Döcu Content

Create inputText Component

BackingBean

XSP inputText on the Fly

Introduction:
Create inputText components to App Browser on the fly using HtmlInputText, and HtmlPanelGrid imports; item that can be used to hide/show records from NSF back-end, based on specific user permissions.

Disclaimer:
Information contained in the following is presented as is. This tutorial assumes you have basic Lotus Notes Configuration and programming knowledge.

Create inputText to Browser

Use below BackingBean and XSP code to create inputText components straight away. You will be creating a skeleton Xpages file to run code and create default inputText component items, load additional items via button... You want to make sure your facesConfig file references the BackingBean in question; areas of interest are highlighted for your convenience.

Here are the Steps, see screenshots

1. Grab BackingBean code below and include it into your App, via a package
2. Create a new Xpages file, replace your code with ours
3. Save and run Xpages code in your Browser
4. Click button, add inputText items to form

Copy and Paste BackingBean

InputTextComponentBackingBean.java;

```
/**
 * Copyright 2014 Dököll Solutions, Inc.
   Licensed under the Apache License, Version 2.0 (the "License");
   you may not use this file except in compliance with the License.
   You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.
 *
```

```
 * @AppName: DöcuContent.nsf
 *  Program: InputTextComponentBackingBean.java
 *  Purpose: Create inputText items on the fly
 *  Created: 2014.02.01.9.19.PM
 *
 *  TODO: Add Strings to inputText items
 */

package com.dokoll.solutions.inc.Utils;

import java.util.ArrayList;
import java.util.Collection;
import javax.faces.component.html.HtmlForm;
import javax.faces.component.html.HtmlInputText;
import javax.faces.component.html.HtmlPanelGrid;


/**
* @author Dököll Solutions, Inc.
* @version 2014.02.01.9.19.PM
*/

//start program
public class InputTextComponentBackingBean {
	//declare component tree variables
	//...
	//declare html formId
	private HtmlForm HtmFrm;
	//declare inputText component types
	private Collection<HtmlInputText> inputCompFirst;
	private Collection<HtmlInputText> inputCompMid;
	private Collection<HtmlInputText> inputCompLast;
	//declare inputText components repository
	private HtmlPanelGrid BaseComponentFirst;
	private HtmlPanelGrid BaseComponentMid;
	private HtmlPanelGrid BaseComponentLast;
	//...
	//getters and setters
	public HtmlForm getHtmFrm() {
		return HtmFrm;
	}
	public void setHtmFrm(HtmlForm htmFrm) {
		this.HtmFrm = htmFrm;
	}
	public HtmlPanelGrid getBaseComponentFirst() {
		return BaseComponentFirst;
	}
	public void setBaseComponentFirst(HtmlPanelGrid baseComponentFirst) {
		this.BaseComponentFirst = baseComponentFirst;
	}
	public HtmlPanelGrid getBaseComponentMid() {
		return BaseComponentMid;
	}
	public void setBaseComponentMid(HtmlPanelGrid baseComponentMid) {
		this.BaseComponentMid = baseComponentMid;
	}
	public HtmlPanelGrid getBaseComponentLast() {
		return BaseComponentLast;
	}
	public void setGridComponent3(HtmlPanelGrid baseComponentLast) {
		this.BaseComponentLast = baseComponentLast;
	}
	@SuppressWarnings("unchecked")
	public InputTextComponentBackingBean() {
```

```
		//create IDs for each inputText Component
		inputCompFirst = new ArrayList<HtmlInputText>();
		for (int j = 0; j < 5; j++) {
			//plug IDs into inputText component
			HtmlInputText inputText = new HtmlInputText();
			BaseComponentFirst = new HtmlPanelGrid();
			inputText.setId("inputText1" + j);
			inputCompFirst.add(inputText);
		}
		//add component(s) to form
		BaseComponentFirst.getChildren().addAll(inputCompFirst);
		//create IDs for each inputText Component
		inputCompMid = new ArrayList<HtmlInputText>();
		for (int j = 0; j < 5; j++) {
			//plug IDs into inputText component
			HtmlInputText inputText = new HtmlInputText();
			BaseComponentMid = new HtmlPanelGrid();
			inputText.setId("inputText2" + j);
			inputCompMid.add(inputText);
		}
		//add component(s) to form
		BaseComponentMid.getChildren().addAll(inputCompMid);
		//create IDs for each inputText Component
		inputCompLast = new ArrayList<HtmlInputText>();
		for (int j = 0; j < 5; j++) {
			inputCompLast = new ArrayList<HtmlInputText>();
			//plug IDs into inputText component
			HtmlInputText inputText = new HtmlInputText();
			BaseComponentLast = new HtmlPanelGrid();
			inputText.setId("inputText3" + j);
			inputCompLast.add(inputText);
		}
		//add component(s) to form
		BaseComponentLast.getChildren().addAll(inputCompLast);
	}
	//button code
	@SuppressWarnings("unchecked")
	public void doLoadComponents() {
		int uComp = BaseComponentFirst.getChildCount();
		//send NEW inputText component to form
		HtmlInputText inputText1 = new HtmlInputText();
		inputText1.setId("inputText1" + uComp);
		BaseComponentFirst.getChildren().add(inputText1);
		//send NEW inputText component to form
		//...
		//int uComp = baseComponentMid.getChildCount();
		HtmlInputText inputText2 = new HtmlInputText();
		inputText2.setId("inputText2" + uComp);
		BaseComponentMid.getChildren().add(inputText2);
		//send NEW inputText component to form
		//...
		//int uComp = baseComponentLast.getChildCount();
		HtmlInputText inputText3 = new HtmlInputText();
		inputText3.setId("inputText3" + uComp);
		BaseComponentLast.getChildren().add(inputText3);
	}

}
```

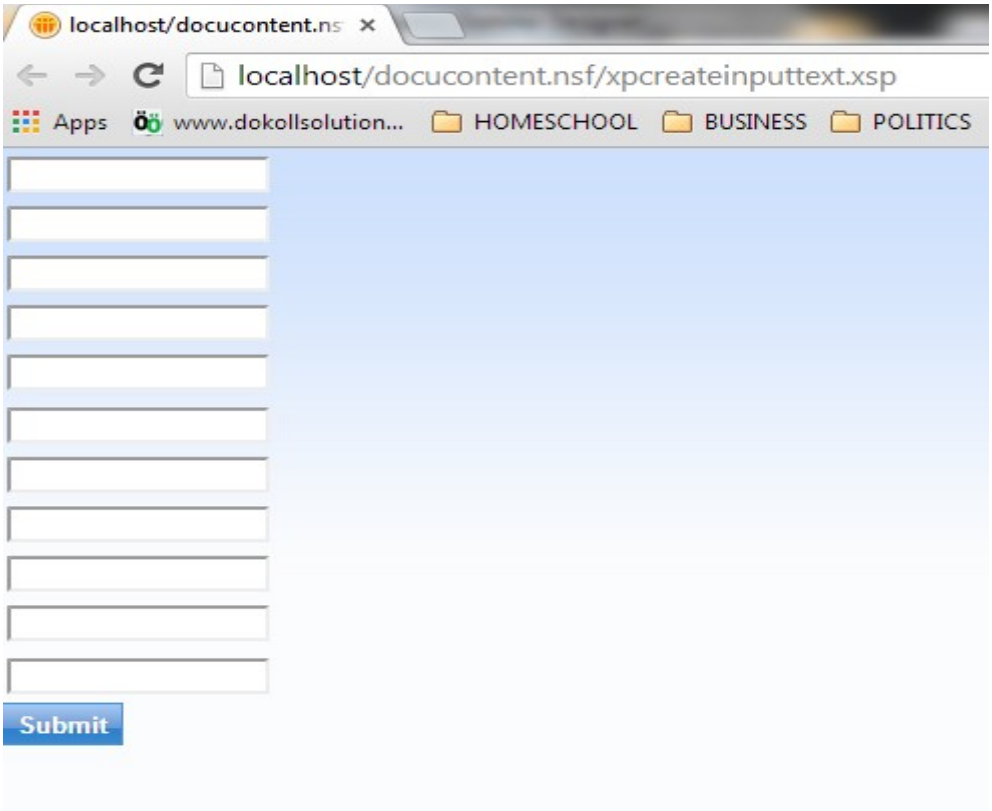
Copy and Paste XSP Code

```
<?xml version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core">
```

```
	<xp:panel
binding="#{InputTextComponentBackingBean.baseComponentFirst}"></xp:panel>
	<xp:panel
binding="#{InputTextComponentBackingBean.baseComponentMid}"></xp:panel>
	<xp:panel
binding="#{InputTextComponentBackingBean.baseComponentLast}"></xp:panel>
	<xp:button value="Submit" id="button1">
		<xp:eventHandler event="onclick" submit="true"
			refreshMode="complete" immediate="false" save="true"
id="eventHandler1">
			<xp:this.action><!
[CDATA[#{javascript:InputTextComponentBackingBean.doLoadComponents()}]]>
</xp:this.action>
		</xp:eventHandler>
	</xp:button>
</xp:view>
```

Run to Browser

Be sure to Reference your BackingBean in facesConfig XML file, and run code



Conclusion:

You can now create inputText components on the fly to your Browser where additional/dynamic functionality is needed in your Apps; information that can be used to help you make your Apps scalable and better service your users.

Questions, comments, please post a brief message on our Contact form on the main site.

Thank you for coming...

Version:2014.02.14.1.27.PM