Integrate with Liferay DXP Frameworks

Learning Objectives

- Understand the Asset Framework and the benefits of its integration with applications
- Know the typical process by which the Search Framework is integrated with applications

Tasks to Accomplish

- Integrate with the Asset Framework
- Integrate with Portal Search

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

Integrate with the Asset Framework

Exercise Goals

- Add the required fields to the Assignment entity
- Implement Asset Resource Management in the Local Service
- Implement an AssetRenderer for the Assignments
- Implement an AssetRendererFactory for the Assignments
- Implement the JSP files

The Asset Framework requires a certain set of fields (columns) from all the entities. We are currently missing the status fields:

- **status**: used to determine an entity's status in the workflow
- **statusByUserId**: status audit field
- **statusByUserName**: status audit field
- **statusDate**: status audit field

We'll also add a status finder for listing Assignments by their status.

Add the Required Fields to the Assignment Entity

1. **Open** the `service.xml` in the *gradebook-service* module.

2. **Add** the status fields. The final `service.xml` will look like this:

```
<?xml version="1.0"?>
<!DOCTYPE service-builder PUBLIC "-//Liferay//DTD Service Builder 7.4.0//EN'
<service-builder dependency-injector="ds" package-path="com.liferay.trainin
        <namespace>Gradebook</namespace>
        <!--<entity data-source="sampleDataSource" local-service="true" nam
                remote-service="false" session-factory="sampleSessionFactor
```

```
        tx-manager="sampleTransactionManager uuid="true""> -->
    <entity local-service="true" name="Assignment" remote-service="true"
        <!-- PK fields -->
        <column name="assignmentId" primary="true" type="long"></co
        <!-- Group instance -->
        <column name="groupId" type="long"></column>
        <!-- Audit fields -->
        <column name="companyId" type="long"></column>
        <column name="userId" type="long"></column>
        <column name="userName" type="String"></column>
        <column name="createDate" type="Date"></column>
        <column name="modifiedDate" type="Date"></column>
        <column name="description" type="String" localized="true" />
        <column name="dueDate" type="Date" />
        <column name="status" type="int" />
        <column name="statusByUserId" type="long" />
        <column name="statusByUserName" type="String" />
        <column name="statusDate" type="Date" />
        <!-- Localization Fields -->
        <column name="title" type="String" localized="true"></colum
        <!-- Order -->
        <order by="asc">
            <order-column name="title" />
        </order>
        <!-- Finders -->
        <!-- Find by groupId -->
        <finder name="GroupId" return-type="Collection">
            <finder-column name="groupId"></finder-column>
        </finder>
        <!-- Reference to Group entity service -->
        <reference entity="Group" package-path="com.liferay.portal"
        <!-- Entity services needed for the integration to Asset fra
```

```
                        <reference entity="AssetEntry"
                                package-path="com.liferay.portlet.asset"></reference
                        <reference entity="AssetLink"
                                package-path="com.liferay.portlet.asset"></reference
                        <reference entity="AssetTag"
                                package-path="com.liferay.portlet.asset"></reference
                </entity>
                <!-- Exceptions -->
                <exceptions>
                        <exception>AssignmentValidation</exception>
                </exceptions>
        </service-builder>
```

3. **Run** the `buildService` task to regenerate the service.

   Take a quick look at the re-generated `com.liferay.training.gradebook.model.AssignmentModel` interface in the *gradebook-api* model. After you added the status fields, the `WorkflowedModel` interface is also implemented, enabling the model to support Workflows:

```
public interface AssignmentModel extends BaseModel<Assignment>, GroupedModel,
LocalizedModel, ShardedModel, StagedAuditedModel, WorkflowedModel {
...
        }
```

As with permissions, the Asset resource lifecycle has to be kept in sync with your entity: whenever you create, update, or delete an Assignment entity, you have to take care of the Asset resource:

Implement Asset Resource Management in the Local Service

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

2. **Implement** the new `updateAsset()` method:

```
        private void updateAsset(
                Assignment assignment, ServiceContext serviceContext)
                throws PortalException {
                assetEntryLocalService.updateEntry(
                        serviceContext.getUserId(), serviceContext.getScopeGroupId(
                        assignment.getCreateDate(), assignment.getModifiedDate(),
                        Assignment.class.getName(), assignment.getAssignmentId(),
                        assignment.getUserUuid(), 0, serviceContext.getAssetCategor
                        serviceContext.getAssetTagNames(), true, true,
                        assignment.getCreateDate(), null, null, null,
                        ContentTypes.TEXT_HTML,
                        assignment.getTitle(serviceContext.getLocale()),
                        assignment.getDescription(), null, null, null, 0, 0,
                        serviceContext.getAssetPriority());
        }
```

3. **Add** the call to the `updateAsset()` to the very end of `addAssignment()` before the `return` statement:

```
        // Update asset resources.
        updateAsset(assignment, serviceContext);
        return assignment;
}
```

4. **Implement** the updating Asset resource in the `updateAssignment()` method:

```
                // Update Asset resources.
                updateAsset(assignment, serviceContext);
                return assignment;
        }
```

5. **Implement** deleting the Asset resource at the very end of the `deleteAssignment()` method before the return statement:

```
        // Delete the Asset resource.
        assetEntryLocalService.deleteEntry(
                Assignment.class.getName(), assignment.getAssignmentId());
        return super.deleteAssignment(assignment);
    }
```

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

The service layer is now ready for the Asset Framework. Next, we'll create the AssetRenderer and AssetRendererFactory components in the *gradebook-web* to take care of displaying the assets in a standard way, for example, in the Asset Publisher portlet.

First, we'll add the required dependencies to the *gradebook-web*.

Implement an Asset Renderer Factory for the Assignments

1. **Open** the `build.gradle` of the *gradebook-web* bundle.
2. **Add** dependency for the AssetHelper utility, located in the `com.liferay.asset.api` bundle, and Asset Display Page Api:

```
    compileOnly group: "com.liferay", name: "com.liferay.asset.api"
    compileOnly group: "com.liferay", name: "com.liferay.asset.display.page.api"
```

3. **Run** Gradle refresh to refresh the dependencies.
4. **Create** a class `com.liferay.training.gradebook.web.asset.model.AssignmentAssetRendererFactory` and implement as follows:

```
package com.liferay.training.gradebook.web.asset.model;

import com.liferay.asset.display.page.portlet.AssetDisplayPageFriendlyURLProvider;
```

```
import com.liferay.asset.kernel.model.AssetRenderer;
import com.liferay.asset.kernel.model.AssetRendererFactory;
import com.liferay.asset.kernel.model.BaseAssetRendererFactory;
import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.portlet.LiferayPortletRequest;
import com.liferay.portal.kernel.portlet.LiferayPortletResponse;
import com.liferay.portal.kernel.portlet.LiferayPortletURL;
import com.liferay.portal.kernel.portlet.PortletURLFactory;
import com.liferay.portal.kernel.security.permission.ActionKeys;
import com.liferay.portal.kernel.security.permission.PermissionChecker;
import com.liferay.portal.kernel.security.permission.resource.ModelResourcePermission;
import com.liferay.portal.kernel.security.permission.resource.PortletResourcePermission;
import com.liferay.portal.kernel.util.Portal;
import com.liferay.training.gradebook.constants.GradebookConstants;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.AssignmentLocalService;
import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;
import com.liferay.training.gradebook.web.constants.MVCCommandNames;

import javax.portlet.PortletRequest;
import javax.portlet.PortletURL;
import javax.portlet.WindowState;
import javax.portlet.WindowStateException;
import javax.servlet.ServletContext;

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

/**
 * Asset renderer factory component for assignments.
 *
 * @author liferay
```

```
 */
@Component(
        immediate = true,
        property = "javax.portlet.name=" + GradebookPortletKeys.GRADEBOOK,
        service = AssetRendererFactory.class
)
public class AssignmentAssetRendererFactory
        extends BaseAssetRendererFactory<Assignment> {

        public static final String CLASS_NAME = Assignment.class.getName();

        public static final String TYPE = "assignment";

        public AssignmentAssetRendererFactory() {
                setClassName(CLASS_NAME);
                setLinkable(true);
                setPortletId(GradebookPortletKeys.GRADEBOOK);
                setSearchable(true);
        }

        @Override
        public AssetRenderer<Assignment> getAssetRenderer(long classPK, int type)
                throws PortalException {

                Assignment assignment = _assignmentLocalService.getAssignment(classPK);

                AssignmentAssetRenderer assignmentAssetRenderer =
                        new AssignmentAssetRenderer(
                                        assignment);

                assignmentAssetRenderer.setAssetDisplayPageFriendlyURLProvider(
                        _assetDisplayPageFriendlyURLProvider);
```

```
        assignmentAssetRenderer.setAssetRendererType(type);
        assignmentAssetRenderer.setServletContext(_servletContext);

        return assignmentAssetRenderer;
}

@Override
public String getType() {
        return TYPE;
}

@Override
public PortletURL getURLAdd(
        LiferayPortletRequest liferayPortletRequest,
        LiferayPortletResponse liferayPortletResponse, long classTypeId) {

        PortletURL portletURL = _portal.getControlPanelPortletURL(
                liferayPortletRequest, getGroup(liferayPortletRequest),
                GradebookPortletKeys.GRADEBOOK, 0, 0, PortletRequest.RENDER_PHASE

        portletURL.setParameter("mvcRenderCommandName", MVCCommandNames.EDIT_ASSI

        return portletURL;
}

@Override
public PortletURL getURLView(
        LiferayPortletResponse liferayPortletResponse,
        WindowState windowState) {

        LiferayPortletURL liferayPortletURL =
                liferayPortletResponse.createLiferayPortletURL(
```

```
                        GradebookPortletKeys.GRADEBOOK, PortletRequest.RENDER_PHA

        try {
                liferayPortletURL.setWindowState(windowState);
        }
        catch (WindowStateException wse) {
        }

        return liferayPortletURL;
}

@Override
public boolean hasAddPermission(
                PermissionChecker permissionChecker, long groupId, long classType
        throws Exception {

        return _portletResourcePermission.contains(
                permissionChecker, groupId, ActionKeys.ADD_ENTRY);
}

@Override
public boolean hasPermission(
                PermissionChecker permissionChecker, long classPK, String actionI
        throws Exception {

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

@Reference
private AssetDisplayPageFriendlyURLProvider
        _assetDisplayPageFriendlyURLProvider;
```

```
        @Reference
        private AssignmentLocalService _assignmentLocalService;

        @Reference(
                target = "(model.class.name=com.liferay.training.gradebook.model.Assignme
        )
        private ModelResourcePermission<Assignment>
                _assignmentModelResourcePermission;

        @Reference
        private Portal _portal;

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

        @Reference
        private PortletURLFactory _portletURLFactory;

        @Reference(
                target = "(osgi.web.symbolicname=com.liferay.training.gradebook.web)"
        )
        private ServletContext _servletContext;

}
```

Implement an AssetRenderer for the Assignments

1. **Create** a class `com.liferay.training.gradebook.web.asset.model.AssignmentAssetRenderer` and implement as follows:

```
package com.liferay.training.gradebook.web.asset.model;

import com.liferay.asset.display.page.portlet.AssetDisplayPageFriendlyURLProvider;
import com.liferay.asset.kernel.model.BaseJSPAssetRenderer;
import com.liferay.asset.util.AssetHelper;
import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.exception.SystemException;
import com.liferay.portal.kernel.model.Group;
import com.liferay.portal.kernel.model.LayoutConstants;
import com.liferay.portal.kernel.portlet.LiferayPortletRequest;
import com.liferay.portal.kernel.portlet.LiferayPortletResponse;
import com.liferay.portal.kernel.portlet.PortletURLFactoryUtil;
import com.liferay.portal.kernel.security.permission.ActionKeys;
import com.liferay.portal.kernel.security.permission.PermissionChecker;
import com.liferay.portal.kernel.service.GroupLocalServiceUtil;
import com.liferay.portal.kernel.theme.ThemeDisplay;
import com.liferay.portal.kernel.util.HtmlUtil;
import com.liferay.portal.kernel.util.PortalUtil;
import com.liferay.portal.kernel.util.StringUtil;
import com.liferay.portal.kernel.util.Validator;
import com.liferay.portal.kernel.util.WebKeys;
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.web.constants.GradebookPortletKeys;
import com.liferay.training.gradebook.web.constants.MVCCommandNames;
import com.liferay.training.gradebook.web.internal.security.permission.resource.Assignmer

import java.util.Locale;

import javax.portlet.PortletRequest;
import javax.portlet.PortletResponse;
import javax.portlet.PortletURL;
import javax.portlet.WindowState;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Asset renderer for assignments.
 *
 * @author liferay
 */
public class AssignmentAssetRenderer extends BaseJSPAssetRenderer<Assignment> {

	public AssignmentAssetRenderer(
		Assignment assignment) {

		_assignment = assignment;
	}

	@Override
	public Assignment getAssetObject() {
		return _assignment;
	}

	@Override
	public String getClassName() {
		return Assignment.class.getName();
	}

	@Override
	public long getClassPK() {
		return _assignment.getAssignmentId();
	}

	@Override
```

```
public long getGroupId() {
        return _assignment.getGroupId();
}

@Override
public String getJspPath(HttpServletRequest request, String template) {

        if (template.equals(TEMPLATE_ABSTRACT) ||
                template.equals(TEMPLATE_FULL_CONTENT)) {

                return "/asset/" + template + ".jsp";
        }

        return null;
}

@Override
public int getStatus() {
        return _assignment.getStatus();
}

@Override
public String getSummary(
        PortletRequest portletRequest, PortletResponse portletResponse) {

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

        int abstractLength = AssetHelper.ASSET_ENTRY_ABSTRACT_LENGTH;

        String summary = HtmlUtil.stripHtml(
                StringUtil.shorten(
```

```
                        _assignment.getDescription(),
                        abstractLength));

        return summary;
}

@Override
public String getTitle(Locale locale) {
        return _assignment.getTitle(locale);
}

@Override
public PortletURL getURLEdit(
                LiferayPortletRequest liferayPortletRequest,
                LiferayPortletResponse liferayPortletResponse)
        throws Exception {

        Group group = GroupLocalServiceUtil.fetchGroup(_assignment.getGroupId());

        if (group.isCompany()) {
                ThemeDisplay themeDisplay =
                        (ThemeDisplay)liferayPortletRequest.getAttribute(
                                WebKeys.THEME_DISPLAY);

                group = themeDisplay.getScopeGroup();
        }

        PortletURL portletURL = PortalUtil.getControlPanelPortletURL(
                liferayPortletRequest, group, GradebookPortletKeys.GRADEBOOK, 0,
                PortletRequest.RENDER_PHASE);

        portletURL.setParameter(
```

```
                "mvcRenderCommandName", MVCCommandNames.EDIT_ASSIGNMENT);
        portletURL.setParameter(
                "assignmentId", String.valueOf(_assignment.getAssignmentId()));
        portletURL.setParameter("showback", Boolean.FALSE.toString());

        return portletURL;
}

@Override
public String getURLView(
                LiferayPortletResponse liferayPortletResponse,
                WindowState windowState)
        throws Exception {

        return super.getURLView(liferayPortletResponse, windowState);
}

@Override
public String getURLViewInContext(
                LiferayPortletRequest liferayPortletRequest,
                LiferayPortletResponse liferayPortletResponse,
                String noSuchEntryRedirect)
        throws Exception {

        if (_assetDisplayPageFriendlyURLProvider != null) {
                ThemeDisplay themeDisplay =
                        (ThemeDisplay)liferayPortletRequest.getAttribute(
                                WebKeys.THEME_DISPLAY);

                String friendlyURL =
                        _assetDisplayPageFriendlyURLProvider.getFriendlyURL(
                                getClassName(), getClassPK(), themeDisplay);
```

```
        if (Validator.isNotNull(friendlyURL)) {
                return friendlyURL;
        }
}

try {
        long plid = PortalUtil.getPlidFromPortletId(
                _assignment.getGroupId(), GradebookPortletKeys.GRADEBOOK
        );

        PortletURL portletURL;

        if (plid == LayoutConstants.DEFAULT_PLID) {
                portletURL = liferayPortletResponse.createLiferayPortletU
                        getControlPanelPlid(liferayPortletRequest),
                        GradebookPortletKeys.GRADEBOOK,
                        PortletRequest.RENDER_PHASE);
        }
        else {
                portletURL =
                        PortletURLFactoryUtil.getPortletURLFactory(
                        ).create(
                                liferayPortletRequest, GradebookPortletKe
                                plid, PortletRequest.RENDER_PHASE
                        );
        }

        portletURL.setParameter(
                "mvcRenderCommandName", MVCCommandNames.VIEW_ASSIGNMENT);
        portletURL.setParameter(
                "assignmentId", String.valueOf(_assignment.getAssignmentI
```

```
                String currentUrl = PortalUtil.getCurrentURL(
                        liferayPortletRequest
                );

                portletURL.setParameter("redirect", currentUrl);

                return portletURL.toString();
        }
        catch (PortalException pe) {
        }
        catch (SystemException se) {
        }

        return null;
}

@Override
public long getUserId() {
        return _assignment.getUserId();
}

@Override
public String getUserName() {
        return _assignment.getUserName();
}

@Override
public String getUuid() {
        return _assignment.getUserUuid();
}
```

```
@Override
public boolean hasEditPermission(PermissionChecker permissionChecker)
        throws PortalException {

        return AssignmentPermission.contains(
                permissionChecker, _assignment, ActionKeys.UPDATE);
}

@Override
public boolean hasViewPermission(PermissionChecker permissionChecker)
        throws PortalException {

        return AssignmentPermission.contains(
                permissionChecker, _assignment, ActionKeys.VIEW);
}

@Override
public boolean include(
                HttpServletRequest request, HttpServletResponse response,
                String template)
        throws Exception {

        request.setAttribute("assignment", _assignment);

        return super.include(request, response, template);
}

public void setAssetDisplayPageFriendlyURLProvider(
        AssetDisplayPageFriendlyURLProvider
                assetDisplayPageFriendlyURLProvider) {

        _assetDisplayPageFriendlyURLProvider =
```

```
                        assetDisplayPageFriendlyURLProvider;
        }

        private AssetDisplayPageFriendlyURLProvider
        _assetDisplayPageFriendlyURLProvider;

        private Assignment _assignment;
}
```

> See Enabling Assets for more information about Asset renderers.

As the final step, we'll implement the JSP files for *abstract* and *full content* Asset views. If you take a look at the `getJspPath()` method in the AssetRenderer just created, you'll see how the file path is built:

```
@Override
public String getJspPath(HttpServletRequest request, String template) {

        return "/asset/" + template + ".jsp";
}
```

Implement the JSP files

1. **Add** imports for the AssetRenderer and WebKeys in the `src/main/resources/META-INF/resources/init.jsp`.

   ```
   <%@ page import="com.liferay.asset.kernel.model.AssetRenderer"%>
   <%@ page import="com.liferay.portal.kernel.util.WebKeys"%>
   ```

2. **Create** a folder `src/main/resources/META-INF/resources/asset` in the *gradebook-web* module.
3. **Implement** the two files in the folder:

abstract.jsp

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

<p>
	<%
		AssetRenderer<?> assetRenderer = (AssetRenderer<?>)request.getAttribute(W
	%>

	<%= HtmlUtil.escape(assetRenderer.getSummary(renderRequest, renderResponse)) %>
</p>
```

full_content.jsp

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

<%
	AssetRenderer<?> assetRenderer = (AssetRenderer<?>)request.getAttribute(WebKeys.A

	String viewEntryURL = assetRenderer.getURLView(liferayPortletResponse, WindowStat

	Assignment assignment = (Assignment)request.getAttribute("assignment");
%>

<aui:a cssClass="title-link" href="<%= viewEntryURL %>">
	<h3 class="title"><%= HtmlUtil.escape(assignment.getTitle(locale)) %></h3>
</aui:a>

<div class="autofit-col autofit-col-expand">
	<%= HtmlUtil.escape(assignment.getDescription()) %>
</div>
```

Deploy and Test

1. **Check** that the *gradebook-web* module redeploys successfully and that you are able to access the Gradebook application in your web browser.

Remember that to be able to show Assets in the Asset Publisher portlet, we also have to integrate to the Search Framework. We are going to do that in the next exercise.

Integrate with Portal Search

Exercise Goals

- Declare dependencies
- Implement a Gradebook registrar class to register with the search framework
- Implement an Assignment Model Document Contributor to control which fields are indexed
- Implement an Assignment Model Indexer Writer Contributor to configure reindexing
- Implement a Gradebook Keyword Query Contributor to control which fields are being queried
- Implement a Gradebook Model Summary Contributor to control the Gradebook summaries returned
- Review the service implementation classes for `@Indexable` annotations
- Reindex the search index
- Test the application

Before version 7.1, there used to be a single indexer component for taking care of everything search indexer-related for an entity. The new design provides a more modular and a clean approach for controlling different aspects of search framework integration. You can still use the old approach, however.

All the available contributors are not covered in this exercise. See the Liferay Help Center for more information. Also, take a look at the optional Module 7 exercise "Enable Workflows for Assignments" where we will cover the `PreFilterContributor`.

Declare Dependencies

1. **Open** the `build.gradle` in the *gradebook-service* module.
2. **Add** the new dependencies as follows:

```
compileOnly group: "com.liferay", name: "com.liferay.portal.search.spi"

compileOnly group: "com.liferay", name: "com.liferay.portal.search.api"
```

The registrar class registers the Assignments with the search framework.

Implement a Gradebook Registrar Class

1. **Create** a class `com.liferay.training.gradebook.internal.search.AssignmentSearchRegistrar` in the *gradebook-service* module.
2. **Implement** as follows:

```
package com.liferay.training.gradebook.internal.search;


import com.liferay.portal.kernel.search.Field;

import com.liferay.portal.search.spi.model.index.contributor.ModelIndexerWriterContributo

import com.liferay.portal.search.spi.model.registrar.ModelSearchRegistrarHelper;

import com.liferay.portal.search.spi.model.result.contributor.ModelSummaryContributor;

import com.liferay.training.gradebook.model.Assignment;


import org.osgi.framework.BundleContext;

import org.osgi.framework.ServiceRegistration;

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

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

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

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


@Component(

        immediate = true

)

public class AssignmentSearchRegistrar {
```

```
@Activate
protected void activate(BundleContext bundleContext) {
        _serviceRegistration = modelSearchRegistrarHelper.register(
                Assignment.class, bundleContext,
                modelSearchDefinition -> {
                        modelSearchDefinition.setDefaultSelectedFieldNames(
                                Field.ASSET_TAG_NAMES, Field.COMPANY_ID,
                                Field.ENTRY_CLASS_NAME, Field.ENTRY_CLASS_PK,
                                Field.GROUP_ID, Field.MODIFIED_DATE, Field.SCOPE_
                                Field.UID);

                        modelSearchDefinition.setDefaultSelectedLocalizedFieldNar
                                Field.DESCRIPTION, Field.TITLE);

                        modelSearchDefinition.setModelIndexWriteContributor(
                                modelIndexWriterContributor);

                        modelSearchDefinition.setModelSummaryContributor(
                                modelSummaryContributor);
                });
}

@Deactivate
protected void deactivate() {
        _serviceRegistration.unregister();
}

@Reference(
        target = "(indexer.class.name=com.liferay.training.gradebook.model.Assigr
)
protected ModelIndexerWriterContributor<Assignment>
```

```
                modelIndexWriterContributor;

        @Reference
        protected ModelSearchRegistrarHelper modelSearchRegistrarHelper;

        @Reference(
                target = "(indexer.class.name=com.liferay.training.gradebook.model.Assign
        )
        protected ModelSummaryContributor modelSummaryContributor;

        private ServiceRegistration<?> _serviceRegistration;

}
```

The model document contributor controls which fields are indexed. This class's contribute method is called each time the add and update methods in the entity's service layer are called.

Implement an Assignment Model Document Contributor

1. **Create** a class `com.liferay.training.gradebook.internal.search.spi.model.index.contributor.AssignmentModelDocumentContributor` in the *gradebook-service* module.
2. **Implement** as follows:

```
package com.liferay.training.gradebook.internal.search.spi.model.index.contributor;

import com.liferay.portal.kernel.language.LanguageUtil;
import com.liferay.portal.kernel.search.Document;
import com.liferay.portal.kernel.search.Field;
import com.liferay.portal.kernel.util.HtmlUtil;
import com.liferay.portal.kernel.util.LocaleUtil;
import com.liferay.portal.kernel.util.LocalizationUtil;
import com.liferay.portal.search.spi.model.index.contributor.ModelDocumentContributor;
import com.liferay.training.gradebook.model.Assignment;
```

```
import java.util.Locale;

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

@Component(
        immediate = true,
        property = "indexer.class.name=com.liferay.training.gradebook.model.Assignment",
        service = ModelDocumentContributor.class
)
public class AssignmentModelDocumentContributor
        implements ModelDocumentContributor<Assignment> {

        @Override
        public void contribute(Document document, Assignment assignment) {

                // Strip HTML.

                String description = HtmlUtil.extractText(assignment.getDescription());
                document.addText(Field.DESCRIPTION, description);

                String title = HtmlUtil.extractText(assignment.getTitle());
                document.addText(Field.TITLE, title);

                document.addDate(Field.MODIFIED_DATE, assignment.getModifiedDate());

                // Handle localized fields.

                for (Locale locale : LanguageUtil.getAvailableLocales(
                        assignment.getGroupId())) {

                        String languageId = LocaleUtil.toLanguageId(locale);
```

```
                        document.addText(
                                LocalizationUtil.getLocalizedName(
                                        Field.DESCRIPTION, languageId),
                                description);
                        document.addText(
                                LocalizationUtil.getLocalizedName(Field.TITLE, languageId
                                title);
                }
        }
}
```

The Model Indexer Writer Contributor configures the re-indexing and batch re-indexing behavior for the model entity. This class's method is called when a re-index is triggered from the Search administrative application found in `Control Panel > Configuration > Search`.

Implement an Assignment Model Indexer Writer Contributor

1. **Create** a class `com.liferay.training.gradebook.internal.search.spi.model.index.contributor.AssignmentModelIndexerWriterContributor` in the *gradebook-service* module.
2. **Implement** as follows:

```
package com.liferay.training.gradebook.internal.search.spi.model.index.contributor;

import com.liferay.portal.kernel.search.Document;
import com.liferay.portal.search.batch.BatchIndexingActionable;
import com.liferay.portal.search.batch.DynamicQueryBatchIndexingActionableFactory;
import com.liferay.portal.search.spi.model.index.contributor.ModelIndexerWriterContributo
import com.liferay.portal.search.spi.model.index.contributor.helper.ModelIndexerWriterDoc
import com.liferay.training.gradebook.model.Assignment;
import com.liferay.training.gradebook.service.AssignmentLocalService;

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

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


@Component(
        immediate = true,
        property = "indexer.class.name=com.liferay.training.gradebook.model.Assignment",
        service = ModelIndexerWriterContributor.class
)
public class AssignmentModelIndexerWriterContributor
        implements ModelIndexerWriterContributor<Assignment> {


        @Override
        public void customize(
                BatchIndexingActionable batchIndexingActionable,
                ModelIndexerWriterDocumentHelper modelIndexerWriterDocumentHelper) {


                batchIndexingActionable.setPerformActionMethod(
                        (Assignment assignment) -> {
                                Document document =
                                        modelIndexerWriterDocumentHelper.getDocument(ass


                                batchIndexingActionable.addDocuments(document);
                        });
        }


        @Override
        public BatchIndexingActionable getBatchIndexingActionable() {


                return dynamicQueryBatchIndexingActionableFactory.getBatchIndexingActiona
                        assignmentLocalService.getIndexableActionableDynamicQuery());
        }


        @Override
```

```
        public long getCompanyId(Assignment assignment) {

                return assignment.getCompanyId();
        }

        @Reference
        protected AssignmentLocalService assignmentLocalService;

        @Reference
        protected DynamicQueryBatchIndexingActionableFactory dynamicQueryBatchIndexingAct

}
```

The Keyword Query Contributor contributes model-specific clauses to the ongoing search query.

Implement a Gradebook Keyword Query Contributor

1. **Create** a class `com.liferay.training.gradebook.internal.search.spi.model.query.contributor.AssignmentKeywordQueryContributor` in the *gradebook-service* module.
2. **Implement** as follows:

```
package com.liferay.training.gradebook.internal.search.spi.model.query.contributor;

import com.liferay.portal.kernel.search.BooleanQuery;
import com.liferay.portal.kernel.search.Field;
import com.liferay.portal.kernel.search.SearchContext;
import com.liferay.portal.search.query.QueryHelper;
import com.liferay.portal.search.spi.model.query.contributor.KeywordQueryContributor;
import com.liferay.portal.search.spi.model.query.contributor.helper.KeywordQueryContribut

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

```
@Component(
	immediate = true,
	property = "indexer.class.name=com.liferay.training.gradebook.model.Assignment",
	service = KeywordQueryContributor.class
)
public class AssignmentKeywordQueryContributor
	implements KeywordQueryContributor {

	@Override
	public void contribute(
		String keywords, BooleanQuery booleanQuery,
		KeywordQueryContributorHelper keywordQueryContributorHelper) {

		SearchContext searchContext =
			keywordQueryContributorHelper.getSearchContext();

		queryHelper.addSearchLocalizedTerm(
			booleanQuery, searchContext, Field.DESCRIPTION, false);
		queryHelper.addSearchLocalizedTerm(
			booleanQuery, searchContext, Field.TITLE, false);
	}

	@Reference
	protected QueryHelper queryHelper;

}
```

The Model Summary Contributor constructs the results summary, including specifying which fields to use.

Implement a Gradebook Model Summary Contributor

1. **Create** a class `com.liferay.training.gradebook.internal.search.spi.model.result.contributor.AssignmentModelSummaryContributor` in the *gradebook-service* module.

2. **Implement** as follows:

```
package com.liferay.training.gradebook.internal.search.spi.model.result.con
import com.liferay.petra.string.StringPool;
import com.liferay.portal.kernel.search.Document;
import com.liferay.portal.kernel.search.Field;
import com.liferay.portal.kernel.search.Summary;
import com.liferay.portal.kernel.util.LocaleUtil;
import com.liferay.portal.kernel.util.LocalizationUtil;
import com.liferay.portal.search.spi.model.result.contributor.ModelSummaryC
import java.util.Locale;
import org.osgi.service.component.annotations.Component;
@Component(
        immediate = true,
        property = "indexer.class.name=com.liferay.training.gradebook.model
        service = ModelSummaryContributor.class
)
public class AssignmentModelSummaryContributor
        implements ModelSummaryContributor {
        @Override
        public Summary getSummary(
                Document document, Locale locale, String snippet) {
                String languageId = LocaleUtil.toLanguageId(locale);
                return _createSummary(
                        document,
                        LocalizationUtil.getLocalizedName(Field.DESCRIPTION,
                        LocalizationUtil.getLocalizedName(Field.TITLE, langu
        }
        private Summary _createSummary(
```

```
                        Document document, String descriptionField, String titleFie
                        String prefix = Field.SNIPPET + StringPool.UNDERLINE;
                        Summary summary = new Summary(
                                document.get(prefix + titleField, titleField),
                                document.get(prefix + descriptionField, descriptionF
                        summary.setMaxContentLength(200);
                        return summary;
                }
        }
```

3. **Rebuild** the service.

The final step is to review when and how indexing is triggered. Indexing is triggered by the Local Service methods annotated with the `@Indexable` annotation. If you take a look at the `com.liferay.training.gradebook.service.base.AssignmentLocalServiceBaseImpl` class, you'll see that the methods for adding, deleting, and updating Assignments are all annotated with `@Indexable`:

```
@Indexable(type = IndexableType.REINDEX)
@Override
public Assignment addAssignment(Assignment assignment) {
        assignment.setNew(true);

        return assignmentPersistence.update(assignment);
}

@Indexable(type = IndexableType.DELETE)
@Override
public Assignment deleteAssignment(long assignmentId)
        throws PortalException {
        return assignmentPersistence.remove(assignmentId);
}
```

As long as our customizations and overloads of these methods in the `AssignmentLocalServiceImpl` call the base class, we don't have to add annotations to trigger indexing. If you want your custom `AssignmentLocalServiceImpl` method to trigger indexing, just annotate it with `@Indexable` and remember that an indexable method has to return the updated entity.

If you have created test Assignments, you have to reindex the search index to get the Assignments to appear on the results list.

Reindex the Search Index

1. **Redeploy** the module and go to `localhost:8080` in your browser.
2. **Go to** `Global Menu > Control Panel > Configuration > Search`.
3. **Reindex** all search indexes.

Test the Application

1. **Use** the portal search bar to search Assignments.
2. **Create** a new Assignment and check whether it appears in the search.

Integrate with Liferay DXP Frameworks Module Quiz

1. Which of the following is *not* a central concept in integrating with the Asset Framework?

   A. Assets

   B. Asset Renderer Helpers

   C. Asset Renderers

   D. Asset Renderer Factories

2. When rendering the assets, the Asset Publisher portlet first finds an Asset Renderer for the model type and then asks for an Asset Renderer Factory service.

   A. True

   B. False

3. Which of the following are true of the Liferay Search Framework? (Choose two)

   A. Liferay DXP includes an internal search engine.

   B. Liferay DXP uses an external search engine.

   C. Liferay DXP content items do not need to be converted to search index documents.

   D. The default search engine is Elasticsearch.

4. Which of the following is not a step to integrate with the Asset Framework?

   A. Add the required fields and references to the entity definitions.

   B. Create an asset renderer factory for the model class.

   C. Manage the asset renderer helper.

   D. Implement the JSP files to support the Asset Publisher.

5. The @Indexable Annotation is a Liferay-provided, method-level annotation to be used with Service Builder that automatically updates the index on entity modification events.

   A. True

   B. False

Answer Key

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