

## TASK-6 CODE

### QUESTION:

- 1- Create a component named "Single Article" and keep two field
  - i) Enable PII (Checkbox)
  - ii) Css Class (Text field)
- 2- If you enable the checkbox then the Css class text field should be enable inside the component dialog and if you disabled the checkbox then field should be disabled user should not be able to enter text in it.
- 3- Create the div element and for the div element the css class name should appear if value is entered.

### ANSWER:

#### JS CODE FOR CHECKBOX ENABLE

```
(function (document, $) {  
    "use strict";  
    console.log("welcome to the page");  
    // when dialog gets injected  
    $(document).on("foundation-contentloaded", function (e) {  
        // if there is already an initial value make sure the according target element becomes visible  
        checkboxShowHideHandler($(".cq-dialog-checkbox-showhide", e.target));  
    });  
    $(document).on("change", ".cq-dialog-checkbox-showhide", function (e) {  
        checkboxShowHideHandler($(this));  
    });  
});
```

```

function checkboxShowHideHandler(el) {
    el.each(function (i, element) {
        if ($(element).is("coral-checkbox")) {
            // handle Coral3 base drop-down
            Coral.commons.ready(element, function (component) {
                showHide(component, element);
                component.on("change", function () {
                    showHide(component, element);
                });
            });
        } else {
            // handle Coral2 based drop-down
            var component = $(element).data("checkbox");
            if (component) {
                showHide(component, element);
            }
        }
    });
}

function showHide(component, element) {
    console.log('showing');
    // get the selector to find the target elements. it's stored as a data- attribute
    var target = $(element).data("cqDialogCheckboxShowhideTarget");
    var $target = $(target);
    if (target) {
        if (component.checked) {
            $target.find('input, select, textarea, button').prop('disabled', false);
        } else {

```

```

    $target.find('input, select, textarea, button').prop('disabled', true).val(null);
    // After disabling the fields, send the component path to the servlet
    var componentPath = $target.closest("form.cq-dialog").attr("action");
    console.log(componentPath);
    sendPathToServlet(componentPath);
  }
}
}
})(document, Granite.$);

```

## SINGLE ARTICLE HTML CODE FOR CHECKBOX

```

<h1>Hello Article</h1>
<div> Enable PII: ${properties.enablePII ? 'checked' : 'unchecked'} </div>
<div data-sly-test.cssClass="${properties.enablePII}">
  <p data-sly-test="${cssClass}">Css Class : ${properties.cssClass}</p>

</div>

```

## CHECKBOX UNABLE AND TEXT FIELD DISABLED

Properties

?

×

✓

☐ Enable PII

CSS Class

#### CHECKBOX ENABLE AND TEXTFIELD ENABLED

Properties

?

×

✓

☒ Enable PII

CSS Class

HEMANTH

#### COMPONENT DISPLAY ON PAGE

✎

✂

🗑

+

↔

# Hello Article

Enable PII: checked  
Css Class : HEMANTH

## **SERVLET CODE FOR DELETETING TEXTFIELD**

```
package com.adobe.aem.guides.demo.core.servlets;
```

```
import java.io.IOException;
```

```
import javax.servlet.Servlet;
```

```
import org.apache.sling.api.SlingHttpServletRequest;
```

```
import org.apache.sling.api.SlingHttpServletResponse;
```

```
import org.apache.sling.api.resource.ModifiableValueMap;
```

```
import org.apache.sling.api.resource.PersistenceException;
```

```
import org.apache.sling.api.resource.Resource;
```

```
import org.apache.sling.api.resource.ResourceResolver;
```

```
import org.apache.sling.api.servlets.SlingAllMethodsServlet;
```

```
import org.osgi.service.component.annotations.Component;
```

```
@Component(
```

```
    service = Servlet.class,
```

```
    property = {
```

```
        "sling.servlet.methods=POST",
```

```
        "sling.servlet.paths=/bin/deleteContentNode"
```

```
    })
```

```
public class Cheching extends SlingAllMethodsServlet {
```

```
    protected void doPost(SlingHttpServletRequest request, SlingHttpServletResponse  
    response) throws IOException {
```

```
        String path = request.getParameter("path");
```

```
        if (path == null || path.isEmpty()) {
```

```

        response.setStatus(SlingHttpServletResponse.SC_BAD_REQUEST);
        response.getWriter().write("Path parameter is missing");
        return;
    }

    // Replace _jcr_content with jcr:content
    String modifiedPath = path.replace("_jcr_content", "jcr:content");

    // Print the received path and modified path
    response.getWriter().write("Received path: " + path + "\n");
    response.getWriter().write("Modified path: " + modifiedPath + "\n");
    ResourceResolver resolver = request.getResourceResolver();
    try {
        Resource resource = resolver.getResource(modifiedPath);
        if (resource != null) {
            ModifiableValueMap valueMap = resource.adaptTo(ModifiableValueMap.class);
            if (valueMap != null) {
                // Remove the "cssClass" property if it exists
                if (valueMap.containsKey("textfield")) {
                    valueMap.remove("textfield");
                }
                resolver.commit();
                response.setStatus(SlingHttpServletResponse.SC_OK);
                response.getWriter().write("Content node deleted and 'cssClass' property removed if it existed.");
            } else {
                response.setStatus(SlingHttpServletResponse.SC_INTERNAL_SERVER_ERROR);
                response.getWriter().write("Unable to adapt resource to ModifiableValueMap.");
            }
        }
    }

```

```
    }  
    } else {  
        response.setStatus(SlingHttpServletResponse.SC_NOT_FOUND);  
        response.getWriter().write("Content node not found.");  
    }  
    } catch (PersistenceException e) {  
        response.setStatus(SlingHttpServletResponse.SC_INTERNAL_SERVER_ERROR);  
        response.getWriter().write("PersistenceException: " + e.getMessage());  
    }  
}}
```