Application Presentation and Customization

Learning Objectives

- Learn about portlet actions and their effects on an application
- Explore the benefits of using tag libraries
- Understand the process of enabling validation in an application

Tasks to Accomplish

- Create the Gradebook Web Module
- Implement the Main View
- Implement the Assignment Editing View
- Implement Validation

Exercise Prerequisites

- Java JDK installed to run Liferay
  - Download here: https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
  - Instructions on installation here: https://www.java.com/en/download/help/download_options.xml
- Preferred development tools (e.g. Blade CLI, Gradle, IntelliJ IDEA with Liferay plugin, etc.) installed with the "Gradebook Workspace" already created
  - This was done in the first training module
- Exercise Prereqs added to workspace or previous training modules completed

Table of Contents

Create the Gradebook Web Module

Exercise Goals

- Create a Liferay MVC portlet module
- Declare dependencies
- Set portlet properties
- Set the portlet name
- Do a final code review
- Deploy the module
- Test the module

Over the next few exercises, we will create the user interface for the Gradebook application. We will be using coding conventions and patterns recommended for Liferay development, leveraging libraries, components, and high-level superclasses to remove the need for boilerplate coding.

We will use the Liferay MVC portlet as a portlet component. The portlet lifecycle and communication between the portlet back-end and user interface will be handled by MVC command components.

The user interface will be implemented with JSP technology. We will be using Liferay tag libraries, which both minimize the need for HTML coding and guarantee a Twitter Bootstrap-based responsive layout.

Create a Liferay MVC Portlet Module

1. **Create** a Liferay MVC Portlet Module in the *modules/gradebook* subfolder with the following attributes:
    - **Project Name**: "gradebook-web"
    - **Project Template**: mvc-portlet
    - **Component Class Name**: "Gradebook"
    - **Package Name**: "com.liferay.training.gradebook.web"

If you're using Blade CLI, the command to create this module is `blade create -t mvc-portlet -p com.liferay.training.gradebook.web -c Gradebook gradebook-web`.

We need to declare dependencies for the Gradebook service (API), Liferay Clay tag library, and Petra function utility:

Declare Dependencies

1. Open the `build.gradle` of *gradebook-web* project.

2. Implement the new dependencies as follows:

```
// Clay taglib.
compileOnly group: 'com.liferay', name: 'com.liferay.frontend.taglib.clay'
// Needed for the Assignments Management Toolbar.
compileOnly group: 'com.liferay', name: 'com.liferay.petra.function'
compileOnly group: "com.liferay.portal", name: "com.liferay.portal.kernel"
compileOnly group: "com.liferay.portal", name: "com.liferay.util.taglib"
compileOnly group: "javax.portlet", name: "portlet-api"
compileOnly group: "javax.servlet", name: "javax.servlet-api"
compileOnly group: "jstl", name: "jstl"
compileOnly group: "org.osgi", name: "org.osgi.service.component.annotations
// Gradebook service.
compileOnly project(":modules:gradebook:gradebook-api")
```

> Note: here how we reference the API (gradebook-api) and not the implementation (gradebook-service). What is Petra? If you developed for pre-7 Liferay, you probably remember the `com.liferay.util.java` utilities. The Petra library family contains the modularized and OSGi-ready versions of those utilities.

We'll have the following requirements for our portlet:

- We don't want the Gradebook portlet to be instanceable, as its data needs to be scoped under a site.
- We'd like the Gradebook portlet to appear in the *Liferay Training* Widgets category instead of the *Sample* category.

Let's change the portlet component properties to match these requirements:

Set Portlet Properties

1. **Open** the `GradebookPortlet` class.
2. **Implement** the changes to component properties as follows:

```
"com.liferay.portlet.display-category=category.training",
"com.liferay.portlet.instanceable=false",
```

It's a good practice to use a fully qualified name of the portlet class as the portlet identifier. We also have to update the name in our resource bundle (we'll discuss localization at later steps):

Set the Portlet Name

1. **Open** the class `com.liferay.training.gradebook.web.constants.GradebookPortletKeys`.
2. **Update** the portlet name constant as follows:

```
public static final String GRADEBOOK = "com_liferay_training_gradebook_web_
```

3. **Open** the file `src/main/resources/content/Language.properties`.
4. **Implement** the contents as follows:

javax.portlet.description.com_liferay_training_gradebook_web_portlet_GradebookPortlet=GRA

javax.portlet.display-name.com_liferay_training_gradebook_web_portlet_GradebookPortlet=GR

javax.portlet.keywords.com_liferay_training_gradebook_web_portlet_GradebookPortlet=GRADEB

javax.portlet.short-title.com_liferay_training_gradebook_web_portlet_GradebookPortlet=GRA

javax.portlet.title.com_liferay_training_gradebook_web_portlet_GradebookPortlet=GRADEBOOK

Do a Final Code Review

build.gradle

```
dependencies {
	// Clay taglib.

	compileOnly group: 'com.liferay', name: 'com.liferay.frontend.taglib.clay'

	// Needed for the Assignments Management Toolbar.

	compileOnly group: 'com.liferay', name: 'com.liferay.petra.function'

	compileOnly group: "com.liferay.portal", name: "com.liferay.portal.kernel"
	compileOnly group: "com.liferay.portal", name: "com.liferay.util.taglib"
	compileOnly group: "javax.portlet", name: "portlet-api"
	compileOnly group: "javax.servlet", name: "javax.servlet-api"
	compileOnly group: "jstl", name: "jstl"
	compileOnly group: "org.osgi", name: "org.osgi.service.component.annotations"

	// Gradebook service.

	compileOnly project(":modules:gradebook:gradebook-api")
}
```

GradebookPortlet.java

```
package com.liferay.training.gradebook.web.portlet;

import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;

import com.liferay.portal.kernel.portlet.bridges.mvc.MVCPortlet;

import javax.portlet.Portlet;

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

/**
 * @author liferay
 */
@Component(
        immediate = true,
        property = {
                "com.liferay.portlet.display-category=category.training",
                "com.liferay.portlet.instanceable=false",
                "javax.portlet.init-param.template-path=/",
                "javax.portlet.init-param.view-template=/view.jsp",
                "javax.portlet.name=" + GradebookPortletKeys.GRADEBOOK,
                "javax.portlet.resource-bundle=content.Language",
                "javax.portlet.security-role-ref=power-user,user"
        },
        service = Portlet.class
)
public class GradebookPortlet extends MVCPortlet {
}
```

GradebookPortletKeys.java

```
package com.liferay.training.gradebook.web.constants;


/**
 * @author liferay
 */
public class GradebookPortletKeys {

        public static final String GRADEBOOK = "com_liferay_training_gradebook_web_portle
}
```

Language.properties

```
javax.portlet.description.com_liferay_training_gradebook_web_portlet_GradebookPortlet=GRA
javax.portlet.display-name.com_liferay_training_gradebook_web_portlet_GradebookPortlet=GR
javax.portlet.keywords.com_liferay_training_gradebook_web_portlet_GradebookPortlet=GRADEB
javax.portlet.short-title.com_liferay_training_gradebook_web_portlet_GradebookPortlet=GRA
javax.portlet.title.com_liferay_training_gradebook_web_portlet_GradebookPortlet=GRADEBOOK
gradebook.caption=Hello from Gradebook!
```

Deploy the Module

1. **Start** your Docker container if it's not running.
2. **Run** `../gradlew deploy` to deploy the module.
    - You should see the following message in the log:

        ```
        STARTED com.liferay.training.gradebook.web_1.0.0
        ```

Test the Module

1. **Open** your browser to http://localhost:8080.
2. **Click** the *Edit* icon on the top right corner of the page.
3. **Expand** the *category.training* category in the *Widgets* menu.
4. **Add** the *Gradebook* portlet on the page.

Implement the Main View

Exercise Goals

- Implement the JSP files
- Implement the MVC render command for showing the Assignment list
- Implement the MVC Render command for showing a single Assignment
- Implement the back-end class for the UI management toolbar
- Test the user interface

By convention, the `init.jsp` file is used to centralize imports, taglib declarations, variable initializations, and any common tasks for all the user interface JSP files. The `init.jsp` is then included in the other JSP files, like `view.jsp`.

We'll use taglib declarations for Clay and Liferay Front-End, Liferay Item Selector, as well as imports for the classes we will be using in the front-end implementation.

Implement the init.jsp

1. **Update** the contents of `src/main/resources/META-INF/resources/init.jsp` with:

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


<%@ taglib uri="http://java.sun.com/portlet_2_0" prefix="portlet" %>
```

```
<%@ taglib prefix="aui" uri="http://liferay.com/tld/aui" %>
<%@ taglib prefix="clay" uri="http://liferay.com/tld/clay" %>
<%@ taglib prefix="liferay-item-selector" uri="http://liferay.com/tld/item-selector" %>
<%@ taglib prefix="liferay-frontend" uri="http://liferay.com/tld/frontend" %>
<%@ taglib prefix="liferay-portlet" uri="http://liferay.com/tld/portlet" %>
<%@ taglib prefix="liferay-theme" uri="http://liferay.com/tld/theme" %>
<%@ taglib prefix="liferay-ui" uri="http://liferay.com/tld/ui" %>

<%@ page import="java.util.Date"%>
<%@ page import="javax.portlet.WindowState"%>

<%@ page import="com.liferay.portal.kernel.language.LanguageUtil"%>
<%@ page import="com.liferay.portal.kernel.portlet.LiferayWindowState"%>
<%@ page import="com.liferay.portal.kernel.util.HtmlUtil"%>

<%@ page import="com.liferay.training.gradebook.model.Assignment"%>
<%@ page import="com.liferay.training.gradebook.web.constants.MVCCommandNames"%>

<liferay-frontend:defineObjects />

<liferay-theme:defineObjects />

<portlet:defineObjects />
```

The `view.jsp` implements the Assignments list view. We'll use the Management Toolbar from Clay as well the Search Container from the Liferay UI tag library to implement the view.

Implement the view.jsp

1. **Update** the contents of portlet *VIEW* mode JSP `src/main/resources/META-INF/resources/view.jsp` with:

```
<%@ include file="init.jsp"%>
```

```
<div class="container-fluid-1280">

	<h1><liferay-ui:message key="Assignments" /></h1>

	<%-- Clay management toolbar. --%>

	<clay:management-toolbar
		disabled="${assignmentCount eq 0}"
		displayContext="${assignmentsManagementToolbarDisplayContext}"
		itemsTotal="${assignmentCount}"
		searchContainerId="assignmentEntries"
		selectable="false"
	/>

	<%-- Search container. --%>

	<liferay-ui:search-container
		emptyResultsMessage="no-assignments"
		id="assignmentEntries"
		iteratorURL="${portletURL}"
		total="${assignmentCount}">

		<liferay-ui:search-container-results results="${assignments}" />

		<liferay-ui:search-container-row
			className="com.liferay.training.gradebook.model.Assignment"
			modelVar="entry">

			<%@ include file="/assignment/entry_search_columns.jspf" %>

		</liferay-ui:search-container-row>
```

```
                <%-- Iterator / Paging --%>

                <liferay-ui:search-iterator
                        displayStyle="${assignmentsManagementToolbarDisplayContext.getDis
                        markupView="lexicon"
                />
        </liferay-ui:search-container>
</div>
```

We need three more JSP files to display a single row on the assignment list, show available actions, and display the *details* view.

Implement the Other JSP Files

1. **Create** a subfolder `src/main/resources/META-INF/resources/assignment`.
2. **Implement** the following three files (notice the JSP fragment suffix `.jspf` in `entry_search_columns.jspf`) in the new subfolder:

entry_search_columns.jspf

```
<%-- Generate assignment view  URL. --%>

<portlet:renderURL var="viewAssignmentURL">
        <portlet:param name="mvcRenderCommandName" value="<%=MVCCommandNames.VIEW_ASSIGNM
        <portlet:param name="redirect" value="${currentURL}" />
        <portlet:param name="assignmentId" value="${entry.assignmentId}" />
</portlet:renderURL>

<c:choose>

        <%-- Descriptive (list) view --%>

        <c:when
```

```
test='${assignmentsManagementToolbarDisplayContext.getDisplayStyle().equa

<%-- User --%>

<liferay-ui:search-container-column-user
        showDetails="<%=false%>"
        userId="<%=entry.getUserId()%>"
/>

<liferay-ui:search-container-column-text colspan="<%=2%>">

        <%
                String modifiedDateDescription =
                        LanguageUtil.getTimeDescription(
                                        request, System.currentTimeMillis
                                        - entry.getModifiedDate().getTime
        %>

        <h5 class="text-default">
                <liferay-ui:message
                        arguments="<%=new String[] {entry.getUserName(),
                        key="x-modified-x-ago" />
        </h5>

        <h4>
                <aui:a href="${viewAssignmentURL}">
                        ${entry.title}
                </aui:a>
        </h4>

</liferay-ui:search-container-column-text>
```

```
        <liferay-ui:search-container-column-jsp
                path="/assignment/entry_actions.jsp" />
</c:when>

<%-- Card view  --%>

<c:when
        test='${assignmentsManagementToolbarDisplayContext.getDisplayStyle().equa

        <%--
                row.setCssClass("lfr-asset-item");
        --%>

        <liferay-ui:search-container-column-text>

                <%-- Vertical card. --%>

                <liferay-frontend:icon-vertical-card
                        actionJsp="/assignment/entry_actions.jsp"
                        actionJspServletContext="<%= application %>"
                        icon="cards2" resultRow="${row}"
                        title="${entry.title}"
                        url="${viewAssignmentURL}">

                        <liferay-frontend:vertical-card-sticker-bottom>

                                <liferay-ui:user-portrait
                                        cssClass="sticker sticker-bottom"
                                        userId="${entry.userId}"
                                />
                        </liferay-frontend:vertical-card-sticker-bottom>
```

```
                        <liferay-frontend:vertical-card-footer>

                                <div class="truncate-text">

                                        <%-- Strip HTML --%>

                                        <%=HtmlUtil.stripHtml(entry.getDescript
                                </div>
                        </liferay-frontend:vertical-card-footer>
                </liferay-frontend:icon-vertical-card>
        </liferay-ui:search-container-column-text>
</c:when>

<%-- Table view --%>

<c:otherwise>

        <liferay-ui:search-container-column-text
                href="${viewAssignmentURL}"
                name="title"
                value="<%= entry.getTitle() %>"
        />

        <liferay-ui:search-container-column-user
                name="author"
                userId="${entry.userId}"
        />

        <liferay-ui:search-container-column-date
                name="create-date"
                property="createDate"
        />
```

```
		<liferay-ui:search-container-column-jsp
			name="actions"
			path="/assignment/entry_actions.jsp"
		/>
	</c:otherwise>
</c:choose>
```

entry_actions.jsp

```
<%@ include file="../init.jsp"%>

<c:set var="assignment" value="${SEARCH_CONTAINER_RESULT_ROW.object}" />

<liferay-ui:icon-menu markupView="lexicon">

	<%-- View action. --%>

	<portlet:renderURL var="viewAssignmentURL">
		<portlet:param name="mvcRenderCommandName"
			value="<%=MVCCommandNames.VIEW_ASSIGNMENT %>" />
		<portlet:param name="redirect" value="${currentURL}" />
		<portlet:param name="assignmentId" value="${assignment.assignmentId}" />
	</portlet:renderURL>

	<liferay-ui:icon message="view" url="${viewAssignmentURL}" />

	<%-- Edit action. --%>

	<portlet:renderURL var="editAssignmentURL">
		<portlet:param name="mvcRenderCommandName"
			value="<%=MVCCommandNames.EDIT_ASSIGNMENT %>" />
```

```
            <portlet:param name="redirect" value="${currentURL}" />
            <portlet:param name="assignmentId" value="${assignment.assignmentId}" />
        </portlet:renderURL>

        <liferay-ui:icon message="edit" url="${editAssignmentURL}" />

        <%-- Delete action. --%>

        <portlet:actionURL name="<%=MVCCommandNames.DELETE_ASSIGNMENT %>" var="deleteAss
            <portlet:param name="redirect" value="${currentURL}" />
            <portlet:param name="assignmentId" value="${assignment.assignmentId}" />
        </portlet:actionURL>

        <liferay-ui:icon-delete url="${deleteAssignmentURL}" />

</liferay-ui:icon-menu>
```

view_assignment.jsp

```
<%@ include file="../init.jsp"%>

<div class="container-fluid-1280">

	<h1>${assignment.title}</h1>

	<h2><liferay-ui:message key="assignment-information" /></h2>

	<div class="assignment-metadata">

		<dl>
			<dt><liferay-ui:message key="created" /></dt>
			<dd>${createDate}</dd>

			<dt><liferay-ui:message key="created-by" /></dt>
			<dd>${assignment.userName}</dd>

			<dt><liferay-ui:message key="assignment-duedate" /></dt>
			<dd>${dueDate}</dd>

			<dt><liferay-ui:message key="description" /></dt>
			<dd>${assignment.description}</dd>
		</dl>
	</div>
</div>
```

Now we have the JSP files in place and need *MVC Command* components to take care of the portlet lifecycle handling and interaction between the user interface and back-end. MVC Commands respond to portlet URLs, which in JSP files are generated with the `<portlet>` tag library.

At this stage, we need two MVC Render Commands: one for displaying the Assignments list and one for displaying a single Assignment.

Before implementing our MVC render commands, implement a constants class to hold the command names. This is a good practice to reduce the risk of typos when referencing the command names.

Implement MVCCommandNames.java

1. **Create** a class `com.liferay.training.gradebook.web.constants.MVCCommandNames`.
2. **Implement** as follows:

```
package com.liferay.training.gradebook.web.constants;

/**
 * @author liferay
 *
 */
public class MVCCommandNames {
	public static final String ADD_ASSIGNMENT = "/gradebook/assignment/add";
	public static final String DELETE_ASSIGNMENT = "/gradebook/assignment/delete";
	public static final String EDIT_ASSIGNMENT = "/gradebook/assignment/edit";
	public static final String VIEW_ASSIGNMENT = "/gradebook/assignment/view";
	public static final String VIEW_ASSIGNMENTS = "/gradebook/assignments/view";
	// These are used for the optional exercise "Implement Submissions".
	public static final String ADD_SUBMISSION = "/gradebook/submission/add";
	public static final String DELETE_SUBMISSION = "/gradebook/submission/delete";
	public static final String EDIT_SUBMISSION = "/gradebook/submission/edit";
	public static final String GRADE_SUBMISSION = "/gradebook/submission/grade";
	public static final String VIEW_SUBMISSION = "/gradebook/submission/view";
}
```

Implement ViewAssignmentsMVCRenderCommand.java

1. **Create** a class `com.liferay.training.gradebook.web.portlet.action.ViewAssignmentsMVCRenderCommand`.
2. **Implement** as follows. Don't worry about the errors with `AssignmentsManagementToolbarDisplayContext`. We'll add that in the next step:

```
package com.liferay.training.gradebook.web.portlet.action;

import com.liferay.portal.kernel.dao.search.SearchContainer;
import com.liferay.portal.kernel.portlet.LiferayPortletRequest;
import com.liferay.portal.kernel.portlet.LiferayPortletResponse;
import com.liferay.portal.kernel.portlet.bridges.mvc.MVCRenderCommand;
import com.liferay.portal.kernel.theme.ThemeDisplay;
import com.liferay.portal.kernel.util.OrderByComparator;
import com.liferay.portal.kernel.util.OrderByComparatorFactoryUtil;
import com.liferay.portal.kernel.util.ParamUtil;
import com.liferay.portal.kernel.util.Portal;
import com.liferay.portal.kernel.util.WebKeys;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.AssignmentService;
import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;
import com.liferay.training.gradebook.web.constants.MVCCommandNames;
import com.liferay.training.gradebook.web.display.context.AssignmentsManagementToolbarDis

import java.util.List;

import javax.portlet.PortletException;
import javax.portlet.RenderRequest;
import javax.portlet.RenderResponse;

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

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

/**
 * MVC command for showing the assignments list.
 *
 * @author liferay
 */
@Component(
        immediate = true,
        property = {
                "javax.portlet.name=" + GradebookPortletKeys.GRADEBOOK,
                "mvc.command.name=/",
                "mvc.command.name=" + MVCCommandNames.VIEW_ASSIGNMENTS
        },
        service = MVCRenderCommand.class
)
public class ViewAssignmentsMVCRenderCommand implements MVCRenderCommand {

        @Override
        public String render(
                RenderRequest renderRequest, RenderResponse renderResponse)
                throws PortletException {

                // Add assignment list related attributes.

                addAssignmentListAttributes(renderRequest);

                // Add Clay management toolbar related attributes.

                addManagementToolbarAttributes(renderRequest, renderResponse);

                return "/view.jsp";
```

```
}

/**
 * Adds assigment list related attributes to the request.
 *
 * @param renderRequest
 */
private void addAssignmentListAttributes(RenderRequest renderRequest) {

        ThemeDisplay themeDisplay =
                (ThemeDisplay) renderRequest.getAttribute(WebKeys.THEME_DISPLAY);

        // Resolve start and end for the search.

        int currentPage = ParamUtil.getInteger(
                renderRequest, SearchContainer.DEFAULT_CUR_PARAM,
                SearchContainer.DEFAULT_CUR);

        int delta = ParamUtil.getInteger(
                renderRequest, SearchContainer.DEFAULT_DELTA_PARAM,
                SearchContainer.DEFAULT_DELTA);

        int start = ((currentPage > 0) ? (currentPage - 1) : 0) * delta;
        int end = start + delta;

        // Get sorting options.
        // Notice that this doesn't really sort on title because the field is
        // stored in XML. In real world this search would be integrated to the
        // search engine  to get localized sort options.

        String orderByCol =
                ParamUtil.getString(renderRequest, "orderByCol", "title");
```

```
        String orderByType =
                ParamUtil.getString(renderRequest, "orderByType", "asc");

        // Create comparator

        OrderByComparator<Assignment> comparator =
                OrderByComparatorFactoryUtil.create(
                        "Assignment", orderByCol, !("asc").equals(orderByType));

        // Get keywords.
        // Notice that cleaning keywords is not implemented.

        String keywords = ParamUtil.getString(renderRequest, "keywords");

        // Call the service to get the list of assignments.

        List<Assignment> assignments =
                _assignmentService.getAssignmentsByKeywords(
                        themeDisplay.getScopeGroupId(), keywords, start, end,
                        comparator);

        // Set request attributes.

        renderRequest.setAttribute("assignments", assignments);
        renderRequest.setAttribute(
                "assignmentCount", _assignmentService.getAssignmentsCountByKeywo
                        themeDisplay.getScopeGroupId(), keywords));

}

/**
 * Adds Clay management toolbar context object to the request.
```

```
	 *
	 * @param renderRequest
	 * @param renderResponse
	 */
	private void addManagementToolbarAttributes(
			RenderRequest renderRequest, RenderResponse renderResponse) {

			LiferayPortletRequest liferayPortletRequest =
				_portal.getLiferayPortletRequest(renderRequest);

			LiferayPortletResponse liferayPortletResponse =
				_portal.getLiferayPortletResponse(renderResponse);

			AssignmentsManagementToolbarDisplayContext assignmentsManagementToolbarDi
				new AssignmentsManagementToolbarDisplayContext(
					liferayPortletRequest, liferayPortletResponse,
					_portal.getHttpServletRequest(renderRequest));

			renderRequest.setAttribute(
				"assignmentsManagementToolbarDisplayContext",
				assignmentsManagementToolbarDisplayContext);

	}

	@Reference
	protected AssignmentService _assignmentService;

	@Reference
	private Portal _portal;
}
```

Implement the MVC Render Command for Showing a Single Assignment

1. **Create** a class `com.liferay.training.gradebook.web.portlet.action.ViewSingleAssignmentMVCRenderCommand`.
2. **Implement** as follows:

ViewSingleAssignmentMVCRenderCommand.java

```
package com.liferay.training.gradebook.web.portlet.action;


import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.portlet.bridges.mvc.MVCRenderCommand;
import com.liferay.portal.kernel.service.UserLocalService;
import com.liferay.portal.kernel.theme.PortletDisplay;
import com.liferay.portal.kernel.theme.ThemeDisplay;
import com.liferay.portal.kernel.util.DateFormatFactoryUtil;
import com.liferay.portal.kernel.util.ParamUtil;
import com.liferay.portal.kernel.util.Portal;
import com.liferay.portal.kernel.util.WebKeys;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.AssignmentService;
import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;
import com.liferay.training.gradebook.web.constants.MVCCommandNames;


import java.text.DateFormat;


import javax.portlet.PortletException;
import javax.portlet.RenderRequest;
import javax.portlet.RenderResponse;


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

```
/**
 * MVC Command for showing the assignment submissions list view.
 *
 * @author liferay
 */
@Component(
        immediate = true,
        property = {
                "javax.portlet.name=" + GradebookPortletKeys.GRADEBOOK,
                "mvc.command.name=" + MVCCommandNames.VIEW_ASSIGNMENT
        },
        service = MVCRenderCommand.class
)
public class ViewSingleAssignmentMVCRenderCommand implements MVCRenderCommand {

        @Override
        public String render(
                RenderRequest renderRequest, RenderResponse renderResponse)
                throws PortletException {

                ThemeDisplay themeDisplay =
                        (ThemeDisplay) renderRequest.getAttribute(WebKeys.THEME_DISPLAY);

                long assignmentId = ParamUtil.getLong(renderRequest, "assignmentId", 0);

                try {

                        // Call the service to get the assignment.

                        Assignment assignment =
                                _assignmentService.getAssignment(assignmentId);
```

```
                DateFormat dateFormat = DateFormatFactoryUtil.getSimpleDateFormat
                        "EEEEE, MMMMM dd, yyyy");

                // Set attributes to the request.

                renderRequest.setAttribute("assignment", assignment);
                renderRequest.setAttribute(
                        "dueDate", dateFormat.format(assignment.getDueDate()));
                renderRequest.setAttribute(
                        "createDate", dateFormat.format(assignment.getCreateDate(

                // Set back icon visible.

                PortletDisplay portletDisplay = themeDisplay.getPortletDisplay();

                String redirect = renderRequest.getParameter("redirect");

                portletDisplay.setShowBackIcon(true);
                portletDisplay.setURLBack(redirect);

                return "/assignment/view_assignment.jsp";

        }
        catch (PortalException pe) {
                throw new PortletException(pe);
        }
}

@Reference
private AssignmentService _assignmentService;
```

```
        @Reference
        private Portal _portal;

        @Reference
        private UserLocalService _userLocalService;
}
```

Our last task at this step is to implement the backing class for the Clay management toolbar. We won't go into implementation details with this class, but you can take a look at the document ation.

Implement the Back-End Class for the UI Management Toolbar

1. **Create** a class `com.liferay.training.gradebook.web.display.context.AssignmentsManagementToolbarDisplayContext`.
2. **Implement** as follows:

```
package com.liferay.training.gradebook.web.display.context;

import com.liferay.frontend.taglib.clay.servlet.taglib.display.context.BaseManagementToo
import com.liferay.frontend.taglib.clay.servlet.taglib.util.CreationMenu;
import com.liferay.frontend.taglib.clay.servlet.taglib.util.DropdownItem;
import com.liferay.frontend.taglib.clay.servlet.taglib.util.DropdownItemList;
import com.liferay.frontend.taglib.clay.servlet.taglib.util.ViewTypeItem;
import com.liferay.frontend.taglib.clay.servlet.taglib.util.ViewTypeItemList;
import com.liferay.portal.kernel.dao.search.SearchContainer;
import com.liferay.portal.kernel.language.LanguageUtil;
import com.liferay.portal.kernel.portlet.LiferayPortletRequest;
import com.liferay.portal.kernel.portlet.LiferayPortletResponse;
import com.liferay.portal.kernel.portlet.PortalPreferences;
import com.liferay.portal.kernel.portlet.PortletPreferencesFactoryUtil;
import com.liferay.portal.kernel.portlet.PortletURLUtil;
import com.liferay.portal.kernel.theme.ThemeDisplay;
```

```
import com.liferay.portal.kernel.util.ParamUtil;
import com.liferay.portal.kernel.util.Validator;
import com.liferay.portal.kernel.util.WebKeys;
import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;
import com.liferay.training.gradebook.web.constants.MVCCommandNames;

import java.util.List;

import javax.portlet.PortletException;
import javax.portlet.PortletURL;
import javax.servlet.http.HttpServletRequest;

/**
        * Assigments management toolbar display context.
        *
        * This class passes contextual information to the user interface
        * for the Clay management toolbar.
        *
        * @author liferay
        */
public class AssignmentsManagementToolbarDisplayContext
        extends BaseManagementToolbarDisplayContext {

        public AssignmentsManagementToolbarDisplayContext(
                LiferayPortletRequest liferayPortletRequest,
                LiferayPortletResponse liferayPortletResponse,
                HttpServletRequest httpServletRequest) {

                super(
                        liferayPortletRequest, liferayPortletResponse, httpServletRequest

                _portalPreferences = PortletPreferencesFactoryUtil.getPortalPreferences(
```

```
                        liferayPortletRequest);

        _themeDisplay = (ThemeDisplay)httpServletRequest.getAttribute(
                WebKeys.THEME_DISPLAY);
}

/**
        * Returns the creation menu for the toolbar
        * (plus sign on the management toolbar).
        *
        * @return creation menu
        */
public CreationMenu getCreationMenu() {

        // Create the menu.

        return new CreationMenu() {
                {
                        addDropdownItem(
                                dropdownItem -> {
                                        dropdownItem.setHref(
                                                liferayPortletResponse.createRend
                                                "mvcRenderCommandName", MVCCommar
                                                "redirect", currentURLObj.toStrir
                                        dropdownItem.setLabel(
                                                LanguageUtil.get(request, "add-as
                                });
                }
        };
}

@Override
```

```
public String getClearResultsURL() {
        return getSearchActionURL();
}

/**
        * Returns the assignment list display style.
        *
        * Current selection is stored in portal preferences.
        *
        * @return current display style
        */
public String getDisplayStyle() {

        String displayStyle = ParamUtil.getString(request, "displayStyle");

        if (Validator.isNull(displayStyle)) {
                displayStyle = _portalPreferences.getValue(
                        GradebookPortletKeys.GRADEBOOK, "assignments-display-sty
                        "descriptive");
        }
        else {
                _portalPreferences.setValue(
                        GradebookPortletKeys.GRADEBOOK, "assignments-display-sty
                        displayStyle);

                request.setAttribute(
                        WebKeys.SINGLE_PAGE_APPLICATION_CLEAR_CACHE, Boolean.TRUE
        }

        return displayStyle;
}
```

```
/**
        * Returns the sort order column.
        *
        * @return sort column
        */
public String getOrderByCol() {

        return ParamUtil.getString(request, "orderByCol", "title");
}

/**
        * Returns the sort type (ascending / descending).
        *
        * @return sort type
        */
public String getOrderByType() {

        return ParamUtil.getString(request, "orderByType", "asc");
}

/**
        * Returns the action URL for the search.
        *
        * @return search action URL
        */
@Override
public String getSearchActionURL() {

        PortletURL searchURL = liferayPortletResponse.createRenderURL();

        searchURL.setProperty(
                "mvcRenderCommandName", MVCCommandNames.VIEW_ASSIGNMENTS);
```

```
        String navigation = ParamUtil.getString(
                request, "navigation", "entries");
        searchURL.setParameter("navigation", navigation);

        searchURL.setParameter("orderByCol", getOrderByCol());
        searchURL.setParameter("orderByType", getOrderByType());

        return searchURL.toString();
}



/**
        * Returns the view type options (card, list, table).
        *
        * @return list of view types
        */
@Override
public List<ViewTypeItem> getViewTypeItems() {
        PortletURL portletURL = liferayPortletResponse.createRenderURL();

        portletURL.setParameter(
                        "mvcRenderCommandName", MVCCommandNames.VIEW_ASSIGNMENTS)

        int delta =
                        ParamUtil.getInteger(request, SearchContainer.DEFAULT_DEL

        if (delta > 0) {
                portletURL.setParameter("delta", String.valueOf(delta));
        }

        String orderByCol =
```

```
                ParamUtil.getString(request, "orderByCol", "title");
        String orderByType =
                ParamUtil.getString(request, "orderByType", "asc");

        portletURL.setParameter("orderByCol", orderByCol);
        portletURL.setParameter("orderByType", orderByType);

        int cur =
                ParamUtil.getInteger(request, SearchContainer.DEFAULT_CUR_PARAM);

        if (cur > 0) {
                portletURL.setParameter("cur", String.valueOf(cur));
        }

        return new ViewTypeItemList(portletURL, getDisplayStyle()) {
                {
                        addCardViewTypeItem();

                        addListViewTypeItem();

                        addTableViewTypeItem();
                }
        };
}

/**
        * Return the option items for the sort column menu.
        *
        * @return options list for the sort column menu
        */
@Override
protected List<DropdownItem> getOrderByDropdownItems() {
```

```
		return new DropdownItemList() {
			{
				add(
					dropdownItem -> {
						dropdownItem.setActive("title".equals(get
						dropdownItem.setHref(
							_getCurrentSortingURL(), "orderBy
						dropdownItem.setLabel(
							LanguageUtil.get(request, "title"
					});

				add(
					dropdownItem -> {
						dropdownItem.setActive(
							"createDate".equals(getOrderByCol
						dropdownItem.setHref(
							_getCurrentSortingURL(), "orderBy
							"createDate");
						dropdownItem.setLabel(
							LanguageUtil.get(request, "create
					});
			}
		};
	}

	/**
		* Returns the current sorting URL.
		*
		* @return current sorting portlet URL
		*
		* @throws PortletException
		*/
```

```
        private PortletURL _getCurrentSortingURL() throws PortletException {
                PortletURL sortingURL = PortletURLUtil.clone(
                        currentURLObj, liferayPortletResponse);

                sortingURL.setParameter(
                        "mvcRenderCommandName", MVCCommandNames.VIEW_ASSIGNMENTS);

                // Reset current page.

                sortingURL.setParameter(SearchContainer.DEFAULT_CUR_PARAM, "0");

                String keywords = ParamUtil.getString(request, "keywords");

                if (Validator.isNotNull(keywords)) {
                        sortingURL.setParameter("keywords", keywords);
                }

                return sortingURL;
        }

        private final PortalPreferences _portalPreferences;
        private final ThemeDisplay _themeDisplay;
}
```

Test the User Interface

1. **Go to** localhost:8080 in your browser.
2. **Refresh** the page to see the changes.
    - Make sure the module has restarted successfully or run `../gradlew deploy` again before refreshing the page.

Implement the Assignment Editing View

Exercise Goals

- Implement the Assignment editing form JSP file
- Implement an MVC render command for switching to the editing view
- Implement an MVC action command for adding an Assignment
- Implement an MVC action command for editing an Assignment
- Implement an MVC action command for deleting an Assignment
- Test the application

Implement the Assignment Editing Form

1. **Create** the file `src/main/resources/META-INF/resources/assignment/edit_assignments.jsp`.
2. **Implement** code as follows:

```
<%@ include file="../init.jsp"%>


<%-- Generate add / edit action URL and set title. --%>


<c:choose>
        <c:when test="${not empty assignment}">
                <portlet:actionURL var="assignmentActionURL" name="<%=MVCCommandNames.EDI
                        <portlet:param name="redirect" value="${param.redirect}" />
                </portlet:actionURL>


                <c:set var="editTitle" value="edit-assignment"/>
        </c:when>
        <c:otherwise>
```

```
                <portlet:actionURL var="assignmentActionURL" name="<%=MVCCommandNames.ADD
                        <portlet:param name="redirect" value="${param.redirect}" />
                </portlet:actionURL>

                <c:set var="editTitle" value="add-assignment"/>
        </c:otherwise>
</c:choose>

<div class="container-fluid-1280 edit-assignment">

        <h1><liferay-ui:message key="${editTitle}" /></h1>

        <aui:model-context bean="${assignment}" model="${assignmentClass}" />

        <aui:form action="${assignmentActionURL}" name="fm">

                <aui:input name="assignmentId" field="assignmentId" type="hidden" />

                <aui:fieldset-group markupView="lexicon">

                        <aui:fieldset>

                                <%-- Title field. --%>

                                <aui:input name="title">

                                </aui:input>

                                <%-- Description field. --%>

                                <aui:input name="description">
                                        <aui:validator name="required" />
```

```
                                </aui:input>

                                <%-- Due date field. --%>

                                <aui:input name="dueDate">
                                        <aui:validator name="required" />
                                </aui:input>
                        </aui:fieldset>
                </aui:fieldset-group>

                <%--Buttons. --%>

                <aui:button-row>
                        <aui:button cssClass="btn btn-primary" type="submit" />
                        <aui:button cssClass="btn btn-secondary" onClick="${param.redirec
                </aui:button-row>
        </aui:form>

</div>
```

If you take a look at the user interface in the browser, you'll see the plus icon for adding assignments. The button is wired in the `getCreationMenu()` method in the `AssignmentsManagemen tToolbarDisplayContext.java` class:

```
public CreationMenu getCreationMenu() {

        // Check if user has permissions to add assignments.

        if (!AssignmentTopLevelPermission.contains(
                        _themeDisplay.getPermissionChecker(),
                        _themeDisplay.getScopeGroupId(), "ADD_ENTRY")) {

                return null;
        }

        // Create the menu.

        return new CreationMenu() {
                {
                        addDropdownItem(
                                dropdownItem -> {
                                        dropdownItem.setHref(
                                                liferayPortletResponse.createRenderURL(),
                                                "mvcRenderCommandName", MVCCommandNames.E
                                                "redirect", currentURLObj.toString());
                                        dropdownItem.setLabel(
                                                LanguageUtil.get(request, "add-assignment
                                });
                }
        };
}
```

The renderURL for switching to the editing view is created, but no MVC Render Command is registered yet for the command name `MVCCommandNames.EDIT_ASSIGNMENT`.

Implement an MVC Render Command for Switching to the Editing View

1. **Create** a class `com.liferay.training.gradebook.web.portlet.action.EditAssignmentMVCRenderCommand`.
2. **Implement** code as follows:

```
package com.liferay.training.gradebook.web.portlet.action;

import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.portlet.bridges.mvc.MVCRenderCommand;
import com.liferay.portal.kernel.theme.PortletDisplay;
import com.liferay.portal.kernel.theme.ThemeDisplay;
import com.liferay.portal.kernel.util.ParamUtil;
import com.liferay.portal.kernel.util.WebKeys;
import com.liferay.training.gradebook.exception.NoSuchAssignmentException;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.AssignmentService;
import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;
import com.liferay.training.gradebook.web.constants.MVCCommandNames;

import javax.portlet.PortletException;
import javax.portlet.RenderRequest;
import javax.portlet.RenderResponse;

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

/**
 * MVC Command for showing edit assignment view.
 *
 * @author liferay
 */
@Component(
```

```
	immediate = true,
	property = {
		"javax.portlet.name=" + GradebookPortletKeys.GRADEBOOK,
		"mvc.command.name=" + MVCCommandNames.EDIT_ASSIGNMENT
	},
	service = MVCRenderCommand.class
)
public class EditAssignmentMVCRenderCommand implements MVCRenderCommand {

	@Override
	public String render(
		RenderRequest renderRequest, RenderResponse renderResponse)
		throws PortletException {

		Assignment assignment = null;

		long assignmentId = ParamUtil.getLong(renderRequest, "assignmentId", 0);

		if (assignmentId > 0) {
			try {

				// Call the service to get the assignment for editing.

				assignment = _assignmentService.getAssignment(assignmentI
			}
			catch (NoSuchAssignmentException nsae) {
				nsae.printStackTrace();
			}
			catch (PortalException pe) {
				pe.printStackTrace();
			}
		}
```

```
            ThemeDisplay themeDisplay =

                    (ThemeDisplay) renderRequest.getAttribute(WebKeys.THEME_DISPLAY);

            // Set back icon visible.

            PortletDisplay portletDisplay = themeDisplay.getPortletDisplay();

            portletDisplay.setShowBackIcon(true);

            String redirect = renderRequest.getParameter("redirect");

            portletDisplay.setURLBack(redirect);

            // Set assignment to the request attributes.

            renderRequest.setAttribute("assignment", assignment);
            renderRequest.setAttribute("assignmentClass", Assignment.class);

            return "/assignment/edit_assignments.jsp";
        }

        @Reference
        private AssignmentService _assignmentService;

}
```

Note the return value of the `render()` method

Test Adding an Assignment

1. **Go to** localhost:8080 in your browser.
2. **Refresh** the page.
3. **Click** the plus button on the gradebook widget.

You should now be able to see the editing form, but it doesn't work yet. We still need to implement an MVC Action Command to handle the form submits.

We need MVC Action Commands to handle adding, editing, and deleting assignments. A single command can handle multiple command names, so we can handle adding and editing cases in the same class. For better modularity, however, we'll choose to implement these use cases in separate classes:

Implement an MVC Action Command for Adding an Assignment

1. **Create** a class `com.liferay.training.gradebook.web.portlet.action.AddAssignmentMVCActionCommand`.
2. **Implement** code as follows:

```
package com.liferay.training.gradebook.web.portlet.action;


import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.portlet.bridges.mvc.BaseMVCActionCommand;
import com.liferay.portal.kernel.portlet.bridges.mvc.MVCActionCommand;
import com.liferay.portal.kernel.service.ServiceContext;
import com.liferay.portal.kernel.service.ServiceContextFactory;
import com.liferay.portal.kernel.theme.ThemeDisplay;
import com.liferay.portal.kernel.util.DateFormatFactoryUtil;
import com.liferay.portal.kernel.util.LocalizationUtil;
import com.liferay.portal.kernel.util.ParamUtil;
import com.liferay.portal.kernel.util.WebKeys;
import com.liferay.training.gradebook.exception.AssignmentValidationException;
import com.liferay.training.gradebook.model.Assignment;
```

```
import com.liferay.training.gradebook.service.AssignmentService;
import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;
import com.liferay.training.gradebook.web.constants.MVCCommandNames;

import java.util.Date;
import java.util.Locale;
import java.util.Map;

import javax.portlet.ActionRequest;
import javax.portlet.ActionResponse;

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

/**
 * MVC Action Command for adding assignments.
 *
 * @author liferay
 */
@Component(
        immediate = true,
        property = {
                "javax.portlet.name=" + GradebookPortletKeys.GRADEBOOK,
                "mvc.command.name=" + MVCCommandNames.ADD_ASSIGNMENT
        },
        service = MVCActionCommand.class
)
public class AddAssignmentMVCActionCommand extends BaseMVCActionCommand {

        @Override
        protected void doProcessAction(
                ActionRequest actionRequest, ActionResponse actionResponse)
```

```
        throws Exception {

        ThemeDisplay themeDisplay =
                (ThemeDisplay) actionRequest.getAttribute(WebKeys.THEME_DISPLAY);

        ServiceContext serviceContext = ServiceContextFactory.getInstance(
                Assignment.class.getName(), actionRequest);

        // Get parameters from the request.

String title = ParamUtil.getString(actionRequest, "title");
String description = ParamUtil.getString(actionRequest, "description", null);
Date dueDate = ParamUtil.getDate(actionRequest, "dueDate", null);

try {

                // Call the service to add a new assignment.

                _assignmentService.addAssignment(
                        themeDisplay.getScopeGroupId(), title, description, dueDa

                sendRedirect(actionRequest, actionResponse);
        }
        catch (AssignmentValidationException ave) {

                ave.printStackTrace();

                actionResponse.setRenderParameter(
                        "mvcRenderCommandName", MVCCommandNames.EDIT_ASSIGNMENT);

        }
        catch (PortalException pe) {
```

```
                        pe.printStackTrace();

                        actionResponse.setRenderParameter(
                                "mvcRenderCommandName", MVCCommandNames.EDIT_ASSIGNMENT);
                }
        }

        @Reference
        protected AssignmentService _assignmentService;

}
```

Implement an MVC Action Command for Editing an Assignment

1. **Create** a class `com.liferay.training.gradebook.web.portlet.action.EditAssignmentMVCActionCommand`.
2. **Implement** code as follows:

```
package com.liferay.training.gradebook.web.portlet.action;

import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.portlet.bridges.mvc.BaseMVCActionCommand;
import com.liferay.portal.kernel.portlet.bridges.mvc.MVCActionCommand;
import com.liferay.portal.kernel.service.ServiceContext;
import com.liferay.portal.kernel.service.ServiceContextFactory;
import com.liferay.portal.kernel.util.DateFormatFactoryUtil;
import com.liferay.portal.kernel.util.ParamUtil;
import com.liferay.training.gradebook.exception.AssignmentValidationException;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.AssignmentService;
import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;
import com.liferay.training.gradebook.web.constants.MVCCommandNames;
```

```
import java.util.Date;

import javax.portlet.ActionRequest;
import javax.portlet.ActionResponse;

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

/**
 * MVC Action Command for editing assignments.
 *
 * @author liferay
 *
 */
@Component(
        immediate = true,
        property = {
                "javax.portlet.name=" + GradebookPortletKeys.GRADEBOOK,
                "mvc.command.name=" + MVCCommandNames.EDIT_ASSIGNMENT
        },
        service = MVCActionCommand.class
)
public class EditAssignmentMVCActionCommand extends BaseMVCActionCommand {

        @Override
        protected void doProcessAction(
                ActionRequest actionRequest, ActionResponse actionResponse)
                throws Exception {

                ServiceContext serviceContext =
                        ServiceContextFactory.getInstance(Assignment.class.getName(), act
```

```
        // Get parameters from the request.

        long assignmentId = ParamUtil.getLong(actionRequest, "assignmentId");
String title = ParamUtil.getString(actionRequest, "title");
String description = ParamUtil.getString(actionRequest, "description", null);
Date dueDate = ParamUtil.getDate(actionRequest, "dueDate", null);

        try {

                // Call the service to update the assignment

                 _assignmentService.updateAssignment(
                        assignmentId, title, description, dueDate, serviceContext

                sendRedirect(actionRequest, actionResponse);
        }
        catch (AssignmentValidationException ave) {

                ave.printStackTrace();

                actionResponse.setRenderParameter(
                        "mvcRenderCommandName", MVCCommandNames.EDIT_ASSIGNMENT);

        }
        catch (PortalException pe) {

                pe.printStackTrace();

                actionResponse.setRenderParameter(
                        "mvcRenderCommandName", MVCCommandNames.EDIT_ASSIGNMENT);
        }
```

```
        }

        @Reference
        protected AssignmentService _assignmentService;
}
```

Implement an MVC Action Command for Deleting an Assignment

1. **Create** a class `com.liferay.training.gradebook.web.portlet.action.DeleteAssignmentMVCActionCommand`.
2. **Implement** code as follows:

```
package com.liferay.training.gradebook.web.portlet.action;

import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.portlet.bridges.mvc.BaseMVCActionCommand;
import com.liferay.portal.kernel.portlet.bridges.mvc.MVCActionCommand;
import com.liferay.portal.kernel.servlet.SessionErrors;
import com.liferay.portal.kernel.servlet.SessionMessages;
import com.liferay.portal.kernel.util.ParamUtil;
import com.liferay.training.gradebook.service.AssignmentService;
import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;

import javax.portlet.ActionRequest;
import javax.portlet.ActionResponse;

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

/**
        * MVC Action Command for deleting assignments.
        *
        * @author liferay
```

```
	*/
@Component(
	immediate = true,
	property = {
		"javax.portlet.name=" + GradebookPortletKeys.GRADEBOOK,
		"mvc.command.name=/gradebook/assignment/delete"
	},
	service = MVCActionCommand.class
)
public class DeleteAssignmentMVCActionCommand extends BaseMVCActionCommand {

	@Override
	protected void doProcessAction(
		ActionRequest actionRequest, ActionResponse actionResponse)
		throws Exception {

		// Get assignment id from request.

		long assignmentId = ParamUtil.getLong(actionRequest, "assignmentId");

		try {

			// Call service to delete the assignment.

			_assignmentService.deleteAssignment(assignmentId);

			// Set success message.

			SessionMessages.add(actionRequest, "assignmentDeleted");
		}
		catch (PortalException pe) {
```

```
                        // Set error messages from the service layer.

                        SessionErrors.add(actionRequest, "serviceErrorDetails", pe);
                }

        }

        @Reference
        protected AssignmentService _assignmentService;
}
```

Test the Application

1. **Go to** localhost:8080 in your browser.
2. **Refresh** the page.
    - Make sure the module has restarted successfully before refreshing the browser window then try adding, editing, viewing, deleting Assignments.

Implement Validation

Exercise Goals

- Customize the AssignmentValidationException to support message stacking
- Implement an Assignment validator interface
- Implement an Assignment validator service component
- Implement validation in the Assignment service
- Implement the feedback message dispatching on the controller layer
- Implement showing feedback on the user interface
- Test the server-side validation
- Implement client-side validation on the user interface
- Test the client-side validation

The Assignment validation process may encounter multiple issues. Content can be too long and have illegal characters at the same time, for example. It would be convenient to provide feedback to the user of all the issues encountered at once. To support message stacking, we have to customize the generated `AssignmentValidationException` class we defined in the `service.xml`.

Customize the AssignmentValidationException to Support Message Stacking

1. **Open** the `com.liferay.training.gradebook.exception.AssignmentValidationException.java` in the *gradebook-api* module.

2. **Implement** a new constructor and code to the class:

```
/**
 * Custom constructor taking a list as a parameter.
 *
 * @param errors
 */
public AssignmentValidationException(List<String> errors) {
        super(String.join(",", errors));
        _errors = errors;
}
public List<String> getErrors() {
        return _errors;
}
private List<String> _errors;
```

3. **Organize** missing imports. Be sure to include `import java.util.List;`.

Implement an Assignment Validator Interface

1. **Go to** the *gradebook-api* module.
2. **Create** an interface `com.liferay.training.gradebook.validator.AssignmentValidator`.
3. **Implement** code as follows:

```
package com.liferay.training.gradebook.validator;

import com.liferay.training.gradebook.exception.AssignmentValidationException;

import java.util.Date;

public interface AssignmentValidator {

	/**
	 * Validates an Assignment
	 *
	 * @param title
	 * @param description
	 * @param dueDate
	 * @throws AssignmentValidationException
	 */
	public void validate(
		String title, String description, Date dueDate)
		throws AssignmentValidationException;
}
```

Export the com.liferay.training.gradebook.validator Package

1. **Open** the `bnd.bnd` file of the *gradebook-api* project.
2. **Export** the `com.liferay.training.gradebook.validator` package. Afterwards the file will look like this:

```
Bundle-Name: gradebook-api

Bundle-SymbolicName: com.liferay.training.gradebook.api

Bundle-Version: 1.0.0

Export-Package:\

        com.liferay.training.gradebook.exception,\

        com.liferay.training.gradebook.model,\

        com.liferay.training.gradebook.service,\

        com.liferay.training.gradebook.service.persistence,\

        com.liferay.training.gradebook.validator

-check: EXPORTS

-includeresource: META-INF/service.xml=../gradebook-service/service.xml
```

Implement an Assignment Validator Service Component

1. **Create** a class `com.liferay.training.gradebook.util.validator.AssignmentValidatorImpl`.
2. **Implement** code as follows:

```
package com.liferay.training.gradebook.util.validator;


import com.liferay.portal.kernel.util.ParamUtil;

import com.liferay.portal.kernel.util.Validator;

import com.liferay.training.gradebook.exception.AssignmentValidationException;

import com.liferay.training.gradebook.validator.AssignmentValidator;


import java.util.ArrayList;

import java.util.Date;

import java.util.List;


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


@Component(
```

```
        immediate = true,
        service = AssignmentValidator.class
)
public class AssignmentValidatorImpl implements AssignmentValidator {

        /**
         * Validates assignment values and throws
         * {AssignmentValidationExceptionException} if the assignment values are not
         * valid.
         *
         * @param title
         * @param description
         * @param dueDate
         * @throws AssignmentValidationExceptionException
         */
        public void validate(
                String title, String description, Date dueDate)
                throws AssignmentValidationException {

                List<String> errors = new ArrayList<>();

                if (!isAssignmentValid(title, description, dueDate, errors)) {
                        throw new AssignmentValidationException(errors);
                }
        }

        /**
         * Returns <code>true</code> if the given fields are valid for an
         * assignment.
         *
         * @param title
         * @param description
```

```
     * @param dueDate
     * @param errors
     * @return boolean <code>true</code> if assignment is valid, otherwise
     *         <code>false</code>
     */

    private boolean isAssignmentValid(
            final String title, final String description,
            final Date dueDate, final List<String> errors) {

        boolean result = true;

        result &= isTitleValid(title, errors);
        result &= isDueDateValid(dueDate, errors);
        result &= isDescriptionValid(description, errors);

        return result;
    }

    /**
     * Returns <code>true</code> if description is valid for an assignment. If
     * not valid, an error message is added to the errors list.
     *
     * @param description
     * @param errors
     * @return boolean <code>true</code> if description is valid, otherwise
     *         <code>false</code>
     */
    private boolean isDescriptionValid(
            final String description, final List<String> errors) {

        boolean result = true;
```

```
        // Verify the description has something

        if (description == "") {
                errors.add("assignmentDescriptionEmpty");
                result = false;
        }

        return result;

}

/**
 * Returns <code>true</code> if due date is valid for an assignment. If not
 * valid, an error message is added to the errors list.
 * Note that this error can't ever happen in the user interface because
 * we are always getting a default value on the controller layer (Action Commands
 * However, this service could be access through the WS Api, which is why we neec
 *
 * @param dueDate
 * @param errors
 * @return boolean <code>true</code> if due date is valid, otherwise
 *         <code>false</code>
 */
private boolean isDueDateValid(
        final Date dueDate, final List<String> errors) {

        boolean result = true;

        if (Validator.isNull(dueDate)) {
                errors.add("assignmentDateEmpty");
                result = false;
```

```
                }

                return result;
        }

        /**
         * Returns <code>true</code> if title is valid for an assignment. If not
         * valid, an error message is added to the errors list.
         *
         * @param title
         * @param errors
         * @return boolean <code>true</code> if the title is valid, otherwise
         *         <code>false</code>
         */

        private boolean isTitleValid(
                final String title, final List<String> errors) {

                boolean result = true;

                // Verify the Title has something

                if (title == "") {
                        errors.add("assignmentTitleEmpty");
                        result = false;
                }

                return result;
        }
}
```

Implement Validation in the Assignment Service

1. **Open** the class `com.liferay.training.gradebook.service.impl.AssignmentLocalServiceImpl`.

2. **Add** a reference to the AssignmentValidator service to the end of the class:

```
@Reference
AssignmentValidator _assignmentValidator;
```

3. **Organize** missing imports.

4. **Add** the validation call to the `addAssignment()` right after the method declaration:

```
public Assignment addAssignment(
        long groupId, String title, String description,
        Date dueDate, ServiceContext serviceContext)
        throws PortalException {
        // Validate assignment parameters.
        _assignmentValidator.validate(title, description, dueDate);
        ...
```

5. **Add** a validation call to `updateAssignment()` right after the method declaration:

```
public Assignment updateAssignment(
        long assignmentId, String title, String description,
        Date dueDate, ServiceContext serviceContext)
        throws PortalException {
        // Validate assignment parameters.
        _assignmentValidator.validate(title, description, dueDate);
        ...
```

6. **Organize** missing imports.
7. **Rebuild** the service.

Validation is now implemented on the service layer. As we call the services on the controller layer through the MVC commands in the *gradebook-web* module, we have to pass the feedback messages from the service layer to the user interface there.

For transporting the messages to the user interface, we'll be using the SessionMessages object for success messages and the SessionErrors object for the error messages.

You've probably noticed the default success message the platform sets when you add an Assignment. We'll silence that because we'll be using our custom message.

Implement Feedback Messages Dispatching on the Controller Layer

1. **Open** the class `GradebookPortlet.java` in the *gradebook-web* module.

2. **Add** the following component property:

    ```
    "javax.portlet.init-param.add-process-action-success-action=false"
    ```

    - Modify the `doProcessAction()` methods of the three MVC Action Command classes in the *gradebook-web* module, calling the service to set the success and error messages for the user interface.

3. **Update** the code of all of the following MVC Action Command classes as follows:

AddAssignmentMVCActionCommand.java

```
package com.liferay.training.gradebook.web.portlet.action;


import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.portlet.bridges.mvc.BaseMVCActionCommand;
import com.liferay.portal.kernel.portlet.bridges.mvc.MVCActionCommand;
import com.liferay.portal.kernel.service.ServiceContext;
import com.liferay.portal.kernel.service.ServiceContextFactory;
import com.liferay.portal.kernel.servlet.SessionErrors;
import com.liferay.portal.kernel.servlet.SessionMessages;
```

```
import com.liferay.portal.kernel.theme.ThemeDisplay;
import com.liferay.portal.kernel.util.DateFormatFactoryUtil;
import com.liferay.portal.kernel.util.LocalizationUtil;
import com.liferay.portal.kernel.util.ParamUtil;
import com.liferay.portal.kernel.util.WebKeys;
import com.liferay.training.gradebook.exception.AssignmentValidationException;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.AssignmentService;
import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;
import com.liferay.training.gradebook.web.constants.MVCCommandNames;

import java.util.Date;
import java.util.Locale;
import java.util.Map;

import javax.portlet.ActionRequest;
import javax.portlet.ActionResponse;

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

/**
 * MVC Action Command for adding assignments.
 *
 * @author liferay
 */
@Component(
        immediate = true,
        property = {
                "javax.portlet.name=" + GradebookPortletKeys.GRADEBOOK,
                "mvc.command.name=" + MVCCommandNames.ADD_ASSIGNMENT
        },
```

```
        service = MVCActionCommand.class
)
public class AddAssignmentMVCActionCommand extends BaseMVCActionCommand {

        @Override
        protected void doProcessAction(
                ActionRequest actionRequest, ActionResponse actionResponse)
                throws Exception {

                ThemeDisplay themeDisplay =
                        (ThemeDisplay) actionRequest.getAttribute(WebKeys.THEME_DISPLAY);

                ServiceContext serviceContext = ServiceContextFactory.getInstance(
                        Assignment.class.getName(), actionRequest);

                // Get parameters from the request.

                        String title = ParamUtil.getString(actionRequest, "title'

                        String description = ParamUtil.getString(actionRequest, '

                        Date dueDate = ParamUtil.getDate(actionRequest, "dueDate", null);
        try {

                        // Call the service to add a new assignment.

                        _assignmentService.addAssignment(
                                themeDisplay.getScopeGroupId(), title, description, dueDa

                        // Set the success message.

                        SessionMessages.add(actionRequest, "assignmentAdded");
```

```
                sendRedirect(actionRequest, actionResponse);
        }
        catch (AssignmentValidationException ave) {

                // Get error messages from the service layer.

                ave.getErrors().forEach(key -> SessionErrors.add(actionRequest, k

                ave.printStackTrace();

                actionResponse.setRenderParameter(
                        "mvcRenderCommandName", MVCCommandNames.EDIT_ASSIGNMENT);

        }
        catch (PortalException pe) {

                // Set error messages from the service layer.

            SessionErrors.add(actionRequest, "serviceErrorDetails", pe);

                pe.printStackTrace();




                actionResponse.setRenderParameter(
                        "mvcRenderCommandName", MVCCommandNames.EDIT_ASSIGNMENT);
        }
}

@Reference
protected AssignmentService _assignmentService;
```

```
}
```

EditAssignmentMVCActionCommand.java

```
package com.liferay.training.gradebook.web.portlet.action;

import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.portlet.bridges.mvc.BaseMVCActionCommand;
import com.liferay.portal.kernel.portlet.bridges.mvc.MVCActionCommand;
import com.liferay.portal.kernel.service.ServiceContext;
import com.liferay.portal.kernel.service.ServiceContextFactory;
import com.liferay.portal.kernel.servlet.SessionErrors;
import com.liferay.portal.kernel.servlet.SessionMessages;
import com.liferay.portal.kernel.util.DateFormatFactoryUtil;
import com.liferay.portal.kernel.util.ParamUtil;
import com.liferay.training.gradebook.exception.AssignmentValidationException;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.AssignmentService;
import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;
import com.liferay.training.gradebook.web.constants.MVCCommandNames;

import java.util.Date;

import javax.portlet.ActionRequest;
import javax.portlet.ActionResponse;

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

/**
 * MVC Action Command for editing assignments.
```

```
 *
 * @author liferay
 *
 */
@Component(
        immediate = true,
        property = {
                "javax.portlet.name=" + GradebookPortletKeys.GRADEBOOK,
                "mvc.command.name=" + MVCCommandNames.EDIT_ASSIGNMENT
        },
        service = MVCActionCommand.class
)
public class EditAssignmentMVCActionCommand extends BaseMVCActionCommand {

        @Override
        protected void doProcessAction(
                ActionRequest actionRequest, ActionResponse actionResponse)
                throws Exception {

                ServiceContext serviceContext =
                        ServiceContextFactory.getInstance(Assignment.class.getName(), act

                // Get parameters from the request.

                long assignmentId = ParamUtil.getLong(actionRequest, "assignmentId");

                String title = ParamUtil.getString(actionRequest, "title");

                String description = ParamUtil.getString(actionRequest, "description", nu

                Date dueDate = ParamUtil.getDate(actionRequest, "dueDate", null);
                try {
```

```
            // Call the service to update the assignment

             _assignmentService.updateAssignment(
                    assignmentId, title, description, dueDate, serviceContext

            // Set the success message.

            SessionMessages.add(actionRequest, "assignmentUpdated");

            sendRedirect(actionRequest, actionResponse);
        }
        catch (AssignmentValidationException ave) {

            // Get error messages from the service layer.

            ave.getErrors().forEach(key -> SessionErrors.add(actionRequest, k

            ave.printStackTrace();

            actionResponse.setRenderParameter(
                    "mvcRenderCommandName", MVCCommandNames.EDIT_ASSIGNMENT);

        }
        catch (PortalException pe) {

            // Get error messages from the service layer.

            SessionErrors.add(actionRequest, "serviceErrorDetails", pe);

            pe.printStackTrace();
```

```
                        actionResponse.setRenderParameter(
                                "mvcRenderCommandName", MVCCommandNames.EDIT_ASSIGNMENT);
                }
        }

        @Reference
        protected AssignmentService _assignmentService;
}
```

DeleteAssignmentMVCActionCommand.java

```
package com.liferay.training.gradebook.web.portlet.action;

import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.portlet.bridges.mvc.BaseMVCActionCommand;
import com.liferay.portal.kernel.portlet.bridges.mvc.MVCActionCommand;
import com.liferay.portal.kernel.servlet.SessionErrors;
import com.liferay.portal.kernel.servlet.SessionMessages;
import com.liferay.portal.kernel.util.ParamUtil;
import com.liferay.training.gradebook.service.AssignmentService;
import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;

import javax.portlet.ActionRequest;
import javax.portlet.ActionResponse;

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

/**
        * MVC Action Command for deleting assignments.
        *
        * @author liferay
        */
@Component(
        immediate = true,
        property = {
                "javax.portlet.name=" + GradebookPortletKeys.GRADEBOOK,
                "mvc.command.name=/gradebook/assignment/delete"
        },
        service = MVCActionCommand.class
```

```
)
public class DeleteAssignmentMVCActionCommand extends BaseMVCActionCommand {

	@Override
	protected void doProcessAction(
		ActionRequest actionRequest, ActionResponse actionResponse)
		throws Exception {

		// Get assignment id from request.

		long assignmentId = ParamUtil.getLong(actionRequest, "assignmentId");

		try {

			// Call service to delete the assignment.

			_assignmentService.deleteAssignment(assignmentId);

			// Set success message.

			SessionMessages.add(actionRequest, "assignmentDeleted");
		}
		catch (PortalException pe) {

			// Set error messages from the service layer.

			SessionErrors.add(actionRequest, "serviceErrorDetails", pe);
		}

	}

	@Reference
```

```
        protected AssignmentService _assignmentService;
}
```

The last thing to do for displaying the error messages from the service layer is to implement the user interface. We'll be using the `<liferay-ui>` tag library for this purpose.

Add an import for the `SessionErrors` class for showing the error message details.

Import SessionErrors Class

1. **Add** an import in `src/main/resources/META-INF/resources/init.jsp`:

```
<%@ page import="com.liferay.portal.kernel.servlet.SessionErrors"%>
```

   - After we add, update, or delete an Assignment successfully, we are redirected to the main list view, implemented with the `view.jsp`.

2. **Add** `<liferay-ui>` tags to `src/main/resources/META-INF/resources/view.jsp` just after the `init.jsp` include:

```
<%@ include file="init.jsp"%>
<liferay-ui:error key="serviceErrorDetails">
        <liferay-ui:message arguments='<%= SessionErrors.get(liferayPortletI
</liferay-ui:error>
<liferay-ui:success key="assignmentAdded" message="assignment-added-success
<liferay-ui:success key="assignmentUpdated" message="assignment-updated-succ
<liferay-ui:success key="assignmentDeleted" message="assignment-deleted-succ
```

3. **Add** `<liferay-ui>` tags to `src/main/resources/META-INF/resources/assignment/edit_assigments.jsp` just after the `init.jsp` include:

```
<%@ include file="../init.jsp"%>


<liferay-ui:error key="serviceErrorDetails">
        <liferay-ui:message key="error.assignment-service-error" arguments='<%= SessionEr
</liferay-ui:error>
<liferay-ui:error key="assignmentTitleEmpty" message="error.assignment-title-empty" />
<liferay-ui:error key="assignmentDescriptionEmpty" message="error.assignment-description-
```

Server-side validation is now implemented. Let's test it.

Test the Server-Side Validation

1. **Open** the Gradebook application in your web browser.
2. **Click** on the plus sign to add an Assignment.
3. **Leave** the *Title* field empty but enter something on the *Description* field.
4. **Submit** the form:

> If you get a `NoSuchMethodError` error in your log, remove and redeploy the modules from the server.

Detecting invalid input early on the user interface, client-side, improves the user experience and reduces server load. Remember, however, that user interface validation, typically JavaScript-based, is more about usability than security: if you disable page JavaScripts, your security is gone.

Take a look at the `edit_assignments.jsp`. Notice the already existing aui:validator tag for the *description* fields:

```
<aui:validator name="required" />
```

We will add two validators for the *title* field. One is for setting the field mandatory, and the other one checks for valid characters.

Implement Client-Side Validation

1. **Open** the `edit_assignments.jsp`.
2. **Add** the following validator tags inside the `<aui:input name="title">` tag:

```
<aui:validator name="required" />


<aui:validator errorMessage="error.assignment-title-format" name="custom">
        function(val, fieldNode, ruleValue) {
                var wordExpression =
                        new RegExp("^[^\\[\\]\\^$<>]*$");


                return wordExpression.test(val);
        }
</aui:validator>
```

The complete `edit_assignments.jsp` file will look like this:

```
<%@ include file="../init.jsp"%>
<liferay-ui:error key="serviceErrorDetails">
        <liferay-ui:message key="error.assignment-service-error" arguments='<%= SessionEr
</liferay-ui:error>
<liferay-ui:error key="assignmentTitleEmpty" message="error.assignment-title-empty" />
<liferay-ui:error key="assignmentDescriptionEmpty" message="error.assignment-description-


<%-- Generate add / edit action URL and set title. --%>


<c:choose>
        <c:when test="${not empty assignment}">
                <portlet:actionURL var="assignmentActionURL" name="<%=MVCCommandNames.EDI
                        <portlet:param name="redirect" value="${param.redirect}" />
                </portlet:actionURL>
```

```
                <c:set var="editTitle" value="edit-assignment"/>
        </c:when>
        <c:otherwise>
                <portlet:actionURL var="assignmentActionURL" name="<%=MVCCommandNames.ADD
                        <portlet:param name="redirect" value="${param.redirect}" />
                </portlet:actionURL>

                <c:set var="editTitle" value="add-assignment"/>
        </c:otherwise>
</c:choose>

<div class="container-fluid-1280 edit-assignment">

        <h1><liferay-ui:message key="${editTitle}" /></h1>

        <aui:model-context bean="${assignment}" model="${assignmentClass}" />

        <aui:form action="${assignmentActionURL}" name="fm">

                <aui:input name="assignmentId" field="assignmentId" type="hidden" />

                <aui:fieldset-group markupView="lexicon">

                        <aui:fieldset>

                                <%-- Title field. --%>

                                <aui:input name="title">
                                        <aui:validator name="required" />
                                        <%-- Custom AUI validator. --%>

                                        <aui:validator errorMessage="error.assignment-tit
```

```
                                        function(val, fieldNode, ruleValue) {
                                                var wordExpression =
                                                        new RegExp("^[^\\[\\]\\^$

                                                return wordExpression.test(val);
                                        }
                                </aui:validator>
                        </aui:input>

                        <%-- Description field. --%>

                        <aui:input name="description">
                                <aui:validator name="required" />
                        </aui:input>

                        <%-- Due date field. --%>

                        <aui:input name="dueDate">
                                <aui:validator name="required" />
                        </aui:input>
                </aui:fieldset>
        </aui:fieldset-group>

        <%--Buttons. --%>

        <aui:button-row>
                <aui:button cssClass="btn btn-primary" type="submit" />
                <aui:button cssClass="btn btn-secondary" onClick="${param.redirec

        </aui:button-row>
    </aui:form>

</div>
```

Test the Client-Side Validation

1. **Open** the Gradebook application in your web browser.
2. **Click** on the plus sign to add an Assignment.
3. **Put** a dollar `$` sign in the title field and leave the *Description* field empty.
4. **Submit** the form.

Application Presentation and Customization Quiz

1. Which of the following are module templates that Liferay DXP provides for building the presentation layer? (Choose all correct answers).

   A. mvc portlet

   B. freemarker-portlet

   C. npm-mvc-portlet

   D. spring-mvc-portlet

   E. lar-mvc-portlet

2. Which of the following is *not* a valid MVC command class? (Choose two)

   A. MVC Render

   B. MVC Resource

   C. MVC Rewrite

   D. MVC Action

3. Liferay-provided tag libraries, such as clay and liferay-theme, contain tags for the most common user interface components but are ultimately superseded by standard tag libraries.

   A. True

   B. False

4. Input validation is typically beneficial to all of the following areas, except _____.

   A. usability

   B. reliability

   C. security

   D. resource usage

5. A feedback process within an application could potentially consist of setting a message key in the back-end, doing a localization in Language.properties, and showing the message on the user interface using the `<liferay-ui:success>` and `<liferay-ui:error>` tags.

   A. True

   B. False

Answer Key

1. A, B, D
2. B, C
3. False
4. B
5. True