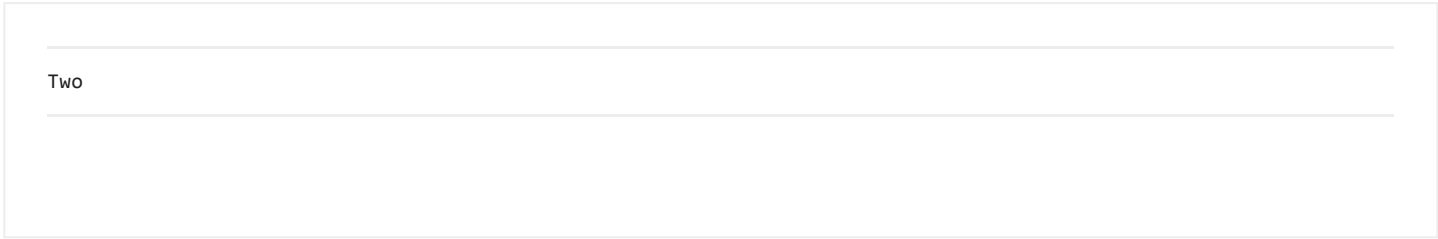
Creating custom renderers for XPages controls

May 10, 2013

One of the things that really annoys me about XPages is the way it renders controls in read only mode. So if you have a combo box in read only mode it would be displayed in a table like this:



Same goes for check box group and radio group. That's a lot of code to display just three characters and plus it completely destroys my CSS! This is where I believe custom renderers can be helpful.

WHAT IS A RENDERER?

A renderer is used to output some HTML markup corresponding to a control to be displayed in a web browser or in the Notes client. Every control renderer must extend from the JSF renderer base class *javax.faces.render.Renderer* or one of it's subclasses. More information available here.

CREATING A RENDERER

To create a Renderer you need to extend the *javax.faces.render.Renderer* class and override its methods *decode*, *encodeBegin*, *encodeChildren* and *encodeEnd* as per your requirement. We would only use the *encodeEnd* method as we just need to output the selected value.

```
package uk.co.pipalia;

import java.io.IOException;
import java.io.Writer;
import javax.faces.component.UISelectOne;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.render.Renderer;

public class ReadOnlyRenderer extends Renderer {
 public void encodeBegin(FacesContext context, UIComponent component)
  throws IOException {
 }

 public void encodeEnd(FacesContext context, UIComponent component)
  throws IOException {
   UISelectOne obj = (UISelectOne) component; // Get combo box object
   Writer writer = context.getResponseWriter(); // Get the write object
   writer.write(obj.getValue().toString()); // Write the value of combo box
 }
}
```

In the above class, in *encodeEnd* method we cast the component object to j*avax.faces.component.UISelectOne* as it is the class for combo box. We then get the value of combo box and write it to the *writer* object.

REGISTERING THE RENDERER CLASS

Now you need to register your custom renderer so that it can be used by your control. For that you need to modify *faces-config.xml*. Open your database in Package Explorer and go to "WebContent > WEB-INF > faces-config.xml". While registering your renderer make sure the *component-family* is same as the one to which the control (i.e. combo box) belongs to.

To find out the component family for your control you can use the code: *getComponent("comboBox1").getFamily()*. In case of combo box it is *javax.faces.SelectOne*.

So now your *faces-config.xml* file will look something like this:

```
      javax.faces.SelectOne
      uk.co.pipalia.type.ReadOnlyRenderer
      uk.co.pipalia.ReadOnlyRenderer
```

SETTING YOUR CUSTOM RENDERER TO CONTROL

To set the custom renderer in your control you can use the value defined in *renderer-type* from the *faces-config.xml*. But again we want this renderer to be activated only if the control is in read only mode.

```
.....
.....
```

In the above code we check if the combo box is in read only mode. If yes then we set the *rendererType* property to *uk.co.pipalia.type.ReadOnlyRenderer*, else we use the renderer type of combo box. Now if you view your XPage then only the value of the combo box would be shown without any tables.

This is very simple example which does not take into consideration if the document is in read only or edit mode. For that you would have to additionally check document mode before setting the renderer type. *But I would leave that for you to explore!*

May 10, 2013 [http://www.pipalia.co.uk/creating-custom-renderers-for-xpages-controls/] | by Naveen Maurya | in custom renderer, java, renderer, xpages .

6 Comments

6 thoughts on “Creating custom renderers for XPages controls”

jjtb somhorst
May 10, 2013 at 5:46 pm

Hi,

this is a great post! I’ve been playing around with the way how controls are being rendered as well and this is a great deal on how to customize that behaviour.

Naveen Maurya Post author
May 11, 2013 at 4:00 am

Thanks! Would love to hear if you have any other use cases where you can use the custom renderers in XPages.

Martin Rolph
July 22, 2013 at 9:42 am

Great article! After being inspired by your approach I've created an alternative version here:

http://openntf.org/XSnippets.nsf/snippet.xsp?id=remove-table-tag-for-listboxes-in-read-mode-for-all-instances-in-one-go

It removes the need to apply the SSJS for the rendererType for every ListBox

---

Naveen Maurya Post author
July 22, 2013 at 10:52 am

That's a really nice technique. Thanks for sharing Martin

Daniel Boston
November 26, 2014 at 5:06 pm

Thanks for this great intro!

For the uninitiated, I'd recommend looking in to FacesUtil.isComponentReadOnly() — this was the "missing link" in getting my own custom renderer for read-only content to work correctly when switching between document modes.

Samir Pipalia
December 2, 2014 at 1:28 am

Thanks Daniel, appreciate your feedback.
