

# CocoonFormsCreatingWidgets

*Note: the following is a rough outline, is not fully tested and may, in fact, be completely incorrect. YMMV*

There are a number of steps required for adding a new Widget to CocoonForms:

1. Create the Java classes for the Widget.
2. Update the `forms-form.xconf` file.
3. Update the XSL which processes the widgets.
4. Rebuild Cocoon.

These steps are described in more detail below.

## Create the Java classes for the Widget

You'll need to create at least three classes to implement the Widget:

1. an implementation of the Widget interface;
2. an implementation of the WidgetDefinition interface; and
3. an implementation of the WidgetDefinitionBuilder interface.

The above interfaces all come from the `org.apache.cocoon.forms.formmodel` package. The existing Cocoon Forms source code provides plenty of examples of how to implement them.

The WidgetDefinitionBuilder parses the form definition document to create a WidgetDefinition. The WidgetDefinition is an **immutable** class that is used as a factory to create Widget instances. Widget instances are what constitute the form's object model that hold the form state.

## Update the forms-form.xconf file

Add a reference to your WidgetDefinitionBuilder implementation in the `forms-form.xconf` file, which can be found in the `conf` directory for the forms block. You have to specify the name of the element for that widget as well. This name should preferably match the string returned from the `getElementName()` method of your Widget implementation.

## Update the XSL which processes the widgets

You'll probably need to update the XSL file(s) which process the widgets. If you are using the samples that come as part of the forms block, this will most likely be `forms-field-styling.xsl` (which is found in the `COCOON_HOME/src/blocks/forms/samples/resources` directory). You will need to add a template which matches the element name of your new widget and generates appropriate output from it. The file contains plenty of templates to use as examples.

## Rebuild Cocoon

After all this, you'll need to rebuild cocoon. This will cause the new classes to be compiled, the updated classes to be recompiled and the `cocoon.xconf` file to be regenerated. It may be a good idea to do a `build clean` first, but that's up to you.

## For very specific widgets

In some rare cases, you may also need to modify the WidgetReplacingPipe. This is needed only for widgets that require a special handling by the template engine, such as repeater, class or union.

### Modify the **EffectWidgetReplacingPipe** class

- Add a new handler class for your widget, as an inner class of EffectWidgetReplacingPipe.
- Add a new instance of your handler class as a member variable.
- Add a class constant for the name of your new widget.
- In the constructor, add your new handler class to the templates map, using the constant as the key.

*Not sure whether this is actually necessary...*

### Modify the **WidgetReplacingPipe** class

You may need to make some changes to the WidgetReplacingPipe class, from the `org.apache.cocoon.forms.transformation` package. This will probably only be necessary if a tag other than `<widget>` is used to refer to the widget in form template files. Examples of this are the repeater and aggregate widgets.

If you do need to make these changes, the methods most likely to be affected are `startElement` and `endElement`.