Adding an Input Field to a Component

In This Topic

Adding a Field to an Element Component

The topic Quick Start explains how to create a very basic element component using the element archetype. The archetype generates the following files (in the case of the HelloWorld example):

Resource	Description
`Activator.java`	Bundle activator
`CustomElement.java`	Interface representing the custom element
`CustomElementController.java`	Controller of the element component
`CustomElementFBO.java`	Form backing object of the element component
`CustomElementImpl.java`	Component and element implementation (business object)
`editCustomElement.jspf`	JSP that renders the element in the Edit environment
`messages_en_US.properties`	US English language file
`messages_nl_NL.properties`	Dutch language file
`showCustomElement.xml`	Descriptor of the JSP rendering the element on the Website environment
`showCustomElement.jspf`	JSP rendering the element on the Website environment
`smallCustomElementIcon.gif`	Small icon associated with the element
`largeCustomElementIcon.gif`	Large icon associated with the element

This part explains how to enhance this component with an additional field called "Author". To add this field to the Hello World element component, the following steps must be performed:

1. Add the property to the business object (`CustomElementImpl.java`).
2. Add the property to the form backing object (`CustomElementFBO.java`).
3. Add a label to the language files to translate “Author” in the supported languages.
4. Add the field to the edit JSP (`editCustomElement.jspf`).
5. Add the field to the show JSP (`showCustomElement.jspf`).

Throughout this topic the HelloWorld element will be used as an example.

Adding a Property to the Business Object

The Business object `CustomElementImpl` represents the data model of the Hello World element. Since we want the author field to be persisted, we will add this property to this class. We define the property as an instance variable and add a getter and setter for it both in the interface and the implementation:

`CustomElement.java`

```
/**
 * Returns the author of the custom element.
 * @return the author of the custom element
 */
String getAuthor();

/**
 * Sets the author of the custom element.
 * @param author the author of the custom element to set
 */
void setAuthor(String author);
```

`CustomElementImpl.java`

```
@Property
public String getAuthor() {
        return JcrUtil.getString(getNode(), "myJcrPrefix:author");
}
public void setAuthor(String author) {
        JcrUtil.setString(getNode(),"myJcrPrefix:author", author);
}
```

Adding a Property to the Form Backing Object (FBO)

Since the author must also be editable in the Editor, we also add the field to the Form backing object, `CustomElementFBO.java`. This is done in a similar way as was done for the business object:

```
private String myAuthor;
public String getAuthor() {
        return myAuthor;
}
public void setAuthor(String author) {
        myAuthor= author;
}
```

Adding a Language Label

Because we want the word “Author” to be translated in the proper language in the Workspace as well as the website environment, we must define a language label for it. The element archetype generated two language files, one for US English (`messages_en_US.properties`) and one for Dutch (`messages_nl_NL.properties`) (additional language files can also be added). When creating language files, be sure that they conform to the development guidelines (G026, G027 and G028 in particular).

To each of these language files the author label must be added. The ID of the label is suffixed by an ‘=’ and followed by the translated value in the language to be added. The following code snippet needs to be added to the messages_en_US.properties and must be repeated for each supported language:

```
helloworldelement.author=Author
```

Adding a Field to the Edit JSP

Because "Author" must be editable by users logged in to the XperienCentral Workspace, we must add the field to the JSP that renders the element (`editCustomElement.jspf`). Standard tags for property editing are supported by the `wmedit` tag library. The `wmedit` tag library generates the HTML for editing a property of a particular type. The property type that the tag should generate the HTML for is provided after the “:” separator. For a regular text input field the edit tag is input. To create an input field for editing the author, add this to the JSP:

```
<fmt:message key="helloworldelement.author" />:
<wmedit:input path="author"/>
```

Libraries must be imported into the JSP, therefore the header of the JSP should contain at least the following lines:

```
<%@ taglib uri="http://java.sun.com/jstl/fmt" prefix="fmt" %>
<%@ taglib prefix="wmedit" uri="http://www.gx.nl/taglib/wmedit"%>
```

Adding a Field to the Show JSP

The "Author" field can now be edited in the Editor but must also be shown in the website environment. We therefore enhance the show JSP (`showCustomElement.jspf`) to display its value:

```
<c:set var="element" value="${presentationcontext.element}" />
Author: ${element.author}
```

Back to Top

Adding a Field to a Panel Component

In Quick Start it is explained how to create a very basic panel component using the panel archetype. This panel consists of only one tab, the "HelloWorld" tab. The archetype generates the following files in case of the HelloWorld example:

Resource	Description
`Activator.java`	The bundle activator
`CustomPanelController.java`	The controller of the whole panel
`CustomPanel.java`	The form backing object of the whole panel
`CustomTabController.java`	The controller of the HelloWorld tab contained by the panel
`CustomTabFBO.java`	The form backing object of the HelloWorld tab contained by the panel
`customTab.jspf`	The JSP that renders the HelloWorld tab
`messages_en_US.properties`	The US English language file
`messages_nl_NL.properties`	The Dutch language file

This part explains how to enhance this component with an additional field named "Author". To add this field to the HelloWorld panel component, perform the following steps:

1. Add the property to the form backing object of the tab on which the field should be displayed (`CustomTabFBO.java`).
2. Add a label to the language files to translate "Author" into the supported languages.
3. Add the field to the edit JSP (`customTab.jspf`).

In this example we do not persist the author. If we do want to persist the value we would define a business object (which has nothing to do with this panel) and persist the entered value in the `onSubmit` of the tab controller.

Adding a Property to the Form Backing Object

Since the author must be editable in the panel within the Editor, we also add the field to the form backing object (`CustomTabFBO.java`):

```
private String myAuthor;
public String getAuthor() {
        return myAuthor;
 }
 public void setAuthor(String author) {
        myAuthor = author;
 }
```

Language Label

See Adding a Language Label for a complete description of adding language labels. The following snippet should be added to each language file:

```
helloworldpanel.author=Author
```

When adding language labels, be sure that they conform to the development guidelines (G027 and G028 in particular).

The following snippet should be added to the `customTab.jspf` (see Adding a Field to the Edit JSP for a complete description of adding a field to the edit JSP):

```
<fmt:message key="helloworldpanel.author" />:
<wmedit:input path="command.author" />
```

Back to Top

Adding a field to a Media Item component

In Quick Start it is explained how to create a very basic media item component using the media item archetype. The archetype generates the following files in case of the HelloWorld example:

Resource	Description
`Activator.java`	Bundle activator
`CustomMediaItemController.java`	Controller of the metadata part of the custom media item
`CustomMediaItemVersion.java`	Interface representing the custom media item version
`CustomMediaItemVersionFBO.java`	Form backing object for the custom metadata part of the custom media item version
`CustomMediaItemVersionImpl.java`	Implementation of the custom media item version
`editMetadata.xml`	Descriptor of the JSP that renders the custom metadata part of the custom media item
`editMetadata.jspf`	JSP that renders the custom metadata part of the custom media item
`messages_en_US.properties`	US English language file
`messages_nl_NL.properties`	Dutch language file
`custommediaitem.gif`	Icon associated with the media item. Used in the menu and Media Repository search results.

This part explains how to enhance this component with an additional metadata field named "Author". To add this field to the custom metadata part of the custom Media item component, perform the following steps:

1. Add the property to the Media item version interface (`CustomMediaItemVersion.java`).
2. Add the property to the Media item version (`CustomMediaItemVersionImpl.java`).
3. Add the property to the form backing object (`CustomMediaItemVersionFBO.java`).
4. Add a label to the language files to translate “Author” in the supported languages.
5. Add the field to the edit JSP (`editMetadata.jspf`).

Adding a Property to the Business Object

The Business object "`CustomMediaItemVersion`" represents the data model of the HelloWorld media item. Since we want the author metadata field to be persisted we add this property to this class. We define the property as instance variable and add a getter and setter for it:

`CustomMediaItemVersion.java`

```
/**
 * Returns the author of the custom media item.
 * @return the author of the custom media item
 */
String getAuthor();

/**
 * Sets the author of the custom media item.
 * @param name the author of the custom media item to set
 */
void setAuthor(String author);
```

`CustomMediaItemVersionImpl.java`

```
@Property
public String getAuthor() {
        return JcrUtil.getString(getPrivateNode(), WCBConstants.NAMESPACE_PREFIX + ":author");
}

public void setAuthor(String author) {
        JcrUtil.setString(getPrivateNode(), WCBConstants.NAMESPACE_PREFIX + ":author", author);
}
```

Since the Author also must be editable in the Editor, we also add the metadata field to the Form backing object of the metadata part (`CustomMediaItemVersionFBO.java`). This is done in a similar way as we did it for the business object:

```
 private String myAuthor;

 public String getAuthor() {
        return myAuthor;
 }

 public void setAuthor(String author) {
        myAuthor = author;
 }
```

Adding a Field to the Edit JSP

See Adding a Field to the Edit JSP for a complete description of adding fields to the edit JSP. The following snippet should be added to the `editMetadata.jspf`:

```
<fmt:message key="helloworldmediaitem.author" />:
<wmedit:input path="author"/>
```

Language Label

The language label "helloworldmediaitem.author should" be present in both `messages_en_US.properties` and `messages_en_US.properties` files as e.g.:

```
helloworldmediaitem.author=Author
```

or

```
helloworldmediaitem.author=Auteur
```

See Adding a Language Label for a complete description of adding language labels. When adding language labels, be sure that they conform to the Plugin Development Guidelines (G027 and G028 in particular).

Back to Top

Setting the Correct Permission Level for HTML and JavaScript

When you add fields to a custom element, page metadata, custom content type and/or a panel that accept HTML or JavaScript and/or are rendered without any escaping mechanism, they should only be accessible to roles with at least the Administrator permission group assigned.

Back to Top