CONFIGURING AND CUSTOMIZING THE TOOLBAR OPTIONS

CUSTOMIZING TOOLBAR OPTIONS

- Liferay supports many different kinds of WYSIWYG editors that can be used in applications to edit content.
- Depending on the content you're editing, you may want to modify the editor to provide a better configuration for your needs.
- Let's look at how to extend the Liferay-supported WYSIWYG editor to add new or modify existing configurations.

EXTENDING THE EDITOR'S CONFIGURATION

- To modify the editor's configuration, you can create a module that has a component that implements the `EditorConfigContributor` interface.
- When you implement this interface, your module will provide a service that modifies the editors you'd like to change.
- A simple example of this is provided below.

EXTENDING TOOLBAR OPTIONS EXAMPLE

- You can create a generic OSGi module using your favorite third-party tool or using the *Blade CLI*.
- A Java class in a module's unique package should extend the `BaseEditorConfigContributor` class.
- Directly above the class's declaration, the following component annotation can be inserted:

```
@Component(
    property = {

    },

    service = EditorConfigContributor.class
)
```

ADDING VIDEO AND CAMERA BUTTONS

- For this example, we will see how to add the video and camera buttons to the Web Content's AlloyEditor. The declaration would look like this:

```
import com.liferay.journal.constants.JournalPortletKeys;

@Component(
    property = {
        "editor.name=alloyeditor",
        "javax.portlet.name=" + JournalPortletKeys.JOURNAL,
        "service.ranking:Integer=100"
    },
    service = EditorConfigContributor.class
)
public class CustomJournalMediaEditorConfigContributor
    extends BaseEditorConfigContributor {
}
```

SPECIFYING CHANGES

- The following method could be added to specify changes.

```
@Override
public void populateConfigJSONObject(
    JSONObject jsonObject, Map<String, Object> inputEditorTaglibAttributes,
    ThemeDisplay themeDisplay,
    RequestBackedPortletURLFactory requestBackedPortletURLFactory) {
}
```

- In the `populateConfigJSONObject` method, you need to instantiate a JSONObject that holds the current configuration of the editor. For instance, you could do something like this:

```
JSONObject toolbarsJSONObject = jsonObject.getJSONObject("toolbars");
```

MODIFYING CONFIGURATION

- With the `JSONObject` holding the editor's configuration, the configuration can be modified.
- Suppose you'd like to add a button to your editor's toolbar.
- To complete this, you'd need to extract the Add buttons out of your toolbar configuration object as a JSONArray, and then add the button to that JSONArray.

ADDING CAMERA CODE

- For example, the following code would add a Camera button to the editor's toolbar:

```
if (toolbarsJSONObject != null) {
    JSONObject addJSONObject = toolbarsJSONObject.getJSONObject("add");

    if (addJSONObject != null) {
        JSONArray buttonsJSONArray = addJSONObject.getJSONArray("buttons");

        buttonsJSONArray.put("camera");
        buttonsJSONArray.put("video");
    }

    addJSONObject.put("buttons", buttonsJSONArray);

    toolbarsJSONObject.put("add", addJSONObject);
}
```

FINISHING UP

- All together it should look like the `02-toolbar-customization-example.java` file in your exercises folder.
- Now if we create a new web content article, the AlloyEditor will have our new options.

Notes: