**Configuring a Picker to Offer Only Specific Soft Types Based on User-specified Criteria including Restricted Life**

**Cycle States**

**Objective**

Configure a picker to only display a specific soft-type of parts (Datecode Release). Allow user to enter criteria which includes name, library and lifecycle state. The states shown to the user must only be ones that represent a state that the objects can have, not all LC states in the system. The user to be able to pick a value for the attribute "Release Stream" (Windchill 8.0, 9.0, etc). The picker

should show the datecode name, not show its number, and also include the library that contains the part. The picker will be used in the process to create Affected Data relationships for a Change Request.

**Associated requirements:**

• Invoking a picker with customized seeded criteria.

• Customizing the columns of a table

**Background**

Problem scenario represents common customization scenario for Item Picker. Picker is common UI component used for searching for referenced objects and selecting one or more of them on calling application. You can customize criteria attributes and result table columns shown in picker. You can configure Item Picker to search for single or multiple object types. You can restrict search to only

in particular container or container types.

**Scope/Applicability/Assumptions**

• Item Picker can be launched from any JSP page using ItemPicker tag.

• Picker returns picked objects as a Jason object by calling JavaScript function on calling page.

• It is the responsibility of calling page to further act on this Jason object.

• Soft type of Part is Date code so it’s fully qualified external form would be like WCTYPE|wt.part.WTPart|org.r\_and\_d.DatecodeRelese

**Intended Outcome:**

Datecode Release picker with Name, context and State as criteria attributes:



**Solution**

A custom JSP tag with appropriate parameters would be used to launch customized Item Picker. To choose the Library, Library Picker would be used.

**Prerequisite knowledge**

To achieve this objective, you need to have an understanding of the following:

• Using JSP tags

• Using xconfManager

• Updating XML document.

• Using resource class files

**Solution Elements**

|  |  |  |
| --- | --- | --- |
| **Element** | **Type** | **Description** |
| PickerAttributes.xml | Xml | Location: <Windchill>\ codebase |
| SearchableTypes.propertis.xconf | Xconf properties file | Location: <Windchill>\ codebase\com\ptc\windchill\ enterprise\search\server\SearchableTypes.properties.xconf |
|  | JavaScript file | Add your callback function in this file. |
| itemPicker.tag | JSP tag | Location: <Windchill>\ codebase\WEB-INF\tags\ itemPicker.tag |

To choose Library, Library Picker is shown.

**Procedure – Launch Item Picker with custom criteria and result table to pick DateCodeRelease objects**

**Using Item Picker Tag**

To show Item Picker you can use ItemPicker tag as shown below:

<%@ taglib prefix="wctags" tagdir="/WEB-INF/tags" %>

…

<tr>

<%-- launching Item picker--%>

<wctags:itemPicker id="myItemPicker" label="Custom Item Picker" pickerTitle="Datecode Release" showVersion="false"

objectType=" WCTYPE|wt.part.WTPart|org.r\_and\_d.myPart " showTypePicker="false" componentId="test.Customization"

typeComponentId="Foundation.customization" searchResultsViewId="wt.part.WTPart.customizedSearchView"/>

</tr>

….

**The important parameter are described below:**

• ObjectType – Fully qualified external form name of the object.

• ComponentId – componentId to be used for configuring criteria attributes.

See section below for details

• TypeComponentId - used for defining set of types and container types to be searched for. In our case this parameter is used to constraint our search only within Library containers.

**Defining ComponentId for configuring criteria attributes**

This section describes, adding new componentId.

<wctags:itemPicker id="myItemPicker" label="Custom Item Picker" pickerTitle=" Datecode Release" showVersion="false"

objectType="WCTYPE|wt.part.WTPart|org.r\_and\_d.myPart " showTypePicker="false" componentId="test.Customization"

typeComponentId="Foundation.customization" searchResultsViewId="wt.part.WTPart.customizedSearchView"/>

ComponentId is resolved from pickerAttributes.xml file to pick up the list of attributes to be shown in picker criteria. Update pickerAttributes.xml as below:

<PickerAttributes>

…

<ComponentID id="test.Customization">

<ObjectType id="WCTYPE|wt.part.WTPart|

org.r\_and\_d.myPart">

<SearchCriteriaAttributes>

<Attributes>

<Name>name</Name>

<DisplayName>NAME\_LABEL</DisplayName>

<IsSearchable>true</IsSearchable>

</Attributes>

<Attributes>

<Name>containerRef</Name>

<DisplayName>CONTEXT\_LABEL</DisplayName>

<IsSearchable>false</IsSearchable>

</Attributes>

<Attributes>

<Name>state.state</Name>

<DisplayName>STATE\_LABEL</DisplayName>

<IsSearchable>true</IsSearchable>

</Attributes>

</SearchCriteriaAttributes>

</ObjectType>

</ComponentID>

…

</PickerAttributes>

Here we have defined search criteria attributes as name, context and state.

DisplayName is resolved from class file : com\ptc\windchill\enterprise\search\client\searchClientResource\_ja.class

**Defining typeComponentId for constraining search to given container types:**

As per problem statement, we need to search only within Library containers. This is achieved by passing typeComponentId. To configure typeComponentId add below property to <Windchill>\codebase\com\ptc\windchill\enterprise\search\server\SearchableTypes.properties.xconf

<Property name="Foundation.customization.containerType" default="wt.inf.library.WTLibrary"/>

And run xconfManager to propagate.

**Customizing the search result table view**

Search result tables use the Windchill R9.0 table common component and hence the column display can be configured by creating a new table view. Please refer [Crating Table View] for details regarding creating and configuring a new view.

Then we can pass the view as below to the picker tab and search results table would be rendered with this view.

<wctags:itemPicker id="genericPicker" objectType="WCTYPE|wt.part.WTPart|org.r\_and\_d.mypart"

label="Part Picker using generic" containerRef="<%=containerRefs.toString()%>"

componentId="scenarioOnePicker" searchResultsViewId="wt.part.WTPart.customizedSearchView" showVersion="false"/>

The view id “wt.part.WTPart.customizedSearchView” should be configured to lookup the java class associated with this id which would define the table view.

Please refer to configuring table views document.

An example to configure this in SearchResultTable.properties.xconf file could be

<Service context="default" name="com.ptc.core.htmlcomp.tableview.ConfigurableTable">

<Option serviceClass="com.ptc.netmarkets.search.views.CustomizedPartView" requestor="java.lang.Object" selector="wt.part.WTPart.customizedSearchView "/>

</Service>

A new java class com.ptc.netmarkets.search.views.CustomizedPartView should be created which would define the view columns for this picker customization.

**Customization Points**

Please refer [Item Picker Best Practice] for further customizing Item Picker.

**Limitations**

As per current implementation, Lifecycle State is not limited only to states that the object can have.

**Sample Code**

<%@ taglib prefix="wctags" tagdir="/WEB-INF/tags" %>

<HTML>

<BODY><TABLE>

<tr>

<%-- launching Item picker--%>

<wctags:itemPicker id="myItemPicker" label="Custom Item Picker" pickerTitle=" Datecode Release" showVersion="false"

objectType=" WCTYPE|wt.part.WTPart|org.r\_and\_d.myPart " showTypePicker="false" componentId="test.Customization"

typeComponentId="Foundation.customization" searchResultsViewId="wt.part.WTPart.customizedSearchView"/>

</tr>

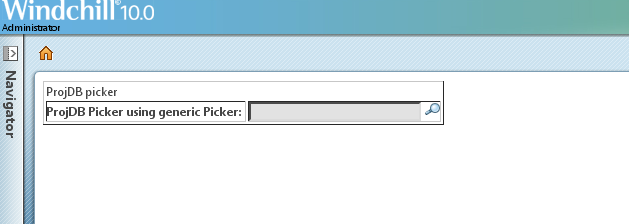
</BODY></TABLE>

</HTML>

**Building a Picker that Enables Users to Select Projects from an External Project DB**

Steps:-

1. Need to write a jsp to configure a generic picker for this requirement. (Refer Appendix: A1). Copy this file to \Windchill\codebase\netmarkets\jsp\search\



1. Customize the search criteria on the picker page:

For this add the attribute definition for our picker component id in \Windchill\codebase\pickerAttributes.xml as below:

<ComponentID id="projDBPickerId">

<ObjectType id="wt.part.WTPart">

<SearchCriteriaAttributes>

<Attributes>

<Name>Name</Name>

<DisplayName>Name</DisplayName>

<IsSearchable>true</IsSearchable>

</Attributes>

<Attributes>

<Name>projId</Name>

<DisplayName>Project ID</DisplayName>

<IsSearchable>true</IsSearchable>

</Attributes>

<Attributes>

<Name>projDesc</Name>

<DisplayName>Project Description</DisplayName>

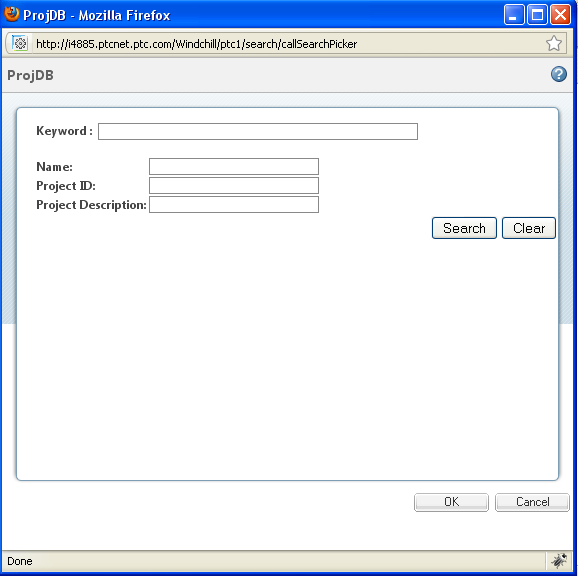
<IsSearchable>true</IsSearchable>

</Attributes>

</SearchCriteriaAttributes>

</ObjectType>

</ComponentID>



1. Configuring the customized view for results table:

For this write a class ProjDbPickerResultTableView that will define the columns that need to be present in the picker view table.

(Refer Appendix A2).

Configure this view in \Windchill\codebase\com\ptc\netmarkets\search\SearchResultsTable.properties.xconf by copying the following line:

<Option serviceClass="com.ptc.netmarkets.search.views.ProjDbPickerResultTableView" requestor="java.lang.Object" selector="wt.fc.Persistable.defaultPickerProjDbView"/>

Run Xconfmanager –p to propagate the changes.

1. Write the Search Results Data fetcher jsp i.e. projDataFetcher.jsp (Refer Appendix A3)

Copy this file to \Windchill\codebase\netmarkets\jsp\search\

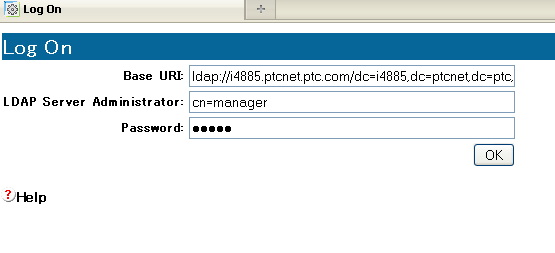
1. Need to write pickedData jsp to display the selected value in the picker. For e.g. projDbPickedData.jsp (Refer Appendix A4)

Copy this file to \Windchill\codebase\netmarkets\jsp\search\

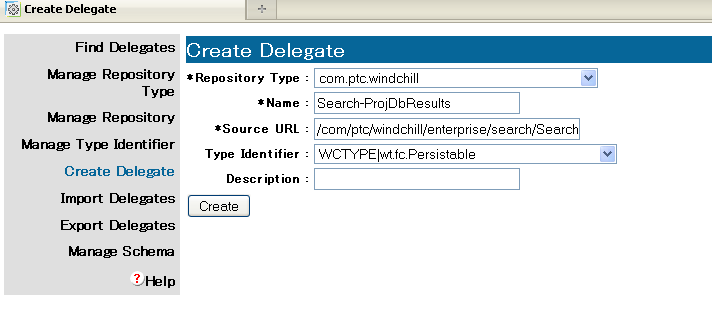
1. Retrieve external data:

For this

* Create an IE Task Search-ProjDbResults.xml. (Refer Appendix A5)
* Write a Delegate ProjDbResultsDelegate.java that will read the input parameters from the task and generate the output group with data for desired columns. (Refer to Appendix A6) Copy this file in the path \Windchill\DevModules\WindchillAdapter\src\com\ptc\core\adapter\server\impl
* Register the task Using Task delegate Administrator. Refer to following snapshot. Password is generally “admin” if you have not changed it during LDAP configuration.



* Then create a new delegate as:



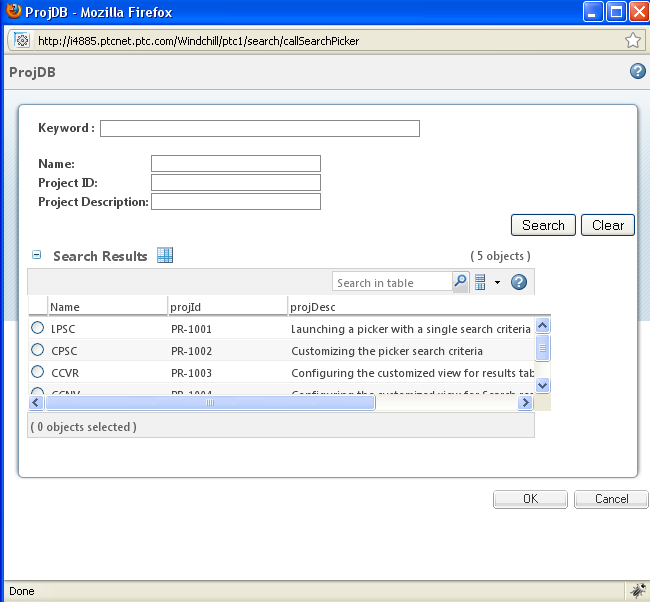
This will register your task. Run the task in browser to test if it works.

* Configure the delegate for the task by copying the following line in \windchill\codebase\wt.adapter.delegates.properties

SEARCHPROJDBRESULTS.WCTYPE|java.lang.Object=com.ptc.core.adapter.server.impl.ProjDbResultsDelegate

1. Test the picker.

The output looks like:



1. Currently values are hardcoded in delegate as well as in projDbPickedData.

**Pickers from table toolbar**

**Objective**

You want to a picker from table toolbar actions.

**Background**

The pickers from table toolbar are used when you have a requirement to launch pickers from some table level action and accordingly populate the tables or do some processing based on the results.

**Scope/Applicability/Assumptions**

Say you want to launch <your picker> from your table. It is assumed that your action jsp file <MyPage>.jsp file in which you are putting picker tag includes

“/netmarkets/jsp/begin.jspf” or “/netmarkets/jsp/beginPopuf.jspf” file and “/netmarkets/jsp/end.jspf” files.

**Intended Outcome**

On click of the action, you should see the corresponding picker getting launched.

**Solution**

Use the specific picker Common Component in your JSP file to include the picker in your application.

**Prerequisite knowledge**

To achieve this objective, you need to have an understanding of the following:

• Basic development involving JSP, JavaScript, Custom taglibs

• The management of resource files customization.

• Windchill xconfmanager concepts.

**Solution Elements**

|  |  |  |
| --- | --- | --- |
| **Element** | **Type** | **Description** |
| pickerAttributes.xml | XML file | You should use this file to customize the search criteria attributes for an object type for your picker.  Runtime location:  <WT\_HOME>\codebase\pickerAttributes.xml |
| searchClientResource.java | .java file | You should use this file to localize the contents of the pickers.  Runtime location:  <WT\_HOME>\src\com\ptc\windchill\enterprise\search\client\searchClientResource.java |
| <your picker>.tag | Custom JSP tag file | This file is contains the information about the supported parameters for this tag.  Runtime location:  <WT\_HOME>\codebase\ WEBINF\tags\<your picker>.tag |
| custom.js | JavaScript file | You should specify custom pickerCallback function for picker in this file.  Runtime location:  <WT\_HOME>\codebase\netmarkets\jsp\search\ custom.js |

**Procedure – Including a <your picker> in a jsp page**

**Configure basic picker:**

<tr>

<wctags:<your picker name> id="myPickerId" inline=”true” pickedAttributes=”<name of the attribute>”

pickerCallback=”<yourpickerCallback function>” />

</tr>

**Customization Points**

Please refer to the <specific picker> Best Practice document for details regarding the parameters.

For pickers that are launched from table toolbar they should have at least following parameters in the tags.

inline=”true”

pickedAttributes=”<comma separated list of attribute names to be

fetched>

pickerCallback=”<name of pickerCallback function>”

multiSelect=”true”

**Generating HTML Tags for ProductView Visualization Within a JSP Page**

To generate HTML code for the ProductView visualization components in a JSP page, use the com.ptc.wvs.common.ui.VisualizationHelper class. See this class entry in your installed Windchill Javadoc for more information.

**General flow of logic:**

The VisualizationHelper can be instantiated via the newVisualizationHelper() factory method.

getCreateDialogWindow() is required to generate the Javascript fragment required by the results returned from getDefaultVisualizationData(String, Boolean, Locale). This code must be embedded somewhere in the HTML returned by the JSP page.

getDefaultVisualizationData(String, Boolean, Locale) method returns an array of String values. Each value corresponds to a different component within the ProductView Visualization Service. Elements in the array can be accessed directly through the indexes returned by the following methods:

getIndexOfVisLink(String, String), clipboardLinkIndex(), printLinkIndex(), viewRepsLinkIndex().

Each String is the base HTML code for the ProductView visualization component and can be embedded into the HTML returned by the

JSP page. It may be an '<A HREF .. /A>' code tag or other simple unformatted HTML tags.

**Note:** The getDefaultVisualizationData(String, Boolean, Locale) method returns the basic HTML components necessary for accessing the ProductView visualization components. Most likely the results will need to be used within proper HTML tags. The responsibility of designing the user interface and properly using the values returned by these methods belongs to the implementer.

**Example usage:**

// comp.ptc.wvs.common.ui.VisualizationHelper is assumed to be imported.

VisualizationHelper visHelper = newVisualizationHelper();

if (visHelper.isWVSEnabled()) //only execute code if WVS is enabled

{

// Embed the create dialog JavaScript somewhere in returned HTML.

// The below is just an example of calling it. Placement within

// HTML is the responsibility of the implementer.

out.println(visHelper.getCreateDialogWindow());

//ObjectReference object or and locale object is assumed to be obtained prior to

the following line

String visData[] = visHelper.getDefaultVisualizationData(or.toString(),new

Boolean(false), locale);

//Obtain array indices for thumbnail, clipboard link, print link, and

representations link

int thumbnailIndex = visHelper.getIndexOfVisLink(

VisualizationHelper.THUMBNAILS\_PROP\_PAGE\_PREF\_KEY,

VisualizationHelper.DEFAULT\_THUMBNAILS\_PROP\_PAGE\_PREF\_VALUE);

int clipboardIndex = visHelper.clipboardLinkIndex();

int printIndex = visHelper.printLinkIndex();

int repsOrMarkupsIndex = visHelper.viewRepsLinkIndex();

/\*\*

\* Print the various HTML code fragments generated for each component.

\* Any HTML formatting code must be wrapped around each of the following lines

\*\*/

out.println(visData[thumbnailIndex]);

out.println(visData[clipboardIndex]);

out.println(visData[printIndex]);

out.println(visData[repsOrMarkupsIndex]);

}

**Appendix:**

**A1:** – Write test.jsp as follows:

<%@ taglib prefix="wctags" tagdir="/WEB-INF/tags" %>

<%@ taglib uri="http://www.ptc.com/windchill/taglib/picker" prefix="p"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@ page import="wt.util.WTProperties,

com.ptc.netmarkets.util.misc.NmActionServiceHelper,

com.ptc.netmarkets.util.misc.NmAction,

com.ptc.netmarkets.util.misc.NmTextBox,

com.ptc.core.components.rendering.guicomponents.TextBox,

com.ptc.core.components.rendering.RenderingContext,

java.util.HashMap,

java.util.Locale,

com.ptc.core.components.rendering.AbstractRenderer,

com.ptc.core.components.rendering.guicomponents.GuiComponentUtil"%>

<%

request.setAttribute("contextAction", "search");

request.setAttribute("pageCheck", "search");

request.setAttribute("currentPage", "PickerExample");

%>

<%@ include file="/netmarkets/jsp/util/begin.jspf"%>

<html>

<body>

<table border="1">

<tr>

ProjDB picker

</tr>

<tr>

<script>

function myCallback(objects, pickerID, displayFieldId) {

//alert("displayFieldId=== "+displayFieldId);

var updateHiddenField = document.getElementsByName(pickerID)[0];

//alert("updateHiddenField=== "+updateHiddenField);

var displayValue = objects.pickedObject[0]["name"];

var oid = objects.pickedObject[0]["oid"];

$(pickerID).value = oid;

$(pickerID+'$label$').value = displayValue;

}

</script>

<%-- launching Proj DB Picker --%>

<wctags:genericPicker id="projDBPicker"

objectType="wt.part.WTPart" componentId="projDBPickerId"

pickedDataFetcher="/Windchill/netmarkets/jsp/search/projDbPickedData.jsp"

searchResultsFetcherURL="/netmarkets/jsp/search/projDataFetcher.jsp"

label="ProjDB Picker using generic Picker"

pickerTitle="ProjDB"

pickerCallback="myCallback"

searchResultsViewId="wt.fc.Persistable.defaultPickerProjDbView"/>

</tr>

</table>

</body>

</html>

<%@ include file="/netmarkets/jsp/util/end.jspf"%>

**A2: Write a class ProjDbPickerResultTableView.java as below:**

package com.ptc.netmarkets.search.views;

import com.ptc.core.components.descriptor.DescriptorConstants.ColumnIdentifiers;

import com.ptc.core.htmlcomp.jstable.JSPropertyDataConstants;

//import com.ptc.core.ui.tableRB;

import wt.clients.beans.table.tableRB;

import com.ptc.core.htmlcomp.createtableview.Attribute;

import com.ptc.core.htmlcomp.tableview.TableColumnDefinition;

import com.ptc.core.htmlcomp.tableview.TableViewDescriptor;

import com.ptc.netmarkets.search.SearchWebConstants;

import java.util.ArrayList;

import java.util.List;

import java.util.Locale;

import java.util.Vector;

import wt.enterprise.enterpriseResource;

import wt.fc.PersistInfo;

import wt.inf.container.WTContained;

import wt.lifecycle.LifeCycleState;

import wt.org.OrganizationServicesHelper;

import wt.part.WTPartMaster;

import wt.util.WTException;

import wt.util.WTMessage;

import wt.util.WTPropertyVetoException;

public class ProjDbPickerResultTableView extends PersistableResultTableView {

@Override

public boolean isPickerOnlyView() {

return true;

}

@Override

public List getOOTBTableViews(String tableId, Locale locale) throws WTException {

Vector columns = new Vector(13);

TableColumnDefinition nameColumnDef = TableColumnDefinition.newTableColumnDefinition(ColumnIdentifiers.NAME, false);

TableColumnDefinition projIdColumnDef = TableColumnDefinition.newTableColumnDefinition("projId", false);

TableColumnDefinition projDescColumnDef = TableColumnDefinition.newTableColumnDefinition("projDesc", false);

TableColumnDefinition oidColumnDef = TableColumnDefinition.newTableColumnDefinition("oid", false);

columns.add(nameColumnDef);

columns.add(projIdColumnDef );

columns.add(projDescColumnDef );

columns.add(oidColumnDef );

//String name = getViewResourceEntryKey(SearchWebConstants.SEARCH\_CLIENT\_RESOURCE, "PROJDB\_TABLEVIEW\_NAME");

//String description = getViewResourceEntryKey(SearchWebConstants.SEARCH\_CLIENT\_RESOURCE, "PROJDB\_TABLEVIEW\_DESC");

String name = "Projdb name";

String description = "Projdb Description";

TableViewDescriptor desc = TableViewDescriptor.newTableViewDescriptor(name, tableId, true, true, columns, null, true, description);

ArrayList result = new ArrayList(1);

result.add(desc);

return result;

}

}

**A3 : Write the Search Results Data fetcher i.e. projDataFetcher.jsp**

<%@ page import="com.ptc.netmarkets.search.beans.\*,

com.ptc.netmarkets.search.SearchResultsParamHelper,com.ptc.netmarkets.search.SearchResultsHelper,

com.infoengine.object.factory.Group"%>

<%@ page import="com.ptc.netmarkets.search.utils.SearchUtils,

com.ptc.netmarkets.search.SearchWebConstants"%>

<%@ page import="java.util.\*"%>

<%

SearchBean searchbean = null;

if(request.getAttribute("searchbean") != null)

{

searchbean = (SearchBean) request.getAttribute("searchbean");

}else{

searchbean = new SearchBean();

}

SearchResultsParamHelper searchResultsParamHelper = null;

if(request.getAttribute("paramHelper") != null)

{

searchResultsParamHelper = (SearchResultsParamHelper) request.getAttribute("paramHelper");

}else{

SearchResultsHelper searchResultsHelper = new SearchResultsHelper(searchbean);

searchResultsParamHelper = new SearchResultsParamHelper(searchbean, request, searchResultsHelper);

}

pageContext.setAttribute("paramHelper", searchResultsParamHelper);

searchbean.getGenHmSearchAttr().put(SearchWebConstants.SEARCH\_TASKNAME, "com/ptc/windchill/enterprise/search/search-ProjDbResults.xml");

%>

**A4: Write pickedData jsp (**projDbPickedData.jsp**) to display the selected value in the picker as below:**

<%@ taglib uri="http://www.ptc.com/infoengine/taglib/core" prefix="ie"%>

<%@ page import="com.ptc.netmarkets.util.NmUtils" %>

<%

//System.out.println("inside projDbPicked Data--------------- ");

String[] oids = request.getParameterValues("selectedOids");

String selectedValue = oids[0];

//System.out.println("selectedValue=== "+selectedValue);

String displayName = "swa";

if(selectedValue.equals("OR:wt.part.WTPart:1"))

displayName = "LPSC";

else if(selectedValue.equals("OR:wt.part.WTPart:2"))

displayName = "CPSC";

else if(selectedValue.equals("OR:wt.part.WTPart:3"))

displayName = "CCVR";

else if(selectedValue.equals("OR:wt.part.WTPart:4"))

displayName = "CCNV";

else if(selectedValue.equals("OR:wt.part.WTPart:5"))

displayName = "CCRV";

StringBuilder sb = new StringBuilder();

sb.append("{\"pickedObject\": [{");

//sb.append("\"name\":\"LPSC\",");

sb.append("\"name\":");

sb.append("\"");

sb.append(displayName);

sb.append("\",");

sb.append("\"oid\":\"");

sb.append(selectedValue);

sb.append("\"");

sb.append("}]}");

String jsonObjectString = sb.toString();

//System.out.println("jsonObjectString============================= "+jsonObjectString);

NmUtils.escapeQuotes(jsonObjectString);

out.print(jsonObjectString);

out.flush();

%>

**A5: Write search-ProjDbResults as below:**

<?xml version="1.0" standalone="yes"?>

<%@taglib uri="http://www.ptc.com/infoengine/taglib/core" prefix="ie"%>

<!DOCTYPE PROCESS>

<PROCESS name="Search ProjDb Results">

<description>Get the list of project dbs</description>

<ie:webject name="search-ProjDbResults" type="OBJ">

<ie:param name="projId" data="PR-1005,PR-1006,PR-1007"/>

<ie:param name="oid" data="PR-1005,PR-1006,PR-1007"/>

<ie:param name="projDesc" data="Launching a picker with a single search criteria,Customizing the picker search criteria,Configuring the customized view for results table"/>

<ie:param name="Name" data="LPSC,CPSC,CCVR"/>

<ie:param name="INSTANCE" data="${@FORM[]supporting-adapter[\*]}"/>

<ie:param name="group\_out" data="${@FORM[]group\_out[]}" default="output"/>

</ie:webject>

</PROCESS>

**A6: Write ProjDbResultsDelegate.java as follows:**

package com.ptc.core.adapter.server.impl;

import java.io.FileWriter;

import java.io.PrintWriter;

import wt.util.WTException;

import com.infoengine.connector.Attribute;

import com.infoengine.object.factory.Att;

import com.infoengine.object.factory.Group;

import com.infoengine.object.factory.ParameterDef;

import com.infoengine.object.factory.Task;

import com.infoengine.object.factory.Webject;

import com.infoengine.object.factory.WebjectDef;

import com.infoengine.object.factory.Element;

@WebjectDef (

name = "search-ProjDbResults",

type = "OBJ",

description = "Get the list of project dbs",

parameters = {

// over-ride TYPE

@ParameterDef(name="projId",maxValues=-1,

description="The type or types to perform the search against. If more than one type " +

"is specified then ATTRIBUTE\_TYPE\_CONTEXT must also be specified.", defaultValue="PR-1005,PR-1006,PR-1007"),

@ParameterDef(name="projDesc",maxValues=-1,defaultValue="Launching a picker with a single search criteria,Customizing the picker search criteria,Configuring the customized view for results table",

description="Specifies whether the current attribute to sort should be sorted ascending or descending."),

@ParameterDef(name="Name",maxValues=-1,defaultValue="LPSC,CPSC,CCVR",

description="Specifies a set of attributes that a both specified to be returned by the query and " +

"that have sorting preferences defined. If multiple values are defined their order determines " +

"the precedence of sorting.")

},

unsupported = {

"DESCRIPTOR", "GROUP\_FILTER",

"NEXT\_OP", "OBJECT\_REF", "REFERENCE\_DELIMITER", "REFERENCE\_OUTPUT\_DELIMITER", "REFERENCE\_EXCEPTIONS", "GROUP\_IN",

"WHERE\_CASE\_SENSITIVITY", "INCLUDE\_ARGS", "INCLUDE\_CONSTRAINTS", "INCLUDE\_DESCRIPTORS"

}

)

public class ProjDbResultsDelegate extends ObjectWebject {

public Task invoke( Task task )

throws WTException {

Task response = new Task();

try {

// Gets parameters from input Task

Group outputGroup = new Group("OUT");

outputGroup.setClassName(groupOutClass);

System.out.println("\n\n\n\n in ProjDbResultsDelegate......................");

Webject webject = task.getWebject();

String projId = webject.paramValue("projId");

String projDesc = webject.paramValue("projDesc");

String Name = webject.paramValue("Name");

String projectId = "PR-100" ;

String[] projectDescription = {"Launching a picker with a single search criteria", "Customizing the picker search criteria","Configuring the customized view for results table","Configuring the customized view for Search results table","Configuring the customized view for picker results table"};

String[] projectName = {"LPSC", "CPSC", "CCVR", "CCNV", "CCRV"};

for(int i =0 ; i< 5; i++){

Att att4, att, att2, att3,att5;

Element node = new Element();

att5 = new Att("obid");

att5.addValue(projectId +( i+1));

node.addAtt(att5);

//node.setUfid("OR:wt.part.WTPart:89642");

att4 = new Att("oid", projectId +( i+1));

node.addAtt(att4);

att = new Att("projId");

att.addValue(projectId +( i+1));

node.addAtt(att);

att2 = new Att("projDesc");

att2.addValue(projectDescription[i]);

node.addAtt(att2);

att3 = new Att("Name");

att3.addValue(projectName[i]);

node.addAtt(att3);

node.setUfid("OR:wt.part.WTPart:"+(i+1));

outputGroup.addElement(node);

}

//PrintWriter pwriter = new PrintWriter(System.out,true);

//outputGroup.toXML(pwriter, true);

response.addVdb(outputGroup);

}catch(Exception e){

}

System.out.println("\n\n\n : End of ProjDbResultsDelegate");

return response;

}

}