Implement Access Control

Learning Objectives

- Understand the relationships between Resources, Actions, Permissions, and Roles as they relate to application security
- Know the typical process by which Permissioning is implemented on the Back-End

Tasks to Accomplish

- Implement Service Module Permissions
- Implement Web Module Permissions

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

Implement Service Module Permissions

Exercise Goals

- Create the resource constants class
- Define the permissions
- Define the permissions definition location
- Implement permission resource management in the AssignmentLocalServiceImpl
- Create the permission registration classes
- Implement permission checking in the remote service class AssignmentServiceImpl
- Rebuild the service
- Test the application

Before proceeding, you **must remove all the test assignments you have created so far**. This is because the existing test entities don't have the resources to support permissioning and will cause errors.

To avoid misspellings in permission properties, we will create a constants class in the *gradebook-api* module.

Create the Resource Constants Class

1. **Create** a class `com.liferay.training.gradebook.constants.GradebookConstants` and implement as follows:

```
package com.liferay.training.gradebook.constants;
public class GradebookConstants {
        public static final String RESOURCE_NAME = "com.liferay.training.gra
}
```

2. **Add** `com.liferay.training.gradebook.constants` to the exported packages in the `bnd.bnd` file. 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.constants,\

        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
```

By default, permissions are defined in the file called `default.xml`.

Define the Permissions

1. **Create** a folder `src/main/resources/resource-actions` in the *gradebook-service* module.
2. **Create** a file `src/main/resources/resource-actions/default.xml` and implement as follows (switch to *Source* mode, if needed):

```
<?xml version="1.0"?>

<!DOCTYPE resource-action-mapping PUBLIC "-//Liferay//DTD Resource Action Mapping 7.4.0/

<resource-action-mapping>

        <model-resource>

                <model-name>com.liferay.training.gradebook.model</model-name>

                <portlet-ref>

                        <portlet-name>com_liferay_training_gradebook_web_portlet_Gradeboo

                </portlet-ref>

            <root>true</root>

            <permissions>

                        <supports>

                                <action-key>ADD_ENTRY</action-key>
```

```
                        <action-key>PERMISSIONS</action-key>
                </supports>
                <site-member-defaults />
                <guest-defaults />
                <guest-unsupported>
                        <action-key>ADD_ENTRY</action-key>
                        <action-key>PERMISSIONS</action-key>
                </guest-unsupported>
        </permissions>
</model-resource>
<model-resource>
        <model-name>com.liferay.training.gradebook.model.Assignment</model-name>
        <portlet-ref>
                <portlet-name>com_liferay_training_gradebook_web_portlet_Gradeboo
        </portlet-ref>
        <permissions>
                <supports>
                        <action-key>DELETE</action-key>
                        <action-key>PERMISSIONS</action-key>
                        <action-key>UPDATE</action-key>
                        <action-key>VIEW</action-key>
                        <action-key>ADD_SUBMISSION</action-key>
                        <action-key>EDIT_SUBMISSION</action-key>
                        <action-key>DELETE_SUBMISSION</action-key>
                        <action-key>GRADE_SUBMISSION</action-key>
                        <action-key>VIEW_SUBMISSIONS</action-key>
                </supports>
                <site-member-defaults>
                        <action-key>VIEW</action-key>
                        <action-key>ADD_SUBMISSION</action-key>
                </site-member-defaults>
                <guest-defaults>
```

```
                        <action-key>VIEW</action-key>
                    </guest-defaults>
                    <guest-unsupported>
                        <action-key>DELETE</action-key>
                        <action-key>PERMISSIONS</action-key>
                        <action-key>UPDATE</action-key>
                        <action-key>ADD_SUBMISSION</action-key>
                        <action-key>DELETE_SUBMISSION</action-key>
                        <action-key>EDIT_SUBMISSION</action-key>
                        <action-key>GRADE_SUBMISSION</action-key>
                        <action-key>VIEW_SUBMISSIONS</action-key>
                    </guest-unsupported>
                </permissions>
        </model-resource>
</resource-action-mapping>
```

> Notice that submission permissions are related to the optional exercises.

Define the Permissions Definition Location

1. **Create** a file `src/main/resources/portlet.properties` in the *gradebook-service* module.
2. **Implement** the file as follows:

```
resource.actions.configs=/resource-actions/default.xml
```

Permissions need container objects for the model entities. When we create an entity, we need to create and bind a resource object to that. Accordingly, we have to take care of cleaning up the resources when we delete the entity:

Implement Permission Resource Management

1. **Open** the class `com.liferay.training.gradebook.service.impl.AssignmentLocalServiceImpl`.
2. **Replace** the `addAssignment()` method with the following:

```
public Assignment addAssignment(
        long groupId, Map<Locale, String> titleMap, String description,
        Date dueDate, ServiceContext serviceContext)
        throws PortalException {

        // Validate assignment parameters.

        _assignmentValidator.validate(titleMap, description, dueDate);

        // Get group and user.

        Group group = groupLocalService.getGroup(groupId);

        long userId = serviceContext.getUserId();

        User user = userLocalService.getUser(userId);

        // Generate primary key for the assignment.

        long assignmentId =
                counterLocalService.increment(Assignment.class.getName());

        // Create assigment. This doesn't yet persist the entity.

        Assignment assignment = createAssignment(assignmentId);

        // Populate fields.

        assignment.setCompanyId(group.getCompanyId());
        assignment.setCreateDate(serviceContext.getCreateDate(new Date()));
        assignment.setDueDate(dueDate);
```

```
        assignment.setDescription(description);
        assignment.setGroupId(groupId);
        assignment.setModifiedDate(serviceContext.getModifiedDate(new Date(
        assignment.setTitleMap(titleMap);
        assignment.setUserId(userId);
        assignment.setUserName(user.getScreenName());

        // Persist assignment to database.

        assignment = super.addAssignment(assignment);

        // Add permission resources.

        boolean portletActions = false;
        boolean addGroupPermissions = true;
        boolean addGuestPermissions = true;

        resourceLocalService.addResources(
                group.getCompanyId(), groupId, userId, Assignment.class.getN
                assignment.getAssignmentId(), portletActions, addGroupPermis
                addGuestPermissions);

        return assignment;
    }
```

3. **Implement** a new signature for deleting assignments as follows:

```
public Assignment deleteAssignment(Assignment assignment)
        throws PortalException {

        // Delete permission resources.

        resourceLocalService.deleteResource(
                assignment, ResourceConstants.SCOPE_INDIVIDUAL);

        // Delete the Assignment

        return super.deleteAssignment(assignment);
}
```

4. **Resolve** missing imports.

Don't worry about the errors when adding a new method. Errors will go away after you rebuild the service.

Now we will create classes for registering the model and top-level resource permissions.

Create the Permission Registrar Classes

1. **Create** a class `com.liferay.training.gradebook.internal.security.permission.resource.definition.AssignmentModelResourcePermissionDefinition` in the *gradebook-service* module.

2. **Implement** as follows:

```
package com.liferay.training.gradebook.internal.security.permission.resourc

import com.liferay.exportimport.kernel.staging.permission.StagingPermission
import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.security.permission.resource.ModelResourcel
```

```
import com.liferay.portal.kernel.security.permission.resource.ModelResourceI
import com.liferay.portal.kernel.security.permission.resource.PortletResourc
import com.liferay.portal.kernel.security.permission.resource.StagedModelPer
import com.liferay.portal.kernel.security.permission.resource.WorkflowedMode
import com.liferay.portal.kernel.security.permission.resource.definition.Mod
import com.liferay.portal.kernel.service.GroupLocalService;
import com.liferay.portal.kernel.workflow.permission.WorkflowPermission;
import com.liferay.training.gradebook.constants.GradebookConstants;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.AssignmentLocalService;

import java.util.function.Consumer;

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

/**
 * @author liferay
 */
@Component(
        immediate = true,
        service = ModelResourcePermissionDefinition.class
)
public class AssignmentModelResourcePermissionDefinition
        implements ModelResourcePermissionDefinition<Assignment> {

        @Override
        public Assignment getModel(long assignmentId)
                throws PortalException {

                return _assignmentLocalService.getAssignment(assignmentId);
        }
```

```
@Override
public Class<Assignment> getModelClass() {

        return Assignment.class;
}

@Override
public PortletResourcePermission getPortletResourcePermission() {

        return _portletResourcePermission;
}

@Override
public long getPrimaryKey(Assignment assignment) {

        return assignment.getAssignmentId();
}

@Override
public void registerModelResourcePermissionLogics(
        ModelResourcePermission<Assignment> modelResourcePermission,
        Consumer<ModelResourcePermissionLogic<Assignment>> modelReso

        modelResourcePermissionLogicConsumer.accept(
                new StagedModelPermissionLogic<>(
                        _stagingPermission,
                        "com_liferay_training_gradebook_web_portlet_
                        Assignment::getAssignmentId));

        // Only enable if you use (optional) workflow support
```

```
                //        modelResourcePermissionLogicConsumer.accept(
                //                new WorkflowedModelPermissionLogic<>(
                //                        _workflowPermission, modelResourc
                //                        _groupLocalService, Assignment::g
        }

        @Reference
        private AssignmentLocalService _assignmentLocalService;

        @Reference
        private GroupLocalService _groupLocalService;

        @Reference(target = "(resource.name=" + GradebookConstants.RESOURCE_
        private PortletResourcePermission _portletResourcePermission;

        @Reference
        private StagingPermission _stagingPermission;

        @Reference
        private WorkflowPermission _workflowPermission;
    }
```

3. **Create** a class for registering the Gradebook portlet resources and top level permissions:

```
com.liferay.training.gradebook.internal.security.permission.resource.definit
```

4. **Implement** as follows:

```
package com.liferay.training.gradebook.internal.security.permission.resource.definition;

import com.liferay.exportimport.kernel.staging.permission.StagingPermission;

import com.liferay.portal.kernel.security.permission.resource.PortletResourcePermissionLo

import com.liferay.portal.kernel.security.permission.resource.StagedPortletPermissionLogi

import com.liferay.portal.kernel.security.permission.resource.definition.PortletResourceP
```

```
import com.liferay.training.gradebook.constants.GradebookConstants;

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

/**
 * @author liferay
 */
@Component(
        immediate = true,
        service = PortletResourcePermissionDefinition.class
)
public class AssignmentPortletResourcePermissionDefinition
        implements PortletResourcePermissionDefinition {

        @Override
        public PortletResourcePermissionLogic[] getPortletResourcePermissionLogics() {

                return new PortletResourcePermissionLogic[] {
                        new StagedPortletPermissionLogic(
                                _stagingPermission,
                                "com_liferay_training_gradebook_web_portlet_GradebookPort
                };
        }

        @Override
        public String getResourceName() {

                return GradebookConstants.RESOURCE_NAME;
        }

        @Reference
```

```
        private StagingPermission _stagingPermission;


}
```

Now let's implement the permission checking in our remote service class. Let's go through the implementation steps first.

We need references to the portlet and model resource permission services:

```
@Reference(
        policy = ReferencePolicy.DYNAMIC,
        policyOption = ReferencePolicyOption.GREEDY,
        target = "(model.class.name=com.liferay.training.gradebook.model.Assignment)"
)
private volatile ModelResourcePermission<Assignment>
        _assignmentModelResourcePermission;


@Reference(
        policy = ReferencePolicy.DYNAMIC,
        policyOption = ReferencePolicyOption.GREEDY,
        target = "(resource.name=" + GradebookConstants.RESOURCE_NAME + ")"
)
private volatile PortletResourcePermission _portletResourcePermission;
```

Note that for a dynamic reference to work, the field **must** be declared with the volatile modifier so that field value changes made by service component runtime are visible to other threads.

Then we'll implement permission checking in `addAssignment()`, `deleteAssignment()` and `updateAssignment()` methods:

```
public Assignment getAssignment(long assignmentId)
        throws PortalException {

        // Check permissions.

        _assignmentModelResourcePermission.check(
                getPermissionChecker(), assignmentId, ActionKeys.VIEW);

        Assignment assignment =
                assignmentLocalService.getAssignment(assignmentId);

        return assignment;
}
```

FindBy finder calls are transformed into *Filtered Finder* queries, which take permissions into account:

```
assignmentPersistence.filterFindByGroupId(groupId);
```

Take a look at the `AssignmentPersistenceImpl` class and for example `filterFindByGroupId()` method to see how filtered finders are implemented.

In this exercise, we'll allow everybody to see and search the assignments list by the keyword search method `getAssignmentsByKeywords`. We just don't allow unauthorized users to view, update or delete them.

Implement Permission Checking in the Remote Service

1. **Implement** the class `com.liferay.training.gradebook.service.impl.AssignmentServiceImpl.java` as follows (You'll see an error for the `filterFindBy` method because it's not yet generated):

```
/**
 * Copyright (c) 2000-present Liferay, Inc. All rights reserved.
```

```
 *
 * This library is free software; you can redistribute it and/or modify it under
 * the terms of the GNU Lesser General Public License as published by the Free
 * Software Foundation; either version 2.1 of the License, or (at your option)
 * any later version.
 *
 * This library is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
 * FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more
 * details.
 */

package com.liferay.training.gradebook.service.impl;

import com.liferay.portal.aop.AopService;
import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.security.permission.ActionKeys;
import com.liferay.portal.kernel.security.permission.resource.ModelResourcePermission;
import com.liferay.portal.kernel.security.permission.resource.ModelResourcePermissionFact
import com.liferay.portal.kernel.security.permission.resource.PortletResourcePermission;
import com.liferay.portal.kernel.service.ServiceContext;
import com.liferay.portal.kernel.util.OrderByComparator;
import com.liferay.training.gradebook.constants.GradebookConstants;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.base.AssignmentServiceBaseImpl;

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

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

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

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

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


/**

 * The implementation of the assignment remote service.

 *

 * <p>

 * All custom service methods should be put in this class. Whenever methods are added, re

 *

 * <p>

 * This is a remote service. Methods of this service are expected to have security checks

 * </p>

 *

 * @author Brian Wing Shun Chan

 * @see AssignmentServiceBaseImpl

 */


 @Component(

        property = {

                "json.web.service.context.name=gradebook",

                "json.web.service.context.path=Assignment"

         },

         service = AopService.class

 )

public class AssignmentServiceImpl extends AssignmentServiceBaseImpl {


        /*

         * NOTE FOR DEVELOPERS:

         *

         * Never reference this class directly. Always use <code>com.liferay.training.gra

         */
```

```
public Assignment addAssignment(
        long groupId, Map<Locale, String> titleMap, String description,
        Date dueDate, ServiceContext serviceContext)
        throws PortalException {

        // Check permissions.

        _portletResourcePermission.check(
                getPermissionChecker(), serviceContext.getScopeGroupId(),
                ActionKeys.ADD_ENTRY);

        return assignmentLocalService.addAssignment(
                groupId, titleMap, description, dueDate, serviceContext);
}

public Assignment deleteAssignment(long assignmentId)
        throws PortalException {

        // Check permissions.

        _assignmentModelResourcePermission.check(
                getPermissionChecker(), assignmentId, ActionKeys.DELETE);

        Assignment assignment =
                assignmentLocalService.getAssignment(assignmentId);

        return assignmentLocalService.deleteAssignment(assignment);
}

public Assignment getAssignment(long assignmentId)
        throws PortalException {
```

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

        // Check permissions.

        _assignmentModelResourcePermission.check(
                getPermissionChecker(), assignment, ActionKeys.VIEW);

        return assignment;
}

public List<Assignment> getAssignmentsByGroupId(long groupId) {

        return assignmentPersistence.filterFindByGroupId(groupId);
}

public List<Assignment> getAssignmentsByKeywords(
        long groupId, String keywords, int start, int end,
        OrderByComparator<Assignment> orderByComparator) {

        return assignmentLocalService.getAssignmentsByKeywords(
                groupId, keywords, start, end, orderByComparator);
}

public long getAssignmentsCountByKeywords(long groupId, String keywords) {

        return assignmentLocalService.getAssignmentsCountByKeywords(
                groupId, keywords);
}

public Assignment updateAssignment(
        long assignmentId, Map<Locale, String> titleMap, String description,
```

```
            Date dueDate, ServiceContext serviceContext)
            throws PortalException {

            // Check permissions.

            _assignmentModelResourcePermission.check(
                    getPermissionChecker(), assignmentId, ActionKeys.UPDATE);

            return assignmentLocalService.updateAssignment(
                    assignmentId, titleMap, description, dueDate, serviceContext);
        }

        @Reference(
            policy = ReferencePolicy.DYNAMIC,
            policyOption = ReferencePolicyOption.GREEDY,
            target = "(model.class.name=com.liferay.training.gradebook.model.Assignme
        )
        private volatile ModelResourcePermission<Assignment>
            _assignmentModelResourcePermission;

        @Reference(
            policy = ReferencePolicy.DYNAMIC,
            policyOption = ReferencePolicyOption.GREEDY,
            target = "(resource.name=" + GradebookConstants.RESOURCE_NAME + ")"
        )
        private volatile PortletResourcePermission _portletResourcePermission;
}
```

Rebuild the Service

1. **Run** the `buildService` task to deploy the changes.

Test the Application

1. **Sign out** of the portal.
2. **Try** to add an assignment. You should get an error message.

Implement Web Module Permissions

Exercise Goals

- Define portlet permissions
- Define the permissions definition location
- Implement the top-level permission resource permission checker class
- Implement the model resource permission checker class
- Implement permission checking in the JSP files
- Implement permission checking in the management toolbar
- Test the application

Define the Permissions

1. **Create** a folder `src/main/resources/resource-actions` in the *gradebook-web* module.
2. **Create** a file `src/main/resources/resource-actions/default.xml` and implement as follows (switch to *Source* mode, if needed):

```
<?xml version="1.0"?>
<!DOCTYPE resource-action-mapping PUBLIC "-//Liferay//DTD Resource Action  apping 7.4.0/
<resource-action-mapping>
        <portlet-resource>
                <portlet-name>com_liferay_training_gradebook_web_portlet_GradebookPortlet
                <permissions>
                        <supports>
                                <action-key>ADD_PORTLET_DISPLAY_TEMPLATE</action-key>
                                <action-key>ADD_TO_PAGE</action-key>
                                <action-key>CONFIGURATION</action-key>
                                <action-key>VIEW</action-key>
                        </supports>
```

```
                        <site-member-defaults>
                                <action-key>VIEW</action-key>
                        </site-member-defaults>
                        <guest-defaults>
                                <action-key>VIEW</action-key>
                        </guest-defaults>
                        <guest-unsupported>
                                <action-key>ADD_PORTLET_DISPLAY_TEMPLATE</action-key>
                                <action-key>ADD_TO_PAGE</action-key>
                                <action-key>CONFIGURATION</action-key>
                        </guest-unsupported>
                </permissions>
        </portlet-resource>
</resource-action-mapping>
```

Define the Permissions Definition Location

1. **Create** a file `src/main/resources/portlet.properties` in the *gradebook-web* module.
2. **Implement** the file as follows:

```
resource.actions.configs=/resource-actions/default.xml
```

Implement a helper class in the *gradebook-web* module for checking top-level permissions. This is a permission checker class we'll call from the user interface.

Implement the Top-Level Resource Permission Checker Class

1. **Create** the class `com.liferay.training.gradebook.web.internal.security.permission.resource.AssignmentTopLevelPermission`.
2. **Implement** as follows:

```
package com.liferay.training.gradebook.web.internal.security.permission.resource;


import com.liferay.portal.kernel.security.permission.PermissionChecker;
```

```
import com.liferay.portal.kernel.security.permission.resource.PortletResourcePermission;
import com.liferay.training.gradebook.constants.GradebookConstants;

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

/**
 * @author liferay
 */
@Component(
	immediate = true
)
public class AssignmentTopLevelPermission {

	public static boolean contains(
		PermissionChecker permissionChecker, long groupId, String actionId) {

		return _portletResourcePermission.contains(
			permissionChecker, groupId, actionId);
	}

	@Reference(
		target = "(resource.name=" + GradebookConstants.RESOURCE_NAME + ")",
		unbind = "-"
	)
	protected void setPortletResourcePermission(
		PortletResourcePermission portletResourcePermission) {

		_portletResourcePermission = portletResourcePermission;
	}

	private static PortletResourcePermission _portletResourcePermission;
```

```
}
```

Next we need to implement a class for checking existing entity permissions.

Implement the Model Resource Permission Checker Class

1. **Create** the class `com.liferay.training.gradebook.web.internal.security.permission.resource.AssignmentPermission`.
2. **Implement** as follows:

```
package com.liferay.training.gradebook.web.internal.security.permission.resource;

import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.security.permission.PermissionChecker;
import com.liferay.portal.kernel.security.permission.resource.ModelResourcePermission;
import com.liferay.training.gradebook.model.Assignment;

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

/**
 * @author liferay
 */
@Component(
        immediate = true,
        service = AssignmentPermission.class
)
public class AssignmentPermission {

        public static boolean contains(
                        PermissionChecker permissionChecker, Assignment assignment,
                        String actionId)
```

```
            throws PortalException {

            return _assignmentModelResourcePermission.contains(
                    permissionChecker, assignment, actionId);
        }

        public static boolean contains(
                PermissionChecker permissionChecker, long assignmentId, String ac
            throws PortalException {

            return _assignmentModelResourcePermission.contains(
                    permissionChecker, assignmentId, actionId);
        }

        @Reference(
            target = "(model.class.name=com.liferay.training.gradebook.model.Assignme
            unbind = "-"
        )
        protected void setEntryModelPermission(
            ModelResourcePermission<Assignment> modelResourcePermission) {

            _assignmentModelResourcePermission = modelResourcePermission;
        }

        private static ModelResourcePermission<Assignment>
        _assignmentModelResourcePermission;

}
```

We'll put our entity permission checking object into the request attributes of our main view so that it can be used in the JSP files.

Implement Permission Checking in the JSP Files

1. **Open** the class `com.liferay.training.gradebook.web.portlet.action.ViewAssignmentsMVCRenderCommand`

2. **Add** a service reference for the permission checker:

```
@Reference
protected AssignmentPermission _assignmentPermission;
```

3. **Add** the checker into the request attributes in the `render()` method:

```
renderRequest.setAttribute("assignmentPermission", _assignmentPermission);
```

Your final class should now look like this:

```
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 com.liferay.training.gradebook.web.internal.security.permission.resource.Assignmer
```

```
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);

        // Add permission checker.

        renderRequest.setAttribute(
                "assignmentPermission", _assignmentPermission);

        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> assigments =
        _assignmentService.getAssignmentsByKeywords(
                themeDisplay.getScopeGroupId(), keywords, start, end,
```

```
                        comparator);

        // Set request attributes.

        renderRequest.setAttribute("assignments", assigments);
        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 assignmentsManagementToolbarD
                new AssignmentsManagementToolbarDisplayContext(
                        liferayPortletRequest, liferayPortletResponse,
                        _portal.getHttpServletRequest(renderRequest));

        renderRequest.setAttribute(
```

```
                        "assignmentsManagementToolbarDisplayContext",
                        assignmentsManagementToolbarDisplayContext);

        }

        @Reference
        protected AssignmentPermission _assignmentPermission;

        @Reference
        protected AssignmentService _assignmentService;

        @Reference
        private Portal _portal;
}
```

So far, everybody has been able to see the assignment actions menu. Now we'll hide them from unauthorized users.

We'll also add an option to manage entity permissions. For that purpose, we'll use the `<liferay-security>` tag library:

Add Option to Manage Entity Permissions

1. **Declare** the `<liferay-security>` taglib in `src/main/resources/META-INF/resources/init.jsp`:

```
<%@ taglib prefix="liferay-security" uri="http://liferay.com/tld/security" %
```

2. **Open** the file `src/main/resources/META-INF/resources/assignment/entry_actions.jsp`
3. **Wrap** all the actions with permission checks so that only authorized users can access the functions and add a permissions menu option. Replace the contents of the file with the following:

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

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

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

	<%-- View action. --%>

	<c:if test="${assignmentPermission.contains(permissionChecker, assignment.assignm
		<portlet:renderURL var="viewAssignmentURL">
			<portlet:param name="mvcRenderCommandName"
				value="<%=MVCCommandNames.VIEW_ASSIGNMENT %>" />
			<portlet:param name="redirect" value="${currentURL}" />
			<portlet:param name="assignmentId" value="${assignment.assignment
		</portlet:renderURL>

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

	<%-- Edit action. --%>

	<c:if test="${assignmentPermission.contains(permissionChecker, assignment.assignm

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

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

	<%-- Permissions action. --%>
```

```
        <c:if test="${assignmentPermission.contains(permissionChecker, assignment.assignm

                <liferay-security:permissionsURL
                        modelResource="com.liferay.training.gradebook.model.Assignment"
                        modelResourceDescription="${assignment.getTitle(locale)}"
                        resourcePrimKey="${assignment.assignmentId}"
                        var="permissionsURL"
                />

                <liferay-ui:icon message="permissions" url="${permissionsURL}" />
        </c:if>

        <%-- Delete action. --%>

        <c:if test="${assignmentPermission.contains(permissionChecker, assignment.assignm

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

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

The last thing we need to do is to hide the plus button on the management toolbar for adding assignments. Let's add a permission check to the management toolbar backing class.

Implement Permission Checking in the Management Toolbar

1. **Open** the class `com.liferay.training.gradebook.web.display.context.AssignmentsManagementToolbarDisplayContext.java`.

2. **Implement** permission checking in the `getCreationMenu()` method as follows:

```
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(
                                                liferayPortletRespo
                                                "mvcRenderCommandNa
                                                "redirect", current
                                        dropdownItem.setLabel(
                                                LanguageUtil.get(re
                                });
                }
        };
}
```

3. **Resolve** missing imports.

Test the Application

1. **Go to** localhost:8080 in your browser after redeploying the module.
2. **Sign out** of your Liferay DXP instance and test whether you can add, edit, or delete Assignments.
3. **Create** a new user with just the *User role*.
    - Test whether you can add, edit, or delete Assignments.
    - Add the *Add Entry* permission to the role and test again.

Implement Access Control Module Quiz

1. Which of the following is *not* a core concept of access control in Liferay DXP?

   A. Roles

   B. Resources

   C. Actions

   D. Assets

2. How are model resources typically identified and referenced?

   A. By the portlet ID defined in javax.portlet.name

   B. By an abbreviation of the entity's class name

   C. By the entity's fully qualified class name

   D. By the portlet ID defined in portlet.xml

3. By convention, top-level actions are referenced by the package name of the respective service and the resource actions by the fully qualified name of the targeted model entity.

   A. True

   B. False

4. Which of the following scopes can be assigned to a Role in Liferay DXP? (Choose all correct answers).

   A. Team

   B. Organization

   C. Global

   D. Site Group

   E. User Group

5. If you are using Service Builder to create your custom entities and want to implement permissioning support for your application, you must define service permissions in the service module and model permissions in the portlet module, typically the "web" module.

   A. True

   B. False

Answer Key

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