```
<!-- Generic screen for Products General Objectives. -->
<screen name="ProductGO" displayCustoms="false">
      <icons>
            <icon name="defaultImageField"/>
      </icons>

<!-- Tabs -->
<tabs name="pgoTabs" displayed="false" placement="top"
prefKey="ProductGO.objectives.selectedTab"
tabManager="com.lectra.pdm.webapp.tabs.ProductTabManager"
tabSelectionStrategy="com.lectra.pdm.webtool.config.tab.ProductGOTabStrategy">

<!-- Objectives tab -->
<tab name="descriptionTab" i18nKey="ProductGO.description.Title"
autoScroll="true" autoHeight="false"
lpfExtPath="http://lectra.com/pdm/productgo#description" />

<!-- Cost and Margin tab  -->
<tab name="costTab" i18nKey="ProductGO.description.cost" autoLoad="true"
autoScroll="true" autoHeight="false">
      <block name="detailcost" displayed="true" columns="4" template="Content"
extendedProfileTag="PrivateArea" titled="true">
```

```
      <block name="detailcostCol2" titled="true"
template="screens\Block_3_hr.html">
         <field name="targetCost.defaultPrice" editable="true"/>
         <field name="targetCost.defaultCollectionPrice" editable="true"/>
         <field name="targetCost.approved" editable="true"/>
      </block>
      …/…
```

2.1.2 lpfExtPath

This information is needed in the **PLM-Fashion\PDM\LectraPLMParam\webtool\screens.xml** file for each LPFExt screen and shows the path of the corresponding XML file. All the XML files are in the **Namespaces** folder.

Example:

```
<!-- Objectives tab -->
<tab name="descriptionTab" i18nKey="ProductGO.description.Title"
autoScroll="true" autoHeight="false"
lpfExtPath="http://lectra.com/pdm/productgo#description" />
```

The description of the content of the **descriptionTab** tab is in the file whose **prefix** is **productgo**, and its name will be **Lectra.PDM.ProductGO**.

The tab content will be described in the **description** block inside this XML file.

This description block is a panel type container. Please refer to 2.1.4 - **Screen description: technical approach**.

2.1.3 Namespaces

An XML description file is structured as follows:

```
<namespace uri="http://lectra.com/pdm/productgo" prefix="productgo"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- comment -->
<element1 name="elt1Name"> …  </element1>
<!-- comment -->
<element2 name="elt2Name"> …  </element2>

<!-- Private field(s) : do not edit -->
<private>     …        </private>
<!-- End of private field(s)        -->

</namespace>
```

Example: Lectra.PDM.ProductGO.xml

The description file must be declared by a **namespace** tag.

This tag **must** contain at least the following attributes:

- **uri**: identifier of the description file. It has a changing part that depends on the file's **prefix**.
- **prefix**: identifier that describes the screen. It is quoted in the **uri** and in the **screens.xml** file. It only identifies the file.
- **xmlns:xsi** with the corresponding value: **« http://www.w3.org/2001/XMLSchema-instance »**
- **xmlns:xs** with the corresponding value: **« http://www.w3.org/2001/XMLSchema »**

The **uri** attribute that contains the identifier of the XML file may be found also in the **namespace.properties** file.

This file contains the identifiers' definition according to all description files.

In each description file there are comments starting with "**<!--**" and ending with "**-->**".

There are also private zones that must not be modified by customers. They start with the following comment:

```
<!-- Private field(s) : do not edit -->
```

and end with:

```
<!-- End of private field(s)        -->
```

2.1.4 Screen description: technical approach

Screen description starts with the definition of the XML file header with the **namespace** presented in 2.1.3 Namespaces.

The screen is then defined with the graphic elements that compose it and that are described in the base xml file (in the **namespaces** folder) and in the xml file of the same name that is in the **custom** sub-folder.

These elements are graphic elements of the ExtJS library and of the LPFExt framework. The most used is `<panel>`, always defined in between `< >`.

Definitions of the «panel»:

`<panel>` is a panel's opening tag

`</panel>` is a panel's closing tag

All the describing characteristics of the « **Panel** » graphic element are defined in the opening tag: layout, colors, width, scroll…

All the other graphic elements of this panel will be defined between the opening tag and the closing tag. A panel is generally composed of other `<panel>` elements, that are themselves composed of `<combobox>`, `<nodecombo>`, `<textarea>`, `<textfield>`, `<richText>`, `<image>`, etc…

2.2 Inheritance

With the new LPFExt framework, access to screens is quicker. Through several XML description files, it implements a system of inheritance for better reading of the screens and a standardization of generic elements.

Unlike the previous description technology, a single key word is now used to define the inheritance: «**inherits**». But now the notion of changing inherited elements is lost.

There are two types of inheritance: internal inheritance and external inheritance.

2.2.1 Internal inheritance

The most commonly used inheritance mode, especially to decline the screens description for the different product categories.

Internal inheritance uses an element declared inside the screen description file.

```
<parent name="parentNameExample">
      <!-- child element -->
      <child inherits="childNameExample">
</parent>

<child name="childNameExample">
      <!-- Title of child -->
      <title>Titre</title>
</child>
```

The <parent> tag takes the <child> tag (defined below) as element. In this example, the <child name= « childNameExample »> tag defines the content of the <child> element.

This is equivalent to writing:

```
<parent name="parentNameExample">
      <!-- child element -->
      <child name="childNameExample">
            <!-- Title of child -->
            <title>Titre</title>
      </child>
</parent>
```

For a given xml description file, it is strongly recommended you add your own inheritances in the dedicated xml file that is in the **namespaces/custom** sub-folder and that has the same name.

Example:

- The **namespaces/Lectra.PDM.ProductGO.xml** description file contains the definition of a description panel that inherits several possible panels:
  - <panel name="description_wrap" titleI18n="description" region="north" border="false" height="300" split="true" collapsible="true" stateId="panel_${#type}" layout="fit" maximizableChild="true">
  - <panel name="description" inherits="description_${#type},description_${#topCategoryName},description" />
  - <plugin xsi:type="xmlmap" ptype="panelCollapsedTitle" />
  - </panel>
- In the **namespaces/custom/Lectra.PDM.ProductGO.xml** file, you can define your own description for a sub-category custom under the following template:

With « MyCustomCategory » defined in the Administration and Configuration module

```
<panel name="description_MyCustomCategory">
     <!-- Add your custom panels here -->
</panel>
```

This sub-category being a type, the *description_MyCustomCategory* panel corresponds to the inheritance defined by *description_${#type}*, that is positioned first.

2.2.2 External inheritance

External inheritance uses an element declared in a description file (see example below: **Common.xml**) and then in using it in another screen description file (see example below: **Ecran.xml**).

```
<namespace uri="http://lectra.com/pdm/example/common" prefix="common"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<child name="childNameExample">
      <!-- Title of child -->
      <title>Titre</title>
</child>

</namespace>
```

Lectra.PDM.Example.Common.xml

```
<namespace uri="http://lectra.com/pdm/example/ecran" prefix="ecran"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<parent name="parentNameExample">
  <!-- child element -->
 <child inherits="http://lectra.com/pdm/example/common#childNameExample">
</parent>

</namespace>
```

Lectra.PDM.Example.Ecran.xml

The <parent> tag takes the <child> tag as an element that indicates the element title but which is defined in another description file.

The value in the "**inherits**" attribute is constructed as follows:

${URI** of the description file that contains the element**}#${NAME of the element**}**

Example:

For example, this inheritance is used for toolbars that are used in several screens.

Namespaces/Lectra.PDM.ChangeTracking.xml uses a toolbar that inherits the paging Toolbar defined in the **namespaces/Lectra.PDM.Common.xml** file.

```
<paging name="pagingToolbar" displayInfo="true" pageSize="25">
  <tbseparator />
  <plugin xsi:type="pagingtoolbarresizer">
       <option>25</option>
       <option>50</option>
       <option>100</option>
  </plugin>
</paging>
```

Lectra.PDM.Common.xml

```
<bbar xsi:type="paging"
inherits="http://lectra.com/pdm/common#pagingToolbar" pageSize="7" />
```

Lectra.PDM.ChangeTracking.xml

2.2.3 Creating new files

It is possible to create your own screens by writing the appropriate XML file. It is recommended to identify these new files by creating them in a dedicated folder: for example in a sub-folder of **namespaces** called **MyCompany**.

- Create the xml file in accordance with the structure shown in 2.1.3 Namespaces.
- Reference the new file in the **namespaces.properties** file.

```
#Lectra namespaces
${DIRECTORY}namespaces/« MyCompany »/Lectra.PDM.Common =
http://lectra.com/pdm/common
```

- It is recommended to reference the new file in the **screensCusto.xml** file that will contain the screens.xml declarations, duplicated and modified, like what is done for the **ProductGO** screen for example:

```
<screen name="ProductGO" displayCustoms="false">
```

```
<icons>
   <icon name="defaultImageField"/>
</icons>

<!-- Tabs -->
  <tabs name="pgoTabs" displayed="false" placement="top"
     prefKey="ProductGO.objectives.selectedTab"
     tabManager="com.lectra.pdm.webapp.tabs.ProductTabManager"
     tabSelectionStrategy="com.lectra.pdm.webtool.config.tab.ProductGOTabSt
     rategy">
      <tab name="descriptionTab" i18nKey="ProductGO.description.Title"
     lpfExtPath="http://lectra.com/pdm/productgo#description_Style" />
  </tabs>
</screen>
```

2.2.4 General properties

- The inheritance is defined by the "**inherits**" attribute and it takes the value of the **name** attribute of the element to be integrated.
- The tag type (**<panel>** for example) calling the inheritance (declaring the "**inherits**" attribute) must be the **same** as the tag that is declared.

2.3 What you need to know on the screens description structure

2.3.1 Searching for a screen starts with reading the screens.xml file

The screen being searched for is identified:

`<screen>` tag followed by a `name` field whose value corresponds to the screen `prefix.`

For this tag, we identify the tab we are interested in, `<tab>` tag, as well as the name of the corresponding xml file and the name of the corresponding panel in the associated **lpfExtPath**.

Example:

```
<!-- Generic screen for Products General Objectives. -->
<screen name="ProductGO" displayCustoms="false">
     <icons>
      <icon name="defaultImageField"/>
      </icons>

      <!-- Tabs -->
      <tabs name="pgoTabs" displayed="false" placement="top"
           prefKey="ProductGO.objectives.selectedTab"
           tabManager="com.lectra.pdm.webapp.tabs.ProductTabManager"
           tabSelectionStrategy="com.lectra.pdm.webtool.config.tab.Product
           GOTabStrategy">
       <!-- Objectives tab -->
       < !-- NOUVELLE TECHNO -->
       <tab name="descriptionTab" i18nKey="ProductGO.description.Title"
           autoScroll="true" autoHeight="false"
           lpfExtPath="http://lectra.com/pdm/productgo#description" />

       <!-- Cost and Margin tab  -->
       < !-- ANCIENNE TECHNO -->
```

```
        <tab name="costTab" i18nKey="ProductGO.description.cost"
            autoLoad="true"  autoScroll="true" autoHeight="false">
            <block name="detailcost" displayed="true" columns="4"
            template="Content" extendedProfileTag="PrivateArea"
            titled="true">
            <block name="detailcostCol2" titled="true"
            template="screens\Block_3_hr.html">


        ../..
```

In the new technology, the path to the corresponding xml file is expressed as follows:

`lpfExtPath="http://lectra.com/pdm/productgo#description"`

`lpfExtPath="http://lectra.com/pdm/ « prefix du fichier xml »# « nom du panel » "`

In this example, the first tab (**Objectives**) corresponds to the **Lectra.PDM.ProducGO** file that has the same prefix as **ProductGO** and as the **description** panel.

Blocks that are described with the previous technology in the **screens.xml** file are now defined by **panels** in the dedicated xml file.

2.3.2 Reading the base xml file

In the given example, the declaration of the **description** panel is searched for in the **Lectra.PDM.ProductGO.xml** file.

```
<panel name="description"
inherits="description_${#type},description_${#topCategoryName},description"
/>
```

This means that the *description* `<panel>`, if it is defined, inherits,

- In first position of <panel name ="*description_${#type}*" …>;
  The *${#type}* defines the current type in the context, to be chosen amongst the possible types: Style, MyCustomStyle, Fabrics, Denim, Fur, Bags, Order, SKU, etc…
- In second place, if the first one is not defined), of <panel name ="*description_${#topCategoryName}*" …>
  The *${#topCategoryName}* defines the current Top Category of the context: Styles, Fabrics, Trims or PackagingLabel.
- In third place, if the other two are not defined, of <panel name ="*description*" …> that is defined by default in the base xml file.

The alternatives must be sought in the document or in the xml file of the same name in the **custom** sub-folder, to see which ones are defined.

2.3.3 Modifying the screen description

In the following paragraphs and through examples we will see how to modify, add `<panel>`, add fields of type `<nodecombo>`, `<textarea>`, `<textfield>`, etc…

- One way to do this is to rewrite the relevant paragraphs (`<panel>`) with the modified content. It is the easiest method to start with.
- A second way is to use the inheritance notion seen in 2.2 - Inheritance. The inheritance is implemented in 6 -.Advanced configuration for adding fields and panels.

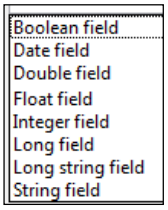
3. CONFIGURATION BY ADDING FIELDS TO « GENERAL OBJECTIVES »

The **General Objectives** screen of Products (**Styles**, **Fabrics**, **Trims**, and **Packaging Labels**) may be configured by modifying the interface description in the **namespaces/Lectra.PDM.ProductGO.xml** file.

3.1 Adding simple fields

3.1.1 Definition

Simple fields are field types that correspond to the following data in the Administration and Configuration module:



All of these steps must be followed:

3.1.2 Step 1: Prerequisite Administration and Configuration module

All the fields you want to add to your screen must be defined in the Administration and Configuration module

You may use the many available fields or create new ones. In this case, please refer to the corresponding Online Help.

> You can also refer to the reminder in Appendix H - LPFExt Directory location Change
>
> In PLM V5R1, the location of the LPFExt files is under the lpf/ext3/namespaces directory.
>
> Reminder of the Administration and Configuration Application

3.1.3 Step 2: Configuration in the dedicated XML file

To add fields to the **General Objectives** screen of **Products**, it is necessary to enter the file that is in the "**custom**" sub-folder **namespaces/custom/Lectra.PDM.ProductGO.xml** (and not the **namespaces/Lectra.PDM.ProductGO.xml** file)

If it does not exist, it is necessary to create it in the custom folder and to structure it as follows:

```
<namespace uri="http://lectra.com/pdm/productgo/custom"
           prefix="productgo"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!--       AJOUTER CONFIGURATION ICI -->
<!-- ____________________________  -->

</namespace>
```

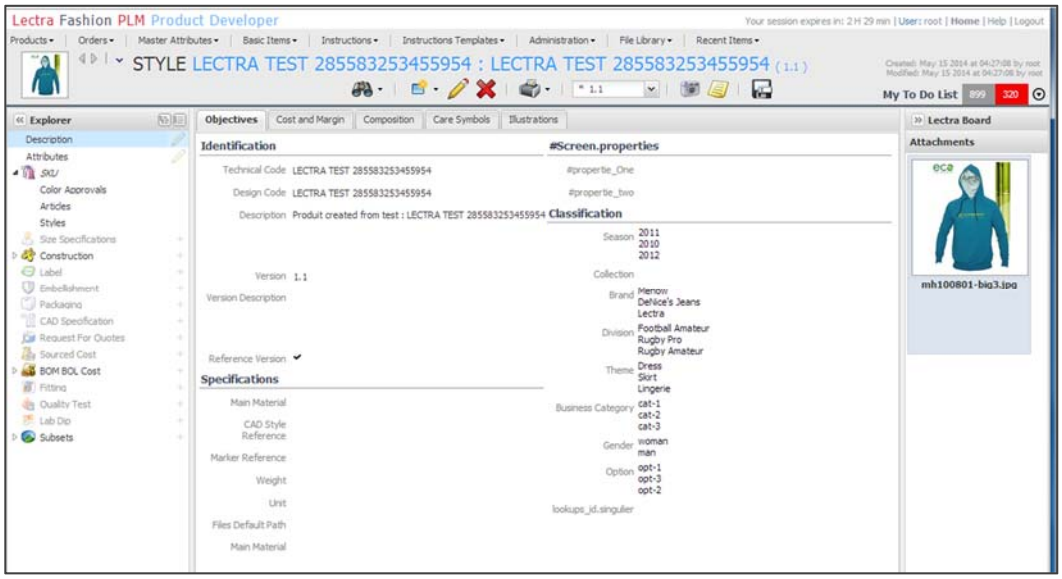
3.1.4 Step 3: Identification of the part to be duplicated and Modification

For each modification:

- Identify the relevant part in the **/namespaces/LectraPDM.ProductGO.xml** file (base xml)
- Copy the <panel> that contains this part in the **/namespaces/custom/LectraPDM.ProductGO.xml** file (custom xml)
- Make the change in this duplicated part.
- Update your screen display: use **F5** to refresh.

3.1.4.1 Identifying the part to be modified in the base XML

- For the STYLE category:



Look for the Style's description Panel = content of the **General Objectives** tab.

In the **/namespaces/LectraPDM.ProductGO.xml** file, look for the part that describes the **GeneralObjectives** for the Style category.

```
<panel name="description_Style" layout="column" autoScroll="true" >
```

This panel is made up of several panels…

Look for the Panel to be modified.

- To modify the «**Identification**» `<panel>`, look for:

```
<panel name="identification" titleI18n="Screen.identification"
inherits="looklikefieldset" >
      …/…
</panel>
```

- To modify the «**Classification**» `<panel>`, look for:

```
<panel name="collection" titleI18n="Screen.collection"
columnWidth="0.5" inherits="looklikefieldset" >
      …/…
</panel>
```

- To modify the «**Specifications**» `<panel>`, look for:

```
<panel  name="detail" titleI18n="Screen.specification"
inherits="looklikefieldset">
      …/…
</panel>
```

- To modify the «**Validation Table**» `<panel>`, look for :

```
<panel name="validationTable"… >
      …/…
</panel>
```

If you want to hide the generated fields in the validation table panel, change the property headerOnly="false" of the panel to headerOnly="true".

If you want to display a validation field in the ProductGo you have to add a specific field according to the type of field you want to display. The name of the field is the concatenation of the name of the table and the name of the field (Ex : table name : 'VT', field name : 'Weight' => field name : 'VT_Weight').

For a number field add the following line :

```
<numberfield isValidationField="true" title18n="Weight" name="Table_Weight" >
```

For a text field add the following line :

```
<textfield isValidationField="true" title18n="Name" name="Table_Name" >
```

For a date field add the following line :

If you want the date to change according to the client timezone:

```
<datefield isValidationField="true" title18n="Birth" name="Table_Birth" >
```

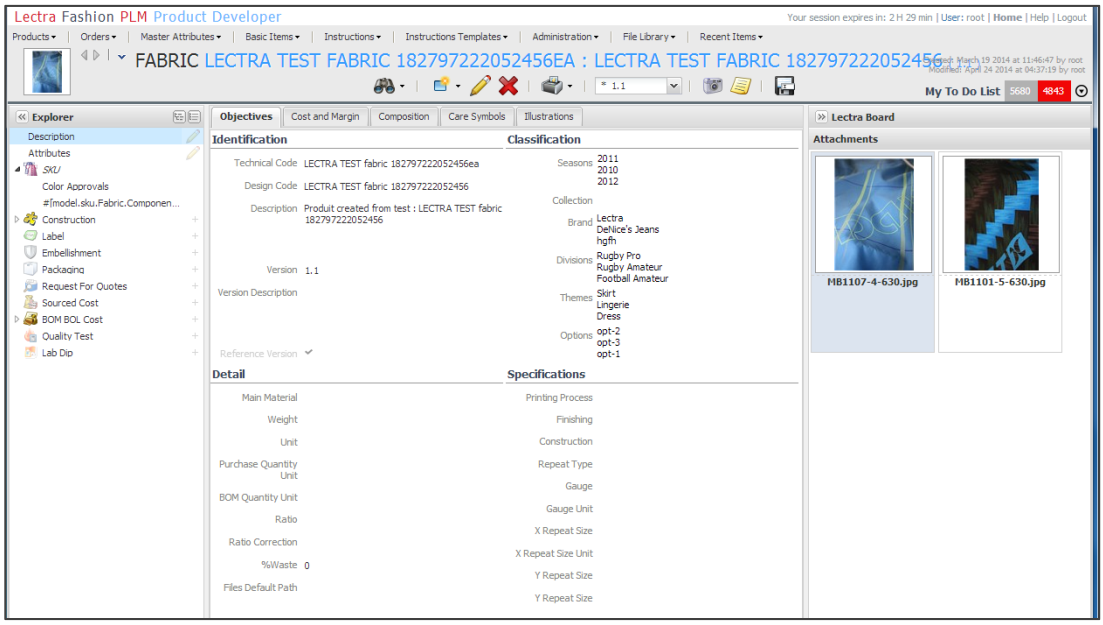
If you don't want the date to change according to the client timezone:

```
<datefield isValidationField="true" title18n="Birth" name="Table_Birth" useServerTimezone="true">
```

For a checkbox field add the following line :

```
<xcheckbox isValidationField="true" titleI18n="International" name="Table_International" readOnly="true" >
```

- For the FABRIC, TRIMS and PACKAGINGLABEL categories:



Look for the Fabric's description Panel.

In the **/namespaces/LectraPDM.ProductGO.xml** file, look for the part that describes the **GeneralObjectives** for the Fabric category

```
<panel name="description_default" layout="column" autoScroll="true" >
```

Follow the panels that are linked by inheritance until you reach the panel that corresponds to the part to modify.

Example: the «**description**» `<panel>` above contains the «**north**» panel that inherits from the «**description_north_default**» panel that contains the «**identification**» <panel> and the «**classification**» <panel>.

Please refer to 2.2 - Inheritance.

Look for the Panel to be modified

– To modify the «**Identification**» `<panel>` of the Fabric, look for :

```
<panel name="identification_default"
titleI18n="Screen.identification" inherits="looklikefieldset">
      …/…
</panel>
```

– To modify the «**Classification**» <panel>, look for:

```
<panel inherits="looklikefieldset" name="classification_default"
titleI18n="Screen.collection" >
      …/…
</panel>
```

– To modify the «**Details**» <panel>, look for:

```
– <panel  name="detail_FabricTrim" inherits="detail_default">
–     …/…
– </panel>
```

– To modify the «**Specifications**» <panel>, look for:

```
– <panel name="specification_Fabric"  inherits="specification_default">
–     …/…
– </panel>
```

3.1.4.2 Duplicating the part to be modified in the XML custom

Copy the **panel** description that contains the **panel** to be modified in the **namespaces/custom/Lectra.PDM.ProductGO.xml** file.

Example: To modify the Style identification panel:

```
<panel name="identification" titleI18n="Screen.identification" inherits="looklikefieldset" >
```

Copy all the Description Style Panel:

```
<panel  name="description_Style" layout="column" autoScroll="true" >
…/…
</panel >
```

3.1.4.3 Adding the xml description of a text field

Once the relevant part is identified and duplicated (if not done already) in the custom XML, add the following line in the **panel** to the desired location:

```
<textfield name="MyCustomStringField" fieldLabelI18n = "MyCustomStringField" />
```

To add other types of simple fields, please refer to Appendix J - Adding other types of simple fields in the xml.

3.1.5 Step 4: Referencing the translation of the added field

If you have added a custom field with the **MyCustomStringField** name in the Administration and Configuration module, the Product Development module screen will display:

If the field name appears with the # character in front, it means that the field translation does not exist.

Translate this text by modifying the **PLM-Fashion\PDM\LectraPLMParam\messages_en.properties** file by adding at the end of the file:

MyCustomStringField = My String Field.

The result will be:

In the Internet Explorer, execute the following command:

http:// « Server name »/pdm/admin.SessionMonitor.reloadConfig.wbx

In the **Internationalization** menu of the Administration and Configuration module, click on **Empty Cache**.

3.2 Adding a list

3.2.1 Definition

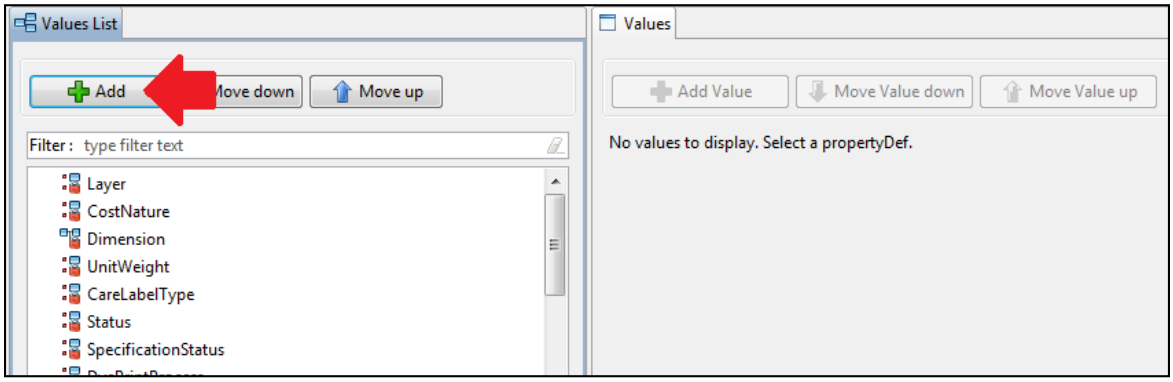
Called **Value Lists** in the application, they are associated with categories by adding «**CustomRole**» as a «**PickList**» «**Target**» for each value list.

In the Product Development module, they are added to the screen via the `<nodecombo />` component.

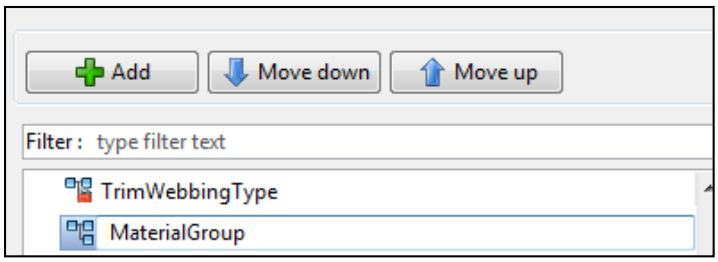
3.2.2 Step 1: Definition of the list in the administration and configuration module and association to the Style category

3.2.2.1 Creating

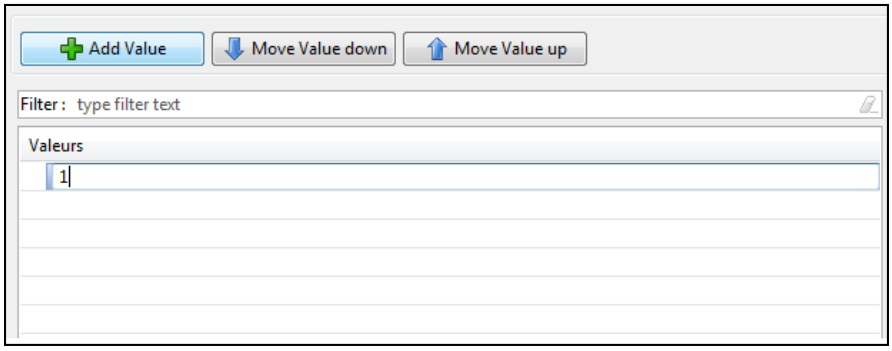
Click **Value Lists** in **Data** menu of the **Administration**.



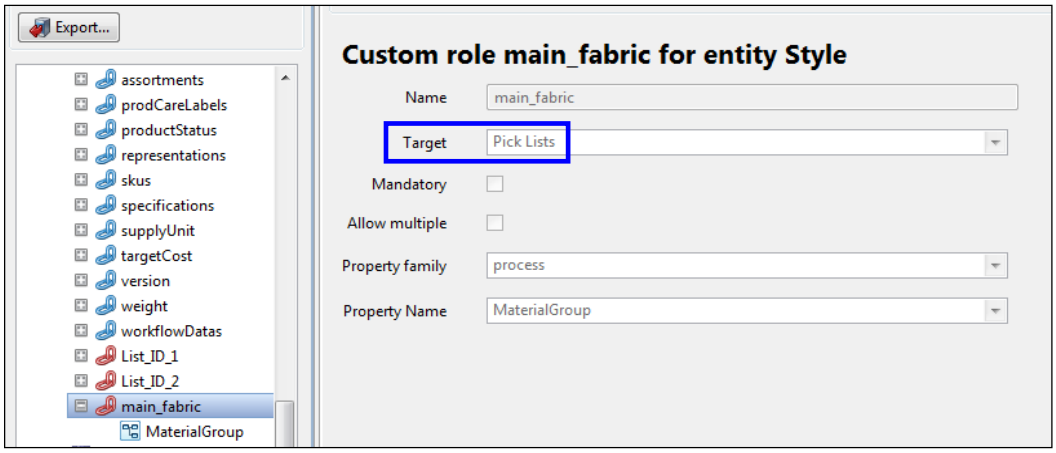
Click on **Add** and enter the name of new Value List.



Select the new list and add new values in right screen:



3.2.2.2 Associating with the CustomRole



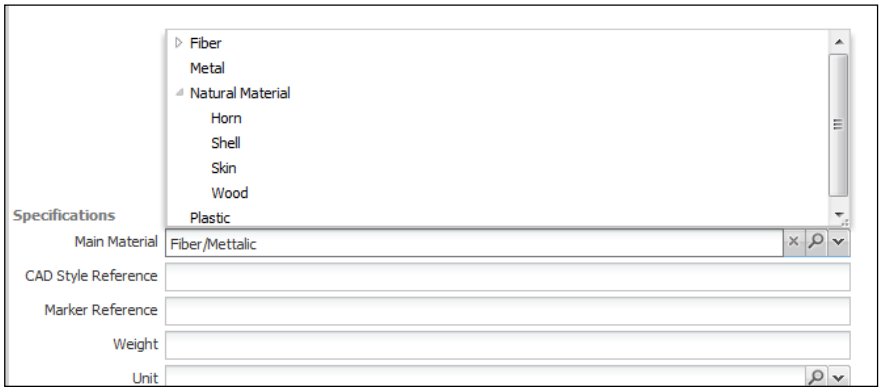
3.2.3 Step 2: Description in the custom/Lectra.PDM.ProductGO.xml file

```
<panel name="description_Style"
    <panel name="detail":
        <nodecombo   fieldLabelI18n="Style.mainFabric"
                     hiddenName="mainFabric" />
```

3.2.4 Step 3: Referencing the translation of the added field

In the **messages_en.properties** file.

3.2.5 Result: Display in the Product Development module



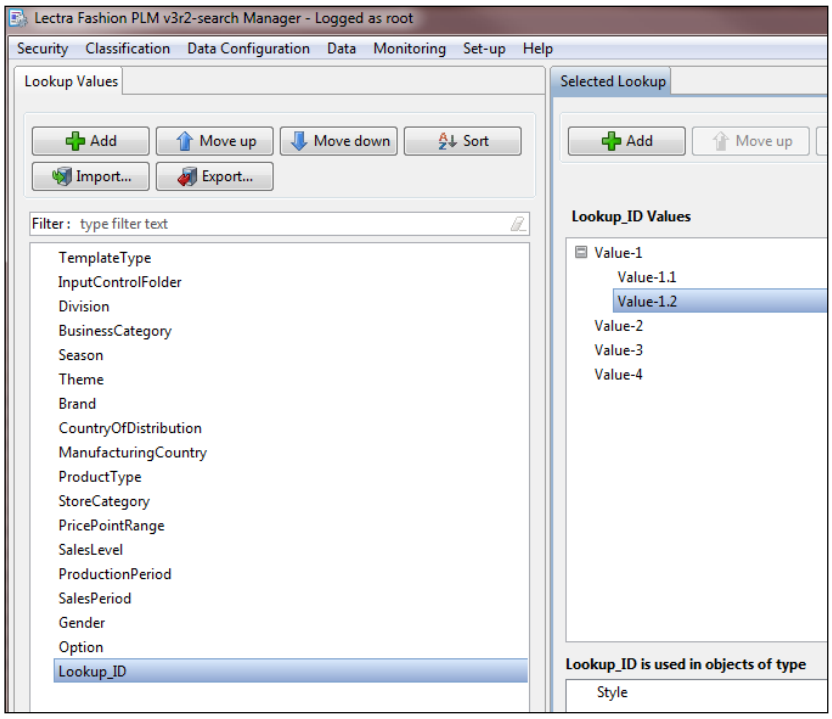
3.3 Adding classification

3.3.1 Definition

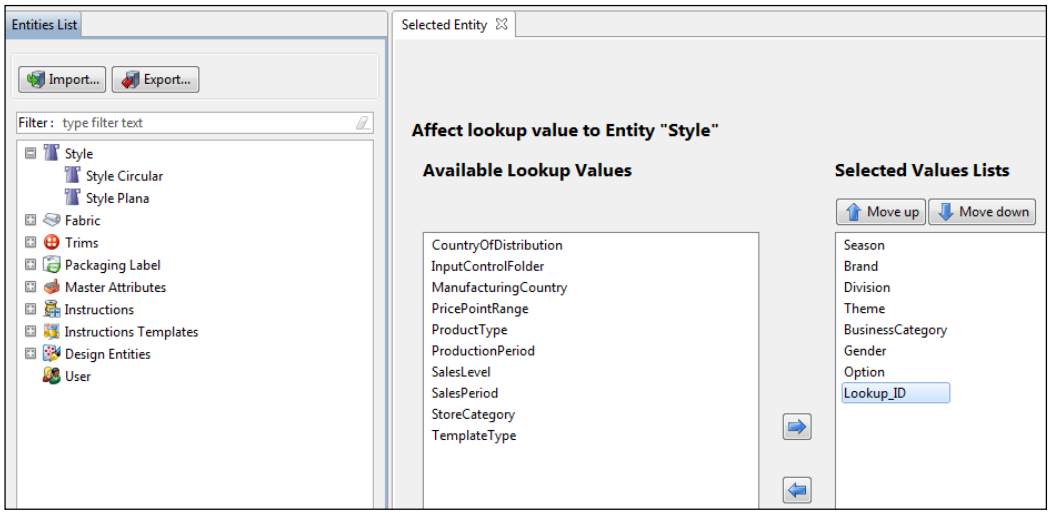
Called **Lookup Values** in the **Classification** menu of the Administration and Configuration module, they can be created by **Define Lookup Values** in a hierarchical way and they are associated with one or several **Affect Lookup Values** categories.

3.3.2 Step 1: Definition in the Administration and Configuration module

3.3.2.1 Creating



3.3.2.2 Associating to the Style category



3.3.3 Step 2: Description in the Lectra.PDM.ProductGO.xml file

In the Product Development module, this new list should appear automatically in the **Classification** part of the relevant Products, on the **GeneralObjectives** page.

3.3.3.1 Automatic description of the classification

- Make sure the **Collection panel** is present in the base XML, the one in the **namespaces** folder.

```
<panel name="collection" titleI18n="Screen.collection" columnWidth="0.5"
          inherits="looklikefieldset">
<nodecombo fieldLabelI18n="${#axeRoleName}.singulier"
          hiddenName="${#classifRoleName}" multiSelect="true"
          propertyName="${#propertyName}" genProductClassif="prodgo" />

<!-- Private field(s) : do not edit -->
<customSerializer type="com.lectra.pdm.lpf.ext.serializers.
                       ProductClassifGeneratorCustomSerializer" />
<!-- End of private field(s)          -->
</panel>
```

- The **Custom Serializer**

  The **Custom Serializer** is used to generate a single field or a group of fields. It is defined in the XML files through the `<customSerializer …/>` tag which is generally located in a private zone:

```
<!-- Private field(s) : do not edit -->
      <customSerializer ...
<!-- End of private field(s)       -->
```

  This definition must not be modified. This will cause a bad page generation.

3.3.3.2 Manual description of the classification

- In a `<panel>` similar to the one above, it is possible to define one's list of `<nodecombo>` by referring to an HVL list defined in the Administration and Configuration module, without doing the **Custom Serializer** process.

Example:

- With a **Pick List** defined in the Administration and Configuration module:
  - <nodecombo fieldLabelI18n="*ApprovalStatus*" hiddenName="*ApprovalStatus*" propertyName="*ApprovalStatus*" multiSelect="*false*" chkVisible="*false*"/>
- With a **Classification Lookup Value** defined in the Administration and Configuration module: «**Brand**» for example
  - <nodecombo fieldLabelI18n="*brands_isa*" hiddenName="*brandsClassif*" propertyFamily="*process*" propertyName="*Brand*" multiSelect="*true*" />

The `hiddenName` should be composed as follows: *"Lookup Value Name"sClassif.*

- If the **Pick List** or the **Classification Lookup Value** is inserted as search criteria in the **Search** panel of the Products explorer, « `.values` » has to be referenced in the « `hiddenName`».

– <nodecombo fieldLabelI18n="*ApprovalStatus*" hiddenName="*ApprovalStatus.values*" propertyName="*ApprovalStatus*" multiSelect="*false*" chkVisible="*false*"/>

3.3.4 Step 3: Referencing the translation of the added field

In the **messages_en.properties** file.

3.3.5 Results: Display in the Product Development module

3.4 Adding a «Properties» panel

It is also possible to configure the screen with a new «panel» composed of existing or new fields.

3.4.1 Step 1: Identification of the panel in which the panel is to be added

The «panel father» in which the new panel is to be inserted must be fully copied in the **custom/Lectra.PDM.ProductGO.xml** file.

If one of the panels of the concerned “panel father” has already been duplicated and configured in the XML file of the custom folder, it is necessary to copy the definition of the panel already modified.

3.4.2 Step 2: Description in the « custom/Lectra.PDM.ProductGO.xml » file

To add the «**properties**» panel to the **GeneralObjectives** screen of the Styles:

3.4.2.1 Duplicating the "description_Style" panel definition

```
<panel name="description_Style" layout="column" autoScroll="true"
                padding="4" >
     <panel columnWidth="0.5" border="false">
     <panel name="identification" titleI18n="Screen.identification"
                inherits="looklikefieldset" >
          …/…
     </panel>
```

```
    <panel name="detail" titleI18n="Screen.specification"
                inherits="looklikefieldset" >
        …/…
    </panel>
    <panel name="collection" titleI18n="Screen.collection"
                columnWidth="0.5" inherits="looklikefieldset" >
        …/…
    </panel>
</panel>
```

To add the «**Properties**» panel to the **GeneralObjectives** screen of the Fabrics, Trims and PackagingLabel:

3.4.2.2 Duplicating the panel definition "description_north_default" or "description_south_default"

```
<panel name="description_north_default" border="false" layout="column">
  <panel name="identification"
            inherits="identification_${#categoryName},identification_${#top
            CategoryName},identification_default" columnWidth="0.5"/>
  <panel name="classification"
            inherits="classification_${#categoryName},classification_${#top
            CategoryName},classification_default" columnWidth="0.5"/>
</panel>
<panel name="description_south_default" border="false" layout="column">
  <panel name="detail" inherits="detail_${#categoryName},
            detail_${#topCategoryName}, detail_default" columnWidth="0.5"/>
  <panel name="specification" inherits="specification_${#categoryName},
            specification_${#topCategoryName}, specification_default"
            columnWidth="0.5"/>
</panel>
```

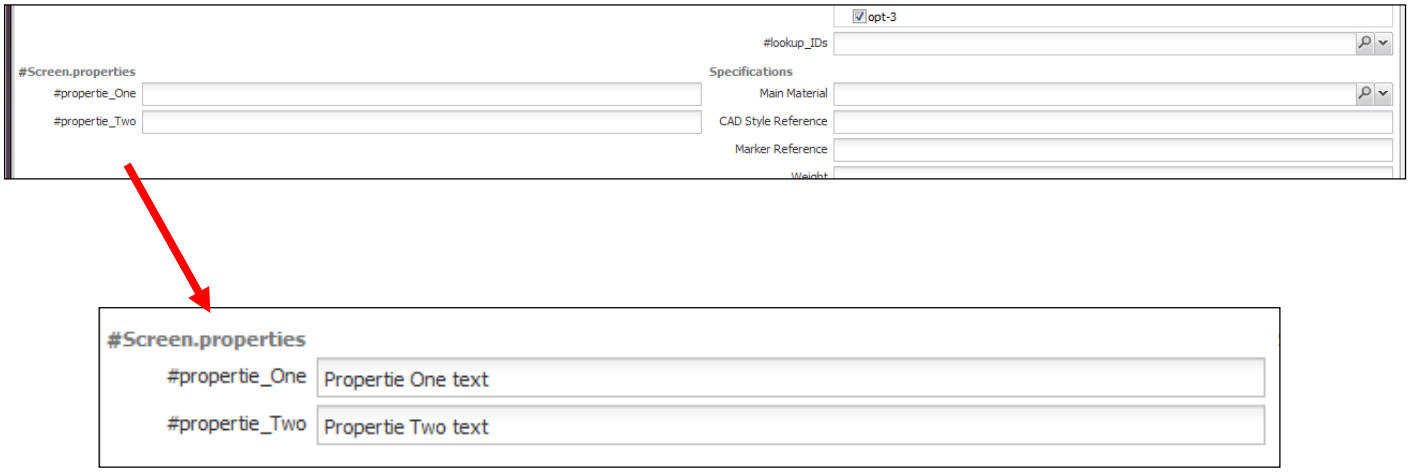
3.4.2.3 Adding the « Properties » panel definition

```
<panel name="description_Style" layout="column" autoScroll="true"
                padding="4">
    <panel name="identification" titleI18n="Screen.identification"
                inherits="looklikefieldset">
        …/…
    </panel>
    <panel name="detail" titleI18n="Screen.specification"
                inherits="looklikefieldset">
        …/…
    </panel>
    <panel name="properties" titleI18n="Screen.properties"
                columnWidth="0.5" inherits="looklikefieldset">
        <textfield fieldLabelI18n="propertie_One" />
        <textfield fieldLabelI18n="propertie_two" />
    </panel>
    <panel name="collection" titleI18n="Screen.collection"
                columnWidth="0.5" inherits="looklikefieldset">
        …/…
    </panel>
</panel>
```

3.4.3 Step 3: Referencing the translation of the added fields names

In the **messages_en.properties** file, translation of **Screen.properties**, **propertie_One**, **propertie_Two**

3.4.4 Result: Display in the Product Development module



4. CONFIGURATION BY ADDING FIELDS IN THE PRODUCT EXPLORER

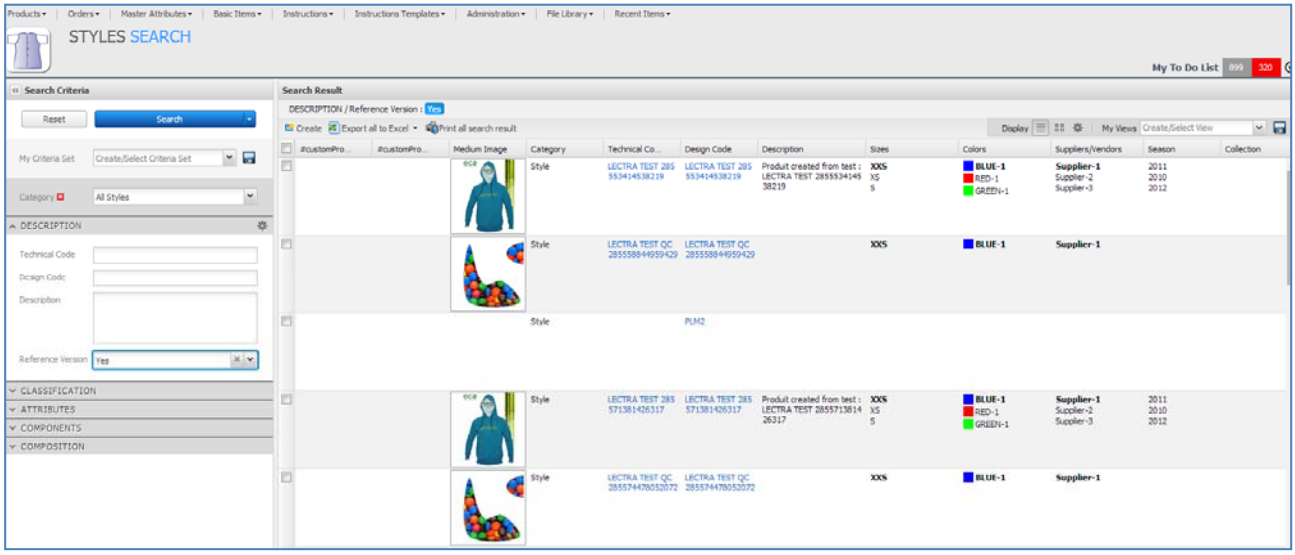
4.1 Prerequisite Administration and Configuration module

The custom fields that will be added to this screen must be defined in the Administration and Configuration module, as described in Appendix H - LPFExt Directory location Change

In PLM V5R1, the location of the LPFExt files is under the lpf/ext3/namespaces directory.

Reminder of the Administration and Configuration Application.

4.2 Presentation of the Product screen



This screen is the same for the different products: **Styles**, **Fabrics**, **Trims** and **PackagingLabel**. It is made up of two customizable parts:

- The **Search** panel displayed on the left and called **Search Criteria**. This part may be configured via the Administration and Configuration module and by adding definitions in the XML file (see below how to do it).
- The **Results** grid, on the right, may be configured directly in the Administration and Configuration module by adding/removing columns, modifying the order of appearance etc. Please refer to the Product Development module User Guide.

To configure the search panel, it is important to understand how the XML file is defined and how to intervene in this definition.

4.3 Customizable XML file

The Products Explorer may be configured by modifying the interface description described in the **Lectra.PDM.Search.Product.xml** file. It is necessary to fill in the file that is in the **custom** sub-

folder **namespaces/custom/Lectra.PDM.Search.Product.xml** (and not the **namespaces/Lectra.PDM.Search.Product.xml)**

If it does not exist, it is necessary to create it in the custom folder and to structure it as follows:

```
<namespace uri="http://lectra.com/pdm/search/product/custom"
           prefix="searchproduct"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:common="http:lectra.com/pdm/common>

<!-- AJOUTER CONFIGURATION ICI -->

<!-- ____________________________  -->
     <explorergrid name="resultGrid_XXX" inherits="resultGrid">
     <!--  XXX : categoryName (Style, Fabric, etc ....) -->
     <!-- Columns description -->
     <!-- securityCells : define the tagNames tagName1, tagName1 etc
           -->
     <!-- tagName:column Name can be controlled before render
           explorer -->
     <colModel inherits="resultGrid/colModel"
           securityColumns="colorTag,supplierTag,descTag"
securityBranch="Explorer" securityType="${#categoryName}"
           colorTag="colors,sizes"
supplierTag="suppliers,seasonsClassif"
           descTag="description,codeAlpha1" />
     <store xsi:type="lpfdirectstore" inherits="resultGrid/store" />
     </explorergrid>

</namespace>
```

4.4 Search panel structure in the XML file

namespaces/Lectra.PDM.Search.Product.xml

The **Search** panel is defined by:

- an explorer `<explorerform name="SearchForm"` which itself defines:
- the panel <panel name="accordionPanel" id="myaccordionPanel"

Each group of criteria (i.e.: “**Description**” group that contains the “**Reference Version**” criterion) is defined by a panel. The content of each «**panel criteria group**» is defined in the file and then referenced in the «**accordionPanel**» panel definition.

Example: For the «Classification» criteria group:

The «**classification**» panel description is done as follows:

```
<!-- ********************************************************** -->
<!--            Collection block definition (Classification)    -->
<!-- ********************************************************** -->
    <panel name="classification" titleI18n="ProductGO.collection" collapsible="true">
        <nodecombo fieldLabelI18n="${#axeRoleName}" hiddenName="${#classifRoleName}.values" forceSelection="true"
            multiSelect="true" propertyName="${#propertyName}" genProductClassif="product"
            chkVisibleExpr="${(#propertyName eq 'Season') or (#propertyName eq 'Division') or (#propertyName eq 'Theme') or

        <!-- Private field(s) : do not edit -->
        <customSerializer type="com.lectra.pdm.lpf.ext.serializers.ProductClassifFromCriteriaGeneratorCustomSerializer" />
        <!-- End of private field(s)        -->
    </panel>
```

Its inclusion in the «**AccordionPanel**» is as follows:

```
<!-- ************************************************************************** -->
<!--            PANEL : Accordion Panel                                         -->
<!-- ************************************************************************** -->
    <panel name="accordionPanel" id="myaccordionPanel" layout="accordionform" activeItem="0"
            autoScroll="true" border="false" isCriteriaPanel="true" style='border-top: 1px solid #fff'>
            <defaults style="border-top: 1px solid #000"/>
        <plugin xsi:type="xmlmap" ptype="panelchildrenvisibility"/>

        <!-- ************************************** -->
        <!--            PANEL : description         -->
        <!-- ************************************** -->
        <panel name="description" inherits="description_${#explorerName},description_${#topCategoryName},description" layout="form"
            border="false" labelWidth="65" labelAlign="left" autoScroll="true" bodyStyle="padding: 14px 20px 10px 20px">
            <defaults anchor="100%" />
        </panel>
        <panel name="classification" inherits="classification_${#explorerName},classification_${#topCategoryName},classification"
            layout="form" border="false" labelWidth="65" labelAlign="left" autoScroll="true" bodyStyle="padding: 14px 20px 10px 20px">
            <defaults anchor="100%" />
        </panel>
        <panel name="axis" inherits="axis_${#explorerName},axis_${#topCategoryName},axis" layout="form"
            border="false" labelWidth="65" labelAlign="left" autoScroll="true" bodyStyle="padding: 14px 20px 10px 20px">
            <defaults anchor="100%" />
        </panel>
        <panel name="mainProduct" inherits="mainProduct_${#explorerName},mainProduct_${#topCategoryName},mainProduct" condition="${!#is
            border="false" labelWidth="65" labelAlign="left" autoScroll="true" bodyStyle="padding: 14px 20px 10px 20px">
            <defaults anchor="100%" />
        </panel>
        <panel name="workflow" inherits="workflow_${#explorerName},workflow_${#topCategoryName},workflow" condition="${!#isPickerExplor
            border="false" labelWidth="65" labelAlign="left" autoScroll="true" bodyStyle="padding: 14px 20px 10px 20px">
            <defaults anchor="100%" />
        </panel>
        <panel name="task" inherits="task_${#explorerName},task_${#topCategoryName},task" condition="${!#isPickerExplorer}" layout="for
            border="false" labelWidth="65" labelAlign="left" autoScroll="true" bodyStyle="padding: 14px 20px 10px 20px">
            <defaults anchor="100%" />
        </panel>
    </panel>
<!-- ************************************************************************** -->
<!--            END PANEL Accordion                                             -->
<!-- ************************************************************************** -->
```

4.5 Identification of the part to be duplicated and Modification

For each modification:

- Identify the relevant part in the **//namespaces/LectraPDM.Search.Product.xml** file (base xml)
- Copy the <panel> that contains this part in the **/namespaces/custom/LectraPDM.Search.Product.xml** file (custom xml)
- Make the change in this duplicated part.
- Update your screen's display: use **F5** to refresh.

4.6 Adding a criterion to an existing group of the search panel

To add a criterion to the «**Description**» group for instance, as seen in the previous example:

4.6.1 Identifying the description block of the «Description» panel in the base XML file

```
<!-- ************************************************************  -->
<!--                    Description block definition                    -->
<!-- ************************************************************  -->
<panel name="description" titleI18n="ProductGO.tree.generalObjectives"
       collapsible="true">
    <textfield name="codeAlpha1.values" fieldLabelI18n="codeAlpha1"
       chkVisible="true" />
    <textfield name="codeAlpha2.values" fieldLabelI18n="codeAlpha2"
       chkVisible="true" />
    <textarea name="description.values" fieldLabelI18n="description"
       chkVisible="true" />
    <twincombo fieldLabelI18n="RefVersion" hiddenName="refVersion.values"
       ignoreFormClear="true" inherits="http://lectra.com/pdm/common#yesno"
       conditionCtrl="com.lectra.lpf.uimodel.FeatureFilter" />
</panel>
```

4.6.2 Duplicating this block in the custom XML file

4.6.3 Adding the fields you want in this block

Example 1: Adding a string of characters for a **Custom_Notes** created in the Administration and Configuration module:

Add the following line at the beginning of the block:

```
    <textarea name="Custom_Notes.values" fieldLabelI18n="Custom_Notes"
       chkVisible="true" />
```

Example 2: Adding a Country List created in the Administration and Configuration module:

Add the following line at the beginning of the block:

```
    <nodecombo fieldLabelI18n="Countries" propertyFamily="process"
       propertyName="Country" multiselect="false" chkVisible="false" />
```

4.7 Adding a group of criteria to the search panel

To the description of the **AccordionPanel** seen in 4.4 - Search panel structure in the XML file, in the **namespaces/Lectra.PDM.Search.Product.xml** file, we will add a new group composed of several fields created in the Administration and Configuration module.

4.7.1 Identifying the «accordionPanel» description block in the base XML file

```
<!-- ************************************************************************** -->
<!--            PANEL : Accordion Panel                                         -->
<!-- ************************************************************************** -->
    <panel name="accordionPanel" id="myaccordionPanel" layout="accordionform" activeItem="0"
            autoScroll="true" border="false" isCriteriaPanel="true" style='border-top: 1px solid #fff'>
            <defaults style="border-top: 1px solid #000"/>
        <plugin xsi:type="xmlmap" ptype="panelchildrenvisibility"/>

        <!-- ************************************** -->
        <!--             PANEL : description        -->
        <!-- ************************************** -->
        <panel name="description" inherits="description_${#explorerName},description_${#topCategoryName},description" layout="form"
            border="false" labelWidth="65" labelAlign="left" autoScroll="true" bodyStyle="padding: 14px 20px 10px 20px">
            <defaults anchor="100%" />
        </panel>
        <panel name="classification" inherits="classification_${#explorerName},classification_${#topCategoryName},classification"
            layout="form" border="false" labelWidth="65" labelAlign="left" autoScroll="true" bodyStyle="padding: 14px 20px 10px 20px">
            <defaults anchor="100%" />
        </panel>
        <panel name="axis" inherits="axis_${#explorerName},axis_${#topCategoryName},axis" layout="form"
            border="false" labelWidth="65" labelAlign="left" autoScroll="true" bodyStyle="padding: 14px 20px 10px 20px">
            <defaults anchor="100%" />
        </panel>
        <panel name="mainProduct" inherits="mainProduct_${#explorerName},mainProduct_${#topCategoryName},mainProduct" condition="${!#is
            border="false" labelWidth="65" labelAlign="left" autoScroll="true" bodyStyle="padding: 14px 20px 10px 20px">
            <defaults anchor="100%" />
        </panel>
        <panel name="workflow" inherits="workflow_${#explorerName},workflow_${#topCategoryName},workflow" condition="${!#isPickerExplor
            border="false" labelWidth="65" labelAlign="left" autoScroll="true" bodyStyle="padding: 14px 20px 10px 20px">
            <defaults anchor="100%" />
        </panel>
        <panel name="task" inherits="task_${#explorerName},task_${#topCategoryName},task" condition="${!#isPickerExplorer}" layout="for
            border="false" labelWidth="65" labelAlign="left" autoScroll="true" bodyStyle="padding: 14px 20px 10px 20px">
            <defaults anchor="100%" />
        </panel>
    </panel>
<!-- ************************************************************************** -->
<!--            END PANEL Accordion                                             -->
<!-- ************************************************************************** -->
```

4.7.2 Duplicating this block in the custom XML file

4.7.3 Adding this new group's description in this block

```
<panel name="CustomGroup" inherits="description_${#explorerName},
      description_${#topCategoryName}, CustomGroup" layout="form"
      border="false" labelWidth="65" labelAlign="left" autoScroll="true"
      bodyStyle="padding: 14px 20px 10px 20px">
      <defaults anchor="100%" />
</panel>
```

Add this block with the correct group name in the desired location, below the «**PANEL: description**» comment.

4.7.4 Definition of these fields as search criteria

Fields that are added in this search panel of the Products screen (whatever the category) are specific fields. Their values are not only displayed (like in the **GeneralObjectives** screen); they help with the product search. These fields must be identified as search criteria in the Administration and Configuration module.

This definition is done through an Excel file whose structure is established and must be correctly entered. This file must be imported into the Administration and Configuration module.

Please refer to the Administration and Configuration module Online Help, paragraph 13: **Search Criteria**

You can also refer to the reminder in Appendix H - LPFExt Directory location Change

In PLM V5R1, the location of the LPFExt files is under the lpf/ext3/namespaces directory.

Reminder of the Administration and Configuration Application

4.8 Adding a column to the results grid

4.8.1 Identifying the description of the results grid

Identify the part in the namespaces/Lectra.PDM.Search.Product.xml file.

```
<!-- ************************************************************ -->
<!-- Grid definition (result): explorerGrid (colModel and store) -->
<!-- ************************************************************ -->
<explorereditablegrid name="resultGrid" explorerSubType="Product"
        explorerType="${#categoryName}" lockColumn="true"
        forceValidation="true" multiCell="true"
        stateId="ExplorerProduct_grid" stateful="true"
        tempCreationId="codeNum2" singleSelectExpr="${#singleSelect}"
        quickCreateMode="false" >

 .../ ...

</explorereditablegrid>
```

This description `<explorereditablegrid>` contains 2 parts to be modified:

- The columns list of the grid :

```
<!-- Columns description -->
<colModel> .../...      </colModel>
```

- The «Store» of necessary data to collect in the base to feed the grid:

```
<!-- Data description -->
<!-- Advice : Do not edit to keep all data definition -->
<store xsi:type="lpfdirectstore" remoteSort="true"
     forceUpdateRecordsOnFail="true">
.../...
</store>
```

4.8.2 Duplicating this block in the custom XML file

If you are not familiar with the 2.2 - Inheritance, duplicate the whole `<explorereditablegrid>` block in the **namespaces/custom/Lectra.PDM.Search.Product.xml** custom file and add the 2 lines necessary to the addition of a column.

4.8.3 Adding the description of a text type column

To add a column that inserts a simple data type:

1. Add the column.
   In the `<colModel>`, add for example :
   For some text:

```
<column      xsi:type="textcolumn"      headerI18n="TEXT_NAME"
dataIndex="TEXT_NAME_INDEX" />
```

   For date, if you want the date to change according to the client timezone:

```
<column      xsi:type="datecolumn"      headerI18n="TEXT_NAME"
dataIndex="DATE_INDEX" />
```

   For date, if you don't want the date to change according to the client timezone:

```
<column      xsi:type="datecolumn"      headerI18n="TEXT_NAME"
dataIndex="DATE_INDEX" useServerTimezone="true" />
```

2. Add the reference to the "Store"
   In the `<store>` block, add for example:

```
<dataField name="TEXT_NAME_INDEX" />

<dataField name=" DATE_INDEX" />
```

You can also refer to Appendix **Erreur ! Source du renvoi introuvable. - Erreur ! Source du renvoi introuvable.**

4.8.4 The store

The Store must be updated for the grid to be correctly completed with the required values. To be updated, each data type contained in the grid must be referenced by a `<dataField>` tag in the Store description. Hence, in the process of retrieving the current «product» and its attributes, only the required values of the «product» attributes will be extracted.

For each attribute necessary for the display, 3 modes allow the definition of an additional filter level. With mode= «**full**», «**shallow**» or «**strict**», the data will be reported in a descending order.

`mode="full"`: total. The entire object (including its attributes) is loaded from the base.

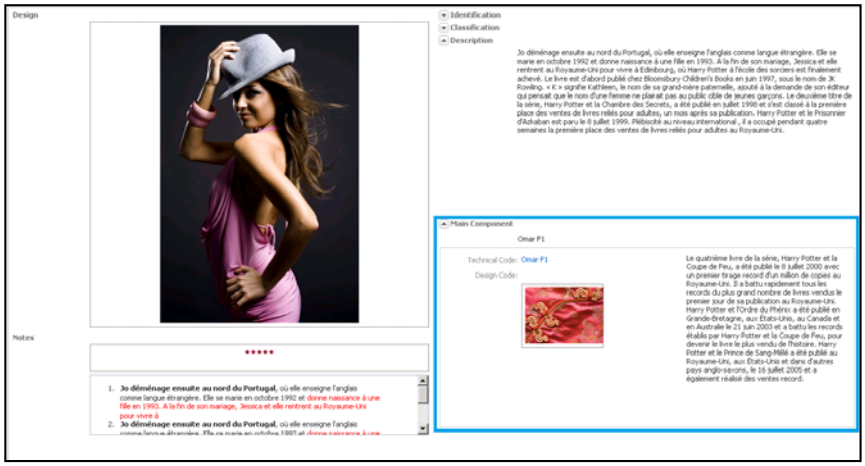
`mode="shallow"`: partial. The simple attributes of the object will be recovered from the base.

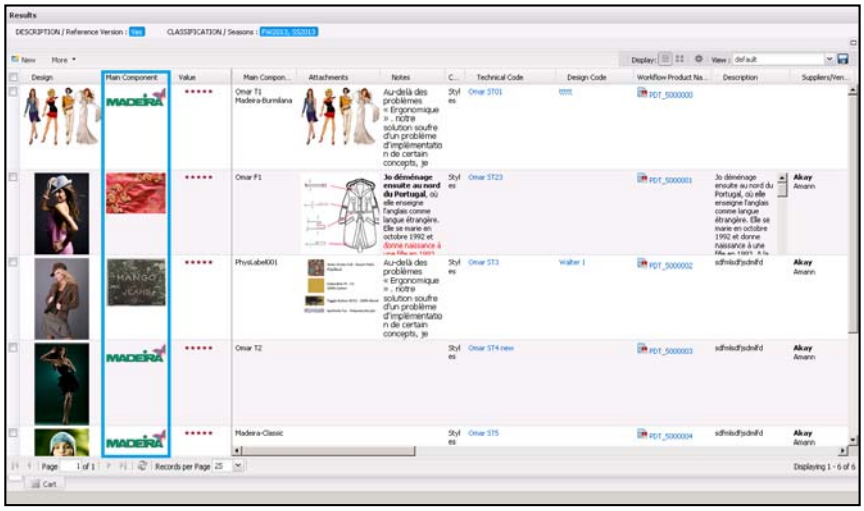
`mode="strict"`: minimal. Only the name is recovered.

4.8.5 Adding columns to Custom Fields

Here is a configuration example:

Objective: Product link in Style GO and get it into search grid.





Step 1: Prerequisite Administration and Configuration module

Definition of a «**Custom Role** » with the «**Product**» «**Target**» on the «**Style**» entity. In the example it is called «**mainComponent**».

Step 2: Adding the «**mainComponent**» panel to the **GeneralObjectives** screen

LectraPLMParam\namespaces\custom\Lectra.PDM.ProductGO.xml file

```
<panel name="mainComponent" titleI18n="mainComponent" columnWidth="1.0"
border="false" collapsible="true" >
        <productcombo hiddenName="mainComponent" openerType="${#type}" />
        <panel  columnWidth="1." layout="column" >
             <panel columnWidth="0.4" border="false">
                <rendererfield name="mainComponent" fieldLabelI18n="technicalCode"
                listenToProductCombo="mainComponent">
                <rendererFn name=
                "Lectra.PDM.Format.productLinkRenderer('codeAlpha1')" />
                </rendererfield>
                <rendererfield name="mainComponent" fieldLabelI18n="studyCode"
                listenToProductCombo="mainComponent">
                <rendererFn name="Lectra.PDM.Format.productLinkRenderer
                ('codeAlpha2')" />
                </rendererfield>
```

```
                <imagefield name="mainComponent.defaultImageField"
                imageSize="large" readOnly="true" submitValue="false"
                listenToProductCombo="mainComponent" >
                    <managerInstance name="Lectra.PDM.Mgrs.imageFieldMgr"/>
            </imagefield>
        </panel>
            <panel columnWidth="0.6" border="false" >
                <textarea name="mainComponent.description"  readOnly="true"
                submitValue="false" listenToProductCombo="mainComponent" />
            </panel>
        </panel>
</panel>
```

Step 3: Adding the column to the results grid of the Products explorer

LectraPLMParam\namespaces\custom\Lectra.PDM.Search.Product.xml file

```
<!-- Grid definition (result) : explorerGrid  (colModel and store)-->
<explorergrid name="resultGrid" explorerSubType="Product"
            explorerType="${#categoryName}" lockColumn="true"
            forceValidation="true" multiCell="true"
            stateId="ExplorerProduct_grid" stateful="true"> <!--
            autoLoadOnRender="true">-->

      <!-- Columns description -->
      <colModel>

      .../...

      <column xsi:type="productcolumn" headerI18n="mainComponent"
            dataIndex="mainComponent" subDataIndex="defaultImageField"
            editable="true"/>
      <column xsi:type="imagecolumn" headerI18n="mainComponent"
            dataIndex="mainComponent" subDataIndex="defaultImageField"/>

      .../....

      </colModel>
      <!-- Data description -->
      <!-- Advice : Do not edit to keep all data definition -->
      <store xsi:type="lpfdirectstore" remoteSort="true"
            forceUpdateRecordsOnFail="true">
      .../...
            <dataField name="mainComponent" mode="shallow">
                  <filter name="defaultImageField" mode="full"/>
            </dataField>
      .../...
      </store>
      .../...
</explorergrid>
```

Code to add

5. CONFIGURATION BY ADDING FIELDS IN THE COLLECTION PLAN

5.1 Prerequisite Administration and Configuration module

The custom fields that will be added to this screen must be defined in the Administration and Configuration module, as described in Appendix H - LPFExt Directory location Change

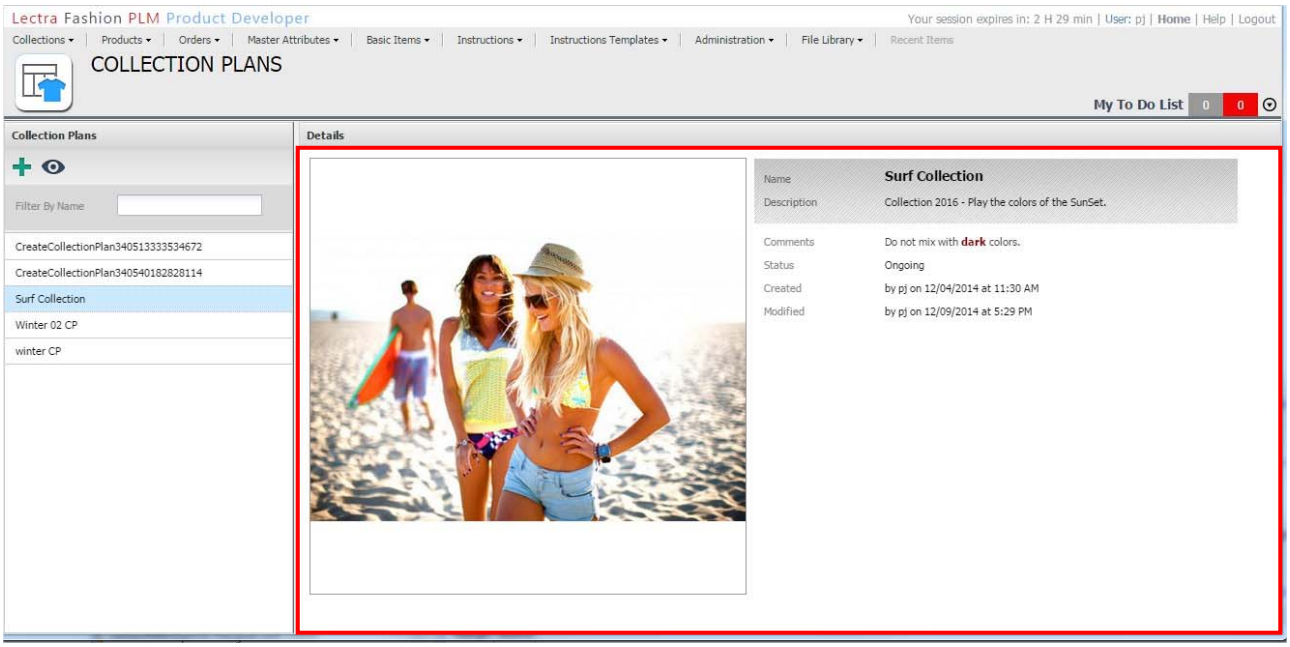
In PLM V5R1, the location of the LPFExt files is under the lpf/ext3/namespaces directory.

Reminder of the Administration and Configuration Application, Appendix K - Adding custom fields and custom roles to General Data of a Collection Plan, Appendix L – Adding custom fields and custom roles to Slot Breakdown grid of a Collection Plan.

5.2 Presentation of the Collection Plan screens

5.2.1 List Collection Plan

5.2.1.1 Presentation of the screen



This screen has a single customizable part: the center (red framed). Note - the information is 'display only'. Edit mode is not possible on this screen.

5.2.1.2 Customizable XML file

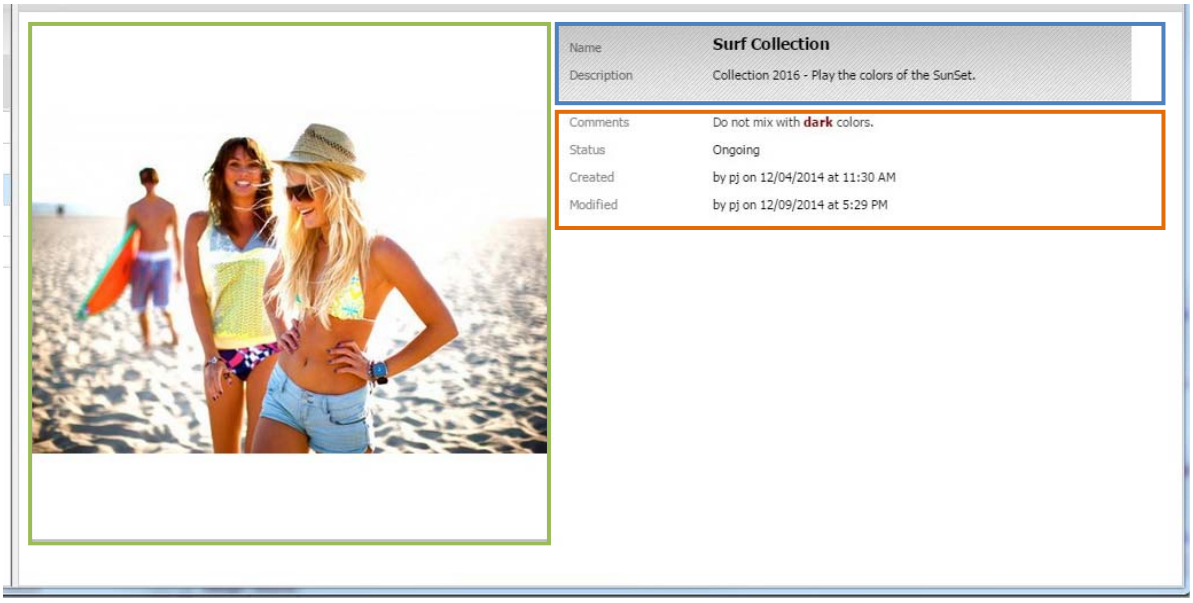
The list collection plan screen may be customized by modifying the **Lectra.PDM.CollectionPlan.ListCollectionPlan.xml** file. It is necessary to fill in the file that is in

the **custom** sub-folder namespaces/**custom**/ Lectra.PDM.CollectionPlan.ListCollectionPlan.xml (and not the original namespaces/ Lectra.PDM.CollectionPlan.ListCollectionPlan.xml)

If it does not exist, it is necessary to create it in the custom folder and to structure it as follows:

```
<namespace xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
uri=http://lectra.com/pdm/listCollectionPlan/custom prefix="listCollectionPlan">

      <fieldset name="listCollectionDetailsDefault" border="false" hidden="true"
layout="column" padding="0">
            <fieldset inherits="listCollectionDetails/listCollectionDetailsLeft" >
                  <!-- See Dictionnary.xml for the available kinds of fields -->
            </fieldset>
            <fieldset inherits="listCollectionDetails/listCollectionDetailsRight" >
                  <!-- See Dictionnary.xml for the available kinds of fields -->
            </fieldset>
      </fieldset>

</namespace>
```

5.2.1.3 Identification of the part to be modified



The center part has this structure:

listCollectionDetails (fieldset)

|_ listCollectionDetailsLeft (fieldset)

|_ **imagefield**

|_ listCollectionDetailsRight (fieldset)

|_ no name (fieldset)

|_ **name (displayfield)**

|_ **description (displayfield)**

|_ no name (fieldset)

|_ **comments (displayfield)**

|_ **status (displayfield)**

|_ **created (displayfield)**

|_ **modified (displayfield)**

5.2.1.4 Adding fields under imagefield of listCollectionDetailsLeft

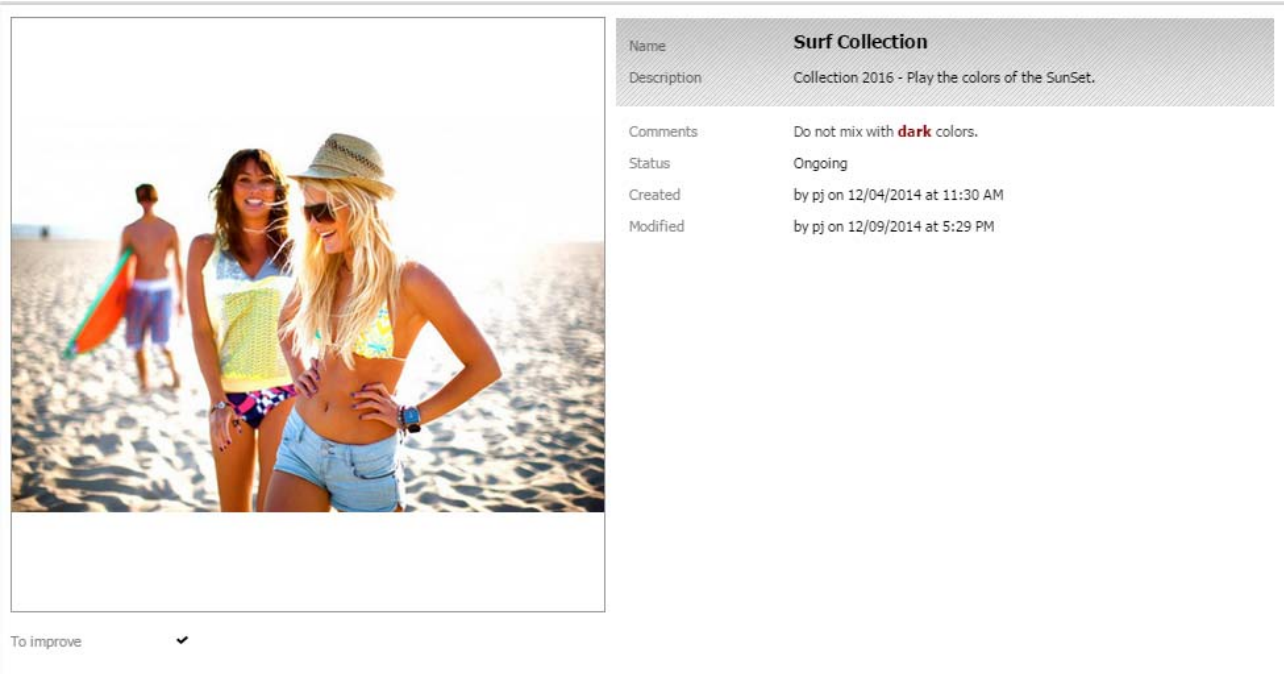
Add fields and roles you have created into Administration and Configuration module, into fieldset tag:

```
<fieldset inherits="listCollectionDetails/listCollectionDetailsLeft" >

    <!-- See Dictionnary.xml for the available kinds of fields -->

</fieldset>
```

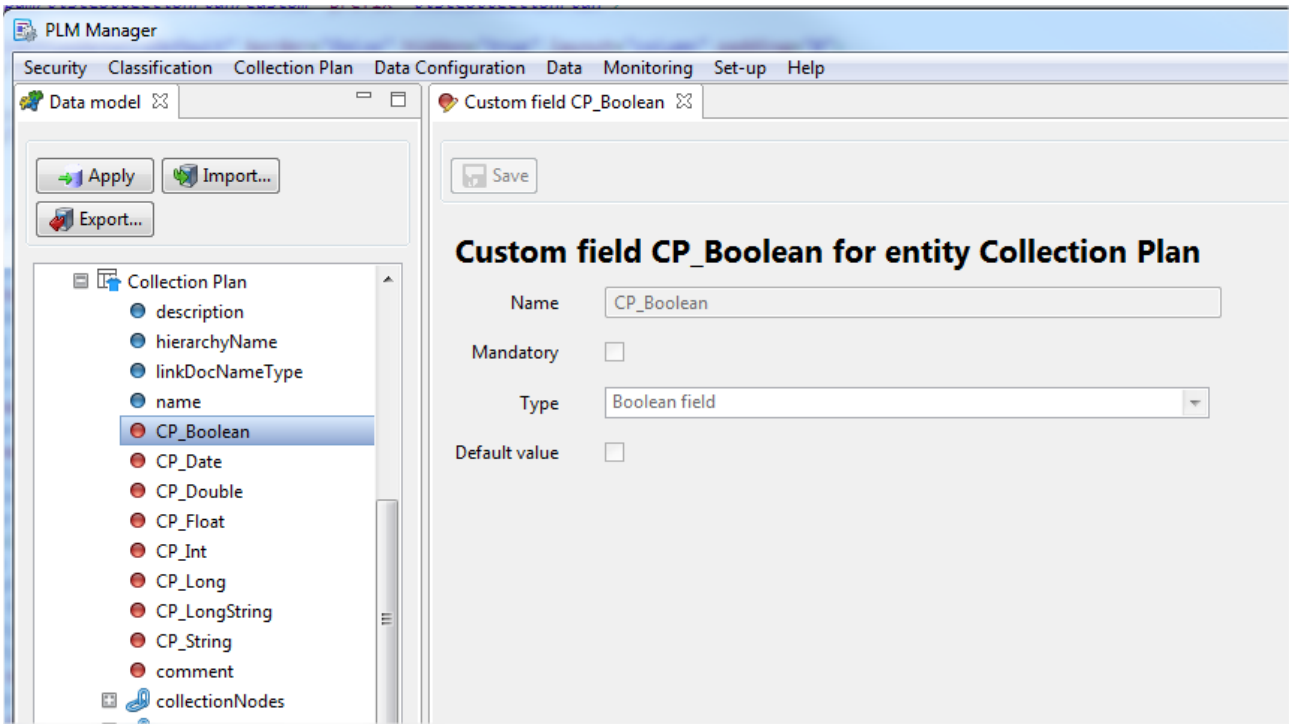
And refresh your browser to see the modification.

Example: a checkbox displayed under the image

```
<fieldset inherits="listCollectionDetails/listCollectionDetailsLeft" >

    <!-- See Dictionnary.xml for the available kinds of fields -->
    <checkbox name="CP_Boolean" fieldLabelI18n="custom.cp.label.boolean" readOnly="true" />

</fieldset>
```



Note checkbox name is the same like created into Administration and Configuration module:



5.2.1.5 Adding fields up to imagefield of listCollectionDetailsLeft

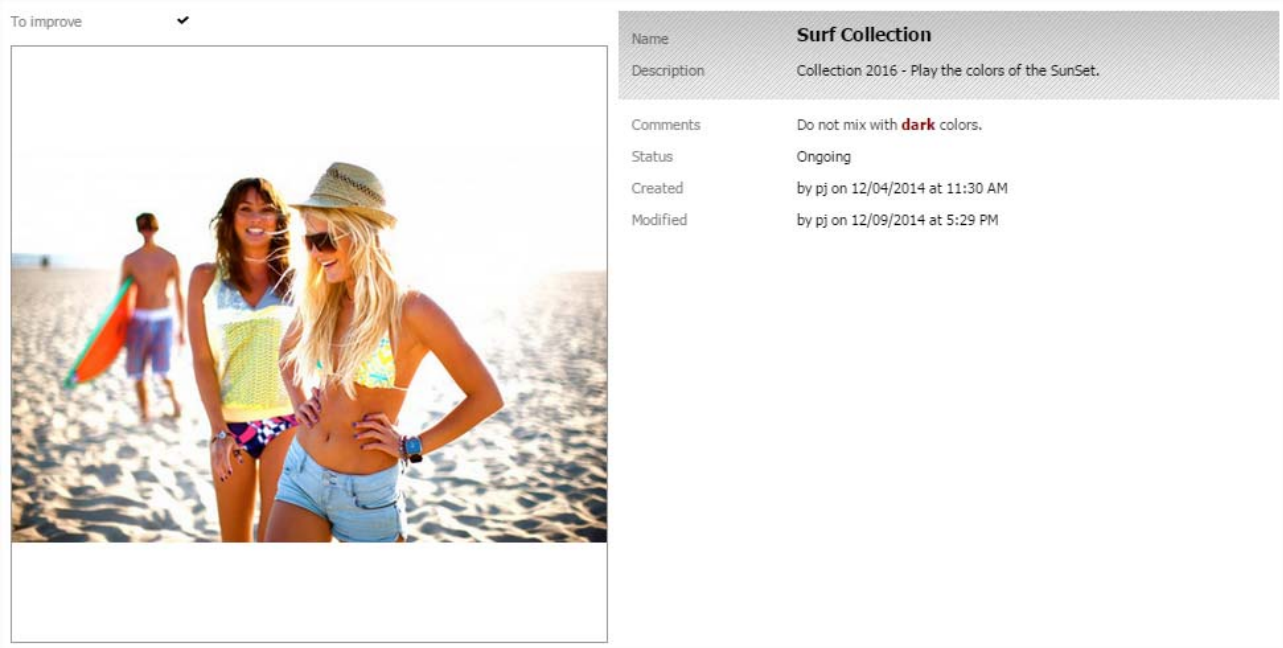
If you need to add fields and roles up to this imagefield, you can no longer use the inheritance mechanism. The fieldset needs to be redefined by copying the one from the original namespaces/Lectra.PDM.CollectionPlan.ListCollectionPlan.xml file:

```
<fieldset name="listCollectionDetailsLeft" padding="10" border="false" margin="-4 0 0 0">
    <imagefield fieldLabelI18n="action.toggle.storyBoard" hideLabel="true" name="attachment" readOnly="true" useREST="true"
    inherits="http://lectra.com/pdm/common#defaultImageFieldEXT5" managerRef="attachmentImageField"
                imageWidth="450" imageHeight="450" width="460" height="460" imageSize="full">
        <managerInstance name="Lectra.PDM.Mgrs.imageFieldRestMgr"/>
    </imagefield>
</fieldset>
```

Following this it needs to be pasted into the custom namespaces/**custom**/Lectra.PDM.CollectionPlan.ListCollectionPlan.xml file, in place of the inherited fieldset, and modify it as required.

Example: a checkbox displayed above the image

```
<fieldset name="listCollectionDetailsLeft" padding="10" border="false" margin="-4 0 0 0">
    <checbox name="CP_Boolean" fieldLabelI18n="custom.cp.label.boolean" readOnly="true" />
    <imagefield fieldLabelI18n="action.toggle.storyBoard" hideLabel="true" name="attachment" readOnly="true" useREST="true"
    inherits="http://lectra.com/pdm/common#defaultImageFieldEXT5" managerRef="attachmentImageField"
                imageWidth="450" imageHeight="450" width="460" height="460" imageSize="full">
        <managerInstance name="Lectra.PDM.Mgrs.imageFieldRestMgr"/>
    </imagefield>
</fieldset>
```



5.2.1.6 Adding fields under fieldsets of listCollectionDetailsRight

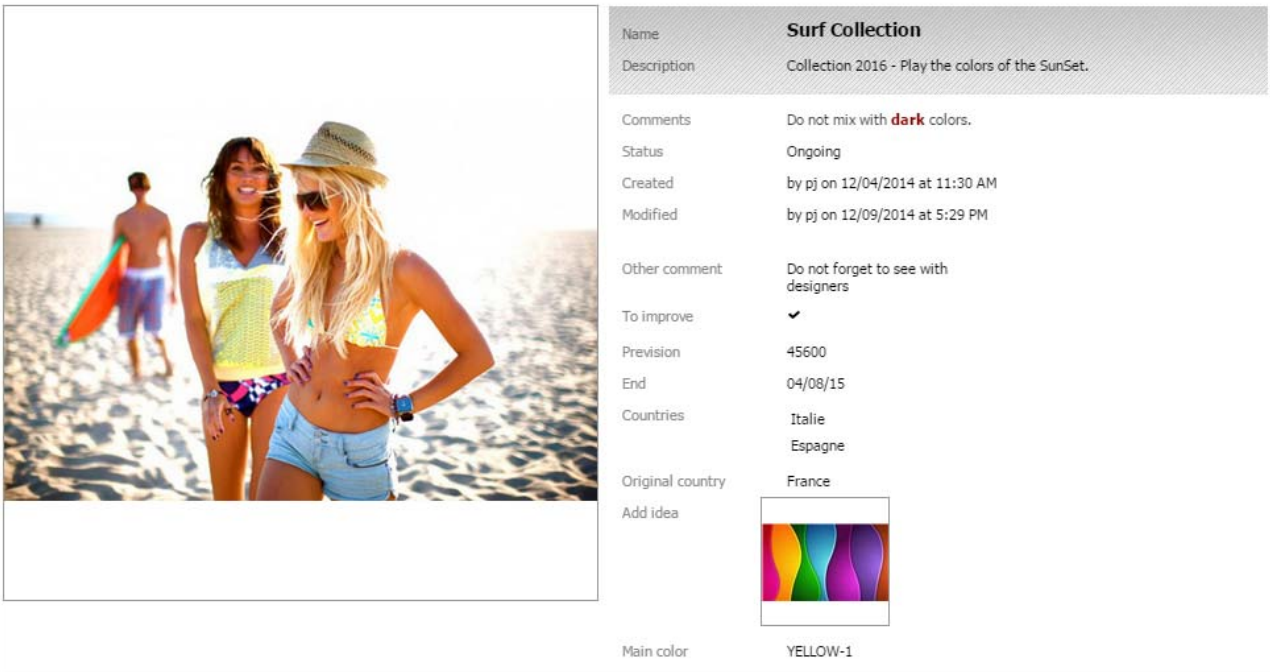
Add fields and roles you have created in the Administration and Configuration module, in the fieldset tag:

```
<fieldset inherits="listCollectionDetails/listCollectionDetailsRight" >

    <!-- See Dictionnary.xml for the available kinds of fields -->

</fieldset>
```

Refresh your browser to view the modification.

Example: a text area, a checkbox, an integer, a date, a multiple combo, a single combo, an image and an envitem combo displayed under the fieldsets

```
<fieldset inherits="listCollectionDetails/listCollectionDetailsRight" >

    <!-- See Dictionnary.xml for the available kinds of fields -->
    <fieldset padding="10" border="false" margin="-3 0 0 0">
        <textarea name="CP_LongString" fieldLabelI18n="custom.cp.label.longstring" readOnly="true" />
        <checkbox name="CP_Boolean" fieldLabelI18n="custom.cp.label.boolean" readOnly="true" />
        <integerfield name="CP_Int" fieldLabelI18n="custom.cp.label.int" readOnly="true" />
        <datefield name="CP_Date" fieldLabelI18n="custom.cp.label.date" readOnly="true" />
        <nodecombo name="CP_Country_Multi" multiSelect="true" propertyName="Country" fieldLabelI18n="custom.cp.label.hvl.country.multi" >
            <bind readOnly="true" />
        </nodecombo>
        <nodecombo name="CP_Country_Single" multiSelect="false" propertyName="Country" fieldLabelI18n="custom.cp.label.hvl.country.single" >
            <bind readOnly="true" />
        </nodecombo>
        <imagefield name="CP_Doc" fieldLabelI18n="custom.cp.label.document" useREST="true"
            inherits="http://lectra.com/pdm/common#defaultImageFieldEXT5" managerRef="selSmallImageField" imageWidth="96" imageHeight="96" readOnly="true" >
            <managerInstance name="Lectra.PDM.Mgrs.imageFieldRestMgr" />
        </imagefield>
        <envitemcombo name="CP_Color" envItemType="MarketingColor" fieldLabelI18n="custom.cp.label.envitem.color" hideTrigger2="true" readOnly="true" />
    </fieldset>

</fieldset>
```



5.2.1.7 Adding fields under first fieldset of listCollectionDetailsRight

If you need to add fields and roles just under this fieldset, you can no longer use the inheritance mechanism. Instead, you have to redefine the embracing listCollectionDetailsRight fieldset, by copying the one from the original namespaces/ Lectra.PDM.CollectionPlan.ListCollectionPlan.xml file:

```
<fieldset name="listCollectionDetailsRight" width="520" padding="10" border="false" margin="-1 0 0 -20">
    <fieldset padding="10" margin="0" border="false" cls="pdm-identitycard">
        <defaults labelSeparator=" " />
        <displayfield name="name" fieldLabelI18n="name" fieldCls="pdm-header-1" />
        <displayfield name="description" fieldLabelI18n="description" />
    </fieldset>
    <fieldset padding="10" border="false" margin="-3 0 0 0">
        <defaults labelSeparator=" " />
        <displayfield name="comment" fieldLabelI18n="comment" />
        <displayfield name="statusName" fieldLabelI18n="status" />
        <displayfield name="formatCreatedDateWithUser" fieldLabelI18n="label.authoring.created.nosemicolon" />
        <displayfield name="formatModifiedDateWithUser" fieldLabelI18n="label.authoring.modified.nosemicolon" />
    </fieldset>
</fieldset>
```

Following this it needs to be pasted into the custom namespaces/**custom**/ Lectra.PDM.CollectionPlan.ListCollectionPlan.xml file, in place of the inherited fieldset, and modify it as required:

Example: a checkbox displayed after the description field.

```
<fieldset name="listCollectionDetailsRight" width="520" padding="10" border="false" margin="-1 0 0 -20">
    <fieldset padding="10" margin="0" border="false" cls="pdm-identitycard">
        <defaults labelSeparator=" " />
        <displayfield name="name" fieldLabelI18n="name" fieldCls="pdm-header-1" />
        <displayfield name="description" fieldLabelI18n="description" />
        <checkbox name="CP_Boolean" fieldLabelI18n="custom.cp.label.boolean" readOnly="true" />
    </fieldset>
    <fieldset padding="10" border="false" margin="-3 0 0 0">
        <defaults labelSeparator=" " />
        <displayfield name="comment" fieldLabelI18n="comment" />
        <displayfield name="statusName" fieldLabelI18n="status" />
        <displayfield name="formatCreatedDateWithUser" fieldLabelI18n="label.authoring.created.nosemicolon" />
        <displayfield name="formatModifiedDateWithUser" fieldLabelI18n="label.authoring.modified.nosemicolon" />
    </fieldset>
</fieldset>
```

Name	Surf Collection
Description	Collection 2016 - Play the colors of the SunSet.
To improve	✔
Comments	Do not mix with **dark** colors.
Status	Ongoing
Created	by pj on 12/04/2014 at 11:30 AM
Modified	by pj on 12/09/2014 at 5:29 PM

5.2.1.8 Adding fields and modifying global listCollectionDetails fieldset

If you need to add fields and roles, to have another layout rendering, see Layout Modification section .

5.2.1.9 Notes about fields and roles in this screen

Here are the different fields and roles (with examples) that you can add in this screen, which are in read only mode:

String field

<textfield name="*CP_String*" fieldLabelI18n="*custom.cp.label.string*" **readOnly="true"** />

LongString field

<textarea name="*CP_LongString*" fieldLabelI18n="*custom.cp.label.longstring*" **readOnly="true"** />

Boolean field

<checkbox name="*CP_Boolean*" fieldLabelI18n="*custom.cp.label.boolean*" **readOnly="true"** />

Long field

<longfield name="*CP_Long*" fieldLabelI18n="*custom.cp.label.long*" **readOnly="true"** />

Integer field

<integerfield name="*CP_Int*" fieldLabelI18n="*custom.cp.label.int*" **readOnly="true"** />

Double field

<doublefield name="*CP_Double*" fieldLabelI18n="*custom.cp.label.double*" **readOnly="true"** />

Float field

<floatfield name="*CP_Float*" fieldLabelI18n="*custom.cp.label.float*" **readOnly="true"** />

Date field

<datefield name="*CP_Date*" fieldLabelI18n="*custom.cp.label.date*" **readOnly="true"** />

HVL pick list example with Country and Multiselect

<nodecombo name="*CP_Country_Multi*" multiSelect="true" propertyName="Country" fieldLabelI18n="*custom.cp.label.hvl.multi*" >

<bind readOnly="true" />

</nodecombo>

HVL pick list example with Country and Singleselect

<nodecombo name="*CP_Country_Single*" multiSelect="false" propertyName="Country" fieldLabelI18n="*custom.cp.label.hvl.single*" >

<bind readOnly="true" />

</nodecombo>

Document target picker

<imagefield name="*CP_Doc*" fieldLabelI18n="*custom.cp.label.document*" useREST="true"

inherits="http://lectra.com/pdm/common#defaultImageFieldEXT5" managerRef="selSmallImageField" imageWidth="96" imageHeight="96" **readOnly="true"** >

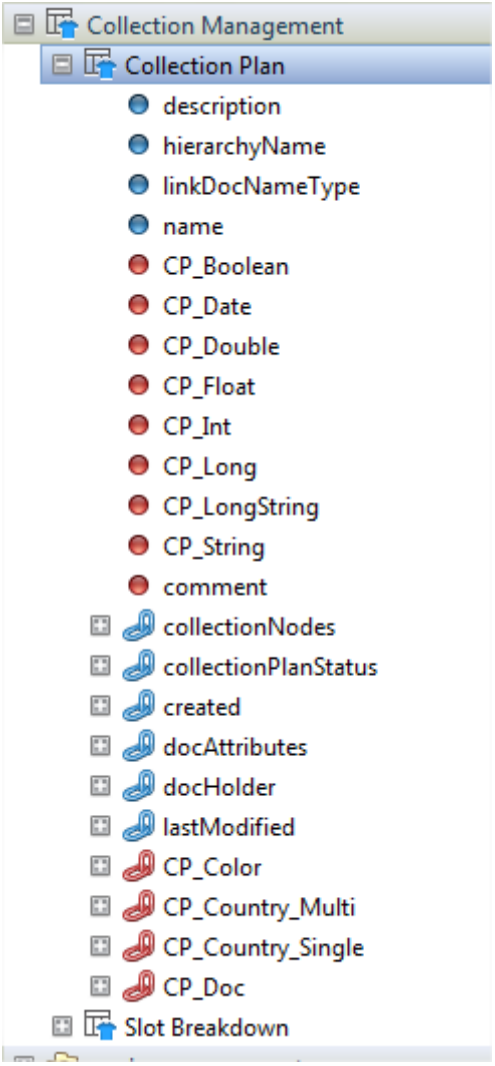
<managerInstance name="Lectra.PDM.Mgrs.imageFieldRestMgr" />

</imagefield>

envitem (multiselect is not allowed) : example with MarketingColor

<envitemcombo name="*CP_Color*" envItemType="MarketingColor" fieldLabelI18n="*custom.cp.label.envitem.color*" hideTrigger2="true**" readOnly="true"** />

- Note *name* value attribute is the same as those created in Administration and Configuration module:



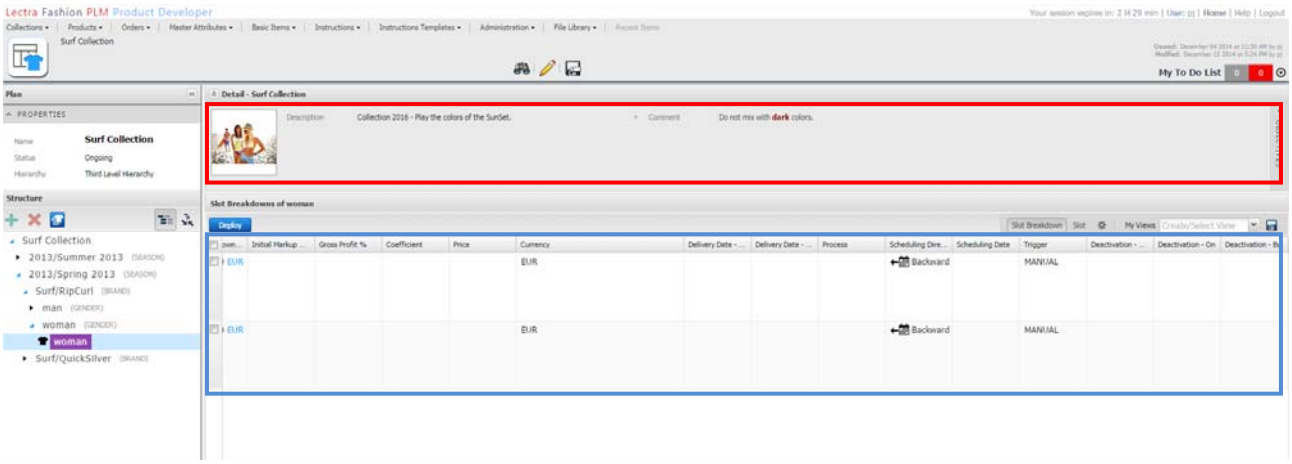
- Note *fieldLabelI18n* value attribute should be defined in PLM-Fashion\PDM\LectraPLMParam\messagesCusto_en.properties file (and other file language if needed; example messagesCusto_fr.properties for French)

```
custom.cp.label.longstring=Other comment
custom.cp.label.boolean=To improve
custom.cp.label.int=Prevision
custom.cp.label.date=End
custom.cp.label.hvl.country.multi=Countries
custom.cp.label.hvl.country.single=Original country
custom.cp.label.document=Add idea
custom.cp.label.envitem.color=Main color
custom.cp.rp.label.date=Begining
custom.cp.rp.label.boolean=Stop
custom.cp.rp.label.string=Comment 2
custom.cp.rp.label.hvl.country.multi=Countries
custom.cp.rp.label.hvl.country.single=Country
custom.cp.rp.label.document=Recent image
custom.cp.rp.label.envitem.color=Main color
custom.cp.rp.label.envitem.unit=Unit
```

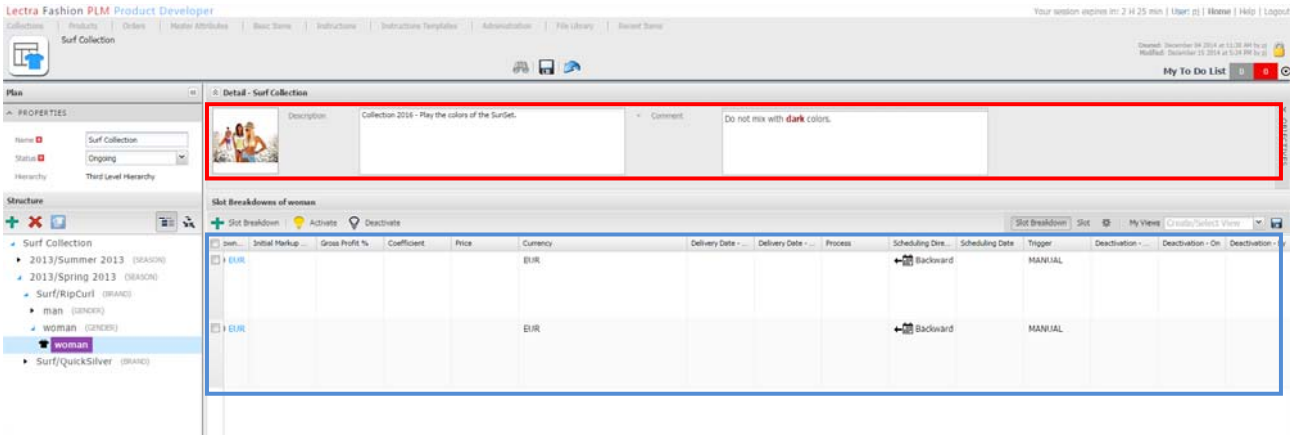
5.2.2 Detail Collection Plan

5.2.2.1 Presentation of the screen

Display mode:



Edit mode:



This screen has two customizable parts: the general data (red framed), and the slot breakdown grid (blue framed). Note there are two modes: display and edit.

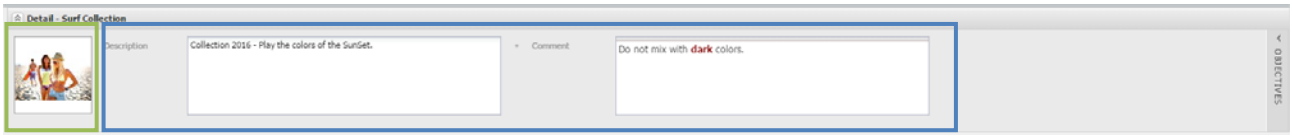
5.2.2.2 Customizable XML file

The list collection plan screen may be customized by modifying the **Lectra.PDM.CollectionPlan.CollectionPlan.xml** file. It is necessary to fill in the file that is in the **custom** sub-folder namespaces/**custom**/ Lectra.PDM.CollectionPlan.CollectionPlan.xml (and not the original namespaces/ Lectra.PDM.CollectionPlan.CollectionPlan.xml)

If it does not exist, it is necessary to create it in the custom folder and to structure it as follows:

```
<namespace xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
uri="http://lectra.com/pdm/collectionPlan/custom"
               prefix="collectionPlan">
       <!-- General Data Form - Display and Edit Mode -->
     <form name="detailsEditDefault" layout="border" id="editForm"
reference="collectionPlanDetailsForm" border="false">
       <panel inherits="detailsEdit/detailsEditWest" />
             <panel inherits="detailsEdit/detailsEditCenter" overflowX="auto" overflowY="auto"
layout="form">
                     <!-- See Dictionnary.xml for the available kinds of fields -->
             </panel>
    </form>
       <!-- Grid Range Plan Slot Breakdown -->
    <grid name="rangeplanslotbreakdowngridDefault" inherits="rangeplanslotbreakdowngrid" >
               <!-- See Dictionnary.xml for the available kinds of fields -->
       </grid>
</namespace>
```

5.2.2.3 Identification of the part to be modified



The general data part has this structure:

detailsEdit (form)

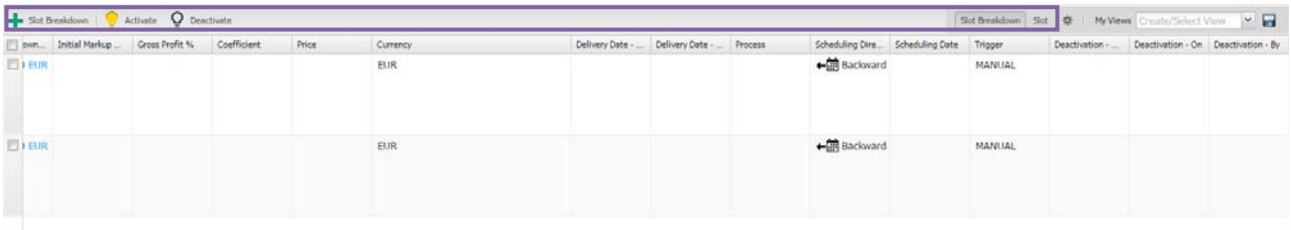
|_ detailsEditWest (panel)

|_ **imagefield**

|_ detailsEditCenter (panel)

|_ **description (textarea)**

|_ **comment (htmleditor)**



The slot breakdown grid part has this structure:

rangeplanslotbreakdowngrid (rangeplangrid)

|_ no name (toolbar containing buttons)

|_ (column 1)

|_ (column 2)

|_ etc.

5.2.2.4 Adding fields under imagefield of detailsEditWest (general data)

Add fields and roles you have created in the Administration and Configuration module, in the panel tag:

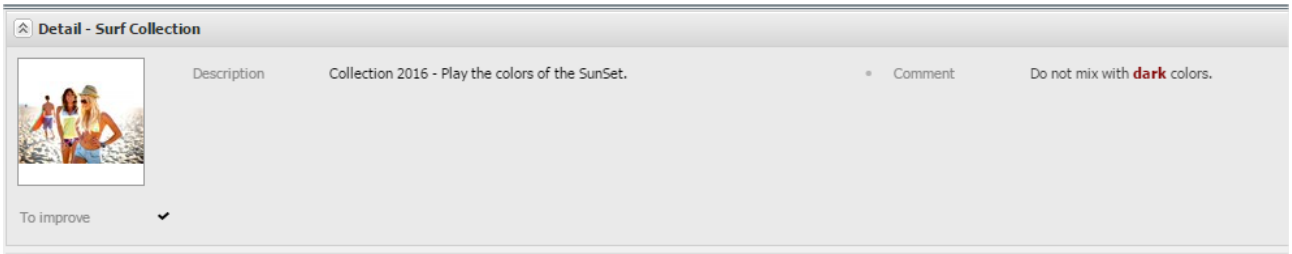
```
<panel inherits="detailsEdit/detailsEditWest" />
```

And refresh your browser to see the modification.

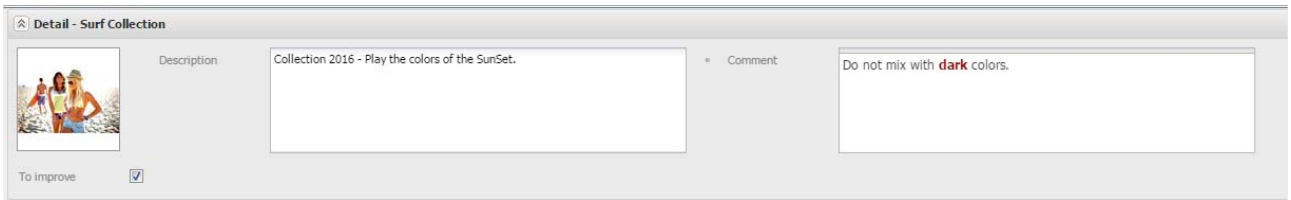
<u>Example:</u> a checkbox displayed under the image

```
<panel inherits="detailsEdit/detailsEditWest" >
    <checkbox name="CP_Boolean" fieldLabelI18n="custom.cp.label.boolean" margin="10 0 0 10">
        <bind readOnly="{!isPageEditable}" value="{collectionPlan.CP_Boolean}" />
    </checkbox>
</panel>
```

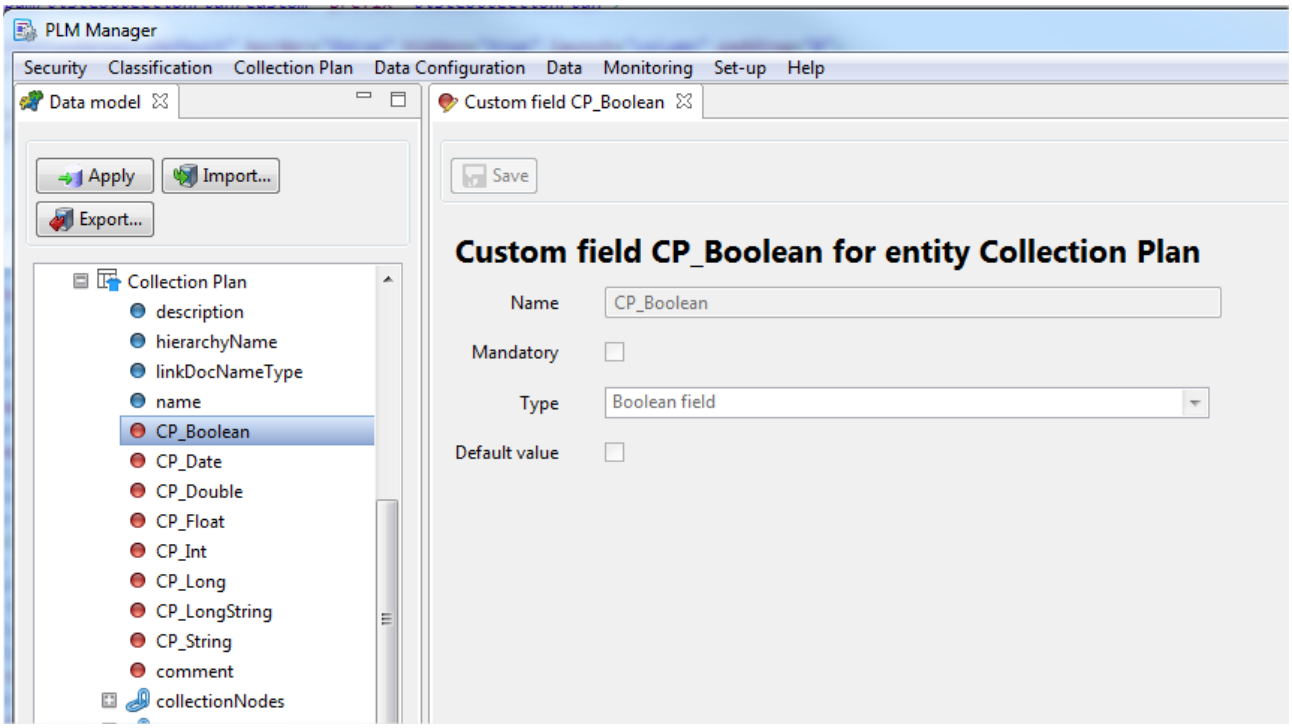
Display mode:



Edit mode:



- Note: the checkbox name is the same as those created in the Administration and Configuration module:



- Note *bind* value attribute is "**{collectionPlan.***created_name***}**". You need to use this template.

- Note: switching between display and edit mode is conditioned by *bind* tag with *readOnly* attribute. You need to use this mechanism.

5.2.2.5 Adding fields up to imagefield of "detailsEditWest" (general data)

If you need to add fields and roles up to this imagefield, you can no longer use the inheritance mechanism. The panel needs to be redefined by copying the one from the original namespaces/ Lectra.PDM.CollectionPlan.CollectionPlan.xml file:

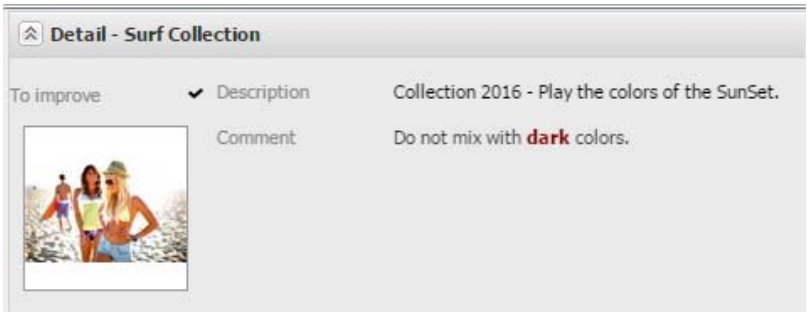
```
<panel name="detailsEditWest" border="false" region="west">
    <imagefield fieldLabelI18n="action.toggle.storyBoard" hideLabel="true" name="attachment"
    useREST="true" inherits="http://lectra.com/pdm/common#defaultImageFieldEXT5" managerRef="selSmallImageField"
                   imageSize="medium" imageWidth="96" imageHeight="96" padding="8 0 0 8" >
        <managerInstance name="Lectra.PDM.Mgrs.imageFieldRestMgr" />
        <bind fileLinkObj="{collectionPlan.attachment}" readOnly="{!isPageEditable}"/>
    </imagefield>
</panel>
```

Then paste it into the custom namespaces/**custom**/ Lectra.PDM.CollectionPlan.CollectionPlan.xml file, in place of the inherited panel, and modify it as required.

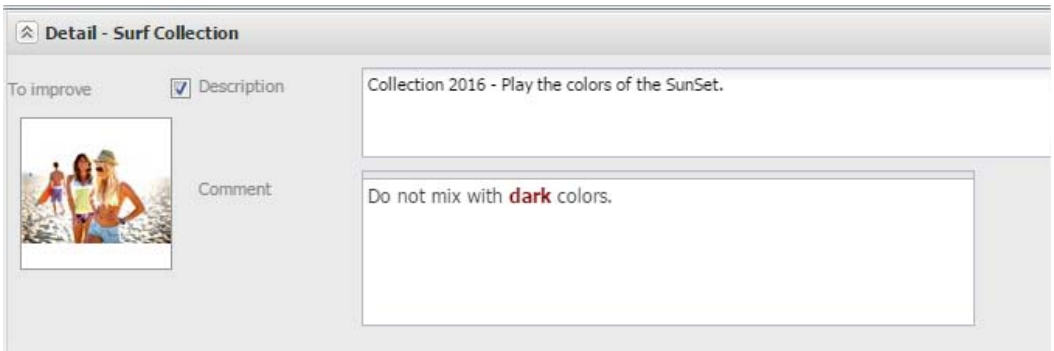
Example: a checkbox displayed up to the image

```
<panel name="detailsEditWest" border="false" region="west">
    <checkbox name="CP_Boolean" fieldLabelI18n="custom.cp.label.boolean" margin="10 0 0 10">
        <bind readOnly="{!isPageEditable}"  value="{collectionPlan.CP_Boolean}"/>
    </checkbox>
    <imagefield fieldLabelI18n="action.toggle.storyBoard" hideLabel="true" name="attachment"
    useREST="true" inherits="http://lectra.com/pdm/common#defaultImageFieldEXT5" managerRef="selSmallImageField"
                   imageSize="medium" imageWidth="96" imageHeight="96" padding="8 0 0 8" >
        <managerInstance name="Lectra.PDM.Mgrs.imageFieldRestMgr" />
        <bind fileLinkObj="{collectionPlan.attachment}" readOnly="{!isPageEditable}"/>
    </imagefield>
</panel>
```

Display mode:



Edit mode:



5.2.2.6 Adding fields under comment and description of detailsEditCenter (general data)

Add fields and roles you have created in the Administration and Configuration module, in the panel tag:

```
<panel inherits="detailsEdit/detailsEditCenter" overflowX="auto" overflowY="auto" layout="form">
```

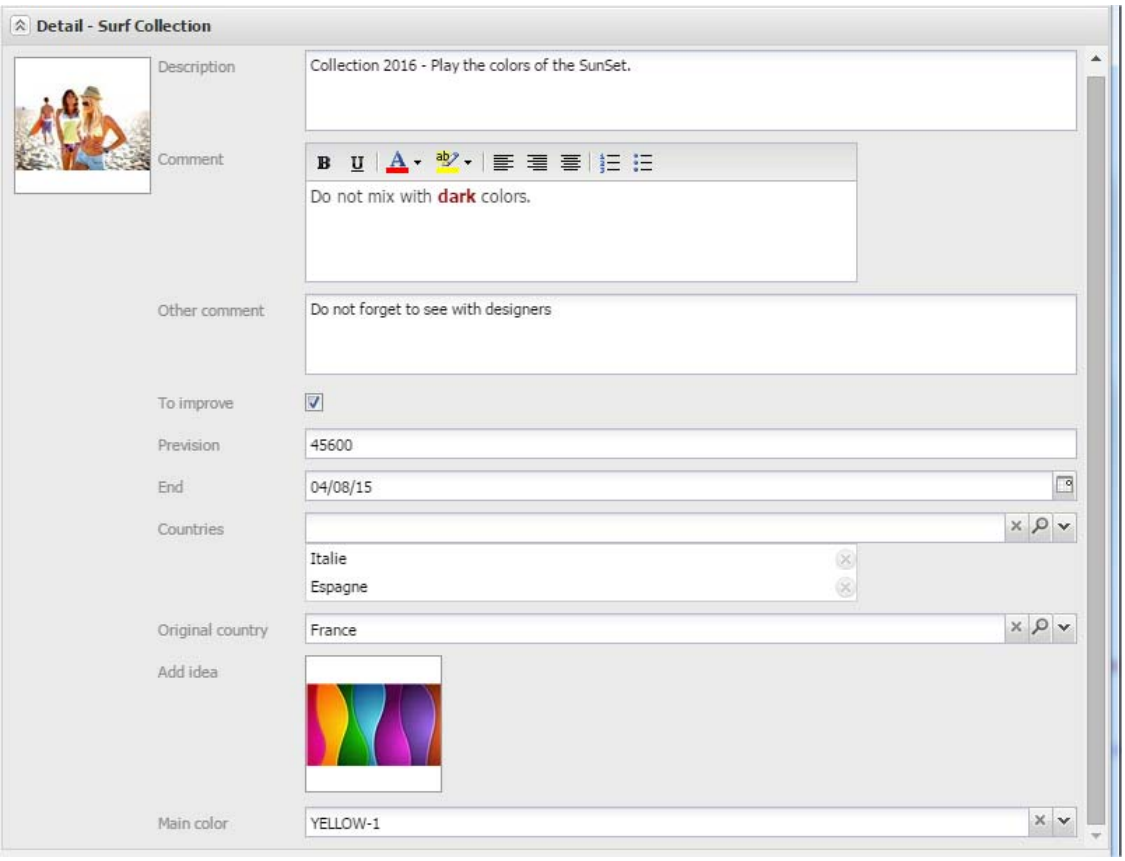
And refresh your browser to see the modification.

Example: a text area, a checkbox, an integer, a date, a multiple combo, a single combo, an image and an envitem combo displayed under comment and description

```
<panel inherits="detailsEdit/detailsEditCenter" overflowX="auto" overflowY="auto" layout="form">
    <!-- See Dictionnary.xml for the available kinds of fields -->
    <textarea name="CP_LongString" fieldLabelI18n="custom.cp.label.longstring" height="100" width="500" margin="10 0 0 10">
        <bind readOnly="{!isPageEditable}" value="{collectionPlan.CP_LongString}" />
    </textarea>
    <checkbox name="CP_Boolean" fieldLabelI18n="custom.cp.label.boolean" margin="10 0 0 10">
        <bind readOnly="{!isPageEditable}" value="{collectionPlan.CP_Boolean}" />
    </checkbox>
    <integerfield name="CP_Int" fieldLabelI18n="custom.cp.label.int" margin="10 0 0 10">
        <bind readOnly="{!isPageEditable}" value="{collectionPlan.CP_Int}" />
    </integerfield>
    <datefield name="CP_Date" fieldLabelI18n="custom.cp.label.date" margin="10 0 0 10" width="250">
        <bind readOnly="{!isPageEditable}" value="{collectionPlan.CP_Date}" />
    </datefield>
    <nodecombo name="CP_Country_Multi" multiSelect="true" propertyName="Country" formatQuery="false"
    fieldLabelI18n="custom.cp.label.hvl.country.multi" width="300" margin="10 0 0 10">
        <bind readOnly="{!isPageEditable}" value="{collectionPlan.CP_Country_Multi}" />
    </nodecombo>
    <nodecombo name="CP_Country_Single" multiSelect="false" displayField="path" propertyName="Country" formatQuery="false"
    fieldLabelI18n="custom.cp.label.hvl.country.single" width="300" margin="10 0 0 10">
        <bind readOnly="{!isPageEditable}" value="{collectionPlan.CP_Country_Single}" />
    </nodecombo>
    <imagefield name="CP_Doc" fieldLabelI18n="custom.cp.label.document" useREST="true"
        inherits="http://lectra.com/pdm/common#defaultImageFieldEXT5" managerRef="selSmallImageField" imageWidth="96" imageHeight="96">
        <managerInstance name="Lectra.PDM.Mgrs.imageFieldRestMgr" />
        <bind readOnly="{!isPageEditable}" fileLinkObj="{collectionPlan.CP_Doc}" />
    </imagefield>
    <envitemcombo name="CP_Color" hiddenName="CP_Color" envItemType="MarketingColor" hideTrigger2="true"
    fieldLabelI18n="custom.cp.label.envitem.color" pageSize="25" >
        <bind readOnly="{!isPageEditable}" value="{collectionPlan.CP_Color}" />
    </envitemcombo>
</panel>
```

Display mode:

Edit mode:



Note the layout has changed: we choose a form layout for this example, with a scrollbar displayed if needed (*overflow* attribute).

5.2.2.7 Notes about fields and roles in this screen (general data part)

Here are different fields and roles (with examples) you can add in this screen, which are in a display or edit mode, depending on the user action (click on edit button):

String field

```
<textfield name="CP_String" fieldLabelI18n="custom.cp.label.string" maxLength="255">
      <bind readOnly="{!isPageEditable}"  value="{collectionPlan.CP_String}"/>
</textfield>
```

Long String field

```
<textarea name="CP_LongString" fieldLabelI18n="custom.cp.label.longstring" >
      <bind readOnly="{!isPageEditable}"  value="{collectionPlan.CP_LongString}"/>
</textarea>
```

Boolean field

```
<checkbox name="CP_Boolean" fieldLabelI18n="custom.cp.label.boolean">
      <bind readOnly="{!isPageEditable}"  value="{collectionPlan.CP_Boolean}"/>
</checkbox>
```

Long field

```
<longfield name="CP_Long" fieldLabelI18n="custom.cp.label.long" >
      <bind readOnly="{!isPageEditable}"  value="{collectionPlan.CP_Long}"/>
</longfield>
```

Integer field

```
<integerfield name="CP_Int" fieldLabelI18n="custom.cp.label.int" >
      <bind readOnly="{!isPageEditable}"  value="{collectionPlan.CP_Int}"/>
</integerfield>
```

Double field

```
<doublefield name="CP_Double" fieldLabelI18n="custom.cp.label.double" >
      <bind readOnly="{!isPageEditable}"  value="{collectionPlan.CP_Double}"/>
</doublefield>
```

Float field

```
<floatfield name="CP_Float" fieldLabelI18n="custom.cp.label.float" >
      <bind readOnly="{!isPageEditable}"  value="{collectionPlan.CP_Float}"/>
</floatfield>
```

Date field

```
<datefield name="CP_Date" fieldLabelI18n="custom.cp.label.date" >
      <bind readOnly="{!isPageEditable}" value="{collectionPlan.CP_Date}"/>
</datefield>
```

HVL Pick list multi field

```
<nodecombo name="CP_Country_Multi" multiSelect="true" propertyName="Country"
fieldLabelI18n="custom.cp.label.hvl.country.multi" formatQuery="false" >
      <bind readOnly="{!isPageEditable}" value="{collectionPlan.CP_Country_Multi}"/>
</nodecombo>
```

HVL Pick list single field

```
<nodecombo name="CP_Country_Single" multiSelect="false" displayField="path"
propertyName="Country" fieldLabelI18n="custom.cp.label.hvl.country.single"
formatQuery="false" >
      <bind readOnly="{!isPageEditable}" value="{collectionPlan.CP_Country_Single}"/>
</nodecombo>
```

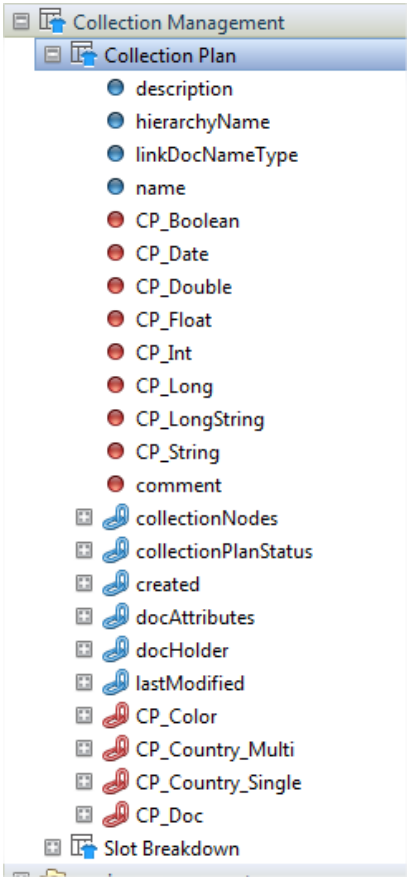
Document target picker

```
<imagefield name="CP_Doc" fieldLabelI18n="custom.cp.label.document"
useREST="true"inherits="http://lectra.com/pdm/common#defaultImageFieldEXT5"
managerRef="selSmallImageField" imageWidth="96" imageHeight="96">
      <managerInstance name="Lectra.PDM.Mgrs.imageFieldRestMgr" />
      <bind readOnly="{!isPageEditable}" fileLinkObj="{collectionPlan.CP_Doc}" />
</imagefield>
```

Envitem field

```
<envitemcombo name="CP_Unit" hiddenName="CP_Unit" envItemType="Unit"
hideTrigger2="true" fieldLabelI18n="custom.cp.label.envitem.unit" pageSize="25"
formatQuery="false">
      <bind readOnly="{!isPageEditable}" value="{collectionPlan.CP_Unit}" />
</envitemcombo>
```

- Note **formatQuery="false"** attribute is needed to allow search on list fields.
- Note *name* value attribute is the same as those created into Administration and Configuration module:



- Note *fieldLabelI18n* value attribute should be defined in PLM-Fashion\PDM\LectraPLMParam\messagesCusto_en.properties file (and other file language if needed; example messagesCusto_fr.properties for French)

5.2.2.8 Adding columns after original columns of slot breakdown grid (grid part)

Add fields and roles you have created in the Administration and Configuration module, in the grid tag:

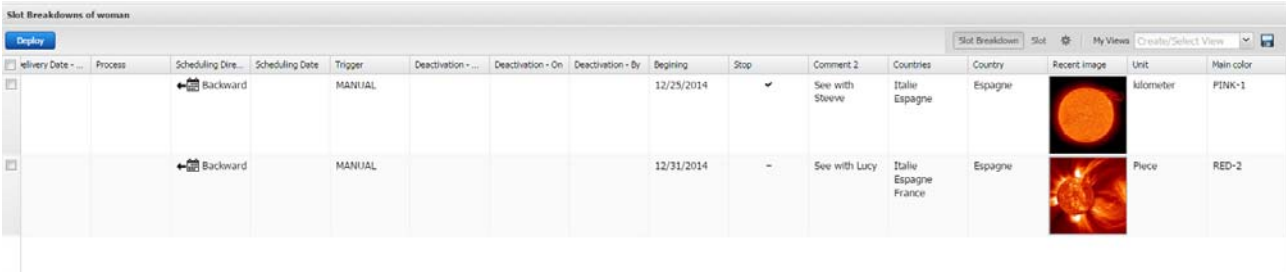
```
<!-- Grid Range Plan Slot Breakdown -->
<grid name="rangeplanslotbreakdowngridDefault" inherits="rangeplanslotbreakdowngrid" >

    <!-- See Dictionnary.xml for the available kinds of fields -->

</grid>
```

And refresh your browser to see the modification.

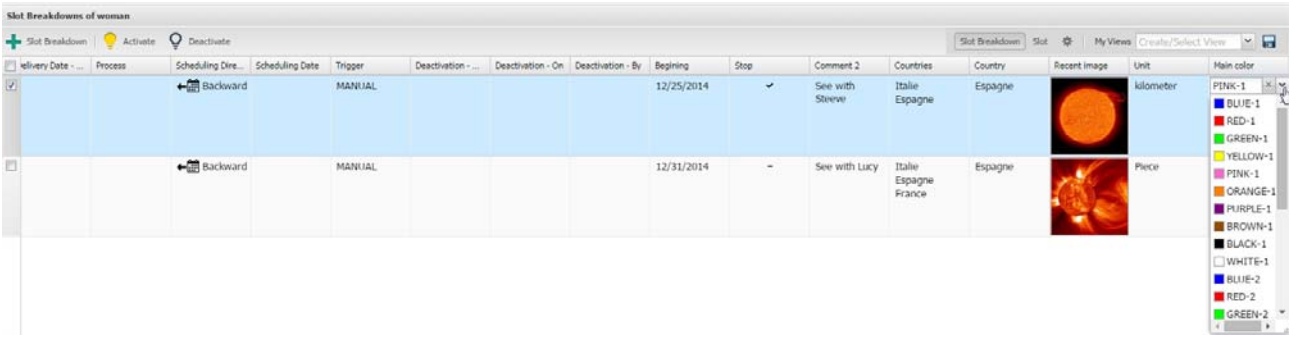
Example: a date column, a checkbox column, a text column, a multiple combo column, a single combo column, an image column, a unit envitem combo column, and a marketing color envitem combo column displayed into the slot breakdown grid.

```
<!-- Grid Range Plan Slot Breakdown -->
<grid name="rangeplanslotbreakdowngridDefault" inherits="rangeplanslotbreakdowngrid">
    <column xtype="datecolumn" dataIndex="CP_RP_Date" headerI18n="custom.cp.rp.label.date" format="${#i18n['i18n.extdateformat']}" editable="true">
        <editorField xsi:type="datefield" format="${#i18n['i18n.extdateformat']}" />
    </column>
    <column xtype="checkboxcolumn" dataIndex="CP_RP_Boolean" headerI18n="custom.cp.rp.label.boolean" headerId="CP_RP_Boolean" editable="true"/>
    <column xtype="textcolumn" dataIndex="CP_RP_String" headerI18n="custom.cp.rp.label.string" headerId="CP_RP_String" editable="true"/>
    <column dataIndex="CP_RP_Country_Multi" headerI18n="custom.cp.rp.label.hvl.country.multi" headerId="CP_RP_Country_Multi" massEditable="false" >
        <editorField xsi:type="nodecombo" multiSelect="true" propertyName="Country" formatQuery="false" />
        <rendererFn name="Lectra.PDM.Format.nodesRenderer()" />
    </column>
    <column dataIndex="CP_RP_Country_Single" headerI18n="custom.cp.rp.label.hvl.country.single" headerId="CP_RP_Country_Single" >
        <editorField xsi:type="nodecombo" multiSelect="false" propertyName="Country" formatQuery="false" />
        <rendererFn name="Lectra.PDM.Format.nodeRenderer()" />
    </column>
    <column xsi:type="imagecolumn" dataIndex="CP_RP_Doc" headerI18n="custom.cp.rp.label.document" headerId="CP_RP_Doc" imageSize="medium"
    useREST="true" editable="true" >
        <imageEditor inherits="http://lectra.com/pdm/common#defaultImageFieldEXT5" managerRef="attachmentImageField">
            <managerInstance name="Lectra.PDM.Mgrs.imageFieldRestMgr" />
        </imageEditor>
     </column>
     <column xsi:type="envitemcolumn" envItemType="Unit" headerI18n="custom.cp.rp.label.envitem.unit" dataIndex="CP_RP_Unit"
     headerId="CP_RP_Unit" sortable="false" editable="true" formatQuery="false">
        <rendererFn name="Lectra.PDM.Rest.Format.otherEnvItemRenderer(false)" />
        <editorConfig>
            <listConfig cls="pdm-collectionplan-rangeplaneditor" resizable="true" />
        </editorConfig>
     </column>
    <column xsi:type="envitemcolumn" envItemType="MarketingColor" headerI18n="custom.cp.rp.label.envitem.color" dataIndex="CP_RP_Color"
    headerId="CP_RP_Color" editable="true" formatQuery="false">
        <rendererFn name="Lectra.PDM.Rest.Format.colorEnvItemRenderer(false)" />
        <editorConfig>
            <listConfig cls="pdm-collectionplan-rangeplaneditor"
                resizable="true" />
            <store pageSize="25">
                <dataField name="name" type="string" />
                <proxy xsi:type="rest" type="rest" url="/marketing-colors">
                    <extraParams restrequest="true" />
                    <reader type="json" />
                </proxy>
            </store>
        </editorConfig>
    </column>
</grid>
```

Display mode:



Edit mode:



5.2.2.9 Notes about fields and roles in this slot breakdown grid (grid part)

Here are different fields and roles (with examples) that you can add in this grid, which are in display or edit mode, depending on the user action (click on edit button):

Date field in grid

```
<column xtype="datecolumn" dataIndex="CP_RP_Date" headerI18n="custom.cp.rp.label.date"
format="${#i18n['i18n.extdateformat']}" editable="true">
      <editorField xsi:type="datefield" format="${#i18n['i18n.extdateformat']}" />
</column>
```

Boolean field in grid

```
<column xtype="checkboxcolumn" dataIndex="CP_RP_Boolean"
headerI18n="custom.cp.rp.label.boolean" headerId="CP_RP_Boolean" editable="true"/>
```

String field in grid

```
<column xtype="textcolumn" dataIndex="CP_RP_String"
headerI18n="custom.cp.rp.label.string" headerId="CP_RP_String" editable="true"/>
```

HVL pick list in grid: example with Country and Multiselect

```
<column dataIndex="CP_RP_Country_Multi" headerI18n="custom.cp.rp.label.hvl.multi"
headerId="CP_RP_Country_Multi" massEditable="false" >
      <editorField xsi:type="nodecombo" multiSelect="true" propertyName="Country"
formatQuery="false" />
      <rendererFn name="Lectra.PDM.Format.nodesRenderer()" />
</column>
```

HVL pick list in grid: example with Country and Singleselect

```
<column dataIndex="CP_RP_Country_Single"
headerI18n="custom.cp.rp.label.hvl.country.single" headerId="CP_RP_Country_Single" >
      <editorField xsi:type="nodecombo" multiSelect="false" propertyName="Country"
formatQuery="false" />
      <rendererFn name="Lectra.PDM.Format.nodeRenderer()" />
</column>
```

Document target in grid

```
<column xsi:type="imagecolumn" dataIndex="CP_RP_Doc"
headerI18n="custom.cp.rp.label.document" headerId="CP_RP_Doc" imageSize="medium"
useREST="true" editable="true" >
      <imageEditor inherits="http://lectra.com/pdm/common#defaultImageFieldEXT5">
            <managerInstance name="Lectra.PDM.Mgrs.imageFieldRestMgr" />
      </imageEditor>
</column>
```

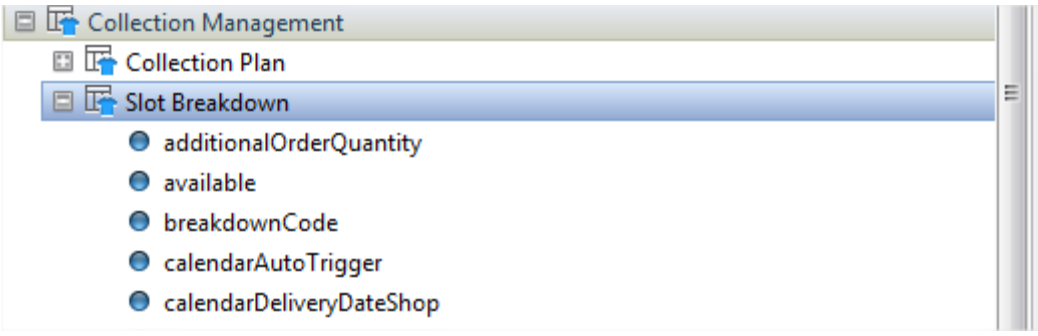
EnvItem (multiselect is not allowed) in grid: example with Unit

```
<column headerI18n="custom.cp.rp.label.envitem.unit" xsi:type="envitemcolumn"
envItemType="Unit" dataIndex="CP_RP_Unit" headerId="CP_RP_Unit" sortable="false"
editable="true" formatQuery="false">
       <rendererFn name="Lectra.PDM.Rest.Format.otherEnvItemRenderer(false)" />
       <editorConfig>
            <listConfig cls="pdm-collectionplan-rangeplaneditor" resizable="true" />
       </editorConfig>
</column>
```

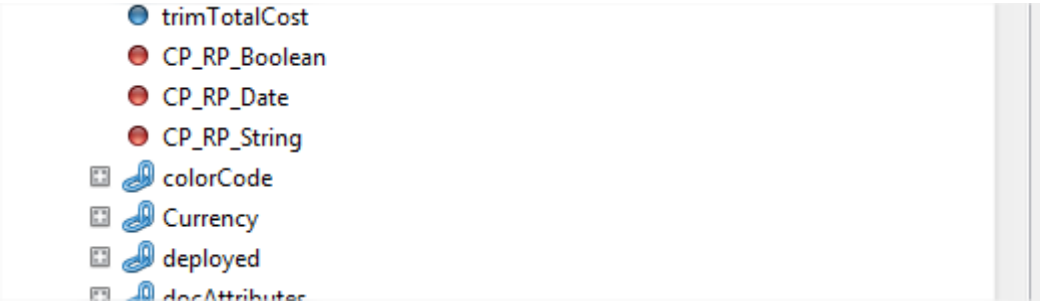
EnvItem (multiselect is not allowed) in grid for Marketing Color only

```
<column headerI18n="custom.cp.rp.label.envitem.color" xsi:type="envitemcolumn"
envItemType="MarketingColor" dataIndex="CP_RP_Color" headerId="CP_RP_Color"
editable="true" formatQuery="false">
      <rendererFn name="Lectra.PDM.Rest.Format.colorEnvItemRenderer(false)" />
      <editorConfig>
            <listConfig cls="pdm-collectionplan-rangeplaneditor"
                        resizable="true" />
            <store pageSize="25">
                  <dataField name="name" type="string" />
                  <proxy xsi:type="rest" type="rest" url="/marketing-colors">
                        <extraParams restrequest="true" />
                        <reader type="json" />
                  </proxy>
            </store>
      </editorConfig>
</column>
```

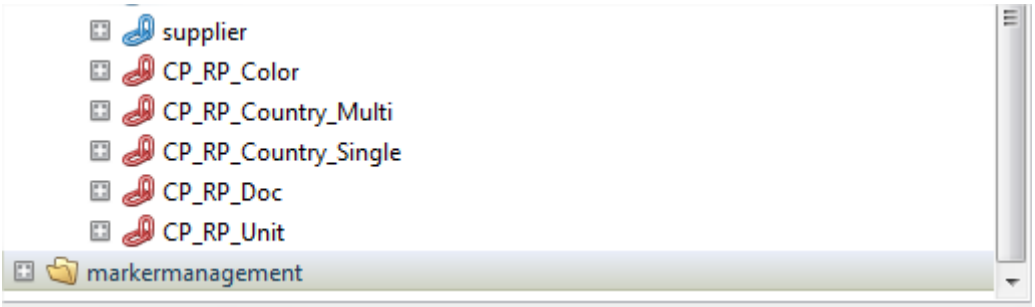
- Note **formatQuery=”false”** attribute is needed to allow search on list fields.
- Note *dataIndex* value attribute is the same as those created in the Administration and Configuration module:



...



...



- Note *headerI18n* value attribute should be defined in PLM-Fashion\PDM\LectraPLMParam\messagesCusto_en.properties file (and other file language if needed; example messagesCusto_fr.properties for French)
- Note *marketingColor envitem* has a store based on `/marketing-colors` url; this is needed to display color thumbnails.

6. ADVANCED CONFIGURATION FOR ADDING FIELDS AND PANELS

6.1 Working with inheritance

Instead of copying entire panel descriptions of the base XML file into the «custom» folder of the XML file, it is possible, if you are confident with the XML description, to use the concept of inheritance to describe its custom description.

Inheritance is defined in 2.2 - Inheritance.

Any graphic element declaration such as the <panel> may be spread to other definitions while keeping the default definition that exists in the file.

If the inheritance according to **#type** or the **#topCategoryName** are already defined or do not meet your needs, it is possible to extend the inheritance for the definition of a new panel be taken into account as a priority.

6.2 Specific example

The panel description inherits 3 definitions of possible panels description:

```
<panel name="description"
inherits="description_${#type},description_${#topCategoryName},description" />
```

If the panel that you want to see applied is different from **<panel name="description_tous_les_types">**, different from **<panel name="description_Styles, Fabrics, Trims, PackagingLabel">**:

- the <panel name= « MaDescription_Custom »> can be created
- the new inheritance can be added to the declaration of the panel description above.

```
<panel name="description"
inherits="MaDescription_Custom",description_${#type},description_${#topCategory
Name},description" />
```

In first position, it will be looked for first, before the description by **type** and **topCategoryName**.

No need to duplicate the XML code: just add the code for

```
<panel name="MaDescription_Custom" …/… />
```

Constraint:

When inheriting Explorer's definition, it is necessary to define the following **<contextparameter>**:

```
<panel name="Explorer_CustomStyle" inherits="Explorer">
      <contextparameter name="explorerTitle" value="model.Style" />
      <contextparameter name="categoryName" value="CustomStyle" />
      <contextparameter name="topCategoryName" value="Style" />
      <contextparameter name="explorerName" value="Style" />
</panel>
```

7. LAYOUT MODIFICATION

Configuring screens can also be done by modifying the screens' presentation.

This can be done by modifying the used layouts, but it implies some changes in the XML screens definitions and to do some advanced XML grammar.

Graphic elements that are used in our XML grammar come from the ExtJS library. Lectra has also developed the LPFExt framework that defines other graphic elements that inherit those from ExtJS.

In the next paragraphs, some elements from the ExtJS library or from the LPFExt framework will be mentioned.

7.1 Layout definition

The layout defines the way to organize the different data blocks in the screen: in columns, in tables, in grid…

Follow the link to see some **Sample layouts**.

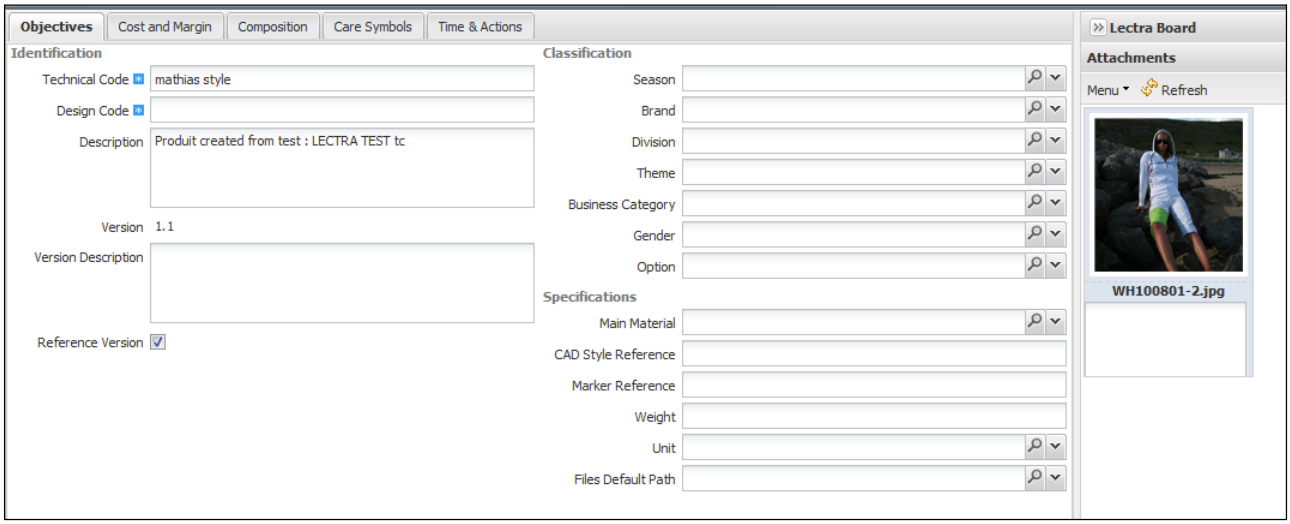
7.2 Only the ExtJSErreur ! Source du renvoi introuvable.have a «layout»

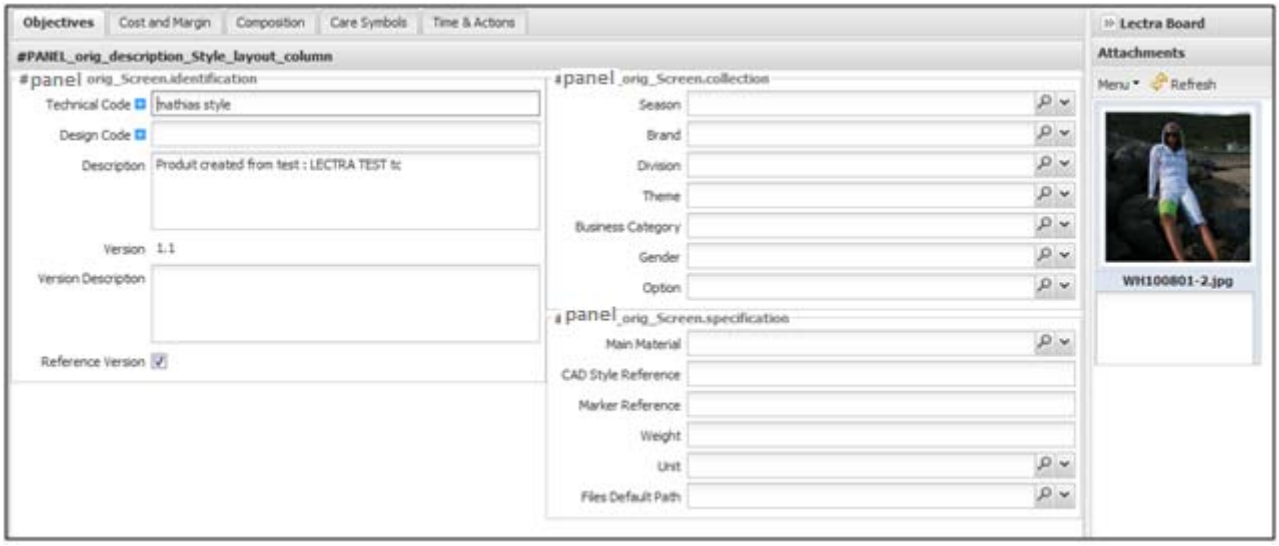
Containers that are used in the studied screens (in previous paragraph) are the «panels», «tabpanels», «gridpanels». To get a full list of the containers, please see the Appendix **Erreur ! Source du renvoi introuvable.** - **Erreur ! Source du renvoi introuvable.**.

7.2.1 The Panel column layout and the GO screen panels of a Style

Already mentioned and handled in previous paragraphs, panels are data blocks commonly used in screens.

The modification of panels is illustrated in the 2 screenshots of the «**style**»'s **General Objectives** screen. Visualization of panels is therefore easier.





The **Objectives** tab Panel is defined with the **column** layout,

```
<panel name="description_Style" titleI18n="PANEL orig description_Style
layout column" layout="column" autoScroll="true" border="true">
```

… and the 3 panels it is composed of are defined with the `columnWidth="0.5"` attribute, which means that they each occupy 50% of the main panel's width.

```
  <panel name="identification" titleI18n="fieldset orig
Screen.identification" columnWidth="0.5" border="true">
```

```
<panel name="collection" titleI18n="fieldset orig Screen.collection"
columnWidth="0.5" border="true">
<panel name="collection" titleI18n="fieldset orig Screen.collection"
columnWidth="0.5" border="true">
```

In the Product Development module, the most commonly used layout attributes are margin and width.

APPENDICES

A. ExtJS 3.4 and ExtJS 5.0.1

PDM V5R1 use ExtJS 3.4 except for (Collection Plan screens, ValidationTable, Product Compositions Tab, Product CareLabel Tab) which are based on ExtJS 6.0.2.

The LPFExt abstraction layer permits it to be partially independent of the ExtJS release used.

In a LPFExt xml description file, tag's attributes are interpreted either by LPFExt or ExtJS.

Concerning ExtJS attributes, some are only compatible with ExtJS 3.4, and others compatible with ExtJS 3.4 and 6.0.2.

Example :

```
      <displayfield fieldLabelI18n="objective" topLabelCls=" " labelAlign="top"
labelSeparator=":" bind="{objectives.processDevelopmentType}" />
```

- fieldLabelI18n : interpreted by LPFExt layer
- labelAlign and labelSeparator : interpreted by ExtJS 3.4 or ExtJS 6.
- topLabelCls : interpreted by ExtJS 6.

With the following instruction

```
      <displayfield fieldLabelI18n="objective" topLabelCls=" " labelAlign="top"
labelSeparator=":" bind="{objectives.processDevelopmentType}" />
```

Is totally interpreted by a Collection Plan screen (ExtJS 5 mode).

In other screens (ExtJS 3.4 mode), the topLabelCls attribute is ignored.

B. Switch between ExtJS 3.4 and ExtJS 6.0.2

The PDM manages the cohabitation between ExtJS 3.4 and ExtJS 6. According the url processed by the server, the system switches between ExtJS 6 or ExtJS 3.4 mode:

- Collection Plan url → Ext 6 mod.
- Other URL: Ext 3 mod

If you want explicitly use ExtJS 5 and higher library on an ExtJS 3.4 mod, you have to suffix the panel name by EXT5.

Example:

```
<!-- BEGIN WIDGET EXT5 DEV MODE -->
<panel name="detail_Panel_PickerEXT5">
   <managerInstance name="Lectra.PDM.Mgrs.aiPickerMgr"/>
      <graphicobjectsviewer name="localgr" width="300" itemId="bdetail"
titleI18n="action.filelink.info" managerRef="boardAIDetail"
inherits="http://lectra.com/pdm/picker#graphicContainer">
      <managerInstance name="Lectra.PDM.Mgrs.aiPickerMgr"/>
      </graphicobjectsviewer>
   <panel itemId="other" managerRef="emptyDetail" border="false"/>
</panel>
```

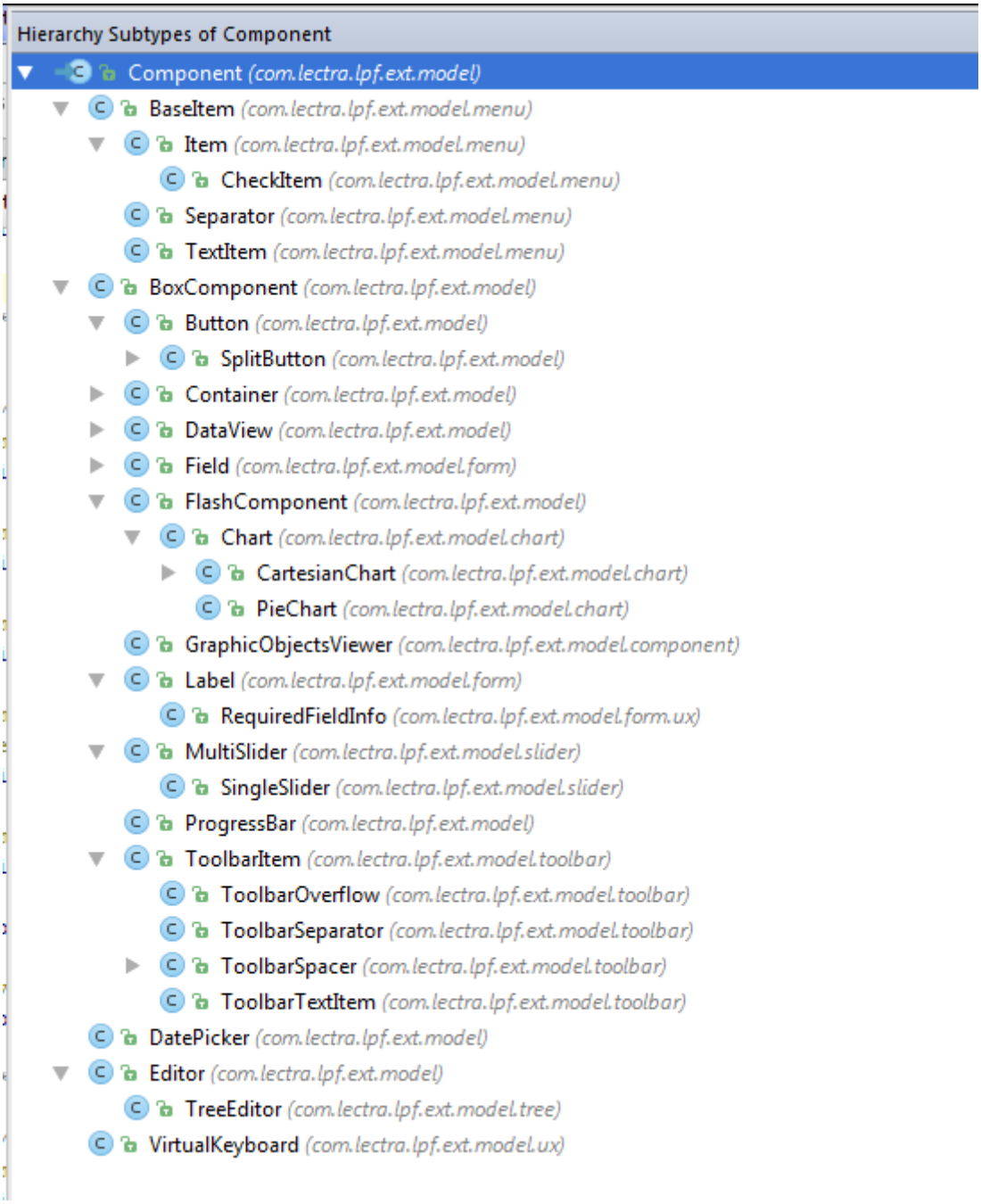
C. List of LPFExt widgets

A LPFExt widget is a graphical component which could be created using the LPFExt xml syntax.

Example of ExplorerGrid widget:

```
<explorerGrid modePick="false"
inherits="http://lectra.com/pdm/search/product#searchGrid">
   <title>${#i18n['search.result']}</title>
</explorerGrid>
```

The list of LPFExt widgets is the following:



Details of Containers subcomponents:

Hierarchy Subtypes of Container

- Container *(com.lectra.lpf.ext.model)*
  - FieldContainer *(com.lectra.lpf.ext.model.form)*
    - MultiSelectExt5 *(com.lectra.lpf.ext.model.form)*
  - GroupComponent *(com.lectra.lpf.ext.model)*
  - Menu *(com.lectra.lpf.ext.model.menu)*
    - StoreMenu *(com.lectra.lpf.ext.model.menu.ux)*
  - Panel *(com.lectra.lpf.ext.model)*
    - BasicForm *(com.lectra.lpf.ext.model.form)*
      - FormPanel *(com.lectra.lpf.ext.model.form)*
    - ButtonGroup *(com.lectra.lpf.ext.model)*
    - ButtonPanel *(com.lectra.lpf.ext.model)*
    - Calendar *(com.lectra.lpf.ext.model.component)*
    - CodeMirror *(com.lectra.lpf.ext.model.ux)*
    - DockAccordionPanel *(com.lectra.lpf.ext.model.dock)*
    - Explorer *(com.lectra.lpf.ext.model.component)*
    - ExplorerFormButtons *(com.lectra.lpf.ext.model.component)*
    - ExplorerViews *(com.lectra.lpf.ext.model.component)*
    - FieldSet *(com.lectra.lpf.ext.model.form)*
    - GridPanel *(com.lectra.lpf.ext.model.grid)*
    - IframePanel *(com.lectra.lpf.ext.model.component)*
    - IndicatorsPanel *(com.lectra.lpf.ext.model.component)*
    - MasterDetailPanel *(com.lectra.lpf.ext.model.ux)*
    - MessagePanel *(com.lectra.lpf.ext.model.component)*
    - Portal *(com.lectra.lpf.ext.model.portal)*
    - Portlet *(com.lectra.lpf.ext.model.portal)*
    - TabPanel *(com.lectra.lpf.ext.model)*
    - Tip *(com.lectra.lpf.ext.model.tip)*
    - TreePanel *(com.lectra.lpf.ext.model.tree)*
    - UnityPanel *(com.lectra.lpf.ext.model.component)*
    - Window *(com.lectra.lpf.ext.model)*
    - WindowBlind *(com.lectra.lpf.ext.model.ux)*
  - PanelHeader *(com.lectra.lpf.ext.model)*
  - PortalColumn *(com.lectra.lpf.ext.model.portal)*
  - Toolbar *(com.lectra.lpf.ext.model.toolbar)*
    - Paging *(com.lectra.lpf.ext.model.toolbar)*
    - PagingToolbar *(com.lectra.lpf.ext.model.toolbar)*
    - StatusBar *(com.lectra.lpf.ext.model.toolbar)*
  - ViewPort *(com.lectra.lpf.ext.model)*

Details of Panel subcomponents:

Hierarchy Subtypes of Panel

- Panel *(com.lectra.lpf.ext.model)*
  - BasicForm *(com.lectra.lpf.ext.model.form)*
    - FormPanel *(com.lectra.lpf.ext.model.form)*
      - ExplorerForm *(com.lectra.lpf.ext.model.component)*
      - PopForm *(com.lectra.lpf.ext.model.component)*
  - ButtonGroup *(com.lectra.lpf.ext.model)*
  - ButtonPanel *(com.lectra.lpf.ext.model)*
  - Calendar *(com.lectra.lpf.ext.model.component)*
  - CodeMirror *(com.lectra.lpf.ext.model.ux)*
  - DockAccordionPanel *(com.lectra.lpf.ext.model.dock)*
  - Explorer *(com.lectra.lpf.ext.model.component)*
  - ExplorerFormButtons *(com.lectra.lpf.ext.model.component)*
  - ExplorerViews *(com.lectra.lpf.ext.model.component)*
  - FieldSet *(com.lectra.lpf.ext.model.form)*
  - GridPanel *(com.lectra.lpf.ext.model.grid)*
    - EditorGridPanel *(com.lectra.lpf.ext.model.grid)*
      - ActionsEditorGridPanel *(com.lectra.lpf.ext.model.grid)*
      - ExplorerEditableGrid *(com.lectra.lpf.ext.model.component)*
      - ExplorerGrid *(com.lectra.lpf.ext.model.component)*
      - LockingEditorGridPanel *(com.lectra.lpf.ext.model.grid)*
      - PropertyGrid *(com.lectra.lpf.ext.model.grid)*
    - ExplorerSelection *(com.lectra.lpf.ext.model.component)*
    - LockingGridPanel *(com.lectra.lpf.ext.model.grid)*
    - PivotGrid *(com.lectra.lpf.ext.model.grid)*
      - PivotGridExtended *(com.lectra.lpf.ext.model.grid)*
  - IframePanel *(com.lectra.lpf.ext.model.component)*
  - IndicatorsPanel *(com.lectra.lpf.ext.model.component)*
  - MasterDetailPanel *(com.lectra.lpf.ext.model.ux)*
  - MessagePanel *(com.lectra.lpf.ext.model.component)*
  - Portal *(com.lectra.lpf.ext.model.portal)*
  - Portlet *(com.lectra.lpf.ext.model.portal)*
  - TabPanel *(com.lectra.lpf.ext.model)*
    - DockTabPanel *(com.lectra.lpf.ext.model.dock)*
    - ExplorerTabs *(com.lectra.lpf.ext.model.component)*
  - Tip *(com.lectra.lpf.ext.model.tip)*
    - ToolTip *(com.lectra.lpf.ext.model.tip)*
      - QuickTip *(com.lectra.lpf.ext.model.tip)*
  - TreePanel *(com.lectra.lpf.ext.model.tree)*
    - ColumnTree *(com.lectra.lpf.ext.model.tree.ux)*
    - RemoteTreePanel *(com.lectra.lpf.ext.model.tree.ux)*
    - TreeGrid *(com.lectra.lpf.ext.model.tree.ux)*
  - UnityPanel *(com.lectra.lpf.ext.model.component)*
  - Window *(com.lectra.lpf.ext.model)*
    - DockWindow *(com.lectra.lpf.ext.model.dock)*
  - WindowBlind *(com.lectra.lpf.ext.model.ux)*

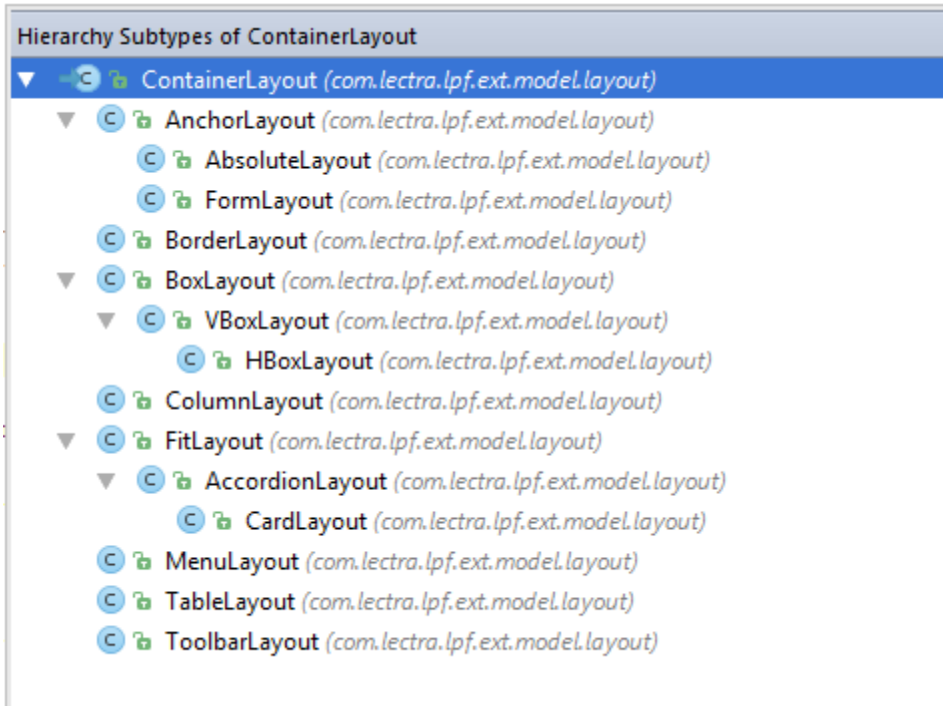
Details of Field subcomponents:

Hierarchy Subtypes of Field

- Field (com.lectra.lpf.ext.model.form)
  - ButtonField (com.lectra.lpf.ext.model.form.ux)
  - Checkbox (com.lectra.lpf.ext.model.form)
    - Radio (com.lectra.lpf.ext.model.form)
    - XCheckbox (com.lectra.lpf.ext.model.form.ux)
  - CheckboxGroup (com.lectra.lpf.ext.model.form)
    - RadioGroup (com.lectra.lpf.ext.model.form)
  - CompositeField (com.lectra.lpf.ext.model.form)
  - DisplayField (com.lectra.lpf.ext.model.form.ux)
  - Hidden (com.lectra.lpf.ext.model.form)
  - HtmlEditor (com.lectra.lpf.ext.model.form)
  - MultiSelect (com.lectra.lpf.ext.model.form)
  - SliderField (com.lectra.lpf.ext.model.form)
  - StaticTextField (com.lectra.lpf.ext.model.form.ux)
  - TextField (com.lectra.lpf.ext.model.form)
    - NumberField (com.lectra.lpf.ext.model.form)
      - DoubleField (com.lectra.lpf.ext.model.form)
      - FloatField (com.lectra.lpf.ext.model.form)
      - IntegerField (com.lectra.lpf.ext.model.form)
      - LongField (com.lectra.lpf.ext.model.form)
      - SpinnerField (com.lectra.lpf.ext.model.form.ux)
    - PickerField (com.lectra.lpf.ext.model.form)
      - TreePickerField (com.lectra.lpf.ext.model.form)
    - TextArea (com.lectra.lpf.ext.model.form)
    - TriggerField (com.lectra.lpf.ext.model.form)
      - ComboBox (com.lectra.lpf.ext.model.form)
        - LovCombo (com.lectra.lpf.ext.model.form.ux)
        - SuperBoxSelect (com.lectra.lpf.ext.model.form.ux)
        - TimeField (com.lectra.lpf.ext.model.form)
        - TwinComboBox (com.lectra.lpf.ext.model.form.ux)
        - TwinTriggerCombo (com.lectra.lpf.ext.model.form)
      - DateField (com.lectra.lpf.ext.model.form)
      - LovField (com.lectra.lpf.ext.model.form.ux)
      - TwinTriggerField (com.lectra.lpf.ext.model.form)
        - AbstractTwinTriggerField (com.lectra.lpf.ext.model.form)
          - ComboWithSelection (com.lectra.lpf.ext.model.form)
            - GridCombo (com.lectra.lpf.ext.model.form)
            - TreeCombo (com.lectra.lpf.ext.model.form)
        - ColorPickerFieldPlus (com.lectra.lpf.ext.model.form.ux)
        - SearchField (com.lectra.lpf.ext.model.form)
  - TreeField (com.lectra.lpf.ext.model.tree.ux)

Some widgets correspond exactly to ExtJS widget (like a Button), and others do not (like TreePickerField).

D. The LPFExt "layouts"



E. The different column types of an LPFExt grid

Hierarchy Subtypes of GridColumn

- GridColumn *(com.lectra.lpf.ext.model.grid)*
  - ActionColumn *(com.lectra.lpf.ext.model.grid)*
  - CellActionsGridColumn *(com.lectra.lpf.ext.model.grid)*
  - CheckColumn *(com.lectra.lpf.ext.model.grid)*
    - GridStepsCheckBoxColumn *(com.lectra.lpf.ext.model.grid)*
  - GridArrayColumn *(com.lectra.lpf.ext.model.grid)*
    - GridStepsArrayColumn *(com.lectra.lpf.ext.model.grid)*
  - GridBooleanColumn *(com.lectra.lpf.ext.model.grid)*
  - GridConstraintDateColumn *(com.lectra.lpf.ext.model.grid)*
  - GridCurrentLcsColumn *(com.lectra.lpf.ext.model.grid)*
  - GridDateColumn *(com.lectra.lpf.ext.model.grid)*
    - GridStepsDateColumn *(com.lectra.lpf.ext.model.grid)*
    - GridWorkflowDateColumn *(com.lectra.lpf.ext.model.grid)*
  - GridGroupColumn *(com.lectra.lpf.ext.model.grid)*
  - GridLongTextColumn *(com.lectra.lpf.ext.model.grid)*
  - GridMatrixColumn *(com.lectra.lpf.ext.model.grid)*
  - GridNumberColumn *(com.lectra.lpf.ext.model.grid)*
  - GridPreviousStepColumn *(com.lectra.lpf.ext.model.grid)*
  - GridProcessTemplateColumn *(com.lectra.lpf.ext.model.grid)*
  - GridResourcesColumn *(com.lectra.lpf.ext.model.grid)*
  - GridStepColumn *(com.lectra.lpf.ext.model.grid)*
  - GridStepsColumn *(com.lectra.lpf.ext.model.grid)*
  - GridStepsSummaryColumn *(com.lectra.lpf.ext.model.grid)*
  - GridStepStatusColumn *(com.lectra.lpf.ext.model.grid)*
  - GridTemplateColumn *(com.lectra.lpf.ext.model.grid)*
  - GridTextColumn *(com.lectra.lpf.ext.model.grid)*
  - GridWorkFlowColumn *(com.lectra.lpf.ext.model.grid)*
    - GridWorkflowTextColumn *(com.lectra.lpf.ext.model.grid)*
  - RowNumberer *(com.lectra.lpf.ext.model.grid)*
  - TreeColumn *(com.lectra.lpf.ext.model.grid)*
  - TreeGridColumn *(com.lectra.lpf.ext.model.tree.ux)*

F. Complete ExtJS documentation

- For ExtJS 3.4**:** http://docs.sencha.com/ext-js/3-4/#!/api
- For ExtJS 5.0.1**:** http://docs-origin.sencha.com/extjs/5.0/5.0.1-apidocs/

and examples

- For ExtJS 3.4: http://dev.sencha.com/deploy/ext-3.4.0/examples/
- For ExtJS 6.0.2: http://examples.sencha.com/extjs/6.0.2/examples/

G. LPFExt widget documentations

Two case to consider:

1. LPFExt widgets which match a ExtJS component
2. LPFExt widgets which don't match a ExtJS component

Case 1:

Look at the documentation of the corresponding ExtJS widget.

Example**:** the documentation of the LPFExt Button widget,

- For ExtJS 6 : http://docs.sencha.com/extjs/6.0/6.0.2-classic/#!/api/Ext.button.Button
- For ExtJS 3.4 : http://docs.sencha.com/extjs/3.4.0/#!/api/Ext.Button

Case 2:

Look at the ExtJS documentation of the LPFExt parent component which matches an ExtJS component.

Example: documentation of the LPFExt FloatField widget.

FloatField has no ExtJS equivalent. The parent of the FloatField is NumberField, which has an ExtJS equivalent, so look at the NumberField ExtJS documentation.

- For ExtJS 5.0.1:
  - http://docs.sencha.com/extjs/6.0/6.0.2-classic/#!/api/Ext.form.field.Number
- For ExtJS 3.4:
  - http://docs.sencha.com/extjs/3.4.0/#!/api/Ext.form.NumberField

To finish, you can look inside the screen's xml file to see how a widget is configured. There are many examples of its use.

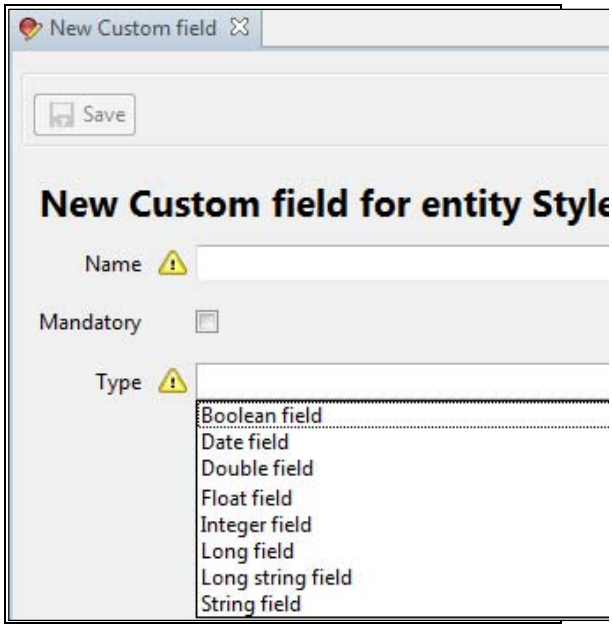
H. LPFExt Directory location Change

In PLM V5R1, the location of the LPFExt files is under the lpf/ext3/namespaces directory.

I. Reminder of the Administration and Configuration Application

To fill the new fields that are to be added to a screen with values, it is necessary to create corresponding values in the Database.

Fields of the following type may be added:

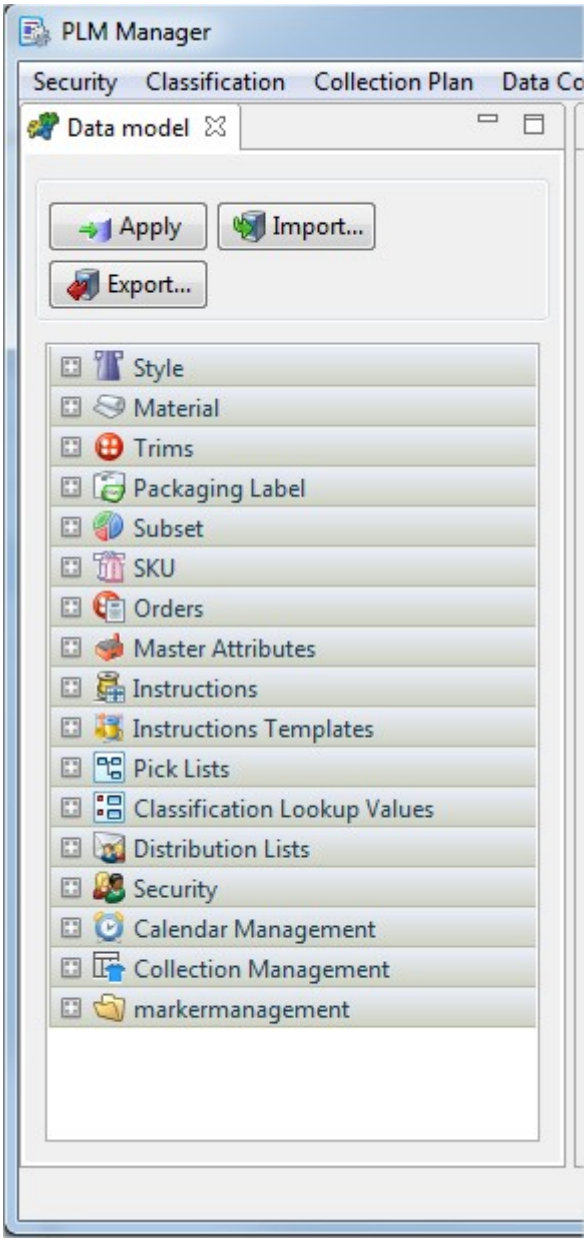


For example: the «MyCustomStringField» on a «Style» must trigger the creation of the «MyCustomStringField» data of the «String Field» type in the Data base.

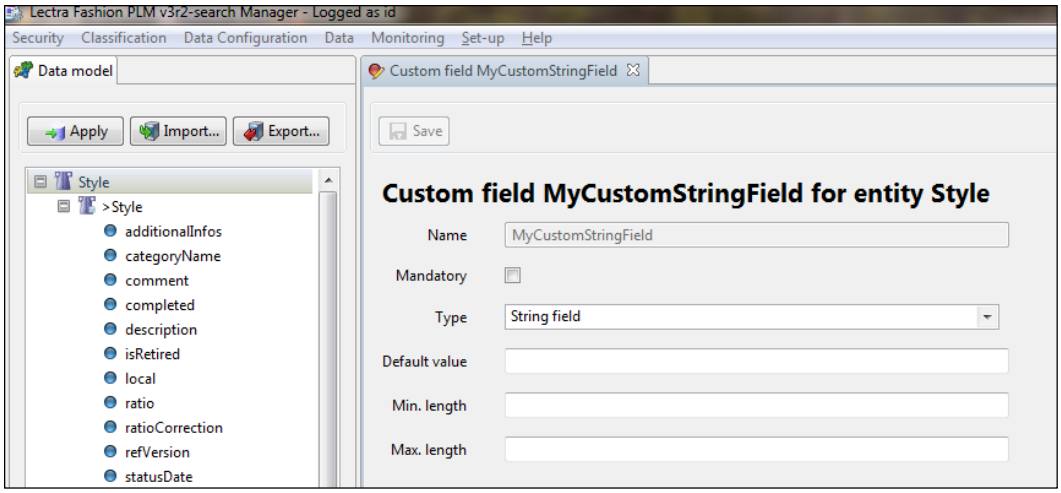
To do this:

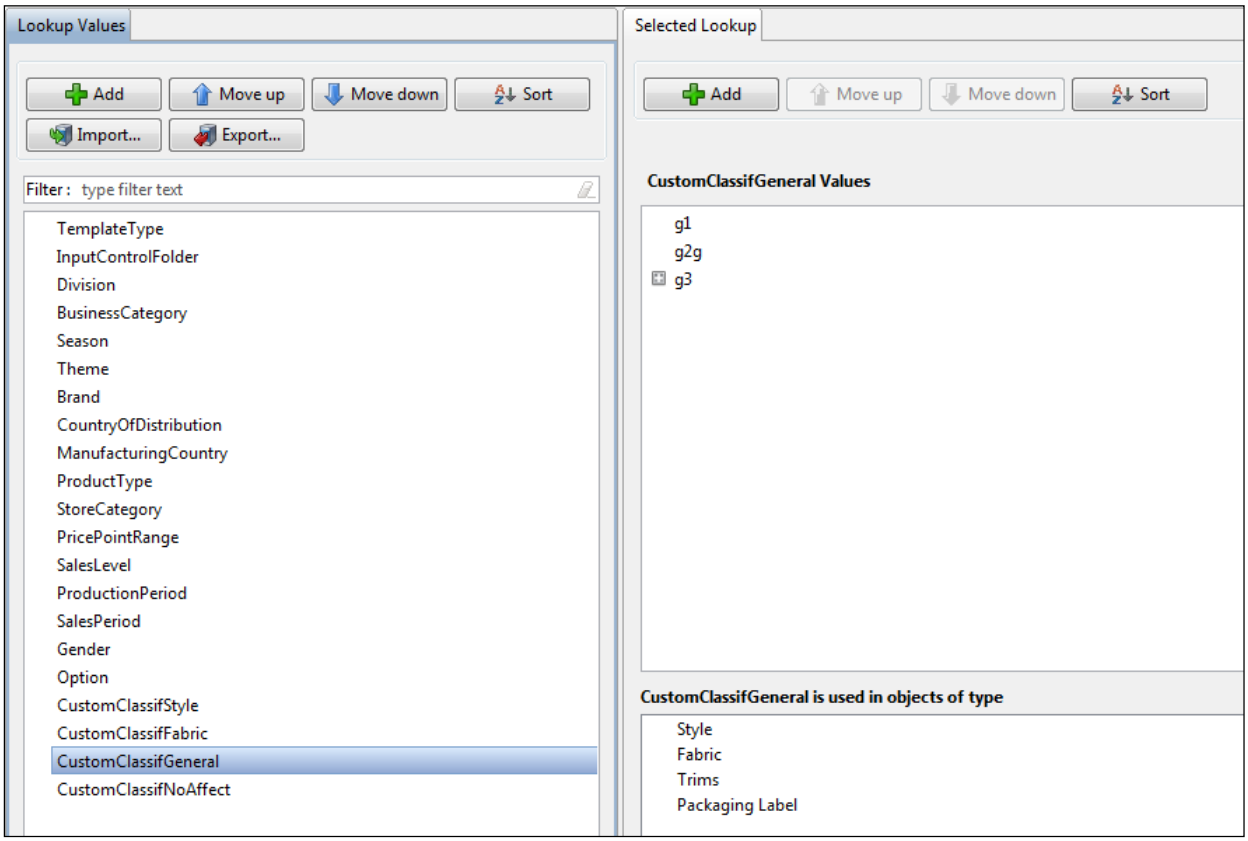
1. Open the Administration and Configuration module

   Data Configuration / Extend Data Model menu to get the panel that describes all the data:



2. Right-click on **Style**.
3. In the context menu, click on **Add Custom Field**.
4. Name the **MyCustomStringField** value.
5. Select its type: **String Field**.
6. Click on **Save** and **apply the config for all subcategories** if you need to extend the definition to the sub-categories.


Lectra Fashion PLM v3r2-search Manager - Logged as id
Security
Classification
Data Configuration
Data
Monitoring
Set-up
Help
Data model
Apply
Import...
Export...
Style
>Style
additionalInfos
categoryName
comment
completed
description
isRetired
local
ratio
ratioCorrection
refVersion
statusDate
Custom field MyCustomStringField
Save
Custom field MyCustomStringField for entity Style
Name
MyCustomStringField
Mandatory
Type
String field
Default value
Min. length
Max. length


Lookup Values
Add
Move up
Move down
Sort
Import...
Export...
Filter : type filter text
TemplateType
InputControlFolder
Division
BusinessCategory
Season
Theme
Brand
CountryOfDistribution
ManufacturingCountry
ProductType
StoreCategory
PricePointRange
SalesLevel
ProductionPeriod
SalesPeriod
Gender
Option
CustomClassifStyle
CustomClassifFabric
CustomClassifGeneral
CustomClassifNoAffect
Selected Lookup
Add
Move up
Move down
Sort
CustomClassifGeneral Values
g1
g2g
g3
CustomClassifGeneral is used in objects of type
Style
Fabric
Trims
Packaging Label

J. Adding other types of simple fields in the xml

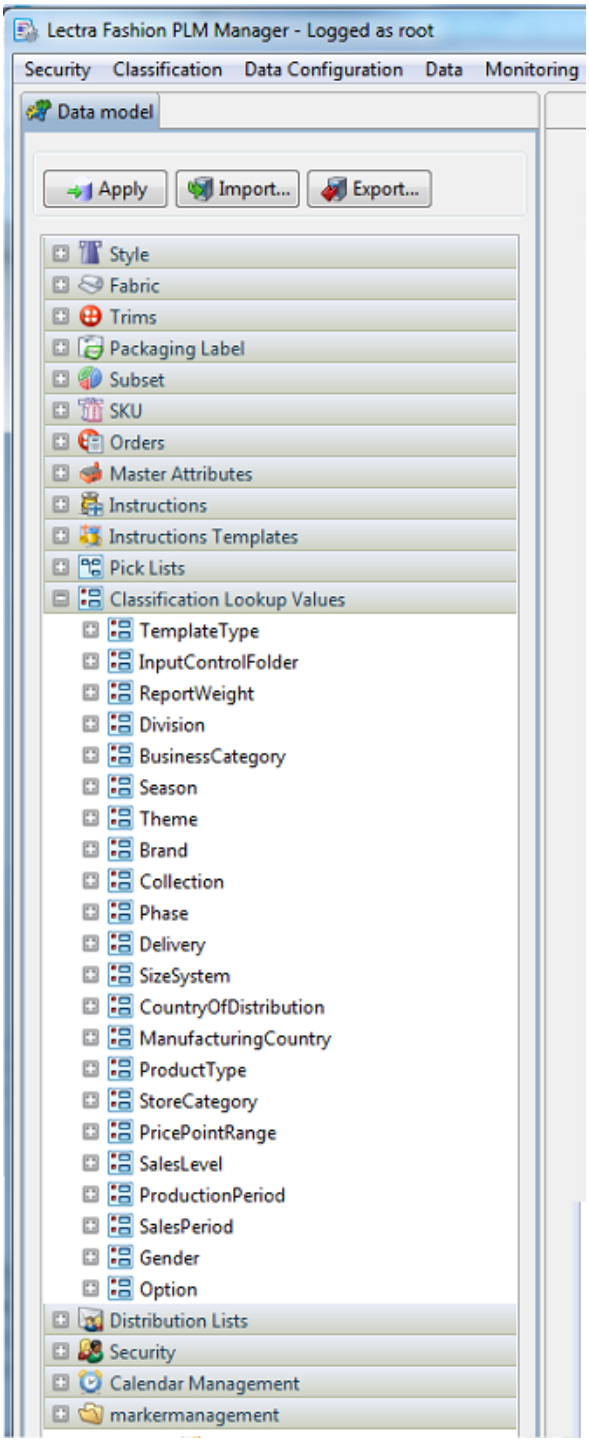
All the different field types definable in the Administration and Configuration module have a corresponding definition in the XML file, according to the following associations:

Boolean field	Twincombo / checkbox
Date field	datefield
Double field	doublefield
Float field	floatfield
Integer field	integerfield
Long field	longfield
Long string field	textarea
String field	textfield

The insertion of the different field types in the xml description file must be done in the namespace/custom/Lectra.PDM.User.xml file.

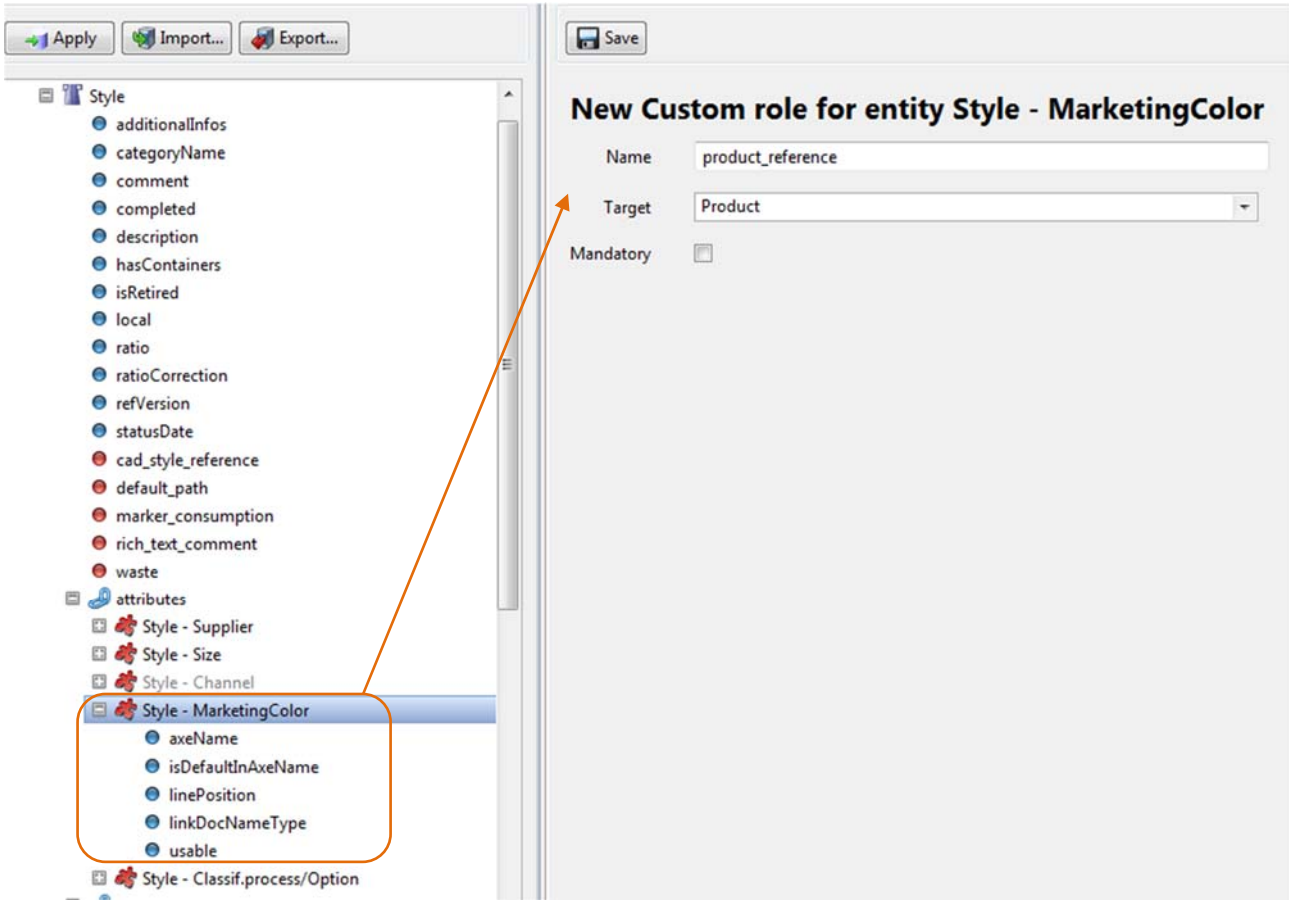
K. Adding «Pick Lists» and «Classification Lookup Values»

By looking at the data Model in the Administration and Configuration module, you can see what has been created:



L. Adding «Product Picker» on product attributes

By looking at the data Model in the Administration and Configuration module, you can create custom role referring to product data model on one of your product attributes:



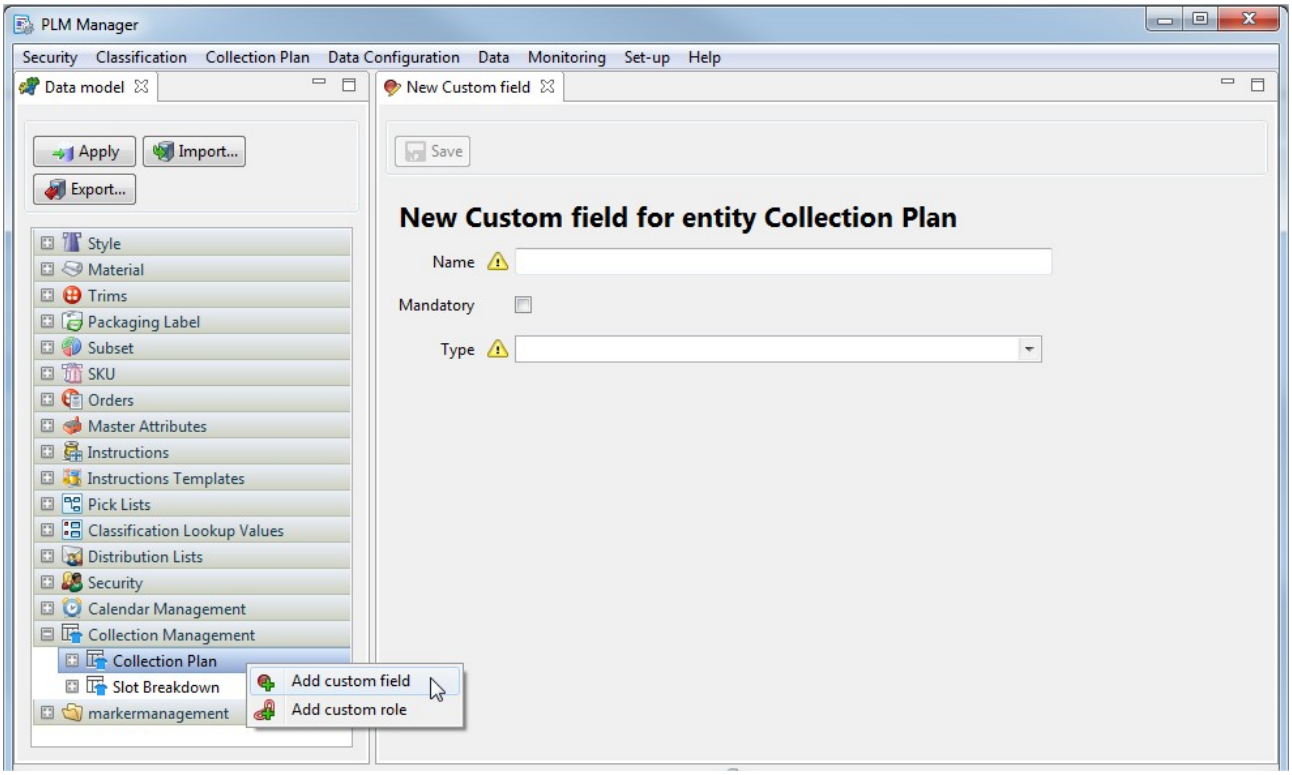
Changes your screensCusto.xml :

```
<screen name="Axis.Style" extends="Axis">
        …/…
        <tab name="colorsTab"…>
                <table name="colors"…>
                        …/…
                        <column name="product_reference"
i18n="product_reference" editable="true"/>
                </table>
        </tab>
        …/…
</screen>
```

Result on Style Attributes screen:

M. Adding custom fields and custom roles to General Data of a Collection Plan

Right click on the “Collection Plan” entry, then choose “Add custom field” or “Add custom role”.



N. Adding custom fields and custom roles to Slot Breakdown grid of a Collection Plan

Right click on the “Slot Breakdown” entry, then choose “Add custom field” or “Add custom role”